On Evaluating the Integration of Reasoning and Action in LLM Agents with Database Question Answering

Anonymous ACL submission

Abstract

This study introduces a new long-form database question answering dataset designed to evaluate how Large Language Models (LLMs) interact with a SQL interpreter. The task necessitates LLMs to strategically generate multiple SQL queries to retrieve sufficient data from a database, to reason with the acquired context, and to synthesize them into a comprehensive analytical narrative. Our findings highlight that this task poses great challenges even for the state-of-the-art `GPT-4` model. We propose and evaluate two interaction strategies, and provide a fine-grained analysis of the individual stages within the interaction. A key discovery is the identification of two primary bottlenecks hindering effective interaction: the capacity for planning and the ability to generate multiple SQL queries. To address the challenge of accurately assessing answer quality, we introduce a multi-agent evaluation framework that simulates the academic peer-review process, enhancing the precision and reliability of our evaluations. This framework allows for a more nuanced understanding of the strengths and limitations of current LLMs in complex retrieval and reasoning tasks.

1 Introduction

Significant advancements in natural language processing have been driven by the development of Large Language Models (LLMs) (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2023), which have become fundamental components of numerous products used by millions, reshaping people's habits on accessing information. Despite their widespread adoption and impact, LLMs face intrinsic limitations due to their design, including limited context window, stochastic nature which makes them less suited for tasks requiring high standards of precision, and extensive computations (Mialon et al., 2023; Ji et al., 2023; Wang et al., 2023a). Many studies have explored ways to mitigate these constraints by augmenting LLMs with modules/tools of complementary features (Nakano et al., 2022; Lewis et al., 2020; Lazaridou et al., 2022; Gao et al., 2023a; Parisi et al., 2022; Schick et al., 2023). In our study, we focus on augmenting LLMs with a symbolic module - a SQL code interpreter - and assess their performance using the long-form database question-answering task that we introduce, illustrated in Figure 1. Such augmentation is inevitable for tasks involving databases, as they often far exceed the size of LLMs' context windows[1], making information retrieval through any means other than SQL inefficient. Additionally, the use of SQL queries brings transparency to the reasoning process of LLM agents, providing a means to validate the accuracy of their generated responses.

LLMs augmented with external modules/tools possess two primary abilities: the capacity to *act*, which involves the use of tools, and the capability to *reason*, which encompasses planning and analyzing the outcomes of actions (Mialon et al., 2023; Madaan et al., 2023; Paul et al., 2023; Yao et al., 2023; Yoran et al., 2023; Shinn et al., 2023). While numerous studies have evaluated these abilities in different contexts, we contend that some of them focus more on evaluating tool selection and tool employment with less focus on evaluating how LLM agents reflect or synthesize the action results (Parisi et al., 2022; Schick et al., 2023; Zhuang et al., 2023; Li et al., 2023). Other research (Shuster et al., 2022; Yao et al., 2023; BehnamGhader et al., 2023) does examine both the action and reasoning capacities of LLM agents, yet the actions' complexity is not as demanding as in studies with a stronger focus on the action aspect. Our goals are twofold: firstly, to introduce a task that places

[1] Enterprise databases can easily store hundreds of millions of records for real-world applications.

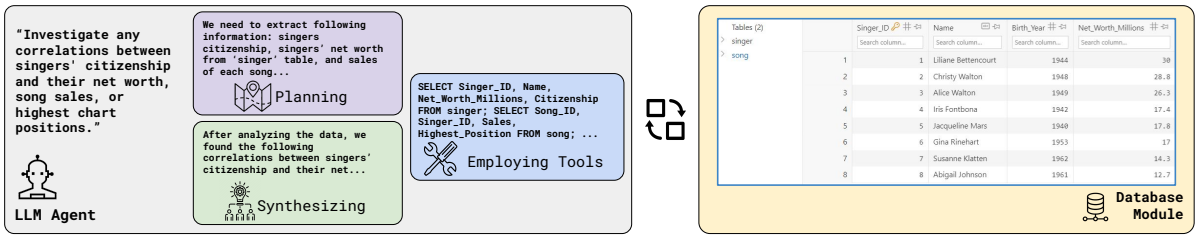


Figure 1: Illustration of our long-form database question answering task. The LLM agent is expected to perform a series of tasks requiring reasoning and actions to interact with the database module.

equal emphasis on the complexities of both action and reasoning, requiring a concerted interaction between them, and secondly, to assess the proficiency of various LLM agents merging these dual aspects into a cohesive performance. Here are our main contributions:

- We introduce a new long-form database question answering task[2], requiring retrieval, reasoning and synthesis of diverse information from database. We develop a systematic approach for collecting questions, databases, and corresponding answers in a way that ensures the answers are definitive and indisputable, lending greater validity to the evaluation process. The task is challenging in retrieval: on average, it requires the formulation of three SQL queries to gather sufficient information to answer the questions.

- We explore the benefits of augmenting LLMs with the SQL code interpreter for our task, by comparing the performance of baseline LLMs given the complete database records but without SQL capabilities against LLM agents that are given database schema and SQL generation capacity.

- In evaluating the performance of agents across all sub-tasks, we identify that planning and tool utilization are the critical challenges in achieving effective coordination. We also delve into the reasons behind their shortcomings. We extend our examination to the generalizability of our results across various LLMs, measuring the disparity in performance between agents using proprietary and open-source LLMs as their foundation.

- Finally, we introduced a multi-agent evaluation framework aimed at enhancing the precision and consistency of the output assessments using GPT-4 evaluators.

[2]Our dataset and codebase will be publicly accessible upon publication.

Property	Value
Evaluation Dataset Size	200
- # Conclusive Questions	98
- # Interpretive Questions	102
Reference Answer Length	
- Conclusive Questions (Avg.)	132
- Interpretive Questions (Avg.)	209
Database size	
- # Tables (Med.)	4
- # Columns (Med.)	4
- # Data Records (Med.)	11

Table 1: Dataset Statistics. **Avg.** stands for average and **Med.** stands for median.

2 Data Collection

In constructing our evaluation dataset, we prioritize a robust set of desiderata. These include the intensive retrieval of diverse information from the database, the application of rigorous reasoning over the information retrieved, and the synthesis of facts and inferences into a coherent and comprehensive long-form answer. Our methodology employs a hybrid annotation framework: we leverage the capabilities of GPT-4 to generate preliminary annotations, then these annotations are selected and refined through manual intervention to ensure quality and relevance. The specifics and quantitative details of our evaluation dataset are presented in Table 1. We detail the collection of questions in Section 2.1 and the acquisition of answers in Section 2.2.

2.1 Question Generation

Our starting point is the databases from the Spider dataset (Yu et al., 2018). We introduce a question generation pipeline designed to generate questions and iteratively refine them, addressing common issues encountered during preliminary experiments with GPT-4 generated queries. This pipeline can be described as **Control-Condense-Confirm**, it begins by exerting **control** over the question generation. We direct GPT-4 to generate questions that

pertain to specific entities or keywords by using the original questions from the Spider dataset as the basis. These questions are instrumental as they concentrate on distinct column sets from various tables, providing a targeted focus that counters the LLM's propensity to formulate overly broad and indistinct questions. Following the initial control, we often find the questions to be exceedingly detailed. To address this, we **condense** the content, removing superfluous information. This pruning process not only ensures the questions remain challenging but also leaves room for the model to demonstrate its inferential capabilities. The final phase is the manual review of questions to **confirm** they are unambiguous and meet all predefined criteria for the task. This step guarantees that the questions are of high quality and align with the specified desiderata of our dataset.

2.2 Answer Annotation

Building upon the question generation strategy outlined in the previous section, the task of annotating answers to questions is generally an effort-intensive task as it requires the formulation of multiple SQL queries. This task is further complicated by the fact that many databases, such as those in the Spider collection, often contain an insufficient number of data records for a comprehensive answer. We propose a method that employs a **Conjecture-Construct-Conclude** strategy to circumvent these issues.

The process begins by prompting `GPT-4` with the question alongside the database schema to **conjecture** an answer. Subsequently, we **construct** database records that corroborate this conjectured answer, formatted as `INSERT` statements. These statements are integrated with the original database's `CREATE` statements, resulting in a bespoke synthetic database aligned with the question. To ensure the integrity of the synthetic database, we execute the merged statements to confirm the absence of errors and manually inspect the data records' alignment with the conjectured answer. As the final step of our method, we task `GPT-4` to **conclude** with a substantiated answer, ensuring that it aligns with the evidences we injected to the synthetic database. This procedure ensures that each question is matched with a definitive answer, backed by verifiable evidence from the database records.

Finally, we examine the question and its corresponding answer. We noticed that a substantial number of questions allow for multiple plausible answers, each subject to interpretation of certain abstract word in the question.[3] To refine the fairness of evaluations against a reference answer, we categorize all questions as either "Interpretive" or "Conclusive". This classification is based on whether the question demands a clear-cut outcome or allows various valid responses. We provide demonstrative examples in Figure 5 of the appendix to illustrate the distinction between these categories. The distribution of questions across these categories is detailed in Table 1.

3 Methods

We aim to evaluate how effective LLMs are at performing a complex task that necessitate working with external modules. We explore five main aspects: (1) the proficiency of LLMs in completing our proposed task through interaction with external modules; (2) the extent of improvement LLMs gain from engaging with external modules; (3) the impact of various interaction strategies on LLM performance and the identification of the most effective one; (4) the challenges that hinder effective interaction; (5) the generalizability of our findings across diverse LLMs and the performance disparities attributed to the usage of different LLMs. We can address the first, second and last aspects by directly evaluating the quality of the final answer generated by LLMs. To delve into the third and fourth aspects, we need to first dissect the "interaction" process within our task into its constituent components.

We propose to decompose the LLMs' expected workflow for our task into three distinct sub-tasks: interaction planning, tool employment, and information synthesis. **Interaction planning** involves the LLM determining its interaction strategy with the external module, considering the question and past interactions. **Tool employment** is the phase where the LLM generates module-specific commands for the actual interaction. **Information synthesis** requires the LLM to review the interaction history and any newly acquired context to compile the key information for the final answer. This framework allows us to refine our second objective into assessing how different compositions of these sub-tasks affect the quality of the final answer, and also to define the most effective composition. The third objective can be addressed by evaluating LLM's execution within each sub-task.

[3] Such as "impact", "success", "notable trends", etc.

While the potential configurations of these sub-tasks are vast, this study will narrow its focus to two primary interaction strategies for feasibility:

- **Sequential**: The LLM agent systematically tackles the sub-tasks in a linear, step-by-step fashion, with predetermined sequence: interaction planning, tool employment, and information synthesis. The agent's focus should be on prioritizing both precision and comprehensiveness throughout each juncture.

- **Iterative**: The LLM agent cyclically alternates between interaction planning and tool employment, similar to the self-ask prompting (Press et al., 2023). The key aspect of interaction planning in this context is to identify the most crucial information to extract from the database given the previous interactions. The strategy calls for the agent to ensure precision in every single interaction and to achieve comprehensiveness by deciding when to terminate the interaction cycle.

Equipped with these strategies, we proceed to empirically explore our central questions.

4 Experiments

4.1 Design

To prove the key areas identified in Section 3, we designed two sets of experiments. The first set evaluates three different types of LLM agents:

- No-Interaction: This LLM is tasked with deriving the final answer with a chain-of-thoughts prompting without engaging with the SQL module, i.e. generating SQL queries. To ensure fairness, we supply the complete database records within the prompt for context.

- Sequential-Interaction: We implement an LLM agent that utilizes the sequential strategy when working with the SQL module. It begins by devising a plan in natural language to identify the needed information and its sources, proceeds to generate SQL queries to retrieve this information, and concludes by integrating the data into the final answer.

- Iterative-Interaction: This strategy employs an LLM agent that iteratively determines the most crucial information to retrieve given the interaction history. The agent articulates this in natural language, crafts the corresponding SQL query, and repeats this process until it elects to stop. The final step involves consolidating the gathered information into a conclusive answer.

The second set of experiment focuses on evaluating the generalizability of our findings across various LLMs, as well as comparing their performance. We tested two proprietary LLMs: `GPT-4` and `GPT-3.5-turbo`, and six open-source LLMs of different sizes and capabilities: `Llama-2-[7, 13]b`, `Code-Llama-[7, 13, 34]b`, and `Mistral-7b`. The `Llama-2` models were tested using their chat versions, while the `Code-llama` and `Mistral` models were evaluated using versions fine-tuned for instruction-following.

4.2 Evaluation

To rigorously assess the performance, we implemented two distinct evaluation methods. Both involve using an LLM for the evaluation process, yet they differ in terms of their reliance on a reference answer. Throughout both evaluation methods, we use `GPT-4` to ensure consistency.

4.2.1 Reference-based Evaluation

In this method, we utilize an LLM to compare the system-generated answer against a reference answer, whose acquirement is detailed in Section 2.2. The evaluation protocol is adapted to suit the nature of the question: for conclusive questions that demand a specific answer, the LLM evaluator provides a straightforward verdict of either *"match"* or *"not match"* and offers a rationale for its decision. For interpretive questions, which permit a spectrum of answers, the LLM assigns a nuanced score ranging from 1 (no match) to 5 (exact match), reflecting the degree of information overlap with the reference. The scoring rubric for this nuanced evaluation is outlined in Figure 7 of the appendix.

4.2.2 Reference-free Evaluation

Assessing LLM performance on individual sub-tasks is essential, yet the multitude of potential answer pathways complicates the annotation process, making reference-based evaluation impractical. To navigate this challenge, we devised a reference-free evaluation using a **multi-agent framework** modeled after the academic peer-review system, as illustrated in Figure 6 in the Appendix. This framework enlists a group of reviewers and meta-reviewers

to evaluate the system outputs. Each reviewer receives the question, database schema, and the LLM agent's output for individual sub-tasks. Their role is to critically assess each output across various dimensions and determine if it is *"perfect"* or *"not perfect"*. Meta-reviewers are then presented with the reviewers' assessments and verdicts. Their task is to discern consensus or discrepancies among the reviewers' opinions, evaluate the validity of their critiques, and render a final decision of *"perfect"* or *"not perfect"*. The ultimate evaluation outcome is derived from the majority ruling among the meta-reviewers. To ensure a diversity of perspectives and avoid uniformity in judgment, we configured each `GPT-4` evaluator with a temperature of 0.7. The specific guidelines used to direct reviewers and meta-reviewers are detailed in Figures 8, 9, 10 in the Appendix. We conducted a manual examination of all the reviews and, based on our findings, adopted a meta-review revision process to enhance the results. The details and rationale behind this approach are elaborated in Section A of the Appendix.

4.3 Results

The experimental outcomes are presented in Figures 2, 3, 4 and Tables 2, 3, 5, addressing the main research questions posed by our study.

LLM Agent Performance on Proposed Task Examining Figures 2, 4, and Tables 2, 5, it becomes evident that even the state-of-the-art `GPT-4`, when utilizing a better interaction strategy, correctly answers only 30% of conclusive questions and achieves an average score of 2.34 on a 1-5 scale. The multi-agent evaluation echoes this, with 36% (IP), 56% (TE), and 67% (IS) of instances deemed perfect for individual step, and only **22%** of instances considered perfect for all steps, indicating considerable room for improvement.

Improvement from SQL Module Interaction Table 3 compares LLM agents' performances with and without SQL module interaction. A significant improvement is observed in LLMs' performance on conclusive questions when augmented with SQL modules. However, this is not the case for interpretive questions. It is important to note that this result applies to instances with small databases where complete records fit within the context window provided to non-interactive LLMs. This suggests that direct interaction with external modules is particularly beneficial for tasks that demand high precision in retrieval, reinforcing our arguments of LLM limitations in Section 1.

The Impact of Interaction Strategies Figure 2 reveals that, generally, a sequential strategy yields better results across most LLMs, with iterative strategies favoring `Llama-2-13b` and `Mistral-7b`. We investigate this trend by analyzing the length of plans, the number of generated SQL queries, and their validity. Notably, when employing iterative strategies, both `Llama-2-13b` and `Mistral-7b` engage minimally with the SQL module, with the `Mistral-7b` agent does not engage with the database at all (empty plan), indicating that it essentially guesses the answers. We set the performance of `Mistral-7b` with iterative strategy as a reference point for guess-based answers, marked by vertical lines in the figures, and make a note that performances close to this baseline likely result from guesswork. Additionally, we can also notice that agents using iterative strategies tend to plan less and interact minimally with SQL modules, contributing to their underperformance compared to sequential strategies.

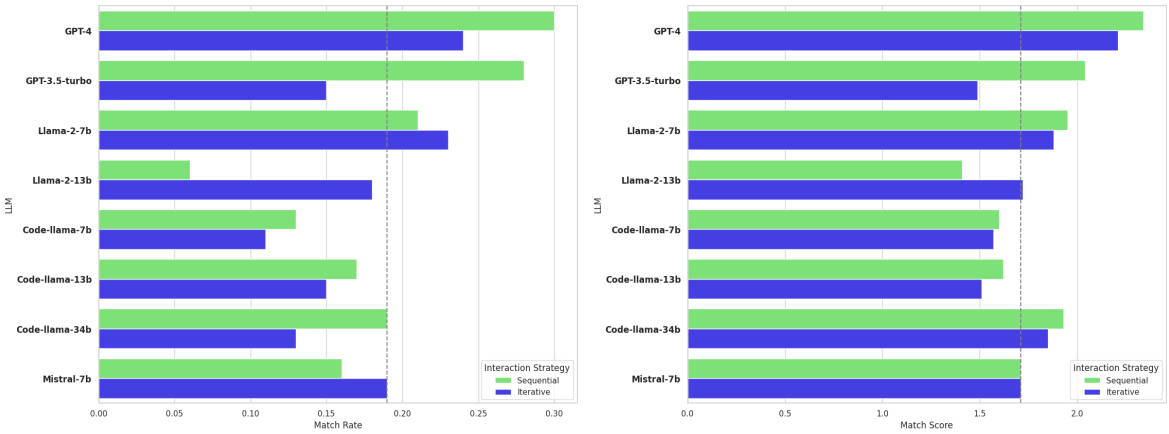
Barriers to Effective Interaction Figures 4 and Table 5 highlight that planning and tool employment (i.e. SQL generation) are the main hurdles preventing agents from performing well on the proposed tasks. Conversely, agents generally excel at synthesizing retrieved information to produce an accurate and comprehensive answer. This indicates that eliciting better interaction planning via more effective prompting and enhancing LLMs' ability to generate multiple SQL queries in parallel from extended text descriptions are promising areas for future research.

Generalizability Across Different LLMs The conclusions regarding interaction strategy appear consistent across various LLMs. However, our hypothesis of interaction barriers primarily holds for the more capable proprietary LLMs like `GPT-4` and `GPT-3.5-turbo`, and to a lesser extent, `Llama-2` and `Code-Llama` models. The remaining LLMs did not engage in any meaningful interaction, thus no discernible patterns were noted.

Interaction Depth and Answer Quality Figure 3 suggests a weak correlation between the number of valid interactions (i.e., agent-generated SQL queries that yield non-empty results) and performance, hinting that more successful retrieval aids

LLM	Interaction Mode	Match Score (C/I)	Plan Length (C/I)	# Generated SQLs (C/I)	# Valid SQLs (C/I)	Answer Length (C/I)
GPT-4	Sequential	0.30 / 2.34	437 / 474	2.96 / 3.28	2.72 / 2.94	197 / 221
	Iterative	0.24 / 2.21	83 / 101	0.99 / 1.18	0.79 / 0.80	157 / 199
GPT-3.5-turbo	Sequential	0.28 / 2.04	297 / 321	2.23 / 2.63	1.84 / 2.30	202 / 193
	Iterative	0.15 / 1.49	223 / 252	1.06 / 1.10	0.89 / 0.81	94 / 90
Llama-2-7b	Sequential	0.21 / 1.95	364 / 347	2.88 / 2.58	1.29 / 1.04	285 / 283
	Iterative	0.23 / 1.88	63 /92	0.65 /1.03	0.13 / 0.14	252 / 292
Llama-2-13b	Sequential	0.06 / 1.41	398 / 401	3.05 / 1.83	1.32 / 0.86	336 / 339
	Iterative	0.18 / 1.72	16 / 16	0.20 / 0.28	0.01 / 0.01	310 / 359
Code-llama-7b	Sequential	0.13 / 1.60	368 / 390	3.25 / 3.67	1.59 / 1.89	306 / 325
	Iterative	0.11 / 1.57	0 / 0	0 / 0	0 / 0	234 / 240
Code-llama-13b	Sequential	0.17 / 1.62	389 / 396	4.34 / 5.36	1.95 / 2.78	292 / 314
	Iterative	0.15 / 1.51	115 / 110	0.66 / 1.59	0.26 / 0.70	243 / 245
Code-llama-34b	Sequential	0.19 / 1.93	359 / 377	2.79 / 3.31	1.39 / 1.72	314 / 352
	Iterative	0.13 / 1.85	39 / 47	0.4 / 0.33	0.28 / 0.18	248 / 294
Mistral-7b	Sequential	0.16 / 1.71	384 / 379	1.01 / 0.91	0.57 / 0.37	231 / 282
	Iterative	0.19 / 1.71	0 / 0	0 / 0	0 / 0	207 / 267

Table 2: Reference-based evaluation results and other measurements of the interaction process. C stands for **conclusive** and I stands for **interpretive**. Valid SQL indicate SQL queries that are generated by the LLM agent that have non-empty execution results.



(a) Match Rate for Instances with Conclusive Questions

(b) Match Score for Instances with Interpretive Questions

Figure 2: Reference-based evaluation results across various interaction strategies and LLMs, with a vertical line representing the performance achieved by a non-interactive LLM agent lacking database context, serving as the baseline for guessing.

in generating more precise and comprehensive answers.

Diversity and Consensus in Multi-agent Evaluation Aggregating multiple diverse evaluations and meta-evaluations from LLMs appears to decrease result variance, as evidenced by the increased consensus among meta-reviewer LLMs compared to reviewer LLMs shown in Table 5. The scores from meta-reviewers are consistently lower than those from reviewers, indicating that the meta-review process critically considers the issues highlighted by reviewers. This multi-tiered review mechanism ensures that our evaluation framework effectively balances both precision and recall.

4.4 Human Evaluation

We conducted human evaluations in both reference-based and reference-free setting, aiming to compare the evaluation outcomes of LLM evaluators with human judgment. As detailed in Table 4, we present the average results from three internal human evaluators and the proportion of instances where human assessment aligned with LLM evaluations regarding the output of our most powerful

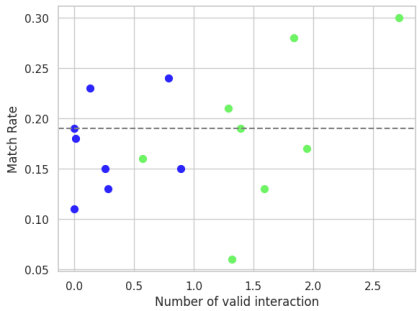
LLM	Interaction Mode	Conclusive Questions Match Rate	Interpretive Questions Match Score
GPT-4	No	0.19	2.37
	Sequential	0.27	2.40
GPT-3.5-turbo	No	0.24	2.12
	Sequential	0.30	2.13

Table 3: LLM agents with vs. without interaction with SQL modules. Entire database records are provided in the context for LLMs that disable interaction. We sampled 161 out of 200 questions to report their reference-based evaluation results because some databases are too large to fit into LLM's context window.

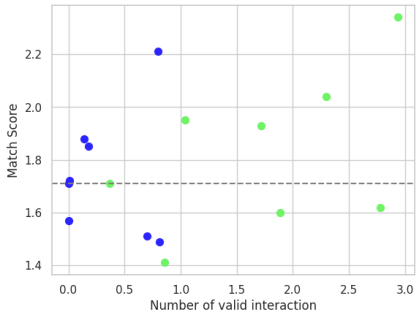
baseline - GPT-4 agent employing a sequential strategy. In both scenarios, human evaluators received identical materials (outputs of GPT-4 agent and scoring guidelines) as provided to the LLM evaluators. In the reference-based setting, we found that human evaluations concurred with LLM assessments in over 80% of cases. A similar level of agreement was observed in the reference-free settings across all IP, TE and IS steps, reinforcing our assertion that significant improvement potential exists in our task. Notably, the most advanced GPT-4 agent primarily faced challenges in planning interactions for interpretive questions, with only 26% of its attempts deemed accurate, whereas planning for conclusive questions demonstrated higher success rate.

4.5 Error Analysis

In our analysis of the GPT-4 agent's performance across the IP, TE and IS steps, we identified several recurrent error types, supplemented by specific examples in Figures 11-15 of the Appendix for clarity. During the planning stage, we observed that the agent often: (1) misinterpreted column contents (Figure 11); (2) made incorrect assumptions about column contents (Figure 12); (3) overlooked key columns essential for analysis to answer the question (Figure 13); (4) exhibited oversight or sampling bias in data retrieval for subsequent analysis (Figure 14). In the tool employment step, the agents often failed to execute statistical analyses, such as computing the correlation coefficient (Figure 15). While Python might be more appropriate for such task, its exploration is beyond the scope of this study. Regarding the information synthesis step, it is important to note that many negative evaluations in this stage stemmed from failures in the previous step. When SQL execution results were lacking, leaving no information for synthesis, the agent struggled to respond accurately, leading to predominantly negative reviews of this phase.



(a) Match Rate vs. Valid Interaction for Instances with Conclusive Questions



(b) Match Score vs. Valid Interaction for Instances with Interpretive Questions

Figure 3: Correlation between answer quality and number of valid interaction (SQL queries that returned non-empty results)

Question type	Reference-based	Reference-free		
	Match / %Agree	IP / %Agree	TE / %Agree	IS / %Agree
Conclusive	0.25 / 0.82	0.60 / 0.83	0.58 / 0.96	0.7 / 0.75
Interpretive	2.2 / 0.83[4]	0.26 / 0.88	0.58 / 0.88	0.76 / 0.84
All	N/A	0.43 / 0.85	0.58 / 0.92	0.73 / 0.80

Table 4: Human evaluation results for reference-based and reference-free settings based on outputs of our strongest baseline - GPT-4 agent with sequential interaction strategy. % Agree represents the proportion of instances in which human and LLM evaluations concur.

5 Related Work

5.1 Augmented Language Models

The use of external tools to augment language model outputs and mitigate model hallucination has been previously studied in other domains and tasks (Mialon et al., 2023). Models augmented with tools like internet search (Lazaridou et al.,

[4]When calculating the agreement for match scores of interpretive questions, we reclassify the original scores (scale of 1 to 5) into three levels: scores of 1 to 3 were assigned as low, a score of 4 as medium, and a score of 5 as high.

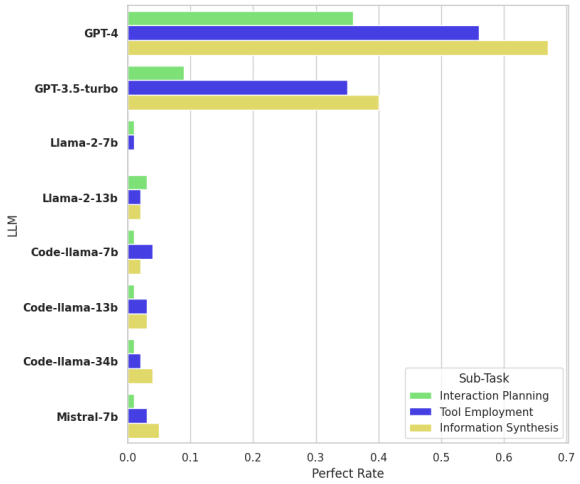


Figure 4: Reference-free multi-agent evaluation results for different sub-tasks and LLMs, all employing sequential interaction strategy.

2022), Python interpreters (Gao et al., 2023b), math equation-generating models (Imani et al., 2023), and question-answering models (Guu et al., 2020) have empirically shown improvements in accuracy in comparison to their counterpart baseline models. Other models like TALM (Parisi et al., 2022) and Toolformer (Schick et al., 2023) for question answering and ToolWriter (Gemmell and Dalton, 2023) for tabular question answering have built on top of these to limit reliance on humans to select tools for question answering models by fine-tuning the models to learn how and when to use tools. Our work differs from these previous works in that we augment language models to use tools in the data-to-text generation domain specifically where the model is expected to not only query from a database with the use of external tools, but also aggregate these results and interpret the data to produce a paragraph-length response to a not necessarily close-ended question.

5.2 Reasoning and Action

Our work draws on elements of frameworks that either prompt the model to repeatedly reason, act upon the reasoning, and update the action plan until the answer is found (Yao et al., 2023) or plan out the different components needed to answer questions before retrieving and generating the answer (Su et al., 2021). Other frameworks used additional language models as the planner to aggregate information retrieved by a diverse inventory of tools (Lu et al., 2023). Our work builds upon some of these frameworks and investigates these in context of the task of long-form data-to-text generation.

5.3 Text-to-SQL

The field of Text-to-SQL has been extensively studied as the standard method for database question answering, with significant contributions from a range of studies (Berant et al., 2013; Zhong et al., 2017; Yu et al., 2018; Yin and Neubig, 2018; Yu et al., 2019; Wang et al., 2020; Yin et al., 2020; Scholak et al., 2021; Ren et al., 2021; Xie et al., 2022; Cheng et al., 2023; Nan et al., 2023, *inter alia*). Despite this, our study innovates by introducing a question-answering task over databases that demands the generation of multiple SQL queries to formulate a single answer.

5.4 Text Generation Evaluation

Development in automatic evaluation metrics have emerged, utilizing LLMs to evaluate the quality of generated texts (Fu et al., 2023; Liu et al., 2023). These methods have also been adapted for evaluating text pertaining to tabular data (Rebuffel et al., 2021) and hallucination detection (Manakul et al., 2023). (Wang et al., 2023b) introduced self-consistency sampling, which has been shown to improve the reasoning performance of the system. This approach involves generating a set of diverse answers and selecting the most common one through majority vote. In our study, we propose a reference-free multi-agent evaluation framework that synthesizes these ideas.

6 Conclusion

In conclusion, our investigation reveals the current limitations of LLMs in complex retrieval and reasoning tasks. Augmentation with a SQL module proved beneficial, particularly for conclusive questions, and pointed to the necessity of strategic interaction planning and proficient tool employment. Our findings stress the need for improvement in these areas to enhance LLM effectiveness. Despite the challenge of varying performance across different models, our multi-agent evaluation framework provides a scalable and rigorous method for assessing agent capabilities. We hope that our proposed task and findings will encourage further investigations in LLMs' capabilities of interacting with external modules, inching towards LLMs capable of handling complex tasks with enhanced precision.

Limitations

This study acknowledges several constraints that much be considered when interpreting the results. First, our evaluation dataset is small in scale, a limitation primarily due to budget constraints. We plan to expand our dataset to enhance the statistical significance of our findings. Second, while this study concentrated on tasks that require intensive action and reasoning capabilities, there is room to explore how LLM agents would perform with external modules on similar tasks with less stringent requirements. Third, this study's investigation is limited to specific modules, leaving the examination of a broader spectrum of modules unaddressed. Expanding our research to include a more diverse set of modules is a direction we plan to explore in our future work. Lastly, it is important to acknowledge a potential bias in our evaluation methodology stemming from the exclusive use of `GPT-4` for generating reference answers as well as for evaluating system-generated responses. This reliance could skew the evaluation in favor of `GPT-4` agent's answers.

References

Parishad BehnamGhader, Santiago Miret, and Siva Reddy. 2023. Can retriever-augmented language models reason? the blame game between the retriever and the language model.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Pal: Program-aided language models.

Carlos Gemmell and Jeffrey Dalton. 2023. Generate, transform, answer: Question specific tool synthesis for tabular data.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback.

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

OpenAI. 2023. Gpt-4 technical report.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Clément Rebuffel, Thomas Scialom, Laure Soulier, Benjamin Piwowarski, Sylvain Lamprier, Jacopo Staiano, Geoffrey Scoutheeten, and Patrick Gallinari. 2021. Data-questeval: A referenceless metric for data-to-text semantic evaluation.

Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8959–8970. PMLR.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. 2022. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 895–909, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, Yidong Wang, Linyi Yang, Jindong Wang, Xing Xie, Zheng Zhang, and Yue Zhang. 2023a. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

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

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools.

Appendix

A Reference-free Evaluation Meta-Review Revision

Upon completing our evaluations and conducting an error analysis based on reviews from our multi-agent framework, we observed that a substantial portion of the negative reviews pertained to the inconsistency between the execution results of SQLs generated by the agent and the database description in the prompt, which included three sample rows to demonstrate data types. It should be noted that these records were intended only for demonstration and do not encompass the entire database. This aspect was not sufficiently emphasized in our prompt, leading `GPT-4` reviewers and meta-reviewers to incorrectly perceive any additional data from SQL execution as erroneous. Furthermore, we identified that reviewers placed a high emphasis on the data type consistency of columns used for joining two tables, even though many join keys in our databases are of compatible types, capable of implicit conversion by the SQL compiler. We believe these two types of criticisms to be overly stringent for evaluating the agent's capability in our task. Consequently, we have instructed `GPT-4` to re-evaluate, considering the meta-reviews of each stage and specifically disregarding the two types of errors mentioned. The original and revised scores, presented in Table 5, demonstrate the considerable impact of these errors on the evaluation of tool employment.

- **Conclusive questions:**
  1. Do dual-enrolled students tend to perform better or worse than their peers in the same degree programs?
  2. Analyze the relationship between teachers' experience and their performance based on the grades received in the courses they have taught.
  3. Investigate any correlations between poker players' performance and factors such as nationality, age, and height.
- **Interpretive questions:**
  1. Compare the success metrics between French and non-French singers.
  2. Analyze the impact of record companies on the success of orchestras based on their performance ratings and attendance.
  3. Analyze the performance of the TV series by language and country, and identify any notable patterns or trends.

Figure 5: Examples of Conclusive and Interpretive Questions

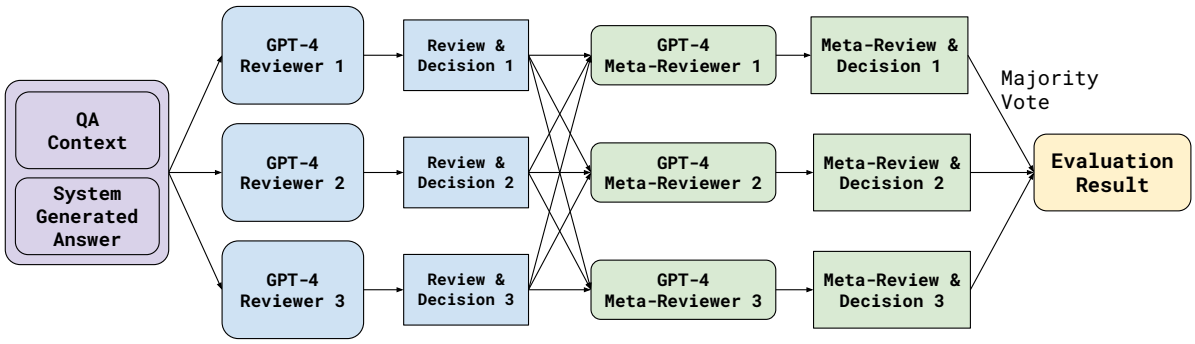


Figure 6: Illustration of our multi-agent evaluation framework. It consists of two tiers of evaluation process.

LLM	Sub-Task	Reviewer		Meta-Reviewer		Revised
		Perf. Rate	Agree.	Perf. Rate	Agree.	Perf. Rate
GPT-4	IP	0.48	0.54	0.28	0.88	0.36
	TE	0.41	0.69	0.28	0.93	0.56
	IS	0.61	0.72	0.53	0.93	0.67
GPT-3.5-turbo	IP	0.14	0.80	0.09	0.96	0.09
	TE	0.18	0.82	0.13	1.00	0.35
	IS	0.26	0.79	0.22	0.95	0.40
Llama-2-7b	IP	0.02	0.97	0.01	0.99	0.01
	TE	0.01	0.99	0.01	1.00	0.01
	IS	0.01	0.99	0.00	1.00	0.00
Llama-2-13b	IP	0.03	0.96	0.02	0.98	0.03
	TE	0.01	0.98	0.01	0.99	0.02
	IS	0.01	0.99	0.01	0.99	0.02
Code-llama-7b	IP	0.02	0.93	0.01	0.99	0.01
	TE	0.01	0.98	0.01	0.99	0.04
	IS	0.02	0.98	0.01	1.00	0.02
Code-llama-13b	IP	0.01	0.96	0.01	1.00	0.01
	TE	0.02	0.98	0.02	1.00	0.03
	IS	0.02	0.97	0.02	1.00	0.03
Code-llama-34b	IP	0.04	0.90	0.01	0.98	0.01
	TE	0.02	0.97	0.01	0.99	0.02
	IS	0.04	0.96	0.04	0.99	0.04
Mistral-7b	IP	0.03	0.96	0.01	0.99	0.01
	TE	0.01	0.98	0.00	1.00	0.03
	IS	0.05	0.94	0.03	0.98	0.05

Table 5: Reference-free multi-agent evaluation - fine-grained results for different LLMs adopting **Sequential** interaction strategy. **IP** stands for Interaction Planning, **TE** stands for Tool Employment, and **IS** stands for Information Synthesis. **Perf. Rate** stands for percentage of instances that (meta-)reviewers considers perfect, and **Agree.** stands for agreement, and it is calculated with the percentage of instances that (meta-)reviewers reach in unanimous agreement. For additional context regarding the revised results, please see Section A of the Appendix.

Given the following inputs:
Question: {question}
Reference (Gold) Answer: {gold_answer}
System Generated Answer: {answer}
Evaluation Process:
Read the gold answer carefully to understand the precise information it conveys.
Examine the system-generated answer to identify the information presented.
Check for the presence of critical information (such as conclusions) from the gold answer in the system-generated answer.
Evaluation Criteria:
The system-generated answer is considered a "Match" if it contains all the critical information from the gold answer. The presence of additional non-contradictory information in the system-generated answer is acceptable, provided that all the information from the gold answer is included.
Output Format:
If the system-generated answer includes all the critical information from the gold answer, the output should be: "Conclusion: Match"
If any critical information from the gold answer is missing or misrepresented in the system-generated answer, the output should be: "Conclusion: Not Match"
Conclusion:

(a) Scoring Metrics for Conclusive Questions

Given the following inputs:
Question: {question}
Reference (Gold) Answer: {gold_answer}
System-Generated Answer: {answer}
Evaluation Process:
Familiarize yourself with the gold answer to understand the full scope of information it contains.
Analyze the system-generated answer to identify the information that has been captured.
Compare the two answers to determine how much of the gold answer's information is reflected in the system-generated answer.
Scoring Metrics:
Score 1: The system-generated answer lacks almost all the key points that the comprehensive gold answer provides.
Score 2: The system-generated answer includes some key points from the gold answer but misses others, and it may include additional details not found in the gold answer.
Score 3: The system-generated answer captures most of the key information from the gold answer, but there are noticeable omissions or additions.
Score 4: The system-generated answer encompasses all key points from the gold answer and also introduces more information not covered in the gold answer.
Score 5: The system-generated answer perfectly mirrors the gold answer, containing all the information with no omissions or additions.
Output Format:
Provide a score between 1 to 5 based on the evaluation. The output should be: "Score: [1/2/3/4/5]"
Score:

(b) Scoring Metrics for Interpretive Questions

Figure 7: Prompts used for evaluating system generated answers for conclusive and interpretive questions

Problem Context:
A planning agent has been tasked to devise a solution to a user question related to a database. Given the question and the database's description, the agent proposes a plan detailing the type of information it would retrieve from the database to answer the question effectively.
Your Task:
You are to evaluate the plan's relevance and comprehensiveness. Assess whether the plan can indeed retrieve the necessary information to address the user's question.
Inputs:
User Question:
{question}
Database Description:
{database_text}
Agent's Proposed Plan:
{plan}
Evaluation Criteria:
Relevance: Does the plan target relevant pieces of information from the database that directly pertain to the user's question?
Comprehensiveness: Is the plan exhaustive, ensuring all necessary pieces of information are retrieved to fully answer the user's question?
Plan Definitions:
Perfect Plan: A plan that is both relevant and comprehensive, ensuring that the user's question can be answered completely without missing any essential data points.
Imperfect Plan: A plan that misses out on some relevant information, or includes unnecessary steps, thus not providing a complete or accurate solution to the user's question.
Response Format:
Rationale: Begin with a detailed explanation of your evaluation. Discuss the strengths or weaknesses of the plan based on the relevance and comprehensiveness criteria.
Final Decision: After providing the rationale, conclude with one of the following decisions:
- Perfect: If you believe the plan meets both the relevance and comprehensiveness criteria effectively.
- Imperfect: If you find the plan lacking in any aspect, be it relevance or comprehensiveness.

(a) Review Criteria for Interaction Planning

Problem Context:
As the "editor-in-chief", you are tasked with evaluating the reviews provided by multiple reviewers on a planning agent's proposed plan to answer a database-related user question. Each review contains a detailed rationale and a final decision.
Your Task:
Your goal is to compare and assess the rationales provided by the reviewers, and then make a final, conclusive decision about the planning agent's proposal. This decision should be based on a comprehensive understanding of the reviewers' perspectives and the evidence they present.
Inputs:
User Question:
{question}
Database Description:
{database_text}
Agent's Proposed Plan:
{plan}
Reviewers' Rationales and Decisions:
{IP_reviews}
Evaluation Criteria:
Review Consistency: Are the reviewers' rationales and decisions consistent with each other?
Evidence Quality: Is the evidence provided in the rationales substantial and convincing enough to make a definitive conclusion?
Final Decision Basis: Does the aggregated perspective of the reviewers lead to a clear final decision?
Response Format:
Rationale: Begin with a detailed explanation comparing the rationales provided by the reviewers. Highlight consistencies or discrepancies among them and discuss how these influenced your final decision.
Final Decision: After providing the rationale, conclude with one of the following decisions:
- Perfect: If the aggregated insights from reviewers suggest that the planning agent's proposal is both relevant and comprehensive.
- Imperfect: If the combined reviews indicate that the planning agent's proposal is lacking in either relevance or comprehensiveness.

(b) Meta-Review Criteria for Interaction Planning

Figure 8: Prompts used for reviewing and meta-reviewing interaction planning

Problem Context:
An agent is given a question, a database for retrieving relevant context, and a plan of how to perform the retrieval. It has been tasked to translate the plan into accurate and executable SQL queries. These queries should correspond to the given plan and effectively retrieve the relevant information from the database to address the user's question, adhering to the database structure provided.
Your Task:
You are to evaluate the correctness and alignment of the SQL queries generated by the agent based on the plan provided. Also, review the execution results to determine if they fulfill the user's requirements as stipulated in the plan.
Inputs:
User Question: {question}
Database Description: {database_text}
Search Plan: {plan}
Agent's Proposed SQL Queries and Execution Results: {sql_results}
Evaluation Criteria:
Correctness: Are the SQL queries syntactically and semantically correct, and do they retrieve the expected data from the database?
Alignment: Do the SQL queries align with the steps outlined in the initial plan?
Execution Results: Does the outcome of the SQL queries correspond to the desired results based on the user's question and the initial plan?
Query Definitions:
Perfect Queries: All SQL queries are correct, aligned, and ensure that the user's question is addressed in accordance with the initial plan.
Imperfect Queries: There is at least one SQL query that has errors, misalignments, or does not produce the expected results as outlined in the initial plan.
Response Format:
Rationale: Begin with a detailed explanation of your evaluation. Address the SQL queries' correctness, their alignment with the initial plan, and the resulting output's relevance to the user's query.
Final Decision: After providing the rationale, conclude with one of the following decisions:
- Perfect: If all SQL queries are correct, aligned with the plan, and the results answer the user's question as expected.
- Imperfect: If you find any discrepancies in correctness, alignment, or the execution results of the proposed SQL queries.

(a) Review Criteria for Tool Employment

Problem Context:
As the "editor-in-chief", you are presented with multiple reviews evaluating an agent's capability to generate SQL queries from a given plan to answer a user question using a specified database. Each review contains an in-depth rationale and a final decision regarding the correctness, alignment, and execution results of the SQL queries.
Your Task:
Your goal is to compare and assess the rationales provided by the reviewers, weighing their evidence and perspectives, and then make a final, conclusive decision regarding the agent's SQL queries based on the aggregated reviews.
Inputs:
User Question: {question}
Database Description: {database_text}
Search Plan: {plan}
Agent's Proposed SQL Queries and Execution Results: {sql_results}
Reviewers' Rationales and Decisions: {TE_reviews}
Evaluation Criteria:
Review Consistency: Do the reviewers agree in their evaluations, or are there conflicting perspectives?
Evidence Quality: Are the rationales provided by reviewers substantial and convincing?
Final Decision Basis: Based on the aggregated insights of the reviewers, is there a clear and justifiable final decision?
Response Format:
Rationale: Begin with a detailed comparison of the rationales provided by the reviewers. Address any consistencies or discrepancies in their evaluations, emphasizing how these observations influenced your final decision.
Final Decision: After analyzing the rationales, conclude with one of the following decisions:
- Perfect: If the collective insights suggest that all the agent's SQL queries are accurate, aligned, and answer the user's question as stipulated.
- Imperfect: If the combined reviews reveal issues in correctness, alignment, or the execution results of the agent's SQL queries.

(b) Meta-Review Criteria for Tool Employment

Figure 9: Prompts used for reviewing and meta-reviewing tool employment

Problem Context:
An agent is presented with a user's question, a plan to extract more context for answering the question, and a search history containing SQL queries used to retrieve this context from the database. The agent's task is to synthesize all the given information to construct a coherent answer to the question.
Your Task:
You are to evaluate the synthesis produced by the agent. Assess whether the agent's response accurately interprets the SQL queries and their execution results. Furthermore, determine if the synthesized answer addresses the user's question both correctly and comprehensively.
Inputs:
User Question:
{question}
Database Description:
{database_text}
Search Plan:
{plan}
SQL Queries and Execution Results:
{sql_results}
Agent's Synthesized Answer:
{answer}
Evaluation Criteria:
Interpretation Accuracy: Does the agent's answer demonstrate a correct understanding of the SQL queries and their execution results?
Answer Correctness: Is the agent's synthesized answer accurate in terms of the given information?
Comprehensiveness: Does the agent's answer cover all aspects of the user's question based on the context retrieved?
Answer Definitions:
Perfect Answer: An answer that accurately interprets the SQL queries and results, and addresses the user's question both correctly and comprehensively.
Imperfect Answer: An answer that either misinterprets the SQL information, or does not completely and accurately address the user's question.
Response Format:
Rationale: Begin with a detailed explanation of your evaluation. Discuss the strengths or weaknesses of the agent's synthesized answer based on the criteria of interpretation accuracy, correctness, and comprehensiveness.
Final Decision: After providing the rationale, conclude with one of the following decisions:
- Perfect: If you believe the agent's answer meets all evaluation criteria effectively.
- Imperfect: If you identify any shortcomings in interpretation accuracy, correctness, or comprehensiveness of the answer.

(a) Review Criteria for Information Synthesis

Problem Context:
As the "editor-in-chief", you are tasked with evaluating multiple reviews that assess an agent's synthesis of an answer based on a user's question, a search plan, and the results of executed SQL queries. Each review contains a detailed rationale and a final decision on the agent's capability to coherently integrate the information and answer the user's question.
Your Task:
Your role is to compare and evaluate the rationales provided by the reviewers, integrating their insights and perspectives. Based on this aggregated understanding, make a final, conclusive decision about the agent's synthesized answer.
Inputs:
User Question:
{question}
Database Description:
{database_text}
Search Plan:
{plan}
SQL Queries and Execution Results:
{sql_results}
Agent's Synthesized Answer:
{answer}
Reviewers' Rationales and Decisions:
{IS_reviews}
Evaluation Criteria:
Review Consistency: Are there shared perspectives among the reviewers, or do they have conflicting views?
Evidence Quality: Do the reviewers present substantial and compelling evidence in their rationales?
Final Decision Basis: Does the collective insight of the reviewers lead to a clear, definitive conclusion about the agent's answer?
Response Format:
Rationale: Start with a comprehensive comparison of the rationales given by the reviewers. Address any commonalities or differences in their evaluations and describe how these factors influenced your final decision.
Final Decision: After dissecting the reviewers' insights, decide on one of the following: - Perfect: If the collective evaluations suggest that the agent's synthesized answer meets all the required criteria.
- Imperfect: If the integrated reviews indicate issues in the agent's interpretation, correctness, or comprehensiveness.

(b) Meta-Review Criteria for Information Synthesis

Figure 10: Prompts used for reviewing and meta-reviewing information synthesis

Database:
Title: "employee"
Header: ['Employee_ID', 'Name', 'Age', 'City']
Title: 'shop'
Header: ['Shop_ID', 'Name', 'Location', 'District', 'Number_products', 'Manager_name']
Title: 'hiring'
Header: ['Shop_ID', 'Employee_ID', 'Start_from', 'Is_full_time']
Title: 'evaluation'
Header: ['Employee_ID', 'Year_awarded', 'Bonus']

Question:
Investigate the relationship between the number of employees and the number of shops in each city.

`GPT-4` **agent plan:**
Information Extraction: To investigate the relationship between the number of employees and the number of shops in each city, we need to extract the following information:
The city where each employee works
The number of employees working in each city
The number of shops in each city
Identification of Relevant Tables and Columns: To gather the necessary information, we need to look into the following tables and columns:
From the 'employee' table, we need the 'Employee_ID' and 'City' columns.
From the 'shop' table, we need the 'Shop_ID' and 'Location' columns.
From the 'hiring' table, we need the 'Shop_ID' and 'Employee_ID' columns.
Description of Retrieval Process: To retrieve the required information, we will perform the following steps:
First, we will join the 'employee' and 'hiring' tables on the 'Employee_ID' column. This will give us a combined table with information about which employee works at which shop.
Next, we will join the resulting table with the 'shop' table on the 'Shop_ID' column. This will give us a table containing information about the city where each employee works.
Now, we will group the data by the 'City' column from the 'employee' table and the 'Location' column from the 'shop' table. This will allow us to count the number of employees and shops in each city.
For each city, we will count the number of unique 'Employee_ID' values to get the total number of employees working in that city.
Similarly, for each city, we will count the number of unique 'Shop_ID' values to get the total number of shops in that city.
Finally, we will create a summary table with columns for 'City', 'Number_of_Employees', and 'Number_of_Shops'. This table will show the relationship between the number of employees and the number of shops in each city.

Figure 11: Example 1 - misinterpreting column contents: agent confuses employee.city with shop.location

Database:
Title: 'Addresses'
Header: ['address_id', 'line_1', 'line_2', 'line_3', 'city', 'zip_postcode', 'state_province_county', 'country', 'other_address_details']
Title: 'Courses'
Header: ['course_id', 'course_name', 'course_description', 'other_details']
Title: 'Departments'
Header: ['department_id', 'department_name', 'department_description', 'other_details']
Title: 'Degree_Programs'
Header: ['degree_program_id', 'department_id', 'degree_summary_name', 'degree_summary_description', 'other_details']
Title: 'Sections'
Header: ['section_id', 'course_id', 'section_name', 'section_description', 'other_details']
Title: 'Semesters'
Header: ['semester_id', 'semester_name', 'semester_description', 'other_details']
Title: 'Students'
Header: ['student_id', 'current_address_id', 'permanent_address_id', 'first_name', 'middle_name', 'last_name', 'cell_mobile_number', 'email_address', 'ssn', 'date_first_registered', 'date_left', 'other_student_details']
Title: 'Student_Enrolment'
Header: ['student_enrolment_id', 'degree_program_id', 'semester_id', 'student_id', 'other_details']
Title: 'Student_Enrolment_Courses'
Header: ['student_course_id', 'course_id', 'student_enrolment_id']
Title: 'Transcripts'
Header: ['transcript_id', 'transcript_date', 'other_details']
Title: 'Transcript_Contents'
Header: ['student_course_id', 'transcript_id']

Question:
Do dual-enrolled students tend to perform better or worse than their peers in the same degree programs?

`GPT-4` **agent plan:**
Information Extraction: To answer the user's question, we need to extract the following information:
The performance of dual-enrolled students in their courses.
The performance of non-dual-enrolled students in the same degree programs as the dual-enrolled students.
Identification of Relevant Tables and Columns: The relevant tables and columns needed to gather the necessary information are:
'Student_Enrolment' table: degree_program_id, semester_id, student_id, and other_details columns.
'Student_Enrolment_Courses' table: student_course_id, course_id, and student_enrolment_id columns.
'Transcript_Contents' table: student_course_id and transcript_id columns.
'Transcripts' table: transcript_id and other_details columns.
Description of Retrieval Process: To retrieve the required information, we would perform the following steps:
a. Identify dual-enrolled students:
Filter the 'Student_Enrolment' table to find rows where the 'other_details' column contains information about dual enrollment. This will give us a list of dual-enrolled students and their corresponding degree_program_id, semester_id, and student_id.
b. Identify non-dual-enrolled students in the same degree programs:
Filter the 'Student_Enrolment' table to find rows where the 'other_details' column does not contain information about dual enrollment and the degree_program_id matches the degree_program_id of the dual-enrolled students identified in step a. This will give us a list of non-dual-enrolled students in the same degree programs as the dual-enrolled students.
c. Retrieve course performance for dual-enrolled students:
For each dual-enrolled student identified in step a, find the corresponding student_enrolment_id in the 'Student_Enrolment_Courses' table. Using the student_course_id from the 'Student_Enrolment_Courses' table, find the corresponding transcript_id in the 'Transcript_Contents' table.
Using the transcript_id, retrieve the performance information (e.g., grades) from the 'Transcripts' table's other_details column. d. Retrieve course performance for non-dual-enrolled students:
For each non-dual-enrolled student identified in step b, find the corresponding student_enrolment_id in the 'Student_Enrolment_Courses' table. Using the student_course_id from the 'Student_Enrolment_Courses' table, find the corresponding transcript_id in the 'Transcript_Contents' table.
Using the transcript_id, retrieve the performance information (e.g., grades) from the 'Transcripts' table's other_details column. e. Compare the performance of dual-enrolled and non-dual-enrolled students:
Calculate the average performance of dual-enrolled students and non-dual-enrolled students using the performance information retrieved in steps c and d.
Compare the average performance of both groups to determine if dual-enrolled students tend to perform better or worse than their peers in the same degree programs.

Figure 12: Example 2 - making incorrect assumptions about column contents: student_enrolment.other_detail does not contain information about dual-enrollment

Database:
Title: 'TV_Channel'
Header: ['id', 'series_name', 'Country', 'Language', 'Content', 'Pixel_aspect_ratio_PAR', 'Hight_definition_TV', 'Pay_per_view_PPV', 'Package_Option']
Title: 'TV_series'
Header: ['id', 'Episode', 'Air_Date', 'Rating', 'Share', '18_49_Rating_Share', 'Viewers_m', 'Weekly_Rank', 'Channel']
Title: 'Cartoon'
Header: ['id', 'Title', 'Directed_by', 'Written_by', 'Original_air_date', 'Production_code', 'Channel']

Question:
Identify common themes or factors in the top-rated episodes and potential reasons for low ratings in the least-rated episodes of a TV series.

`GPT-4` **agent plan:**
Information Extraction: To answer the user's question, we need to extract the following information:
Top-rated episodes of the TV series
Least-rated episodes of the TV series
Common themes or factors in the top-rated episodes
Potential reasons for low ratings in the least-rated episodes
Identification of Relevant Tables and Columns: From the given database, the relevant tables and columns to gather the necessary information are:
'TV_series' table: This table contains information about the episodes, their air dates, ratings, and other related data. The relevant columns from this table are:
'Episode': The name of the episode
'Rating': The rating of the episode
'Air_Date': The date the episode was aired
'Channel': The channel on which the episode was aired
'TV_Channel' table: This table contains information about the channels, their content, and other related data. The relevant columns from this table are:
'id': The unique identifier of the channel
'Content': The type of content aired on the channel
'Package_Option': The package option for the channel
Description of Retrieval Process: To retrieve the required information, we would perform the following steps:
Filter the 'TV_series' table to only include rows related to the TV series in question.
Sort the filtered 'TV_series' table by the 'Rating' column in descending order to identify the top-rated episodes.
Sort the filtered 'TV_series' table by the 'Rating' column in ascending order to identify the least-rated episodes.
For the top-rated episodes, join the 'TV_series' table with the 'TV_Channel' table on the 'Channel' and 'id' columns, respectively. This will provide information about the content and package options for the channels on which the top-rated episodes were aired.
Analyze the joined data to identify common themes or factors in the top-rated episodes, such as the type of content aired on the channels, the package options, or any patterns in the air dates.
For the least-rated episodes, join the 'TV_series' table with the 'TV_Channel' table on the 'Channel' and 'id' columns, respectively. This will provide information about the content and package options for the channels on which the least-rated episodes were aired.
Analyze the joined data to identify potential reasons for low ratings in the least-rated episodes, such as the type of content aired on the channels, the package options, or any patterns in the air dates.

Figure 13: Example 3 - overlooking key columns: only tv_channel.package_option is retrieved for the analysis, while many other columns could also be influential factors, such as tv_channel.country, tv_channel.language, etc.

Database:
Title: 'singer'
Header: ['Singer_ID', 'Name', 'Birth_Year', 'Net_Worth_Millions', 'Citizenship']
Title: 'song'
Header: ['Song_ID', 'Title', 'Singer_ID', 'Sales', 'Highest_Position']

Question:
Compare the sales and highest chart positions of top-selling songs by singers who have over 300,000 sales.

`GPT-4` **agent plan:**
Information Extraction: To answer the user's question, we need to extract the following information:
Singer names with over 300,000 sales
The titles of their top-selling songs
The sales of these top-selling songs
The highest chart positions of these top-selling songs
Identification of Relevant Tables and Columns: To gather the necessary information, we will need to use the following tables and columns from the database:
From the 'singer' table, we will need the 'Singer_ID' and 'Name' columns
From the 'song' table, we will need the 'Title', 'Singer_ID', 'Sales', and 'Highest_Position' columns
Description of Retrieval Process: To retrieve the required information, we will follow these steps:
First, we will filter the 'song' table to only include rows with 'Sales' greater than 300,000. This will give us a list of songs that meet the sales criteria.
Next, we will join the filtered 'song' table with the 'singer' table using the 'Singer_ID' column as the common key. This will allow us to associate each song with its respective singer.
After joining the tables, we will group the resulting dataset by the 'Singer_ID' and 'Name' columns. This will help us identify the top-selling songs for each singer.
Within each group, we will sort the songs by their 'Sales' in descending order. This will allow us to identify the top-selling song for each singer.
Finally, we will extract the 'Name', 'Title', 'Sales', and 'Highest_Position' columns for the top-selling song of each singer. This will give us the information needed to compare the sales and highest chart positions of top-selling songs by singers who have over 300,000 sales.

Figure 14: Example 4 - exhibiting oversight in data retrieval for subsequent analysis: the agent's strategy involves selecting singers with over 300,000 sales by filtering for individual songs that have achieved 300,000 sales. This approach overlooks singers who may have accumulated over 300,000 sales in total but not through a single song.

Database:
Title: 'employee'
Header: ['Employee_ID', 'Name', 'Age', 'City']
Title: 'shop'
Header: ['Shop_ID', 'Name', 'Location', 'District', 'Number_products', 'Manager_name']
Title: 'hiring'
Header: ['Shop_ID', 'Employee_ID', 'Start_from', 'Is_full_time']
Title: 'evaluation'
Header: ['Employee_ID', 'Year_awarded', 'Bonus']

Question:
Is there a correlation between employee age and their employment duration or full-time status?

`GPT-4` **tool employment:**
SQL Statement 1:
SELECT e.Employee_ID, e.Age, h.Start_from, h.Is_full_time FROM employee e JOIN hiring h ON e.Employee_ID = h.Employee_ID;
Execution Result 1:

[(1, 23, '2009', 'T'), (2, 29, '2003', 'T'), (3, 43, '2011', 'F'), (4, 30, '2012', 'T'), (5, 29, '2013', 'T'), (6, 28, '2010', 'F'), (7, 29, '2008', 'T'), (11, 25, '2014', 'T'), (12, 35, '2011', 'F'), (13, 45, '2010', 'T'), (14, 27, '2015', 'T'), (15, 37, '2013', 'F'), (16, 29, '2016', 'T'), (17, 28, '2014', 'T'), (18, 30, '2012', 'F'), (19, 50, '2005', 'T'), (20, 40, '2009', 'F')]

SQL Statement 2:
SELECT e.Employee_ID, e.Age, (2022 - h.Start_from) AS Employment_Duration, h.Is_full_time FROM employee e JOIN hiring h ON e.Employee_ID = h.Employee_ID;

Execution Result 2:

[(1, 23, 13, 'T'), (2, 29, 19, 'T'), (3, 43, 11, 'F'), (4, 30, 10, 'T'), (5, 29, 9, 'T'), (6, 28, 12, 'F'), (7, 29, 14, 'T'), (11, 25, 8, 'T'), (12, 35, 11, 'F'), (13, 45, 12, 'T'), (14, 27, 7, 'T'), (15, 37, 9, 'F'), (16, 29, 6, 'T'), (17, 28, 8, 'T'), (18, 30, 10, 'F'), (19, 50, 17, 'T'), (20, 40, 13, 'F')]

Figure 15: Example 5 - failing to execute statistical analyses