SPREADSHEETBENCH: Towards Challenging Real World Spreadsheet Manipulation

Zeyao Ma[1,*,‡], **Bohan Zhang**[1,*,‡], **Jing Zhang**[1,†], **Jifan Yu**[2], **Xiaokang Zhang**[1], **Xiaohan Zhang**[3], **Sijia Luo**[1], **Xi Wang**[1], **Jie Tang**[2]
[1] Renmin University of China [2] Tsinghua University [3] Zhipu.AI
https://spreadsheetbench.github.io

Abstract

We introduce SPREADSHEETBENCH, a challenging spreadsheet manipulation benchmark exclusively derived from real-world scenarios, designed to immerse current large language models (LLMs) in the actual workflow of spreadsheet users. Unlike existing benchmarks that rely on synthesized queries and simplified spreadsheet files, SPREADSHEETBENCH is built from 912 real questions gathered from online Excel forums, which reflect the intricate needs of users. The associated spreadsheets from the forums contain a variety of tabular data such as multiple tables, non-standard relational tables, and abundant non-textual elements. Furthermore, we propose a more reliable evaluation metric akin to online judge platforms, where multiple spreadsheet files are created as test cases for each instruction, ensuring the evaluation of robust solutions capable of handling spreadsheets with varying values. Our comprehensive evaluation of various LLMs under both single-round and multi-round inference settings reveals a substantial gap between the state-of-the-art (SOTA) models and human performance, highlighting the benchmark's difficulty.

1 Introduction

Today, millions of users engage with spreadsheets across various fields, including management, marketing, and finance [1, 2, 3]. Assisting users with spreadsheet manipulation can significantly reduce their workload, thereby enhancing productivity and efficiency [4, 5]. Initially, program synthesis aided spreadsheet manipulation [6, 7], followed by deep learning methods for automated manipulation [8, 9]. More recently, leveraging the powerful capabilities of LLMs [10], spreadsheet agents such as SheetCopilot[11] and SheetAgent [12] have been developed to assist people in manipulating spreadsheets via natural language instructions. These agents employ LLMs to transform instructions into solutions, typically presented as executable code for spreadsheet operations.

However, current spreadsheet-related agents [11, 12] are still far from being truly helpful assistants for automating spreadsheet tasks. Users of Copilot and similar agents find these tools useful for reducing the effort of searching online, but they do not necessarily improve task completion time or success rates [13]. A critical reason for this shortcoming is that the benchmarks [11, 12, 14] used to evaluate these agents do not truly reflect real and challenging user demands. The primary limitations could be summarized as follows.

First, these benchmarks either use the self-instruct technique [15] to synthesize queries based on a limited set of real-world spreadsheet manipulation demands [11, 12], or they rely on crowdworkers

[*] Equal Contribution.
[†] Corresponding author.
[‡] Work was done when interned at Zhipu AI.

38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks.

https://doi.org/10.52202/079017-3007

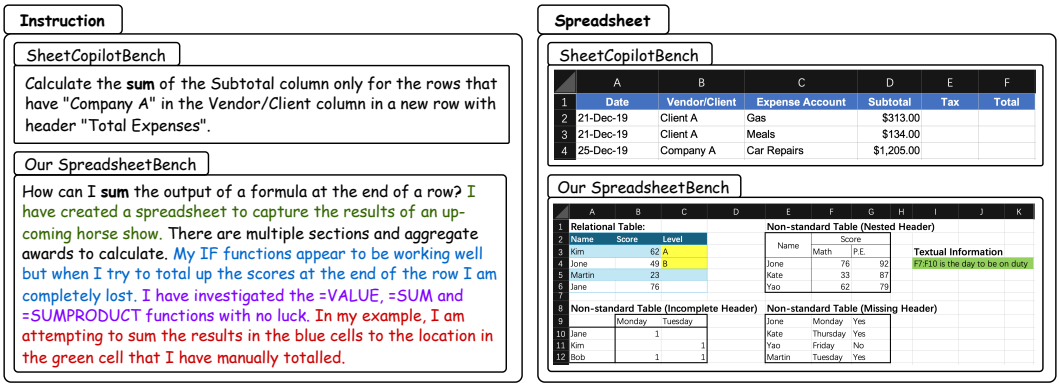


Figure 1: Comparison of previous SheetCopilotBench and our benchmark. The instructions in our benchmark are more complex, reflecting real user demands, issues encountered, previous user attempts, and an output example. The spreadsheets in our benchmark organize data more flexibly. The figure provides a summary of the various data organization types used in our spreadsheets, with three non-standard relational tables featuring nested header, incomplete header and missing header, in addition to cells containing pure textual information and non-textual elements like colors.

to manually create instructions for given spreadsheets based on a few manipulation examples [14]. As illustrated in the left part of Figure 1, real user questions collected from online forums (e.g., excelforum.com) differ significantly from synthetic queries, presenting complex demands that often require additional context, such as the users' previous attempts and encountered issues.

Second, the spreadsheets used in these benchmarks are overly simplified [11, 12], typically containing only one regular relational table, similar to the tables used in TableQA tasks [16, 17] and the tables in databases of Text2SQL tasks [18, 19]. While both spreadsheets and relational tables store tabular data in a two-dimensional structure, there are significant differences that make spreadsheets more challenging to handle. As shown in the right part of Figure 1, spreadsheets offer a flexible way of organizing data without distinct table boundaries or structures, permitting the inclusion of multiple tables in non-standard formats—such as nested, incomplete, or missing headers—and the presence of free-form text within a single cell. Additionally, spreadsheets support various non-textual elements (e.g., color, bold) compared to the relational tables in databases [18] or the CSV/JSON format tables in TableQA tasks [16, 20]. These distinct features are not fully utilized in current benchmarks.

Third, these benchmarks for LLMs typically involve a single test for each instruction, where the predicted answer, derived from the solution generated by the LLMs applied to the given spreadsheet, is compared to the ground truth answer [11, 12]. This evaluation, however, may result in false positive solutions that are tailored to the specific spreadsheet and do not generalize to other spreadsheets with varying values. A robust solution from LLMs should be able to manage multiple spreadsheets, even corner cases, as inputs, apply the solution, and consistently yield correct answers.

To address the identified need, we introduce a challenging benchmark called SPREADSHEETBENCH. Our benchmark features complex user instructions alongside spreadsheets that are flexibly organized by real-world users. These resources are gathered exclusively from four online forums for addressing Excel problems, which are highly ranked by several reputable websites including general Google Search, Feedspot.com—known for its forum search functionality—and Deskbright.com, a site focused on online education, particularly in Microsoft Excel. The instructions often require detailed contextual information for clarification, including descriptions of attempted solutions, incorrect answers received, and the issues encountered. The spreadsheets exhibit diverse formats, potentially featuring multiple tables within a single spreadsheet, non-standard tables that extend beyond conventional relational structures, cells with only textual content, and other non-textual elements.

We also propose an Online Judge (OJ)-style metric for more reliable benchmarking, adapted from coding skill assessments [21]. This metric is well-suited for spreadsheet manipulation tasks where LLMs are tasked with generating code solutions. Within our benchmark, each instruction is associated with multiple input-output spreadsheets serving as test cases, sharing a similar structure but differing in data. A code solution must pass all these test cases to ensure evaluation reliability and accuracy.

https://doi.org/10.52202/079017-3007

Conclusively, we have developed SPREADSHEETBENCH comprising 912 instructions and 2,729 test cases, with an average of three test cases per instruction. These instructions cover 10 primary manipulation categories and involve spreadsheets with tables extending beyond 100 columns and 20,000 rows. Moreover, 35.7% of the spreadsheets contain multiple tables, and 42.7% feature non-standard relational tables. This benchmark is notable for **its real-world instructions, diverse spreadsheet formats, and a comprehensive testing strategy**.

We conduct a thorough evaluation of various types of models, including commonly studied TableQA models, open-source LLMs for general tasks and coding tasks, the most advanced close-source models, and spreadsheet-specific LLMs. The performance of these models vary widely, with scores ranging from 0.05% to 23.65% according to our proposed OJ-style evaluation metric. Notably, some methods score as low as 0%, highlighting the benchmark's difficulty. The results also suggest the importance of improving coding abilities within LLMs for spreadsheet manipulation, and indicate that multi-round prompting, allowing LLMs to read spreadsheet data and error feedback from a code compiler, can increase the probability of correct responses.

2 SPREADSHEETBENCH

In this section, we begin by outlining the task formulation, followed by an introduction to the benchmark construction pipeline and the presentation of resulting benchmark statistics. Finally, we introduce the proposed OJ-style evaluation metric.

2.1 Task Formulation

We denote the dataset of a benchmark as $\mathcal{D} = \{(q_i, c_i, a_i)\}$, where q_i represents an instruction, c_i denotes a spreadsheet, and a_i is the corresponding answer of q_i on c_i. The objective of spreadsheet manipulation is to enable an LLM to generate a code-based solution s_i, apply it to q_i to obtain a predicted answer $\hat{a}_i$, which can then be compared with the ground truth answer a_i.

In this benchmark, an instruction q_i involves the modification of either several cells or the entire sheet in a spreadsheet file, which we refer to as *cell-level* and *sheet-level* manipulation, respectively.

The spreadsheet c_i stores tabular data in a two-dimensional format, offering a flexible way for data organization without a distinct table boundary or structure. A single sheet in a spreadsheet file can contain multiple tables in diverse formats, free-form textual information, and non-textual elements.

An answer a_i refers to the modified spreadsheet resulting from the application of a solution, which is expected to be a piece of code generated by LLMs. We generate code as solutions, as the goal is to manipulate spreadsheets rather than query information from them. Note that $\mathcal{D}$ does not include solutions because we allow for various solutions by LLMs, with our focus solely on the correctness of final answers after applying the solutions.

2.2 Benchmark Construction

The construction of the benchmark follows a four-stage pipeline: 1) data sourcing, 2) data filtering, 3) data formatting, and 4) test case construction, as illustrated in Figure 2. Subsequently, we detail the perturbation techniques used to prevent data leakage.

1. Data Sourcing. Our objective is to collect high-quality and representative spreadsheet manipulation questions from real-world sources. As depicted in the initial section of Figure 2, we utilize general Google Search, the specialized forum search engine Feedspot.com, and the Microsoft Excel-focused online learning platform Deskbright.com to identify four popular and regularly updated forums and blog sites—ExcelForum, Chandoo, MrExcel, and ExcelGuru—as our data sources (detailed information available in Appendix B.1). We specifically target posts that fall under categories such as *General Questions*, *Formula*, *VBA & Macros*, *Formatting*, *Pivot Table*, and *Power Query* to ensure relevance to spreadsheet manipulation.

2. Data Filtering. Since the original posts in forums are diverse with various questions, as shown in the second part of Figure 2, we select the question by four main criteria: 1) is solved, 2) solely pertains to spreadsheet manipulation, 3) is feasible and testable, and 4) is representative.

 https://doi.org/10.52202/079017-3007

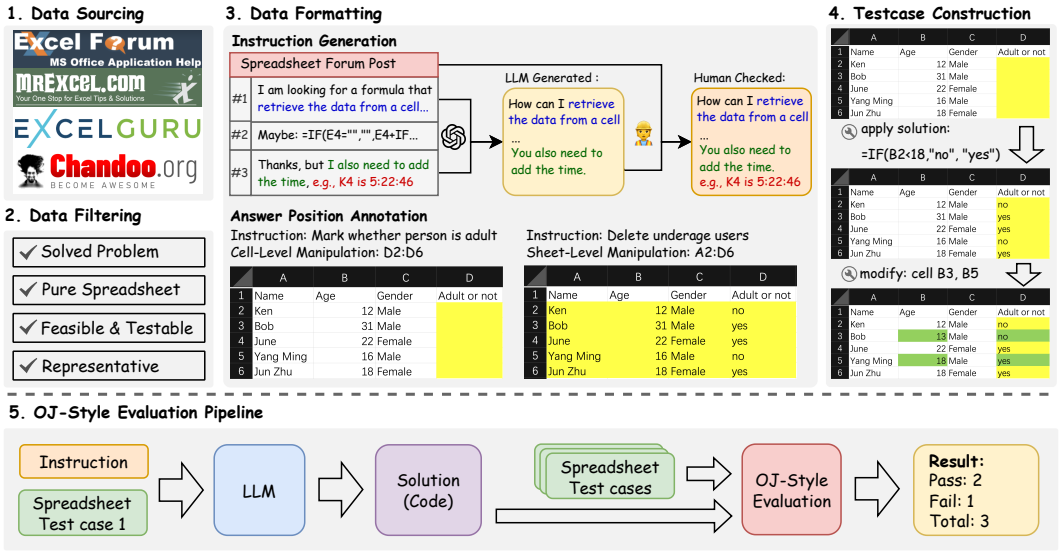


Figure 2: The benchmark construction pipeline and OJ-style evaluation of our benchmark.

Initially, we directly identify questions tagged as "solved" from all posts. For posts without explicit tags, we employ GPT-4 to determine their solved status, as patterns such as thank-you notes are often observed at the end of a post indicating the question is solved.

Next, we focus on extracting questions strictly related to spreadsheet manipulation. Most popular spreadsheet forums primarily discuss Excel-related issues involving software functions like input boxes, buttons, and forms. Unlike InstructExcel [14], our goal is to establish a software-independent benchmark centered solely on spreadsheet manipulation. Consequently, we discard all irrelevant software-specific questions using keywords such as "input boxes", "buttons", and "forms", and engage annotators to check the remaining posts.

Finally, we select questions that are both feasible and representative. To ensure feasibility, aided by annotators, we exclude posts without spreadsheet attachments, those with excessive replies, or ambiguous presentations. To identify representative posts, we filter those with high view counts and ask the annotators to retain both simple and difficult questions.

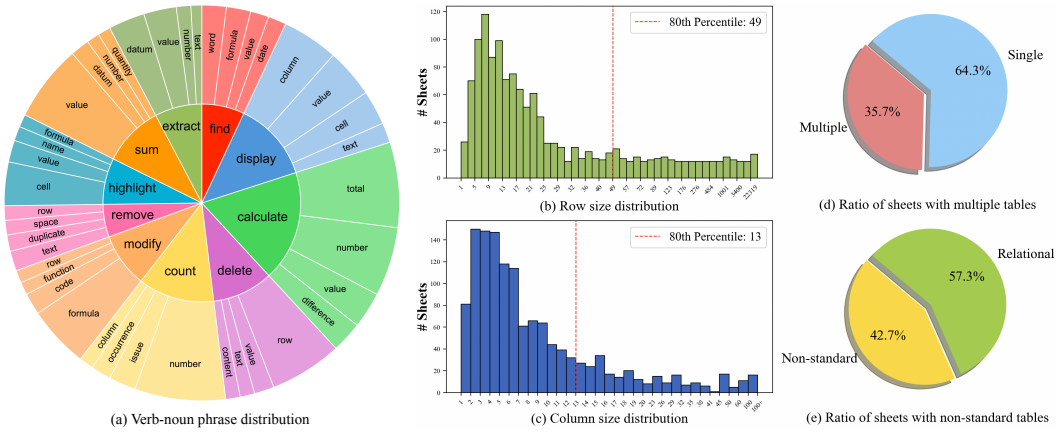
3. Data Formatting. According to the task definition outlined in Section 2.1, it is necessary to formulate the instruction, the spreadsheet, and the answer in a manner that facilitates convenient and accurate automatic evaluation.

Instruction Generation. The original posts are often cluttered, with the description of the user question dispersed across multiple replies. Therefore, we need to extract and condense these into a self-contained instruction from all replies. Initially, we utilize GPT-4 to recreate a coherent instruction from the original post. Annotators then verify this instruction to resolve any issues arising from possible incomplete context extraction from the post.

Answer Position Annotation. Regarding solution derivation, forum users typically discuss solutions in formulas or VBA code. Annotators are responsible for identifying the solution from the replies and applying it to the corresponding spreadsheet to derive the answers. However, discrepancies in the placement of answers by annotators compared to those predicted by LLMs can lead to mistaken evaluations. To mitigate this issue, we allow annotators to incorporate specific constraints in the instructions that annotate the answer position. As described in Section 2.1, answer positioning should consider both cell-level and sheet-level manipulation. For cell-level manipulations, annotators should specify exact cell positions, such as "D2:D6". For sheet-level manipulations, the entire range of the spreadsheet, such as "A2:D6", needs to be included in the instruction.

4. Test Case Construction. According to observations on online forums, it is evident that some solutions provided by forum users may be applicable only to the specific example spreadsheet provided, potentially failing when applied to other scenarios with slight variations in the spreadsheet data.

https://doi.org/10.52202/079017-3007



(a) Verb-noun phrase distribution (b) Row size distribution (c) Column size distribution (d) Ratio of sheets with multiple tables (e) Ratio of sheets with non-standard tables

Figure 3: Key statistics of SPREADSHEETBENCH.

To enhance the reliability of user-provided solutions, it is essential to create multiple spreadsheets and develop multiple test cases for each instruction. This preparation is essential for the subsequent OJ-style evaluation metric.

Specifically, for instructions that solely offer a solution without derived answers from forum posts, we derive the answer for the original provided spreadsheet by manually manipulating the input spreadsheet using the given solution presented by formulas or VBA code. The original spreadsheet, along with the derived answer, forms an input-output test case for the current instruction.

Subsequently, annotators are requested to modify the data in the original spreadsheet twice to generate two distinct spreadsheets. By applying the same solution to each of these modified spreadsheets, we can obtain two corresponding answers. It's important to note that annotators do not make casual alterations to the spreadsheet data; instead, they are instructed to concentrate on real-world corner cases (e.g., cells that should contain zeros but remain empty). Annotators are also instructed to make valid modifications consistent with the question's requirements.

Ultimately, for each instruction suitable for multiple test cases, we derive three test cases from the original single spreadsheet-answer data, as illustrated in the fourth part of Figure 2. Then $\mathcal{D} = \{(q_i, c_i, a_i)\}$ is changed to $\mathcal{D} = \{(q_i, \{(c_{ij}, a_{ij})\})\}$.

Against Data Leakage. Datasets initially obtained from online forums may be susceptible to data leakage issues, given that many LLMs are pre-trained using a vast corpus of web text [22, 23].

The above mentioned strategies, along with some new approaches, can introduce perturbations to the collected data to mitigate the data leakage problem:

Instruction Generation: The aforementioned instruction generation strategy, carried out by GPT-4 and human annotators, involves revising the original questions in the posts, thereby preventing LLMs from memorizing the original questions.

Spreadsheet Modification: The test case construction strategy outlined above involves modifying the original provided spreadsheets, preventing LLMs from memorizing the original spreadsheets.

Answer Position Changing: We also alter the position of the tabular data in the original spreadsheets and the corresponding answer in the resulting spreadsheets. By doing so, the originally provided solution from the posts cannot be directly used to derive answers with changed positions, thus preventing LLMs from memorizing the provided solution from the posts.

Software Independence. Spreadsheet manipulation is not inherently software-independent, as some functions may not be available (or may change name) changing software. We employ three methods to ensure the software independence of our benchmark as much as possible. (1) *Software-independent instruction.* We only select questions that focus on the manipulation of the value of cells in the spreadsheet and try to avoid questions involving special components or advanced operations that only exist in specific software. (2) *Software-independent spreadsheet files.* All the spreadsheet files in our dataset are in the XLSX format, which is widely supported across multiple spreadsheet

https://doi.org/10.52202/079017-3007

Table 1: Comparison between SPREADSHEETBENCH and other spreadsheet manipulation benchmarks across four aspects: data source, instructions, spreadsheets, and evaluation metrics.

Benchmark	SheetCopilotBench [11]	InstructExcel [14]	SheetRM [12]	Ours
Data Source	Self-Instruct	Manual Annotation	Self-Instruct	Forum & Blog
Instructions	221	4850	201	912
Ave. Instruction Words	27.9	9.8	-	85.7
Spreadsheet Files	29	940	25	2729
Single Sheet	26	572	-	2019
Multiple Sheet	3	368	-	710
Sheets	32	1694	83	3917
Non-standard Tables	✘	✓	✘	✓
Multiple Tables	✘	✓	✘	✓
Additional Info.	✘	✘	✘	✓
Evaluation	Exact Match (EM)	EM & Similarity	Sub-task EM	OJ-style EM
Ave. Test Cases	1	1	1	3

applications, such as Microsoft Excel, Google Sheets, LibreOffice Calc, and WPS Office. Though some spreadsheet software may not provide full support for the XLSX format, we have made an effort to exclude any content that may be incompatible with non-Excel software. (3) *Software-independent Inference.* Instead of utilizing software-dependent tools (e.g., TypeScript and the Excel API) to interact with spreadsheet files[14], we allow models to manipulate spreadsheet files in diverse ways, such as Python code, because we only compare whether the results after manipulations are correct.

Annotation Team. The construction of our benchmark requires high-quality human annotations. We employ a team of 20 individuals who specialize in Excel and have extensive experience in annotation. All team members hold a bachelor's degree and are compensated in line with market rates. For data annotation, we allocate a predetermined quantity of data to annotators on a daily basis, and the allocation is modified in accordance with local regulations regarding work hours. For data validation, we hire two experienced validators with bachelor's degrees for the first round of quality checks. Moreover, two authors who hold master's degrees perform a secondary round of quality assessments to guarantee the high quality of data.

2.3 Benchmark Statistics

Data Statistics. We have developed SPREADSHEETBENCH comprising 912 instructions and 2,729 test cases, with an average of three test cases per instruction. We analyze it and derive several observations, as depicted in Figure 3. In Figure 3(a), the instructions in our benchmark cover a broad spectrum of spreadsheet manipulation types, including find, extract, sum, highlight, remove, modify, count, delete, calculate, and display. The outer circle illustrates the objects being manipulated, showcasing the diversity of the benchmark.

Figures 3(b) and (c) display the distributions of row and column sizes in our spreadsheet files, revealing long-tailed distributions. Specifically, 80% of spreadsheets have a row size spanning from 1 to 49, and a column size spanning from 1 to 13. It is noteworthy that the row or column size in the long tail part is substantial, indicating the challenging nature of our benchmark.

Figure 3(d) demonstrates that more than one-third of spreadsheets contain multiple tables in one sheet. Figure 3(e) reveals that nearly half of the spreadsheets contain non-standard relational tables with nested, incomplete or missing headers. These observations indicate that the SPREADSHEETBENCH significantly increases the complexity of understanding and manipulating the spreadsheets.

Comparison with Previous Benchmarks. Table 1 compares SPREADSHEETBENCH with previous spreadsheet manipulation benchmarks. In addition, numerous benchmarks exist for table-related tasks. These include table question answering (TableQA)[16, 24, 25, 26], Table-to-text[27, 28], and table data analysis [29, 30, 31]. We have not included benchmarks for these table-related tasks, as their instructions are specifically tailored for QA and use CSV/JSON files as input, which significantly diverges from tasks that involve spreadsheet manipulation.

SPREADSHEETBENCH is sourced exclusively from real-world data and exhibits a higher average word count per instruction. We have collected a larger number of spreadsheet files, including both single-sheet and multiple-sheet formats in each file. Our spreadsheet files contain multiple sheets with non-standard relational tables and multiple tables within a single sheet. Real-world questions often involve additional explanations within the spreadsheet, a characteristic not present in previous benchmarks. Furthermore, we employ OJ-style evaluation metrics with three test cases per instruction.

The benchmarks SheetCopilotBench [11] and SheetRM [14] generate instructions using LLMs' self-instruct techniques, which are quite different from users' actual queries. Furthermore, the manipulated spreadsheets in these benchmarks are relatively simple, seldom featuring multiple sheets and tables, and non-standard relational tables. On the other hand, InstructExcel [14] offers a more comprehensive benchmark, particularly with complex spreadsheet files that cover various features. Even though the instructions in InstructExcel are created by annotators, they remain relatively simple. The average word count per instruction is only 9.8, compared to 85.7 in our benchmark. Additionally, InstructExcel does not include additional explanations or multiple test cases for the instructions, making it less complex and potentially less reliable than our benchmark.

2.4 Evaluation Metrics

Exact match is a commonly employed metric for spreadsheet manipulation tasks [11, 12], but it may only work for the current spreadsheets and may not be robust to variations in the data within the spreadsheet. To address this, we adopt an approach similar to the online judge system commonly used for assessing coding capability [21]. An Online Judge (OJ) style evaluation involves inputting multiple test cases into a code script and comparing each test case's output to the ground truth result. This method is more comprehensive and reliable than evaluating the code script just once. In our benchmark context, each test case is a spreadsheet file with a similar table structure but different cell values, as mentioned in Section 2.2. This allows us to evaluate a code solution more thoroughly and determine its effectiveness in handling corner cases. Figure 2 illustrates this process, where an LLM is expected to generate a solution given an instruction and a spreadsheet test case. Subsequently, the solution is applied to multiple spreadsheet test cases for OJ-style evaluation. In contrast to some benchmarks that perturb data in tables to create new data samples and perform one model inference for each data sample [32], we only require one model inference to generate a solution for all test cases of each instruction.

We establish two distinct scoring criteria to calculate the final score. The *soft restriction* adheres to the scoring principles of the OJ system from the IOI, granting partial credit when a solution only passes some test cases [33]. The calculation is as follows:

$$S_{soft} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left(\frac{1}{|T_i|} \sum_{j=1}^{|T_i|} \mathbb{1}_{r_{ij}=ACC} \right). \tag{1}$$

The *hard restriction* follows the ICPC scoring rules of the OJ system, where no partial credit is awarded [33]. The score is determined as follows:

$$S_{hard} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbb{1}_{r_{ij}=ACC, \forall j=1,2,\ldots,|T_i|}. \tag{2}$$

In the above equations, T_i represents the test cases (i.e., $\{(c_{ij}, a_{ij})\}$) of the i-th instruction, and r_{ij} indicates the result of the model's solution applied on the j-th test case, respectively. Specifically, r_{ij} is marked as Accept (ACC) if the applied solution's answer on the j-th test case (i.e., the resulting spreadsheet) match the ground truth answer. The soft restriction metric does not penalize models for failing to provide a flawless solution that addresses every common and corner cases, whereas the hard restriction metric encourages models to strive for the most perfect solution.

https://doi.org/10.52202/079017-3007

Task Subset	% of Total	Accuracy
Rows ($\leq$ 50)	75.19	**20.63**
Rows ($>$ 50)	24.81	11.50
Columns ($\leq$ 10)	65.53	**22.50**
Columns ($>$ 10)	34.47	10.51
Single Tab.	62.90	**21.12**
Multiple Tab.	37.10	13.71
Relational Tab.	55.54	**19.50**
Non-standard Tab.	44.46	16.95

Table 3: Overall soft restriction of GPT-4o on different subsets (%).

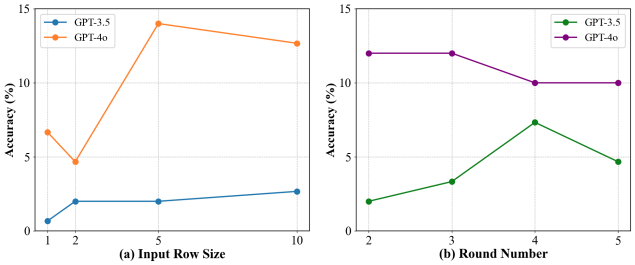


Figure 4: Impact of input row size and round number in terms of overall soft restriction.

3 Experiments

Baselines. We evaluate LLMs across five categories: (1) *TableQA models*, including fine-tuned TaPEx (based on BERT) [34], TaPas (based on BART) [35], and prompt-based Binder (GPT-3.5) [36]; (2) *Open-source code models*, including CodeQwen (7B) [37] and DeepseekCoder (33B) [38]; (3) *Open-source general models*, including Mixtral 8x7B [39] and Llama 3 (70B) [40]; (4) *Close-source models*, including GPT-3.5[1] and GPT-4o [10]; (5) *Spreadsheet-specific methods or products*, including SheetCopilot [11] and Copilot in Excel[2].

Evaluation Interface. For TableQA models, we ask the model to generate text results based on the content of the spreadsheet files. For Spreadsheet-specific methods or products, we manually evaluate these methods within their products. We only sample five percent of the data samples from our benchmark due to the costly and time-consuming process of evaluating the entire benchmark. For other models, we use Python code to modify and interact with the spreadsheet files.

Inference Setting. We evaluate LLMs under two distinct settings: 1. *Single Round*: In this mode, we present the model with the initial few rows of spreadsheet files within the prompt, allowing for only one inference. Since TableQA models can only produce textual answers, we conduct separate inferences for each test case and compare the resulting textual answers to the ground truth. For non-TableQA models, we instruct them to generate a code-based solution for all test cases of a given instruction. 2. *Multi-Round*: Building on the single-round prompt setting, we incorporate additional prompt that utilizes the ReAct technique [41] and code execution feedback [42] to enhance the accuracy of code solutions produced by LLMs over multi-round conversation. Specifically, we offer LLMs two choices: to generate code for retrieving spreadsheet file contents or to provide the ultimate solution. For the first choice, we supply code execution outcome to the subsequent round. In the second, the conversation concludes once the desired spreadsheet file is produced. In both instances, we furnish error feedback if the code fails to execute, enabling the model to refine its code in subsequent iterations. We impose a limit of five rounds for the multi-round setting.

Human Performance. We hire four annotators who are experts in Excel to obtain the human performance of our benchmark. To minimize costs, we use a subset of fifty instructions and three test cases for each instruction to conduct our human evaluation. Annotators are asked to provide formula solutions or VBA code to finish each instruction instead of filling in the answer in the target position directly. The process of human evaluation is completed within one day.

Overall Performance. The results shown in Table 2 indicate that current LLMs and spreadsheet agents are inadequate in managing complex spreadsheet manipulation tasks as required by real-world scenarios. Even the most advanced spreadsheet agent, Copilot in Excel, only achieves an accuracy of roughly 20%. GPT-4o, the SOTA LLM, scores around 17% in accuracy, aligning with Copilot in Excel's performance. Open-source LLMs significantly underperform compared to the SOTA model, likely due to their limited comprehension and coding proficiency. Binder shows the poorest results, while TaPEx and TaPas, designed specifically for TableQA, score zero across all metrics (omitted from the table). This underscores the distinction in difficulty between TableQA and spreadsheet manipulation. Overall, there is a substantial gap between existing LLMs or products and human performance produced by Excel experts, emphasizing the critical need for advancement in LLMs tailored for spreadsheet manipulation.

[1] `https://platform.openai.com/overview`
[2] `https://www.microsoft.com/microsoft-copilot/`

Table 2: Performance of representative models on SPREADSHEETBENCH (%).

Model	Soft Restriction (↑)			Hard Restriction (↑)		
	Cell-Level	Sheet-Level	Overall	Cell-Level	Sheet-Level	Overall
Binder (GPT-3.5)	1.58	0.05	1.17	0.00	0.00	0.00
CodeQwen (7B)	0.36	0.76	0.51	0.36	0.29	0.33
w / Multi-Round	1.49	7.14	3.66	0.89	6.29	2.97
DeepseekCoder (33B)	0.59	5.81	2.60	0.36	5.14	2.20
w / Multi-Round	3.15	8.76	5.31	1.96	6.86	3.85
Mixtral-8x7B	2.97	3.33	3.11	2.32	2.57	2.42
w / Multi-Round	3.39	4.67	3.88	2.32	3.71	2.85
Llama-3 (70B)	0.18	3.14	1.32	0.00	2.86	1.10
w / Multi-Round	1.13	7.90	3.74	0.71	7.14	3.18
GPT-3.5	1.31	3.99	2.34	0.71	3.13	1.64
w / Multi-Round	3.33	13.11	7.09	2.50	9.97	5.37
GPT-4o	15.03	**23.65**	18.35	**11.94**	**19.94**	**15.02**
w / Multi-Round	13.49	22.51	16.96	10.52	17.66	13.27
SheetCopilot (GPT-4)*	16.67	10.00	14.00	-	-	-
Copilot in Excel*	**23.33**	15.00	**20.00**	-	-	-
Human Performance	75.56	65.00	71.33	66.67	55.00	62.00

Alongside the performance evaluation of each model, our findings include the following insights:

(1) *Code Model vs. General Model.* Given that spreadsheet manipulation solutions primarily rely on coding, LLMs tailored for coding, such as DeepseekCoder, exhibit superior performance compared to open-source general models like Llama-3 (70B). This highlights the need to enhance coding capacities within LLMs for effective spreadsheet manipulation.

(2) *Single Round vs. Multiple Round.* The majority of LLMs experience significant improvement in their capabilities through multi-round conversation, with some models achieving nearly a tenfold increase. However, GPT-4o's performance slightly declines in a multi-round setting. This is due to GPT-4o's adherence to instructions, which leads to duplicated content when retrieving the spreadsheet, as it includes the initial spreadsheet rows already provided in the prompt. Other LLMs, unable to follow the retrieval instruction, rely on the given initial rows. We provide the performance of GPT-4o without the initial spreadsheet rows in Appendix C.

(3) *Soft Restriction vs. Hard Restriction.* The shift from soft to hard restrictions leads to a modest decrease in LLM performance, suggesting that the solutions produced by these models may not remain effective when the spreadsheet content is altered. Given that real-world spreadsheet applications often involve frequent content changes, it is crucial to develop solutions that are robust to such modifications. The introduced OJ-style evaluation metrics are well-suited to assess such robustness.

Analysis. We also conduct analysis on the factors that influence the performance of LLMs on the spreadsheet manipulation task.

(1) *Great spreadsheet complexity leads to diminished performance.* Our benchmark is organized into four categories based on row and column size, the presence of multiple tables, and the inclusion of non-standard tables, resulting in eight distinct subsets, as detailed in Table 3. Our analysis reveals that model performance notably declines when dealing with spreadsheets that have extensive rows and columns, multiple tables, and non-standard structures.

(2) *Increase row size, no significant performance gain.* We assess the effect of row size in the prompt under a single round setting. Figure 4(a) shows that expanding input rows from 5 to 10 doesn't notably enhance performance. This could be due to the excessive context length when processing additional rows.

(3) *GPT4o does not benefit from the multi-round setting.* As shown in Figure 4(b), the performance of GPT-3.5 improves as the round number increases from 2 to 4. However, GPT-4o, the current most advanced LLM, does not benefit from the multi-round setting. Our analysis shows that GPT-4o's

https://doi.org/10.52202/079017-3007

initial solutions have a substantially higher rate of executability and accuracy than other models. Consequently, the additional step of repeatedly retrieving spreadsheet content and executing code does not markedly improve GPT-4o's performance.

Case Study. We provide two cases (Figure 6 and Figure 9) and conduct qualitative analysis of the LLM's performance on different characteristics of the spreadsheet. To get the code solution of these two cases, we utilize the multi-round setting that is discussed in the inference setting section.

Example 1 shows an instruction that manipulates a non-standard relational table with formatting in spreadsheet. Both GPT-3.5 and GPT-4 understand the question and read the spreadsheet properly. However, due to the lack of coding ability, GPT-3.5 uses the wrong element ''`correct_results[cell.column]`'' (marked as red), as shown in Figure 7. The correct one should be ''`correct_results[cell.column - 1].value`''. In Figure 8, GPT-4o can generate the correct Python code, highlighting the importance of coding ability for the accuracy of spreadsheet manipulation.

Example 2 shows an instruction that manipulates two tables in one sheet with formatting. The two GPT-4 inference results both show issues with understanding the structure of the spreadsheet. For the first result of GPT-4o in Figure 10, misalignment occures when reading the tables. "5A1" belong to the column names "Green" but is highlighted in yellow. "5A3" belong to the column names "Amber" but is highlighted in red. For the second result of GPT-4o in Figure 11, the model did not read the whole lookup table in column E to G. The rows after the tenth row have been ignored. Thus, "1A4", "D2", and "1A1" are not highlighted.

These two examples show that proficient coding skills and a solid understanding of various spreadsheet structures are essential for the spreadsheet manipulation task.

4 Conclusion

We create a rigorous benchmark, SPREADSHEETBENCH, for evaluating LLMs' capacity in spreadsheet manipulating. Compared with existing benchmarks, SPREADSHEETBENCH incorporates 912 authentic instructions sourced from online forums, more diverse spreadsheet files with rich formats, multiple tables, irregular tables, and more comprehensive testing with multiple test suites, ensuring better coverage of corner cases.

Acknowledgments and Disclosure of Funding

The authors would like to thank the annotators from Zhipu.AI who participated in the research and data collection process. This work is supported by the National Key Research & Develop Plan (2023YFF0725100) and National Natural Science Foundation of China (62322214, 62076245, U23A20299, 62072460, 62172424, 62276270).

References

[1] S. Rahman, K. Mack, M. Bendre, R. Zhang, K. Karahalios, and A. Parameswaran, "Benchmarking spreadsheet systems," in *Proceedings of the 2020 acm sigmod international conference on management of data*, pp. 1589–1599, 2020.

[2] M. Jackson and M. Staunton, *Advanced modelling in finance using Excel and VBA*. John Wiley & Sons, 2006.

[3] W. L. Winston, *Marketing analytics: Data-driven techniques with Microsoft Excel*. John Wiley & Sons, 2014.

[4] S. Gulwani, "Automating string processing in spreadsheets using input-output examples," in *Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ACM, 2011.

[5] S.-C. Cheung, W. Chen, Y. Liu, and C. Xu, "Custodes: automatic spreadsheet cell clustering and smell detection using strong and weak features," in *Proceedings of the 38th International Conference on Software Engineering*, pp. 464–475, 2016.

[6] S. Gulwani, "Automating string processing in spreadsheets using input-output examples," *ACM Sigplan Notices*, vol. 46, no. 1, pp. 317–330, 2011.

[7] S. Gulwani, W. R. Harris, and R. Singh, "Spreadsheet data manipulation using examples," *Communications of the ACM*, vol. 55, no. 8, pp. 97–105, 2012.

[8] X. Chen, P. Maniatis, R. Singh, C. Sutton, H. Dai, M. Lin, and D. Zhou, "Spreadsheetcoder: Formula prediction from semi-structured context," in *International Conference on Machine Learning*, pp. 1661–1672, PMLR, 2021.

[9] S. Chen, Y. He, W. Cui, J. Fan, S. Ge, H. Zhang, D. Zhang, and S. Chaudhuri, "Auto-formula: Recommend formulas in spreadsheets using contrastive learning for table representations," *Proceedings of the ACM on Management of Data*, vol. 2, no. 3, pp. 1–27, 2024.

[10] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[11] H. Li, J. Su, Y. Chen, Q. Li, and Z.-X. ZHANG, "Sheetcopilot: Bringing software productivity to the next level through large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[12] Y. Chen, Y. Yuan, Z. Zhang, Y. Zheng, J. Liu, F. Ni, and J. Hao, "Sheetagent: A generalist agent for spreadsheet reasoning and manipulation via large language models," *arXiv preprint arXiv:2403.03636*, 2024.

[13] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Chi conference on human factors in computing systems extended abstracts*, pp. 1–7, 2022.

[14] J. Payan, S. Mishra, M. Singh, C. Negreanu, C. Poelitz, C. Baral, S. Roy, R. Chakravarthy, B. Van Durme, and E. Nouri, "Instructexcel: A benchmark for natural language instruction in excel," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4026–4043, 2023.

[15] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," in *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[16] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1470–1480, 2015.

[17] L. Nan, C. Hsieh, Z. Mao, X. V. Lin, N. Verma, R. Zhang, W. Kryściński, H. Schoelkopf, R. Kong, X. Tang, *et al.*, "Fetaqa: Free-form table question answering," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 35–49, 2022.

[18] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[19] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, 2018.

[20] Y. Katsis, S. Chemmengath, V. Kumar, S. Bharadwaj, M. Canim, M. Glass, A. Gliozzo, F. Pan, J. Sen, K. Sankaranarayanan, *et al.*, "Ait-qa: Question answering dataset over complex tables in the airline industry," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pp. 305–314, 2022.

[21] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–34, 2018.

[22] A. Elangovan, J. He, and K. Verspoor, "Memorization vs. generalization: Quantifying data leakage in nlp performance evaluation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1325–1335, 2021.

 https://doi.org/10.52202/079017-3007

[23] Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, W.-t. Yih, D. Fried, S. Wang, and T. Yu, "Ds-1000: A natural and reliable benchmark for data science code generation," in *International Conference on Machine Learning*, pp. 18319–18345, PMLR, 2023.

[24] F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T.-S. Chua, "Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3277–3287, 2021.

[25] W. Chen, M.-W. Chang, E. Schlinger, W. Wang, and W. W. Cohen, "Open question answering over tables and text," *arXiv preprint arXiv:2010.10439*, 2020.

[26] Z. Cheng, H. Dong, Z. Wang, R. Jia, J. Guo, Y. Gao, S. Han, J.-G. Lou, and D. Zhang, "Hitab: A hierarchical table dataset for question answering and natural language generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1094–1110, 2022.

[27] Y. Zhao, Z. Qi, L. Nan, L. J. Flores, and D. Radev, "Loft: Enhancing faithfulness and diversity for table-to-text generation via logic form control," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 554–561, 2023.

[28] Y. Zhao, H. Zhang, S. Si, L. Nan, X. Tang, and A. Cohan, "Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 160–175, 2023.

[29] X. He, M. Zhou, X. Xu, X. Ma, R. Ding, L. Du, Y. Gao, R. Jia, X. Chen, S. Han, *et al.*, "Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 18206–18215, 2024.

[30] J. Li, N. Huo, Y. Gao, J. Shi, Y. Zhao, G. Qu, Y. Wu, C. Ma, J.-G. Lou, and R. Cheng, "Tapilot-crossing: Benchmarking and evolving llms towards interactive data analysis agents," *arXiv preprint arXiv:2403.05307*, 2024.

[31] X. Hu, Z. Zhao, S. Wei, Z. Chai, G. Wang, X. Wang, J. Su, J. Xu, M. Zhu, Y. Cheng, *et al.*, "Infiagent-dabench: Evaluating agents on data analysis tasks," *arXiv preprint arXiv:2401.05507*, 2024.

[32] W. Zhou, M. Mesgar, H. Adel, and A. Friedrich, "Freb-tqa: A fine-grained robustness evaluation benchmark for table question answering," *arXiv preprint arXiv:2404.18585*, 2024.

[33] G. Cormack, I. Munro, T. Vasiga, and G. Kemkes, "Structure, scoring and purpose of computing competitions," *Informatics in education*, vol. 5, no. 1, pp. 15–36, 2006.

[34] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J.-G. Lou, "TAPEX: Table pre-training via learning a neural SQL executor," in *International Conference on Learning Representations*, 2022.

[35] J. Herzig, P. K. Nowak, T. Mueller, F. Piccinno, and J. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4320–4333, 2020.

[36] Z. Cheng, T. Xie, P. Shi, C. Li, R. Nadkarni, Y. Hu, C. Xiong, D. Radev, M. Ostendorf, L. Zettlemoyer, N. A. Smith, and T. Yu, "Binding language models in symbolic languages," *ICLR*, vol. abs/2210.02875, 2023.

[37] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[38] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li, *et al.*, "Deepseek-coder: When the large language model meets programming–the rise of code intelligence," *arXiv preprint arXiv:2401.14196*, 2024.

[39] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[40] AI@Meta, "Llama 3 model card," 2024.

[41] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The Eleventh International Conference on Learning Representations*, 2022.

[42] T. Zheng, G. Zhang, T. Shen, X. Liu, B. Y. Lin, J. Fu, W. Chen, and X. Yue, "Opencodeinterpreter: Integrating code generation with execution and refinement," *arXiv preprint arXiv:2402.14658*, 2024.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1
   (b) Did you describe the limitations of your work? [Yes] See Appendix A.1
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix A.2
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] See Appendix A.3
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 3 and Appendix C
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Due to the excessive cost of executing API-based models and methods.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Seciton 1
   (b) Did you mention the license of the assets? [Yes] See Appendix A.4
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Abstract
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix B
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix A.3
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] See Appendix B
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] See Appendix A.3

https://doi.org/10.52202/079017-3007

A Broader Discussion

A.1 Limitation

The major limitation of SPREADSHEETBENCH lies in data selection and test case construction process. (1) *Data Selection*: As part of our data selection process, we eliminate posts that lack an acknowledged response or are difficult to formalize. Although the proportion of these posts that contain valuable spreadsheet manipulation questions is low, they may also be appropriate for our benchmark under careful human annotation. (2) *Test Case Construction*: Our benchmark has a significantly higher number of questions compared to online judge competitions. Consequently, we did not meticulously devise corner cases for each question, taking into account all possible scenarios and edge cases. However, we still request annotators to remain vigilant regarding potential corner cases while annotating each question (See Figure 13 for an example).

A.2 Potential Impact

SPREADSHEETBENCH are constructed based on real user queries and spreadsheet files. Our objective is to improve the proficiency of LLMs in comprehending and manipulating spreadsheets, with the ultimate aim of automating the spreadsheet manipulation process and reduce human workload in the future. Besides, our benchmark consists of tabular data and corresponding questions, making it suitable for evaluating the anbility of LLMs in understanding and manipulating structured data. Our benchmark exclude any content that could violate personal privacy, contain explicit material, depict violence, or involve other sensitive subjects during our human annotation process. Thus, we believe that the probability of our benchmark causing adverse effects on safety, security, discrimination, surveillance, deception, harassment, human rights, bias, and fairness is extremely minimal.

A.3 Ethical Consideration

In this section, we discuss the ethical considerations regarding our data construction process. (1) *Data Risk Control*: During our annotation process, we filter out the content in questions or spreadsheet files that is inappropriate for presentation to a general audience, such as contents that depict violence, sensitive subjects, etc. In addition, two leaders from the annotation team and two authors conduct an additional verification to ensure that our benchmark does not contain any instances of personally identifiable information. (2) *Annotator Treatment and Consent*: We recruit crowdsourced annotators for the construction of our benchmark (See B.2 for details). We have formalized employment agreements with all the annotators and remunerated them according to the predetermined wage criteria and working schedule. All employment agreements adhere to local regulations. (3) *Copyright*: Our benchmark is indirectly derived from posts on public Excel forums and blogs. The majority of the raw data is acquired from the ExcelForum website using Python crawler scripts, while the data from the remaining three websites is gathered through web browsers. To prevent copyright infringement, we initially consult the Robots Exclusion Protocol[3] of the ExcelForum website. The Robots Exclusion Protocol implemented by the ExcelForum website[4] specifies that Bytespider is prohibited from accessing all directories, Mediapartners-Google is prohibited from crawling the xls files on the website, and all crawlers are prohibited from crawling the search.php directory. Our crawler follows this Robots Exclusion Protocol of the website. The crawler we use is not affiliated with Bytespider or Mediapartners-Google, and it does not crawl the search.php directory. Though we follow the Robots Exclusion Protocol, considering potential licensing and copyright risks, we will refrain from engaging in any secondary distribution of the raw data.

A.4 Maintenance Plan

The benchmark is licensed under CC BY-SA 4.0. RUC KBReasoning group will maintain this benchmark in the long term. We will continue to update the benchmark to fix labelling errors, instructions, or other necessary modifications, and all data will be versioned. We also welcome all contributors interested in our benchmark. If you have any questions or want to extend/augment/build on/contribute to the dataset, feel free to contact us via Github or E-mail (zeyaoma@ruc.edu.cn, zbhmint@bit.edu.cn).

[3] `https://en.wikipedia.org/wiki/Robots.txt`

[4] `https://www.excelforum.com/robots.txt`

B Details of Data Collection

In this section, we introduce our data collection process. First, we present a concise overview of the four Excel websites that we use. Subsequently, we introduce the detail of our data annotation process. Finally, we show some data examples to intutively demonstrate the high quality of our data and the difference compared to previous benchmarks.

B.1 Data Source

The data for our benchmark is obtained from four well-known Excel forums and blogs. These websites contain diverse spreadsheet-related questions in the real world and high-quality answers. Below is a brief introduction.

ExcelForum[5] is one of the most popular spreadsheet forums in the world, containing 1 million threads, 5 million posts, and 1.3 million members as of May 2024. The highest number of concurrent online users recorded was 7,174. ExcelForum provides a detailed categorization of the spreadsheet-related problem. The main categories are *Excel General*, *Excel Programming / VBA / Macros*, *Excel Formulas & Functions* and *Excel Charting & Pivots*. The forum enforces strict rules to ensure the quality of threads, such as eliminating duplicate or ambiguous questions. Moreover, the use of AI tools (e.g., ChatGPT, GPT-4) to create forum questions and answers is not permitted. Although verifying AI-generated content can be challenging, the forum strives to minimize the presence of such content, which enhance the authenticity of the questions and the reliability of the answers.

MrExcel[6] is a widely recognized online platform that hosts more than 1 million discussion threads. Unlike ExcelForum, MrExcel does not employ a detailed classification of questions. Instead, most questions are posted in one category called *Excel Questions*, which contains over 1 million threads. MrExcel adopts a number of rules to ensure the quality of threads and the authenticity of the answers (i.e., the forbidden of AI tools).

ExcelGuru[7] is an online platform designed to enhance one's understanding and proficiency in Excel and Power business intelligence (BI), which is an advanced function provided by Excel. While the forum section of ExcelGuru is relatively small, the article section offers a number of excellent blogs and articles focused on Excel pivot tables and power queries. Pivot tables and power queries are two powerful features that are used for data summarization, analysis, and reporting. They extract meaningful insights from one table and create a new table to present them. Each blog or article includes a complex question with a detailed solution, which is suitable for the data construction of our benchmark.

Chandoo[8] is famous for its exceptional blogs, which are characterised by their superior quality and frequent updates. On average, Chandoo publishes one blog per week, resulting in a grand total of over two thousand blogs. While the publication of blogs is less frequent than forum posts, they offer well-organized questions, answers in spreadsheet format, and detailed instructional guides, making them more convenient for data formatting and a valuable addition to our benchmark. The blogs on Chandoo covers a wide variety of topics, including *VBA Macros*, *Excel Challenges*, *Formula Forensics*, *Excel Howtos*, etc, which are all related to realistic user demands and advanced spreadsheet manipulation techniques. Each blog contains a detailed solution and corresponding Excel files.

B.2 Data Annotation

We conduct three key annotation tasks during the construction of SPREADSHEETBENCH, including (1) Data Selection, (2) Data Formatting, and (3) Test Case Generation. We sequentially perform the three mentioned annotations and meticulously verify the data obtained from the previous annotation before each subsequent annotation to ensure the quality of the data.

(1) Data Selection. We aim to select the data that is feasible and testable from a preliminary processed dataset. The preliminary processed dataset is obtained by using keywords and LLM to exclude data that are unrelated to spreadsheet manipulation, such as software usage questions (See Figure 18) and

[5] https://www.excelforum.com/
[6] https://www.mrexcel.com/
[7] https://excelguru.ca/
[8] https://chandoo.org/

https://doi.org/10.52202/079017-3007

questions related to components in Excel, such as button, userform, etc (See Figure 19). The data selection pipeline is illustrated in Figure 22. First, we discard posts that do not include the spreadsheet files corresponding to the question, as these types of posts often focus on software usage or other questions that are not related to spreadsheet manipulation. Subsequently, we eliminate posts that lack an acknowledged solution. While certain posts may contain user-provided solutions, it is important to note that the questioner may not acknowledge or confirm whether these solutions adequately address their needs. Finally, we try to obtain the answer spreadsheet files corresponding to the solution. Some answerers provide both the solution and the answer spreadsheet files, and we can directly download these files as the initial answer. Others may only offer the solution (e.g., formula, VBA code), and we need to manually apply these solutions in the attached spreadsheet files of the problem. In this situation, we need to check whether the provided solution can be successfully executed. Otherwise, we will discard the posts that contain solutions that cannot be executed.

(2) Data Formatting. We aim to format the post into a self-contained question, an instruction type, an answer position, and a spreadsheet file corresponding to the question. This data formatting process transforms the raw post into testable data and is feasible for our evaluation metrics (See Figure 12 for an example). The annotation team follows the instruction in Figure 23 for the data formatting task. First, we need to do the question verification to check whether the questions summarized by LLM are complete and accurate. Second, we need to check the spreadsheet file corresponding to the question. This involves removing some of the text-based responses and relocating the tabular data to the appropriate location. Subsequently, we need to obtain the spreadsheet files corresponding to the solution, by applying the solution (e.g., formula, VBA code) to a specific position determined by the answerer. Finally, we need to annotate the content of the solution, the answer position, and statistical information related to the spreadsheet content, including whether the spreadsheet file contains multiple tables and non-standard relational tables.

(3) Test Case Generation. We aim to generate two more test cases based on the original questions and the spreadsheet file. The annotation manual is shown in Figure 24. Annotators are asked to modify the position applied by the solution to change the values of cells in answer position. For solutions in formula format, the answer position change immediately once the position applied by the solution are modified. For other situation (e.g., VBA code solution), annotators need to rerun the solution to obtain the updated answer. Annotators must comprehend both the questions and the corresponding solution for each modification. Additionally, they are instructed to carefully consider the exceptional scenarios in the questions and are encouraged to alter the cell value to simulate this exceptional situation (See Figure 13 for an example).

B.3 Data Examples

As mentioned in Section 2.3, SPREADSHEETBENCH incorporates complex questions based on real-world scenarios and diverse types of tables in spreadsheet files, compared to previous benchmarks. In this section, we present four data examples in SPREADSHEETBENCH to illustrate the attributes of real-world problems and the challenging of our benchmark.

Figure 14 shows a cell-level manipulation data example which aims to extract a specific part of a text string from a column of cells. The instruction contains the demand of the user and an example manipulating action of the question, which rarely occur in synthetic instructions of the previous benchmarks. Furthermore, the table within the spreadsheet file is a non-standard relational table that lacks a complete table header. The final result is required to be filled in the cells from B3 to B14, which ensure the uniqueness of the answer.

Figure 15 shows a sheet-level manipulation data example aimed at creating a grid for a duty assignment schedule. The instruction contains the demand of the user, the current formula solution and the error encountered by the user. Besides, the spreadsheet file contains two relational tables in one sheet. The final result should be filled in a two dimensional position (i.e., F2 to T22).

Figure 16 shows a sheet-level manipulation data example aims to search across sheets based on a sheet name and two dates (i.e., first date and last date). The instruction contains a complex requirements, a detailed explanation, a manipulation example and the error encountered by the user. Moreover, the spreadsheet file contains multiple sheets, including the table to be manipulated (e.g., sheets named BUYING or SALES) and examples of possible answers (e.g., sheets named CASE1 or CASE2). The final result should be inserted into a two-dimensional location on the sheet named SH2.

Figure 17 shows a test case modified from the question in Figure 16. This test case is constructed following the user's requirements in the instruction. The user expects to obtain a result that will search within a specific sheet name (cell G2). Only the rows with a date falling between the first date (cell C2) and last date (Cell E2) should be returned. Therefore, we modify the value of the sheet name (cell G2) to construct a new test case, as highlighted in yellow. This test case yields search results in the SALES sheet, rather than the original case that searches in the BUYING sheet. The values in the answer position also change, as highlighted in red. By modifying the sheet name, we can alter the sheet that is being searched, enabling a more comprehensive assessment of the solution's effectiveness.

C Details of Result Inference

In this section, we introduce the inference details and human evaluation settings of our main experiments. Subsequently, we carry out supplementary experiments on our multi-round inference setting and present a detailed analysis of our evaluation metrics.

C.1 Deployment Environment and Model Information

The baseline models are divided into five types, including *TableQA models*, *Open-source code models*, *Open-source general models* and *Spreadsheet-specific methods or products*. For open-source models that need to be deployed on our own, we utilize *PyTorch*, *transformers*, and *vLLM* to load the model. We conduct experiments and obtain inference results on an Ubuntu 22.04 server with graphic cards that contain 8 NVIDIA A100 SXM 80GB GPUs. For spreadsheet-specific methods or products, we evaluate within Google Sheets[9] and Microsoft Excel[10] on Windows 10 and macOS. For the detailed information of each baseline model, please refer to the website url in Table 4.

Table 4: Information of baseline models.

Model	Size	Website Url
TaPEx	400M	`https://huggingface.co/microsoft/tapex-large-finetuned-wtq`
TaPas	340M	`https://huggingface.co/google/tapas-large-finetuned-wtq`
Binder	*	`https://github.com/xlang-ai/Binder`
CodeQwen1.5	7B	`https://huggingface.co/Qwen/CodeQwen1.5-7B-Chat`
DeepseekCoder	33B	`https://huggingface.co/deepseek-ai/deepseek-coder-33b-instruct`
Mixtral-8x7B	47B	`https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1`
Llama-3	70B	`https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct`
GPT-3.5	*	`https://platform.openai.com/overview`
GPT-4	*	`https://platform.openai.com/overview`
SheetCopilot	*	`https://github.com/BraveGroup/SheetCopilot`
Copilot in Excel	*	`https://www.microsoft.com/microsoft-copilot/`

C.2 Details of Experimental Setup

All the baseline LLMs share a unified hyper-parameter (temperature to 1 and top_p to 1) on two inference settings. The prompt on single round setting and multiple round setting are shown in Figure 25 and Figure 26, respectively. For spreadsheet-specific methods or products including SheetCopilot and Copilot in Excel, two of the authors manually upload the spreadsheet files to Google Sheets or the cloud storage of Excel to evaluate the corresponding instructions. As a result of the financial and time costs, we limit our assessment to fifty instructions and one test case per instruction.

[9] `https://www.google.com/sheets/about/`

[10] `https://www.microsoft.com/microsoft-365/excel`

https://doi.org/10.52202/079017-3007

Table 5: Impact of different inference settings on the performance of GPT-3.5 and GPT-4o (%).

Model	Inference Setting	Soft Restriction	Hard Restriction
GPT-3.5	Single + 5 Rows	5.38 (±2.19)	4.00 (±1.55)
	Multiple + Execution Feedback + 5 Rows	**6.58** (±1.42)	**4.50** (±1.50)
	Multiple + ReAct + Execution Feedback	5.50 (±0.33)	4.38 (±2.12)
	Multiple + ReAct + Execution Feedback + 5 Rows	6.04 (±0.59)	4.25 (±0.75)
GPT-4o	Single + 5 Rows	12.79 (±1.71)	10.63 (±1.88)
	Multiple + Execution Feedback + 5 Rows	**16.08** (±1.08)	**13.50** (±1.00)
	Multiple + ReAct + Execution Feedback	13.54 (±1.27)	10.88 (±1.12)
	Multiple + ReAct + Execution Feedback + 5 Rows	14.67 (±1.77)	11.75 (±1.25)

D Additional Experiments

D.1 Analysis of Inference Settings

The main result presented in Table 2 reveals an intriguing observation: the performance of GPT-4o in multi-round setting is slightly lower to that in single-round setting, which is contrary to the result of other models. In this section, we perform comprehensive experiments to determine the influence of various components in the multi-round setting on the outcomes. Furthermore, we perform multiple times of experiments for each inference setting to ensure the statistical significance of the results. Below is the introduction of four inference settings:

(1) *Single + 5 Rows*: This setting is identical to the single-round setting that we employed in the main experiments. We add five rows of the spreadsheet file in pandas dataframe format to the prompt and obtain the direct code solution of LLM in the single round setting.

(2) *Multiple + Execution Feedback + 5 Rows*: Compared to setting (1), LLMs is able to refine the code solution within five rounds if the solution fails to execute. The executor will provide the error traceback to the LLM to start the conversation of next round.

(3) *Multiple + ReAct + Execution Feedback*: Compared to setting (2), we introduce the ReAct mechanism to prompt LLMs to autonomously retrieve the content of the spreadsheet file, while excluding the five rows from the prompt.

(4) *Multiple + ReAct + Execution Feedback + 5 Rows*: This setting is identical to the multi-round setting that we employed in the main experiments. Based on the setting (3), we include the five rows of the spreadsheet file in the prompt.

The experiments are performed on a sample of 200 data points (21.9% of the entire benchmark) and we use GPT-3.5 and GPT-4o for evaluation. We conduct four runs for each setting and calculate the average performance, as illustrate in Table 5. The following findings have been discovered:

1. LLMs benefit from multi-round setting. As shown in Table 5, the performance of GPT-3.5 and GPT-4 improves in three multi-round settings when compared to the single-round setting. This may be attributed to either the ReAct mechanism, the feedback of code execution, or both, which we will determine later. Besides, the performance of GPT-4o on multi-round setting is higher than on single-round setting, which is inconsistent with the main result in 2. This is due to our practice of conducting four times of experiments and calculating the average result, which enhances the stability and reliability of our findings. We also enforce strict alignment between single-round and multi-round prompts by deleting one sentence that ask the LLM to answer carefully in the single-round prompt.

2. LLMs perform worse when adding ReAct mechanism. Comparing settings (2), (3), and (4), we find that using only the feedback from code execution achieves the highest performance. While the ReAct mechanism provide the model with an additional way to obtain spreadsheet information (instead of only 5 rows from the prompt), they are unable to effectively utilize this additional data. While the ReAct mechanism provides the model with an additional way to obtain spreadsheet information (beyond just the 5 rows from the prompt), it is unable to effectively utilize this additional data. Furthermore, existing LLMs lack the ability to make flexible decisions about whether to proceed with reading the content of the spreadsheet or which additional sections should be read. For instance, the model may continue to read the content of the first five rows of the spreadsheet even though this

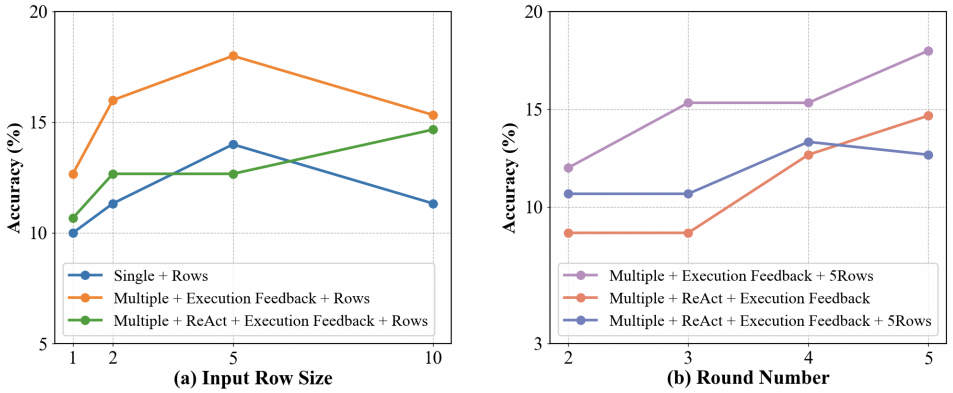


Figure 5: The impact of input row size and round number on GPT-4o across different inference settings. Accuracy represents the value of the overall soft restriction.

information has already been provided in the prompt. This phenomenon demonstrates that the current state-of-the-art LLM still lacks spreadsheet understanding and reasoning ability.

D.2 Analysis of Impact Factors

Based on the four inference settings introduced in Appendix D.1, we conduct an analysis of two key factors in these settings, including the the number of rows provided in the prompt and the round number in multiple round setting.

The experiments are performed on a sample of 50 data points. For setting (1), (2) and (4), we modify the number of rows specified in the prompt to 1, 2, 5, and 10 rows, and analyze the impact on the performance of GPT-4o. For setting (2), (3), and (4), we modify the number of rounds of the interactive inference environment. The following findings have been discovered:

1. The performance of LLMs benefits from a limited number of rows. As illustrated in Figure 5(a), the performance of GPT-4o improves across all three settings as the row size increases from 1 row to 5 rows. However, in two of the settings, the performance declines as the row size increases to 10 rows. This phenomenon is consistent with our observation in the result of Appendix D.1. LLMs fail to process and comprehend a relatively large number of rows, thus requiring improvements in LLMs for understanding tabular data.

2. The performance of LLMs increases with the number of rounds. As illustrated in Figure 5(b), GPT-4o shows an overall upward trend in performance as the number of rounds increases in all three settings. This result is inconsistent with the result in Figure 4(b), as we conduct the experiments multiple times and obtain an average result, which is much more stable. We also find that setting (3) is comparable to setting (4) when the number of rounds is four and five. This phenomenon illustrates that the current LLMs are incapable of effectively retrieving information beyond the first five rows of the spreadsheet in the ReAct mechanism.

D.3 Analysis of Evaluation Metrics

As our evaluation metrics are based on exact match of the cells in the answer position, we conduct a human evaluation to verify the reliability of the metrics. Specifically, we sample 50 instructions from the entire benchmark with three test cases of each and sample the corresponding inference result of GPT-4o. We follow DS-1000 [23] and report the following four indexes:

(1) *Test Case Level False Discovery Rate*: Among all the test cases that successfully pass our automated evaluation, none of them are deemed incorrect by our annotator.

(2) *Test Case Level False Omission Rate*: Among all the test cases that fail our automatic evaluation, 3.8% of them are deemed correct by our annotator.

(3) *Instruction Level False Positive Percentage*: Among all the instructions, none of them have any incorrect sample predictions that pass our automatic metric.

https://doi.org/10.52202/079017-3007

(4) *Instruction Level False Negative Percentage*: Among all instructions, 4% instructions contain at least one correct sample prediction that fails to pass our automatic metric.

Our evaluation metrics do not misclassify accurate results as inaccurate ones, as we achieve a 0% error rate in indices (1) and (3). It may misclassify a small proportion of correct test cases to incorrect ones. This occurs because the code solution generated by LLMs may produce additional content beyond the intended answer (See Figure 20 and Figure 21). In general, our evaluation metric is reliable, as it accurately identifies correct results and only misclassifies a small fraction of correct test cases as incorrect.

Example 1 (Non-standard Table with Color)

Question:

Instruction:

I have a spreadsheet where in Row 3 the correct results are displayed, and other rows consist of people's choice selections that tally correct results. I want cells that match the correct result in Row 3 to be highlighted in yellow, similar to an example I did manually in column I. How do I achieve this automatic highlighting for correct results?

Input Spreadsheet:

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																			
2			WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT						
3	CORRECT RESULT		HOME	AWAY	HOME	AWAY	DRAW	AWAY	HOME	AWAY	DRAW	DRAW	HOME						
4																			
5	PLAYERS	SCORE THIS WEEK	Spurs Southampton	Villa Man City	Leicester Baggies	Arsenal Everton	Liverpool Newcastle	West Ham Chelsea	Sheffield Brighton	Wolves Burnley	Leeds Man Unt	Villa Baggies	Leicester Crystal Pal		THIS WEEKS TOTAL		LAST WEEKS TOTAL		TOTAL OVERALL
6	Dunalin	5	HOME	AWAY	HOME	AWAY	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		5		169		174
7	Albert	4	HOME	AWAY	HOME	HOME	HOME	DRAW	AWAY	HOME	AWAY	HOME	HOME		4		156		160
8	Bolton	6	HOME	AWAY	HOME	HOME	HOME	AWAY	AWAY	HOME	AWAY	DRAW	HOME		6		157		163
9	Biggin	6	HOME	AWAY	HOME	HOME	HOME	AWAY	HOME	HOME	AWAY	HOME	HOME		6		153		159
10	Jellybaby	5	HOME	DRAW	HOME	HOME	HOME	DRAW	HOME	HOME	DRAW	AWAY	HOME		5		149		154
11	Dorreen 1	5	HOME	AWAY	HOME	AWAY	HOME	AWAY	DRAW	HOME	AWAY	AWAY	DRAW		5		148		153
12	Malcolm	4	HOME	AWAY	HOME	DRAW	HOME	DRAW	DRAW	HOME	AWAY	HOME	HOME		4		146		150
13	Princess	5	AWAY	AWAY	HOME	HOME	HOME	AWAY	AWAY	AWAY	AWAY	AWAY	HOME		5		144		149
14	Brummie	3	HOME	AWAY	DRAW	DRAW	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		3		137		140
15	Angleseyite	4	HOME	DRAW	HOME	HOME	HOME	HOME	DRAW	HOME	DRAW	HOME	HOME		4		135		139
16	Timb	6	DRAW	AWAY	HOME	DRAW	HOME	DRAW	HOME	HOME	DRAW	DRAW	HOME		6		134		140
17	John	6	HOME	AWAY	DRAW	DRAW	HOME	HOME	HOME	HOME	DRAW	DRAW	HOME		6		128		134
18	Ianjax	0													0		116		116
19	Filipe	0													0		106		106

Ground Truth Spreadsheet:

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																			
2			WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT						
3	CORRECT RESULT		HOME	AWAY	HOME	AWAY	DRAW	AWAY	HOME	AWAY	DRAW	DRAW	HOME						
4																			
5	PLAYERS	SCORE THIS WEEK	Spurs Southampton	Villa Man City	Leicester Baggies	Arsenal Everton	Liverpool Newcastle	West Ham Chelsea	Sheffield Brighton	Wolves Burnley	Leeds Man Unt	Villa Baggies	Leicester Crystal Pal		THIS WEEKS TOTAL		LAST WEEKS TOTAL		TOTAL OVERALL
6	Dunalin	5	HOME	AWAY	HOME	AWAY	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		5		169		174
7	Albert	4	HOME	AWAY	HOME	HOME	HOME	DRAW	AWAY	HOME	AWAY	HOME	HOME		4		156		160
8	Bolton	6	HOME	AWAY	HOME	HOME	HOME	AWAY	AWAY	HOME	AWAY	DRAW	HOME		6		157		163
9	Biggin	6	HOME	AWAY	HOME	HOME	HOME	AWAY	HOME	HOME	AWAY	HOME	HOME		6		153		159
10	Jellybaby	5	HOME	DRAW	HOME	HOME	HOME	DRAW	HOME	HOME	DRAW	AWAY	HOME		5		149		154
11	Dorreen 1	5	HOME	AWAY	HOME	AWAY	HOME	AWAY	DRAW	HOME	AWAY	AWAY	DRAW		5		148		153
12	Malcolm	4	HOME	AWAY	HOME	DRAW	HOME	DRAW	DRAW	HOME	AWAY	HOME	HOME		4		146		150
13	Princess	5	AWAY	AWAY	HOME	HOME	HOME	AWAY	AWAY	AWAY	AWAY	AWAY	HOME		5		144		149
14	Brummie	3	HOME	AWAY	DRAW	DRAW	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		3		137		140
15	Angleseyite	4	HOME	DRAW	HOME	HOME	HOME	HOME	DRAW	HOME	DRAW	HOME	HOME		4		135		139
16	Timb	6	DRAW	AWAY	HOME	DRAW	HOME	DRAW	HOME	HOME	DRAW	DRAW	HOME		6		134		140
17	John	6	HOME	AWAY	DRAW	DRAW	HOME	HOME	HOME	HOME	DRAW	DRAW	HOME		6		128		134
18	Ianjax	0													0		116		116
19	Filipe	0													0		106		106

Figure 6: The question of Example 1 with non-standard table and color.

https://doi.org/10.52202/079017-3007

Example 1 (Non-standard Table with Color)

Result of GPT-3.5 (Wrong: Coding Ability)

Code Solution:

```
import openpyxl
from openpyxl.styles import PatternFill

# Load the spreadsheet
spreadsheet_path = '/mnt/data/spreadsheet/CF_24784/1_CF_24784_input.xlsx'
workbook = openpyxl.load_workbook(spreadsheet_path)

# Get the target sheet
sheet_name = 'Sheet1'
sheet = workbook[sheet_name]

# Set the range of cells to modify
answer_position = 'D6:N17'

# Get the correct result cells in Row 3
correct_results = sheet[3]

# Define the yellow fill style
yellow_fill = PatternFill(start_color="FFFF00", end_color="FFFF00", fill_type="solid")

# Iterate over the cells within the specified range
for row in sheet[answer_position]:
    for cell in row:
        # Compare the cell value with the corresponding correct result in Row 3
        if cell.value == correct_results[cell.column]:
            # Apply the yellow fill to the matching cells
            cell.fill = yellow_fill

# Save the modified spreadsheet to a new file
output_path = '/mnt/data/outputs/multi_row_react_exec_gpt-3.5-turbo-16k/1_CF_24784_output.xlsx'
workbook.save(output_path)
```

Execution Result:

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																			
2			WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT						
3	CORRECT RESULT		HOME	AWAY	HOME	AWAY	DRAW	AWAY	HOME	AWAY	DRAW	DRAW	HOME						
4																			
5	PLAYERS	SCORE THIS WEEK	Spurs Southampton	Villa Man City	Leicester Baggies	Arsenal Everton	Liverpool Newcastle	West Ham Chelsea	Sheffield Brighton	Wolves Burnley	Leeds Man Unt	Villa Baggies	Leicester Crystal Pal		THIS WEEKS TOTAL		LAST WEEKS TOTAL		TOTAL OVERALL
6	Dunalin	5	HOME	AWAY	HOME	AWAY	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		5		169		174
7	Albert	4	HOME	AWAY	HOME	HOME	HOME	DRAW	AWAY	HOME	AWAY	HOME	HOME		4		156		160
8	Bolton	6	HOME	AWAY	HOME	HOME	HOME	AWAY	AWAY	HOME	AWAY	DRAW	HOME		6		157		163
9	Biggin	6	HOME	AWAY	HOME	HOME	HOME	AWAY	HOME	HOME	AWAY	HOME	HOME		6		153		159
10	Jellybaby	5	HOME	DRAW	HOME	HOME	HOME	DRAW	HOME	HOME	DRAW	AWAY	HOME		5		149		154
11	Dorreen 1	5	HOME	AWAY	HOME	AWAY	HOME	AWAY	DRAW	HOME	AWAY	AWAY	DRAW		5		148		153
12	Malcolm	4	HOME	AWAY	HOME	DRAW	HOME	DRAW	DRAW	HOME	AWAY	HOME	HOME		4		146		150
13	Princess	5	AWAY	AWAY	HOME	HOME	HOME	AWAY	AWAY	AWAY	AWAY	AWAY	HOME		5		144		149
14	Brummie	3	HOME	AWAY	DRAW	DRAW	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		3		137		140
15	Angleseyite	4	HOME	DRAW	HOME	HOME	HOME	HOME	DRAW	HOME	DRAW	HOME	HOME		4		135		139
16	Timb	6	DRAW	AWAY	HOME	DRAW	HOME	DRAW	HOME	HOME	DRAW	DRAW	HOME		6		134		140
17	John	6	HOME	AWAY	DRAW	DRAW	HOME	HOME	HOME	HOME	DRAW	DRAW	HOME		6		128		134
18	Ianjax	0													0		116		116
19	Filipe	0													0		106		106

Figure 7: GPT-3.5 fails on Example 1 because of its lack of coding ability.

Example 1 (Non-standard Table with Color)

Result of GPT-4o (Correct)

Code Solution:

```
import pandas as pd
from openpyxl import load_workbook
from openpyxl.styles import PatternFill

def highlight_correct_cells(input_path, output_path, sheet_name):
    # Load the spreadsheet and the specific sheet
    book = load_workbook(input_path)
    sheet = book[sheet_name]

    # Define the fill for highlighting (Yellow Color)
    yellow_fill = PatternFill(start_color="FFFF00", end_color="FFFF00", fill_type="solid")

    # Get the correct answers from row 3 (D3 to N3)
    correct_results = [cell.value for cell in sheet[3][3:14]]  # This gets the values from D3 to N3

    # Iterate over the specified range D6:N17
    for row in range(6, 18):  # Excel rows 6 to 17 inclusive
        for col, correct_result in enumerate(correct_results, start=4):  # Excel columns D to N
            cell = sheet.cell(row=row, column=col)
            if cell.value == correct_result:
                cell.fill = yellow_fill

    # Save the modified workbook
    book.save(output_path)

# Specify parameters
spreadsheet_path = '/mnt/data/spreadsheet/CF_24784/1_CF_24784_input.xlsx'
output_path = '/mnt/data/outputs/multi_row_react_exec_gpt-4o-2024-05-13/1_CF_24784_output.xlsx'
sheet_name = 'Sheet1'

# Call the function to highlight the correct cells
highlight_correct_cells(spreadsheet_path, output_path, sheet_name)
```

Execution Result:

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																			
2			WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT	WEEKS RESULT						
3	CORRECT RESULT		HOME	AWAY	HOME	AWAY	DRAW	AWAY	HOME	AWAY	DRAW	DRAW	HOME						
4																			
5	PLAYERS	SCORE THIS WEEK	Spurs Southampton	Villa Man City	Leicester Baggies	Arsenal Everton	Liverpool Newcastle	West Ham Chelsea	Sheffield Brighton	Wolves Burnley	Leeds Man Unt	Villa Baggies	Leicester Crystal Pal		THIS WEEKS TOTAL		LAST WEEKS TOTAL		TOTAL OVERALL
6	Dunalin	5	HOME	AWAY	HOME	AWAY	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		5		169		174
7	Albert	4	HOME	AWAY	HOME	HOME	HOME	DRAW	AWAY	HOME	AWAY	HOME	HOME		4		156		160
8	Bolton	6	HOME	AWAY	HOME	HOME	HOME	AWAY	AWAY	HOME	AWAY	DRAW	HOME		6		157		163
9	Biggin	6	HOME	AWAY	HOME	HOME	HOME	AWAY	HOME	HOME	AWAY	HOME	HOME		6		153		159
10	Jellybaby	5	HOME	DRAW	HOME	HOME	HOME	DRAW	HOME	HOME	DRAW	AWAY	HOME		5		149		154
11	Dorreen 1	5	HOME	AWAY	HOME	AWAY	HOME	AWAY	DRAW	HOME	AWAY	AWAY	DRAW		5		148		153
12	Malcolm	4	HOME	AWAY	HOME	DRAW	HOME	DRAW	DRAW	HOME	AWAY	HOME	HOME		4		146		150
13	Princess	5	AWAY	AWAY	HOME	HOME	HOME	AWAY	AWAY	AWAY	AWAY	AWAY	HOME		5		144		149
14	Brummie	3	HOME	AWAY	DRAW	DRAW	HOME	HOME	AWAY	HOME	AWAY	HOME	HOME		3		137		140
15	Angleseyite	4	HOME	DRAW	HOME	HOME	HOME	HOME	DRAW	HOME	DRAW	HOME	HOME		4		135		139
16	Timb	6	DRAW	AWAY	HOME	DRAW	HOME	DRAW	HOME	HOME	DRAW	DRAW	HOME		6		134		140
17	John	6	HOME	AWAY	DRAW	DRAW	HOME	HOME	HOME	HOME	DRAW	DRAW	HOME		6		128		134
18	Ianjax	0													0		116		116
19	Filipe	0													0		106		106

Figure 8: GPT-4o gets the right answer on Example 1.

https://doi.org/10.52202/079017-3007

Example 2 (Multiple Table in One Sheet with Color)

Question:

Instruction:

How can I apply conditional formatting to a cell in the 'score 1' or 'score 2' columns in my Excel 2016 spreadsheet, so that it highlights in Green, Yellow, or Red depending on whether the manually entered value matches any value within the respective color-coded range of cells in the Green, Yellow, Red columns as shown in my attached example sheet? Please refer to the attachment for the highlighted color

Input Spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K
1											
2		Score 1	Score 2		Green	Amber	Red				
3		5A1	5A4		5A1	5A3	5A4				
4		5A3	5A1		5A2	4A3	5A-				
5		5A4	5A3		4A1	3A3	4A4				
6		5A2	1A4		4A2	2A3	4A-				
7		4A3	D2		3A1	1A3	3A4				
8		5A-	1A1		3A2	A3	3A-				
9					2A1	B3	2A4				
10					2A2	C3	2A-				
11					1A1	D1	1A4				
12					1A2	D2	1A-				

Ground Truth Spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K
1											
2		Score 1	Score 2		Green	Amber	Red				
3		5A1	5A4		5A1	5A3	5A4				
4		5A3	5A1		5A2	4A3	5A-				
5		5A4	5A3		4A1	3A3	4A4				
6		5A2	1A4		4A2	2A3	4A-				
7		4A3	D2		3A1	1A3	3A4				
8		5A-	1A1		3A2	A3	3A-				
9					2A1	B3	2A4				
10					2A2	C3	2A-				
11					1A1	D1	1A4				
12					1A2	D2	1A-				

Figure 9: The question of Example 2 with multiple table and color.

Example 2 (Multiple Table in One Sheet with Color)

The First Result of GPT-4o (Wrong: Misunderstanding of table structure)

Code Solution:

```
import openpyxl
from openpyxl.styles import PatternFill
from openpyxl.formatting.rule import FormulaRule

# Load the workbook and select the required sheet
spreadsheet_path = '/mnt/data/spreadsheet/CF_12429/1_CF_12429_input.xlsx'
output_path = '/mnt/data/outputs/multi_row_react_exec_gpt-4o-2024-05-13/1_CF_12429_output.xlsx'

wb = openpyxl.load_workbook(spreadsheet_path)
ws = wb['Sheet1']

# Define color fills
green_fill = PatternFill(start_color='00FF00', end_color='00FF00', fill_type='solid')
amber_fill = PatternFill(start_color='FFFF00', end_color='FFFF00', fill_type='solid')
red_fill = PatternFill(start_color='FF0000', end_color='FF0000', fill_type='solid')

# Apply conditional formatting to the range B2:C5 (scores)
for row in range(2, 6):  # Corresponds to rows 3 to 6 in Excel (Python is 0-indexed)
    for col in ['B', 'C']:  # Score 1 and Score 2 columns
        cell = f'{col}{row + 1}'  # Convert to Excel-style cell reference
        score_cell_value = ws[cell].value

        # Check against the color-coded columns (D, E, F in Excel or 4, 5, 6 in Python)
        if score_cell_value in [ws[f'D{row + 1}'].value for row in range(2, 6)]:
            ws[cell].fill = green_fill
        elif score_cell_value in [ws[f'E{row + 1}'].value for row in range(2, 6)]:
            ws[cell].fill = amber_fill
        elif score_cell_value in [ws[f'F{row + 1}'].value for row in range(2, 6)]:
            ws[cell].fill = red_fill

# Save the modified workbook
wb.save(output_path)
```

Execution Result:

	A	B	C	D	E	F	G	H	I	J	K
1											
2		Score 1	Score 2		Green	Amber	Red				
3		5A1	5A4		5A1	5A3	5A4				
4		5A3	5A1		5A2	4A3	5A-				
5		5A4	5A3		4A1	3A3	4A4				
6		5A2	1A4		4A2	2A3	4A-				
7		4A3	D2		3A1	1A3	3A4				
8		5A-	1A1		3A2	A3	3A-				
9					2A1	B3	2A4				
10					2A2	C3	2A-				
11					1A1	D1	1A4				
12					1A2	D2	1A-				

Figure 10: GPT-4o fails on Example 2 because of misunderstanding of table structure.

https://doi.org/10.52202/079017-3007

Example 2 (Multiple Table in One Sheet with Color)

The Second Result of GPT-4o (Wrong: Incomplete reading of table content)

Code Solution

```
import openpyxl
from openpyxl.styles import PatternFill
from openpyxl.formatting.rule import FormulaRule

# Load the workbook and select the desired sheet
spreadsheet_path = '/mnt/data/spreadsheet/CF_12429/1_CF_12429_input.xlsx'
output_path = '/mnt/data/outputs/multi_gpt-4o-2024-05-13/1_CF_12429_output.xlsx'
workbook = openpyxl.load_workbook(spreadsheet_path)
sheet = workbook['Sheet1']

# Define the fills for Green, Amber, and Red
green_fill = PatternFill(start_color="00FF00", end_color="00FF00", fill_type="solid")
yellow_fill = PatternFill(start_color="FFFF00", end_color="FFFF00", fill_type="solid")
red_fill = PatternFill(start_color="FF0000", end_color="FF0000", fill_type="solid")

# Get the ranges for color-coding
green_range = 'E2:E10'  # Assuming the Green values are in this range
amber_range = 'F2:F10'  # Assuming the Amber values are in this range
red_range = 'G2:G10'    # Assuming the Red values are in this range

# Apply conditional formatting rules
for row in range(3, 9):  # Rows B3:C8
    for col in ['B', 'C']:
        cell = f"{col}{row}"

        # Green condition
        sheet.conditional_formatting.add(cell, FormulaRule(formula=[f'COUNTIF({green_range},{cell})>0'], fill=green_fill))

        # Amber condition
        sheet.conditional_formatting.add(cell, FormulaRule(formula=[f'COUNTIF({amber_range},{cell})>0'], fill=yellow_fill))

        # Red condition
        sheet.conditional_formatting.add(cell, FormulaRule(formula=[f'COUNTIF({red_range},{cell})>0'], fill=red_fill))

# Save the modified file
workbook.save(output_path)
```

Execution Result:

	A	B	C	D	E	F	G	H	I	J	K
1											
2		Score 1	Score 2		Green	Amber	Red				
3		5A1	5A4		5A1	5A3	5A4				
4		5A3	5A1		5A2	4A3	5A-				
5		5A4	5A3		4A1	3A3	4A4				
6		5A2	1A4		4A2	2A3	4A-				
7		4A3	D2		3A1	1A3	3A4				
8		5A-	1A1		3A2	A3	3A-				
9					2A1	B3	2A4				
10					2A2	C3	2A-				
11					1A1	D1	1A4				
12					1A2	D2	1A-				

Figure 11: GPT-4o fails on Example 2 because of incomplete reading of table content.

Original Question:

Get data from one table to another with any method? (match/index)??

Hello,
can not find out how to solve this task? Was trying with match/index method but I did not work? Can you help me someone? Thanks

Category Category
HDD SPD FFI ADW HDW LAD TWC TWC LAD HDW ADW FFI SPD HDD
Brand Persil 2005 Brand Persil
Bref 730 4575 Silan
Cif 3772 3867 3121 Pur
Clin 2269 Bref
Coccolino 678 Bloo
Domestos 25 8416 Clin
Finish 7647
Jar 5922 2903
Lenor 7636
Pur 8913
Silan 5531
TS 2502 7343
Dylon 18217
Bloo 1230

Attached Files
Task 6.xlsx (10.6 KB, 5 views) Download

Original Answer:

Re: Get data from one table to another with any method? (match/index)??

Welcome to the forum.

INDEX MATCH MATCH does work.

In M6 copies across and down:

=INDEX($C$6:$I$19,MATCH($L6,$B$6:$B$19,0),MATCH(M$5,$C$5:$I$5,0))

For your Slovak locale:

=INDEX($C$6:$I$19;MATCH($L6;$B$6:$B$19;0);MATCH(M$5;$C$5:$I$5;0))

Attached Files
Task 6 AliGW.xlsx (11.2 KB, 3 views) Download

Instruction:

I'm struggling to transfer data from one table to another using Excel. I attempted to use the INDEX/MATCH method but it didn't work for me. Specifically, I have one table with categories like HDD, SPD, FFI, ADW, HDW, LAD, TWC, and brands such as Persil, Bref, Cif, Clin, Coccolino, Domestos, Finish, Jar, Lenor, Pur, Silan, TS, Dylon, and Bloo, with various numerical values beneath each brand and category. How can I correctly apply the INDEX/MATCH method to copy data across and down in this scenario? Get the data from blue table to the red table by using any method.

Spreadsheet:

			Category						
			HDD	SPD	FFI	ADW	HDW	LAD	TWC
Brand		Persil			2005				
		Bref	730					4575	
		Cif	3772					3867	3121
		Clin							2269
		Coccolino			678				
		Domestos	25					8416	
		Finish				7647			
		Jar				5922	2903		
		Lenor			7636				
		Pur					8913		
		Silan			5531				
		TS					2502	7343	
		Dylon						18217	
		Bloo							1230

		Category						
		TWC	LAD	HDW	ADW	FFI	SPD	HDD
Brand	Persil							
	Silan							
	Pur							
	Bref							
	Bloo							
	Clin							

Instruction Type: Sheet-Level Manipulation

Answer Position: M6:S11

Figure 12: An example of the data formatting process. We revise the questions in the post and include supplementary details. We annotate the answer position based on the location applied by the formula. We apply the extracted solution from the post to the problem attachments (i.e., the attached spreadsheets)

https://doi.org/10.52202/079017-3007

Instruction:

How do I create a formula in cell J3 that assigns a value based on the value in cell H3, such that if H3 is equal to or less than 3, J3 should display 32.00; if H3 is greater than 3 but not greater than 6, J3 should display 38.00; and if H3 is greater than 6, J3 should display 55.00?

Spreadsheet:

	A	B	C	D	E	F	H	J	K	L	M	N
1												
2					Width	Length	Depth	Cost	~Cost		Approxmite Retail	
3			Acrylic boxes	.25 SIDES .125" TOP	22	22	3		0.00		0.00	◄ Acrylic Box ►
4			Add Mat price and double fitting		$33.00	$15.97					48.97	◄ Bars & Canvas ►
5		MULTIPLIER			St Bars	Canvas					48.97	◄ Total ►
6		8.00	0								Plus fitting etc..	

Instruction Type: Cell-Level Manipulation

Answer Position: J3

The solution of GPT-4o on Case 1: (Correct)

	A	B	C	D	E	F	H	J	K	L	M	N
1												
2					Width	Length	Depth	Cost	~Cost		Approxmite Retail	
3			Acrylic boxes	.25 SIDES .125" TOP	22	22	3	32.00	256.00		512.00	◄ Acrylic Box ►
4			Add Mat price and double fitting		$33.00	$15.97					48.97	◄ Bars & Canvas ►
5		MULTIPLIER			St Bars	Canvas					560.97	◄ Total ►
6		8.00	256								Plus fitting etc..	

The solution of GPT-4o on Case 2 (Corner Case): (Wrong)

	A	B	C	D	E	F	H	J	K	L	M	N
1												
2					Width	Length	Depth	Cost	~Cost		Approxmite Retail	
3			Acrylic boxes	.25 SIDES .125" TOP	22	22		32.00	256.00		512.00	◄ Acrylic Box ►
4			Add Mat price and double fitting		$33.00	$15.97					48.97	◄ Bars & Canvas ►
5		MULTIPLIER			St Bars	Canvas					560.97	◄ Total ►
6		8.00	256								Plus fitting etc..	

The ground truth answer on Case 2 (Corner Case):

	A	B	C	D	E	F	H	J	K	L	M	N
1												
2					Width	Length	Depth	Cost	~Cost		Approxmite Retail	
3			Acrylic boxes	.25 SIDES .125" TOP	22	22			0.00		0.00	◄ Acrylic Box ►
4			Add Mat price and double fitting		$33.00	$15.97					48.97	◄ Bars & Canvas ►
5		MULTIPLIER			St Bars	Canvas					48.97	◄ Total ►
6		8.00	0								Plus fitting etc..	

Figure 13: An example of the corner test case. In a practical situation, it is possible for cell H3 to be empty, and consequently, the value in cell J3 should also be empty. However, GPT-4o treats the value of empty cells as zero, leading to an incorrect result in cell J3.

https://doi.org/10.52202/079017-3007

Instruction:

How can I extract a specific part of a text string from cell A3, search for it in a separate range 'codes2', find the corresponding value in another range 'month', and then place that value in cell B3? For example, If any part of A3 contains the value in row "codes2", populate the corresponding value in row "month", and place that value in cell B3. A3= "2014_09_Drop2_Reminder". Search in "codes2" for value (in this case it would be "09"). Find corresponding value in "month" (in this case it would be "SEP"). Place that value ("SEP") in cell B3.

Spreadsheet:

	A	B	C	D	E
1					
2	Listing				
3	2014_09_D2_Reminder_EN			01	JAN
4	2014_09_D2_Reminder_SP			02	FEB
5	2014_09_Conserve_EN			03	MAR
6	2014_09_Conserve_SP			04	APR
7	2014_10_Renew_1_FSOL			05	MAY
8	2014_10_Renew_2_SADV			06	JUN
9	2014_10_Renew_3_RHSP			07	JUL
10	2014_10_Renew_4_SRGE			08	AUG
11	2014_10_Renew_5_SRGE			09	SEP
12	2014_10_Renew_6_KNKW			10	OCT
13	2014_11_Advanced_NTX_EN			11	NOV
14	2014_11_Advanced_NTX_SP			12	DEC

Instruction Type: Cell-Level Manipulation

Answer Position: B3:B14

Answer (spreadsheet):

	A	B	C	D	E
1					
2	Listing				
3	2014_09_D2_Reminder_EN	SEP		01	JAN
4	2014_09_D2_Reminder_SP	SEP		02	FEB
5	2014_09_Conserve_EN	SEP		03	MAR
6	2014_09_Conserve_SP	SEP		04	APR
7	2014_10_Renew_1_FSOL	OCT		05	MAY
8	2014_10_Renew_2_SADV	OCT		06	JUN
9	2014_10_Renew_3_RHSP	OCT		07	JUL
10	2014_10_Renew_4_SRGE	OCT		08	AUG
11	2014_10_Renew_5_SRGE	OCT		09	SEP
12	2014_10_Renew_6_KNKW	OCT		10	OCT
13	2014_11_Advanced_NTX_EN	NOV		11	NOV
14	2014_11_Advanced_NTX_SP	NOV		12	DEC

Figure 14: A cell-level manipulation example question involves manipulating a non-standard relational table (missing header on column D and column E).

https://doi.org/10.52202/079017-3007

Instruction:

I am working on creating a grid for a duty assignment schedule that places an 'x' in cells to indicate a match between names and dates. I tried using the formula =INDEX($E$1, MATCH(1, ($J3=$F$3:$F$18)*($L$2=$G$3:$G$18), 0)) and it works for the first cell. However, when a name is repeated in the schedule, I receive a #REF error, and other variations of the formula yield an #N/A error. I am looking for assistance to correct this issue.

Spreadsheet:

Date	Dir
2022/1/5	Jennifer
2022/1/12	Cal
2022/1/19	Carla
2022/1/26	Rodney
2022/2/2	Paul
2022/2/9	Mark B.
2022/2/16	Mark W.
2022/2/23	Breanda
2022/3/2	Marianne
2022/3/9	Randy
2022/3/16	Walter
2022/3/23	Chris
2022/3/30	Latoya
2022/4/6	Rosemarie
2022/4/13	Jennifer
2022/4/20	Ward
2022/4/27	Cal
2022/5/4	Carla
2022/5/11	Rodney
2022/5/18	Paul
2022/5/25	Mark B.

Date	Breand	Cal	Carla	Chris	Jennife	Latoya	Marianr	Mark B.	Mark W	Paul	Randy	Rodne	Rosema	Walter	Ward
2022/1/5					x										
2022/1/12		x													
2022/1/19			x												
2022/1/26												x			
2022/2/2										x					
2022/2/9								x							
2022/2/16									x						
2022/2/23	x														
2022/3/2							x								
2022/3/9															
2022/3/16															
2022/3/23															
2022/3/30															
2022/4/6															
2022/4/13															
2022/4/20															
2022/4/27															
2022/5/4															
2022/5/11															
2022/5/18															
2022/5/25															

Instruction Type: Sheet-Level Manipulation

Answer Position: F2:T22

Answer (spreadsheet):

Date	Dir
2022/1/5	Jennifer
2022/1/12	Cal
2022/1/19	Carla
2022/1/26	Rodney
2022/2/2	Paul
2022/2/9	Mark B.
2022/2/16	Mark W.
2022/2/23	Breanda
2022/3/2	Marianne
2022/3/9	Randy
2022/3/16	Walter
2022/3/23	Chris
2022/3/30	Latoya
2022/4/6	Rosemarie
2022/4/13	Jennifer
2022/4/20	Ward
2022/4/27	Cal
2022/5/4	Carla
2022/5/11	Rodney
2022/5/18	Paul
2022/5/25	Mark B.

Date	Breand	Cal	Carla	Chris	Jennife	Latoya	Marianr	Mark B.	Mark W	Paul	Randy	Rodne	Rosema	Walter	Ward
2022/1/5					x										
2022/1/12		x													
2022/1/19			x												
2022/1/26												x			
2022/2/2										x					
2022/2/9								x							
2022/2/16									x						
2022/2/23	x														
2022/3/2							x								
2022/3/9											x				
2022/3/16														x	
2022/3/23				x											
2022/3/30						x									
2022/4/6													x		
2022/4/13					x										
2022/4/20															x
2022/4/27		x													
2022/5/4			x												
2022/5/11												x			
2022/5/18										x					
2022/5/25								x							

Figure 15: A sheet-level manipulation example question involves manipulating multiple tables in one sheet (first table from A1 to B22; second table from E1 to T22).

Instruction:

I have an Excel workbook with multiple sheets, each sheet containing around 7000 rows of data. I need a macro that will search across sheets based on a sheet name and two dates. Specifically, the search criteria should be inputted into cells C2 and E2 for the dates, and cell G2 for the sheet name, on a summary sheet (SH2). If these cells are filled, the macro should pull the date, invoice, and total from the last row containing the word 'TOTAL' in column A of the specified sheet, and insert a row to sum the totals in column D. If C2 and E2 are empty, but G2 has a sheet name , it should bring all the data from the named sheet and add a total row as previously described. If C2, E2, and G2 are all empty, it should clear existing data in SH2. Each time the macro runs, it should clear previous data in SH2 before bringing in new data to prevent duplication. However, if the same dates are entered for data that has already been copied, it should not copy the data again but should only bring data up to the last row based on the current date. Additionally, there is a requirement that the data be centered and formatted as numbers with thousands separators (e.g., 1,000.00), rather than aligned right and without number formatting, which the current macro is doing. Lastly, if C2, E2, and G2 are all empty, it should trigger the macro to clear data in SH2, but I'm encountering a 'subscript out of range' error when attempting to do this.

Spreadsheet:

	A	B	C	D	E	F	G	H	I	J
1	ITEM	DATE	NAME	INVOICE	GOODS	TYPE	PR	QTY	UNIT	TOTAL
2	1	2023/5/25	CR-1000	STVG-1000	ATR	AM1	MTR	5.00	25.00	125.00
3	2	2023/5/25	CR-1000	STVG-1000	ATR	AM2	PO	4.00	35.00	140.00
4	3	2023/5/25	CR-1000	STVG-1000	ATR	AM1	SO	2.00	45.00	90.00
5	TOTAL	2023/5/25	CR-1000	STVG-1000						355.00
6	1	2023/5/28	CR-1001	STVG-1001	ATR	AM1	GR	2.00	23.00	46.00
7	2	2023/5/28	CR-1001	STVG-1001	ATR	AM2	PO	2.00	35.00	70.00
8	TOTAL	2023/5/28	CR-1001	STVG-1001						116.00
9	1	2023/7/28	CR-1000	STVG-1002	ATR	AM1	MTR	2.00	25.00	50.00
10	2	2023/7/28	CR-1000	STVG-1002	ATR	AM2	PO	2.00	35.00	70.00
11	3	2023/7/28	CR-1000	STVG-1002	ATR	AM1	SO	2.00	45.00	90.00
12	TOTAL	2023/7/28	CR-1000	STVG-1002						210.00
13	1	2023/7/28	CR-1000	STVG-1003	ATR	AM2	SO	2.00	45.00	330.00
14	TOTAL	2023/7/28	CR-1000	STVG-1003						330.00
15	1	2023/8/27	CR-1002	STVG-1004	VBGF	HJH1	HGF	2.00	40.00	80.00
16	2	2023/8/27	CR-1002	STVG-1004	ASDW	MMN	SO	2.00	50.00	100.00
17	TOTAL	2023/8/27	CR-1002	STVG-1004						180.00
18										

BUYING | SALES | SH2 | CASE1 | CASE2 | +

	A	B	C	D	E	F	G
1			FIRST DATE		LAST DATE		SHEET NAME
2			2023/5/25		2023/5/28		BUYING
3							
4	ITEM	DATE	INVOICE	TOTAL			
5							
6							
7							
8							
9							
10							
11							

BUYING | SALES | SH2 | CASE1 | CASE2 | +

Instruction Type: Sheet-Level Manipulation

Answer Position: 'SH2'!A5:D10

Answer (spreadsheet):

	A	B	C	D	E	F	G
1			FIRST DATE		LAST DATE		SHEET NAME
2			2023/5/25		2023/5/28		BUYING
3							
4	ITEM	DATE	INVOICE	TOTAL			
5	1	2023/5/25	STVG-1000	355.00			
6	2	2023/5/28	STVG-1001	116.00			
7	Total			471.00			
8							
9							
10							
11							

BUYING | SALES | SH2 | CASE1 | CASE2 | +

Figure 16: A sheet-level manipulation example question involves manipulating multiple tables on multiple sheets (first table in sheet named BUYING, from A1 to J17; second table in sheet named SH2, from A4 to D10).

https://doi.org/10.52202/079017-3007

Instruction:

I have an Excel workbook with multiple sheets, each sheet containing around 7000 rows of data. I need a macro that will search across sheets based on a sheet name and two dates. Specifically, the search criteria should be inputted into cells C2 and E2 for the dates, and cell G2 for the sheet name, on a summary sheet (SH2). If these cells are filled, the macro should pull the date, invoice, and total from the last row containing the word 'TOTAL' in column A of the specified sheet, and insert a row to sum the totals in column D. If C2 and E2 are empty, but G2 has a sheet name , it should bring all the data from the named sheet and add a total row as previously described. If C2, E2, and G2 are all empty, it should clear existing data in SH2. Each time the macro runs, it should clear previous data in SH2 before bringing in new data to prevent duplication. However, if the same dates are entered for data that has already been copied, it should not copy the data again but should only bring data up to the last row based on the current date. Additionally, there is a requirement that the data be centered and formatted as numbers with thousands separators (e.g., 1,000.00), rather than aligned right and without number formatting, which the current macro is doing. Lastly, if C2, E2, and G2 are all empty, it should trigger the macro to clear data in SH2, but I'm encountering a 'subscript out of range' error when attempting to do this.

Spreadsheet:

	A	B	C	D	E	F	G	H	I	J
1	ITEM	DATE	NAME	INVOICE	GOODS	TYPE	PR	QTY	UNIT	TOTAL
2	1	2023/5/27	CR-1000	FRVG-10000	ATR	AM1	MTR	2.00	30.00	60.00
3	2	2023/5/27	CR-1000	FRVG-10000	ATR	AM2	PO	2.00	40.00	80.00
4	TOTAL	2023/5/27	CR-1000	FRVG-10000						140.00
5	1	2023/5/29	CR-1001	FRVG-10001	ATR	AM1	GR	1.00	30.00	30.00
6	TOTAL	2023/5/29	CR-1001	FRVG-10001						30.00
7	1	2023/7/30	CR-1000	FRVG-10002	ATR	AM2	PO	1.00	40.00	40.00
8	2	2023/7/30	CR-1000	FRVG-10002	ATR	AM1	SO	1.00	50.00	50.00
9	TOTAL	2023/7/30	CR-1000	FRVG-10002						90.00
10	1	2023/7/30	MMR-1000	FRVG-10003	ATR	AM2	SO	2.00	50.00	100.00
11	TOTAL	2023/7/30	MMR-1000	FRVG-10003						100.00
12										

BUYING | SALES | SH2 | CASE1 | CASE2 | +

	A	B	C	D	E	F	G
1			FIRST DATE		LAST DATE		SHEET NAME
2			2023/7/28		2023/8/27		SALES
3							
4	ITEM	DATE	INVOICE	TOTAL			
5							
6							
7							
8							
9							
10							
11							

BUYING | SALES | SH2 | CASE1 | CASE2 | +

Instruction Type: Sheet-Level Manipulation

Answer Position: 'SH2'!A5:D10

Answer (spreadsheet):

	A	B	C	D	E	F	G
1			FIRST DATE		LAST DATE		SHEET NAME
2			2023/7/28		2023/8/27		SALES
3							
4	ITEM	DATE	INVOICE	TOTAL			
5	1	2023/7/30	FRVG-10002	90.00			
6	2	2023/7/30	FRVG-10003	100.00			
7	Total			190.00			
8							
9							
10							

Figure 17: A test case modified from the question in Figure 16.

https://doi.org/10.52202/079017-3007

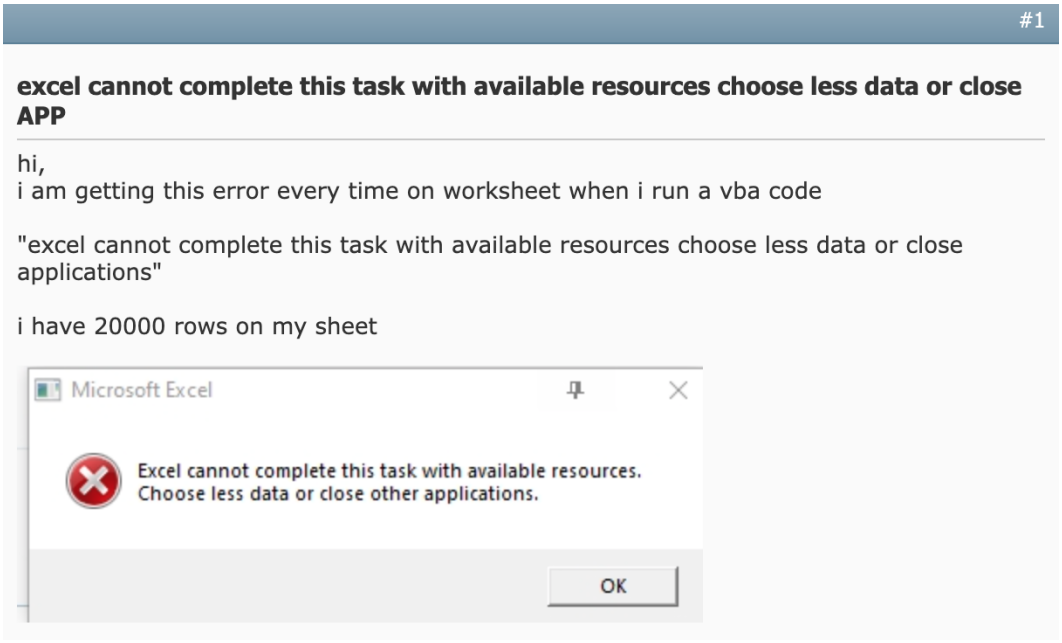


Figure 18: An example question about software usage, which are excluded from the dataset.

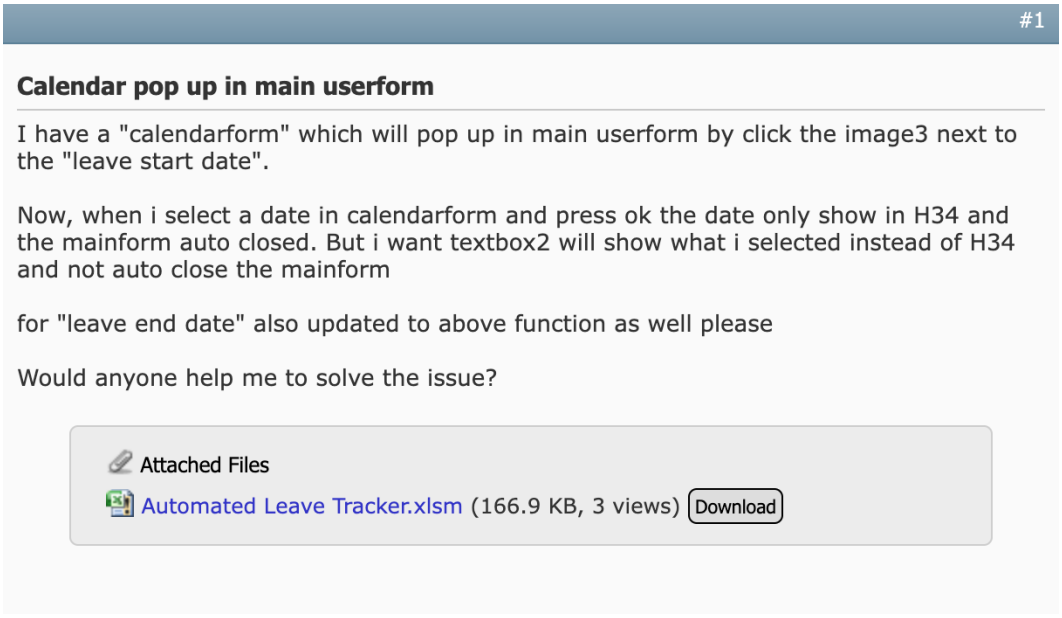


Figure 19: An example question related to component in Excel, which are excluded from the dataset.

https://doi.org/10.52202/079017-3007

	A	B	C	D	E	F
1	Invoice	Reference	line	Item	Sku	Tracking
2	10001	9837643	1	RPC0094-001	17343D	1Z4360098654321
3	10002	9837644	1	RPC0094-001	17343D	1Z4360098654322
4	10002	9837644	2	RPC0094-001	17263D	1Z4360098654322
5	10002	9837644	1	RPC0097-001	17343D	1Z4360098654323
6	10002	9837644	2	RPC0097-001	17263D	1Z4360098654323
7	10003	9837645	1	RPC0068-001	17232B	1Z4360098654324
8	10004	9837646	1	RPC0094-001	17343D	1Z4360098654325
9	10005	9837647	1	RPC0068-001	17232B	1Z4360098654326
10				None-001		
11				None-001		
12				None-001		
13				None-001		
14				None-001		
15				None-001		

Figure 20: A misclassified test case. The code solution successfully generates the correct answer (highlighted in yellow). However, additional content (highlighted in red) is also generated, resulting in incorrect judgments by our evaluation metrics.

	A	B	C	D	E	F	G	H	I
1	XXMTKGMA	2010022	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	2022/01/26	14862.42
2	XXMRKAMA	2010023						2022/02/1	20832.33
3	XXMTKGMB	2010024						2022/02/1	13923.56
4	XXERK1MK	2010025						2022/04/1	23528.95
5	XXERK1MG	2010025						2022/04/1	22557.38

Figure 21: A misclassified test case. The code solution successfully preserve the target rows (highlighted in yellow). However, additional table headers (highlighted in red) are also generated, resulting in incorrect judgments by our evaluation metrics.

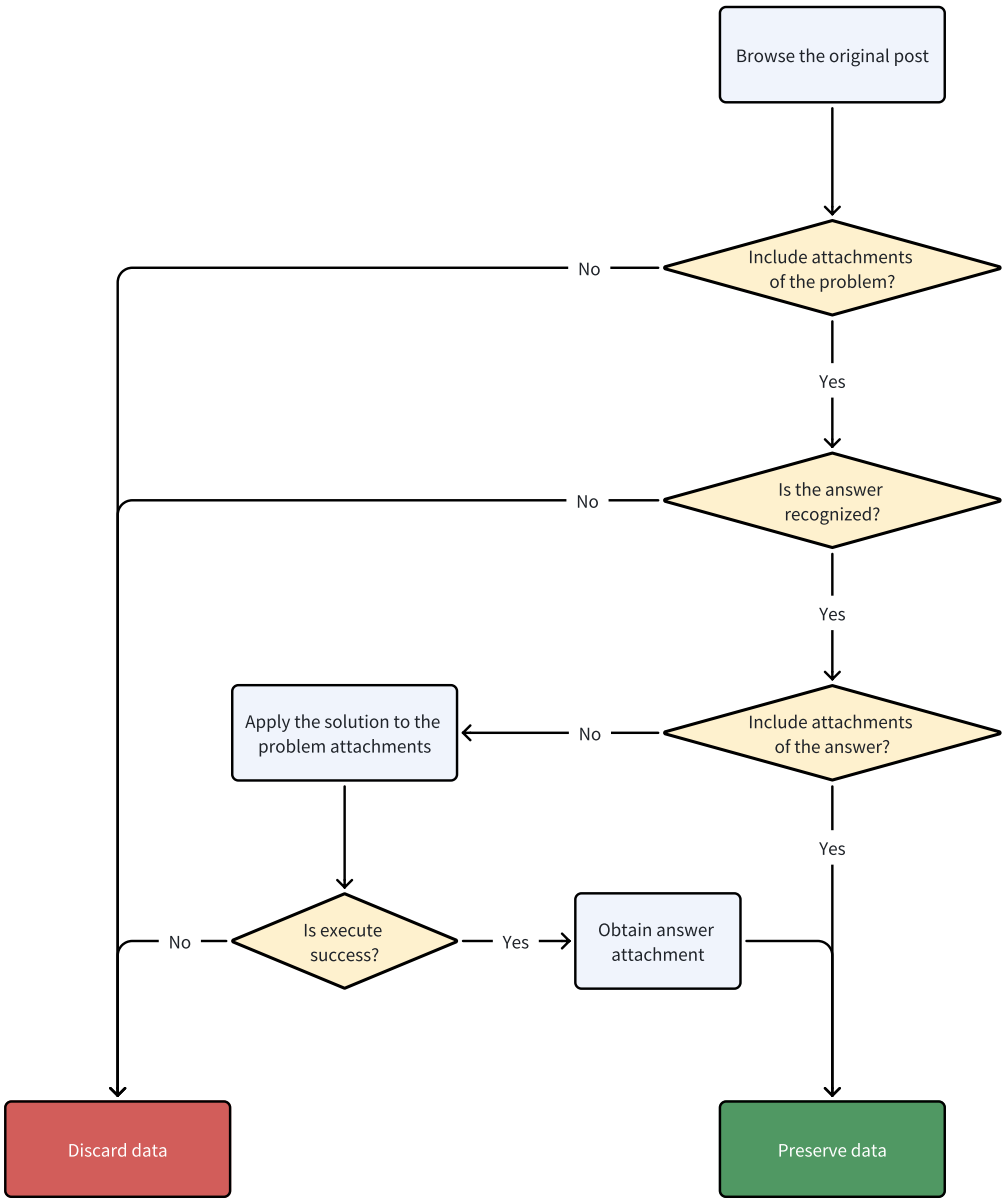


Figure 22: The flow chart of data selection process.

https://doi.org/10.52202/079017-3007

Annotation manual of data formatting task:

[1. Question Verification]
1. You need to compare the original post text with the question regenerated by the large language model based on the post content. If the question generated by the large model misses some information from the original post, you need to add this missing information to the question and fill it in under the column named "Revised Question".
2. You need to check if the attachment corresponding to the question contains any additional textual information that can be added to the question, and add it to the question when appropriate.

[2. Spreadsheet Verification]
1. You need to check if the spreadsheet provided by the asker already contains text-based answers in the target locations. If the text-based answers are identical to the results after the application of the solution, you need to delete some of the text-based answers and only keep a portion as examples.
2. You need to check if the data in the spreadsheet is too far from the A1 position, for instance, if the data in the spreadsheet starts from G200. In this case, you need to move the data in the spreadsheet closer to the A1 position for easier evaluation later on.

[3. Solution Verification and Application]
1. If the post provides an Excel file containing the answers, you need to check if the content in the file answers the asker's question.
2. If the post does not provide an Excel file with the answers, you need to apply the solution provided by the author (e.g., formulas or VBA code) to a specific location in the spreadsheet, which is generally specified by the responder.

[4. Data Annotation]
1. Solution Annotation: Record the solutions, such as formulas or VBA code.
2. Answer Position Annotation: Record the cells where the solutions are applied, such as A1:B10.
3. Spreadsheet Content Annotation: Based on the examples of multi-tables and non-standard tables we provide, determine if there are multi-tables and non-standard tables in the file.

Figure 23: The annotation manual of data formatting task.

Annotation manual of test case generation task:

We aim to modify the data within the solution's position to change the value in the cells of answer position. The steps are as follows:

1. Copy the original Excel file of the problem.
2. Locate the input position for the answer and the solution's position in the Excel summary sheet.
3. By understanding the question posed, you need to modify the data within the solution's position until the value in the answer cell changes. Record the location and sheet name of the changed cell in the Excel summary sheet (e.g., 'Sheet1'!A10). During modification, you may simulate corner case, such as deleting the cell's value, to mimic potential real-world incidents.
4. At this point, we obtain a new Excel file. Compared to the original Excel file, both the content of the table (i.e., data within the solution's position) and the values in the cells of answer position have changed.

For each data set, you need to modify it twice following the above steps, resulting in three pairs of test data (including the data from Task 1), named: 1_id_input.xlsx, 1_id_answer.xlsx, 2_id_input.xlsx, 2_id_answer.xlsx, 3_id_input.xlsx, 3_id_answer.xlsx.

Again, you have to make sure that the values in the cells that correspond to the solution's location in each of the three answer files are different.
- For the first modification: Ensure that at least one cell's value within the solution's position changes. Record the location and sheet name of one of the changed answer cells in the Excel summary sheet (e.g., 'Sheet1'!A10).
- For the second modification: Ensure that at least three cells' values within the solution's position change (if the solution's position contains fewer than three cells, change the value of just one cell). Record the locations and sheet names of three changed answer cells in the Excel summary sheet, separating the cell locations with spaces after the exclamation mark (e.g., 'Sheet1'!A10 A11 A12).

Figure 24: The annotation manual of test case generation task.

You are a spreadsheet expert who can manipulate spreadsheets through Python code.

You need to solve the given spreadsheet manipulation question, which contains six types of information:
- instruction: The question about spreadsheet manipulation.
- spreadsheet_path: The path of the spreadsheet file you need to manipulate.
- spreadsheet_content: The first few rows of the content of speadsheet file.
- instruction_type: There are two values (Cell-Level Manipulation, Sheet-Level Manipulation) used to indicate whether the answer to this question applies only to specific cells or to the entire worksheet.
- answer_position: The position need to be modified or filled. For Cell-Level Manipulation questions, this field is filled with the cell position; for Sheet-Level Manipulation, it is the maximum range of cells you need to modify. You only need to modify or fill in values within the cell range specified by answer_position.
- output_path: You need to generate the modified spreadsheet file in this new path.

Below is the spreadsheet manipulation question you need to solve:
instruction
{instruction}

spreadsheet_path
{spreadsheet_path}

spreadsheet_content
{spreadsheet_content}

instruction_type
{instruction_type}

answer_position
{answer_position}

output_path
{output_path}

You should generate Python code for the final solution of the question.

Figure 25: The prompt format used in single round setting.

https://doi.org/10.52202/079017-3007

You are a spreadsheet expert who can manipulate spreadsheets through Python code.

You need to solve the given spreadsheet manipulation question, which contains six types of information:
- instruction: The question about spreadsheet manipulation.
- spreadsheet_path: The path of the spreadsheet file you need to manipulate.
- spreadsheet_content: The first few rows of the content of speadsheet file.
- instruction_type: There are two values (Cell-Level Manipulation, Sheet-Level Manipulation) used to indicate whether the answer to this question applies only to specific cells or to the entire worksheet.
- answer_position: The position need to be modified or filled. For Cell-Level Manipulation questions, this field is filled with the cell position; for Sheet-Level Manipulation, it is the maximum range of cells you need to modify. You only need to modify or fill in values within the cell range specified by answer_position.
- output_path: You need to generate the modified spreadsheet file in this new path.

Below is the spreadsheet manipulation question you need to solve:
instruction
{instruction}

spreadsheet_path
{spreadsheet_path}

spreadsheet_content
{spreadsheet_content}

instruction_type
{instruction_type}

answer_position
{answer_position}

output_path
{output_path}

The solution of the question can be generate through a multi-turn interaction and you can do two types of actions.
1. Spreadsheet information acquisition: You can generate Python code to obtain the information in the spreadsheet file. In the next turn, the execution result of you Python code will provide to you.
2. Question solution generation: You can generate Python code for the final solution of the question. If error occur when executing code, the error traceback will provide to you for code refinement.

Figure 26: The prompt format used in multiple round setting.