SheetAgent: A Generalist Agent for Spreadsheet Reasoning and Manipulation via Large Language Models

Yibin Chen [*1] **Yifu Yuan** [*2] **Zeyu Zhang** [2] **Yan Zheng** [1] **Jinyi Liu** [2] **Fei Ni** [2] **Jianye Hao** [2]

Abstract

Spreadsheet manipulation is crucial for daily tasks and boosts efficiency significantly. Recent efforts have leveraged large language models (LLMs) for automating these tasks, yet complex, real-world scenarios involving multi-step reasoning and ambiguous instructions remain underexplored. We introduce **SheetRM**, a benchmark designed to address real-life challenges through long-horizon, multi-category tasks requiring intricate reasoning. To tackle these challenges, we propose **SheetAgent**, an innovative autonomous agent built on LLMs. SheetAgent features three synergistic modules: a Planner, an Informer, and a Retriever. These modules collaborate to enable advanced reasoning and precise spreadsheet manipulation autonomously, through iterative task reasoning and reflection. Our extensive testing shows that SheetAgent enhances pass rates by 20–30% across various benchmarks, significantly improving accuracy and showcasing superior reasoning capabilities in spreadsheet tasks.

1. Introduction

Tabular data is vital in fields like scientific research, finance, and marketing, where spreadsheets support tasks such as calculations, analysis, and visualization (Kandel et al., 2012; Hasan et al., 2020; Edeling et al., 2021). These tasks often require significant repetitive work and expertise (Gulwani, 2011; Chen et al., 2021b). While automation has simplified basic functions, complex real-world scenarios demand advanced reasoning, such as identifying specific content within multiple, ambiguously instructed sheets (Chen et al., 2021b; Li et al., 2023a). This complexity highlights the need for more sophisticated automation to enhance productivity in handling intricate spreadsheet tasks.

[*]Equal contribution [1]School of New Media and Communication, Tianjin University [2]College of Intelligence and Computing, Tianjin University. Correspondence to: Yan Zheng <yanzheng@tju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Designing an advanced method for spreadsheet manipulation requires integrating sophisticated reasoning and handling capabilities. Research shows that LLMs are adept at tasks like table question answering and fact verification (Ye et al., 2023; Jiang et al., 2023). This motivates us to question whether LLMs can effectively manage complex spreadsheet tasks involving challenging reasoning elements. Developing such an agent faces several hurdles: (1) **Dynamic Changes in Sheet Content**: Complex tasks often involve multiple operations that dynamically alter the content of a spreadsheet. Continuously reloading the entire spreadsheet into LLMs is unfeasible due to token limitations and the risk of generating inaccurate information (Cheng et al., 2022; Ye et al., 2023). (2) **Limited Table Understanding**: Although adept with natural language, LLMs have a constrained understanding of tabular data (Li et al., 2023b). (3) **Lack of Comprehensive Benchmark**: Current benchmarks like SheetCopilot do not sufficiently mimic real-world demands, often simplifying tasks and overlooking the need for multi-step reasoning and long-horizon operations (Li et al., 2023a).

To bridge the dataset gap in spreadsheet manipulation, we introduce SheetRM, a benchmark designed to develop and evaluate LLM-based agents equipped with advanced reasoning capabilities and precise manipulation skills. SheetRM also supports flexible automatic evaluation. Additionally, we present SheetAgent, a generalist agent that leverages LLMs for sophisticated sheet reasoning and manipulation. Our tests show that SheetAgent significantly outperforms existing baselines in spreadsheet tasks and competes well in table reasoning tasks without needing fine-tuning. Our contributions are manifold: (1) We introduce SheetRM, a benchmark tailored for LLM-based agents, fostering the development and evaluation of agents that can manipulate spreadsheets with complex reasoning needs. (2) We develop SheetAgent, a versatile LLM-based agent that enhances the interaction between humans and spreadsheets through combined capabilities in sheet manipulation and reasoning. (3) SheetAgent demonstrates a notable 20–30% increase in pass rates across various benchmarks, underscoring its enhanced reasoning and manipulation skills.

2. SheetRM Benchmark

Unlike existing datasets (Payan et al., 2023; Li et al., 2023a) that focus on precise spreadsheet manipulation, our objec-

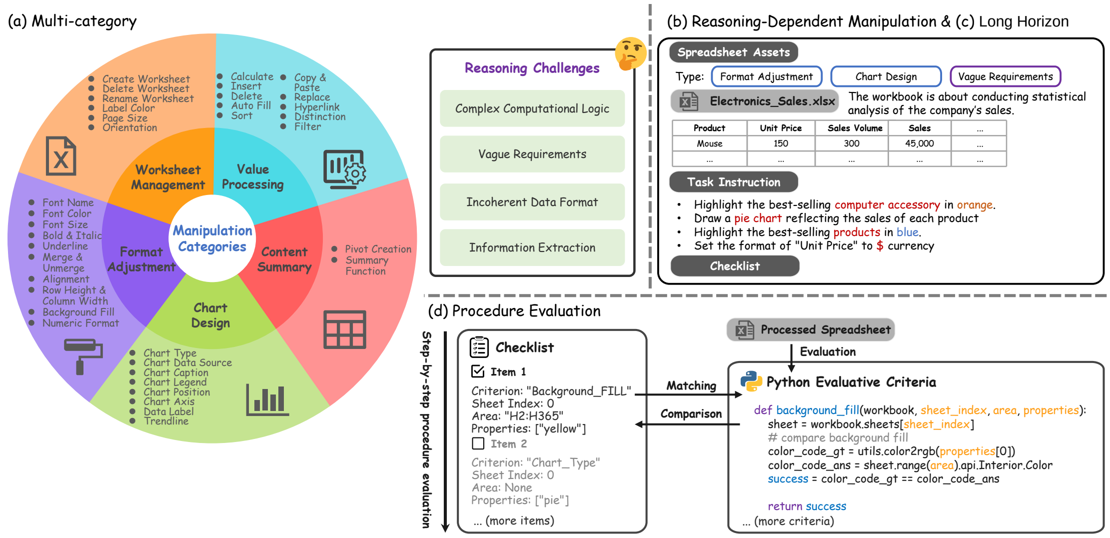

Figure 1. Overview and features of SheetRM: **(a) Multi-category**: SheetRM includes real-life tasks across multiple manipulation categories and reasoning challenges, testing both manipulation and reasoning skills. **(b & c) Long horizon and reasoning-dependent manipulation**: Features tasks with multiple components including spreadsheet data, task category descriptions, and long horizon task instructions. A checklist supports the evaluation of procedural execution. **(d) Procedure evaluation**: SheetRM conducts automatic step-by-step task evaluation using a checklist and specific evaluative criteria to ensure thorough procedure assessment.

tive with the SheetRM dataset is to create a more realistic setting by incorporating complex multi-step reasoning and ambiguous requirements, bridging the gap between simulated scenarios and real-world applications. We source real-life spreadsheets from the internet and identify a diverse set of operations commonly employed in practical scenarios. For detailed data construction procedure, please refer to Appendix A.1. Our SheetRM dataset is characterized by the following elements:

- **Reasoning-dependent manipulation**: Tasks require multi-step reasoning across spreadsheets.
- **Multi category**: The dataset includes 5 broad types and 36 subtypes of manipulation tasks, each paired with 4 reasoning challenges, testing both manipulation and reasoning skills.
- **Long horizon**: Tasks consist of several subtasks, posing challenges of dynamic spreadsheet changes.
- **Procedure evaluation**: An automated program evaluation method in SheetRM assesses not only the completion of the entire task but also the individual subtasks.

2.1. Task Schema

Each task in SheetRM is defined by the following three parts:

Spreadsheet Assets. Each task corresponds to a spreadsheet file. We summarize the contents of the spreadsheet in a one-sentence natural language overview as context, aiming to stimulate the internal knowledge of LLMs.

Task Instruction. A task instruction outlines the high-level goal expressed in natural language. A series of operations on the target spreadsheet is required to complete a task instruction.

Checklist. A task is paired with a checklist designed to evaluate its completion. Each item in the checklist corresponds to the evaluation of a fine-grained operation with tailored criteria. An example is illustrated in Fig. 1.

2.2. Automatic Evaluation

We have designed a model-free automatic evaluation system specifically tailored to assess fine-grained operations within our dataset. This system enables us to monitor the performance of intermediate sub-tasks. As shown in Fig. 1, the evaluation is structured around a checklist for each task instruction. Each item on the checklist is defined by a `(Criterion, Sheet Index, Area, Properties)` tuple. We use `(Sheet Index, Area)` to pinpoint the specific region within the spreadsheet for comparison. The `Criterion` is then applied to confirm if the identified region conforms to the specified `Properties`. This approach allows for a comprehensive evaluation of LLMs by meticulously assessing the execution of each task step.

3. SheetAgent Framework

To quantify the challenges posed by SheetRM, we introduce an LLM-based agent framework SheetAgent. As outlined in Fig. 2, SheetAgent consists of three key components: the Planner, the Informer, and the Retriever. We detail these

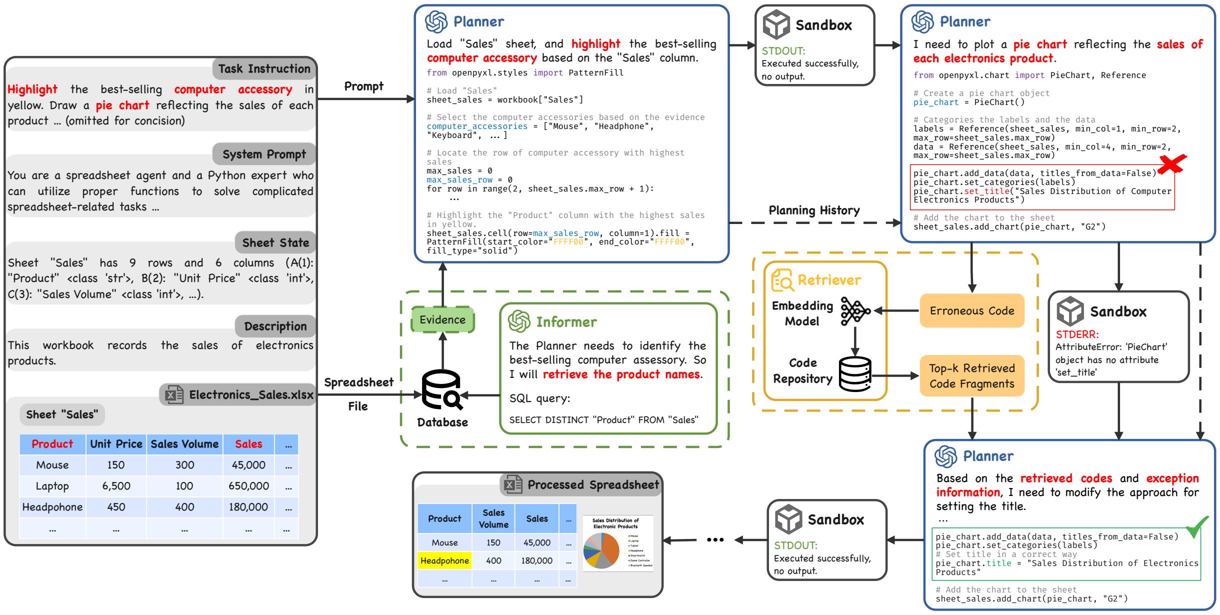

Figure 2. **An Overview of SheetAgent.** SheetAgent comprises three key components, including the Planner, the Informer, and the Retriever. The Planner interacts with the target spreadsheet in a ReAct way via a virtual sandbox. The Informer provides subtask-specific SQLs, the execution results of which serve as the evidence for the Planner to handle reasoning challenges. The Retriever is invoked to retrieve similar tutorial code snippets upon encountering an error, effectively correcting the error.

components in the following sections.

3.1. Proficient Spreadsheet Manipulation with Planner

Achieving precision in complex spreadsheet tasks requires a robust feedback mechanism. We employ a closed-loop planning process where the Planner, utilizing feedback and reflection, interacts with the spreadsheet in a ReAct manner. Initially, the Planner is fed with concatenated task instruction I, system prompt P^P, description D, and the initial sheet state s_0, which includes row and column counts, headers, and data types. At each step t, the Planner generates an action $a_t = \text{Planner}(a_t|I, P^P, D, s_t, h_{t-1})$, leveraging the current sheet state and previous planning history h_{t-1}. This action is tested in a sandbox, yielding feedback $o_t = \text{Sandbox}(a_t)$. If errors are detected, the Planner revises the action to $a_t^* = \text{Planner}(a_t^*|I, P^P, D, s_t, h_{t-1}, o_t)$. Successful actions update the spreadsheet to s_{t+1} and refresh the planning history to $h_t = (h_{t-1}, o_t, a_t)$. This process ensures accurate spreadsheet manipulation based on essential information rather than the entire dataset.

3.2. Accurate Spreadsheet Perception with Informer

Maintaining awareness of the spreadsheet's state alone is insufficient for the Planner to overcome the reasoning challenges illustrated in Fig. 1. For example, identifying computer accessories as per Fig. 2 requires discerning specific products amidst potentially frequent changes in the spreadsheet due to multiple modifications, challenging due to token limitations.

To address this, we introduced the Informer, which generates task-specific SQL queries. Initially, data from the target spreadsheet is extracted and stored in a lightweight database. At each step t, the Informer aims to identify entries that align with both the task instruction I and the state at that step. It combines the system prompt P^I, the task instruction I, and a sequence of previous actions $A_{t-1} = (a_1, \cdots, a_{t-1})$ into its input. The Informer then executes $q_t = \text{Informer}(q_t|P^I, I, A_{t-1}, s_t)$, where A_{t-1} helps trace the Planner's reasoning, enabling the generation of precise and robust SQL queries. The execution of these queries produces evidence e_t, which the Planner uses to more effectively perceive and interact with the spreadsheet, thus better addressing the reasoning challenges.

3.3. Robust Solution Generation with Retriever

The Retriever aids the Planner in task planning by providing error correction support through relevant code examples from a repository. We gather high-quality code snippets from GitHub and categorize them according to the manipulation types shown in Fig. 1. Using Milvus (Wang et al., 2021), an open-source vector database, these snippets are vectorized into a set $\boldsymbol{V}$ and stored. When an error is detected by the sandbox, the Retriever is activated. This process is defined as $\boldsymbol{S} = \text{top}_k(\cos(\mathcal{E}(C_q), vi)|vi \in \boldsymbol{V})$, where $\boldsymbol{S}$ represents the k highest similarity scores between the erroneous code snippet C_q and the vectors. The embedding function $\mathcal{E}$ utilizes any pre-trained language model to find the top-k similar code snippets C^k_{ret}. These are then used to enhance the Planner's replanning process with $a^*_t = \text{Planner}(a^*_t|I, P, D, s_t, ht-1, o_t, C^k_{ret})$, facilitating the generation of more accurate and reliable actions.

4. Experiment

4.1. Experiment Setup

Dataset. We employ a diverse array of benchmarks to evaluate SheetAgent's manipulation and reasoning capabilities. For manipulation, we use the *SheetCopilot Benchmark* (SCB) (Li et al., 2023a) and our own *SheetRM*. For assessing reasoning skills, we test on table reasoning datasets such as *WikiTableQuestions* (WTQ) (Pasupat & Liang, 2015) and *TabFact* (Chen et al., 2019).

Evaluation Metrics. For manipulation tasks, following SheetCopilot, we use Exec@1 and Pass@1 metrics, along with SubPass@1 to measure the success rate of subtasks within each task, assessing instruction following capabilities. For reasoning tasks, we utilize accuracy for WTQ and TabFact.

Baselines. We select various baselines for comprehensive comparison. Refer to Appendix C.1 for detailed introduction.

4.2. Main Results

SheetAgent enhances spreadsheet manipulation accuracy. Utilizing `gpt-3.5-turbo-0613` as the LLM backbone, SheetAgent aligns with SheetCopilot benchmarks. Despite a one-shot prompt limitation, Table 1 indicates that SheetAgent significantly surpasses SheetCopilot by achieving a 16.8 higher Pass@1, showcasing superior utilization of LLMs for spreadsheet tasks. Additionally, even without the Informer and Retriever, SheetAgent outperform other baselines in Exec@1 and Pass@1 metrics, proving the robustness of Python code generated by GPT-3.5 over VBA or custom APIs.

SheetAgent demonstrates advanced reasoning ability for more complex long-horizon spreadsheet tasks. Extensive testing on the SheetRM dataset reveals SheetAgent's capabilities in demanding scenarios, shown in Table 2. Notably, SheetAgent achieves a SubPass@1 of 69.1, more than doubling the score of SheetCopilot with GPT-4, and a Pass@1 of 30.8, demonstrating profound reasoning strength where SheetCopilot struggles. Furthermore, SheetAgent's Exec@1 of 92.5 markedly exceeds that of SheetCopilot, highlighting the durability and adaptability of its solutions.

SheetAgent achieves competitive performance in table reasoning tasks. We remove the Retriever in this setting. SheetAgent still outperforms in table reasoning tasks like WTQ as detailed in Table 3. SheetAgent outshines baselines and matches the best prompting-based methods, affirming the efficacy of the Planner and Informer. These components collaboratively enhance table structure and generate impactful SQLs, boosting the system's reasoning capabilities significantly. This synergy plays a vital role in SheetAgent's exceptional performance in table reasoning tasks.

Table 1. Performance comparison of different methods for SCB. VBA and SheetCopilot results are from Li et al. (2023a). † denotes results derived from a subset of SCB.

Method	Exec@1	Pass@1
VBA	77.8	37.1
SheetCopilot† (GPT-4)	65.0	55.0
SheetCopilot (GPT-3.5)	87.3	44.3
SheetAgent (GPT-3.5)	**94.1**	**61.1**
w/o Informer+Retriever	88.7	50.7

Table 2. Results of different methods for our SheetRM dataset. SheetAgent significantly surpasses other methods.

Method	Exec@1	Pass@1	SubPass@1
SheetCopilot (GPT-3.5)	66.7	0	15.8
SheetCopilot (GPT-4)	50.2	1.7	30.4
SheetAgent (GPT-3.5)	**92.5**	**30.8**	**69.1**

Table 3. Results of different methods on WTQ (denotation accuracy) and TabFact (accuracy) test set. The results are sourced from the original paper. Best results are highlighted in bold.

Method	WTQ	TabFact
Fine-tuning based LLMs		
TAPAS	48.8	81.0
UnifiedSKG	49.3	83.7
TAPEX	57.5	84.2
Prompting based LLMs		
Binder	61.9	85.1
DATER	**65.9**	85.6
StructGPT	52.2	**87.6**
SheetAgent (GPT-3.5)	63.3	84.8

5. Conclusion

In this work, we present SheetRM, a sophisticated and realistic benchmark designed to evaluate and foster the development of generalist agents adept at intricate spreadsheet manipulations and advanced reasoning tasks. Additionally, we introduce SheetAgent, a robust framework that utilizes the capabilities of LLMs to address these complex challenges. Our extensive experiments demonstrate the proficiency of SheetAgent in both reasoning and manipulation. We anticipate that SheetRM will become a foundational benchmark for advancing generalist agents focused on spreadsheet tasks. Moreover, we envision SheetAgent facilitating the automation of tedious and repetitive spreadsheet transactions, significantly enhancing workflow efficiency.

References

Chang, Y., Wang, X., Wang, J., Wu, Y., Zhu, K., Chen, H., Yang, L., Yi, X., Wang, C., Wang, Y., et al. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Chen, W., Wang, H., Chen, J., Zhang, Y., Wang, H., Li, S., Zhou, X., and Wang, W. Y. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*, 2019.

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

Cheng, Z., Xie, T., Shi, P., Li, C., Nadkarni, R., Hu, Y., Xiong, C., Radev, D., Ostendorf, M., Zettlemoyer, L., et al. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*, 2022.

Edeling, A., Srinivasan, S., and Hanssens, D. M. The marketing–finance interface: A new integrative review of metrics, methods, and findings and an agenda for future research. *International Journal of Research in Marketing*, 38(4):857–876, 2021.

Gulwani, S. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices*, 46 (1):317–330, 2011.

Hasan, M. M., Popp, J., and Oláh, J. Current landscape and influence of big data on finance. *Journal of Big Data*, 7 (1):1–17, 2020.

Herzig, J., Nowak, P. K., Mueller, T., Piccinno, F., and Eisenschlos, J. 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.

Jiang, J., Zhou, K., Dong, Z., Ye, K., Zhao, W. X., and Wen, J.-R. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*, 2023.

Kandel, S., Paepcke, A., Hellerstein, J. M., and Heer, J. Enterprise data analysis and visualization: An interview study. *IEEE transactions on visualization and computer graphics*, 18(12):2917–2926, 2012.

Li, H., Su, J., Chen, Y., Li, Q., and Zhang, Z. Sheetcopilot: Bringing software productivity to the next level through large language models. *arXiv preprint arXiv:2305.19308*, 2023a.

Li, P., He, Y., Yashar, D., Cui, W., Ge, S., Zhang, H., Fainman, D. R., Zhang, D., and Chaudhuri, S. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*, 2023b.

Liu, Q., Chen, B., Guo, J., Ziyadi, M., Lin, Z., Chen, W., and Lou, J.-G. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*, 2021.

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

Pasupat, P. and Liang, P. 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.

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

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., Guo, X., Li, C., Xu, X., et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 2614–2627, 2021.

Xie, T., Wu, C. H., Shi, P., Zhong, R., Scholak, T., Yasunaga, M., Wu, C.-S., Zhong, M., Yin, P., Wang, S. I., et al. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 602–631, 2022.

Ye, Y., Hui, B., Yang, M., Li, B., Huang, F., and Li, Y. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 174–184, 2023.

A. Details of SheetRM Benchmark

A.1. Dataset Construction

We gather and refine publicly available spreadsheets through a selection and cleaning process. Tasks are generated with both human and GPT-4 annotation. All the tasks are attached with verified answers, which enables model-free evaluation. The statistics of our dataset are shown in Table 4. We refer to Appendix A.2 for more detailed statistics.

Table 4. Basic statistics of SheetRM.

Item	Count
# Spreadsheet[1] Files	25
# Sheets	83
# Average Rows per File	296.37
# Average Columns per File	24.15
# Task Instructions	201
# Subtasks	1024

Spreadsheet Files Collection. We initially collect real-world spreadsheets from a public examination question bank. We filter out files that are protected, corrupted, or otherwise inaccessible. To ensure diversity, we select files covering multiple meaningful domains. We require most files to have at least 2 sheets, with each file containing a minimum of 20 rows and 5 columns in total to guarantee scale sufficiency. Besides, any external dependencies (e.g., images and web pages) are converted into natural language or embedded sheets if feasible. We finally shortlist 25 spreadsheets with a total of 83 sheets. On average, each spreadsheet contains 296.37 rows and 24.15 columns.

Task Generation. We begin by referring to websites about spreadsheet software skills and consult corporate staff about commonly used spreadsheet operations in their work. As shown in Fig. 1, we conclude five coarse operation categories and their fine-grained specific operations for manipulation. Drawing insights from common table reasoning datasets like WikiTableQuestions and TabFact, we summarize four challenges in the process of sheet reasoning: (1) complex computation logic, (2) vague requirements, (3) incoherent data format and (4) information extraction. We detail these challenges in the Appendix A.3. Then, we instruct GPT-4 to propose *realistic* tasks that mimic user requests adhering to four guidelines: the tasks should only involve predefined operations, cover diverse manipulation categories, exhibit a long-horizon nature by encompassing multiple subtasks, and incorporate at least one subtask that presents the specified reasoning challenges. This process yields a compilation of 1,657 subtasks. We eliminate semantically redundant entries for identical files to maintain uniqueness. To guarantee quality, our internal annotators manually validate subtasks using programming and specialized software such as Excel. Certain unreasonable subtasks are excluded throughout this process. By combining these subtasks for different spreadsheets considering horizon and complexity, we ultimately assemble 201 task instructions, encompassing a total of 1024 subtasks. Full prompts are available in Appendix E.1.

A.2. Detailed Statistics of Dataset

Spreadsheet Files. We provide more detailed statistics of our SheetRM dataset. We collect spreadsheets covering multiple fields. As illustrated in Fig. 3 (Left), we categorize these spreadsheet files into five main fields, reflecting the significant areas where spreadsheets are frequently employed to handle a variety of tasks. We manually annotate a short natural language description as a summary for each spreadsheet file, aiming to stimulate inherent knowledge of LLMs. Each description provides an overview for LLMs to better understand the background information. We provide the descriptions in Table 5.

Task Instruction. We cluster the commonly used operation when working with spreadsheets into five categories, namely **Value Processing**, **Worksheet Management**, **Format Adjustment**, **Chart Design**, and **Content Summary**. For each manipulation category, we further break it down into fine-grained operations. We believe these operations can cover most spreadsheet affairs. The description of these operations is introduced in Table 6. Fig. 3 (Right) demonstrates the distribution of verb-noun phrases within our 201 task instructions. We highlight the ten most frequent root verbs and their four primary associated nouns, showcasing the diversity of task instructions in the SheetRM dataset. Additionally, we show the distributions of the number of manipulation categories and subtasks for these task instructions (see Fig. 4 (Left)). The majority of tasks span 2 or 3 manipulation categories, with a decent portion encompassing 4 categories, underscoring the

[1]A spreadsheet is a collection of sheets that are organized into a document. A table represents a structured arrangement of data in rows and columns. Each sheet within the spreadsheet contains a table.

diversity of tasks in the SheetRM dataset. We further count the number of subtasks in each task. As displayed in Fig. 4, each task includes at least 2 sub-tasks, with the most complex extending to 10. Predominantly, the tasks vary in length from 3 to 7. This reflects the long horizon feature of SheetRM, which poses a significant challenge to LLMs.

A.3. Explanation of Reasoning Challenges

Our SheetRM dataset stands out from other spreadsheet manipulation collections due to its emphasis on reasoning-dependent manipulation. Specifically, each task incorporates reasoning challenges. We draw inspiration from several popular table reasoning tasks, including table question answering datasets WikiTableQuestions and FeTaQA (Nan et al., 2022), and table fact verification task TabFact. We analyze cases within these datasets that most models struggled with and identify four types of reasoning challenges, namely **Complex Computational Logic**, **Vague Requirements**, **Incoherent Data Format**, and **Information Extraction**. We find that these reasoning challenges are prevalent in real-world spreadsheet manipulation tasks due to the diversity of human expression. Thus, integrating practical insights, we incorporate these reasoning challenges into our spreadsheet manipulation tasks. We elaborate these challenges with descriptions and specific examples:

Complex Computational Logic

Description:
Problems that require more than one reasoning steps to be solved.
Example Sheet:

Name	Date of Entry	Educational Qualification	Salary
Alice	3/1/2001	Master	11,100
Bob	12/1/2006	Bachelor	10,350
...	...	...	...
John	1/9/2011	Doctor	41,100

Instruction:
Which period, 2001-2006 or 2007-2012, had a higher proportion of employees with bachelor's degrees? For the period with the higher proportion, calculate the average salary of the undergraduate employees and put it in cell E1.
Challenge:
To fulfill this instruction, the capability of multi-step reasoning is required.

Vague Requirements

Description:
Problems that refer to incomplete or ambiguous specifications which lack clarity and precision, making it challenging to understand and fulfill the intended goals or objectives.
Example Sheet:

BookID	Book Name	Unit Price
BK-83024	VB Programming	38
BK-83026	Access Programming	35
...	...	...
BK-83029	Network Technology	43

Instruction:
Highlight database-related books in yellow.
Challenge:
To fulfill this instruction, Reasoning over the sheet contents to identify which books are relevant to the database.

Incoherent Data Format

Description:
Problems that arise when the description provided pertains to the spreadsheet data, yet the units or formats mentioned do not align with those represented in the spreadsheet.
Example Sheet:

Name	Date of Birth
Alice	12/27/1964
Bob	9/28/1974
...	...
John	7/19/1987

Instruction:
Mark the names of employees born after 1985-1-1 in red.
Challenge:
To fulfill this instruction, the “Date of Birth” column should be inferred to align the format.

Information Extraction

Description:
Problems that require specific information to be extracted from the spreadsheet.
Example Sheet:

Venue	Opponent	Final Score
Memphis, Tennessee, USA	Jim Courier	7-5, 6-7, 6-7
Australian Open, Melbourne, Australia	Pete Sampras	6-7, 4-6, 4-6
...	...	...
Estoril, Portugal	Albert Costa	6-2, 3-6

Instruction:
Extract the scores from the first round of the finals into the new column ”First Round Score”.
Challenge:
To fulfill this instruction, Information about the “Final Score” is required to determine how to extract the first round score.

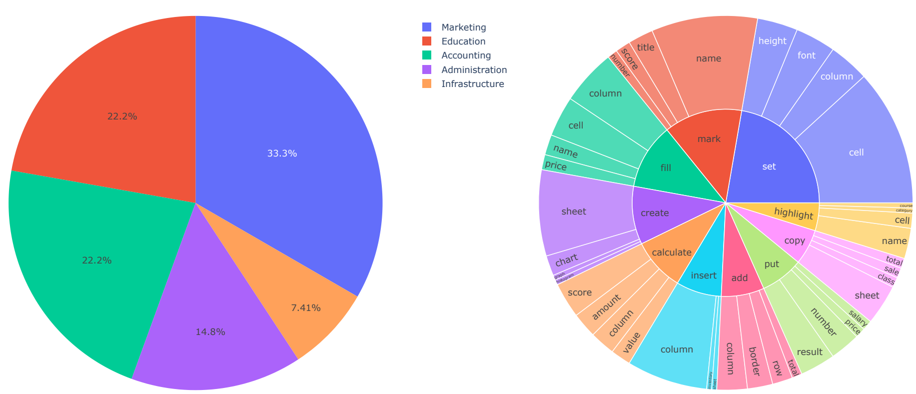

Figure 3. (Left) Distribution of fields to which the spreadsheet files belong. (Right) An illustration of verb-noun phrases in the task instructions. We count the top 10 most frequent root verbs and their associated nouns, ranking the top four for each. These verb-noun combinations showcases the diversity of the generated instructions.

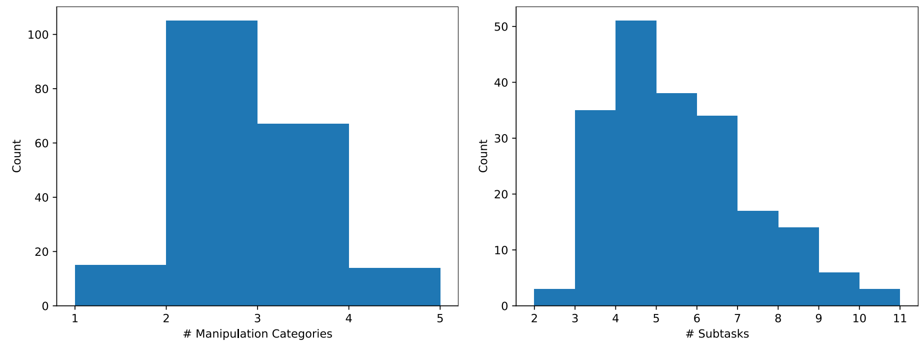

Figure 4. (Left) Distribution of manipulation categories in each task. (Right) Histogram of the task length.

Table 5. A short natural language description of the spreadsheet files we collect in SheetRM dataset.

Spreadsheet File	**Description**
BookSales	This workbook presents data related to book sales.
StudentsGrade	This workbook is about organizing and analyzing student transcripts for first-grade students.
ABProductSales	This workbook presents data related to product A and B.
Reimbursement	The workbook shows the company's travel expense reimbursement status for the year 2013.
ElectronicsSales	The workbook is about conducting statistical analysis of the company's sales.
PayrollSummary	The workbook is the March 2014 employee salary sheet.
TeachingFees	This workbook shows the teaching situation and instructor hourly fees for the courses in the Teaching Research Office in the year 2012.
Deposit	The workbook is a bank deposit journal.
ComputerBookSales	The workbook depicts the sales figures for computer-related books in December 2012.
ScienceMajorGrade	The workbook shows the final exam grades for the Information and Science major.
PersonnalInformation	This workbook is the personnel file information of company employees.
ComputerBookSales2	This workbook represents the sales statistics of computer-related books.
AppliancesSales	This workbook shows the sales statistics of various household appliances.
DepartmentSales	This workbook documents the sales performance of company's products in the first half of the year.
QuartersSales	This workbook summarizes the sales performance for the first two quarters.
FinalGrade	This workbook provides a detailed analysis of students' final grades.
ParkingFees	This workbook keeps track of parking fees and the associated rates.
LivingCosts	This workbook displays an individual's monthly expense report.
StudentsGrade2	This workbook displays the grades for each subject in the class.
LawMajorGrade	This workbook presents the final grade analysis of law students from the 2012 cohort.
YearsSales	This workbook documents the sales statistics of company products in 2012 and 2013.
YearEndSalary	This workbook provides the year-end salary details of employed staff members.
AirQuality	This workbook illustrates the air quality data for major cities in China.
SalesAndPurchase	This workbook is a record of this year's sales and purchase data.
PersonnelChange	This workbook contains the personal details of company employees for the year 2019, including their entry and departure information.

Table 6. Description of each fine-grained operation involved in SheetRM dataset.

Manipulation Category	**Operation**	**Description**
Value Processing	Calculate	Calculations and statistics.
	Insert	Insert rows or columns.
	Delete	Delete cells, rows or columns.
	Auto Fill	Fill according to the control relationship.
	Sort	Sort rows or columns in ascending or descending order.
	Copy & Paste	Copy and paste cell values.
	Replace	Replace the values of a cell at a specified location.
	Hyperlink	Set up hyperlinks.
	Distinction	Remove duplicates.
	Filter	Filter specified cells according to certain conditions.
Worksheet Management	Create Worksheet	Create a new worksheet.
	Delete Worksheet	Delete the specified worksheet.
	Rename Worksheet	Rename the specified worksheet.
	Label Color	Modify the color of worksheet name labels.
	Page Size	Modify page size.
	Orientation	Set the page orientation.
Format Adjustment	Font Name	Set the font category.
	Font Color	Set the font color.
	Font Size	Set the font size.
	Bold & Italic	Set the font to be bold or slanted.
	Underline	Underline cell contents.
	Merge & Unmerge	Merge or split cells.
	Alignment	Align cells horizontally or vertically.
	Row Height & Column Width	Set cell row height or column width.
	Background Fill	Set cell background fill color.
	Numeric Format	Set cell number formatting.
Chart Design	Chart Type	Set the Chart Type.
	Chart Data Source	Set the data source for the chart.
	Chart Caption	Set the title of the chart.
	Chart Legend	Set the Chart Legend.
	Chart Position	Specify where to place the chart.
	Chart Axis	Set the axes of a chart.
	Data Label	Set data labels for charts.
	Trendline	Add a trendline to the chart.
Content Summary	Pivot Creation	Create pivot table.
	Summary Function	Set statistical functions of the pivot.

B. Explanations of The Code-Centric Design in Planner

What is generated by the Planner is crucial for precise manipulation. Li et al. (2023a) introduces a set of virtual APIs as the action space for its proposed agent. However, these APIs lack scalability and are prone to hallucinations when invoked due to conflicts with the inherent knowledge of LLMs. Considering the strong code generation capabilities of LLMs (Chang et al., 2023), we assign the Planner to generate codes to control spreadsheets. During the process of dataset construction (Appendix A.1), we find that Python, compared with VBA, is suitable for manipulating spreadsheets and aligns well with existing training corpus (Chen et al., 2021a; Roziere et al., 2023) for LLMs. We assess various Python libraries for spreadsheet control and decide to primarily use *openpyxl*[2] and *pandas*[3] as a combination of them can cover all operations shown in Fig. 1.

C. Experimental Details

C.1. Selection of Baselines

For SCB, we compare with SheetCopilot (Li et al., 2023a), an LLM-based agent framework, and VBA (Li et al., 2023a), a method that generates and runs VBA code. For the SheetRM dataset, we select SheetCopilot as the baseline with different LLM backbones. For the table-based reasoning task, we group the baselines into two aspects, i.e., fine-tuning based LLMs and prompting-based LLMs. For WikiTableQuestions and TabFact, we select TAPAS (Herzig et al., 2020), UnifiedSKG (Xie et al., 2022), and TAPEX (Liu et al., 2021) as fine-tuning based LLMs, Binder (Cheng et al., 2022), DATER (Ye et al., 2023) and StructGPT (Jiang et al., 2023) as prompting-based LLMs.

C.2. Implementation Details of SheetAgent

In the experiments, we adopt GPT-3.5 and GPT-4[4] (i.e., `gpt-4-0613`) as the LLM backbones for our SheetAgent. Multiple versions of GPT-3.5 are involved for alignment. Specifically, for SCB, WikiTableQuestions and TabFact, we use `gpt-3.5-turbo-0613`. For our SheetRM, we employ `gpt-3.5-turbo-1106`. We adopt 1 in-context example for SCB and 2 for the rest datasets. The version of Milvus we use is `2.3.1`. Note that the Planner and the Informer in SheetAgent do not share the backbone.

C.3. Table Representation Examples

We investigate what representation formats can better help LLMs to reason over tables. We explore 4 prevalent table formats: JSON, DFLoader, Markdown, and HTML. We provide an illustration of these formats, as shown in Fig. 5. Notably, DFLoader is represented by the corresponding Python code snippet that utilizes the *pandas* DataFrame API to define the table.

D. Ablation Study

D.1. Ablation of proposed Components in SheetAgent.

Table 7 showcases the impact of the designed modules for SheetRM. We observe a dramatic decline of Pass@1 without Informer, which indicates that Informer can perform precise queries in the face of dynamically changing spreadsheets, highlighting its significance for tasks involving reasoning challenges. The Exec@1 drops sharply due to the removal of Retriever, demonstrating that high-quality tutorial examples enable the Planner to generate more robust and reliable code. Finally, when both the Informer and Retriever are removed, SheetAgent shows the poorest performance across all metrics, underscoring that both strong reasoning and precise manipulation capabilities are indispensable for tackling the challenging tasks of SheetRM. Interestingly, even with only the Planner module, SheetAgent maintains decent performance compared to SheetCopilot. This highlights the benefits of a code-centric approach over solely relying on language commands.

[2]`https://openpyxl.readthedocs.io`
[3]`https://pandas.pydata.org`
[4]`https://platform.openai.com/docs/models`

HTML

```
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Age</th>
      <th>Occupation</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>28</td>
      <td>Engineer</td>
    </tr>
    <tr>
      <td>Emily</td>
      <td>36</td>
      <td>Teacher</td>
    </tr>

      ...
  </tbody>
</table>
```

Markdown

```
| Name    |   Age | Occupation   |
|:--------|------:|:-------------|
| John    |    28 | Engineer     |
| Emily   |    36 | Teacher      |
| Michael |    29 | Doctor       |
|   ...   |  ...  |     ...      |
```

DFLoader

```
pd.DataFrame({
    "Name": ["John", "Emily", "Michael", ...],
    "Age": [28, 36, 29, ...],
    "Occupation": ["Engineer", "Teacher", "Doctor", ...]
})
```

JSON

```
{
    {"Name":"John","Age":28,"Occupation":"Engineer"},
    {"Name":"Emily","Age":36,"Occupation":"Teacher"},
    {"Name":"Michael","Age":29,"Occupation":"Doctor"},
    ...
}
```

Figure 5. An illustration of 4 different table representations we use in our experiments.

D.2. Ablation Study on Temperature

We conduct evaluations of our method using the proposed SheetRM dataset under varying conditions by adjusting the temperature settings to investigate the impact of temperature on the performance of LLMs. For these experiments, `gpt-3.5-turbo-1106` is selected as the LLM backbone. Our findings reveal that our method, SheetAgent, achieves its best performance at a temperature of 0.0, with minor performance fluctuations observed at a temperature of 0.2. However, a noticeable decline in performance across all metrics occurs when the temperature is increased to 0.4. This trend suggests that higher temperature settings lead to more unpredictable outcomes from SheetAgent, reflecting a decrease in the stability and reliability of the solutions it generates.

Table 7. Ablation study of different proposed components in SheetAgent on SheetRM dataset.

Method	**Exec@1**	**Pass@1**	**SubPass@1**
SheetAgent (GPT-3.5)	92.5	30.8	69.1
w/o Informer	96.5	12.4	65.5
w/o Retriever	83.1	20.9	62.7
w/o Informer+Retriever	87.1	10.0	56.7

E. Prompts

E.1. Prompt for Subtask Generation

The subtask generation stage involves two aspects, namely generating subtasks with diverse fine-grained operations, and generating subtasks with four reasoning challenges. Fig. 6 lists the prompt for the first aspect. To ensure the generation quality, we prompt GPT-4 to choose 4-5 fine-grained operation at a time. To narrow the gap with realistic requirements, we ask GPT-4 to express in a tone of real-life users. Moreover, an in-context example is provided to teach GPT-4. With these prerequisites, GPT-4 can continuously generate diverse and sufficient subtasks. Fig. 7-10 showcase the prompts for

Table 8. Ablation study on the temperature of LLM.

Temperature	Exec@1	Pass@1	SubPass@1
0.0	92.5	30.8	69.1
0.2	94.5	28.9	67.4
0.4	89.1	25.9	63.7

generating subtaks with 4 reasoning challenges. Particularly, GPT-4 is prompted under the principle that the generated subtasks should only be solved by reasoning over spreadsheets. This guarantees the existence of reasoning factors in the subtasks to some extent. For the last three challenges, we ask GPT-4 to annotate response with its thinking logic so that we could verify that it makes sense.

E.2. Prompt for Planner

Fig. 11 lists the prompt template for the Planner in SheetAgent. The Planner is prompted to mainly use *openpyxl* and *pandas* to manipulate spreadsheets. We also prompt Planner to reason and plan in a ReAct way. It can invoke *Python* tool to interact with a Python sandbox for solution evaluation, and *Answer* tool to submit the answer corresponding to the question.

E.3. Prompt for Informer

The prompt for the Informer is shown in SheetAgent. To increase the robustness and reliability of generated SQLs, we provide the Informer with the table schemas of all sheets, along with 3 example rows.

Generation of Subtasks

System prompt
Role
As a spreadsheet expert, you have the ability to formulate specific questions for given spreadsheets. These questions are utilized to evaluate the large language model's capabilities to manipualte spreadsheets.

Constraints
1.Choose 4-5 fine-grained operations from the classification below. Use the provided spreadsheet to create tasks, and then merge them into a complete question.
2.Generate an appropriate number of questions each time.
3.Generate questions from the user's perspective, considering elements such as thought process and tone of speech.
4.Simplify the language by focusing only on subproblems composed of fine-grained operations.
5.List the fine-grained operations involved behind each problem. For example, (fine-grained operation：Numeric Format, Auto Fill, Font Color)
6.Make each question more complex and comprehensive.

Fine-grained operations
Here are the fine-grained operations you can choose within the five categories:
A.Value Processing:Calculate,Insert,Delete,Auto Fill,Sort,Copy & Paste,Replace,Hyperlink,Distinction,Filter
B.Worksheet Management:Create Worksheet,Delete Worksheet,Rename Worksheet,Label Color,Page Size,Orientation
C.Format Adjustment:Font Name,Font Color,Font Size,Bold & Italic,Underline,Merge & Unmerge,Alignment,Row Height & Column Width,Background Fill,Numeric Format
D.ChartDesign:Chart Type,Chart Data Source,Chart Caption,Chart Legend,Chart Position,Chart Axis,Data Label,Trendline
E:ContentSummary:Pivot Creation,Summary Function

In-context example
I will give you an example first:
Given a spreadsheet:
Sheet name "Order Details":
Orderid	Date	Bookstore Name	Bookid	Bookname	Unit Price	Sales Volume	Subtotal	Purchaser
BTW-08634	2012/10/31	Dingsheng Bookstore	BK-83024	VB Language Programming	38	36	1,368.0	Jiguang Ma
BTW-08633	2012/10/30	Boda Bookstore	BK-83036	Database Principles	37	49	1,813.0	Ruiguo Zhang
BTW-08632	2012/10/29	Boda Bookstore	BK-83032	Information Security Technology	39	20	780.0	Qianghui Huang

Referring to the details provided in the table above, I'll present the following complex computational logic questions:
1.In sheet "Order Details", adjust "Unit Price" and "Subtotal" to accounting with 2 decimal places and CNY currency symbol format. Fill the "Bookname" column based on "Bookid". Identify the bookstore with the largest decrease in "Subtotal" in January to October 2012 compared to January to October 2011.（fine-grained operation：Numeric Format，Auto Fill, Font Color）
2.In sheet "Order Details", fill in the "BookiD" based on the "Bookname". Add a column named "Adjusted Quantity" to the last column of the sheet. Calculate the data in this column by multiplying 20 with the current values in the "Subtotal" column and then dividing by the maximum value of Subtotal.（fine-grained operation：Auto Fill, Calculate, Shading, Font Color）

User instruction
Now it's your turn.
Given a spreadsheet:
Sheet name "Grade":
Student Number	Name	Class	Chinese	Mathematics	English	Biological	Geography	History	Politics	Total Score	Average Score
120305	Hongwei Bao	class 3	91.5	89	94	92	91	86	86	629.5	89.93
120203	Wandi Chen	class 2	93	99	92	86	86	73	92	621	88.71
120206	Beida LI	class 2	100.5	103	104	88	89	78	90	652.5	93.21

GPT response
Sort the students' names alphabetically. Identify the student who has the highest average score after sorting. (fine-grained operation: Sort, Calculate)
Merge cells to group students based on their "Class". Find out which class has the highest total sum of all subjects combined. (fine-grained operation: Merge & Unmerge, Calculate, Sort)
Create a new column called "Science Total", which sums up the scores for "Biological", "Geography", and "History". Then, find the student with the lowest sum in "Science Total". (fine-grained operation: Calculate, Insert)
Adjust the "Total Score" and "Average Score" columns to display only one decimal place. After this adjustment, determine if there are any students whose average score is exactly equal to the rounded "Average Score". (fine-grained operation: Numeric Format, Operation)

Figure 6. Prompt for generating subtasks with diverse fine-grained operations.

Generation of Subtasks with Reasoning Challenges

Complex Computational Logic

System prompt

Role

As a spredsheet expert, you have the ability to formulate specific questions for given spreadsheets. These questions are utilized to evaluate the large language model's comprehension of spreadsheets.

Constraints

1.Begin by grasping the content of the provided worksheets, then formulate questions related to "Complex Computational Logic." These types of questions necessitate the use of multiple statistical procedures for resolution.
2.Keep the focus of the question on asking, without delving into the specifics of the procedural steps.
3.The guiding principle for formulating questions is that they should only be solvable by referencing the content of the worksheets.
4.Generate an appropriate number of questions each time you approach the task.
5.Conclude each question with a fine-grained operation, integrating it with the problem to present the result.
6.Elevate the complexity and comprehensiveness of each question.

Fine-grained operations

Here are the fine-grained operations you can choose within the five categories:
A.Value Processing: Calculate,Insert,Delete,Auto Fill,Sort,Copy & Paste,Replace,Hyperlink,Distinction,Filter
B.Worksheet Management: Create Worksheet,Delete Worksheet,Rename Worksheet,Label Color,Page Size,Orientation
C.Format Adjustment: Font Name,Font Color,Font Size,Bold & Italic,Underline,Merge & Unmerge,Alignment,Row Height & Column Width,Background Fill,Numeric Format
D.ChartDesign: Chart Type,Chart Data Source,Chart Caption,Chart Legend,Chart Position,Chart Axis,Data Label,Trendline
E:ContentSummary: Pivot Creation,Summary Function

In-context example

I will give you an example first:
Given a spreadsheet
Sheet name "Order detail":
Orderid	Date	Bookstore Name	Bookid	Bookname	Unit Price	Sales Volume	Subtotal	Purchaser
BTW-08634	2012/10/31	Dingsheng Bookstore	BK-83024	VB Language Programming	38	36	1,368.0	Jiguang Ma
BTW-08633	2012/10/30	Boda Bookstore	BK-83036	Database Principles	37	49	1,813.0	Ruiguo Zhang
BTW-08632	2012/10/29	Boda Bookstore	BK-83032	Information Security Technology	39	20	780.0	Qianghui Huang

Referring to the details provided in the table above, I'll present the following complex computational logic questions:
1.In sheet "Statistical Report", identify the quarter with the highest book sales in 2011, then determine which bookstore has the most sales in this quarter, and fill its name of bookstore in B6 cell.（fine-grained operation：Calculate）
2.Identify the bookstore with the largest decrease in "Subtotal" in January to October 2012 compared to January to October 2011. Then, determine the best-selling book with the most volume in that year for this bookstore and highlight its row in red font in the "Statistical Report" sheet.（fine-grained operation：Font Color）

User instruction

Given a spreadsheet:
Sheet name "Grade":
Student Number	Name	Class	Chinese	Mathematics	English	Biological	Geography	History	Politics	Total Score	Average Score
120305	Hongwei Bao	class 3	91.5	89	94	92	91	86	86	629.5	89.93
120203	Wandi Chen	class 2	93	99	92	86	86	73	92	621	88.71
120206	Beida LI	class 2	100.5	103	104	88	89	78	90	652.5	93.21

GPT response

From the "Grade" sheet, calculate the average score of each subject across all classes, identify the subject with the lowest class average, and replace the title of that subject with "Needs Improvement" in cell D1.（fine-grained operation：Replace）
In the "Grade" sheet, determine the student with the highest total score for each class, then create a new worksheet named "Top Students" and copy & paste the information of these students into the new worksheet.（fine-grained operation：Copy & Paste）
On the "Grade" sheet, filter the students who scored above 90 in English and then sort these records in descending order by their Total Score. Once sorted, apply a background fill of light green to these rows.（fine-grained operation：Background Fill）
Within the "Grade" sheet, identify the class with the highest average total score and then merge cells A1 and B1 and label this merged cell with "Best Performing Class".（fine-grained operation：Merge & Unmerge）

Figure 7. Prompt for generating subtasks with “complex computational logic” reasoning challenge.

Generation of Subtasks with Reasoning Challenges

Vague Requirements

System prompt

Role

As a spreadsheet expert, you have the ability to formulate specific questions for given spreadsheets. These questions are utilized to evaluate the large language model's comprehension of spreadsheets.

Constraints

1.First, grasp the content of the provided worksheet, and then formulate questions about "Vague Requirements". Questions about vague requirements involve using synonyms or generic terms to identify statistical objects instead of directly using their real names.
2.Focus the questions on asking without delving into the specifics of how to accomplish it.
3.The principle for generateing questions is that they should only be answerable by referencing the contents of the worksheet.
4.Generate an appropriate number of questions each time.
5.Generate questions from the user's perspective, considering elements such as thought process and tone of speech.
6.Conclude each question with a fine-grained operation, integrating it with the problem to present the result.
7.Specify the referenced object; for instance, when mentioning "Office-related books", it refers to "MS Office Advanced Applications" and "Word Applications."

Fine-grained operations

Here are the fine-grained operations you can choose within the five categories:
A.Value Processing: Calculate,Insert,Delete,Auto Fill,Sort,Copy & Paste,Replace,Hyperlink,Distinction,Filter
B.Worksheet Management:Create Worksheet,Delete Worksheet,Rename Worksheet,Label Color,Page Size,Orientation
C.Format Adjustment:Font Name,Font Color,Font Size,Bold & Italic,Underline,Merge & Unmerge,Alignment,Row Height & Column Width,Background Fill,Numeric Format
D.ChartDesign:Chart Type,Chart Data Source,Chart Caption,Chart Legend,Chart Position,Chart Axis,Data Label,Trendline
E:ContentSummary:Pivot Creation,Summary Function

In-context example

I will give you an example first:
Given a spreadsheet:
Sheet name "Statistical Report":
Orderid	Date	Bookstore Name	Bookid	Bookname	Unit Price	Sales Volume	Subtotal	Purchaser
BTW-08634	2012/10/31	Dingsheng Bookstore	BK-83024	VB Language Programming	38	36	1,368.0	Jiguang Ma
BTW-08633	2012/10/30	Boda Bookstore	BK-83036	Database Principles	37	49	1,813.0	Ruiguo Zhang
BTW-08632	2012/10/29	Boda Bookstore	BK-83032	Information Security Technology	39	20	780.0	Qianghui Huang

Referring to the details provided in the table above, I'll present the following vague requirements questions:
1.In sheet "Statistical Report", determine the month with the highest sales volume of programming language-related books("programming language-related books" are synonyms or generic terms for "VB Language Programming" and "C Language Programming") in 2012, and then calculate its proportion to the total sales volume of that month. Put the result in A1 of the new sheet (fine-grained operation: Calculate)
2. In sheet "Statistical Report", mark the Office-related books("Office-related books" are synonyms or generic terms for "MS Office Advanced Applications" and "Word Applications") in red font color. (fine-grained operation: Font Color)

User instruction

Now it's your turn.
Given a spreadsheet:
Sheet name "Grade":
Student Number	Name	Class	Chinese	Mathematics	English	Biological	Geography	History	Politics	Total Score	Average Score
120305	Hongwei Bao	class 3	91.5	89	94	92	91	86	86	629.5	89.93
120203	Wandi Chen	class 2	93	99	92	86	86	73	92	621	88.71
120206	Beida LI	class 2	100.5	103	104	88	89	78	90	652.5	93.21

GPT response

In the "Grade" sheet, figure out which student performed best in language courses (language courses is a generic term for "Chinese" and "English") and put the student's name and average score of the language courses in A1 and B1 of a new worksheet. (fine-grained operation: Calculate, Create Worksheet)
In the "Grade" sheet, determine the average score for science courses (science courses is a generic term for "Biological", "Geography") in class 3, filter the students with average science courses scores below this average and highlight their name in yellow. (fine-grained operation: Calculate, Filter, Background Fill)

Figure 8. Prompt for generating subtasks with "vague requirements" reasoning challenge.

Generation of Subtasks with Reasoning Challenges
Incoherent Data Format

System prompt
Role
As a spreadsheet expert, you have the ability to formulate specific questions for given spreadsheets. These questions are utilized to evaluate the large language model's comprehension of spreadsheets.

Constraints
1.Start by comprehending the content of the provided form, and then generate questions related to "Incoherent Data Format" where the description involves the same meaning but different units or data formats.
2.Focus the questions on asking without delving into the specifics of how to accomplish it.
3.The principle for generateing questions is that they should only be answerable by referencing the contents of the worksheet.
4.Generate an appropriate number of questions each time.
5.Conclude each question with a fine-grained operation, integrating it with the problem to present the result.
6.Generate questions from the user's perspective, considering elements such as thought process and tone of speech.
7.List equivalent data at the end of the question.

Fine-grained operations
Here are the fine-grained operations you can choose within the five categories:
A.Value Processing:Calculate,Insert,Delete,Auto Fill,Sort,Copy & Paste,Replace,Hyperlink,Distinction,Filter
B.Worksheet Management:Create Worksheet,Delete Worksheet,Rename Worksheet,Label Color,Page Size,Orientation
C.Format Adjustment:Font Name,Font Color,Font Size,Bold & Italic,Underline,Merge & Unmerge,Alignment,Row Height & Column Width,Background Fill,Numeric Format
D.ChartDesign:Chart Type,Chart Data Source,Chart Caption,Chart Legend,Chart Position,Chart Axis,Data Label,Trendline
E:ContentSummary:Pivot Creation,Summary Function

In-context example
I will give you an example first:
Given a spreadsheet:
Sheet name "Employee Information":
|Job number| Name|Sex|Section|Educational background|Telephone|Telephone type|Date of birth|
19	Chen Junjie	Female	Technical department	Master	12383605517	Cell phone	1980/11/16
20	Tong Minru	Male	Technical department	Undergraduate course	12733582889	Cell phone	1985/6/28
21	Gong Junxi	Male	Technical department	Master	90870409	Landline	1969/4/24

Referring to the details provided in the table above, I'll present the following incoherent data format questions:
1. In sheet "Employee Information", italicize the owner's name of 86-12733582889.（The phone numbers 86-12733582889 and 12733582889 are equivalent.）（fine-grained operation：Bold & Italic）
2. In sheet "Employee Information", highlight the names of people born on July 21, 1981 in red.（The date July 21, 1981 and 1981/7/21 are equivalent)（fine-grained operation：Font Color）

User instruction
Now it's your turn.
Given a spreadsheet:
Sheet name "Payroll":
Name	Department	Position	Id Number	Gender	Date Of Birth	Age	EducationalBackground	On-Board Time	Length Of Service (years)	Contract Monthly Wages	Monthly Seniority Wage	Basic Monthly Wage
Baifeng Dao	Management department	General manager	110108196301020119	Male	1963/1/2	115	Doctor	1981/2/1	34	40000	1700	41700
Chunqiu Ding	Administration department	Secretary	110105198903040128	Female	1989/3/4	115	Junior college	2012/3/1	3	4800	60	4860
Xiaocui Ma	Management department	Manager	310108197712121139	Male	1977/12/12	115	Master	2003/7/1	12	12000	360	12360

GPT response
In the current sheet, apply a green background fill to the cell of the employee who joined the company in February 1981. (The date 'Feb-1981' and '1981/2/1' are equivalent.) (fine-grained operation: Background Fill)
Please adjust the Basic Monthly Wage cell for Chunqiu Ding to display it in thousands. (given that $4860 is equivalent to $4.86K.) (fine-grained operation: Numeric Format)

Figure 9. Prompt for generating subtasks with "incoherent data format" reasoning challenge.

Generation of Subtasks with Reasoning Challenges
Information Extraction

System prompt
Role
As a spreadsheet expert, you have the ability to formulate specific questions for given spreadsheets. These questions are utilized to evaluate the large language model's comprehension of spreadsheets.

Constraints
1.First, grasp the content of the provided form and then generate questions related to Information Extraction. The Information Extraction problem involves extracting additional information from the content in the sheet.
2.Focus the questions on asking without delving into the specifics of how to accomplish it.
3.The principle for generateing questions is that they should only be answerable by referencing the contents of the worksheet.
4.Generate an appropriate number of questions each time.
5.Generate questions from the user's perspective, considering elements such as thought process and tone of speech.
6.Conclude each question with a fine-grained operation, integrating it with the problem to present the result.
7.Provide examples of the extracted sources and content. For example, extract the birthday "1986-05-15" from the string "220303198605153610."

Fine-grained operations
Here are the fine-grained operations you can choose within the five categories:
A.Value Processing:Calculate,Insert,Delete,Auto Fill,Sort,Copy & Paste,Replace,Hyperlink,Distinction,Filter
B.Worksheet Management:Create Worksheet,Delete Worksheet,Rename Worksheet,Label Color,Page Size,Orientation
C.Format Adjustment:Font Name,Font Color,Font Size,Bold & Italic,Underline,Merge & Unmerge,Alignment,Row Height & Column Width,Background Fill,Numeric Format
D.ChartDesign:Chart Type,Chart Data Source,Chart Caption,Chart Legend,Chart Position,Chart Axis,Data Label,Trendline
E:ContentSummary:Pivot Creation,Summary Function

In-context example
I will give you an example first:
Given a spreadsheet:
Sheet name "Statistical Report":
Orderid	Date	Bookstore Name	Bookid	Bookname	Unit Price	Sales Volume	Subtotal	Purchaser	PurchaserID
BTW-08634	2012/10/31	Dingsheng Bookstore	BK-83024	VB Language Programming	38	36	1,368.0	Jiguang Ma	211322198509260317
BTW-08633	2012/10/30	Boda Bookstore	BK-83036	Database Principles	37	49	1,813.0	Ruiguo Zhang	211481198401154411
BTW-08632	2012/10/29	Boda Bookstore	BK-83032	Information Security Technology	39	20	780.0	Qianghui Huang	522324197508045617

Referring to the details provided in the table above, I'll present the following information extraction questions:
1.In sheet "Statistical Report", bold the name of the buyer with the surname "Ma" (Extract the last name "Ma" from "Hongyu Ma") . (fine-grained operation: Bold & Italic)
2. In sheet "Statistical Report", extract the buyer's date of birth based on the Purchaser ID, create a new column labeled "Birthday," and put the result (Extract the birthday "1986-05-15" from "220303198605153610") . (fine-grained operation: Font Color)

User instruction
Now it's your turn.
Given a spreadsheet:
Sheet name "Championship":
Outcome	Date	Venue	Surface	Opponent in the final	Score in the final
Runner-up	February 15, 1993	Memphis, Tennessee, USA	Hard (i)	Jim Courier	7-5, 6-7(4-7), 6-7(4-7)
Winner	May 17, 1993	Coral Springs, Florida, USA	Clay	David Wheaton	6-3, 6-4
Runner-up	July 26, 1993	Washington D.C., USA	Hard	Amos Mansdorf	6-7(3-7), 5-7

GPT response
In the provided worksheet, could you filter out and display all matches that have a "Score in the final" that includes a tie-breaker set? (Extract the score "6-7(4-7)" as an example of a tie-breaker set). (fine-grained operation: Filter)
In the workbook, identify all the occasions where the final match was won in straight sets and label these rows with a distinct background color (for example, the final against David Wheaton with the score 6-3, 6-4). (fine-grained operation: Background Fill)

Figure 10. Prompt for generating subtasks with "information extraction" reasoning challenge.

Planner

System prompt
Role
You are a spreadsheet agent and a python expert who can find proper functions to solve complicated spreadsheet-related tasks based on language instructions.

Prerequisites
1. I will show you the headers (along with data type) and row numbers of spreadsheets for your reference.
2. Your partner, "Informer," aids in task completion by providing sheet content represented in {table_representation}, known as "potentially helpful information". This information might be truncated due to token limits, so it's essential to deduce the complete information from what is provided.
3. Please provide step-by-step solutions without explanation.
4. You can use any python library, but when it comes to manipulating spreadsheets, you shoud primarily use the openpyxl and pandas library, which has been already imported as `openpyxl` and `pd`.
5. You should only give one python code snippet at a time. Try not to add comments, and if you must, keep them as concise as possible.
6. The python code snippet should be started with ```python and enclosed with ```.
7. If you want to see the output of a value, you should print it out with `print(x)` instead of `x`.

Response Format Guidance
1. If you think a python code snippet is needed, write using the following output format:
Think: (what you need to solve now and how to solve)
Action: Python
Action Input: (your python code snippet, which should be in accordance with above prerequisites)
2. If you think there is a question to be answered, give your answer using the following format:
Think: (how do you get the answer)
Action: Answer
Action Input: (your answer)
2. If you think task instruction is accomplished, finish with the following format:
Finish: Done!

In context example
...

Instruction
Now it's your turn. This Workbook presents data related to book sales. The workbook is already loaded as `workbook` using openpyxl, you only need to load the sheet(s) you want to use manually. Besides, the workbook will be automatically saved, so you don't need to save it manually.

Sheet state: Sheet "Order Details" has 635 rows (Including the header row) and 8 columns (A(1): "Orderid" (<class 'str'>), B(2): "Date" (<class 'datetime.datetime'>), C(3): "Bookstore Name" (<class 'str'>), D(4): "Bookid" (<class 'str'>), E(5): "Bookname" (<class 'NoneType'>), F(6): "Unit Price" (<class 'int'>), G(7): "Sales Volume" (<class 'int'>), H(8): "Subtotal" (<class 'int'>)). Sheet "Numbering Reference" has 18 rows (Including the header row) and 3 columns (A(1): "Bookid" (<class 'str'>), B(2): "Bookname" (<class 'str'>), C(3): "Unit Price" (<class 'int'>)). Sheet "Statistical Report" has 5 rows (Including the header row) and 2 columns (A(1): "Statistical Item" (<class 'str'>), B(2): "Statistical Value" (<class 'int'>)).

Task instruction: Format the "Order Details" worksheet, adjust "Unit Price" and "Subtotal" to accounting with 2 decimal places and CNY currency symbol format. Fill the "Bookname" column based on Bookid. Identify the bookstore with the largest decrease in sales subtotal in 2012 compared to January to October 2011. Then, determine the best-selling book in that year for this bookstore and highlight its row in red in the "Numbering Reference" worksheet.

Please provide your first step according to the "Response Format Guidance".

Planner response
Think: Apply accounting format with 2 decimal places and CNY currency symbol to the "Unit Price" and "Subtotal" columns in the "Order Details" worksheet.
Action: Python
Action Input:
```python
# Load the "Order Details" sheet
order_details_sheet = workbook["Order Details"]

# Apply accounting format with 2 decimal places and CNY currency symbol to "Unit Price" and "Subtotal" columns
for cell in order_details_sheet["F"][1:] + order_details_sheet["H"][1:]:
    cell.number_format = '"¥"#,##0.00_);[Red]("¥"#,##0.00)'
```
......

Figure 11. A prompt template for the Planner.

Informer

System prompt
You are a table retrieval expert who team up with a spreadsheet agent named "Planner" to accomplish complicated spreadsheet-related tasks. Your partner excel in manipulating spreadsheets. However, some of manipulations require an understanding of specific content of the spreadsheet. Therefore, your role is to retrieve key information from spreadsheets for your partner's reference.

In context example
...

Instruction
Now it's your turn. To make it easier for you to retrieve, all sheets are stored in a SQLite database.
Table schema of "First semester final grades":

```
CREATE TABLE "First semester final grades" (
  "Student Number" INTEGER,
  "Name" TEXT,
  "Class" REAL,
  ...
)
/*
3 example rows:
SELECT * FROM "First semester final grades" LIMIT 3;
{
  "1":{"Student Number":120305,"Name":"Hongwei Bao","Class":null,"Chinese":91.5,"Mathematics":89,"English":94,"Biological":92,"Geography":91,"History":
86,"Politics":86,"Total Score ":null,"Average Score":null},
  "2":{"Student Number":120203,"Name":"Wanchi Chen","Class":null,"Chinese":93.0,"Mathematics":99,"English":92,"Biological":86,"Geography":86,"History":
73,"Politics":92,"Total Score ":null,"Average Score":null},
  ...
}
*/
```

Task instruction: Calculate the total score and average score of each student, and fill in the "Total Score" and "Average Score" columns respectively. Fill in the "Class" column according to the "Student Number". There are three classes: 1, 2, and 3. Fill the cells with scores of no less than 110 points in the three subjects of Chinese, Mathematics, and English in red respectively, and mark the cells with scores above 95 points in the other four subjects in red font.

Previous completed subtasks of the spreadsheet agent (which is presented from your partner's point of view and is more of your partner's thought while finishing the subtasks):
1. Load the "First semester final grades" sheet and calculate the total and average scores for each student.

Given the task instruction and subtasks your partner has completed, predict what your patner will solve next, and determine what content is most needed by your partner. Write a SQLite select statement to achieve this goal. If there is no need to retrieve information from any spreadsheet, please type "pass".
Based on above, responde in the following format:
Think: (how do you think)
Action: (your SQLite statement, e.g., `SELECT * FROM w WHERE age < 18;`, or "pass")

Informer response
Think: According to the task instruction and completed subtasks, the next step for the Planner is more likely be to fill the "Class" column. The rule for filling "Class" depends on the "Student Number" column, so I should provide the agent with the "Student Number" information.
Action: SELECT "Student Number" FROM "First semester final grades" LIMIT 3;

Figure 12. A prompt template for the Informer.