UCSC at SemEval-2025 Task 8: Question Answering over Tabular Data

Neng Wan Sicong Huang Esha Ubale Ian Lane
University of California, Santa Cruz
{newan, shuan213, eubale, ialane}@ucsc.edu

Abstract

Table question answering (Table QA) remains challenging due to the varied structures of tables and the complexity of queries, which often require specialized reasoning. We introduce a system that leverages large language models (LLMs) to generate executable code as an intermediate step for answering questions on tabular data. The methodology uniformly represents tables as dataframes and prompts an LLM to translate natural-language questions into code that can be executed on these tables. This approach addresses key challenges by handling diverse table formats, enhancing interpretability through code execution. Experimental results on the DataBench benchmarks demonstrate that the proposed code-then-execute approach achieves high accuracy. Moreover, by offloading computation to code execution, the system requires fewer LLM invocations, thereby improving efficiency. These findings highlight the effectiveness of an LLM-based coding approach for reliable, scalable, and interpretable Table QA. [1]

1 Introduction

As structured data becomes increasingly prevalent across a wide range of domains—such as finance, healthcare, scientific research, and business—the task of answering questions over tabular data (Table QA) has emerged as a critical challenge in natural language processing (NLP) (Jin et al., 2022). Despite recent advancements in large language models (LLMs) and retrieval-augmented generation (RAG) (Liu et al., 2023), the inherent complexity of table structures continues to pose significant difficulties. Many tables contain nested headers, multi-row dependencies, and implicit relationships, which collectively complicate reasoning and information retrieval processes (Raja et al., 2021).

[1] Our code can be found here https://github.com/NengWan/TabularQA2024

To address these challenges, the DataBench benchmark provides a structured framework for evaluating Table QA models (Osés Grijalba et al., 2024). However, achieving high performance on DataBench remains difficult, as existing models often struggle to reason over extensive tables, handle intricate queries, and produce clear, interpretable answers. In this study, we propose a system that harnesses the coding capabilities of large language models (LLMs) to autonomously generate, validate, and execute code for extracting precise answers from designated datasets (Ye et al., 2025). By providing the LLM with a given question and an initial preview of the dataset, we prompt it to generate code that retrieves the relevant information. Furthermore, we implement both immediate and post-execution verification mechanisms to enhance the accuracy of the generated responses.

Our evaluation examines multiple models, including `LLAMA3-8b` (Grattafiori et al., 2024), `GPT-4o-mini` (OpenAI et al., 2024b), and `o1-mini` (OpenAI et al., 2024a). Although the transition to GPT-based models yields substantial improvements in test set accuracy, certain challenges persist—particularly the system's limited capacity for self-reflection and self-error-identification. This paper provides an in-depth analysis of the system architecture, presents detailed ablation studies, and evaluates model performance, thereby highlighting both the strengths and limitations of the proposed approach.

2 Background

2.1 Dataset: DataBench

For our experiments, we use DataBench, a benchmark for Question Answering over Tabular Data. It consists of structured tables paired with natural language questions and their answers. The dataset covers diverse domains such as finance, healthcare, and sports, incorporating complex queries

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 2050–2058
July 31 - August 1, 2025

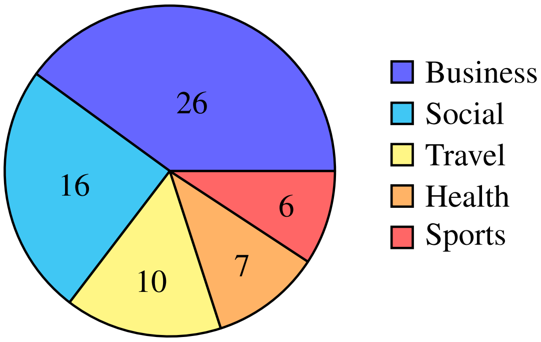

Figure 1: Proportion of datasets across different domains in the DataBench dataset.

that require aggregation, filtering, and multi-hop reasoning. Gold-standard annotations ensure reliable evaluation.

During development, we were provided with training and development sets, each containing seven columns: *question, answer, type, column used, column type, sample answer,* and *dataset*. The test set, in contrast, includes only *question* and *dataset* columns for answer generation. The train-dev datasets comprise 65 source tables, ranging from celebrity tweets, Forbes billionaire lists, and Billboard lyrics. The distribution of different dataset domains is illustrated in Figure 1

The test set consists of 15 datasets, each available in two versions: a full dataset and a lite version. Task *Test_All* contains **1468 rows**, whereas Task *Test_Lite* is a significantly smaller subset with only **18 rows** (approximately **1%** of the full dataset). This reduction in data volume significantly impacts answer accuracy, as discussed in later sections. We participated in both tracks.

2.2 Related Work

Extracting insights from complex tables is a growing challenge in data science and information retrieval. Table QA integrates structured data querying with natural language understanding, addressing difficulties in retrieving precise answers from large databases. Unlike traditional text-based QA, table QA requires reasoning over diverse structures, fine-grained cell information, and contextual dependencies (Jin et al., 2022).

Early methods relied on SQL-based models like **SQLNet** (Xu et al., 2017), which mapped natural language to SQL queries using sequence-to-sequence architectures. While effective for simple databases, these models struggled with complex multi-table schemas and schema dependencies. Neural approaches have since improved table QA by directly mapping questions to table semantics without explicit schema encoding. Transformer-based models such as **TAPAS** (Herzig et al., 2020) and **TaBERT** (Yin and Neubig, 2020) jointly encode natural language and tabular data, leveraging cell-aware and column-level embeddings.

Further advancements, including **Tuta** (Wang et al., 2021) and **TabFact** (Chen et al., 2020), enhance table representation for fact verification and comprehension, though they often require domain-specific fine-tuning. Schema-linking and retrieval-augmented generation (RAG) have also shown promise: **Zhu** (Zhu et al., 2021) improved complex query answering by integrating schema knowledge, while **Duncan** (Duncan et al., 2022) demonstrated that RAG clarifies ambiguous queries by retrieving external context.

Despite progress, challenges persist, including handling noisy data, adapting to unseen table schemas, and efficiently processing large-scale tables. Our approach seeks to address these gaps by improving generalizability, enhancing interpretability through transparent execution, and preserving data privacy via schema-based reasoning.

3 System Overview

We utilize the coding capabilities of large language models (LLMs) to generate code to query the data. Initially, we provide the model with the given question along with the first five rows of the designated dataset. In the initial prompt, we instruct the LLM to generate code capable of extracting the necessary information to produce the correct answer.

Once the code is generated, we employ two verification approaches. (i) immediate validation of the generated code, allowing the LLM to make corrections if necessary. (ii) correct the code after execution: if the execution fails, we provide the LLM with the error message, prompting it to generate a revised, executable version of the code. Finally, we obtain and output the results derived from the corrected code.

3.1 Challenges

Our objective is to develop a fully automated pipeline capable of processing a given question, comprehending its intent, generating the corresponding code, executing it, and obtaining the results. Additionally, the system incorporates an automated verification mechanism to assess the cor-

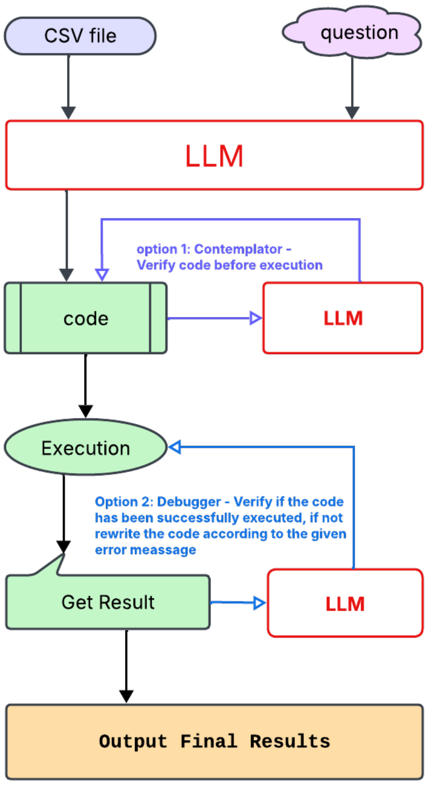

Figure 2: System overview

rectness of the generated code based on the given question.

A fundamental limitation of the system is its inability to engage in self-reflection, which has impeded further model improvement.

As detailed in the following sections, we have introduced two optional self-reflection mechanisms for the model. The first approach involves prompting the model to enter a **Contemplative** mode after code generation before the code execution, where it is explicitly instructed to assess the feasibility and correctness of the generated code. After that we feed the code into the model to get the final results.

The second approach involves an iterative refinement process, wherein, if the generated code fails to execute, the error message is fed back to the LLM. This enables the model to systematically diagnose and correct the errors until a fully executable version of the code is produced.

3.2 Methodology

The overall configuration of our system is defined by a system prompt that specifies the role and responsibilities of the LLM. In particular, the LLM is tasked with understanding the dataset and generate code that can extract answer to the question from the dataset. The prompt also delineates the required output style; for instance, the generated code should output answers as 'raw' strings.

Algorithm 1 Contemplator

Require: A question q, and a dataset preview D_5 (the first five lines of the designated dataset)
Ensure: Final answer a

Input: Question q, Dataset preview D_5
Output: Final answer a
for each q and D_5:
  LLM generate code $\rightarrow C_1$
  LLM verify C_1:
    if error: Regenerate C_2
    else: C_1
  return C^*
Execute the final corrected code C^*:
    Output the final answer a.

In addition to the system prompt, a detailed user prompt is provided. In our experiments, we evaluate two types of user prompts. In the first type, the prompt instructs the LLM to generate code that, given a specific question and the first five rows of the corresponding dataset, is capable of extracting the correct answer from the complete dataset. The prompt also includes a starter code snippet, shown in Appendix A. In the second experimental condition, the desired output format is explicitly defined (shown in Appendix B. We expect that these measures will significantly enhance the accuracy of the answers produced by the system.

Our initial approach entails returning the generated code to the LLM alongside the query:

> "Given the question, can this code produce the correct answer?"

In essence, this procedure prompts the LLM to engage in a form of self-assessment regarding its own output. The details of this methodology are presented in Algorithm 1 - the *Contemplator* .

Our second approach involves enabling the model to assess and rectify its own errors. We refer to this method as the *Debugger* approach. Essentially, the debugger prompt provides the original question along with the corresponding error message, and instructs the LLM to regenerate code that incorporates this feedback (Algorithm 2).

Algorithm 2 Debugger

Require: A question q, and a dataset preview D_5 (the first five lines of the designated dataset)
Ensure: Final answer a

Input: Question q, Dataset preview D_5
Output: Final answer a
for each q and D_5:
  LLM generate code $\rightarrow C_1$
  Execute C_1:
    while error:
      error message $\rightarrow LLM$
      Regenerate C_2
  Execute C^*:
    Output the final answer a.

3.3 Evaluation Metrics

We employed the evaluation metrics provided by the organizers. For results in boolean or numeric formats, the evaluation involves counting the number of exact matches. In the case of list-type answers, the procedure first verifies whether the lengths of the lists are identical; if so, it further assesses whether the individual elements match exactly. Ultimately, the overall accuracy score is computed by dividing the number of correct answers by the total number of answers.

3.4 Experimental Setup:

Initially, we evaluated the system using `LLAMA3-8b`; however, due to a marked improvement in performance, we promptly transitioned to `GPT-4o-mini`. While the majority of our experiments were executed with `GPT-4o-mini`, we also conducted tests using `o1-mini`, which yielded a significant enhancement in answer accuracy on the test dataset. Nonetheless, given that running `o1-mini` requires considerably more time, the competition results were produced exclusively with `GPT-4o-mini`.

4 Results

4.1 Ablation and Model Performance Analysis

We examine the effectiveness of specifying answer format. Our findings indicate that, in most cases, providing an explicit answer format results in improved accuracy. Additionally, we compared the model's performance under the contemplative mode versus the debugger mode. The results reveal that deferring code verification until an error occurs leads to better performance, whereas a double-checking approach—where the model is queried on whether it has produced the correct answer—appears to obscure the model's judgment and substantially diminish performance. In the most extreme case, this approach resulted in a 20% reduction in the performance score on the development set (from 0.909 to 0.706); see Table 2 for further details.

We observed that transitioning from `GPT-4o-mini` to `o1-mini` resulted in a significant improvement in accuracy on both test sets, with an increase of 0.09 on the full test set and 0.05 on the test lite set. Interestingly, this switch was accompanied by a reduction in accuracy on the development set.

As presented in Table 1, our models, configured with the optimal settings discussed previously, demonstrate a significant performance improvement over the state-of-the-art model reported in the original DataBench paper(Grijalba et al., 2024). The average scores across all our models improved by 13% to 28%. Notably, our model demonstrates consistently high accuracy on Boolean questions when provided with a substantial amount of table data. The highest observed accuracy, 95.3%, was achieved by GPT-4o-mini on Boolean questions within the validation set. Furthermore, the accuracy for categorical answers approaches that of boolean questions, indicating robust performance across different answer types.

Conversely, numerical answer accuracy is comparatively lower, which may be attributed to discrepancies arising from the model generating precise floating-point numbers, whereas the reference answers are rounded to two decimal places. This observation aligns with known issues related to floating-point precision and rounding errors in computational systems . Additionally, a reduction in table size correlates with a marked decline in answer accuracy, a reduction in table size is associated with a significant decline in answer accuracy, particularly affecting numerical responses, as evidenced by the performance on the Test Lite dataset.

4.2 Study on different *top_p* values

Table 3 shows the effect of varying *top_p*. *top_p* is a hyperparameter employed in nucleus sampling (Holtzman et al., 2020), a technique used for text generation in language models. It establishes a cumulative probability threshold, ensuring that only the minimal set of tokens whose combined probability is at least *top_p* is considered during sam-

Table 1: Model Accuracy by Answer Type

Prompt	Model	Avg	Boolean	Category	Number	List[Category]	List[Number]
Code Prompt 1	chatgpt3.5	63.0	52.7	73.3	75.9	56.7	56.5
Provided format Prompt (Validation set)	GPT-4o-mini	**91.3**	95.3	95.3	89.1	92.2	84.4
	o1-mini	88.1	92.2	93.8	92.2	76.6	85.9
Provided format Prompt (Test set)	GPT-4o-mini	75.1	93.8	75.7	70.5	58.3	69.2
	o1-mini	83.1	93.8	82.4	80.1	75.0	80.2
Provided format Prompt (Test Lite set)	GPT-4o-mini	78.7	71.3	39.2	16.0	25.0	15.4
	o1-mini	84.2	70.5	45.9	17.3	26.4	17.6

model	Code Correction	Prompts	Val	Test	Test_lite
4o-mini	before	Naive	0.762	0.651	0.672
4o-mini	before	Provided format	0.706	0.661	0.695
4o-mini	after	Naive	0.9	0.718	0.764
4o-mini	after	Provided format	0.909	0.743	0.795
o1-mini	after	Provided format	0.881	0.831	0.843

Table 2: Activating debugger mode after an error, rather than before, significantly improved answer accuracy. Providing answer formats for all datasets slightly boosted accuracy. (*top_p* = 0.7, temperature = 0.1)

Top_p	Val	Test	Test_lite
0.1	0.906	0.751	0.782
0.4	0.913	0.741	0.780
0.7	0.906	0.743	0.795
0.9	0.897	0.747	0.789
1.0	0.9	0.745	0.787

Table 3: Comparison on different *top_p* values for all datasets with temperature = 0.1 and with answer format provided

Temperature	Val	Test	Test_lite
0.1	0.897	0.747	0.795
0.5	0.9	0.749	0.787
1.0	0.894	0.741	0.789

Table 4: Comparison on different temperature values for all datasets with *top_p* = 0.9 and with answer format provided

pling. Our experiments, which varied the *top_p* parameter, indicate that its impact on model performance is minimal. In this section, all temperature values are set to the empirically determined optimum of 0.1.

4.3 Study on different temperature values

Table 4 shows the effect if varying temperature. Temperature is a hyperparameter that adjusts the randomness in the sampling process of language models. It operates by scaling the model's logits prior to applying the softmax function; consequently, lower temperature values yield outputs that are more deterministic and focused, whereas higher temperature values engender increased variability and creativity in the generated responses. Our empirical evaluations in Table 4 demonstrate that the model exhibits optimal and stable performance when the temperature is set to 0.5. Accordingly, in this section, all *top_p* values have been fixed at 0.9.

We can conclude that the temperature value does impact the model performance. The best temperature should be set to 0.5.

4.4 Error Analysis

One frequently encountered error arises from the inherent instability of OpenAI's API, which can result in no code being generated and an output of "None/Error." Another prevalent issue occurs when the answer is numerical: while the correct value is rounded to two decimal places, the code produced by the LLM returns a float with full precision. For instance, consider the query:

> "What is the standard deviation of the 'ISI' column?"

The correct answer is 4.55, whereas the LLM-generated answer is 4.5594771752160375

In addition, ambiguities in natural language can lead to errors, especially when the LLMs process complex or lengthy sentences. For example, consider the query:

> "List the usernames of the authors who provided a username and wrote more than 4 reviews. If there are none, answer with an empty list."

The model might focus only on the initial instruction to "list the usernames" and overlook the condition about authors who wrote more than four reviews. As a result, it may generate code that lists all usernames, ignoring the specified criteria.

To address such issues, we implemented a rewriter function designed to clarify complex questions. This approach improved performance on some intricate queries but negatively impacted simpler Boolean questions, leading to an overall decrease in accuracy.

Moreover, upon reviewing the answer comparisons, it appears that the LLM's response may sometimes be correct, even if it doesn't exactly match the expected answer; for example:

> **Query:** List the 2 players with the most steals overall.
> **LLM Answer:** {'Chris Paul', 'James Harden'}
> **Ground Truth:** {'Chris Paul', 'Russell Westbrook', 'James Harden'}

Nonetheless, certain discrepancies can be attributed to model hallucinations.

5 Conclusion

In this study, we introduced a code-generation-based approach to Table Question Answering (Table QA) using large language models (LLMs). By translating natural language questions into executable code, our method improves interpretability, reduces LLM invocations, and ensures high accuracy across diverse table formats. Evaluation on the DataBench benchmark demonstrated its effectiveness, with explicit answer formatting and deferred code validation enhancing performance. While `o1-mini` achieved the best test set accuracy, trade-offs in computational efficiency were observed. Despite challenges like ambiguous queries and occasional hallucinations. Our findings highlight the promise of LLM-driven code execution for scalable and interpretable Table QA.

References

Danqi Chen, Pengcheng Xie, Shiyu Wang, et al. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ethan Duncan, Luheng He, Michael A. Hellman, et al. 2022. Retrieval-augmented question answering over tabular data. Stanford NLP Final Project Report.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jorge Osés Grijalba, Luis Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with databench: A large-scale empirical evaluation of llms. In *Proceedings of LREC-COLING 2024*, Turin, Italy.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, et al. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

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

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023. A comprehensive evaluation of ChatGPT's zero-shot text-to-SQL capability. *arXiv preprint arXiv:2303.13547*.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024a. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13471–13488, Torino, Italy. ELRA and ICCL.

Sachin Raja, Ajoy Mondal, and C V Jawahar. 2021. Visual understanding of complex table structures from document images. *Preprint*, arXiv:2111.07129.

Bin Wang, Zhen Zhang, Lujun Hou, et al. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xinyi Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. In *2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Junyi Ye, Mengnan Du, and Guiling Wang. 2025. DataFrame QA: A universal llm framework on dataframe question answering without data exposure. In *Proceedings of the 16th Asian Conference on Machine Learning (ACML)*, pages 575–590. PMLR.

Pengcheng Yin and Graham Neubig. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2021. Multilingual semantic parsing with language-dependent schema linking. In *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

A Starter Code

The code should begin with:

```python
import pandas as pd

def get_result(csv_file):
    ...
    return result
```

B Result Output Format

The acceptable outputs include:

- Boolean values (e.g., `True`, `False`);
- A categorical value (e.g., `Flat`);
- An integer or a floating-point number (e.g., `77` or `198.7995642701525`);
- A list of categories (e.g., `[Central, Northern, Mission, Southern]`);
- A list of numbers (e.g., `[2018, 2019, 2022, 2021]`).