Text-to-SQL Evaluation at Scale: A Semi-Automatic Approach for Benchmark Data Generation

Melody Xuan
Amazon
Bellevue, WA, USA
qiminx@amazon.com

Zhenyu Zhang
Amazon
Bellevue, WA, USA
zhenyuzh@amazon.com

Çağatay Demiralp
Amazon
Bellevue, WA, USA
cagatayd@amazon.com

Xinyang Shen
Amazon
Bellevue, WA, USA
xinyans@amazon.com

Ozalp Ozer
Amazon
Bellevue, WA, USA
ozalpo@amazon.com

Niti Kalra
Amazon
Bellevue, WA, USA
nitkalra@amazon.com

ABSTRACT

The practical effectiveness of text-to-SQL systems is often limited by the scarcity of domain-specific benchmark datasets. Manually curating these benchmarks struggles with translating ambiguous business question amidst complex metrics, domain knowledge, and data structures. To address this bottleneck, we propose a novel semi-automatic approach for generating high-quality text-SQL pairs via reverse engineering. Our method generates diverse SQL queries by sampling from and altering prepared SQL templates, then employs Large Language Models to synthesize corresponding natural language questions grounded in these valid SQL instances. We demonstrate this methodology through a case study in an e-commerce seller domain, yielding 702 diverse seller question-SQL pairs covering sales and inventory scenarios. Domain expert validation confirmed 95.11% question generation accuracy, significantly reducing manual efforts while maintaining high fidelity. This methodology can be adapted to other domains, facilitating the development of more reliable text-to-SQL systems across various industrial applications.

KEYWORDS

large language models, evaluation, benchmark, text-to-SQL

ACM Reference Format:
Melody Xuan, Zhenyu Zhang, Çağatay Demiralp, Xinyang Shen, Ozalp Ozer, and Niti Kalra. 2025. Text-to-SQL Evaluation at Scale: A Semi-Automatic Approach for Benchmark Data Generation. In *Proceedings of Agentic & GenAI Evaluation KDD2025: KDD workshop on Evaluation and Trustworthiness of Agentic and Generative AI Models (KDD'25).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Practical text-to-SQL systems often involve domain-specific complexities and interactions with databases that extend far beyond general benchmarks. It has been demonstrated that existing state-of-the-art text-to-SQL systems often fail to address real-world enterprise setting queries, despite their impressive performance with the existing text-to-SQL benchmarks [2, 7]. We note several challenges associated with developing domain-specific text-to-SQL systems: 1) the schema complexity of large-scale enterprise databases and incurred ambiguity in user queries, 2) the need to incorporate domain-specific business metrics and temporal analyses that require sophisticated computations, 3) the dual requirement of maintaining high accuracy for high-stakes business decisions while ensuring broad query coverage and 4) the need for scalable evaluation methods to assess system performance.

A critical barrier to developing robust text-to-SQL systems in practice is the scarcity of domain-specific benchmark datasets, which is often attributed to data privacy concerns, regulatory restrictions, and most importantly, the resource-intensive process of creating high-quality evaluation datasets. Unlike general-purpose benchmark datasets, domain-specific settings often involve complex, or sensitive database schemas that require expert knowledge to interpret accurately. Writing natural language questions that reflect practical analytical needs require not only domain knowledge but also a deep understanding of the databases. Moreover, aligning each question with a correct, executable SQL query is a non-trivial task, often requiring manual verification or domain expert input. These constraints make the manual annotation process time-consuming and expensive, limiting scalability. Additionally, domain-specific queries my rely on temporal analysis and contextual assumptions that are difficult to capture in a standardized annotation format, further complicating both data curation and downstream evaluation. To address this crucial gap, we propose a semi-automatic methodology that significantly reduces the manual efforts required to curate ground-truth question-SQL pairs, thereby enabling more comprehensive and efficient evaluation of text-to-SQL systems in domain-specific contexts. While full automation might offer faster dataset generation, it would risk missing critical domain-specific nuances and producing unrealistic query variations. Our approach strategically combines human expertise in crafting SQL query templates with automated processes for question generation, while maintaining human oversight for quality control and domain-specific refinements, striking a balance between efficiency and reliability in enabling more comprehensive evaluation of text-to-SQL systems in domain-specific contexts.

Our proposed methodology involves a reverse-engineering strategy by transforming the text-to-SQL task to a SQL-to-text problem. Our method begins with a carefully selected set of "gold" SQL queries representing common domain-specific user queries. These queries are transformed into templates with strategic placeholders for business metrics, temporal information or other relevant parameters. We prepare those templates via a manual process. By systematically sampling and augmenting these templates using those placeholders, we effectively create a diverse pool of ground-truth SQL queries. We then leverage large language models (LLMs) to interpret the resulting SQL codes and generate natural language questions that correspond to each of the SQL query respectively. This SQL-to-text approach offers several key advantages. By starting with valid SQL queries, we guarantee query validity and correct reference to table and column names. We can systematically control coverage of different SQL patterns and complexities. Additionally, this enables more cost-effective quality control as reviewing generated questions is typically easier than reviewing generated SQL. Lastly, as confirmed in our numerical results, generating questions from SQL is much easier than generating SQL from questions, with SQL-to-text conversion achieves 57.41% higher accuracy on complex SQL queries than text-to-SQL tasks. The resulting dataset consists of question-SQL pairs, which can be used to evaluate a text-to-SQL system's performance by measuring the execution accuracy based on the query results from the system and the ground-truth SQL.

We demonstrate the effectiveness of our approach through a case study in the seller domain. Specifically, we applied our strategies while developing a text-to-SQL system that supports analytical capabilities to provide sellers with personalized business support and insights. We developed a dataset through our proposed methodology, achieving 95.11% accuracy on the validity of the question and answer pairs. Using this dataset for evaluation, we develop an LLM-based text-to-SQL interface in the seller domain. Our key contributions in this paper include:

- **Semi-automatic data generation and evaluation**: We introduce a novel semi-automatic evaluation data curation process, significantly reducing the manual effort required to establish ground-truth question-SQL pairs in the seller domain. By leveraging popular SQL queries augmented with sampled fields, our approach automates diverse question and ground-truth generation with accuracy of 95.11%, streamlining the evaluation of practical text-to-SQL systems. To the best of our knowledge, this is the first effort to delve into automatic evaluation method for seller data domain. Our proposed automatic method is domain-agnostic and can easily generalized to other domains.
- **Seller-domain text-to-SQL benchmark dataset**: We address the critical gap in text-to-SQL research for the e-commerce seller domain by introducing a benchmark dataset comprising of 702 diverse natural language seller queries paired with their corresponding SQL statements. Prepared using our novel data curation methodology to ensure both diversity and real-world applicability, this dataset spans key seller areas ranging from sales analytics to inventory management.
- **Seller-domain text-to-SQL framework**: We present a specialized framework that addresses unique challenges in the seller domain by handling complex business metrics analysis, query ambiguity and high precision requirement. Our framework leverages domain-aware query generation to effectively navigate complex e-commerce database structure, while employing a multi-phase prompting approach to ensure high accuracy and coverage. Through robust context interpretation, the framework handles varied natural language expressions while maintaining precision above 94% for high-stake seller decisions.

1.1 Related Work

The evaluation of text-to-SQL systems has traditionally relied on established benchmark datasets [2, 5, 7, 8, 19]. While these benchmarks aggregated thousands of examples from various domains, they often fail to capture domain-specific complexities. Notably, BEAVER's focus on enterprise data warehouses has highlighted the significant gaps between academic benchmarks and real-world business requirements [2]. Furthermore, while LLM-as-a-judge approaches have shown promise in evaluating general knowledge-based questions [9, 16], their application to data-dependent insights remains challenging due to the open-ended nature of such queries and the need for domain-specific validation.

Recent efforts to address the high cost of manual annotation have explored synthetic data generation approaches, including LLM-based text-SQL pair generation [18], template-based methods for various SQL dialects [14]. However, these approaches primarily focus on general-purpose datasets and don't address the unique challenges of domain-specific data curation, particularly in cases where representative user queries are scarce.

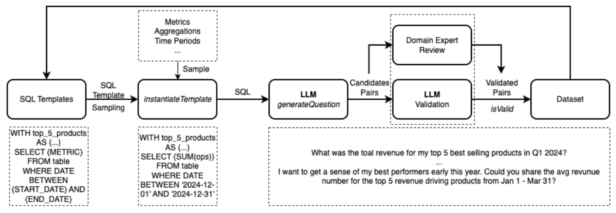
2 SEMI-AUTOMATIC BENCHMARK DATASET GENERATION

We now outline our methodology for semi-automatic benchmark data generation. While LLMs can generate question-SQL pairs directly, such synthetic outputs often contain unrealistic questions and require substantial manual validation effort. To address this challenge, we introduce a reverse-engineering approach that leverages existing validated SQL queries as our starting point, rather than attempting to generate queries from scratch or relying solely on manual annotation. This approach enables us to generate synthetic question-SQL pairs that are grounded in valid SQL queries, creating higher-quality questions that are more diverse and realistic [11]. Our procedure involves three main steps: data preparation, generation and validation, with the final step incorporating both automated and expert review to ensure data quality. The general question generation algorithm is presented in Algorithm 1. The overall data curation workflow with an example is illustrated in Figure 1.

Data Preparation. Our methodology begins with a carefully selected set of validated SQL queries that represent common domain-specific use cases. We transform these queries into templates by identifying key components that can be parameterized, such as metric columns, filtering conditions, and temporal references. To

Algorithm 1 Semi-automatic Question-SQL pair generation

```
Require: Set of SQL templates $T$
Require: Set of template specific parameters $C_t$ for each template
    $t \in T$, including possible table columns, aggregation functions,
    and time periods
Require: Total number of question-SQL pairs $N_t$ to be generated
    for each template
Require: Question generator: $generateQuestion(SQL, context)$
Require: Question-SQL pair validator: $isValid(q, SQL)$
1:  $D = \{\}$                                  ▷ Validated pairs of question-SQLs
2:  for each $t \in T$ do
3:      $n = 0$
4:      while $n < N_t$ do
5:          Sample $c \sim C_t$
6:          $SQL = instantiateTemplate(t, c)$        ▷ Generate SQL
    using template and parameters
7:          $q = generateQuestion(SQL, context)$        ▷ Generate
    question using SQL and relevant contexts such as table schema
8:          if $isValid(q, SQL)$ then   ▷ Validate with LLMs and/or
    domain expert
9:              $D \leftarrow D \cup \{(q, SQL)\}$
10:             $n = n + 1$
11:         end if
12:     end while
13: end for
14: return $D$
```



```
WITH top_5_products
AS (...)
SELECT {METRIC}
FROM table
WHERE DATE
BETWEEN
{START_DATE} AND
{END_DATE}
```

```
WITH top_5_products
AS (...)
SELECT {SUM(ops)}
FROM table
WHERE DATE
BETWEEN '2024-12-
01' AND '2024-12-31'
```

What was the toal revenue for my top 5 best selling products in Q1 2024?
...
I want to get a sense of my best performers early this year. Could you share the avg revunue number for the top 5 revenue driving products from Jan 1 - Mar 31?

Figure 1: Semi-Automatic Benchmarking Dataset Generation Workflow

ensure diversity in our ground-truth pairs, we implement systematic sampling strategies for these parameters based on their types. For metric columns sampling, we select key columns from the table schema and the relevant aggregation functions such as SUM, COUNT or AVG. For filter conditions with threshold values, we can sample from the statistical distributions based on their historical data or leverage domain-specific rules. For temporal analysis, we incorporate both dynamic time computation and fixed date values. For example, time references can be sampled as either specific dates (e.g., "2024-12-31") or relative references (e.g., "yesterday") that automatically adjust based on the query execution date. This approach reflects real-word user behavior where temporal references are often contextual and relative, while simultaneously expanding the template's applicability across different time periods.

Data Generation. Using these parameterized SQL templates, we create new SQL queries by systematically sampling values for the parameters (See Figure 1 for an example). We then leverage LLMs to generate several natural language questions that accurately reflect the underlying query intent. Note that generated questions for the same SQL query can be similar questions that are paraphrases to each other. To improve generation accuracy, we prompt the LLMs by providing the annotated table schema, temporal reference information and the resulting SQL query as contexts. This context helps ensure that generated questions can exhibit natural language variations while align with the provided SQL query.

Data Validation. We implement a two-stage validation process combining automated LLM-based filtering with domain expert review. The first stage utilizes LLMs to verify the consistency between generated questions and their corresponding SQL queries. This verification process can be achieved by leveraging LLMs to check whether the generated questions can be answered by the SQL query, or whether they are paraphases of each. To improve validation accuracy, we include the original validated SQL queries as few-shot examples in the LLM prompts. The second stage involves domain expert review to ensure the relevance and accuracy of generated questions to real seller use cases. This can help discard those that do not reflect real seller use cases or cannot be fully addressed with the ground-truth SQL query. This hybrid approach balances efficiency with quality assurance.

3 A CASE STUDY: QUESTION-SQL GENERATION FOR THE SELLER-DOMAIN

We demonstrate our proposed data generation strategy through a case study in the seller domain. We consider an LLM-empowered conversational assistant that enables sellers to better access and analyze their business data through natural language inquiries, delivering data-driven answers in narrative format. We developed a text-to-SQL framework for the seller-domain. Note that in contrast to general text-to-SQL tasks [3, 6, 8, 13, 19, 20], the seller-domain SQL query generation entails sophisticated adaptations in handling the following key challenges.

Ambiguity handling. Sellers may use ambiguous terminology when referring to business metrics or time period, further complicating intent-metric matching.

Complex metrics analysis. Seller metrics can have overlapping definitions or varying terminology, and may require further derivation and calculations, as they are not directly stored in the database.

High precision requirement. As sellers may perform subsequent high-stake actions, it is crucial to prioritize accuracy for SQL generation while ensuring decent coverage of seller questions.

3.1 Seller-Domain Benchmark Data Generation

Following the method outline in Section 2, we prepared a benchmark dataset that is designed for evaluating practical seller-domain text-to-SQL systems. The dataset covers business data from sales, traffic, inbound shipments and inventory. In handling the challenges of evaluating practical systems, we tailor the question-SQL pairs to reflect potential real-world communication errors, ambiguous temporal expressions and business context. The prompt for generating potential seller questions based on given SQL queries can be found in Appendix A.

3.2 A Seller-Domain Text-to-SQL Framework

We describe a high-level framework that addresses above challenges in the seller domain through three key components: table selection, SQL query generation, and response formulation, where the latter two steps are achieved via prompting LLMs. As illustrated in Figure 2, the process begins by identifying relevant tables and columns from complex database schema based on sellers' natural language queries. This contextual information is incorporated to prompt an LLM, which then generates a SQL query accordingly. After query execution, an LLM formulates a coherent response, translating raw database output back into natural language to address the seller query.

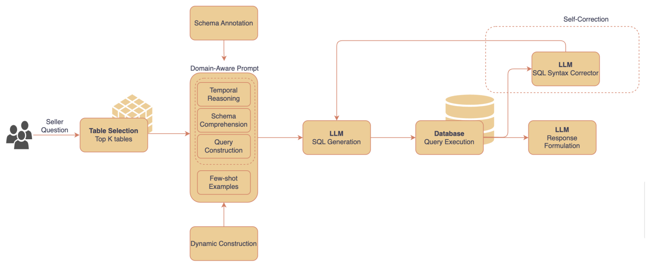


Figure 2: Proposed seller-domain text-to-SQL framework.

Table selection. The seller domain typically involves large-scale databases with a large number of tables (>50), which poses challenges for LLMs in utilizing information efficiently [10]. To improve the accuracy of metrics analysis based on seller queries, our framework performs top K relevant tables retrieval. Only these tables and corresponding columns are then considered in the subsequent SQL query generation step. Inspired by contextual retrieval from [12], we represent each table beyond its description, enriching it with contexts such as metadata and exemplars and then apply similarity search between tables and seller queries.

SQL query generation. In addressing the seller-domain-specific challenges, our framework employs a structured three-phase chain-of-thought (CoT) approach [17] embedded within a prompt template, focusing on selective aspects of SQL generation. In the first phase, the system performs temporal context analysis by maintaining an explicit representation of various time units, allowing for precise interpretation of relative time expressions and handling of calendar-based aggregations. This temporal reasoning is further enhanced by using default values to resolve time ambiguity and rolling window calculations to supplement LLMs' calculation capabilities. This is particularly crucial for business metrics that require comparative analysis across different time periods. The second phase focuses on schema comprehension, following a structured approach to understand complex metrics derivations, table relationships, column dependencies and data type compatibility. This phase is augmented by table schema annotation, where descriptions of both tables and columns are included to facilitate LLMs' understanding of domain-specific terminologies and thus better handling ambiguity in metric definitions. The final phase conducts query construction, following explicit reasoning steps to produce valid SQL queries while maintaining consistency with the original seller queries.

We also augment the prompt with LLMs' few-shot learning capabilities to improve SQL generation accuracy [1]. We include in the prompt key example triplets containing the seller question, reasoning and the correct SQL query, where the reasoning component explicitly captures the cognitive process of translating seller inquiry into structure query elements. The CoT few-shot exemplars benefits complicated query generation and are strategically selected to cover diverse use cases and temporal patterns. Currently, the CoT few-shot examples are generated via manual efforts. To create these reasoning at scale, we consider a Self-Taught Reasoner (STaR) approach which iteratively leverages a small number of rationale examples and a large dataset without rationales, to bootstrap the ability to perform successively more complex reasoning [4].

Final response generation. The final component of the framework addresses the challenge of generating accurate natural language responses from SQL query results for sellers. This step is crucial as it handles multiple sources of ambiguity while ensuring factual accuracy in the generated response.

3.3 Text-to-SQL Evaluations

We address ambiguity in seller queries by performing ambiguity-aware evaluations using a hierarchical matching approach that progresses through multiple levels of evaluations beyond exact result matching. This evaluation process not only ensures execution accuracy, but also aligns with seller experience improvement strategies by generating contextual answers or requesting potential clarifications. This process starts with strict comparison to execution results of ground-truth SQL. This level includes ground-truth answers as default interpretation of the original seller questions, allowing assessment of the system's capability in providing consistent answers. We then expand to consider variations in time periods, metric selection and result granularity that correspond to alternative interpretations of the seller query. By leveraging subset relationships between the generated SQL's execution results and the expanded ground-truth values, this hierarchical approach systematically evaluates the generated SQL's accuracy in a way that reflects both flexibility and fidelity to the ground-truth.

4 EXPERIMENTS

Metrics. To measure the system's performance in accurately retrieving data based on seller queries, we utilize the **execution accuracy** evaluation metric, which assesses the execution accuracy of the generated SQL query by comparing its execution result against the ground-truth results [8]. Alternative metrics include exact match or component matching between SQL queries. However, they are less relevant here since multiple SQL queries can be used to generate the same response [15].

Data. We utilize business data from five distinct tables: seller business report data, seller's inventory data, inbound and shipment creation data.

4.1 Results

Benchmarking and dataset evaluation. Through our proposed semi-automatic data curation process, we compiled a comprehensive evaluation dataset of 702 questions and their corresponding ground-truth SQL queries. The dataset consists of 255 manually prepared questions that were analyzed and summarized to prepare the SQL templates. The remaining 447 question-SQL pairs are curated through the semi-automatic evaluation methods in Section 2 for a more holistic evaluation of our proposed text-to-SQL system.

As validated by domain experts, the seller-domain dataset includes diverse expressions, time period references, metric selection and implicit ambiguity, achieving accuracy of 95.11% in generating valid problem-SQL pairs, where each generated question is correctly answerable by its corresponding SQL query. This accuracy remains consistent across varying SQL complexities. In contrast, the text-to-SQL model only achieves an accuracy of 37.70% when converting seller questions to their corresponding complex SQL queries. This significantly reduces the time and efforts in collecting seller questions and their respective ground-truth SQL queries.

Text-to-SQL evaluations. We evaluate our text-to-SQL system on our benchmark dataset and report the results in Table 1. We also benchmark our model performance against a vanilla LLM-based text-to-SQL model (Claude Sonnet 3.0). Overall, the framework achieved an overall accuracy of 94.3%, with 83.05% execution accuracy (`EA`) and the 11.25% categorized as alternative interpretations (`Alt`) of the seller questions. These alternative interpretations arise from ambiguities within the seller questions, such as multiple potential metrics, unclear time references or varying levels of data granularity. Since those alternative interpretations are valid responses to the ambiguous seller questions, we include them as part of a broad accuracy metric. However, further enhancements in interpreting seller intents are necessary to further improve user experience, particularly for ambiguous or complex questions.

Notably, the text-to-SQL framework outperforms the vanilla text-to-SQL system that is not tailored to the seller domain by yielding higher accuracy. The curated dataset, generated through the proposed semi-automatic evaluation procedure validates the reliability of the proposed the framework. Error analysis revealed that inaccuracies primarily arose from incorrect temporal analysis (e.g., misinterpreting date ranges), errors in column selection or computation, and SQL-related errors that lead to either execution failure or invalid results.

Ablation study. We present an ablation study in Table 2 to evaluate the contributions of individuals modules in our text-to-SQL prompt. For simplicity, we conduct the study on 255 questions that are manually collected. Removing any components results in a noticeable drop in the overall accuracy, with the largest drop occurring when Chain of Thought (CoT) reasoning is removed. While CoT is crucial for handling complex queries, schema annotations and few shot examples also enhance the system in reducing error rates. Note that the full system achieves the best accuracy at the cost of incurring the highest latency, due to the computational cost of incorporating all modules.

5 CONCLUSIONS AND NEXT STEPS

We tackle the challenges of developing domain-specific text-to-SQL systems by introducing a scalable semi-automatic data curation methodology. Through our case study in the seller domain, we demonstrate the effectiveness of this approach by developing both an evaluation benchmark dataset and an end-to-end text-to-SQL framework. While our curated dataset primarily serves evaluation purposes in this paper, it has broader potential applications for continuous improvements of text-to-SQL systems. Fully automating data curation and evaluations remain important research questions, especially for more complex data insights that may require more sophisticated reasoning. Moving forward, we plan to augment our question-SQL pairs with reasoning annotations, enabling systematic evaluation of reasoning capabilities in the seller domain.

Table 1: Evaluation results for text-to-SQL. EA: exact match with the ground-truth query execution result. Alt: alternative interpretation of the seller questions.

	Accuracy Types			Error Source			Accuracy
	Overall	EA	Alt	Time	SQL	Others	
Seller-Domain Text-to-SQL	94.30%	83.05%	11.25%	2.71%	1.42%	1.57%	94.30%
Vanilla Text-to-SQL	78.06%	60.54%	17.52%	12.82%	6.70%	2.42%	78.06%

Table 2: Ablation study for removing selected modules from our proposed text-to-SQL system.

	Overall Acc	EA	Alt	Error	Latency (P50/P90)
Text-to-SQL System	**93.73%**	**81.18%**	12.55%	**6.27%**	5.2s/8.3s
[-] Few Shot Examples	91.76 %(↓)	77.65%	14.12%	8.24%	4.3s/6.9s
[-] Schema Annotation	88.24%(↓)	76.08%	**12.16%**	11.77%	4.1s/6.5s
[-] CoT	78.04%(↓)	61.18%	16.86%	21.96%	2.0s/3.7s

REFERENCES

[1] Tom B. Brown, Benjamin Mann, and et al. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165

[2] Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2024. BEAVER: an enterprise benchmark for text-to-sql. *arXiv preprint arXiv:2409.02038* (2024).

[3] Zachary Huang Pavan Kalyan Damalapati and Eugene Wu. 2023. Data Ambiguity Strikes Back: How Documentation Improves GPT's Text-to-SQL. *NeurIPS* (2023).

[4] Zelikman Eric, Wu Yuhuai, Mu Jesse, and Goodman. Noah. 2022. STaR: Bootstrapping Reasoning With Reasoning. *36th Conference on Neural Information Processing Systems* (2022).

[5] Wuwei Lan, Zhiguo Wang, Anuj Chauhan, Henghui Zhu, Alexander Li, Jiang Guo, Sheng Zhang, Chung-Wei Hang, Joseph Lilien, Yiqun Hu, et al. 2023. Unite: A unified benchmark for text-to-sql evaluation. *arXiv preprint arXiv:2305.16265* (2023).

[6] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. *arXiv preprint arXiv:2106.11455* (2021).

[7] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763* (2024).

[8] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems* 36 (2024).

[9] Zheng Lianmin, Chiang Wei-Lin, Sheng Ying, Zhuang Siyuan, Wu Zhanghao, Zhuang Yonghao, Lin Zi, Li Zhuohan, Li Dacheng, Xing Eric, Zhang Hao, Gonzalez Joseph E., and Stoica Ion. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS* (2023).

[10] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173.

[11] Alisia Lupidi, Carlos Gemmell, Nicola Cancedda, Jane Dwivedi-Yu, Jason Weston, Jakob Foerster, Roberta Raileanu, and Maria Lomeli. 2024. Source2synth: Synthetic data generation and curation grounded in real data sources. *arXiv preprint arXiv:2409.08239* (2024).

[12] Zakariae ALAMI MERROUNI, Bouchra FRIKH, and Brahim OUHBI. 2019. Toward contextual information retrieval: a review and trends. *Procedia computer science* 148 (2019), 191–200.

[13] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *NeurIPS* (2023).

[14] Mohammadreza Pourreza, Ruoxi Sun, Hailong Li, Lesly Miculicich, Tomas Pfister, and Sercan O Arik. 2024. Sql-gen: Bridging the dialect gap for text-to-sql via synthetic data and model merging. *arXiv preprint arXiv:2408.12733* (2024).

[15] Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. 2023. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL (extended). *arXiv preprint arXiv:2306.00739* (2023).

[16] Adlakha1 Vaibhav, BehnamGhader Parishad, Lu XingHan, Meade Nicholas, and Reddy Siva. 2023. Evaluating Correctness and Faithfulness of Instruction-Following Models for Question Answering. *CoRR* (2023).

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *36th Conference on Neural Information Processing Systems* (2022).

[18] Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. Synthesizing text-to-SQL data from weak and strong LLMs. *arXiv preprint arXiv:2408.03256* (2024).

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

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

SQL generation prompt

```
You are an expert data analyst specializing in transforming e-commerce sellers' business data requests into efficient and valid SQLite queries.
Here is the table schema you have access to:
<table_schema>
{table_schema}
</table_schema>
<instruction>
Think step-by-step to generate the SQL query:
- First provide your reasoning on how to generate the SQL query in <reasoning> tags.
    - Analyze the associated time in the seller question
        - Today is {today_date} (today_week_day) in YYYY-MM-DD format.
        - This week starts from {start_of_week} (Sunday) and ends on
{end_of_week} (Saturday).
    - Interpret 'last/past' periods as complete calendar units before the current date, unless explicitly stated otherwise.
...
</instruction>
Here are some examples:
<examples>
{examples}
</examples>
Here is the seller's question:
<query>
{query}
</query>
Strictly follow the above instructions and provide the SQL query that would retrieve the data. Do not provide explanations after the SQL code.
```

Figure 4: A simplified SQL generation prompt

A PROMPTS

Data generation prompt

```
You are an expert data analyst helping e-commerce sellers understand their business performance. Your task is to generate authentic seller questions that can be answered using the results of the following SQL query:
<table schema>
{table_schema}
</table schema>
<sql>
{query}
</sql>
To complete the task follow these steps:
1. Analyze the intent of the SQL query and document your analysis in <analysis> tags.
2. Consider an e-commerce seller's perspective and common business questions. Map the SQL query output to real business scenarios.
3. Based on your reasoning, generate 3-5 realistic seller questions that
- can be fully answered by the query results without additional analysis or reasoning steps.
- Reflect varying levels of data literacy.
- Include natural language variations and occasional typos to mirror authentic seller communication.
Put each question in a <question> tag.
```

Figure 3: A simplified data generation prompt