RePanda: Pandas-powered Tabular Verification and Reasoning

Anonymous ACL submission

Abstract

Fact-checking tabular data is essential for ensuring the accuracy of structured information. However, existing methods often rely on black-box models with opaque reasoning. We introduce *RePanda*, a structured fact verification approach that translates claims into executable `pandas` queries, enabling interpretable and verifiable reasoning.

To train *RePanda*, we construct *PanTabFact*, a structured dataset derived from TabFact train set, where claims are paired with executable queries generated using DeepSeek-Chat and refined through automated error correction. Fine-tuning DeepSeek-coder-7B-instruct-v1.5 on *PanTabFact*, *RePanda* achieves 84.09% accuracy on TabFact test set. To evaluate Out-of-Distribution (OOD) generalization, we interpret question-answer pairs from WikiTableQuestions as factual claims and refer to this dataset as *WikiFact*. Without additional fine-tuning, *RePanda* achieves 84.72% accuracy on *WikiFact*, significantly outperforming all other baselines and demonstrating strong OOD robustness.

Beyond fact verification, *RePanda* extends to tabular question answering by generating executable queries that retrieve precise answers. To support this, we introduce *PanWiki*, a dataset mapping WikiTableQuestions to `pandas` queries. Fine-tuning on *PanWiki*, *RePanda* achieves 75.1% accuracy in direct answer retrieval. These results highlight the effectiveness of structured execution-based reasoning for tabular verification and question answering.

1 Introduction

Fact verification is a critical task in artificial intelligence, with applications in journalism, financial auditing, and scientific research. As misinformation continues to proliferate, automated fact-checking has become an essential tool for verifying claims against structured sources such as tabular data. Unlike textual fact verification, which matches claims against unstructured text, tabular fact verification requires reasoning over structured numerical and categorical data, making it a fundamentally different and more challenging task(Herzig et al., 2020; Gu et al., 2022).

Despite recent progress in large language models (LLMs), their ability to reason over structured tabular data remains limited. LLMs are pre-trained predominantly on unstructured text, where semantic relationships are inferred through sequential token dependencies. However, tabular data encodes information in a structured format where relationships are often implicit, requiring operations such as aggregation, filtering, and comparison across multiple rows and columns (Liu et al., 2021; Eisenschlos et al., 2020).

A major challenge lies in LLMs' limited understanding of structured data, such as tables, when processing tabular information. Models like TAPAS (Herzig et al., 2020) and TAPEX (Liu et al., 2021) attempt to address this by incorporating table-aware pretraining to improve table understanding. However, these approaches flatten tables into sequences, losing structural integrity and making it difficult to capture relationships between rows and columns (Gu et al., 2022). As a result, they struggle with complex table operations, such as aggregation and multi-row comparisons, which are essential for fact verification. Moreover, approaches like PASTA (Gu et al., 2022) use sentence-table cloze pretraining to improve table understanding but rely on predefined operations, which may not generalize to complex, unseen queries.

Another fundamental limitation of existing approaches is the lack of interpretability. Many LLM-based fact-checking models function as black-box classifiers, predicting whether a claim is true or false without explicitly showing the reasoning steps. This lack of transparency makes it difficult to ver-

ify results, particularly in high-stakes applications such as legal and financial audits (Eisenschlos et al., 2020). Ideally, a fact-checking system should provide a structured reasoning process that can be independently validated.

To address these challenges, we propose a novel approach that reformulates tabular fact verification and question-answering as a structured representation learning task. Rather than assuming LLMs inherently understand tabular structures, we task them to construct explicit reasoning steps as executable `pandas` queries. Since `pandas` queries are designed for tabular operations (e.g., filtering, counting, or aggregating), they provide transparent and interpretable reasoning steps on how the answer is obtained.

To train our model, we construct *PanTabFact*, an augmented fact-checking dataset based on TabFact. Using the DeepSeek-Chat model (Guo et al., 2025), we translate TabFact statements into corresponding `pandas` queries, explicitly encoding the logical reasoning required for verification. We further refine *PanTabFact* through automated error correction to ensure syntactical validity and execution robustness. We then fine-tune the DeepSeek-coder-7B-instruct-v1.5 (Guo et al., 2025) model on *PanTabFact*, effectively distilling structured reasoning from DeepSeek-Chat, which has 671B parameters. Despite the reduced scale, our 7B-parameter model achieves on-par performance with DeepSeek-Chat in fact verification and strong generalization to unseen tabular structures.

To evaluate the generalization ability of our method, we conduct OOD experiments on *WikiFact*, a dataset we derived from WikiTableQuestions (Pasupat and Liang, 2015). Since this dataset is originally designed for question answering, we convert question-answer pairs in the test set into fact-checking claims to match our verification setup. Without any additional fine-tuning on WikiTableQuestions, our model, trained on *PanTabFact*, achieves **84.72%** accuracy on *WikiFact*, exhibiting strong robustness to unseen tabular formats and domain shifts, significantly outperforming all other baselines.

Beyond fact verification, we extend our structured approach to tabular question answering, where the goal is to extract precise answers rather than classify claims as true or false. To achieve this, we construct *PanWiki*, a dataset derived from WikiTableQuestions, by converting each question into a corresponding `pandas` query using DeepSeek-Chat. This dataset consists of 1,200 training examples, ensuring each query correctly retrieves the expected answer from the table. We fine-tune the DeepSeek-coder-7B-instruct-v1.5 model on *PanWiki* and evaluate it on the WikiTableQuestions test set, achieving **75.1%** accuracy, comparable to state-of-the-art methods despite the small size of the training data. This demonstrates the broader potential of structured representation learning for tabular reasoning, extending its utility beyond fact verification.

1.1 Contributions and Paper Organization

Our contributions are as follows:

- Execution-Based Fact Verification: We introduce *RePanda* (Reason with Pandas), a method that translates natural language claims into executable `pandas` queries, ensuring **interpretable** fact verification. Unlike black-box classifiers, *RePanda* explicitly encodes the reasoning process, allowing users to inspect, validate, and debug fact-checking decisions through executable queries (Section 3).
- *PanTabFact*: A Structured Fact-Checking Dataset: We construct *PanTabFact*, an augmented version of TabFact where each claim is paired with a `pandas` query generated using DeepSeek-Chat (Section 3).
- Strong OOD Generalization: We derive *WikiFact* by converting question-answer pairs from the WikiTableQuestions test set into factual claims for fact verification. Without any additional fine-tuning, *RePanda* achieves **84.72%** accuracy on *WikiFact*, surpassing state-of-the-art methods in out-of-distribution settings (Section 4).
- Extending *RePanda* to Question Answering with *PanWiki*: We introduce *PanWiki*, a dataset where questions from WikiTableQuestions are converted into `pandas` queries for structured question answering. Fine-tuning on *PanWiki*, *RePanda* achieves **75.1%** accuracy, demonstrating its applicability beyond fact verification (Section 4).

The rest of this paper is organized as follows: Section 2 reviews prior work. Section 3 details dataset construction, model architecture, and training. Section 4 presents experimental results, and Section 5 concludes the paper.

2 Related Work

2.1 Fact Verification with Tabular Data

Fact verification over tabular data has been extensively studied, with datasets like TabFact (Chen et al., 2019) serving as key benchmarks for evaluating models' ability to verify claims about structured data. Early methods relied on sequence-based models such as Table-BERT (Chen et al., 2019), which linearized tables before applying a pre-trained transformer for classification. However, these methods struggled with complex numerical reasoning and lacked interpretability.

More advanced approaches, such as TAPAS (Herzig et al., 2020) and TAPEX (Liu et al., 2021), incorporated table-aware pretraining to improve structured data comprehension. TAPAS extended BERT with table-specific positional embeddings, while TAPEX introduced pretraining over table-based tasks, treating table-based reasoning as a weakly supervised semantic parsing task (Yin et al., 2020). However, both TAPAS and TAPEX function as black-box models, making their decision-making process difficult to interpret, as they do not explicitly provide reasoning steps for their fact-checking predictions. PASTA (Gu et al., 2022), focused on sentence-table cloze pretraining, aiming to teach models table operations such as filtering, aggregation, and comparison. While PASTA improves structured reasoning, it relies on a predefined set of operations, which may limit its applicability to more complex or novel table structures that require reasoning beyond these fixed operations.

Our approach differs by explicitly translating claims into executable `pandas` queries, ensuring transparent and verifiable fact verification. Unlike previous models, our method explicitly encodes reasoning steps, making the verification process both interpretable and executable. Importantly, rather than relying on dataset-specific patterns, our approach focuses on translating claims into structured `pandas` queries, enabling it to generalize more effectively to unseen tables and diverse tabular formats.

2.2 Structured Representation Learning for Tables

Recent research has explored improving structured reasoning by integrating execution-based frameworks. Program-driven methods, such as ProgV-GAT (Yang et al., 2020), employ graph neural networks (GNNs) to capture logical relationships within tables, while ReasTAP (Zhao et al., 2022) applies symbolic reasoning to enhance table comprehension. Additionally, models such as StructGPT (Jiang et al., 2023) and Struct-X (Tan et al., 2024) encode structured data using graph-based attention mechanisms, but often require complex architectures.

In contrast, our method leverages `pandas` queries, which naturally define structured table operations (e.g., filtering, aggregation, and row selection) and enable direct execution for fact verification. This aligns with trends in tool-augmented reasoning, where models generate structured outputs (such as SQL, Python scripts, or execution traces) to improve interpretability (Yao et al., 2023; Wei et al., 2022). Furthermore, we evaluate our approach in out-of-distribution (OOD) settings using WikiTableQuestions (Pasupat and Liang, 2015), demonstrating strong generalization without additional fine-tuning.

2.3 Question Answering over Tabular Data

While fact verification focuses on binary classification (entailed vs. refuted), table-based question answering (QA) presents additional challenges, requiring compositional reasoning over structured data (Chen et al., 2020). Methods such as TAPEX (Liu et al., 2021) model QA as an SQL generation task, while TabLaP (Wang et al., 2024a) treats LLMs as planning agents, generating Python-based execution plans to improve numerical reasoning.

Chain-of-Table (Wang et al., 2024b) extends Chain-of-Thought prompting to tabular settings, guiding LLMs through step-by-step execution of table transformations. Similarly, SynTQA (Zhang et al., 2024) leverages text-to-SQL conversion for structured QA, but still struggles with interpretable reasoning steps.

Our method extends fact verification to QA by generating executable `pandas` queries, demonstrating that structured representation learning enhances both fact-checking and QA performance. Despite training on only 1,200 QA pairs, our approach achieves competitive results compared to state-of-the-art QA models, highlighting the effectiveness of structured execution in table-based reasoning.

3 Method

3.1 Problem Formulation

Given a structured table $\mathcal{T}$ and a natural language statement s, the goal of tabular fact verification is to determine whether s is *entailed* or *refuted* based on $\mathcal{T}$. Instead of directly classifying statements, we introduce a structured reasoning approach by translating s into an executable pandas query q_s. The execution result of q_s on $\mathcal{T}$ provides a verifiable, interpretable decision process for fact verification.

Formally, we define a function f_θ, parameterized by a language model, that maps a statement to a pandas query:

$$q_s = f_\theta(s, \mathcal{T}) \tag{1}$$

The execution of q_s on $\mathcal{T}$ produces a verification result to classify s as *entailed* or *refuted*.

3.2 Dataset Construction

We construct two datasets to train our model for fact verification and question answering: *PanTabFact* for fact-checking and *PanWiki* for tabular QA.

3.2.1 *PanTabFact*: A Fact-Checking Dataset

PanTabFact is a structured dataset derived from TabFact (Chen et al., 2019). Since TabFact consists of tables and annotated claims labeled as entailed or refuted but lacks explicit reasoning steps, we augment it with structured queries to enable execution-based verification. Specifically, we use the DeepSeek-Chat model to generate pandas queries corresponding to each claim, ensuring explicit reasoning for fact verification. Details on *PanTabFact* can be found in Appendix A.1.1.

Query Generation: For each statement-table pair $(s, \mathcal{T})$ in TabFact, we prompt DeepSeek-Chat to generate an equivalent pandas query q_s. The query should encode the logical operation required to verify s based on $\mathcal{T}$. Figure 1 illustrates an example from *PanTabFact*.

3.2.2 *PanWiki*: A Question-Answering Dataset

PanWiki is a dataset derived from WikiTableQuestions (Pasupat and Liang, 2015) for training *RePanda* in tabular question answering. Each question in WikiTableQuestions is augmented with a pandas query that, when executed, produces the corresponding answer from the dataset. *PanWiki* has 1200 data entries. Details on *PanWiki* can be found in Appendix A.1.2.

🏆 1952 VFL Season

home team	home team score	away team	away team score	venue	crowd
essendon	22.15 (147)	st kilda	7.9 (51)	windy hill	12000
carlton	12.16 (88)	collingwood	9.13 (67)	princes park	42662
south melbourne	10.14 (74)	melbourne	8.18 (66)	lake oval	24000
north melbourne	11.9 (75)	geelong	15.13 (103)	arden street oval	15000
richmond	13.11 (89)	footscray	5.14 (44)	punt road oval	11000
hawthorn	7.6 (48)	fitzroy	12.12 (84)	glenferrie oval	12000

Data source: TabFact Dataset
Statement: The home team that played at Lake Oval was *North Melbourne*
Label: *False*
Pandas query: `df[df['venue'] == 'lake oval']['home team'].eq('north melbourne').any()`
Pandas eval: *False*

Figure 1: An example from PanTabFact.

Query Generation: For each question-table-answer tuple $(q, \mathcal{T}, a)$ in WikiTableQuestions, we prompt DeepSeek-Chat to generate a pandas query q_q that extracts a when executed on $\mathcal{T}$.

3.2.3 Error Correction

Since model-generated queries may contain syntactical or logical errors, the training dataset creation process includes an automated error correction pipeline with three post-processing stages.

- *Logic Correction:* Verifies if the execution of the pandas query produces the expected answer. If flawed, we pass the original query and expected outcome back to original model for logical refinement. This stage is only applied in training dataset creation not in the inference phase.
- *Syntax Correction:* Iteratively refines queries that fail to execute on $\mathcal{T}$ due to runtime errors.
- *Filtering:* Queries that fail execution or do not match the ground-truth entailment label are removed, ensuring training dataset quality.

The Error Correction and Filtering steps are applied in both Fact-Checking and Question-Answering settings.

3.3 Model Framework

We fine-tune DeepSeek-coder-7B-instruct-v1.5 (Guo et al., 2025) to generate pandas queries for both fact verification and question answering. The model is trained autoregressively to generate structured queries that, when executed on a given table, provide verifiable reasoning for fact-checking or directly retrieve answers.

For fact-checking, we fine-tune the model on *PanTabFact*, where each claim in TabFact is paired with a corresponding pandas query. The model

learns to generate queries that determine whether a claim is entailed or refuted based on the table.

For question answering, we fine-tune the model on *PanWiki*, a dataset derived from WikiTableQuestions where each question is paired with a pandas query that retrieves the correct answer when executed. Unlike fact verification, where queries output Boolean values, here the queries extract the precise answer.

Training optimizes the negative log-likelihood of the correct query:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(q_t | q_{<t}, s, \mathcal{T}; \theta) \quad (2)$$

where q_t is the t-th token in the generated query and T is the query length.

3.4 Out-of-Distribution (OOD) Generalization

To assess the OOD generalization of *RePanda*, we derive *WikiFact*, a fact verification dataset, from WikiTableQuestions (Pasupat and Liang, 2015). Since WikiTableQuestions is originally designed for question answering, we transform each question-answer pair $(q, \mathcal{T}, a)$ into a factual statement s_q that asserts a as the correct answer based on $\mathcal{T}$. This enables us to evaluate *RePanda* in an unseen fact verification setting without any additional fine-tuning.

4 Experiments

In this section, we outline our experimental setup, present the key results for fact verification and question answering, and compare *RePanda* with state-of-the-art baselines. We evaluate its performance on both in-distribution (TabFact) and out-of-distribution (*WikiFact*) settings, providing a detailed comparison with existing models in OOD scenarios. Additionally, we assess the effectiveness of *RePanda* in tabular question answering.

4.1 Experimental Setup

We fine-tune DeepSeek-coder-7B-instruct-v1.5 on both PanTabFact (fact verification) and PanWiki (question answering). The model is trained in an autoregressive manner to generate pandas queries conditioned on input claims (for fact-checking) or questions (for QA) alongside tabular contexts.

4.1.1 Training

Training is conducted with the TRL library by HuggingFace (von Werra et al., 2020). we use the AdamW optimizer with a learning rate of 2e-4 and cosine learning rate scheduling. We train for 4 epochs with a batch size of 4, applying weight decay to prevent overfitting. Fact verification queries are optimized to generate Boolean outputs, while QA queries are trained to extract precise answers from tables.

4.1.2 Inference & Evaluation

During inference, the model generates a pandas query for each input, which is then executed to obtain the final verification result (for fact-checking) or extracted answer (for QA). The output is compared against the ground-truth answer:

Fact Verification Accuracy:

$$y = \mathbb{I}\left(f_\theta(s, \mathcal{T}) = GT\right) \quad (3)$$

where y is the correctness indicator, f_θ is the trained model, and GT is the expected Boolean label.

Question Answering Accuracy:

$$y = \mathbb{I}\left(f_\theta(q, \mathcal{T}) \approx a\right) \quad (4)$$

where q is the input question and a is the ground-truth answer.

Furthermore, we apply Syntax Correction at inference. If a query fails due to syntax errors, we pass the error message back into DeepSeek-coder-7B-instruct-v1.5, prompting 4 iterative refinements until a valid, executable query is obtained.

This setup allows *RePanda* to perform structured verification and answer extraction across both in-distribution and out-of-distribution tabular data.

4.2 In-Distribution Evaluation on PanTabFact

We first evaluate *RePanda* on the TabFact test set to assess its in-distribution performance. We compare *RePanda*, which generates structured pandas queries for fact verification, against several baselines to evaluate the effectiveness of execution-based reasoning. The baselines include:

Finetuned-Direct: DeepSeek-coder-7B-instruct-v1.5 fine-tuned to classify statements as entailed or refuted directly, without generating pandas queries.

ZeroShot-Pandas: A zero-shot DeepSeek-coder-7B-instruct-v1.5 model that generates pandas queries without fine-tuning.

ZeroShot-Direct: A zero-shot DeepSeek-coder-7B-instruct-v1.5 model that directly classifies

claims as entailed or refuted without structured reasoning.

Table 1 presents the accuracy of each method on the TabFact test set.

Table 1: Fact verification accuracy on the TabFact test set. *RePanda* significantly outperforms baselines, demonstrating the effectiveness of structured representation learning and knowledge transfer.

Method	Accuracy (%)
RePanda (Fact-Checking)	**84.09**
Finetuned-Direct	67.85
ZeroShot-Pandas	51.82
ZeroShot-Direct	50.76

RePanda achieves **84.09%** accuracy, significantly outperforming the direct classification baseline (Finetuned-Direct) by **16.24%**. Furthermore, it surpasses the ZeroShot-Direct model, which achieves only **50.76%** accuracy—close to random guessing—by a margin of **33.33%**. These results highlight the effectiveness of `pandas`-based structured learning, allowing *RePanda* to learn structured reasoning through execution-based fact verification while maintaining strong accuracy.

The stark contrast with ZeroShot baselines highlights the challenge of verifying tabular claims without fine-tuning, as the base model lacks prior exposure to structured data. *RePanda* improves both accuracy and interpretability by translating claims into executable `pandas` queries, explicitly encoding the reasoning process. Unlike black-box classifiers, *RePanda* provides a transparent verification pipeline where users can inspect the generated queries to validate the logic behind each decision. This structured approach enables an auditable fact-checking process, allowing errors or misclassifications to be traced back to specific reasoning steps, enhancing trust in the verification process.

4.3 Out-of-Distribution Generalization

To evaluate the robustness of *RePanda* beyond in-distribution fact verification, we assess its generalization on out-of-distribution (OOD) tabular data using *WikiFact* dataset. This enables us to test whether *RePanda*, trained only on *PanTabFact*, can transfer effectively to an unseen dataset without additional fine-tuning.

Performance on *WikiFact* without further Fine-Tuning. We evaluate *RePanda* on *WikiFact* without fine-tuning. The model, trained solely on *PanTabFact*, is tested directly on the transformed fact verification statements from WikiTableQuestions. Table 2 presents the accuracy results.

Table 2: Fact verification accuracy on *WikiFact* dataset without further fine-tuning.

Method	Accuracy (%)
RePanda (Fact-Checking)	**84.72**
Finetuned-Direct	74.10
ZeroShot-Pandas	59.92
ZeroShot-Direct	53.20

RePanda achieves **84.72%** accuracy on *WikiFact*, demonstrating strong generalization despite being trained solely on *PanTabFact*. It outperforms Finetuned-Direct (**74.10%**) by **10.62%** while also offering interpretability over the black-box Finetuned-Direct method. Zero-shot models perform significantly worse, with ZeroShot-Direct at **53.20%**, reinforcing the importance of knowledge transfer from DeepSeek-Chat by fine-tuning. This improvement stems from *RePanda*'s ability to learn a structured representation that generalizes beyond specific tabular distributions, allowing it to adapt effectively to unseen tables. Since all examples in *WikiFact* are factually correct, one might argue that *RePanda*'s high accuracy on *WikiFact* stems from the model consistently classifying examples as correct. However, in the next section, we demonstrate that this is not the case. *RePanda* achieves **87%** accuracy on the balanced dataset we synthesized.

Comparison with Existing Methods on OOD Data. To further evaluate OOD generalization, we compare *RePanda* with state-of-the-art tabular fact verification models.

TAPEX (Liu et al., 2021): A table-pretrained model using SQL-based execution.

TAPAS (Herzig et al., 2020): A transformer-based model optimized for table-based classification.

PASTA (Gu et al., 2022): A fact-checking model trained on synthesized sentence-table cloze tasks.

For this experiment, we randomly sample 300 instances from *WikiFact*. Since this dataset is derived from question-answer pairs, all statements are originally true based on the provided tables. To introduce refuted claims, we use DeepSeek-Chat to slightly modify each correct statement, altering its

content based on the table to generate a factually incorrect version. This results in a balanced dataset of 300 true and 300 false statements, allowing us to evaluate how effectively each model distinguishes between entailed and refuted claims in an OOD setting.

Table 3 reports the accuracy for both the original (all true) and altered (all false) claims.

Table 3: Comparison of fact verification accuracy on 300 original and 300 modified *WikiFact* statements.

Method	All False	All True	Overal
RePanda	**88.33**	**85.67**	**87.00**
TAPEX	41.00	59.33	50.16
TAPAS	55.00	65.33	60.16
PASTA	47.67	51.67	49.67

RePanda significantly outperforms prior methods, achieving **88.33%** accuracy on the altered statements and **85.67%** on the original ones. Compared to TAPEX, which achieves only 41.00% accuracy on the altered set, our model demonstrates a 47.33 percentage point improvement, highlighting its superior performance in OOD setting. Similarly, TAPAS and PASTA struggle with distinguishing between entailed and refuted statements, reinforcing the benefits of structured query-based reasoning.

These results suggest that structured reasoning through `pandas` queries provides a more robust fact verification mechanism, improving both accuracy and generalization to unseen tabular distributions. To further validate that our approach effectively captures structured reasoning, we evaluate the zero-shot performance of the much larger DeepSeek-Chat model (671B parameters) on the same fact verification tasks. As detailed in Appendix A.2, *RePanda* achieves results comparable to this significantly larger model and even surpasses it on TabFact, demonstrating successful knowledge transfer into a compact, fine-tuned model.

4.4 Application to Tabular Question Answering

To evaluate the broader applicability of our structured query generation approach, we apply *RePanda* to tabular question answering using the WikiTableQuestions dataset. Unlike fact verification, where the goal is to determine whether a claim is true or false, question answering requires extracting precise answers from tables.

We fine-tune DeepSeek-coder-7B-instruct-v1.5 on *PanWiki*, a dataset of 1,200 question-answer pairs from WikiTableQuestions, enriched with `pandas` queries generated using DeepSeek-Chat. Despite the limited training data, our method achieves performance on par with state-of-the-art models. Table 4 provides a comparative analysis.

Table 4: Comparison of tabular question answering accuracy on WikiTableQuestions. Our model uses only 1,200 training examples, significantly fewer than other methods.

Method	Accuracy (%)
TabLaP (Wang et al., 2024a)	**76.6**
SynTQA (GPT)(Zhang et al., 2024)	**74.4**
Mix SC(Liu et al., 2023)	73.6
SynTQA (RF)(Zhang et al., 2024)	71.6
CABINET(Patnaik et al., 2024)	69.1
Chain-of-Table (Wang et al., 2024b)	67.31
Tab-PoT (Xiao et al., 2024)	66.78
RePanda (Finetuned-Pandas for QA)	**75.1**

RePanda achieves **75.1%** accuracy, performing competitively with models like TabLaP and SynTQA (GPT), despite training on only 1,200 examples. In contrast, most existing approaches rely on significantly larger datasets and task-specific optimizations. These results highlight the potential of structured query generation for table-based QA, demonstrating that a `pandas`-based execution framework provides a lightweight yet effective approach to reasoning over structured data.

4.5 Ablation Study: Effect of Error Correction

To assess the impact of our automated correction pipeline, we conduct an ablation study comparing our full model with a variant that omits error correction. We evaluate performance on TabFact, *WikiFact*, and WikiTableQuestions to quantify how syntax and execution refinements contribute to accuracy.

Setup. Our error correction pipeline in inference consists of a single step:

- **Syntax Correction:** Addresses runtime execution failures by analyzing error messages and iteratively refining the query until a valid execution is obtained.

Figure 2 illustrates an example of error correction applied during training dataset creation.

🏆 1957 formula one season

race name	circuit	date	winning driver	constructor
xi gran premio ciudad de buenos aires	buenos aires	27 january	juan manuel fangio	maserati
vii gran premio di siracusa	syracuse	7 april	peter collins	lancia - ferrari
xvii pau grand prix	pau	22 april	jean behra	maserati
v glover trophy	goodwood	22 april	stuart lewis - evans	connaught - alta
x gran premio di napoli	posillipo	28 april	peter collins	lancia - ferrari
xxiii grand prix de reims	reims - gueux	14 july	luigi musso	lancia - ferrari
v grand prix de caen	caen	28 july	jean behra	brm
ix brdc international trophy	silverstone	14 september	jean behra	brm
v gran premio di modena	modena	22 september	jean behra	maserati

Data source: TabFact Dataset
Statement: 1957 formula one season *jean behra* be the only one to use the same constructor 2 race in a row
Label: *True*
Pandas query:
```
df['constructor'].eq('brm').shift().fillna(False) &
df['constructor'].eq('brm') & df['winning
driver'].eq('jean behra')
```
Pandas eval: *ERROR*
Error: ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.any() or a.all().
Corrected Pandas:
```
((df['constructor'].eq('brm') &
df['winning driver'].eq('jean
behra')).shift().fillna(False) &
(df['constructor'].eq('brm') & df['winning
driver'].eq('jean behra'))).any()
```
Corrected Pandas eval: *True*

Figure 2: An example of Error Correction.

Results. Table 5 presents accuracy results with and without corrections modules. Removing error correction results in significant performance degradation across all datasets.

Table 5: Effect of error correction on accuracy in inference time. The absence of error correction leads to a substantial drop in performance.

Dataset	No Corr.	With Corr.
TabFact	78.02	**84.09**
WikiFact	74.43	**84.72**
WQA	67.59	**75.1**

Analysis. The results emphasize the importance of error correction in structured query generation. Without it, many generated queries fail due to syntax errors. While the underlying logic of the pandas queries is often correct, minor syntax issues—such as missing parentheses or incorrect function calls—can lead to execution failures and misclassifications. Applying error correction significantly enhances reliability by ensuring that structured reasoning remains executable and interpretable.

5 Conclusion

We introduced *RePanda*, a structured approach for tabular fact verification that translates claims into executable pandas queries, ensuring interpretable and accurate verification. To support execution-based reasoning, we constructed *PanTabFact*, an augmented version of TabFact with structured queries generated via DeepSeek-Chat and refined through automated error correction. Fine-tuning DeepSeek-coder-7B-instruct-v1.5 on *PanTabFact*, *RePanda* achieved **84.09%** accuracy on TabFact and **84.72%** on *WikiFact* without additional fine-tuning, demonstrating strong out-of-distribution (OOD) generalization.

Beyond fact verification, we introduced *PanWiki*, a structured QA dataset with 1200 data entries derived from WikiTableQuestions. Fine-tuning *RePanda* on *PanWiki*, we achieved **75.1%** accuracy in table-based QA, showcasing the broader applicability of execution-based reasoning.

Unlike black-box classifiers, *RePanda* explicitly encodes reasoning steps through executable pandas queries, ensuring transparent, verifiable, and interpretable fact-checking and question answering. By leveraging structured execution rather than implicit model predictions, *RePanda* enables users to trace and validate the reasoning behind each decision. Its strong performance across diverse tabular distributions demonstrates the effectiveness of execution-based reasoning, setting a new standard for accuracy, generalization, and reliability in tabular fact verification and QA.

6 Limitations

One limitation of our work is that we focused solely on fact verification using datasets where each entry consists of a single table. This constraint means our approach has not been evaluated on more complex cases involving multiple tables, cross-table reasoning, or hierarchical data structures. As a result, its effectiveness in scenarios requiring multi-table aggregation or relational inferences remains unexplored. Future work could extend our methodology to handle fact verification across multiple interconnected tables, improving its applicability to real-world datasets with richer relational structures.

References

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. *arXiv preprint arXiv:2010.00571*.

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. Pasta: table-operations aware fact verification via sentence-table cloze pre-training. *arXiv preprint arXiv:2211.02816*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Tianyang Liu, Fei Wang, and Muhao Chen. 2023. Rethinking tabular data understanding with large language models. *arXiv preprint arXiv:2312.16702*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. Cabinet: Content relevance based noise reduction for table question answering. *arXiv preprint arXiv:2402.01155*.

Xiaoyu Tan, Haoyu Wang, Xihe Qiu, Yuan Cheng, Yinghui Xu, Wei Chu, and Yuan Qi. 2024. Struct-x: Enhancing large language models reasoning with structured data. *arXiv preprint arXiv:2407.12522*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Yuxiang Wang, Jianzhong Qi, and Junhao Gan. 2024a. Accurate and regret-aware numerical problem solver for tabular question answering. *arXiv preprint arXiv:2410.12846*.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2024b. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Bin Xiao, Burak Kantarci, Jiawen Kang, Dusit Niyato, and Mohsen Guizani. 2024. Efficient prompting for llm-based generative internet of things. *arXiv preprint arXiv:2406.10382*.

Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. Program enhanced fact verification with verbalization and graph attention network. *arXiv preprint arXiv:2010.03084*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: synergizing reasoning and acting in language models (2022). *arXiv preprint arXiv:2210.03629*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Siyue Zhang, Anh Tuan Luu, and Chen Zhao. 2024. Syntqa: Synergistic table-based question answering via mixture of text-to-sql and e2e tqa. *arXiv preprint arXiv:2409.16682*.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. 2022. Reastap: Injecting table reasoning skills during pre-training via synthetic reasoning examples. *arXiv preprint arXiv:2210.12374*.

A Appendix

A.1 Training Dataset Creation

A.1.1 PanTabFact

To construct the training dataset, we generate pandas queries for statements in TabFact using DeepSeek-Chat. The dataset undergoes multiple correction phases to improve syntax, and logical accuracy. Table 6 summarizes the statistics at each stage.

Table 6: RePanda statistics across different correction phases. Accuracy represents the proportion of correctly classified executable queries.

Phase	Correct	Accuracy (%)
Initial Generation	73,172	79.29
Logic Correction	84,023	91.05
Syntax Correction	88,299	95.68

The initial generation phase produces many syntax errors. Logic correction refines logical inconsistencies before execution. In addition, syntax correction resolves execution failures, resulting in 88,299 valid queries (95.68% of the original TabFact dataset) in the final dataset. The prompts used for every stage of training dataset creation can be found in Table 8.

A.1.2 PanWikiQA

To construct the question-answering training dataset, we generate pandas queries for WikiTableQuestions using DeepSeek-Chat. The dataset consists of 1,200 training examples created with the instruction prompt in Table 9. The correctness of generated pandas queries is determined by whether their execution produces the exact answer given in the WikiTableQuestions dataset.

Unlike the fact-checking dataset, we did not apply any correction modules in this setting, as our goal was only to showcase that the method is also effective for question answering.

A.2 Zero-Shot Performance of DeepSeek-Chat

To assess the baseline performance of a larger instruction-tuned model in a `pandas`-based setting, we evaluated DeepSeek-Chat (zero-shot) on both TabFact and *WikiFact* in fact verification setting. The model was prompted to generate `pandas` queries corresponding to given claims, using the format outlined in Table 9 for both datasets. Since this is an inference-time evaluation, the Correct Logic module was not applied. The results of this zero-shot experiment are presented in Table 7.

Table 7: Zero-shot accuracy of DeepSeek-Chat on TabFact and *WikiFact* testsets in fact verification setting, before and after error correction.

Dataset	No Corr.	With Corr.
TabFact	73.38	**82.62**
WikiFact	78.23	**85.39**

Analysis. The results in Table 7 show that DeepSeek-Chat, a 671B parameter instruction-tuned model, demonstrates strong zero-shot fact verification capabilities when prompted to generate `pandas` queries. Notably, after applying error correction, its accuracy improves significantly, highlighting the importance of structured execution refinement. Since our training data is derived from this large model, the fact that *RePanda* achieves similar performance—and even surpasses it in the case of TabFact (84.09%)—indicates that our fine-tuning approach effectively transfers structured reasoning knowledge into a much smaller model. Furthermore, as shown in Table 2, DeepSeek-7B achieves only 59.92% accuracy when tested zero-shot on *WikiFact*, whereas *RePanda* reaches 84.72% without any fine-tuning on this dataset. This demonstrates that knowledge transfer from DeepSeek-Chat significantly enhances the structured reasoning ability of our smaller model, enabling it to generalize effectively to unseen tabular distributions.

Table 8: *Prompts used for generation and refinement of PanTabFact.*

Generation	You are a Python expert specializing in pandas. Your task is to translate the given natural language statement into a single-line pandas expression. This expression must be valid and executable to verify the truth of the statement using the provided table. Consider the following: 1. The table is represented as a pandas DataFrame named df. 2. Do not include explanations, comments, or multiline outputs. 3. Ensure the output is concise, correct, and when run outputs either True or False, and strictly in the following Json Format with a single key "PANDA": "PANDA": "<your Pandas code>"
Correct Logic	You are an expert in Python with a specialization in pandas. Your task is to verify and correct a given pandas code that translates a natural language statement into a pandas expression. The corrected pandas code must accurately evaluate the truth of the statement when applied to the given table. Requirements: 1. The table is represented as a pandas DataFrame named df. 2. The pandas code must evaluate to a boolean value (True or False) using the snippet: str(bool(eval(pandas_code))). 3. The corrected pandas code should match the truth value indicated by the provided "Label". 4. Ensure the output is concise, correct, and when run outputs either True or False, and strictly in the following Json Format with a single key "CORRECT PANDA": "CORRECT PANDA": "<your Pandas code>"
Correct Syntax	You are a Python expert specializing in pandas. Your task is to correct a pandas code that translates a given natural language statement into a pandas expression. The code, along with the specific error it contains, is provided. Your corrected pandas_code must be valid and executable by running the code snippet str(bool(eval(pandas_code))) ensuring it accurately evaluates the truth of the statement using the provided table with no errors. Make sure the pandas_code is of type boolean. Consider the following: 1. The table is represented as a pandas DataFrame named df. 2. Do not include explanations, comments, or multiline outputs. 3. Ensure the output is concise, correct, and when run outputs either True or False, and strictly in the following Json Format with a single key "CORRECT PANDA": "CORRECT PANDA": "<your Pandas code>"

Table 9: *Prompt used for generating PanWiki.*

Task	**Prompt**
Generation	You are a Python expert specializing in pandas. You are given a table, a question, and an answer. Your task is to translate the given natural language question into a single-line pandas expression. This expression, which acts like a query, must be valid and executable so that running the pandas expression will output the answer to the question. Consider the following: 1. The table is represented as a pandas DataFrame named df. 2. Do not include explanations, comments, or multiline outputs. 3. Ensure the output is concise, correct, and when run, it outputs the correct given answer, and strictly follows the Json format: {"PANDA": "<your Pandas code>"}