DBRAG: Multi-Table Retrieval-Augmented Generation for Complex Database Queries

Prince Larbi Ampofo **Ryoji Kubo** **Djellel Difallah**
NYU Abu Dhabi, UAE
{pla252,ryojikubo,djellel}@nyu.edu

Abstract

Recent advancements in large language models have introduced new capabilities for reasoning over structured data, particularly through program-aided tools that can analyze tables. However, many existing methods address single-table scenarios or assume that the relevant tables are already provided. In practice, users often issue complex data exploration queries over entire databases, where relevant information may be distributed across multiple relations. In this work, we introduce DBRAG, a retrieval-augmented generation framework tailored for multi-table question answering. DBRAG first retrieves candidate tables using an offline table index, enriches their summaries with query-relevant rows, and uses an LLM to rerank the candidates. A program-aided reasoner then selects the required tables and executes operations over their full contents, keeping the initial prompt context compact. Experiments on the Spider, GeoQuery, and ATIS datasets used in this study demonstrate improvements in table retrieval and multi-table question answering.

1 Introduction

Tabular data structures are a widely used data model for storing information in various domains. In real-world applications, these tables often contain complex, interrelated data with relational dependencies. When interpreted, relational data can lead to improved decision-making, enhanced operational efficiency, and the discovery of new hidden insights (Fey et al., 2024). Given the inherent complexity of structured query languages, such as SQL, using natural language can lower the barrier for data exploration (Li et al., 2024). Consequently, table question-answering (QA) has emerged as an important research area focused on developing systems that can understand natural language queries and extract relevant insights from tabular data (Jin et al., 2022; Lu et al., 2025).

Although significant progress has been made in single-table QA scenarios (Liu et al., 2022; Herzig et al., 2020; Nan et al., 2022; Zhu et al., 2021), the more challenging and practical task of multi-table QA remains relatively underexplored. Multi-table QA involves answering questions that require the integration of information from multiple tables, necessitating sophisticated reasoning and advanced data fusion techniques (Pal et al., 2023). This challenge is further compounded by the complex relationships between tables, such as identifying joinable keys or overlapping attributes, particularly when such relationships lack a well-structured explicit relational schema, as is often the case in web tables and data lakes (Cafarella et al., 2008; Chapman et al., 2020).

Many existing studies assume that the relevant tables are already provided (Chen et al., 2024b; Pal et al., 2023; Wang et al., 2024). In an open-domain table QA setting, the system must instead retrieve them from a larger corpus. This requires considering both query-table relevance and table-table relationships: a table with limited direct relevance may still be needed to connect complementary information (Chen et al., 2024a).

Large language models (LLMs) have demonstrated remarkable proficiency in various natural language processing tasks (Minaee et al., 2024). Their adaptation to QA via retrieval-augmented generation (RAG) has led to substantial performance gains (Lewis et al., 2020). In the RAG framework, relevant information is first efficiently retrieved from a pre-indexed large corpus and used to augment the query provided to the LLM, thereby mitigating issues such as hallucinations and outdated knowledge.

Despite the effectiveness of the RAG framework in single-table QA contexts (Chen et al., 2024b), its application to multi-table QA remains underexplored. The challenges of multi-table QA stem from several factors. First, multi-table QA requires

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 635–646
November 6-10, 2026 ©2026 Association for Computational Linguistics

the system to reason across multiple tables, often necessitating different types of joins and aggregations. Second, the sheer volume of tables in a typical database complicates the retrieval process, as the system must identify not only the main tables relevant to a query but also any auxiliary tables needed to connect and augment the data. Third, each table can vary significantly in size, both in the number of columns and rows. For instance, data warehouses often contain wide tables with billions of rows. These challenges, combined with the limitations of current LLM context windows, make the direct application of LLMs to multi-table QA non-trivial, thereby necessitating novel approaches.

To tackle the aforementioned challenges, we propose a novel methodology that extends the RAG framework to encompass the entire database. Our approach decomposes the problem of multi-table QA into two critical subtasks, *multi-table retrieval* and *multi-table reasoning*. ① In the retrieval step, we adopt a two-pronged approach to identify the relevant tables in a scalable manner. First, table search quickly selects a subset of tables relevant to the query from the entire database using a table index. Next, we enrich the tables of interest by augmenting them with query-relevant rows. The collected context is then fed into an LLM to select the most promising tables. ② In *multi-table reasoning*, we propose a novel reasoning protocol designed to answer the natural language question given the retrieved tables. Specifically, we design a multi-table-aware Chain-of-Thought prompting technique combined with a program-aided reasoner that executes operations over the retrieved tables and returns the final answer. Our experiments on multi-table QA datasets demonstrate that DBRAG not only improves the retrieval of relevant tables but also enhances the reasoning task, resulting in higher overall QA performance in the evaluated settings.

In summary, we make the following contributions:

- ⋆ We introduce a RAG approach for multi-table question answering in settings where the relevant tables must be retrieved from a large table corpus.
- ⋆ We propose DBRAG, a novel framework that decomposes multi-table QA into two subtasks: a two-stage retrieval strategy that leverages table indexing and context enrichment, and a multi-table reasoning protocol that integrates Chain-of-Thought prompting with program-aided execution.
- ⋆ We demonstrate through comprehensive experiments that DBRAG improves both table retrieval and reasoning performance, resulting in effective end-to-end QA execution.

Reproducibility Code, datasets, prompts, relevant parameters, and details of the experimental setup are available in the project repository: `https://github.com/colab-nyuad/DBRAG`

2 Related Work

In this section, we provide a brief overview of retrieval-augmented generation and then discuss the most pertinent works on table QA, highlighting both the inspiration and the novelty of our contributions.

LLMs and RAG. Large language models have revolutionized natural language processing with their advanced text comprehension and reasoning capabilities. However, they are not designed to handle domain-specific knowledge, rapidly changing information, private data, or long-tail entities. These limitations result in issues such as hallucinations (Huang et al., 2025), spurious explanations (Camburu, 2020), and inconsistencies (Elazar et al., 2021; Adiwardana et al., 2020). Recent techniques have improved reasoning through prompting methods, such as Chain-of-Thought (CoT) (Wei et al., 2022) for step-by-step reasoning, and Self-Consistency (Wang et al., 2023) for selecting the most consistent answer across sampled reasoning paths.

To overcome these challenges, recent research has focused on injecting domain-specific knowledge into language models by integrating external sources and leveraging domain corpora. Retrieval-augmented generation (RAG) has proven effective thanks to a two-stage retrieve-and-read framework, in which the retriever quickly extracts relevant data from large corpora (Gao et al., 2023; Fan et al., 2024). By incorporating accurate, up-to-date information, RAG can enhance the LLM's response pertinence, reduce hallucinations, and produce more reliable, contextually relevant outputs.

Language Models for Table Understanding. Numerous studies have focused on pretraining table-specific language models (LMs) for downstream tasks. Building on the success of masked language modeling, TAPAS (Herzig et al., 2020) introduces a BERT-like architecture enhanced with positional embeddings to encode table-specific

information. During pretraining, the model reconstructs masked table elements. In contrast to TAPAS's reconstruction approach, TAPEX (Liu et al., 2022) pretrains a language model to execute SQL operations on tables, motivated by the intuition that accurate SQL execution reflects robust table comprehension. To advance LM capabilities in table retrieval, Herzig et al. (2021) develop a dense retriever leveraging contrastive learning, pretraining, and negative sample mining. By employing TAPAS as the base encoder, they demonstrate that dense retrieval outperforms traditional methods like BM25 (Robertson and Zaragoza, 2009) in open-domain single-table retrieval.

LLMs for Single-table QA. Ye et al. (2023) develop a method that enables an LLM to construct sub-evidences and sub-questions to answer natural language questions over large tables. Their approach employs in-context learning by providing a small set of manually crafted examples to guide table deconstruction and reasoning. Chen et al. (2024b) introduce a RAG framework for single-table QA, retrieving relevant schema and cells to address scalability and context length limitations when feeding large tables to the LLM. Inspired by the effectiveness of reasoning chains in textual contexts, Wang et al. (2024) propose using an LLM agent to dynamically plan table operations based on prior steps, forming an iterative reasoning process. They demonstrate that enabling LLMs to generate a reasoning chain improves performance in single-table QA.

Multi-table QA. While single-table QA has garnered research attention, studies on multi-table QA remain limited. Pal et al. (2023) make a pioneering contribution by pretraining the TAPEX model on a multi-table dataset with SQL queries and tabular answers, followed by fine-tuning on natural language questions. However, their method assumes the relevant tables are already provided to the model and does not address scenarios where flattened table representations exceed the model's maximum sequence length. Chen et al. (2024a) introduce a method focused on multi-table retrieval, solving a mixed-integer program that jointly optimizes table-query relevance and table-table relevance. This approach adapts to databases lacking key-foreign-key constraints. It uses flattened table representations for coarse-grained retrieval, while downstream question answering relies on SQL generation and execution.

3 Methodology

3.1 Motivation and Overview

As mentioned in the introduction, directly presenting all tables to an LLM can be expensive and time-consuming. Additionally, methods that pass complete table information often run into context length limitations, especially with large tables. To make matters worse, users issuing ad hoc queries often do not know which tables are relevant or what the schema entails. To overcome these challenges and limit the number of tables presented to the LLM, we propose an approach that employs a multi-stage ranking mechanism for efficiency and mitigates context length issues by summarizing the selected tables. Additionally, we leverage a prompting technique that facilitates Chain-of-Thought-style multi-table reasoning, allowing the LLM to interact programmatically with the tables. Figure 1 shows an overview of our proposed framework, DBRAG. In the following, we begin by formally defining the setup, followed by descriptions of each component.

3.2 Problem Formulation and Requirements

We formally define the multi-table QA task as follows. Given a natural language query q and a database $\mathcal{D} = \{t_1, t_2, \ldots, t_N\}$, where each table t_i is described by a name t_i^{Name} and a schema $t_i^{\text{Schema}} = \{c_1, c_2, \ldots, c_n\}$, the objective is to produce an answer a that accurately addresses q.

A standard RAG framework for multi-table QA can be decomposed into two stages: ① *Multi-table QA Retrieval:* In this stage, the goal is to select a concise subset of tables, $\mathcal{R}_q \subset \mathcal{D}$, where $|\mathcal{R}_q| = M \ll N$. Ideally, this subset comprises only the tables necessary to answer the query. ② *Multi-table QA Reasoner:* Here, an LLM is provided with the query q, the retrieved tables $\mathcal{R}_q$, and an appropriate prompt. The LLM processes this input to generate the final answer.

To develop an effective solution, we consider the following requirements:

(R1) Retrieval Diversity: Ranking tables independently by query similarity can overlook complementary tables needed for joins. The reranking prompt therefore asks the LLM to consider direct relevance, joinability, and collective coverage; this is a selection criterion rather than a formal diversity guarantee.

(R2) Reasoner Context Size: Full table contents can exceed the LLM context limit. Compact summaries guide table selection and opera-

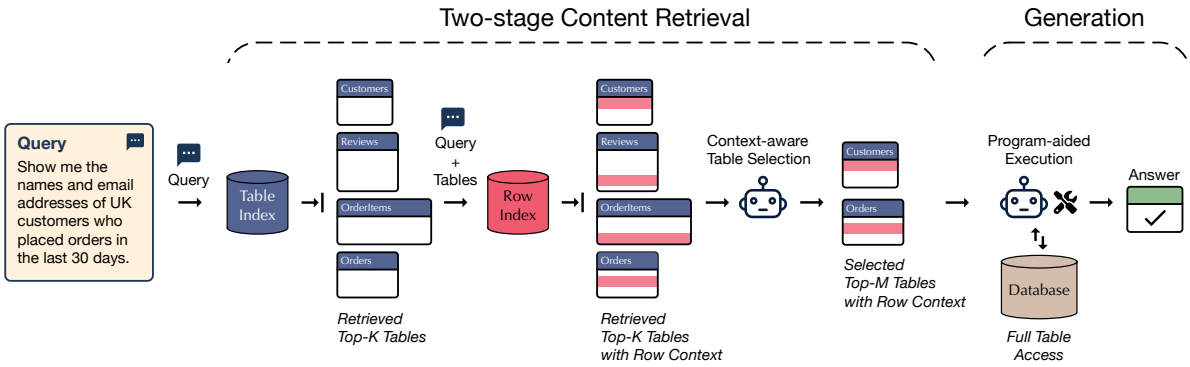


Figure 1: The DBRAG execution workflow. The Table Index retrieves top-K candidates; the Row Index supplies query-relevant context rows. An LLM reranks the candidates to select top-M tables. These tables are then passed to a program-aided reasoner, which uses a multi-table CoT prompt to generate the final answer.

tion construction, while execution tools access the full retrieved tables. Context-row selection therefore does not restrict answer computation to those rows.

3.3 Multi-table Retrieval

To enable scalable retrieval of multiple relevant tables while addressing (**R1**), we decompose the retriever into a two-step process. First, an efficient encoder is used to perform coarse-grained ranking, identifying a broad subset of top-K tables from the corpus. Then, an LLM is utilized to conduct fine-grained reranking of these top-K tables to ensure higher precision in relevance and to identify the top-M tables to form $\mathcal{R}_q$.

Table Index: Coarse-grained Ranking. We employ an efficient encoder f_{enc} to embed the coarse-grained summary of each table, which primarily consists of the table name, schema, and randomly sampled rows. For each table t_i, we form its *coarse-grained table summary* as follows:

$$s_i = \{t_i^{\text{Name}}, t_i^{\text{Schema}}, t_i^{\text{RandomRows}}\},$$

and compute its embedding $\mathbf{e}_i = f_{\text{enc}}(s_i)$. The query q is embedded as $\mathbf{q} = f_{\text{enc}}(q)$, and tables are ranked by the cosine similarity between $\mathbf{q}$ and $\mathbf{e}_i$. The top-K tables constitute the coarse-rank candidate set $\mathcal{C}_q$. Since these embeddings are independent of the query, they can be precomputed and stored in a *Table Index*.

Row Index: Context Retrieval. To further enrich the candidate set $\mathcal{C}_q$, we augment each table with query-relevant row information. This process involves building a row embedding index by encoding each row in the database and storing it in a precomputed *Row Index*. Specifically, for each row r with column names $c_1, c_2, \ldots, c_n$ and corresponding cell values $v_1, v_2, \ldots, v_n$, we form a row summary $s^r = \{c_1, v_1, c_2, v_2, \ldots, c_n, v_n\}$, and compute its embedding as $\mathbf{e}^r = f_{\text{enc}}(s^r)$.

Adding the row context is crucial in DBRAG, as it enhances the system's understanding of which tables should be combined to answer the query. Accordingly, for each table $t_i \in \mathcal{C}_q$, we query the Row Index to retrieve the rows most relevant to the query q. Rows are ranked within each candidate table by the cosine similarity between $\mathbf{q}$ and $\mathbf{e}^r$. The selected rows replace the random rows in the summary, retaining the table name and schema, to form the *fine-grained table summary*:

$$s_i^+ = \{t_i^{\text{Name}}, t_i^{\text{Schema}}, t_i^{\text{RelevantRows}}\},$$

where $t_i^{\text{RelevantRows}}$ denotes the set of L relevant rows retrieved from the index.

Note: Although the Row Index can be queried directly to obtain a coarse-grained ranking, we use the Table Index to first narrow the candidate set. This design also allows alternative row context selections, such as *Random Rows*, to be compared within the same retrieval framework (Section 4.1).

LLM-based Fine-grained Reranking. Next, we employ a language model $\mathcal{L}$ to perform fine-grained reranking of the candidate set $\mathcal{C}_q$. This step is crucial for addressing (**R1**), as encoder-based coarse-grained ranking scores each table independently against the query. This focus on individual relevance can limit the selection of tables providing complementary information necessary for answering the query through joins and unions. The LLM-based reranker leverages its reasoning capabilities to consider relationships among the candidate tables, enabling the identification of relevant tables.

Given the query q, a ranking prompt p_{rank}, and the fine-grained table summaries $\{s_i^+ \mid t_i \in \mathcal{C}_q\}$, the LLM selects the top-M tables (with $M < K$) best suited to answer the query, denoted by $\mathcal{R}_q$:

$$\mathcal{R}_q = \mathcal{L}\Big(q,\, p_{\text{rank}},\, \{s_i^+ \mid t_i \in \mathcal{C}_q\},\, M\Big).$$

This reduced subset of tables, which is semantically relevant to the query and potentially complementary in providing the answer, is passed to the reasoning stage, which we explore next.

3.4 Multi-table Reasoning

To support queries that may require combining data from a varying number of tables while limiting context size (**R2**), we design a new multi-table reasoner. This component dynamically identifies the subset of tables needed from the retrieved set and generates an execution plan using an LLM integrated with execution tools to run code over the retrieved tables.

Table Selection. First, the LLM is instructed to identify the specific subset of tables $\mathcal{R}_q^*$ from $\mathcal{R}_q$ that are necessary to answer the query. It dynamically evaluates the schema and context rows of each table using the set of table summaries:

$$S_q^{\mathcal{R}} = \{s_i^+ \mid t_i \in \mathcal{R}_q\}.$$

This step helps determine which tables can be effectively joined or unioned to construct the answer.

Program-aided CoT Reasoning. Next, given the selected tables $\mathcal{R}_q^*$, the LLM generates a sequence of programmatic actions. Inspired by Wang et al. (2024), we adopt a chain-of-thought prompting approach that decomposes the execution into a series of discrete data operations. To facilitate interaction with the tables, we integrate a programmatic function `tool` into the prompt, which enables the LLM to perform iterative data manipulation on the tables.

At this stage, the LLM combines table selection with execution guidance in a unified prompt p_{reason}:

$$ans = \mathcal{L}(q, p_{reason}, S_q^{\mathcal{R}}, \texttt{tool}).$$

During the execution, intermediate results are retained for use in subsequent operations, and once the chain of operations reaches an end, the final answer is returned to the user.

4 Experiments

We evaluate our method and baselines focusing on two key tasks: multi-table retrieval and multi-table reasoning.

Datasets. Existing tabular QA benchmarks have largely focused on either limited-domain scenarios or single-table contexts. To address this gap, MTQA (Pal et al., 2023) introduced a multi-table QA dataset derived from the cross-domain Spider dataset (Yu et al., 2018), which spans tables from 138 distinct domains. Their work adapts the original text-to-SQL pairs by executing the SQL queries to obtain tabular answers and excluding examples with errors or empty results. In our experiments, we also use the MTQA versions of GeoQuery (GeoQ; Zelle and Mooney, 1996) and ATIS (Price, 1990; Dahl et al., 1994), two smaller multi-table QA datasets built on databases of United States geography and flight booking systems, respectively. Appendix A details the methodology for constructing the database corpora and the queries. It also includes a summary of the key statistics of each dataset.

Experimental Setup and Parameters. In our method, DBRAG, we utilize `GPT-3.5-turbo`[1] as the backbone LLM. For efficient dense retrieval, we employ OpenAI's `text-embedding-3-small`[2] as the encoder f_{enc}. Specifically, we set the coarse-grained table budget to $K = 50$ and the fine-grained table budget to $M = 5$, a design choice that balances precision and recall across diverse user queries. Unless otherwise specified, the number of Row Context examples added to the table summaries is set to $L = 2$. Additionally, all the LLM-based experiments are run with a self-consistency parameter of 5, where we take the majority vote from five independent runs.

4.1 Multi-table Retrieval Performance

We begin by evaluating the table retrieval performance. In the following experiments, we focus on the larger Spider dataset, which presents a more challenging retrieval setup, and omit GeoQ and ATIS; their limited number of tables consistently results in perfect recall across all baselines.

[1]The specific version is `gpt-3.5-turbo-0125`. https://platform.openai.com/docs/models

[2]https://platform.openai.com/docs/guides/embeddings

Method	Queries				Average
	1-table (556)	2-table (366)	3-table (57)	4-table (6)	
Base Retrieval Methods					
Contriever	86.3	74.9	60.2	87.5	80.6
DTR	90.5	75.6	57.9	83.3	83.0
Coarse-rank	96.0	83.5	64.3	91.7	89.5
Reranking Methods					
JAR (Contriever)	91.4	90.4	78.4	**100**	90.3
JAR (DTR)	91.2	83.5	70.2	91.7	87.1
JAR (Coarse-rank)	96.8	90.3	73.7	91.7	93.0
DBRAG (Name and Schema)	89.4	77.5	68.4	83.3	83.7
DBRAG (Random Rows)	95.7	94.7	**87.7**	**100**	94.9
DBRAG (Relevant Rows)	**97.1**	**95.0**	87.1	**100**	**95.8**

Table 1: Multi-table retrieval performance (Recall@5 on Spider). Best result in **bold**, second-best result underlined. Query counts (by relevant tables) in parentheses.

Baselines. Following the work of Chen et al. (2024a) on table retrieval, we consider the following baseline methods:

- Contriever (Izacard et al., 2022): A text retrieval method that does not incorporate any table-specific model design. It ranks tables based on the cosine similarity between the embeddings of the question and the flattened tables.
- DTR (Herzig et al., 2021): A method specifically designed for table retrieval. TAPAS-large is fine-tuned under this framework, and the relevance of each table to the question is computed accordingly.
- Coarse-rank: We use our encoder-based coarse ranking (see Section 3.3) and select the top $K = M = 5$ tables, thereby bypassing the LLM-based fine-grained reranking step.
- JAR (Chen et al., 2024a): This method employs a base retrieval method and solves a mixed-integer program to identify relevant tables. We evaluate three versions: (i) JAR (Contriever), (ii) JAR (DTR), and (iii) JAR (Coarse-rank), which use the JAR algorithm to rerank the results of Contriever, DTR, and our coarse-rank approach, respectively.

DBRAG Ablations. We evaluate several variations of our method to analyze the contribution of each component.

- DBRAG (Name and Schema): Modifies the fine-grained summary so that it includes only the table name and the table schema.
- DBRAG (Random Rows): Modifies the fine-grained summary to include the table name, table schema, and random rows from the table. In this setup, the LLM-based reranking receives random rows instead of the relevant rows.
- DBRAG (Relevant Rows): This is our default setup described in Section 4.

Evaluation Metrics. To assess the quality of multi-table retrieval, we compare the retrieved tables against the relevant ground truth tables. Specifically, we use Recall@5 as the metric to evaluate retrieval performance. Recall@5 measures the proportion of required tables retrieved among the top five candidates, averaged across queries. For a fair evaluation, we retrieve the top $M = 5$ tables from each method as the set of retrieved tables.

Retrieval Results & Discussion. The results of multi-table retrieval are presented in Table 1, with detailed performance metrics for queries involving different numbers of relevant tables. It can be observed that as the number of relevant tables required to answer a question increases, retrieval performance generally decreases. However, queries with four relevant tables show improved performance, although this group contains only six queries. Our method, DBRAG, which uses fine-grained table summaries with relevant rows, achieves the best overall performance.

The comparison with Coarse-rank highlights the significance of LLM-based reranking in improving retriever performance. Additionally, the comparisons with DBRAG (Name and Schema) demonstrate the importance of incorporating row information alongside the table name and schema for accurate retrieval. Finally, the comparison with DBRAG (Random Rows) underscores the effectiveness of retrieving relevant rows in enhancing retrieval performance.

Query	Llama 3.3 (70B-Instruct)	Qwen2.5 (32B-Instruct)	GPT-3.5 (Turbo-0125)
1-table (556)	96.9	96.4	**97.1**
2-table (366)	88.7	84.6	**95.0**
3-table (57)	73.1	76.6	**87.1**
4-table (6)	91.6	83.3	**100**
Average	92.5	90.8	**95.8**

Table 2: Fine-grained reranking with different LLMs (Recall@5 on Spider; query counts in parentheses).

Fine-grained Reranking with Different LLMs. In this experiment, we evaluate the performance of DBRAG in fine-grained reranking using three different LLMs, while fixing the default parameters (see Section 4). We report Recall@5 on the Spider dataset. Table 2 shows that GPT-3.5 delivers the best results, especially as query complexity increases (i.e., when more tables are required). Although Llama 3.3 (Llama Team, 2024) and Qwen2.5 (Qwen Team, 2025) perform competitively on single-table queries, they do not keep up with multi-table scenarios. These results underscore the importance of advanced reasoning for effective fine-grained reranking for complex queries. These results show that the choice of reranking model affects performance under the fixed retrieval setup.

4.2 Multi-table Reasoning Performance

Next, we turn our attention to evaluating the end-to-end performance of multi-table QA encompassing retrieval and reasoning capabilities. In the following, we use the set of retrieved tables $\mathcal{R}_q$ obtained using DBRAG during the multi-table retrieval phase for the three datasets, and feed them to several baselines.

Baselines. We compare our method against the following baselines:

- ReadTable (Chen et al., 2024b): This is a common baseline that embeds the entire table in the prompt and asks the LLM to solve the problem using that data. However, if the table exceeds the model's context length, the question is marked as a failure.
- MTQA (Pal et al., 2023): A multi-table reasoning baseline that pretrains the TAPEX model on a multi-table dataset containing SQL queries and tabular answers, and then fine-tunes it on natural language questions.

Ablations. We compare our method with the following variations:

- DBRAG (Name and Schema): This setup restricts the table summaries to include only the table name and schema, without context rows.
- DBRAG (w/o Selection): Our approach that performs program-aided reasoning without the table-selection step (Section 3.4).
- DBRAG (Random Rows): Here, the relevant rows in the table summaries are replaced with random rows.

Evaluation Metrics. The performance of multi-table reasoning is measured by comparing the ground truth answer table derived from the answer SQL query with the table generated by the model. Following Pal et al. (2023), we employ metrics that determine if the returned table's features precisely match those of the ground truth answer.

The evaluation metrics assess table similarity at different levels of granularity. **Table EM** measures whether the returned table exactly matches the target table, requiring both row and column ordering to be identical; since many queries do not depend on ordering, additional metrics are used. **Row EM** assesses how many rows in the returned table exactly match those in the target table, ignoring row order but requiring exact value ordering within each row. Similarly, **Column EM** evaluates how many columns in the returned table match those in the target table, disregarding column order while still requiring exact value ordering within each column. Finally, **Cell EM** provides the most relaxed evaluation by measuring how many individual cells match exactly between the two tables.

For Row EM, Column EM, and Cell EM, we report the macro versions of these metrics to mitigate the bias introduced by queries returning large tables. Statistics about table size and distribution are provided in Appendix A.

Reasoning Results. Table 3 presents the multi-table reasoning results on Spider, GeoQ, and ATIS. We observe that DBRAG outperforms MTQA and ReadTable on most metrics, particularly on Spider, where many queries fail because their tables exceed the LLM's context limit when using ReadTable. In our evaluations, MTQA shows lower performance than DBRAG across all datasets, which may stem from its lack of an explicit table selection step when handling complex multi-table queries. DBRAG's improvements on GeoQ and ATIS further validate its efficacy on domain-specific tasks, although sim-

Dataset	Method	Table EM	Row EM			Column EM			Cell EM		
			P	R	F1	P	R	F1	P	R	F1
Spider	ReadTable	29.6	35.6	35.9	35.8	33.6	33.8	33.7	40.1	41.3	40.7
	MTQA	10.6	22.0	22.6	22.3	12.6	12.6	12.6	28.4	28.2	28.3
	DBRAG (w/o Selection)	43.2	50.3	49.5	49.9	49.7	50.4	50.1	59.7	59.4	59.6
	DBRAG (Name and Schema)	41.3	48.5	47.5	48.0	47.5	48.1	47.8	57.9	56.2	57.0
	DBRAG (Random Rows)	44.4	51.7	50.8	51.3	50.3	50.7	50.5	60.9	59.5	60.2
	DBRAG (Relevant Rows)	**44.9**	**52.7**	**51.9**	**52.3**	**51.3**	**51.8**	**51.5**	**62.6**	**61.6**	**62.1**
GeoQ	ReadTable	42.3	55.3	50.1	52.6	50.1	58.9	54.1	63.9	68.2	66.0
	MTQA	47.2	55.8	53.9	54.8	47.2	47.2	47.2	55.8	53.9	54.8
	DBRAG (w/o Selection)	43.5	57.5	53.5	55.4	50.2	58.9	54.2	66.0	70.5	68.2
	DBRAG (Name and Schema)	31.6	44.8	39.6	42.0	36.4	42.3	39.1	50.3	51.0	50.6
	DBRAG (Random Rows)	45.1	58.5	52.6	55.4	51.8	59.3	55.3	66.3	68.2	67.2
	DBRAG (Relevant Rows)	**47.8**	**60.3**	**55.0**	**57.5**	**54.6**	**62.9**	**58.4**	**68.7**	**71.2**	**69.9**
ATIS	ReadTable	80.2	85.2	86.1	85.6	81.9	84.9	83.4	87.0	91.9	89.4
	MTQA	75.6	82.4	90.4	86.2	75.7	76.7	76.2	82.5	91.6	86.8
	DBRAG (w/o Selection)	76.7	83.6	84.0	83.8	77.4	79.1	78.3	85.0	89.8	87.3
	DBRAG (Name and Schema)	65.1	71.6	73.3	72.4	66.9	68.6	67.7	73.5	77.9	75.7
	DBRAG (Random Rows)	80.2	87.7	87.5	87.6	81.0	82.6	81.8	88.7	90.1	89.4
	DBRAG (Relevant Rows)	**84.9**	**91.2**	**91.9**	**91.5**	**85.3**	**86.1**	**85.7**	**92.3**	**93.9**	**93.1**

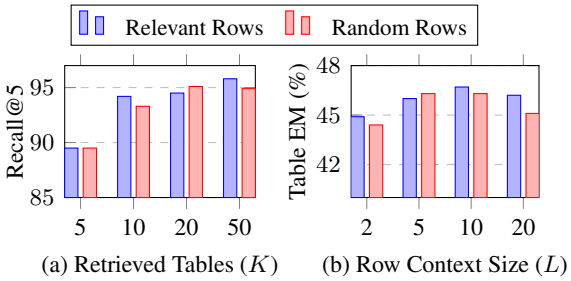
Table 3: Performance evaluation of multi-table reasoning. Best results in **bold** and second-best results underlined.

pler methods like ReadTable perform reasonably well in these setups.

Finally, the ablation studies confirm the importance of the main components of DBRAG. For instance, DBRAG outperforms its DBRAG (w/o Selection) alternative, and comparisons with the DBRAG (Name and Schema) and DBRAG (Random Rows) versions show that including relevant rows in table summaries improves answer accuracy.

4.3 Varying Table and Row Context Budget

Finally, we evaluate the impact of table and row context budgets on both table retrieval and reasoning performance on the Spider dataset. In the retrieval setup (Fig. 2(a)), we vary the coarse-grained ranking budget K, which determines the number of retrieved tables provided to the LLM for fine-grained reranking. Our results indicate that increasing K generally improves retrieval performance for both approaches, and importantly, the setup using relevant row context mostly outperforms that using random row context. For the reasoning setup (Fig. 2(b)), we vary the number of rows L included in the table summary. The experiment reveals that while increasing L initially enhances answer accuracy, performance reaches a plateau beyond which additional rows do not consistently improve performance. This aligns with the intuition that only a few rows are needed to grasp the structure and semantics of tabular information. These findings underscore the effectiveness of compact table summaries in the evaluated setting.



(a) Retrieved Tables (K) (b) Row Context Size (L)

Figure 2: DBRAG parameter variation on (a) retrieval and (b) reasoning on the Spider dataset.

5 Conclusions

We propose DBRAG, a retrieval-augmented generation framework tailored for multi-table question answering. DBRAG overcomes the challenges of complex multi-table relationships by decomposing the task into efficient table retrieval and multi-table reasoning. Our approach leverages a two-stage retrieval strategy, first retrieving candidate tables from the Table Index and then reranking them using query-relevant row context, followed by multi-table chain-of-thought prompting and program-aided execution for precise reasoning. Extensive experiments on multi-table QA datasets demonstrate improvements in both table retrieval and end-to-end QA performance. Our results validate DBRAG as an effective solution for complex, multi-table database question answering. Future work will explore further optimizations and extend support for heterogeneous data integration such as semi-structured data and knowledge graphs.

6 Limitations

Building and maintaining the Row Index can be computationally expensive for databases with billions of records. More efficient index construction and maintenance strategies, including sampling approaches potentially drawing on the frequency-based cell selection used by Chen et al. (2024b), could help address these challenges. We leave these issues, along with dynamic schema updates and production deployment, for future work.

While the queries in the datasets involve multiple tables, their answers can be obtained by executing SQL queries. In real-world scenarios, however, users may pose more complex queries, such as queries for open-ended exploration or interpretation of trends in the data, which may require analytical reasoning beyond SQL-based operations (Ma et al., 2023). The capability of DBRAG to handle such advanced queries remains unexplored in this paper.

Additionally, real-world large databases often present challenges such as noisy data, inconsistencies, and missing information (Hai et al., 2023). Our method has not been evaluated in these settings, where the model would need to navigate and address potential flaws in the data itself.

Finally, our comparisons cover a limited set of baselines for multi-table retrieval and reasoning. We do not include comparisons with advanced solvers which could provide further insights into the performance of our approach.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549.

Oana-Maria Camburu. 2020. *Explaining deep neural networks*. Ph.D. thesis, University of Oxford, UK.

Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: A survey. *The VLDB Journal*, 29(1):251–272.

Peter Baile Chen, Yi Zhang, and Dan Roth. 2024a. Is table retrieval a solved problem? Exploring join-aware multi-table retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2687–2699. Association for Computational Linguistics.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024b. TableRAG: Million-token table understanding with language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In *KDD*, pages 6491–6501. ACM.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. 2024. Position: Relational deep learning - graph representation learning on relational databases. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pages 13592–13607. PMLR / OpenReview.net.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data lakes: A survey of functions and systems. *IEEE Trans. on Knowl. and Data Eng.*, 35(12):12571–12590.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 512–519, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via

pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: Recent advances. *Preprint*, arXiv:2207.05270.

Patrick S. H. 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 33*, pages 9459–9474. Curran Associates, Inc.

Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024. The dawn of natural language to SQL: Are we fully ready? *Proceedings of the VLDB Endowment*, 17(11):3318–3331.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

Llama Team. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Weizheng Lu, Jing Zhang, Ju Fan, Zihao Fu, Yueguo Chen, and Xiaoyong Du. 2025. Large language model for table processing: a survey. *Frontiers of Computer Science*, 19(2):192350.

Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. InsightPilot: An LLM-empowered automated data exploration system. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 346–352, Singapore. Association for Computational Linguistics.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *Preprint*, arXiv:2402.06196.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryscinski, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir R. Radev. 2022. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Vaishali Pal, Andrew Yates, Evangelos Kanoulas, and Maarten de Rijke. 2023. MultiTabQA: Generating tabular answers for multi-table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6322–6334, Toronto, Canada. Association for Computational Linguistics.

P. J. Price. 1990. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Qwen Team. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*. OpenReview.net.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *SIGIR*, pages 174–184, New York, NY, USA. Association for Computing Machinery.

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.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Datasets

Pal et al. (2023) introduced a multi-table QA dataset based on the cross-domain Spider dataset, which comprises tables from 138 domains (Yu et al., 2018). Their work adapts the original text-to-SQL pairs by executing the SQL queries to obtain tabular answers and excluding examples with errors or empty results. Each sample includes a natural language question paired with its corresponding SQL query answer. Furthermore, we utilize GeoQuery (GeoQ; Zelle and Mooney, 1996) and ATIS (Price, 1990; Dahl et al., 1994), which are multi-table QA datasets built on a database of United States geography and a relational database for flight booking systems, respectively. In this work, we adopt Pal et al. (2023)'s version of these datasets, which uses Iyer et al. (2017)'s text-to-SQL samples for GeoQ and ATIS. In this version, the ATIS and GeoQ datasets include many unintelligible questions provided as disconnected sentences (e.g., "list la"). Consequently, we synthesized natural queries for the ATIS and GeoQ datasets based on the corresponding SQL queries. These queries, along with the code used for their generation, are available in the project repository. Table 4 summarizes the statistics for each dataset.

Table 5 presents an overview of additional dataset statistics of the three datasets focusing on their content. For each dataset, we report the total and unique number of questions, details on table joins (ranging from single-table to five-table questions), and summary statistics on table dimensions including the average, minimum, and maximum number of rows and columns.

	ATIS	GeoQ	Spider
# train+valid samples	429	579	6715
# test samples	86	253	985
# tables	11	7	729

Table 4: Dataset Statistics

Metric	ATIS	GeoQ	Spider
Total questions	515	832	7700
Total unique questions	422	737	7656
Total unique tables (name)	11	7	571
Total unique tables (content)	11	7	729
1-table questions	190	717	4387
2-table questions	180	110	2592
3-table questions	79	5	655
4-table questions	66	0	56
5-table questions	0	0	10
Avg. no. of rows	82.29	175.36	1165.35
Min. no. of rows	14	32	2
Max. no. of rows	374	386	510437
Avg. no. of columns	5.3	4.4	5.99
Min. no. of columns	2	2	2
Max. no. of columns	15	6	48

Table 5: Additional Dataset Statistics

B Evaluation Metrics Details

This section complements the reasoning metrics described in Section 4.2. For Row EM, Column EM, and Cell EM, *precision* is the fraction of features (rows, columns, or cells) in the returned table that exactly match a feature in the target table, and *recall* is the fraction of features in the target table that are exactly matched in the returned table; the F1-score is their harmonic mean. Pal et al. (2023) compute these scores by pooling features across all returned and target tables in the test set (i.e., micro-averaging), so a few queries with very large returned or target tables can dominate the results. We instead compute precision and recall for each query, average them across queries (i.e., macro-averaging) so that every query contributes equally, and report F1 as the harmonic mean of the macro-averaged precision and recall.

C DBRAG Prompts

Reranking Prompt

```
Given a question and a set of candidate tables, rank tables by relevance to the given question while
also considering their potential to be joined with others to enhance the completeness of the answer.
Each table has a table_id, table_name, columns, and some rows (optional).

- Question -
{question}

- Candidate Tables -
{candidate_tables}

- Considerations for Ranking -
1. Direct Relevance: Does the table contain column names or row values (if available) that strongly
align with the terms/entities in the question?
2. Joinability: Can the table be meaningfully joined with others based on shared column names or
overlapping row values (if available)?
3. Coverage & Completeness: Does the table alone or when joined with others provide a comprehensive
answer to the question?

- Requirements -
1. Always return exactly {M} unique table_ids
2. Order from most to least relevant
3. Consider both standalone value and join potential
4. Prioritize direct relevance over theoretical join possibilities

- Response Format -
[table_id1, table_id2, table_id3, table_id4, table_id5]
```

Table 6: Fine-grained reranking prompt, including relevance, joinability, and coverage criteria.

Reason Prompt

```
You are working with {retrieved_dfs} dataframes in Python named df1, df2, and so on. Think step by
step, utilizing the tools provided to answer the given question by executing a series of dataframe
manipulation operations. Always start by identifying the appropriate dataframes. Then, create a chain
of actions and execute it on the appropriate dataframes with the execute_dataframe_code tool. Finally,
use the final_structured_output tool in your last call to return the final dataframe with sql-style
column names.

Example chain input format:
<BEGIN> -> action1 -> ... action2 -> action3 -> <END>

Always continue the chain with the above format for example:
df_dic['df1'].merge(df_dic['df2'], on='personId') -> inter.mean(axis=1) -> <END>

Always refer to your dataframes as df_dic[dataframe_name].
For example, instead of writing df1.groupby(...) you should write df_dic['df1'].groupby(...).
If you continue from the current state of the dataframe, refer to it as inter.

- Example 1 -
Question: List the creation year, name and budget of each department.
Chain Logic: We first need to select the appropriate dataframe(s), join dataframe(s) if necessary, then
select the creation year, name and budget columns.

- Example 2 -
Question : What are the names of the states where at least 3 heads were born?
Chain Logic: We first need to select the appropriate dataframe(s), join dataframe(s) if necessary, then
group by the state names, count the number of heads born in each state and filter for states with at
least 3 heads born.

Use the following tools: execute_dataframe_code, final_structured_output

{finegrained_table_summaries}

Always think step by step. Begin!
```

Table 7: Program-aided multi-table reasoning prompt using Python dataframe tools.