

FinAI Data Assistant: LLM-based Financial Database Query Processing with the OpenAI Function Calling API

Juhyeong Kim

juhyeong.kim@miraasset.com
nonconvexopt@gmail.com
Mirae Asset Global Investments
AI Quant Lab, MODULABS
Seoul, Republic of Korea

Youngbin Lee

youngandbin@elicer.com
Elice
AI Quant Lab, MODULABS
Seoul, Republic of Korea

Yejin Kim

yejin.kim.ds@meritz.com
Meritz Fire & Marine Insurance
AI Quant Lab, MODULABS
Seoul, Republic of Korea

Hyunwoo Byun

hyunwoo.byun@miraasset.com
Mirae Asset Global Investments
Seoul, Republic of Korea

Abstract

We present **FinAI Data Assistant**, a practical approach for natural-language querying over financial databases that combines large language models (LLMs) with the OpenAI Function Calling API. Rather than synthesizing complete SQL via text-to-SQL, our system routes user requests to a small library of vetted, parameterized queries, trading generative flexibility for reliability, low latency, and cost efficiency. We empirically study three questions: (RQ1) whether LLMs alone can reliably recall or extrapolate time-dependent financial data without external retrieval; (RQ2) how well LLMs map company names to stock ticker symbols; and (RQ3) whether function calling outperforms text-to-SQL for end-to-end database query processing. Across controlled experiments on prices and fundamentals, LLM-only predictions exhibit non-negligible error and show look-ahead bias primarily for stock prices relative to model knowledge cutoffs. Ticker-mapping accuracy is near-perfect for NASDAQ-100 constituents and high for S&P 500 firms. Finally, FinAI Data Assistant achieves lower latency and cost and higher reliability than a text-to-SQL baseline on our task suite. We discuss design trade-offs, limitations, and avenues for deployment.

CCS Concepts

- Information systems → Query languages; Information retrieval query processing;
- Computing methodologies → Natural language processing.

Keywords

Database Management, Natural Language Processing, Large Language Models

ACM Reference Format:

Juhyeong Kim, Yejin Kim, Youngbin Lee, and Hyunwoo Byun. 2025. FinAI Data Assistant: LLM-based Financial Database Query Processing with the

CIKM'25, Seoul

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of The 34th ACM International Conference on Information and Knowledge Management (CIKM'25)*, <https://doi.org/10.1145/nnnnnnn.nnnnnnn>.

OpenAI Function Calling API. In *Proceedings of The 34th ACM International Conference on Information and Knowledge Management (CIKM'25)*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

Financial analysts, portfolio managers, and operations teams increasingly expect to access structured market and fundamental data using natural language. However, directly answering such queries with an LLM alone remains challenging for three reasons. First, LLMs have knowledge cutoffs and do not observe live data, making temporal accuracy difficult without external retrieval. Second, free-form generation can hallucinate field names, units, or filters, which is problematic in regulated settings where determinism, auditability, and cost control are paramount. Third, generic text-to-SQL pipelines often emit verbose or invalid queries, incurring latency and operational cost while still requiring extensive guardrails.

To address these challenges, we develop **FinAI Data Assistant**, a system that integrates an LLM with the OpenAI Function Calling API and a small set of vetted, parameterized SQL templates tailored to common financial data tasks (e.g., market prices and firm fundamentals). The LLM performs high-level intent classification and argument extraction; execution is delegated to trusted linking functions that instantiate the appropriate query with validated parameters. This design preserves the usability of natural-language interaction while ensuring stable latency, predictable cost, and reduced surface area for errors.

We organize our study around three research questions (RQs): (RQ1) Is integration with external databases necessary for accurate handling of temporal financial information before and after model knowledge cutoffs? (RQ2) To what extent can OpenAI LLMs correctly map company names to stock ticker symbols? (RQ3) Does the Function Calling approach deliver higher accuracy and efficiency than text-to-SQL for end-to-end database query processing?

Our contributions are threefold. *Design:* We describe a pragmatic architecture that uses a compact, reusable query library instead of generating full SQL from scratch. *Evaluation:* We present controlled experiments that quantify limitations of LLM-only recall, measure ticker-mapping coverage, and compare Function Calling with a text-to-SQL baseline in terms of latency, cost, and reliability. *Findings:*

We observe that (i) LLM-only predictions produce non-trivial errors and exhibit look-ahead bias mainly for prices, (ii) ticker mapping is near-perfect for NASDAQ-100 and strong for S&P 500 names, and (iii) FinAI Data Assistant is consistently faster and cheaper than text-to-SQL on our task suite while maintaining perfect task completion in our cases.

2 Related Works

Natural language interfaces to data (NLIDB) have a long history across databases and NLP. Recent surveys synthesize progress on semantic parsing for data access and, more specifically, LLM-based database interfaces [4, 15, 20, 31]. Core challenges recur throughout: schema linking, compositional generalization, disambiguation under incomplete context, and execution-time robustness under syntactic and semantic constraints.

Text-to-SQL. Early neural approaches such as Seq2SQL and SQL-Net targeted single-table settings (e.g., WikiSQL) via constrained decoding or sketch-based generation [25, 30]. The field subsequently shifted to cross-domain, multi-table parsing with the introduction of Spider and CoSQL, catalyzing research on schema encoding and conversational querying [28, 29]. State-of-the-art parsers model rich schema relations and execution constraints (e.g., RAT-SQL, BRIDGE, SmBoP, PICARD), or unify structured grounding tasks in a text-to-text formulation (UnifiedSKG) [8, 16, 19, 21, 24]. With the advent of LLMs, in-context learning, decomposition, and self-correction further improve robustness (e.g., DIN-SQL, RESDSQL), while surveys provide comprehensive comparisons of prompting and tool-augmented strategies [7, 13, 20, 31]. Despite progress, pure text-to-SQL pipelines can emit lengthy or invalid queries, require extensive guardrails, and incur variable latency and cost in production.

Tool use and function calling. A complementary line of work studies LLM agents that reason while invoking tools or APIs. ReAct interleaves reasoning and acting, Toolformer explores self-supervised tool-use training, and Gorilla maps natural language to large API surfaces [14, 18, 27]. In practical systems, function calling constrains the model to produce structured arguments for pre-registered capabilities, enabling determinism and safer integration [11]. Our approach adopts this paradigm for financial data access: instead of synthesizing arbitrary SQL, we route requests to a compact library of vetted, parameterized queries, balancing usability with auditability, latency, and predictable cost.

LLMs in finance and forecasting. Domain-specific LLMs (e.g., BloombergGPT) and open initiatives (e.g., FinGPT) investigate financial knowledge and downstream utility [23, 26]. In time series, LLMs show promising zero-shot behavior [3], yet recent studies question whether apparent forecasting skill reflects memorization or ex-post information, proposing ex-ante benchmarks and careful evaluation protocols [1, 9, 10, 12]. Relatedly, prior work highlights look-ahead bias risks in finance-oriented LLM pipelines (e.g., GPT-based sentiment for return prediction) [2]. These findings motivate our RQ1 analysis and our decision to integrate explicit database retrieval rather than rely on model recall of time-varying quantities. Recent studies have actively investigated the portfolio-optimization capabilities of large language models (LLMs) [5, 6].

Evaluation methodology. For statistical comparisons under unequal variances and sample sizes, we follow standard practice using Welch’s *t*-test with degrees-of-freedom approximation per Satterthwaite [17, 22]. Our experimental design, however, emphasizes end-to-end metrics (latency, token cost, task completion) that reflect practical system behavior under realistic financial queries.

3 Methods

3.1 Preliminaries: OpenAI Function Calling API

OpenAI introduced the Function Calling API to allow developers to define functions and have models produce structured outputs (e.g., JSON arguments) that can be executed by external systems. This mechanism bridges natural-language inputs and programmatic actions.

Advantages.

- **Structured Output** — Models emit standardized JSON arguments, reducing ambiguity and simplifying downstream processing.
- **System Integration** — Natural-language requests can be mapped to function calls that interact with databases, APIs, or computation engines.
- **Error Reduction** — Compared with free-form text, structured outputs reduce parsing errors and improve reliability in multi-step pipelines.
- **Workflow Automation** — Natural language serves as a high-level interface for automating complex workflows.

Limitations.

- **Design Overhead** — Developers must carefully specify schemas; poor schemas yield incorrect or incomplete outputs.
- **Latency and Cost** — Function invocation introduces overhead relative to pure text generation.
- **Security** — As model outputs can trigger actions, validation and sanitization are essential.

Basic Usage. A typical implementation has three stages: (1) **Function Definition** specifying names, descriptions, and parameter schemas; (2) **Model Invocation** with both user input and function specs so the model can choose whether/how to call a function; and (3) **Function Execution** where returned JSON arguments are validated and dispatched to application code. The flow is thus *natural language → structured JSON → function execution*.

3.2 FinAI Data Assistant

We propose **FinAI Data Assistant**, a database query processing method that integrates OpenAI LLMs and Function Calling to answer natural-language questions about financial data in a fast and economical manner. We assume the service provider maintains existing financial databases (e.g., market and fundamental data). Implementation proceeds by defining a *minimal, common set* of parameterized SQL statements for each data type. This reduces redundancy relative to text-to-SQL systems and lowers both cost and error rate.

Concretely, we build separate base statements for market data and fundamental data. We then implement linking functions that execute these statements with validated parameters. Finally, an

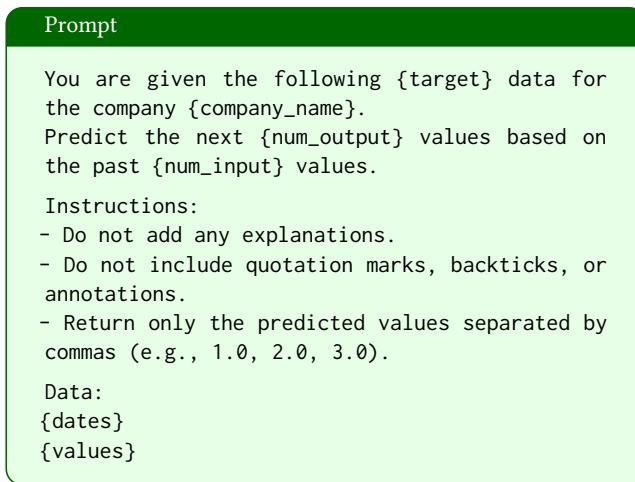
LLM determines which linking function to use and what parameter values to supply. This design concentrates complexity in a small, auditable query library while preserving the flexibility of natural-language interaction.

4 Experiments

4.1 Necessity of LLM–Database Integration (RQ1)

We evaluate whether LLMs alone can accurately recover financial time series given only a company name and short histories. Specifically, we ask models to predict stock prices (daily closes) and fundamental metrics (revenue and net income) at specified periods. For stock prices, the model predicts 10 daily values from the preceding 10 business days; for fundamentals, it predicts 8 quarters from the previous 8 quarters. Dates are explicitly provided.

We use the following prompt template:



For each ticker i with horizon length T , we compute the log mean squared error (MSE):

$$\log(\text{MSE}_i) = \log \left(\frac{1}{T} \sum_{t=1}^T (\hat{x}_{it} - x_{it})^2 \right), \quad (1)$$

where \hat{x}_{it} and x_{it} denote predictions and ground truth, respectively.

Table 1: Average log(MSE) across tickers. Left values use data before each model’s knowledge cutoff; right values use the latest data after the cutoff.

Model	Stock Price	Revenue	Net Income
gpt-4o	-2.54 -1.39	6.94 6.41	6.21 6.24
gpt-4.1	-2.13 -1.33	6.33 6.18	5.88 5.83
gpt-5	-2.18 -1.05	7.18 6.63	6.19 6.16

Table 1 shows that LLM-only predictions incur non-negligible error, indicating that models cannot reliably retrieve exact financial values from short histories without database access.

As a supplementary analysis, we test for look-ahead bias with respect to each model’s knowledge cutoff via a one-sided Welch’s

t-test. [22] Welch’s T-test allow us to compare the means of group with different variances. Let

$$X_i = \log(\text{MSE}_i^b), \quad i \in G_1, \quad n_1 = |G_1| \quad (2)$$

$$Y_j = \log(\text{MSE}_j^a), \quad j \in G_2, \quad n_2 = |G_2| \quad (3)$$

where superscripts b and a denote before and after the knowledge cutoff. The test statistic is

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{s_X^2}{n_1} + \frac{s_Y^2}{n_2}}}, \quad (4)$$

with degrees of freedom approximated by the Welch–Satterthwaite formula [17]:

$$v = \frac{\left(\frac{s_X^2}{n_1} + \frac{s_Y^2}{n_2} \right)^2}{\frac{(s_X^2/n_1)^2}{n_1-1} + \frac{(s_Y^2/n_2)^2}{n_2-1}}. \quad (5)$$

We obtain p-values using the Student’s t distribution with v degrees of freedom.

Table 2: P-values for one-sided Welch’s t-test.

Model	Stock Price	Revenue	Net Income
gpt-4o	0.0003	0.8764	0.4681
gpt-4.1	0.0137	0.6262	0.5381
gpt-5	0.0003	0.8568	0.5252

At the 0.05 significance level, only the stock price comparisons reject the null hypothesis, suggesting look-ahead bias is present primarily for stock prices, not for revenue or net income.

4.2 Basic Company Information Understanding (RQ2)

To address RQ2, we evaluate OpenAI LLMs on a ticker-retrieval task: given a company name, return its stock ticker. Strong performance would indicate that models retain sufficient entity knowledge to support common financial queries. Figure 1 summarizes results.

For NASDAQ-100 constituents, accuracy is effectively perfect for both the latest set and the pre-cutoff set for each model (with a single miss for gpt-4.1). For S&P 500 companies, accuracy increases with newer models; as expected, performance for the latest constituents is slightly lower than for the pre-cutoff set, but overall accuracy remains high. These results suggest that OpenAI LLMs can reliably map company names to tickers, especially for widely covered firms.

4.3 Database Query Performance (RQ3)

Figure 2 compares end-to-end cost, latency, and task accuracy. Blue denotes **FinAI Data Assistant** (Function Calling); orange denotes the text-to-SQL baseline. A representative task is: “What are the latest 10 daily closing prices for NVIDIA?” A system succeeds only if it returns the requested values correctly and completely.

Cost. We compute cost as the sum of input and output tokens multiplied by model-specific rates. Text-to-SQL incurs substantially higher and more variable costs than FinAI Data Assistant across models, while the latter remains negligible in comparison.

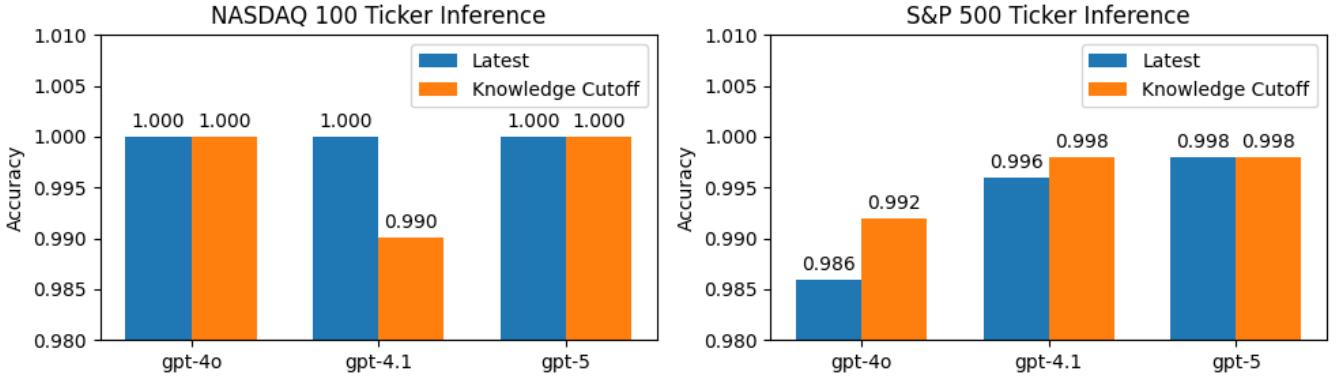


Figure 1: Ticker coverage test. We provide a company name and ask the model for the corresponding ticker. “Latest” denotes accuracy for the most recent index constituents. “Knowledge Cutoff” denotes accuracy for the same sets restricted to membership before each model’s knowledge cutoff.

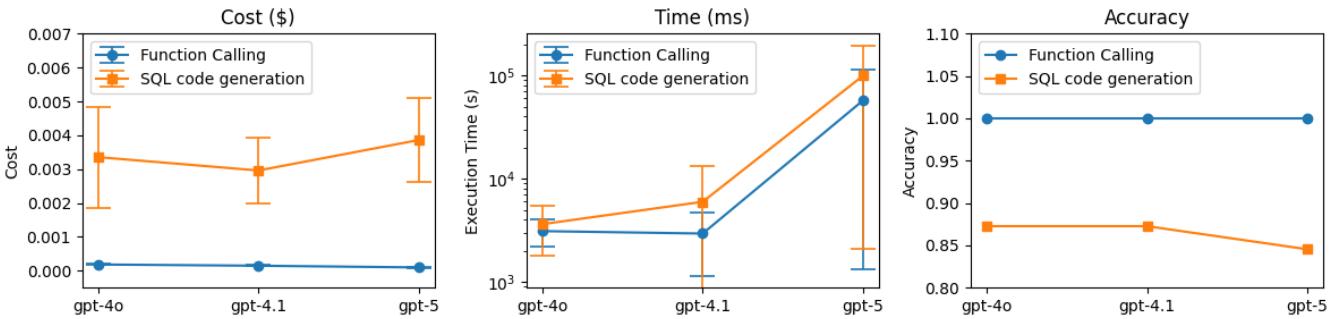


Figure 2: Cost, time, and accuracy for database query processing: Function Calling (FinAI Data Assistant) versus text-to-SQL. All metrics are measured for successful completion of each test case. Error bars in the cost and time plots indicate ± 1 standard deviation.

Latency. We measure total time to completion. For FinAI Data Assistant, this includes function-call inference and linking-function execution (including SQL). For text-to-SQL, it includes SQL generation and execution. On a log scale, text-to-SQL is consistently slower.

Accuracy. Within our test suite, FinAI Data Assistant completed all cases successfully, whereas text-to-SQL exhibited failures across models. While our tasks are intentionally practical rather than adversarial, the results indicate that text-to-SQL is more error-prone than our function-calling approach under typical usage.

In sum, FinAI Data Assistant delivers strong practical performance relative to text-to-SQL in our evaluations.

5 Conclusion

We introduced **FinAI Data Assistant**, an LLM-driven system for natural-language access to financial databases that leverages the OpenAI Function Calling API and a small library of parameterized SQL templates. Our experiments support three conclusions. First, LLMs alone cannot reliably reconstruct time-varying financial values from short histories and show look-ahead bias primarily for prices, underscoring the need for explicit database integration

(RQ1). Second, OpenAI LLMs exhibit near-perfect ticker mapping for NASDAQ-100 and high accuracy for S&P 500 companies, enabling dependable entity resolution in common workflows (RQ2). Third, the function-calling approach achieves lower cost and latency with higher reliability than a text-to-SQL baseline on our task suite (RQ3).

Acknowledgments

This research was supported by the **Brian Impact Foundation**, a non-profit organization dedicated to the advancement of science and technology for all.

This document was prepared by the **AI Solution Team** at **Mirae Asset Global Investments** for informational purposes only. **Mirae Asset Global Investments** makes no representation or warranty as to the accuracy, completeness, or reliability of the information contained herein and accepts no liability for any consequences arising from its use. This document does not constitute investment advice, a recommendation, or an offer to buy or sell any financial products, and should not be relied upon as a basis for investment decisions.

References

- [1] Leland D. Crane, Akhil Karra, and Paul E. Soto. 2025. *Total Recall? Evaluating the Macroeconomic Knowledge of Large Language Models*. Finance and Economics Discussion Series (FEDS) 2025-044. Board of Governors of the Federal Reserve System. doi:10.17016/FEDS.2025.044
- [2] Paul Glasserman and Cader Lin. 2024. Assessing Look-Ahead Bias in Stock Return Predictions Generated by GPT Sentiment Analysis. *The Journal of Financial Data Science* 6, 1 (2024), 25–42. doi:10.3905/jfds.2023.1.143
- [3] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023. Large Language Models Are Zero-Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems (NeurIPS)*. <https://arxiv.org/abs/2310.07820>
- [4] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024. Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. *arXiv preprint* (2024). arXiv:2406.08426 [cs.DB] <https://arxiv.org/abs/2406.08426>
- [5] Yejin Kim, Youngbin Lee, Juhyeong Kim, and Yongjae Lee. 2025. GuRuAgents: Emulating Wise Investors with Prompt-Guided LLM Agents. arXiv:2510.01664 [cs.AI]
- [6] Youngbin Lee, Yejin Kim, Juhyeong Kim, Suin Kim, and Yongjae Lee. 2025. LLM-Enhanced Black-Litterman Portfolio Optimization. arXiv:2504.14345 [q-fin.PM]
- [7] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 13067–13075. <https://ojs.aaai.org/index.php/AAAI/article/view/26535>
- [8] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics. doi:10.18653/v1/2020.findings-emnlp.438
- [9] Yachuan Liu, Xiaochun Wei, Lin Shi, Xinnuo Li, Bohan Zhang, Paramveer Dhillon, and Qiaozhu Mei. 2025. ExAnte: A Benchmark for Ex-Ante Inference in Large Language Models. *arXiv preprint* (2025). arXiv:2505.19533 [cs.LG] doi:10.48550/ arXiv.2505.19533
- [10] Juan Felipe Lopez-Lira, Yuexi Tang, and Lin Zhu. 2025. The Memorization Problem: Can We Trust LLMs’ Economic Forecasts? *arXiv preprint* (2025). arXiv:2504.14765 [econ.GN] <https://arxiv.org/abs/2504.14765>
- [11] OpenAI. 2023. Function calling with the OpenAI API. <https://platform.openai.com/docs/guides/function-calling>. Accessed 2025-08-31.
- [12] Van Pham and Scott Cunningham. 2024. Can Base ChatGPT be Used for Forecasting without Additional Optimization? *arXiv preprint* (2024). arXiv:2404.07396 [econ.GN] doi:10.48550/arXiv.2404.07396
- [13] Mohammadreza Pourreza and Davood Rafiee. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. https://proceedings.neurips.cc/paper_files/paper/2023/hash/72223cc66f63ca1aa59edaec1b3670e6-Abstract-Conference.html
- [14] Ishita Prasad, Aman Madaan, Shishir G. Patil, Mayee F. Chen, Percy Liang, Joseph E. González, and Ion Stoica. 2024. Gorilla: Large Language Model Connected with Massive APIs. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*. https://proceedings.neurips.cc/paper_files/paper/2024/file/e4c61f578ff07830f5c37378dd3ecb0d-Paper-Conference.pdf
- [15] Abdul Quamar, Vasilis Eftymiou, Chuan Lei, and Fatma Özcan. 2022. Natural Language Interfaces to Data. *Foundations and Trends in Databases* 11, 4 (2022), 319–414. doi:10.1561/1900000078
- [16] Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-Autoregressive Bottom-Up Semantic Parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. <https://aclanthology.org/2021.naacl-main.29/>
- [17] Frank E. Satterthwaite. 1946. An Approximate Distribution of Estimates of Variance Components. *Biometrics Bulletin* 2, 6 (1946), 110–114. doi:10.2307/3002019
- [18] Timo Schick, Jane Dwivedi-Yu, Roberto Densi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf
- [19] Frederik Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding for Semantic Parsing. *arXiv preprint arXiv:2109.05093* (2021). <https://arxiv.org/abs/2109.05093>
- [20] Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. 2025. A Survey on Employing Large Language Models for Text-to-SQL Tasks. *arXiv preprint arXiv:2407.15186* (2025). <https://arxiv.org/abs/2407.15186>
- [21] 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*. Association for Computational Linguistics. <https://aclanthology.org/2020.acl-main.677/>
- [22] B. L. Welch. 1947. The Generalization of "Student's" Problem when Several Different Population Variances are Involved. *Biometrika* 34, 1–2 (1947), 28–35. doi:10.1093/biomet/34.1-2.28
- [23] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabrowski, Mark Dredze, Sebastian Gehrmann, Prabhjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. *arXiv preprint arXiv:2303.17564* (2023). <https://arxiv.org/abs/2303.17564>
- [24] 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*. <https://arxiv.org/abs/2201.05966> arXiv:2201.05966
- [25] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries from Natural Language Without Reinforcement Learning. *arXiv preprint arXiv:1711.04436* (2017). <https://arxiv.org/abs/1711.04436>
- [26] Hongyang (Bruce) Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. FinGPT: Open-Source Financial Large Language Models. *arXiv preprint arXiv:2306.06031* (2023). <https://arxiv.org/abs/2306.06031>
- [27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Peter J. Liu, and Karthik Narasimhan. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/2210.03629>
- [28] Tao Yu, Rui Zhang, He Yang 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 Fabri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. 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*. Association for Computational Linguistics. [https://aclanthology.org/D19-1204/](https://aclanthology.org/D19-1204)
- [29] 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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium. [https://aclanthology.org/D18-1425/](https://aclanthology.org/D18-1425)
- [30] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *arXiv preprint arXiv:1709.00103* (2017). <https://arxiv.org/abs/1709.00103>
- [31] Xiaohu Zhu, Qian Li, Lizhen Cui, and Yongkang Liu. 2024. Large Language Model Enhanced Text-to-SQL Generation: A Survey. *arXiv preprint* (2024). arXiv:2410.06011 [cs.DB] <https://arxiv.org/abs/2410.06011>

A Test cases for Database Query Performance (RQ3) experiment.

test cases

Return NVDA's stock price for the past 1 days.
 Return Walmart's stock price for the past 1 days.
 Return MSFT's stock price for the past 5 days.
 Return Meta's stock price for the past 5 days.
 Return Coca-Cola's revenue for 1 quarter.
 Return TSLA's total asset for 5 quarter.
 Return Costco's yearly net income for 1 year.
 Return AMZN's yearly interest expense for 5 year.
 Return AAPL's stock price for the past 3 days.
 Return GOOGL's stock price for the past 10 days.
 Return TSLA's stock price for the past 8 weeks.
 Return AMZN's stock price for the past 12 months.
 Show JPM's stock price between 2024-11-01 and 2024-12-31.
 Return NVDA's stock price for the past 1 year with weekly periodicity.
 Return META's stock price for the past 5 days
 Return MSFT's stock price for the past 6 months.
 Return Apple's revenue for the last 4 quarters.
 Return Alphabet's net income for 2 quarters.
 Return Amazon's total assets for 3 quarters.
 Return Tesla's total liabilities for 4 quarters.
 Return Microsoft's operating income for 5 quarters.
 Return Nvidia's R&D expense for 4 quarters.
 Return Walmart's SG&A expense for 3 quarters.
 Return Coca-Cola's gross profit for 2 quarters.
 Return Costco's operating cash flow for 4 quarters.
 Return Meta's capital expenditures for 4 quarters.
 Return the income statement for AAPL for the last 2 quarters.
 Return the balance sheet for MSFT for the last 1 quarter.
 Return the cash flow statement for AMZN for the last 3 quarters.
 Return NVDA's income statement between 2023 Q2 and 2024 Q1.
 Return TSLA's balance sheet between 2022 Q4 and 2023 Q3.
 Return WMT's cash flow statement for 2021 Q1 to 2022 Q2.
 Return META's yearly net income for the past 3 years.
 Return ORCL's yearly revenue for the past 5 years.
 Return NFLX's yearly interest expense for the past 2 years.
 Return PEP's yearly dividends paid for the past 3 years.
 Return INTC's yearly total asset for the past 5 years.
 Return IBM's yearly EPS (diluted) for the past 3 years.
 Return AAPL's stock price for the past 7 days.
 Return MSFT's stock price for the past 14 days.
 Return TSLA's stock price for the past 30 days.
 Return META's stock price for the past 90 days.
 Return AMZN's stock price for the past 180 days.
 Return NVDA's stock price for the past 2 years.
 Return GOOGL's stock price for the past 3 years.
 Return NFLX's stock price for the past 5 days.
 Return ORCL's stock price for the past 10 days.
 Return IBM's stock price for the past 2 weeks.
 Return INTC's stock price for the past 6 months.
 Return META's stock price for the past 1 year.
 Return JPM's stock price between 2024-01-01 and 2024-03-31.
 Return PEP's stock price between 2023-05-01 and 2023-06-15.
 Return WMT's stock price for the past 2 years with monthly periodicity.
 Return KO's stock price for the past 4 years with yearly periodicity.

Return MSFT's daily closing price for the past 20 days.
 Return TSLA's stock price for the past 15 days.
 Return AAPL's stock price from 2023-01-01 to 2023-06-30.
 Return AMZN's stock price for the past 3 months.
 Return AAPL's quarterly revenue for the past 6 quarters.
 Return MSFT's quarterly net income for the past 8 quarters.
 Return TSLA's quarterly operating income for the past 10 quarters.
 Return NVDA's quarterly gross profit for the past 12 quarters.
 Return META's quarterly cost of revenue for the past 4 quarters.
 Return AMZN's quarterly EPS (basic) for the past 6 quarters.
 Return WMT's quarterly EPS (diluted) for the past 8 quarters.
 Return KO's quarterly total liabilities for the past 4 quarters.
 Return PEP's quarterly total equity for the past 6 quarters.
 Return INTC's quarterly research and development expense for 5 quarters.
 Return NFLX's quarterly SG&A expense for 3 quarters.
 Return ORCL's quarterly interest expense for 4 quarters.
 Return IBM's quarterly income tax expense for 2 quarters.
 Return META's quarterly total assets for 5 quarters.
 Return JPM's quarterly net interest income for 3 quarters.
 Return BAC's quarterly provisions for credit losses for 4 quarters.
 Return GS's quarterly revenue for 6 quarters.
 Return MS's quarterly net income for 8 quarters.
 Return C's quarterly EPS for 5 quarters.
 Return V's quarterly operating margin for 3 quarters.
 Return AAPL's yearly revenue for the past 10 years.
 Return MSFT's yearly operating income for the past 7 years.
 Return TSLA's yearly net income for the past 5 years.
 Return NVDA's yearly R&D expense for the past 8 years.
 Return META's yearly capital expenditures for the past 6 years.
 Return AMZN's yearly total asset for the past 4 years.
 Return WMT's yearly SG&A expense for the past 3 years.
 Return KO's yearly net income for the past 12 years.
 Return PEP's yearly gross profit for the past 9 years.
 Return INTC's yearly total assets for the past 15 years.
 Return NFLX's yearly operating cash flow for the past 7 years.
 Return ORCL's yearly dividends paid for the past 5 years.
 Return IBM's yearly EPS (basic) for the past 8 years.
 Return META's yearly gross profit for the past 6 years.
 Return JPM's yearly interest income for the past 4 years.
 Return BAC's yearly interest expense for the past 5 years.
 Return GS's yearly operating revenue for the past 7 years.
 Return MS's yearly income before tax for the past 6 years.
 Return C's yearly net income for the past 3 years.
 Return V's yearly transaction revenue for the past 5 years.
 Return the income statement for TSLA for the last 3 quarters.
 Return the balance sheet for NVDA for the last 2 quarters.
 Return the cash flow statement for META for the last 4 quarters.
 Return AAPL's income statement for the past 2 years.
 Return MSFT's balance sheet for the past 5 years.
 Return AMZN's cash flow statement for the past 3 years.
 Return WMT's income statement from 2020 to 2022.
 Return KO's balance sheet between 2018 and 2020.
 Return PEP's cash flow statement from 2021 to 2023.
 Return NFLX's income statement for 2022 Q1 to 2023 Q4.
 Return ORCL's balance sheet for 2019 to 2021.
 Return IBM's cash flow statement between 2020 and 2022.