ITUNLP at IberLEF-PRESTA: A Zero-Shot Code Generation Approach for Question Answering over Spanish Tabular Data

Atakan Site[1,*,†], Emre Hakan Erdemir[1,†] and Gülşen Eryiğit[1]

[1]Department of Artificial Intelligence and Data Engineering, Istanbul Technical University, Istanbul, Turkey

Abstract

This paper presents our zero-shot, LLM-driven code generation approach for solving the IberLEF 2025 - PRESTA: Question Answering over Tabular Data in Spanish task. Our approach relies on a Python code generation framework that employs state-of-the-art large language models (LLMs), including OpenAI o3, Qwen3, DeepSeek-R1, DeepSeek-V3, Llama 4, to generate executable Pandas code via optimized prompting strategies. Experimental results show that different LLMs vary in their effectiveness for code generation, and our hybrid configuration achieved the highest accuracy among the seven participating teams in the shared task. Specifically, our system reached 90% accuracy on the development set and 87% on the test set, demonstrating the viability of zero-shot methods for tabular question answering.

Keywords

Tabular Question Answering, Large Language Models, Zero-Shot Code Generation, Executable Code Generation, Error Correction

1. Introduction

Question Answering (QA) is a core task in Natural Language Processing (NLP), traditionally focused on retrieving relevant information from unstructured sources. However, many real-world applications rely heavily on structured data, which often carries broader and more precise semantic representations. A prominent example of such data is tabular data, where information is organized into rows and columns with consistent feature sets. Unlike unstructured text, tabular data encodes complex, heterogeneous relationships that require complex reasoning and processing strategies. Conventional approaches to querying tabular data primarily rely on structured query languages such as SQL. While these methods offer high precision and deterministic behaviors, they struggle to generalize across semantically equivalent natural language expressions and require technical expertise. These limitations are especially evident when users query structured data using natural language, which is often ambiguous, context-dependent, and underspecified. To address these challenges, recent work has focused on developing question answering systems specifically designed for tabular data.

The task of converting natural language queries into executable logical forms is commonly referred to as semantic parsing [1, 2]. Early approaches to semantic parsing over tabular data typically relied on task-specific logical form grammars, requiring manual adaptation to each table schema [3]. While effective in narrow domains, such methods often failed to generalize to open-domain or heterogeneous tabular settings.

More recent work has shifted toward end-to-end neural architectures, particularly pre-trained transformer models, which jointly model both the natural language query and the underlying tabular structure [4]. These models remove the need for explicit logical form supervision by directly producing answers or intermediate executable forms. The emergence of large language models (LLMs) has enabled promising advances in zero-shot and few-shot tabular question answering [5]. These models demonstrate strong cross-domain generalization capabilities by leveraging in-context learning, thus reducing

IberLEF 2025, September 2025, Zaragoza, Spain

*Corresponding author.

†These authors contributed equally.

site21@itu.edu.tr (A. Site); erdemire21@itu.edu.tr (E. H. Erdemir); gulsenc@itu.edu.tr (G. Eryiğit)

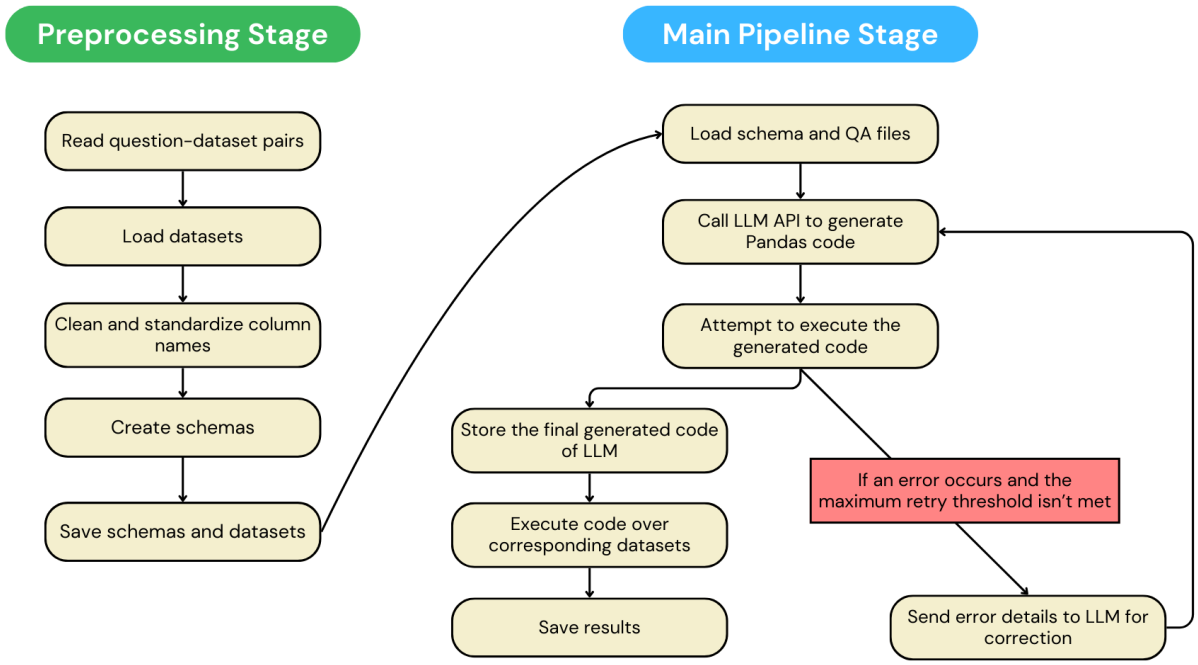

Figure 1: Our proposed framework [14] .

or eliminating the need for task-specific fine-tuning. However, they also introduce several limitations, including high computational cost, restricted context window, lack of guaranteed execution correctness and limited interpretability. To tackle these challenges and advance the development of robust tabular question-answering systems, SemEval-2025 Task 8 [6] introduced a benchmark comprising two subtasks, targeting various question formats and table domains in English. As a follow-up, the IberLEF 2025 - PRESTA task [7], organized as part of the Iberian Languages Evaluation Forum (IberLEF) [8], extends this initiative to the Spanish language through the DataBenchSPA benchmark dataset. This progression enables a broader evaluation of state-of-the-art methods in tabular question answering and offers a new testing ground for assessing the generalization capabilities of LLM-based approaches across languages and domains.

In this paper, we evaluate our previously proposed unified framework, originally developed for SemEval-2025 Task 8 [6], in the context of the IberLEF 2025 - PRESTA task [7]. This updated version integrates a range of recent state-of-the-art LLMs, including DeepSeek-R1 [9], DeepSeek-V3-0324 [10], Qwen3-235B-A22B [11], OpenAI o3 [12], and Llama-4-Maverick-17B-128E-Instruct [13], to address tabular question answering in Spanish. The system uses an inference-time prompting approach to generate executable Python code using the Pandas[1] library, which is executed in a controlled environment to ensure safe and accurate access to tabular data. To improve robustness, we incorporate an iterative error-handling mechanism: when code execution fails, the faulty code and corresponding error message are sent back to the LLM for correction, with a maximum of three iterations. This process enhances the overall reliability of the system, particularly in handling linguistically or structurally complex queries.

We observe that our hybrid configuration combining OpenAI o3 (high) and Qwen3-235B-A22B achieves the highest overall accuracy, reaching 90% on the development set and 87% on the test set. Among individual models, Qwen3-235B-A22B yields the best performance, with 86% on the development set and 85% on the test set. All code is available on our GitHub repository[2].

[1] https://pandas.pydata.org/
[2] https://github.com/erdemire21/iberlef-presta-itunlp

2. Related Work

This section provides an overview of recent developments in LLMs, with a particular emphasis on their applications in tabular question answering.

The advent of the Transformer architecture [15] has driven major breakthroughs in language modeling, enabling substantial improvements across a wide range of NLP tasks. As a result, adapting Transformer-based models to structured representations has become increasingly common. Initial efforts in this area primarily explored novel embedding techniques [16], pre-training objectives [17], and architectural adjustments [18] for tabular inputs. One dominant approach involved training Transformer-based models from scratch specifically for tabular data [4, 19]. While these models showed promise in constrained settings, they often struggled with efficiency and scalability, especially when deployed across heterogeneous domains. In practice, pre-trained language models tend to underperform when applied directly to task-specific tabular datasets without extensive adaptation.

In recent years, LLMs have revolutionized the way tabular data tasks are approached. Models such as GPT-3 [5] and Llama [20] have demonstrated strong generalization capabilities through zero-shot and few-shot learning, allowing them to perform competitively on a wide range of NLP benchmarks with minimal or no task-specific supervision. This shift has paved the way for using a single, general-purpose model across diverse tasks, including table-based reasoning, without the need for fine-tuning or specialized architectures. This trend marks a substantial departure from earlier methods that required training models specifically for each domain or task. Instead, LLMs enable task solving via in-context learning, where the model is guided by natural language instructions or few-shot examples. In the context of tabular question answering, this allows for greater flexibility, especially in low-resource or multilingual settings such as Spanish, where labeled data is scarce.

Despite their impressive capabilities, deploying LLMs in tabular QA remains challenging. One of the most prominent limitations is the limited context window, which can hinder the model's ability to handle large tables or multiple tables simultaneously. When essential portions of the table are truncated due to token limits, the model's outputs can become unreliable. Furthermore, LLMs may particularly hallucinate when reasoning requires precise alignment between the question and table content.

To address these limitations, recent research has increasingly relied on the in-context learning capabilities of LLMs. The success of LLM-based tabular question answering systems is highly dependent on how both the tabular input and natural language queries are represented. For table understanding, schema-guided prompts and exemplars are commonly employed to help the model attend to the structural semantics of the data. In parallel, the formulation of the query itself plays a critical role: strategies such as query decomposition have been shown to improve reasoning and interpretability capabilities [21]. Another approach is transforming queries into intermediate representations such as Python code or SQL queries, enabling structured execution [22, 23]. These techniques have collectively enabled LLMs to generalize across diverse table structures and query intents without task-specific fine-tuning.

Recently, efforts have been made to expand the scope of tabular question answering to non-English languages and more structurally complex settings. For instance, TableEval [24] introduces a multilingual benchmark that includes a strong emphasis on Chinese table understanding. Similarly, [25] investigates tabular QA in low-resource Indic languages, applying translation and cross-lingual transfer methods with multilingual encoders. In the context of Spanish, DataBenchSPA [26] offers a domain-diverse benchmark to evaluate QA models on real-world tabular data. These works emphasize the growing importance of evaluating LLM-based systems in multilingual and low-resource contexts, where linguistic variability and domain shifts can significantly affect performance. However, despite these recent advances, studies addressing tabular question answering in languages other than English remain limited. This highlights the need for systematic evaluation in diverse languages, as exemplified by the PRESTA task's focus on semantically rich, non-English tabular data.

Building on previous studies and our prior work for SemEval-2025 Task 8 [14], we extend our unified framework to evaluate its effectiveness on the IberLEF 2025 - PRESTA task. Specifically, we examine how well the system generalizes to a new language (Spanish) and benefits from more recent and powerful

LLMs. The PRESTA task introduces distinct challenges which allow us to test the robustness of our architecture. Experimental results show that integrating newer models such as Qwen3-235B-A22B within the same architecture leads to improved performance. These findings highlight not only the adaptability of the original framework but also its practical value in real-world settings where tabular data must be processed in multiple languages without task-specific fine-tuning.

3. Data

The DataBenchSPA dataset [26], used as part of the competition, contains 250 questions from 8 different domains, each containing question-answer pairs written in Spanish. For the IberLEF 2025 - PRESTA competition, DataBenchSPA was used as the basis, and a new test split was introduced specifically for the task. The exact dataset statistics are presented in Table 1. The train and development splits contain the following columns:

- **question**: The natural language question.
- **answer**: The response to the question for the task.
- **type**: The type of the answer, which can be **boolean, number, category, list[category], list[number]**.
- **dataset**: The name of the dataset from which the question is derived.

Table 1
DataBenchSPA dataset statistics (used for PRESTA task).

Split	Questions	Datasets
Train	150	6
Dev	100	4
Test (PRESTA Task)	100	10

In contrast to the train and development splits, which include the columns ‘question‘, ‘answer‘, ‘type‘, and ‘dataset‘, the test split for the PRESTA competition contains only the ‘question‘ and ‘dataset‘ columns to prevent data leakage and ensure a fair evaluation.

Although the dataset provides structured train and development splits with detailed annotations, this study did not utilize these data for training, as we preferred a zero-shot approach that does not involve fine-tuning.

4. System Overview

Our approach involves two main steps in providing an answer to questions over tabular data: preprocessing and then code generation and execution. The complete workflow, originally developed for SemEval-2025 Task 8 [14] is illustrated in Figure 1.

4.1. Preprocessing

Our preprocessing steps include obtaining the given questions and datasets from the competition platform, followed by a series of normalization and standardization techniques, and finally, the creation of a schema for each dataset for LLM prompting. Each dataset is transformed using a set of rules applied to column names. First, Spanish special characters (e.g., accented vowels) in column names are normalized to their English counterparts, following the mappings shown in Table 2. Second, all spaces and non-word characters are replaced with underscores, except for trailing special characters, which are removed. Third, all column names are converted to lowercase, and duplicate columns are renamed by appending a number to each duplicate. For example, if there are two columns named "Educación" and "Educación@", they would be transformed to "educacion" and "educacion_2", respectively.

Table 2
Spanish character normalization mappings for column names.

Spanish	English	Spanish	English
á	a	Á	a
é	e	É	e
í	i	Í	i
ó	o	Ó	o
ú	u	Ú	u
ü	u	Ü	u
ñ	n	Ñ	n

After normalization and standardization, we construct a schema for each dataset to enhance the LLM's understanding of the table structure. The schemas include each dataset's name, each column, each column's data type, five unique values from each column, and the total number of unique values that the column contains. The example values are limited to a total of 100 characters to avoid excessive verbosity and potential token overload. Examples of the constructed schemas can be found in Appendix A (e.g., see the schema for the Dormir dataset in Appendix A.1). Note that the schemas presented in the appendix have been truncated for brevity due to space constraints.

4.2. Code Generation and Execution

The code generation step is performed using a prompt that includes the question, detailed instructions, and the corresponding dataset schema. Although the questions are in Spanish, prompts are constructed in English to maximize LLM performance, given their stronger capabilities in English. A detailed breakdown of the code generation prompt is provided in Appendix B. The generated code is executed in a controlled environment, where dynamic imports are extracted, and the execution output is captured in its original format. In cases where execution fails, an error-handling mechanism is triggered. The system captures the error message along with the faulty code and sends it to the LLM for automatic correction (see Appendix B.2 for the prompt structure). The LLM then generates a revised version of the code. This iterative process runs until the predefined threshold is met. If the provided code remains faulty after the maximum number of attempts, execution is terminated for that query. The execution result from the last successfully executed code is then set as the final answer for the corresponding question.

5. Experimental Setup

Our zero-shot framework was evaluated on the official development and test datasets provided for the IberLEF 2025 - PRESTA task. These datasets are derived from the DataBenchSPA benchmark [26], with a dedicated test split compiled for the PRESTA competition. Building on our findings from SemEval-2025 Task 8 [14], we selected the most recent and capable models available at the time of system development to ensure optimal performance. We report results for individual models as well as for a hybrid configuration, which achieved the highest overall accuracy.

Specifically, for the configuration labeled **'o3 (high) + Qwen3-235B-A22B'**, we employed a fallback strategy. The o3 (high) model served as the primary answer generator. For this model, an additional instruction (see Appendix B.3) was added to the prompt due to its tendency to generate overly complex code which led to subpar performance. For any question where o3 (high) failed to produce executable code (resulting in an 'Error' placeholder in its output line), we substituted the answer generated by the Qwen3-235B-A22B model for that specific question. This fallback mechanism was designed to maximize coverage and leverage the complementary strengths of both models.

Additionally, based on our previous study and preliminary experiments, we set the maximum number of error correction iterations to three, as attempts beyond this threshold rarely yielded additional

Table 3
Results on the IberLEF 2025 - PRESTA task (DataBenchSPA benchmark) across all models.

Models	PRESTA Task (DataBenchSPA)	
	Dev	**Test**
o3 (high) + Qwen3-235B-A22B	**90**	**87**
o3 (high)	85	85
DeepSeek-V3-0324	83	79
Qwen3-235B-A22B	86	85
Llama-4-Maverick-17B-128E-Instruct	80	79
DeepSeek-R1	84	81

improvements. To evaluate system performance, we used Accuracy, the official evaluation metric of the IberLEF 2025 - PRESTA task. Furthermore, we analyzed the impact of our iterative error-handling mechanism on execution reliability by measuring error reduction rates across different models. These evaluations provide insights into both models' accuracy and execution robustness in tabular question answering.

6. Results

The performance of the models is presented in Table 3. Our results demonstrate strong performance across different model configurations, with the hybrid approach achieving the best results. We observe that our hybrid configuration o3 (high) + Qwen3-235B-A22B achieves the highest accuracy, reaching 90% on the development set and 87% on the test set. Among individual models, Qwen3-235B-A22B shows strong performance with 86% accuracy on the development and 85% on the test set, while o3 (high) maintains consistent performance with 85% accuracy on both splits. On the test set, both Qwen3-235B-A22B and o3 (high) deliver the best individual model performance at 85%. DeepSeek-R1 follows with 84% accuracy on the development set and 81% on the test set. These results highlight the benefits of combining complementary models and showcase the effectiveness of recent LLMs in Spanish tabular question answering tasks. Notably, while DeepSeek-R1 was our best-performing model in SemEval-2025 Task 8, the improvements observed in this study reflect how advancements in LLMs directly scale our system's performance. Furthermore, the results confirm the feasibility of applying our framework to non-English tabular question answering, underscoring its adaptability across languages.

Moreover, in the official evaluation for the IberLEF 2025 - PRESTA task, our best-performing hybrid configuration achieved first place among 7 competitors. These results further validate the effectiveness of our approach in zero-shot tabular question answering. At the time of this paper's submission, due to limited details on other solutions, we were unable to evaluate our performance relative to other zero-shot systems in the PRESTA competition. However, our manual inspection suggest that the test datasets are more challenging than the development set. The variation in performance across test conditions reveals differing levels of generalization among models, with our hybrid configuration consistently delivering robust results on both datasets.

In addition, as shown in Table 4, our error-handling mechanism substantially reduces the number of execution errors, demonstrating not only its effectiveness but also its necessity for ensuring reliable execution. It should be noted that there appears to be a general trend between the initial error rate and accuracy, where models achieving higher accuracy often generate less faulty code initially, though some exceptions can be observed (e.g., DeepSeek-R1 on the development set had few initial errors but was not the top performer). This suggests that while better-performing models may inherently produce more reliable code, the error correction mechanism remains crucial for overall execution efficiency and robustness.

To analyze error patterns and the impact of our correction mechanism in greater detail, we grouped errors into three main categories: Runtime, Degenerate Loop, and Syntax. Notably, the Runtime category

Table 4
Error progression on the IberLEF 2025 – PRESTA task (DataBenchSPA benchmark) across all models. The numbers indicate errors after 0, 1, and 2 correction attempts respectively.

Models	Dev	Test
o3 (high)	$4 \rightarrow 3 \rightarrow 3$	$6 \rightarrow 4 \rightarrow 2$
DeepSeek-V3-0324	$10 \rightarrow 1 \rightarrow 1$	$12 \rightarrow 3 \rightarrow 1$
Qwen3-235B-A22B	$5 \rightarrow 0 \rightarrow 0$	$4 \rightarrow 0 \rightarrow 0$
Llama-4-Maverick-17B-128E-Instruct	$20 \rightarrow 5 \rightarrow 5$	$8 \rightarrow 3 \rightarrow 2$
DeepSeek-R1	$1 \rightarrow 0 \rightarrow 0$	$9 \rightarrow 6 \rightarrow 6$

includes diverse errors such as KeyError and ValueError, but for simplicity, we report them under a single label. Our findings also indicate that some errors transform into different types across iterations.

We define Degenerate Loop errors as cases where an LLM repeatedly generates identical or nearly identical output sequences, continuing indefinitely until it reaches its maximum token limit.

Table 5 presents the distribution of error types across models and iterations. Results show that most initial failures are due to Runtime errors, while Syntax and Loop errors are less frequent but may persist across multiple correction attempts. Specifically, initial Syntax errors were most prominent for the Llama-4-Maverick-17B-128E-Instruct model on the development set, and were also present for Qwen3-235B-A22B. The DeepSeek models and o3 (high) did not exhibit Syntax errors in their initial generation. Degenerate Loop errors were initially observed for DeepSeek-R1 and Qwen3-235B-A22B on the development set, while DeepSeek-V3-0324, Llama-4-Maverick-17B-128E-Instruct, and o3 (high) did not encounter this error type in their first attempts. As shown in Figures 2 and 3, while the specific 'Degenerate Loop' error type is often resolved in subsequent iterations (as also indicated by Table 5 where loop counts tend to decrease), some queries initially exhibiting this error may still ultimately result in failure. This can occur if the loop is replaced by a different, unresolvable error type. These instances, however, represent a very small fraction of the total errors encountered.

Finally, Figure 2 provides an overview of error resolution across iterations, showing that most runtime errors are resolved within the first two attempts. Figure 3 further breaks down specific error types, such as FileNotFoundError, KeyError, and NameError, offering a more fine-grained view.

7. Conclusions

In conclusion, this paper presented the solution developed by the ITUNLP group for the IberLEF 2025 - PRESTA task, addressing the tabular question answering problem in zero-shot settings. Our method demonstrated strong performance, with the hybrid configuration of o3 (high) + Qwen3-235B-A22B achieving 90% accuracy on the development set and 87% on the test set. Our system ranked first among 7 competitors in the PRESTA competition. As these systems may have employed fine-tuning or few-shot learning techniques, further analysis will be possible once the working notes papers from the PRESTA shared task are published.

Importantly, our findings highlight the viability of deploying a single, unified framework across different languages, emphasizing the critical role of multilingual generalization in practical, real-world applications. We also believe that integrating more recent models such as Qwen3-235B-A22B into our existing architecture would lead to even better performance on earlier benchmarks, including our previous submission for SemEval-2025 Task 8. This demonstrates the scalability of our framework as LLMs continue to evolve.

Finally, our study reaffirms the value of zero-shot approaches: unlike fine-tuned systems that are often limited to specific tasks or domains, our method remains broadly applicable, language-agnostic, and easily extensible. In future work, we aim to extend this framework to handle more complex scenarios such as multi-table reasoning, thereby further enhancing its applicability beyond current benchmarks.

Declaration on Generative AI

During the preparation of this work, the author(s) used OpenAI GPT-4o in order to paraphrase, reword, and spelling check. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] J. M. Zelle, R. J. Mooney, Learning to parse database queries using inductive logic programming, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2, AAAI'96, AAAI Press, 1996, p. 1050–1055.

[2] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on Freebase from question-answer pairs, in: D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, S. Bethard (Eds.), Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1533–1544. URL: https://aclanthology.org/D13-1160/.

[3] P. Pasupat, P. Liang, Compositional semantic parsing on semi-structured tables, 2015. URL: https://arxiv.org/abs/1508.00305. arXiv:1508.00305.

[4] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, J. Eisenschlos, Tapas: Weakly supervised table parsing via pre-training, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020. URL: http://dx.doi.org/10.18653/v1/2020.acl-main.398. doi:10.18653/v1/2020.acl-main.398.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, 2020. URL: https://arxiv.org/abs/2005.14165. arXiv:2005.14165.

[6] J. Osés-Grijalba, L. A. Ureña-López, E. Martínez Cámara, J. Camacho-Collados, SemEval-2025 task 8: Question answering over tabular data, in: Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025), Association for Computational Linguistics, Vienna, Austria, 2025.

[7] J. Osés-Grijalba, L. A. Ureña-López, E. M. Cámara, J. Camacho-Collados, Overview of PRESTA at IberLEF 2025: Question Answering Over Tabular Data In Spanish, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2025), co-located with the 41st Conference of the Spanish Society for Natural Language Processing (SEPLN 2025), CEUR-WS. org, 2025.

[8] J. Á. González-Barba, L. Chiruzzo, S. M. Jiménez-Zafra, Overview of IberLEF 2025: Natural Language Processing Challenges for Spanish and other Iberian Languages, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2025), co-located with the 41st Conference of the Spanish Society for Natural Language Processing (SEPLN 2025), CEUR-WS. org, 2025.

[9] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, et al., Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL: https://arxiv.org/abs/2501.12948. arXiv:2501.12948.

[10] DeepSeek-AI, Deepseek-v3 technical report, 2024. URL: https://arxiv.org/abs/2412.19437. arXiv:2412.19437.

[11] Q. Team, Qwen3 technical report, 2025. URL: https://arxiv.org/abs/2505.09388. arXiv:2505.09388.

[12] OpenAI, Openai o3 technical chart, https://openai.com/index/introducing-o3-and-o4-mini/, 2025. Accessed: 2025-06-06.

[13] M. AI, LLaMA 4 - Maverick 17B 128E Instruct, https://huggingface.co/meta-llama/Llama-4-Maverick-17B-128E-Instruct, 2025. Accessed: 2025-06-06.

[14] A. Site, E. H. Erdemir, G. Eryigit, ITUNLP at Semeval-2025 task 8: Question-answering over tabular data: A zero-shot approach using llm-driven code generation, in: Proceedings of the 19th

International Workshop on Semantic Evaluation (SemEval-2025), Association for Computational Linguistics, Vienna, 2025.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. URL: https://arxiv.org/abs/1706.03762. arXiv:1706.03762.

[16] P. Yin, G. Neubig, W. tau Yih, S. Riedel, Tabert: Pretraining for joint understanding of textual and tabular data, 2020. URL: https://arxiv.org/abs/2005.08314. arXiv:2005.08314.

[17] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, D. Zhang, Tuta: Tree-based transformers for generally structured table pre-training, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, ACM, 2021. URL: http://dx.doi.org/10.1145/3447548.3467434. doi:10.1145/3447548.3467434.

[18] X. Huang, A. Khetan, M. Cvitkovic, Z. Karnin, Tabtransformer: Tabular data modeling using contextual embeddings, 2020. URL: https://arxiv.org/abs/2012.06678. arXiv:2012.06678.

[19] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, J.-G. Lou, Tapex: Table pre-training via learning a neural sql executor, 2022. URL: https://arxiv.org/abs/2107.07653. arXiv:2107.07653.

[20] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, 2023. URL: https://arxiv.org/abs/2302.13971. arXiv:2302.13971.

[21] B. Yang, C. Tang, K. Zhao, C. Xiao, C. Lin, Effective distillation of table-based reasoning ability from LLMs, in: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, N. Xue (Eds.), Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 5538–5550. URL: https://aclanthology.org/2024.lrec-main.492/.

[22] Y. Cao, S. Chen, R. Liu, Z. Wang, D. Fried, API-assisted code generation for question answering on varied table structures, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 14536–14548. URL: https://aclanthology.org/2023.emnlp-main.897/. doi:10.18653/v1/2023.emnlp-main.897.

[23] S. Zhang, A. T. Luu, C. Zhao, SynTQA: Synergistic table-based question answering via mixture of text-to-SQL and E2E TQA, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2024, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 2352–2364. URL: https://aclanthology.org/2024.findings-emnlp.131/. doi:10.18653/v1/2024.findings-emnlp.131.

[24] J. Zhu, J. Wang, B. Yu, X. Wu, J. Li, L. Wang, N. Xu, Tableeval: A real-world benchmark for complex, multilingual, and multi-structured table question answering, 2025. URL: https://arxiv.org/abs/2506.03949. arXiv:2506.03949.

[25] V. Pal, E. Kanoulas, A. Yates, M. de Rijke, Table question answering for low-resourced Indic languages, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 75–92. URL: https://aclanthology.org/2024.emnlp-main.5/. doi:10.18653/v1/2024.emnlp-main.5.

[26] J. O. Grijalba, L. A. U. López, J. Camacho-Collados, E. M. Cámara, Towards quality benchmarking in question answering over tabular data in spanish, Proces. del Leng. Natural 73 (2024) 283–296. URL: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6617.

A. Example Schemas

A.1. ES_02_40dB_Dormir

```
Column Name: identificador, Data type -- uint16, -- Example values: 1, 2,
    3, 4, 5, Total unique elements: 2000
Column Name: sexo, Data type -- category, -- Example values: Hombre,
    Mujer, Total unique elements: 2
Column Name: edad_recodificada, Data type -- category, -- Example values:
    55-64, 65+, 35-44, 45-54, 25-34, Total unique elements: 6
Column Name: tamano_de_habitat, Data type -- category, -- Example values:
    10.001 - 20.000, 20.001 - 50.000, < 10.000, 100.001 - 500.000, Total
    unique elements: 6
Column Name: provincia, Data type -- category, -- Example values:
    Albacete, Ciudad Real, Girona, Asturias, Granada, Total unique
    elements: 50
Column Name: comunidad_autonoma, Data type -- category, -- Example values:
    Castilla - La Mancha, Catalu a, Asturias, Principado de, Total unique
    elements: 17
Column Name: nivel_educativo_del_encuestado, Data type -- category, --
    Example values: Licenciatura, Grado - 2   Ciclo (Universitarios,
    Licenciados superiores, Facultades, Escuelas t cn..., Total unique
    elements: 9
Column Name: regimen_laboral_del_encuestado, Data type -- category, --
    Example values: Cuenta ajena (p.e: empleado), Cuenta propia (p.e:
    aut nomo o empresario), Total unique elements: 2
Column Name: profesion_del_encuestado, Data type -- category, -- Example
    values: Mando IntermedioEmpleado a nivel medio de Empresas
    Administraci n P blica u Oficiales Ej rcito Oc..., Total unique
    elements: 24
Column Name: clase_social, Data type -- category, -- Example values: A1,
    A2, C, E1, D, Total unique elements: 7
Column Name: tamano_de_habitat_recodificada, Data type -- category, --
    Example values: M s de 10.000, Menos de 10.000, Total unique elements:
    2
Column Name: clase_social_recodificada, Data type -- category, -- Example
    values: Alta / Media Alta, Media, Media baja / Baja, Total unique
    elements: 3
Column Name: situacion_laboral_recodificada, Data type -- category, --
    Example values: Trabaja, Pensionista, Estudiante, Parado, Trabajo del
    hogar no remunerado, Total unique elements: 5
Column Name: educacion_recodificada, Data type -- category, -- Example
    values: Tercer grado, segundo ciclo y m s, Tercer grado, primer ciclo,
    Total unique elements: 5
Column Name: ponderacion, Data type -- float64, -- Example values:
    1.0562153017104814, 0.9891829936759208, 1.1776869300720032, Total
    unique elements: 1419
Column Name: edad, Data type -- uint8, -- Example values: 58, 66, 42, 70,
    37, Total unique elements: 67
```

Listing 1: Schema for ES_02_40dB_Dormir (truncated for brevity).

A.2. ES_04_CEA_Barometro_Andaluz_Septiembre_2023

```
Here are the columns for the dataset
Column Name: identificador_de_encuesta, Data type -- uint16, -- Example
    values: 1, 2, 3, 4, 5, Total unique elements: 3600
Column Name: peso, Data type -- float64, -- Example values:
    1.2500192224089317, 0.8259640766746262, 1.201067534371049, Total unique
    elements: 8
Column Name: provincia, Data type -- category, -- Example values: Sevilla,
    Almer a, M laga, Ja n, C diz, Total unique elements: 8
Column Name: tamano_de_habitat, Data type -- category, -- Example values:
    De 10.001 a 20.000 habitantes, De 50.001 a 100.000 habitantes, Total
    unique elements: 5
Column Name: sexo, Data type -- category, -- Example values: Hombre,
    Mujer, Total unique elements: 2
Column Name: edad, Data type -- uint8, -- Example values: 46, 66, 29, 19,
    73, Total unique elements: 70
Column Name: edad_cod, Data type -- category, -- Example values: 45-54,
    65-74, 25-34, 18-24, 35-44, Total unique elements: 7
Column Name: conoce_juanma_moreno, Data type -- category, -- Example
    values: S  , No, Total unique elements: 2
Column Name: valoracion_juanma_moreno_1_10, Data type -- float64, --
    Example values: 7.0, 8.0, 4.0, 2.0, 6.0, Total unique elements: 10
Column Name: conoce_juan_espadas, Data type -- category, -- Example
    values: S  , No, Total unique elements: 2
Column Name: valoracion_juan_espadas_1_10, Data type -- category, --
    Example values: 1, 6, 3, NS, 7, Total unique elements: 12
Column Name: conoce_manuel_gavira, Data type -- category, -- Example
    values: S  , No, Total unique elements: 2
Column Name: valoracion_manuel_gavira_1_10, Data type -- float64, --
    Example values: 2.0, 4.0, 1.0, 5.0, 3.0, Total unique elements: 10
Column Name: conoce_inmaculada_nieto, Data type -- category, -- Example
    values: S  , No, Total unique elements: 2
Column Name: valoracion_inmaculada_nieto_1_10, Data type -- float64, --
    Example values: 2.0, 5.0, 9.0, 6.0, 8.0, Total unique elements: 10
Column Name: conoce_jose_ignacio_garcia, Data type -- category, -- Example
    values: S  , No, Total unique elements: 2
```

Listing 2: Schema for ES_04_CEA_Barometro_Andaluz_Septiembre_2023.

B. Code Generation Prompts

B.1. Pandas Code Generation without Error Handling

Natural Language to Python Code with Pandas

Generate a python code to answer this question: `{question}` that strictly follows the instructions below:

The code should return a print statement with the answer to the question.
The code should leave the answer be and not print anything other than the variable that holds the answer.
Please write a single Python code block that answers the following question and prints the result in one line at the end.
If the question doesn't specifically ask for it, don't use unique() or drop_duplicates() functions.

If it is a Yes or No question, the answer should be a boolean.
Do not include any explanations, comments, or additional code blocks.
Do not print intermediate steps just the answer.
Do not interact with the user.
Never display any sort of dataframes or tables.
Your output can never take more than a single line after printing and it can never be any sort of objects such as pandas or numpy objects, series etc.
Your output must be one of the following:

Boolean: True/False
Category/String: A value
Number: A numerical value
List[category/string]: ['cat', 'dog']
List[number]: [1, 2, 3]
So the outputs have to be native python

Given the dataset schema {schema}

The following python code made for pandas for the parquet file {dataset_name}.parquet reads the parquet file and running it returns the answer that is enough to answer the question `{question}`

B.2. Pandas Code Generation with Error Handling

The following prompt replaces the part after the schema is given of the previous prompt.

Natural Language to Python Code with Pandas - Error Correction

The following codes generated an error when executed:

```
{code_1}/{error_1},
{code_2}/{error_2},
... %
```

Error: {error_msg} Solve the error and provide the corrected code
The following python code made for pandas for the parquet file {dataset_name}.parquet reads the parquet file and running it returns the answer that is enough to answer the question `{question}` with the error fixed

B.3. o3-Specific Prompt Modification

For the o3 model specifically, the following additional instruction was appended to the main code generation prompt to address its tendency to generate overly complex code:

Additional Instruction for o3 Model

Generate the simplest possible pandas code that correctly answers the question. Avoid unnecessary complexity, helper functions, or overly defensive programming unless strictly required by the question's logic. Prefer direct pandas operations.

This modification was necessary because the o3 model initially generated overly complex code, which led to subpar performance. The simplified instruction significantly improved its execution success rate.

C. Error Analysis

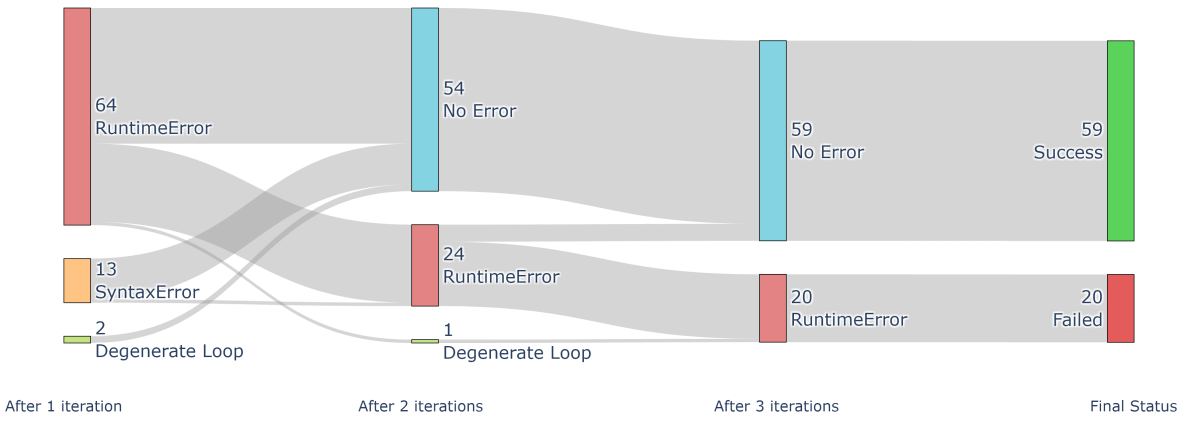

Figure 2: Error evolution and resolution across iterations (Aggregated over all models).

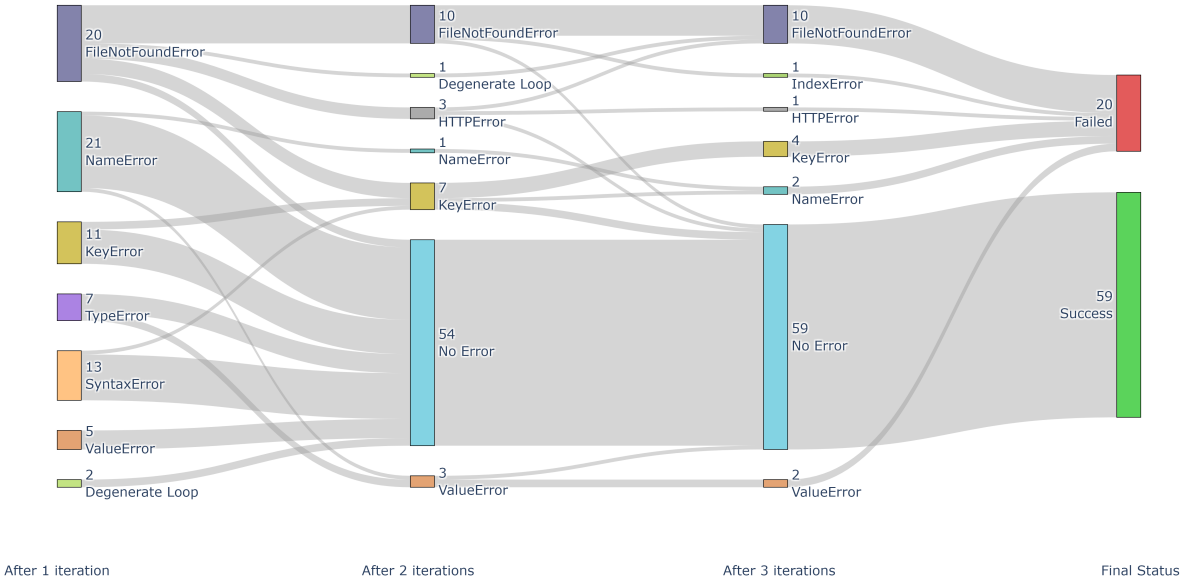

Figure 3: Fine-grained error evolution across iterations (Runtime error breakdown).

Table 5
Top error types and their distribution across iterations.

Models	Iteration	Runtime	Syntax	Degenerate Loop	Total
DeepSeek-R1 (Dev)	1	0	0	1	1
	2	0	0	0	0
	3	0	0	0	0
DeepSeek-R1 (Test)	1	9	0	0	9
	2	5	0	1	6
	3	6	0	0	6
DeepSeek-V3 (Dev)	1	10	0	0	10
	2	1	0	0	1
	3	1	0	0	1
DeepSeek-V3 (Test)	1	12	0	0	12
	2	3	0	0	3
	3	1	0	0	1
Llama-4-Maverick-17B-128E-Instruct (Dev)	1	9	11	0	20
	2	5	0	0	5
	3	5	0	0	5
Llama-4-Maverick-17B-128E-Instruct (Test)	1	8	0	0	8
	2	3	0	0	3
	3	2	0	0	2
o3 (high) (Dev)	1	4	0	0	4
	2	3	0	0	3
	3	3	0	0	3
o3 (high) (Test)	1	6	0	0	6
	2	4	0	0	4
	3	2	0	0	2
Qwen3-235B-A22B (Dev)	1	3	1	1	5
	2	0	0	0	0
	3	0	0	0	0
Qwen3-235B-A22B (Test)	1	3	1	0	4
	2	0	0	0	0
	3	0	0	0	0