PARTAB: Partition-Aware Reasoning with Structured Evidence for Scalable Table Understanding

Md Mahadi Hasan Nahid
University of Alberta
mnahid@ualberta.ca

Davood Rafiei
University of Alberta
drafiei@ualberta.ca

Abstract

Large Language Models (LLMs) have shown strong capabilities in table reasoning, but their effectiveness degrades as tables grow in size and complexity due to irrelevant context and difficulty localizing the evidence required for reasoning. Existing approaches typically reason over either the full table or a single reduced view, which can still obscure important row-column relationships. We introduce **PARTAB** (**P**artition-**A**ware **R**easoning over **Tab**les), a framework that constructs a structured evidence interface between the LLM and the table. PARTAB represents query-relevant evidence as semantically coherent, row-linked table regions and performs hierarchical selection over column groups and row-level partitions before composing the selected evidence for answer generation. We evaluate PARTAB on multiple table reasoning benchmarks, covering question answering, fact verification, and numerical reasoning. PARTAB consistently improves over full-table prompting and several recent table reasoning methods, achieving strong performance on WikiTableQuestions and TabFact while remaining competitive on numerical reasoning. Additional analyses show that semantic partitioning and targeted evidence selection improve evidence localization, substantially reduce the reasoning context, and provide larger benefits on complex tables. These results demonstrate the value of structured, partition-aware evidence construction for scalable table reasoning.

1 Introduction

Large Language Models (LLMs) have demonstrated strong performance on table reasoning tasks such as question answering and fact verification by operating directly over serialized tabular inputs (Cheng et al., 2023; Liu et al., 2022). However, their effectiveness degrades significantly as tables grow in size and complexity (Wang et al., 2024). This raises a central research question: *how can LLMs scale to large and noisy tables while accurately localizing and reasoning over the relevant evidence?*

A key challenge is that, in many table reasoning tasks, the required evidence is localized to a small set of regions, consisting of specific combinations of rows and columns, while existing approaches require reasoning over the entire table (Nahid and Rafiei, 2024b). As table size increases, irrelevant content introduces noise, leading to attention dilution, lost-in-the-middle effects, and incorrect row–column binding (Wang et al., 2025a). As a result, models often fail to identify the correct evidence or produce inconsistent intermediate reasoning steps, even when the necessary information is present (Li et al., 2025; Wu et al., 2025).

Existing approaches to LLM-based table reasoning largely follow two paradigms. The first treats reasoning as executable program generation, most commonly through Text-to-SQL translation (Yu et al., 2018; Li et al., 2023; Cheng et al., 2023). By delegating computation to database engines, these systems achieve strong performance on structured and aggregation-heavy queries (e.g., counting, sorting, and numerical comparison) (Nahid and Rafiei, 2024a). However, symbolic execution relies on constructing precise logical predicates and therefore struggles when tables contain noisy textual descriptions, free-form attributes, or semi-structured content commonly observed in real-world data (Abhyankar et al., 2025).

The second paradigm performs direct reasoning by prompting LLMs with serialized tables (Herzig et al., 2020; Wang et al., 2024). While this approach offers flexibility under ambiguity and noise, its performance degrades as table size increases, making it difficult for LLMs to capture row–column relationships and identify the correct cell values (Ye et al., 2023; Chen, 2023).

Retrieval and pruning strategies partially mitigate this issue by selecting subsets of rows or

columns (Abhyankar et al., 2025; Wang et al., 2025b), but they still require reasoning within a single monolithic view and remain vulnerable to incomplete or fragmented evidence selection (Lin et al., 2023; Nahid and Rafiei, 2024b).

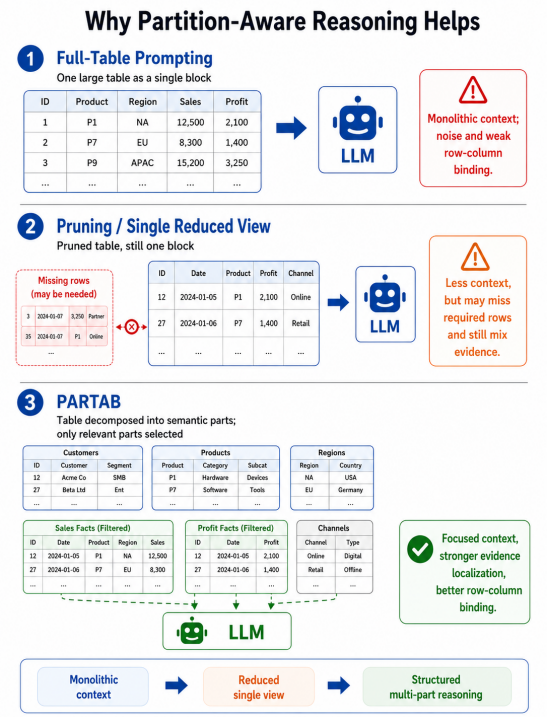

Figure 1: Comparison of reasoning behaviors under different table reasoning paradigms. **(I) Full-table prompting** exposes the model to the entire table, increasing irrelevant context and making evidence localization more difficult in long and wide tables. **(II) Single-view pruning** reduces context size but may remove required evidence, leading to incomplete reasoning and incorrect aggregation. *All data shown are synthetic and for illustration only.*

These limitations suggest that the core issue is not only *what* information is retrieved, but *how* reasoning is structured. We argue that table reasoning should be reframed as a **partitioned reasoning problem**, where reasoning is performed over semantically coherent subsets of the table rather than a single global view (see Figure 1). This perspective leads to three key research questions: (1) how can we provide LLMs with only the relevant evidences (**table parts**) of a table for a given query? (2) how can we construct partitions that preserve semantic coherence and relational structure? and (3) how can we select the minimal set of partitions that jointly contain sufficient evidence for correct reasoning?

To address these questions, we introduce **PARTAB** *(**Par**tition-**A**ware **R**easoning over **Tab**les)*, a framework that introduces a structured evidence interface between the LLM and the table. Rather than collapsing selected evidence into a single reduced table, PARTAB constructs a query-conditioned evidence state consisting of independently addressable, row-linked table regions. The LLM then reasons over this structured state by composing evidence across the selected regions. In this way, partitioning serves not merely as a preprocessing operation for context reduction, but as an explicit mechanism for organizing and controlling table reasoning.

We evaluate PARTAB on multiple challenging table reasoning benchmarks, including question answering and fact verification tasks. Results demonstrate consistent gains over full-table prompting and pruning-based baselines, outperforming several recent methods, particularly on large and noisy tables. Further analysis demonstrate that partition-aware reasoning improves evidence localization and yields more interpretable intermediate reasoning traces.

Our contributions are threefold. (1) We introduce **PARTAB**, which constructs a query-conditioned structured evidence state consisting of semantically organized, independently addressable, and row-linked table regions. (2) We introduce a modular pipeline that combines semantic grouping, group selection, and row-linked part selection to improve alignment and reduce noise during reasoning. (3) We demonstrate that the benefits of partition-aware reasoning increase with table size and structural complexity, supporting its use as a scalable reasoning interface for large and noisy tables.

2 Methodology

We implement **PARTAB** as a modular partition-aware reasoning pipeline for table question answering (see **Figure 2**). Given a question q and a table T, the system produces an answer a through four stages: (1) *Question Analyzer*, (2) *Partition Builder*, (3) *Group and Part Selector*, (4) *Answer Executor*.

The key idea is to replace single-view reasoning over the entire table with **localized reasoning over structured table parts**. For example, a question about "the highest population city" should only involve the columns related to city and population, and a small subset of relevant rows, rather than the full table. **Algorithm 1** summarizes the pipeline.

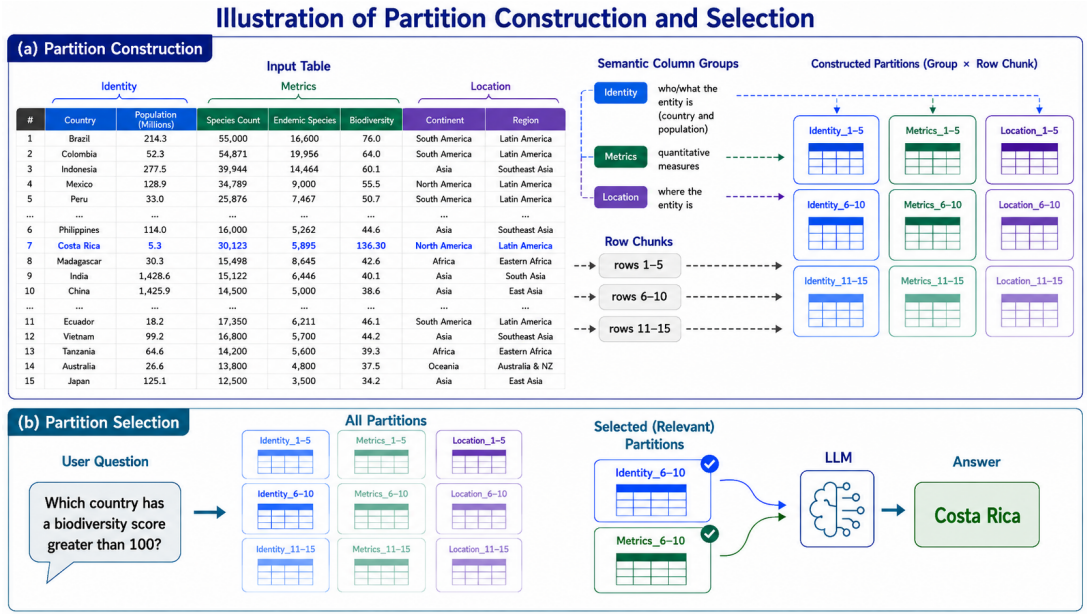

Figure 2: Overview of PARTAB. Given a question and table, PARTAB constructs semantically coherent table partitions and performs localized understanding over selected partitions enabling reliable reasoning over large and noisy tables. *All data shown are synthetic and for illustration only.*

Algorithm 1 PARTAB: Partition-Aware Table Reasoning

Require: Question q and table T
Ensure: Predicted answer a
$z_q \leftarrow \textsc{Analyze}(q)$
$\Pi(T) \leftarrow \textsc{BuildPartitions}(T, q)$
$G_q \leftarrow \textsc{SelectGroups}(q, z_q, \Pi(T))$
$\mathcal{P}_q \leftarrow \textsc{SelectParts}(q, G_q, \Pi(T))$
if $\mathcal{P}_q = Null$ **then**
  $\mathcal{P}_q \leftarrow \textsc{BasicSelect}(G_q, \Pi(T))$
end if
$a \leftarrow \textsc{Execute}(q, \mathcal{P}_q)$
return a

2.1 Table Preprocessing

We first normalize the input table to ensure consistent schema representation. Column names are cleaned by lowercasing, removing or replacing special characters (e.g., replacing whitespace with underscores), and standardizing formatting. We then insert a `row_id` column when absent:

$$T' = \texttt{EnsureRowID}(\texttt{CleanColumns}(T)).$$

The `row_id` serves as a stable key for linking information across partitions in later stages.

2.2 Question Analyzer

In this stage, we aim to identify the reasoning structure of the question to guide downstream partitioning and selection. Given a question q, an LLM-based analyzer produces a structured representation:

$$z_q = \texttt{Analyze}(q),$$

which encodes attributes such as question type (lookup, aggregation, comparison, etc.), required data processing operations (filtering, sorting, aggregation), and expected answer type.

The intuition is that different questions require different table regions. For example, an aggregation query focuses on filtered rows that should not be part of the aggregation, while a comparison query requires multiple candidate rows. The analyzer provides lightweight signals that guide both partition selection and reasoning.

2.3 Partition Builder

This stage decomposes the table into semantically coherent parts to enable localized reasoning. Rather than operating on a single serialized table, PARTAB constructs multiple structured views by partitioning both columns and rows.

Semantic Column Grouping. Columns are grouped into semantically coherent sets:

$$G = \{g_1, g_2, \ldots, g_m\}.$$

Each non-key column must belong to exactly one group, while `row_id` acts as a universal key. Every column that is not assigned by the mode is placed into a fallback `other` group. For example, in a table of countries, columns such as `population`, `area`, and `gdp` may form a "statistics" group, while `name` and `region` form a "metadata" group.

Chunked Part Construction. Each semantic group g_i is further partitioned along the row dimension into fixed-size chunks of size c, where each chunk is identified by the corresponding range of `row_id`s. Formally,

$$P_{i,j} = T'[r_{j:j+c-1}, \{\texttt{row\_id}\} \cup g_i],$$

where i indexes the semantic group and j indexes the row chunk. Each part is stored together with its associated metadata, including the group name, row range, column set, and the corresponding data content. In our implementation, we use a default chunk size of $c = 5$. This stage creates multiple localized views of the table. Unlike pruning, which produces a single reduced table, PARTAB produces a *set of complementary parts*, enabling flexible evidence composition.

2.4 Group and Part Selector

This stage identifies the minimal set of table regions required for answering the question.

Group Selection. We first select relevant column groups:

$$G_q \subseteq G.$$

The selector is implemented as an LLM module that receives: (1) the question, (2) the question analysis output, and (3) the list of available column groups. It is instructed to select only the groups necessary for answering the question. This step reduces the search space before fine-grained part selection and acts as a schema-level routing mechanism. For example, a population-related query should ignore groups such as timestamps or descriptions.

Part Selection. We then select specific row chunks within the chosen groups:

$$\Pi_q(T) = \{P \in \Pi(T) \mid \texttt{group}(P) \in G_q\}.$$

We consider three selection strategies:

Basic Selection. This mode selects all candidate parts:

$$\mathcal{P}_q = \Pi_q(T),$$

and it serves as the simplest partition-aware baseline.

Similarity-Based Selection. This strategy ranks parts using TF–IDF similarity with the question and select top-k:

$$\mathcal{P}_q = \texttt{TopK}_{\text{TF-IDF}}(q, \Pi_q(T)).$$

LLM-Based Selection. In this mode, an LLM is used to directly predict the minimal set of relevant parts.

This stage ensures that reasoning operates over a *focused yet sufficient* subset of the table, reducing noise while preserving necessary evidence.

2.5 Answer Executor

This stage performs reasoning over selected parts to produce the final answer. Here selected parts are serialized into a compact context:

$$C_q = \texttt{Serialize}(\mathcal{P}_q).$$

$$a = \texttt{Execute}(q, C_q).$$

Each part is rendered with metadata (group, row range, columns) and a preview of rows. The prompt explicitly instructs the model to answer using only the provided table parts, treat `row_id` as the linking key across parts, avoid relying on outside knowledge, and return only a short final answer. If the answer cannot be determined from the selected parts, the model is instructed to return a fixed abstention response.

3 Evaluation and Experimental Setup

Datasets. We evaluate PARTAB on three benchmarks covering both table question answering and fact verification tasks. **WikiTableQuestions** (Pasupat and Liang, 2015) evaluates compositional reasoning over Wikipedia tables, and we report Exact Match (EM) accuracy following standard protocols. **TabFact** (Chen et al., 2020) focuses on table-based fact verification, where the task is to determine whether a statement is entailed or refuted by a table; we report classification accuracy (Acc). **TableBench** (Wu et al., 2025) evaluates reasoning over large and complex tables and includes two subsets: **Numerical Reasoning (NR)** and **Fact Checking (FC)**. We report Exact Match (EM) for both NR and FC tasks.

Implementation. We adopt a modular design in PARTAB, where different components are instantiated with models suited to their functional roles. We use the lightweight and cost-efficient `GPT-5-nano` for partition construction to limit inference cost, and `GPT-4o-mini` for answer execution to enable fair comparison with prior work to demonstrate the effect of partition-aware reasoning. To assess robustness across model families, we evaluate our method using two further

Method	WikiTableQuestions EM	TabFact Acc	TableBench NR (EM)	TableBench FC (EM)
BINDER (Cheng et al., 2023)	38.03	67.50	-	-
DATER (Ye et al., 2023)	52.00	74.70	-	-
ReAcTable (Zhang et al., 2024b)	57.09	-	-	-
End-to-End Prompting	59.43	73.22	59.50	66.66
Chain-of-Thought (CoT)	64.31	75.99	-	-
Chain-of-Table (Wang et al., 2024)	68.53	85.09	-	-
TabSQLify (Nahid and Rafiei, 2024b)	68.74	78.30	66.25	70.83
Rank_Agent (Wu et al., 2025)	-	-	**78.34**	**87.50**
PoTable (Mao et al., 2026)	64.73	88.93	-	-
H-Star (Abhyankar et al., 2025)	74.93	89.42	69.52	79.17
TableMaster (Cao and Liu, 2026)	78.13	90.12	-	-
PARTAB (Ours)	**79.31**	**90.48**	70.33	82.71

Table 1: Downstream performance on table reasoning benchmarks. PARTAB achieves strong results across question answering, fact verification, and numerical reasoning, outperforming several recent baselines on WikiTableQuestions and TabFact while remaining competitive on TableBench.

LLM backbones, `Gemini-2.5-Flash-Lite` and `DeepSeek-v4-Flash`. All answer-generation experiments use zero-shot prompting with concise instructions, without few-shot demonstrations, chain-of-thought, or program-of-thought prompting. Detailed prompts and illustrative examples are provided in **Appendix F**.

The default configuration sets the row chunk size to $c = 5$ and uses top-$k = 6$ for similarity-based retrieval, where similarity is computed using a basic TF-IDF representation. [1]

4 Results and Discussion

We analyze PARTAB through three empirical questions: (1) whether gains come from structured evidence construction rather than additional inference budget; (2) the contribution of semantic partitioning and hierarchical selection; and (3) how the benefit of partition-aware reasoning changes with table complexity.

Downstream Results. Table 1 shows that PARTAB performs strongly across diverse table reasoning tasks. It achieves the best result on WikiTableQuestions (79.31 EM), surpassing TableMaster (78.13) and earlier methods, and reaches 90.48 accuracy on TabFact, indicating effective evidence localization and semantic grounding. On TableBench, PARTAB remains competitive with 70.33 EM on numerical reasoning and 82.71 accuracy on fact checking. Although specialized systems such as Rank_Agent perform better on aggregation-heavy tasks, PARTAB achieves strong results without full-table execution. Overall, PARTAB provides a balanced and robust approach across semantic and numerical reasoning by improving accuracy through localized reasoning over partitioned table regions.

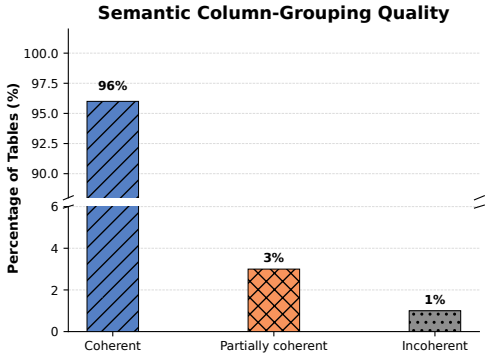

Figure 3: Partition construction performance. Manual evaluation of semantic column-grouping quality over 100 tables.The generated groups are fully coherent in 96% of the evaluated cases.

Partition Construction Performance. We evaluate the quality of partition construction by assessing the semantic coherence of the generated column groups. Since column grouping is produced by an LLM without a unique ground-truth decomposition, we conduct a manual evaluation to determine whether the induced groups align with meaningful semantic structures in the table. As shown in Figure 3, 96% of the groups are fully coherent, indicating that the model reliably captures high-level schema structure. The remaining 4% of errors mainly involve semantically related columns being separated or unrelated columns being grouped together. Overall, these results show that PARTAB constructs robust and meaningful

[1] We will release the code and data upon acceptance of the paper.

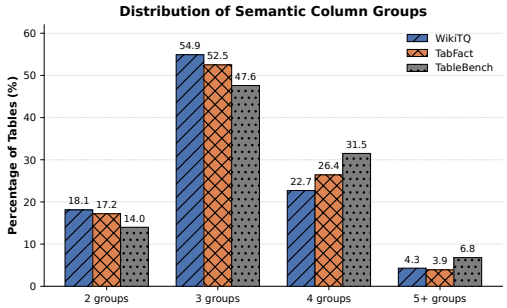

Figure 4: Distribution of the number of semantic column groups generated per table. Most tables are decomposed into three or four groups across all evaluated datasets.

partitions for downstream reasoning.

Selection Strategy	WikiTQ	TabFact	TableBench	
	EM	Acc	NR (EM)	FC (EM)
Basic Selection	74.71	87.88	64.43	79.26
Similarity-based Selection	67.25	85.87	58.69	59.25
LLM-based Selection	**79.31**	**90.48**	**70.33**	**82.71**

Table 2: Comparison of part-selection strategies across table reasoning benchmarks. The best result in each column is shown in bold.

Partition Selection Performance. Table 2 shows that effective part selection is critical for partition-aware reasoning. LLM-based selection achieves the best performance across all benchmarks, including WikiTableQuestions (79.31 EM), TabFact (90.48 Acc), and TableBench (70.33 / 82.71), demonstrating the importance of semantic understanding in identifying minimal yet sufficient partitions.

In contrast, similarity-based selection performs the worst, indicating that surface-level similarity is insufficient for capturing true relevance, particularly for compositional queries. Basic selection performs better but still includes unnecessary context, which can reduce reasoning precision compared to more targeted selection.

Distribution of Semantic Column Groups. The Column Grouping prompt provides a soft preference for constructing between two and five semantic groups. This is not a hard constraint; the model may produce additional groups when required by a more complex schema.

Figure 4 shows that most tables are represented by three or four semantic groups. For WikiTQ, 77.6% of tables contain three or four groups; the corresponding proportions are 78.9% for TabFact and 79.1% for TableBench. Six groups are generated in at most 0.2% of examples.

These results indicate that the grouping procedure produces compact and interpretable schema organizations rather than excessively fragmenting thetable. At the same time, the model retains the ability to generate more fine-grained groupings for unusually wide or semantically diverse schemas.

LLM Backbone	WikiTQ	TabFact	TableBench	
	EM	Acc	NR (EM)	FC (EM)
`GPT-4o-mini`	**79.31**	90.48	**70.33**	**82.71**
`Gemini-2.5-Flash-Lite`	71.23	83.56	61.29	64.53
`DeepSeek-v4-Flash`	78.30	**91.70**	70.23	75.34

Table 3: Performance of PARTAB across different answer-execution backbones.

Robustness Across LLM Backbones. We evaluate PARTAB using three answer-execution backbones with different capabilities and model families. Table 3 shows that the framework remains effective across all three models. DeepSeek-v4-Flash closely matches GPT-4o-mini on WikiTQ and TableBench-NR and obtains the highest TabFact accuracy. Although Gemini-2.5-Flash-Lite produces lower absolute performance, it remains capable of following the same partition-aware reasoning pipeline.

These results indicate that PARTAB is not tied to a single proprietary model. Nevertheless, the performance variation across backbones shows that final accuracy still depends on the executor's ability to follow structured instructions, maintain row–column alignment, perform numerical operations, and reason faithfully over selected table parts.

Frontier Models on Large and Complex Tables. To examine whether partition-aware reasoning remains useful when the executor has stronger reasoning capabilities and a larger context window, we construct a hard subset for each benchmark. An example is included in the hard subset when its table contains more than 45 rows or more than 10 columns. These examples are particularly challenging due to attention dilution, increased irrelevant context, and greater difficulty in evidence localization.

Table 4 compares PARTAB with full-table prompting using the same backbone. PARTAB improves performance for every backbone–dataset combination. The average improvement is $+18.96$ points for GPT-4o-mini, $+9.03$ points for GPT-5-mini, and $+12.65$ points for DeepSeek-v4-Pro.

The largest improvements are observed on TabFact-Hard, where PARTAB improves accuracy

LLM Backbone	Method	WikiTQ-Hard		TabFact-Hard		TB-NR-Hard	
		EM	Δ	EM	Δ	EM	Δ
`GPT-4o-mini`	Full Table	45.80	–	55.32	–	40.47	–
	PARTAB (Ours)	**55.72**	**+9.92**	**80.85**	**+25.53**	**61.90**	**+21.43**
`GPT-5-mini`	Full Table	60.30	–	57.14	–	57.45	–
	PARTAB (Ours)	**61.83**	**+1.53**	**78.26**	**+21.12**	**61.90**	**+4.45**
`DeepSeek-v4-Pro`	Full Table	58.01	–	55.32	–	54.76	–
	PARTAB (Ours)	**59.54**	**+1.53**	**89.36**	**+34.04**	**57.14**	**+2.38**

Table 4: Performance of full-table prompting and PARTAB on hard subsets containing large or wide tables. WikiTQ-Hard and TB-NR-Hard are evaluated using Exact Match (EM), while TabFact-Hard is evaluated using accuracy. Each Δ denotes the absolute improvement of PARTAB over full-table prompting with the same backbone.

by 25.53, 21.12, and 34.04 points for GPT-4o-mini, GPT-5-mini, and DeepSeek-v4-Pro, respectively. The gains on WikiTQ-Hard and TableBench-NR-Hard become smaller for stronger models, suggesting that frontier models can partially compensate for long-context difficulty on some compositional and numerical questions. However, the consistent improvements show that stronger reasoning capabilities and larger context windows do not fully eliminate challenges in evidence localization and reasoning over large tables.

Partition Structure and Context Reduction. We analyze the structures generated by the Partition Builder across the three evaluation datasets. Table 5 reports the average number of semantic column groups, columns per group, candidate partitions, and selected partitions.

Across datasets, each table is decomposed into approximately 3.1–3.3 semantic groups, with roughly two non-key columns in each group. Although partition construction creates between 9.84 and 17.56 candidate parts per table, the selector retains only 2.44–3.88 parts on average.

Relative to the candidate partition set, this corresponds to a reduction of 77.9% on WikiTQ, 75.2% on TabFact, and 75.2% on TableBench. Therefore, PARTAB exposes the final answer executor to only a small fraction of the available table regions.

Dataset	Partition Construction			Partition Selection	
	Groups / Table	Columns / Group	Parts / Table	Selected Parts	Reduction (%)
WikiTQ	3.13	2.10	17.56	3.88	77.9
TabFact	3.17	2.11	9.84	2.44	75.2
TableBench	3.32	2.31	12.27	3.04	75.2

Table 5: Statistics of partition construction and selection. Reduction is computed as $1 - \frac{\text{selected parts}}{\text{candidate parts}}$.

Component-Level Ablations. We conduct component-level ablations on WikiTQ to isolate the contribution of the Question Analyzer, semantic column grouping, and joint row–column partitioning. The results are shown in Table 6.

Configuration	WikiTQ	
	EM	Δ
Full PARTAB	**79.31**	–
PARTAB w/o Question Analyzer	72.21	−7.10
PARTAB with Random Column Grouping	68.84	−10.47
PARTAB with Row Chunking Only	57.19	−22.12

Table 6: Component-level ablation results on WikiTQ. Each Δ denotes the absolute change in exact match relative to the full PARTAB configuration.

Removing the Question Analyzer decreases exact match by 7.10 points, showing that the structured question representation improves schema-level routing and partition selection. Replacing semantic column groups with randomly constructed groups reduces performance by 10.47 points, directly demonstrating that semantic coherence is important rather than grouping alone.

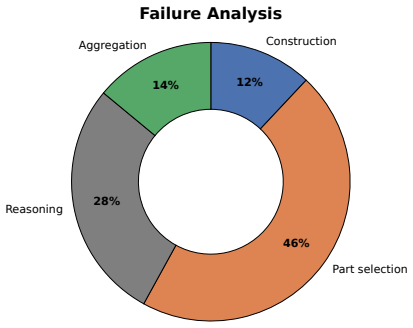

Figure 5: Distribution of failure types based on 100 incorrect predictions. Part-selection errors constitute the largest source of failure.

The largest degradation occurs when PARTAB uses row chunking without semantic column grouping, resulting in a 22.12-point reduction. This result indicates that row segmentation alone does not provide the structured evidence organization required for reliable table reasoning. Instead, the performance gains arise from jointly localizing rel-

evant rows and semantically related columns.

Error Analysis. We manually analyze 100 incorrect predictions and categorize the failures into four types (Figure 5). Part-selection errors are the most common (46%), occurring when relevant partitions are missed or row coverage is insufficient, indicating that evidence localization remains the main bottleneck. Reasoning errors account for 28%, where sufficient evidence is present but the executor fails in comparison, inference, or computation. Aggregation errors represent 14% and mainly arise from questions requiring near-global row coverage. Construction errors are least frequent (12%), consistent with the high semantic coherence of the generated column groups and suggesting that partition construction is more reliable than downstream selection. We provide additional experimental results, and analysis in the **Appendix A** and **B**.

5 Related Work

LLM-based Table Reasoning. Large language models (LLMs) have enabled strong progress in table question answering and fact verification (Cao and Liu, 2026; Zhou et al., 2025; Lu et al., 2025; Fang et al., 2024). Early models such as TAPAS (Herzig et al., 2020) and TAPEX (Liu et al., 2022) demonstrated the effectiveness of pretraining for tabular reasoning, while recent prompting-based approaches improve flexibility but struggle with long tables due to attention limitations and incomplete evidence utilization (Tyagi et al., 2025).

Text-to-SQL and Symbolic Reasoning. Another line of work formulates table reasoning as Text-to-SQL generation (Yu et al., 2018; Li et al., 2023), enabling reliable aggregation through executable queries. While effective for structured data, these approaches rely on precise predicate construction and often fail under noisy textual conditions (Wang et al., 2025c; Evkarpidi and Tutubalina, 2025).

Decomposition and Retrieval. To address scalability, prior work decomposes tables or retrieves relevant substructures. Methods such as TabSQLify (Nahid and Rafiei, 2024b), TabSD (Wang et al., 2025b), and Tree-of-Table (Ji et al., 2025) reduce reasoning complexity through structured decomposition. Recent approaches further explore sub-table retrieval and structured reasoning, including ITR (Lin et al., 2023), PieTa (Lee et al., 2025), RoT (Zhang et al., 2025b), and CABINET (Patnaik et al., 2024). Long-context benchmarks highlight persistent performance degradation as table size increases (Li et al., 2025; Wang et al., 2025a; Wu et al., 2025). Recent retrieval-augmented approaches such as TableRAG (Chen et al., 2024; Yu et al., 2025) improve scalability by retrieving relevant table content, but do not explicitly structure reasoning within tables.

Hybrid and Tool-Augmented Reasoning. Hybrid frameworks combine LLM reasoning with external tools such as SQL or programs. Representative approaches include ReAcTable (Zhang et al., 2024b), Weaver (Khoja et al., 2025), E^5 (Zhang et al., 2024c), TIDE (Yang et al., 2025), and TABDSR (Jiang et al., 2025), as well as broader neuro-symbolic systems (Zhang et al., 2024a; Cheng et al., 2023; Zhang et al., 2025a; Ye et al., 2023; Abhyankar et al., 2025). These methods improve numerical reasoning and interpretability but often depend on execution pipelines or structured representations.

Existing approaches typically rely on full-table reasoning, single-view decomposition or retrieval, or symbolic execution, each involving trade-offs among flexibility, scalability, and evidence completeness. **PARTAB** instead constructs a structured evidence interface in which query-relevant information is organized into semantically coherent, row-linked table regions. By combining semantic grouping with hierarchical partition selection and cross-region evidence composition, PARTAB enables localized reasoning while preserving the relational structure needed to connect evidence across the table.

6 Conclusion

We introduced **PARTAB**, a partition-aware framework that constructs a structured evidence interface between LLMs and tables. PARTAB organizes query-relevant evidence into semantically coherent, row-linked regions and composes the selected regions for reasoning rather than relying on a full table or single reduced view. Experiments across multiple table reasoning benchmarks show strong performance, improved evidence localization, and substantial context reduction, with larger benefits on complex tables. These results demonstrate that structured, partition-aware evidence construction provides an effective reasoning interface for scalable table understanding.

Limitations

PARTAB focuses on inference-time reasoning over tables, with particular emphasis on settings where the evidence required for a query is localized to a subset of rows and columns. The framework is designed to improve evidence localization and structured reasoning over large and noisy tables, rather than to replace symbolic execution for exhaustive aggregation or to address broader settings such as complex multi-table operations, temporal reasoning, or database-scale query processing.

Despite its effectiveness, PARTAB has several limitations. First, it relies on LLM-based components for question analysis, grouping, and selection, making it sensitive to prompt design and model variability, with errors potentially propagating across stages. Second, PARTAB does not explicitly enforce global completeness for aggregation tasks, which can lead to missing evidence when full-table coverage is required. Third, the current heuristic partitioning strategy (fixed chunk size and LLM-based grouping) may not generalize optimally across diverse table structures. Finally, although each step operates on reduced context, the multi-stage pipeline can introduce additional latency compared to single-pass prompting.

Future work includes integrating symbolic execution for reliable aggregation, improving part selection through learned or hybrid retrieval mechanisms, and extending PARTAB to more complex settings such as multi-table reasoning and temporal queries.

Ethics Statement

PARTAB is a general-purpose framework for table reasoning and does not introduce or collect sensitive user data. However, like other LLM-based reasoning systems, it may produce incorrect answers when relevant evidence is omitted or downstream reasoning fails. Such errors could be consequential if the system were applied without verification in high-stakes domains. PARTAB also requires multiple LLM inference calls, introducing additional computational cost relative to single-pass prompting, although our experiments show substantially lower token usage than several multi-stage baselines. We therefore view PARTAB primarily as a research framework and recommend appropriate validation and human oversight for consequential applications.

References

Nikhil Abhyankar, Vivek Gupta, Dan Roth, and Chandan K. Reddy. 2025. H-STAR: LLM-driven hybrid SQL-text adaptive reasoning on tables. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8841–8863, Albuquerque, New Mexico. Association for Computational Linguistics.

Lang Cao and Hanbing Liu. 2026. Tablemaster: A recipe to advance table understanding with language models. *Preprint*, arXiv:2501.19378.

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. 2024. TableRAG: Million-token table understanding with language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Zhoujun Cheng, Tianyang Jiang, Qianhui Zhang, and Lei Li. 2023. Binding language models in symbolic languages for table question answering. In *International Conference on Learning Representations (ICLR)*.

Nikolas Evkarpidi and Elena Tutubalina. 2025. Bridging the gap between open-source and proprietary large language models in table question answering.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models on tabular data: Prediction, generation, and understanding. *ACM Computing Surveys*.

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*, page 4320–4333. Association for Computational Linguistics.

Deyi Ji, Lanyun Zhu, Siqi Gao, Peng Xu, Hongtao Lu, Jieping Ye, and Feng Zhao. 2025. Tree-of-table: Unleashing the power of llms for enhanced large-scale table understanding. In *International Conference on Learning Representations (ICLR)*.

Changjiang Jiang, Fengchang Yu, Haihua Chen, Wei Lu, and Jin Zeng. 2025. TabDSR: Decompose, sanitize, and reason for complex numerical reasoning in tabular data. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3172–3196, Suzhou, China. Association for Computational Linguistics.

Rohit Khoja, Devanshu Gupta, Yanjie Fu, Dan Roth, and Vivek Gupta. 2025. Weaver: Interweaving SQL and LLM for table reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 28282–28308, Suzhou, China. Association for Computational Linguistics.

Wonjin Lee, Kyumin Kim, Sungjae Lee, Jihun Lee, and Kwang In Kim. 2025. Piece of table: A divide-and-conquer approach for selecting subtables in table question answering.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Liyao Li, Jiaming Tian, Hao Chen, Wentao Ye, Chao Ye, Haobo Wang, Ningtao Wang, Xing Fu, Gang Chen, and Junbo Zhao. 2025. LongTableBench: Benchmarking long-context table reasoning across real-world formats and domains. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11927–11965, Suzhou, China. Association for Computational Linguistics.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Zhenhua Lin, Jian-Guang Lou, and Dongmei Zhang. 2022. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations (ICLR)*.

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

Qingyang Mao, Qi Liu, Zhi Li, Mingyue Cheng, Zheng Zhang, and Rui Li. 2026. Potable: Towards systematic thinking via plan-then-execute stage reasoning on tables. *Preprint*, arXiv:2412.04272.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024a. NormTab: Improving symbolic reasoning in LLMs through tabular data normalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3569–3585, Miami, Florida, USA. Association for Computational Linguistics.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024b. TabSQLify: Enhancing reasoning capabilities of LLMs through table decomposition. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5725–5737, Mexico City, Mexico. Association for Computational Linguistics.

Md Mahadi Hasan Nahid and Davood Rafiei. 2025. Prism: Agentic retrieval with llms for multi-hop question answering.

Md Mahadi Hasan Nahid, Davood Rafiei, Weiwei Zhang, and Yong Zhang. 2026. Rethinking schema linking: A context-aware bidirectional retrieval approach for text-to-SQL. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4516–4546, Rabat, Morocco. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. CABINET: Content relevance-based noise reduction for table question answering. In *The Twelfth International Conference on Learning Representations*.

Rishit Tyagi, Mohit Gupta, and Rahul Bouri. 2025. Agentic large language models for question answering over tabular data.

Lanrui Wang, Mingyu Zheng, Hongyin Tang, Zheng Lin, Yanan Cao, Jingang Wang, Xunliang Cai, and Weiping Wang. 2025a. Needleinatable: Exploring long-context capability of large language models towards long-structured tables. In *39th Conference on Neural Information Processing Systems (NeurIPS 2025)*.

Yuxiang Wang, Junhao Gan, and Jianzhong Qi. 2025b. Tabsd: Sql-based table decomposition for free-form table question answering.

Zhongyuan Wang, Richong Zhang, Zhijie Nie, and Hangyu Mao. 2025c. General table question answering via answer-formula joint generation.

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*.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, and 1 others. 2025. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.

Zhen Yang, Ziwei Du, Minghan Zhang, Wei Du, Jie Chen, Zhen Duan, and Shu Zhao. 2025. Triples as the key: Structuring makes decomposition and verification easier in LLM-based tableQA. In *The Thirteenth International Conference on Learning Representations*.

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 *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 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.

Xiaohan Yu, Pu Jian, and Chong Chen. 2025. TableRAG: A retrieval augmented generation framework for heterogeneous document reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14063–14082, Suzhou, China. Association for Computational Linguistics.

Junwen Zhang, Pu Chen, and Yin Zhang. 2025a. Tablemoe: Neuro-symbolic routing for structured expert reasoning in multimodal table understanding. *Preprint*, arXiv:2506.21393.

Siyue Zhang, Anh Tuan Luu, and Chen Zhao. 2024a. SynTQA: Synergistic table-based question answering via mixture of text-to-SQL and E2E TQA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2352–2364, Miami, Florida, USA. Association for Computational Linguistics.

Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. 2025b. RoT: Enhancing table reasoning with iterative row-wise traversals. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 559–579, Suzhou, China. Association for Computational Linguistics.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M. Patel. 2024b. Reactable: Enhancing react for table question answering. *Proc. VLDB Endow.*, 17(8):1981–1994.

Zhehao Zhang, Yan Gao, and Jian-Guang Lou. 2024c. e^5: Zero-shot hierarchical table analysis using augmented LLMs via explain, extract, execute, exhibit and extrapolate. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1244–1258, Mexico City, Mexico. Association for Computational Linguistics.

Wei Zhou, Bolei Ma, Annemarie Friedrich, and Mohsen Mesgar. 2025. Table question answering in the era of large language models: A survey.

A Additional Results

This appendix reports additional experiments and diagnostics conducted to further evaluate the effectiveness, robustness, and efficiency of PARTAB. Unless stated otherwise, PARTAB uses the default LLM-based selection strategy and the same evaluation metrics as the main paper.

Budget-Matched Full-Table Control. To distinguish the contribution of partition-aware reasoning from the additional inference budget, we compare PARTAB with a full-table baseline that uses the same number of LLM calls. The budget-matched baseline repeatedly reasons over the complete serialized table but does not perform semantic grouping or hierarchical partition selection.

Method	LLM Calls	WikiTQ EM
Full-table (Budget-matched)	5	64.61
PARTAB	5	**79.31**

Table 7: Budget-matched comparison between repeated full-table prompting and PARTAB.

As shown in Table 7, the five-call full-table baseline obtains 64.61 exact match on the evaluated WikiTQ subset, whereas PARTAB obtains 79.31, an improvement of 14.70 points. Thus, repeatedly prompting the model with the full table does not reproduce the benefits of explicitly constructing a localized evidence state.

This control indicates that the performance improvement is attributable primarily to semantic grouping, schema-level routing, and row-level partition selection rather than merely to additional LLM interactions.

Dataset	Fallback Rate (%)	Evidence Recall (%)
WikiTQ	2.70	87.0†
TabFact	1.90	–
TableBench	0.24	–

Table 8: Selection stability across the evaluation datasets. †Evidence recall is based on manual annotation of 100 WikiTQ examples.

Selection Stability and Fallback Frequency. The LLM-based Part Selector may occasionally return an empty or malformed selection. In such cases, PARTAB invokes the deterministic BASIC-SELECT fallback to preserve evidence coverage.

Table 8 reports the empirical fallback frequency. The fallback is activated for only 2.7% of WikiTQ examples, 1.9% of TabFact examples, and 0.24% of TableBench examples. Most fallback cases are caused by malformed or unparsable structured outputs rather than an explicit inability to identify relevant evidence.

These low rates show that the LLM-guided selection procedure is generally stable. The fallback primarily serves as a robustness safeguard for rare generation or parsing failures and does not drive the reported performance.

Evidence Coverage. The evaluated datasets do not provide gold cell-, row-, or column-level evidence annotations. We therefore manually inspect 100 WikiTQ examples and determine whether the partitions selected by PARTAB contain all evidence necessary to answer the corresponding question.

The selected partitions achieve an evidence recall of 87%. This result indicates that the compact contexts constructed by PARTAB contain sufficient evidence in most cases. The remaining failures primarily involve aggregation-heavy questions requiring broad row coverage or cases where a required row occurs in an unselected partition.

This analysis separates evidence-localization failures from downstream execution errors. In particular, an incorrect final answer does not necessarily imply that partition selection failed, because the answer executor may still make a reasoning or computation error after receiving the correct evidence.

B Additional Analysis

Accuracy–Efficiency Trade-off. We further compare the inference efficiency of PARTAB with full-table prompting and recent multi-stage table reasoning systems. Table 9 reports the number of LLM calls, average tokens processed per query, and TabFact accuracy.

The default LLM-based configuration of PARTAB uses five LLM calls and approximately 2.3k tokens per query. In comparison, Chain-of-Table requires up to 25 calls and approximately 13.2k tokens, while TableMaster uses 11 calls and approximately 3.3k tokens. Despite using substantially fewer calls, PARTAB obtains the highest accuracy among the compared methods.

The similarity-based variant is the most token-efficient configuration, using approximately 1.6k tokens per query. However, its lower accuracy shows that reducing context is beneficial only when

Method	LLM Calls	Avg. Tokens / Query	TabFact Acc.
Full-table prompting	1	2.1k	73.22
Chain-of-Table	≤ 25	13.2k	85.09
TableMaster	11	3.3k	90.12
PARTAB – Basic	4	2.1k	87.88
PARTAB – Similarity	4	**1.6k**	85.87
PARTAB – LLM-based	5	2.3k	**90.48**

Table 9: Accuracy–efficiency comparison on TabFact. Token counts cover the complete per-query pipeline. Bold values indicate the lowest token usage and the highest accuracy, respectively.

the selected evidence is semantically adequate. Aggressive context reduction based on lexical similarity can omit evidence required for compositional reasoning.

Method	Avg. Selected Parts ↓	WikiTQ (EM)
Full Table	17.56	59.43
Basic Selection	6.50	74.71
Similarity-based	5.36	67.25
LLM-based	**3.88**	**79.31**

Table 10: Partition reduction efficiency on WikiTableQuestions. We report the average number of selected parts and Exact Match (EM) accuracy.

Partition Reduction Efficiency. Table 10 shows that PARTAB significantly reduces the amount of context required for reasoning while improving performance. Compared to full-table prompting, all partition-based methods select substantially fewer parts, leading to more focused reasoning.

Among them, LLM-based selection achieves the best trade-off, reducing the average number of selected parts to 3.88 while achieving the highest accuracy (79.31 EM). This indicates that effective partition selection not only improves efficiency but also enhances reasoning quality by filtering out irrelevant context. In contrast, similarity-based selection reduces the number of parts but suffers from lower accuracy, highlighting that aggressive reduction without semantic understanding can harm performance.

Dataset Structural Statistics. The evaluated datasets differ substantially in table size and structure, as shown in Table 11. WikiTQ contains the longest tables, including examples with up to 517 rows. TableBench contains examples with up to 176 rows, whereas TabFact tables contain at most 47 rows.

All three benchmarks also contain wide schemas, with maximum column counts between 20 and 21. These statistics motivate partitioning along both table dimensions. Long tables require row-level evidence localization, while wide tables require schema-level routing to prevent irrelevant columns from diluting the executor's attention.

C Discussions

When Partition-Aware Reasoning Helps. The additional experiments provide a more precise characterization of when partition-aware reasoning is beneficial. The largest improvements occur on long or wide tables, where the evidence required by a question occupies only a small portion of the complete table. In such settings, full-table prompting exposes the model to many irrelevant values, increasing attention dilution and making it harder to localize and reason over the relevant evidence.

Partitioning is particularly effective for lookup, comparison, and fact verification questions. These tasks usually depend on a localized set of attributes and rows, making it possible to construct a compact evidence state without losing task-relevant information. The large gains on TabFact-Hard across all three model backbones provide strong evidence for this behavior.

Partition-aware reasoning is also useful when table cells contain noisy or free-form text that is difficult to filter using exact symbolic predicates. Semantic grouping and LLM-based selection can identify conceptually relevant attributes even when the question and table use different surface forms.

Relationship to Full-Table and Single-View Reasoning. The benefit of PARTAB is not simply that it passes fewer tokens to the answer executor. A pruning method can also reduce context length, but usually constructs a single reduced table. If that reduced view omits a required column or row, the missing evidence cannot be recovered during reasoning.

Dataset	Number of Rows			Number of Columns		
	Min	Avg	Max	Min	Avg	Max
WikiTableQuestions	5	25.65	517	3	6.39	21
TabFact	5	13.41	47	5	6.29	20
TableBench	6	16.75	176	4	6.64	20

Table 11: Structural statistics of the evaluation datasets.

In contrast, PARTAB preserves a collection of complementary, row-linked table parts. Semantic group selection first identifies the relevant schema subspace, after which part selection identifies relevant row chunks within that subspace. The universal `row_id` key allows evidence from different semantic groups to be aligned during answer execution.

Thus, partitioning acts as a reasoning control mechanism rather than only a preprocessing operation. The model reasons over multiple structured table regions while preserving their relational correspondence instead of receiving either the complete table or a single flattened sub-table.

Relationship to Text-to-SQL Reasoning. Text-to-SQL methods are highly effective when a question can be translated into precise filtering, joining, sorting, or aggregation operations. Executing the generated query in a database engine also provides reliable numerical computation.

However, the evaluated benchmarks include questions requiring semantic verification, implicit comparison, and interpretation of noisy or free-form textual cells. These operations are not always naturally expressible as exact SQL predicates. Text-to-SQL systems may therefore fail during predicate construction even when the necessary evidence is present.

PARTAB addresses a complementary bottleneck: how to localize and structure evidence before semantic reasoning. It does not replace symbolic execution. For completeness-sensitive aggregation questions, a promising extension would combine partition-aware semantic localization with deterministic SQL or program execution.

Remaining Limitations. The additional analyses also clarify several limitations of the current framework. First, part selection remains the dominant source of error. Although the selected partitions obtain 87% evidence recall on the manually annotated WikiTQ sample, the remaining selection failures can directly prevent correct downstream reasoning.

Second, localized evidence selection is less suitable for questions requiring exhaustive aggregation over most or all table rows. Selecting only a subset of partitions can violate the completeness requirement of counting, global extrema, or large-scale aggregation operations.

Third, the effectiveness of the pipeline remains dependent on the instruction-following and structured-output capabilities of the selected LLM backbone.

A promising extension is an adaptive global–local controller that first identifies whether the question requires localized semantic evidence or global table completeness. Localized questions could use the current partition-aware pipeline, while completeness-sensitive questions could be routed to full-table or symbolic execution.

D Extended Literature Review

LLM-based Table Question Answering. Recent advances have significantly improved Table Question Answering (TableQA) using large language models. Surveys highlight rapid progress in LLM-based table understanding while identifying scalability and reasoning reliability as major challenges (Zhou et al., 2025; Lu et al., 2025; Fang et al., 2024; Cao and Liu, 2026). Direct prompting approaches enable flexible semantic reasoning but struggle with long tables due to attention limitations and incomplete evidence utilization (Tyagi et al., 2025). Earlier transformer-based models such as TAPAS demonstrated the effectiveness of pretraining for tabular reasoning, establishing foundations for modern LLM-based approaches (Herzig et al., 2020).

Text-to-SQL and Symbolic Table Reasoning. A dominant paradigm formulates table reasoning as Text-to-SQL generation, translating natural language queries into executable programs evaluated by database engines (Yu et al., 2018; Nahid et al., 2026). Benchmarks such as BIRD emphasize realistic database reasoning scenarios requiring accu-

rate aggregation and efficient execution (Li et al., 2023). Symbolic execution enables reliable numerical reasoning; however, these approaches depend on constructing structured predicates and often struggle when filtering conditions require semantic interpretation over noisy textual cells. Recent work integrating LLM planning with executable reasoning further highlights both the strengths and limitations of symbolic pipelines (Wang et al., 2025c; Evkarpidi and Tutubalina, 2025; Liu et al., 2022).

Decomposition and Context Reduction for Long Tables. To address scalability challenges, several works reduce reasoning complexity through decomposition strategies. These include question decomposition and table decomposition methods that divide large tables into smaller reasoning units. TabSQLify introduces SQL-guided subtable extraction to improve reasoning efficiency on large tables (Nahid and Rafiei, 2024b), while TabSD extends decomposition to free-form noisy tables (Wang et al., 2025b). Structured reasoning frameworks such as Tree-of-Table further organize hierarchical reasoning over tabular data (Ji et al., 2025). Long-context benchmarks including LongTableBench and NeedleInATable demonstrate that reasoning performance degrades substantially as table size increases (Li et al., 2025; Wang et al., 2025a; Wu et al., 2025).

Recent work has explored decomposition, retrieval, and hybrid execution for table reasoning. Methods such as ITR (Lin et al., 2023) and PieTa (Lee et al., 2025) focus on retrieving or constructing relevant sub-tables to reduce long-table noise, while RoT (Zhang et al., 2025b) performs row-wise reasoning to improve scalability. CABINET (Patnaik et al., 2024) addresses noisy tables through content relevance weighting, and TABDSR (Jiang et al., 2025) combines decomposition with program-based reasoning for numerical tasks. Hybrid frameworks such as ReAcTable (Zhang et al., 2024b) and Weaver (Khoja et al., 2025) integrate LLM reasoning with external tools, while E^5 (Zhang et al., 2024c) and TIDE (Yang et al., 2025) introduce structured intermediate representations for improved reasoning and verification. In contrast, PARTAB treats partitioning as a reasoning control mechanism, combining semantic grouping with row-linked selection to enable structured reasoning over partitioned table views.

While decomposition improves evidence localization, restricting reasoning to selected subsets introduces a critical trade-off: required evidence may be omitted when tasks demand exhaustive global computation.

Hybrid Neural–Symbolic Table Reasoning. An emerging direction combines LLM reasoning with executable intermediate representations such as SQL queries, programs, or dataframe operations. Hybrid systems leverage LLMs for interpretation while relying on deterministic execution engines for numerical correctness and interpretability (Wang et al., 2025c; Evkarpidi and Tutubalina, 2025; Abhyankar et al., 2025). Approaches such as SynTQA demonstrate complementary strengths between symbolic execution and end-to-end reasoning under different task conditions (Zhang et al., 2024a; Cheng et al., 2023). Neuro-symbolic routing frameworks such as TableMoE further explore structured expert reasoning for tabular understanding (Zhang et al., 2025a; Ye et al., 2023).

Retrieval-Augmented Reasoning over Structured Data. Retrieval-Augmented Generation (RAG) has recently been extended to structured and semi-structured data sources, including tables and databases (Tyagi et al., 2025; Nahid and Rafiei, 2025). Table-aware retrieval methods index table regions or schema components and retrieve relevant fragments prior to reasoning, improving scalability by avoiding full-context processing. Recent work has explored retrieval-augmented approaches for scalable table reasoning. Chen et al. (2024) introduce TableRAG, which enables large-scale table understanding by retrieving relevant table segments from million-token contexts. Similarly, Yu et al. (2025) propose a retrieval-augmented framework for heterogeneous document reasoning that combines text retrieval with SQL-based execution over tabular data. However, retrieval-based approaches inherit an important limitation: incomplete retrieval can produce confident but incorrect answers, particularly for aggregation or counting tasks requiring full-table access.

Limitations of Existing Approaches. Despite substantial progress, existing methods expose complementary trade-offs. Symbolic approaches provide reliable aggregation but depend on precise structured predicates and can struggle with noisy or free-form textual content. Direct LLM reasoning offers greater semantic flexibility but becomes less reliable as table size and irrelevant context increase.

Decomposition and retrieval methods improve scalability by reducing context, yet may omit evidence required for completeness-sensitive operations.

These trade-offs suggest that effective table reasoning should distinguish *semantic evidence localization* from *global execution*. Rather than treating any single reasoning paradigm as universally sufficient, a more general design should coordinate localized semantic reasoning with exhaustive or symbolic computation when required. **PARTAB** addresses the first part of this problem by constructing structured, query-relevant evidence states over table partitions, while complementary global or symbolic execution remains an important direction for completeness-sensitive reasoning.

E LLM Usage

All technical content, contributions, analyses, and implementations are entirely our own. Large language models were used only to improve clarity and readability, and generating charts and diagrams. LLMs were occasionally used for minor editing and code debugging, with all outputs reviewed and verified by the authors.

F Illustrative Examples and Prompt Template

We present three illustrative examples to demonstrate how PARTAB operates across different reasoning scenarios, including numerical computation, comparison, and entity lookup. These examples highlight how partition-aware reasoning enables effective evidence localization and improves reasoning accuracy by focusing on relevant subsets of the table. Figure 6, 7, 8 illustrates a representative examples of three different categories.

We also provide the detail prompt used to implement our method in Figure 9, 10, 11, 12, 13.

Input Table and Question

Table (Preview):

row_id	rank	peak	elevation_m	...
1	1	galdhøpiggen	2469	...
2	2	jiehkkevárri	1833	...
3	3	snøhetta	2286	...
4	4	store lenangstind	1625	...
⋮	⋮	⋮	⋮	⋮

Question: How much higher is the elevation of the mountain with the highest elevation than the mountain with the lowest elevation?

Step 1: Question Analysis

{ "question_type": "multi_step",
"needs_row_filtering": false,
"needs_aggregation": true,
"needs_comparison": true,
"needs_sorting": false,
"expected_answer_type": "number" }

Step 2: Column Grouping

{ "peak_identity": ["rank", "peak"],
"peak_metrics": ["elevation_m", "prominence_m", "isolation_km"],
"location": ["municipality", "county"] }

Step 3: Group Selection

Required groups: `["peak_metrics"]`

Step 4: Partition Selection

Selection mode: basic
Selected parts: `peak_metrics_1_5, ..., peak_metrics_16_19`

Total partitions: 12
Selected parts: 4

Step 5: Answer Execution

Predicted Answer: 1207 m
Gold Answer: 1207
Exact Match: True

Figure 6: Illustrative example of PARTAB on a multi-step numerical reasoning task. The table is truncated for clarity; ellipses (. . .) indicate omitted columns and rows. PARTAB isolates the relevant metric columns, selects a small subset of partitions, and correctly computes the difference between the highest and lowest elevations.

Input Table and Question

Table (Preview):

row_id	country	biodiversity	...
1	Belize	3771	...
2	Costa Rica	13630	...
3	El Salvador	3618	...
4	Guatemala	9927	...
⋮	⋮	⋮	⋮

Question: Which country in Central America is the most biodiverse?

Step 1: Question Analysis

{ "question_type": "comparison",
"needs_row_filtering": true,
"needs_aggregation": false,
"needs_comparison": true,
"needs_sorting": true,
"expected_answer_type": "entity" }

Step 2: Column Grouping

{ "geography": ["country"],
"vertebrates": ["amphibians", "birds", "mammals", "reptile", "total_terrestrial_vertebrates"],
"plants_and_biodiversity": ["vascular_plants", "biodiversity"] }

Step 3: Group Selection

Required groups: `["geography", "plants_and_biodiversity"]`

Step 4: Partition Selection

Selection mode: similarity
Selected parts: `geography_1_5`, `geography_6_7`, `plants_and_biodiversity_1_5`, `plants_and_biodiversity_6_7`

Total partitions: 6
Selected parts: 4

Step 5: Answer Execution

Predicted Answer: Costa Rica
Gold Answer: Costa Rica
Exact Match: True

Figure 7: Illustrative example of PARTAB on a comparison-based reasoning task. The table is truncated for clarity; ellipses (. . .) indicate omitted columns and rows. PARTAB selects the relevant semantic groups and partitions, enabling accurate identification of the most biodiverse country.

Input Table and Question

Table (Preview):

row_id	heat	rank	name	...
1	1	1	Gabriela Szabo (ROU)	...
2	1	2	Paula Radcliffe (GBR)	...
3	2	3	Fernanda Ribeiro (POR)	...
4	1	4	Harumi Hiroyama (JPN)	...
5	1	5	Roberta Brunet (ITA)	...
⋮	⋮	⋮	⋮	⋮
30	...	...	Li Wei (CHN)	...

Question: What country does Li Wei represent?

Step 1: Question Analysis

{ "question_type": "lookup",
"needs_row_filtering": true,
"needs_aggregation": false,
"needs_comparison": false,
"needs_sorting": false,
"expected_answer_type": "entity" }

Step 2: Column Grouping

{ "competition": ["heat", "rank"],
"athlete": ["name"],
"outcome": ["result", "notes"] }

Step 3: Group Selection

Required groups: `["athlete"]`

Step 4: Partition Selection

Selection mode: llm
Selected parts: `athlete_1_5, ..., athlete_26_30`

Total partitions: 27
Selected parts: 6

Step 5: Answer Execution

Predicted Answer: CHN
Gold Answer: China
Exact Match: True

Figure 8: Illustrative example of PARTAB on a lookup-based reasoning task. The table is truncated for clarity; ellipses (...) indicate omitted rows and columns. PARTAB identifies the relevant row and extracts the country information from the athlete name field.

Question Analyzer Prompt

You are a question analyzer for table reasoning.

You will receive a natural language question over a table. Your task is to infer the likely reasoning requirements of the question. In particular, determine the question type, whether row filtering is needed, whether aggregation, comparison, or sorting is required, and the expected answer type.

Question: {QUESTION}

Return STRICT JSON only in the following format:

```
{
"question_type": "lookup",
"needs_row_filtering": true,
"needs_aggregation": false,
"needs_comparison": false,
"needs_sorting": false,
"expected_answer_type": "entity"
}
```

Allowed values for `question_type`*:*
`lookup, aggregation, comparison, counting, sorting, multi_step, other`

Allowed values for `expected_answer_type`*:*
`entity, number, boolean, text, list, other`

Only output a valid JSON object. Do not include "‘`json` *or any additional explanation.*

Figure 9: Question Analyzer Prompt Template used to infer reasoning requirements from a natural language question.

Column Grouping Prompt

You are designing semantic column groups for partition-aware reasoning over a table.

You will receive:
- The question
- The table schema
- A few sample rows from the table

Your goal is to group the table columns into a small number of semantically coherent groups that support partition-aware reasoning. The universal linking key is `row_id`*. Do not include* `row_id` *in any semantic group. Each non-key column must appear in exactly one group. Prefer 2 to 5 groups when possible, and use short, meaningful group names.*

Question: {QUESTION}
Table schema: {SCHEMA}
Sample rows: {TABLE_PREVIEW}

Return STRICT JSON only in the following format:

```
{
"column_groups": {
"group_name_1": ["col1", "col2"],
"group_name_2": ["col3", "col4"]
},
"key_columns": ["row_id"]
}
```

Only output a valid JSON object. Do not include "‘`json` *or any additional explanation.*

Figure 10: Column Grouping Prompt Template used to construct semantically coherent column groups for partition-aware table reasoning.

Group Selection Prompt

You are selecting the minimum required column groups needed to answer a table question.

You will receive:
- The original question
- The question analysis output
- The available column groups

Your goal is to select the minimum set of column groups required to answer the question. If the question requires filtering, lookup, comparison, aggregation, or sorting, include the relevant groups. Return only valid group names from the provided list.

Question: {QUESTION}
Question analysis: {QUESTION_ANALYSIS}
Available column groups: {COLUMN_GROUPS}

Return STRICT JSON only in the following format:

```
{
"required_groups": ["group1", "group2"]
}
```

Only output a valid JSON object. Do not include “‘`json` *or any additional explanation.*

Figure 11: Group Selection Prompt Template used to identify the minimum required semantic groups for answering a question.

LLM-based Part Selection Prompt

You are selecting the minimum required table parts needed to answer a question.

You will receive:
- The original question
- A list of candidate table parts, where each part includes its part name, group name, row range, columns, and table content

Your goal is to select the minimum set of part names needed to answer the question. Only return valid part names from the provided candidate list.

Question: {QUESTION}
Candidate table parts: {PARTS}

Return STRICT JSON only in the following format:

```
{
"required_part_names": ["part1", "part2"]
}
```

Only output a valid JSON object. Do not include “‘`json` *or any additional explanation.*

Figure 12: LLM-based Part Selection Prompt Template used to select the minimum required table parts from the candidate partition set.

Answer Execution Prompt

You are a Table Question Answering assistant.

Answer the question using ONLY the provided table parts. The linking key across parts is `row_id`*. Do not use outside knowledge. Do not explain your reasoning. If the answer cannot be found from the table parts, return exactly:* `I cannot find it.`

Question: {QUESTION}
Table parts: {CONTEXT}

Return only the final short answer.

Figure 13: Answer Execution Prompt Template used to generate the final answer from the selected table parts.