Walk Before You Run: The Importance of Data Exploration for Data Analysis Agents

Yike Yuan
University of Michigan
Ann Arbor, Michigan, USA
yikeyuan@umich.edu

Virum Ranka
University of Michigan
Ann Arbor, Michigan, USA
virum@umich.edu

Tina Lasisi
University of Michigan
Ann Arbor, Michigan, USA
tlasisi@umich.edu

Lin Ma
University of Michigan
Ann Arbor, Michigan, USA
linmacse@umich.edu

ABSTRACT

LLM-based data-analysis tools are increasingly used to help users analyze messy spreadsheets and workbooks, from answering questions over uploaded files to generating code, summaries, and visualizations. Existing tools include chat-with-data interfaces and code-executing analysis agents that inspect uploaded files, run analysis code, and return final outputs. These systems are often evaluated by the correctness of their final downstream answers. However, reliable data analysis also depends on an earlier step: understanding what the dataset contains before solving the requested task. For complex workbooks, this *Data Exploration* step includes identifying the logical tables behind the physical sheets, interpreting column semantics, recovering keys and relationships, and detecting quality issues. In current tools and benchmarks, this step is usually left implicit: a system may produce a plausible final answer without exposing whether it has actually formed a correct, inspectable understanding of the dataset. This creates a gap between downstream task performance and the dataset understanding needed for reliable, human-checkable analysis.

Our key contribution is to identify this overlooked gap, make Data Exploration a first-class evaluation target, and show through downstream experiments that stronger Data Exploration support improves task performance. To evaluate dataset understanding directly, we introduce two benchmark settings: a real multi-sheet workbook benchmark based on a Vitamin D study dataset, and an extension of DSBench with schema-fixed Data Exploration artifacts. In both settings, systems are evaluated by the quality of a structured artifact capturing tables, columns, semantic roles, relationships, and profiling signals. Our results show that strong LLMs and data-analysis agents still miss important logical structure even when they read spreadsheet content. Furthermore, explicit Data Exploration support often improves downstream correctness — suggesting it should be treated as a first-class, inspectable stage in LLM data-analysis workflows and a natural human-in-the-loop checkpoint where domain experts can review and correct the artifact before downstream analysis proceeds.

VLDB Workshop Reference Format:
Yike Yuan, Virum Ranka, Tina Lasisi, and Lin Ma. Walk Before You Run. VLDB 2026 Workshop: DASHSys: Systems for Data-centric Agents with Human-in-the-loop.

1 INTRODUCTION

LLM-based data-analysis tools are increasingly used to help users work with messy spreadsheets and workbooks: a user uploads a file, asks a natural-language question, and expects the system to return reliable answers, calculations, plots, tables, or summaries [7]. Reliable analysis in this setting is challenging because many real tasks begin not from a clean relational database, but from spreadsheet files whose logical structure is only partially explicit. In multi-sheet workbooks, the visible layout of cells may not coincide with the logical data objects needed for analysis: one sheet may contain several tables, a wide matrix may encode a normalized relation, repeated period columns may imply a long fact table, and categorical fields may serve as reusable dimensions.

Existing LLM-based data-analysis tools already make this workflow much easier, including chat-with-data interfaces, spreadsheet agents, and autonomous data-science agents [11, 15, 17, 22]. They can inspect uploaded files, generate or execute analysis code, and return downstream outputs from natural-language requests. However, these workflows usually expose the final analysis result, not a separate representation of what the system believes the dataset contains before analysis begins. This leaves a key part of the analysis process implicit: whether the system has formed a faithful understanding of the dataset before solving the user's task.

We call this prerequisite phase **Data Exploration**: the process of constructing a dataset-grounded understanding before downstream analysis begins. For complex workbooks, Data Exploration includes identifying logical tables behind physical sheets, interpreting column semantics, recovering keys and relationships, and detecting lightweight quality signals. If this pre-analysis understanding is wrong, later reasoning can still look plausible while being grounded in the wrong schema. Conversely, even when a system answers correctly, an end-to-end score alone may not reveal whether it understood the dataset faithfully or arrived at the answer through

Proceedings of the VLDB Endowment. ISSN 2150-8097.

brittle reasoning. This makes failures hard to localize and successes hard to trust.

Figure 1 illustrates the distinction. The common workflow moves directly from an uploaded workbook and user question to downstream execution and final outputs. Our proposed workflow inserts an explicit Data Exploration artifact before downstream analysis: a structured, dataset-grounded representation of the system's understanding of the workbook. The goal is not merely to make the model explain its reasoning, but to externalize what it has inferred about the dataset in a form that can be inspected, corrected, reused, and evaluated independently. This also creates a concrete human-in-the-loop checkpoint: a domain expert who understands what the data represents — but may not write analysis code — can review and correct the artifact before downstream reasoning commits to a potentially flawed dataset interpretation.

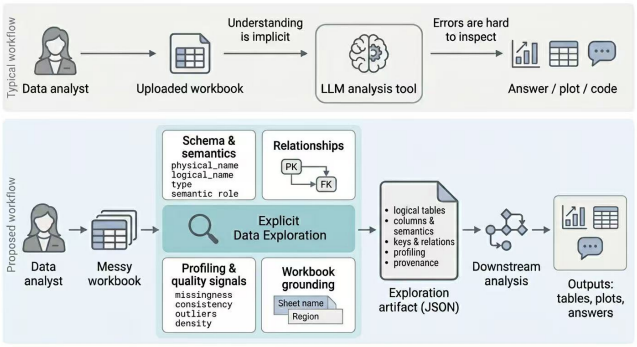

Figure 1: Typical LLM data-analysis workflows versus our proposed workflow with explicit Data Exploration.

A similar gap appears in evaluation. Recent benchmarks have made substantial progress in measuring LLM data-analysis capability. For example, DS-1000 studies data-science code generation with automatic evaluation [10], InfiAgent-DABench evaluates question-driven data analysis over CSVs [6], DataSciBench studies end-to-end data-science workflows [21], and DSBench pushes realism with competition-derived data-analysis tasks [7]. These benchmarks are valuable, and current systems can score impressively on them: the ChatGPT agent, for instance, reports 89.9% pass@1 on DSBench's data-analysis tasks, surpassing the human baseline of 64.1% [13]. Yet these evaluations are still primarily centered on downstream task success. They usually ask whether the system produced the right answer, code, or artifact, rather than whether it first built a correct and inspectable understanding of the dataset. As a result, the Data Exploration gap remains under-tested.

This paper studies that gap by making Data Exploration an explicit object of evaluation. We develop two benchmark settings for evaluating dataset understanding directly. First, we construct a multi-sheet workbook benchmark based on a real-world Vitamin D study dataset acquired from the anthropology department in our institution, where reliable analysis requires understanding repeated measurements, participant identifiers, timepoints, non-data rows, and measurement-quality structure. Second, we extend 12 DSBench data-analysis tasks with schema-fixed Data Exploration artifacts and ground-truth annotations for messy-workbook datasets, while keeping the original downstream questions unchanged. In both settings, systems are evaluated by the quality of a structured artifact that records logical tables, columns, semantic roles, relationships, profiling signals, and source grounding.

We use these benchmarks and downstream experiments to test whether the Data Exploration gap matters in practice. Our benchmark results show that the hardest failures are not surface reading errors, but mistakes in logical schema recovery, implicit entity identification, and relation inference. In downstream tasks, we vary the amount and reliability of Data Exploration support available before task execution. Across the real workbook and selected DSBench tasks, stronger Data Exploration support often improves downstream correctness, although the gains depend on whether the system actually uses and re-grounds the artifact.

The main lesson is that Data Exploration should not be treated as an unobserved byproduct of downstream reasoning. It is a distinct source of both capability and failure: a system can fail because it misunderstands the dataset before analysis begins, and it can improve when that understanding is made explicit. This also has implications for human-in-the-loop data-analysis agents. A structured Data Exploration artifact gives domain experts a concrete object to inspect, correct, and reuse before downstream analysis proceeds. Rather than positioning the human only as a query writer or final-output reviewer, explicit Data Exploration creates an earlier intervention point where human judgment about variables, entities, relationships, and data quality can directly shape the analysis.

Contributions. This paper makes the following contributions:

- **Benchmark settings for explicit Data Exploration evaluation.** We develop a real multi-sheet workbook benchmark based on a Vitamin D study dataset and extend 12 DSBench data-analysis tasks with schema-fixed Data Exploration artifacts, ground-truth annotations, prompts, and an automatic evaluation pipeline for the main structural, semantic, relational, and grounding components of Data Exploration.
- **Evidence that current systems have persistent Data Exploration gaps.** Evaluating representative LLMs and data-analysis agents shows that dataset understanding remains difficult even when systems can read workbook contents. The largest gaps arise in logical schema recovery, implicit entity identification, and relation inference, rather than in surface-level extraction alone.
- **Downstream evidence that Data Exploration support improves analysis.** Through control/middle/treatment experiments on the real workbook and 12 DSBench tasks, we show that explicit or oracle Data Exploration support often improves downstream correctness and helps localize failures — and because the artifact is inspectable before execution begins, it also gives domain experts a natural point to intervene before misunderstandings propagate.

To support reproducibility, we release the code, prompts, schema template, evaluation pipeline, and benchmark assets at https://github.com/coconut0621/walk-before-you-run.

2 BACKGROUND AND MOTIVATION

Data analysis over messy workbooks. We focus on LLM-based data analysis in which a system must turn uploaded spreadsheet files and a natural-language request into reliable downstream outputs, such as answers, tables, plots, cleaned datasets, feature files, or summaries. Unlike question answering over a single clean table, realistic workbook tasks require the system to infer the relevant data objects, valid records, field meanings, and cross-sheet relationships before performing the requested analysis. Examples include answering questions over a financial model, cleaning triplicate measurements from a study workbook, or computing statistics from a multi-sheet spreadsheet whose logical tables are not explicit. These tasks are common in real analysis workflows, and mistakes in this early interpretation step can silently affect later calculations.

LLM development for data analysis: tool use and reasoning. Recent progress in LLM-based data analysis has been driven in part by two capabilities: tool use and inference-time reasoning. With tool use, models can invoke external functions, inspect uploaded files, execute analysis code, and retrieve information during task solving [13, 16, 20]. With inference-time reasoning, models can decompose a request into intermediate steps before producing a final answer [3, 12]. Both capabilities are valuable for downstream execution, but neither by itself guarantees dataset-specific structural understanding. A model can write and run code over the wrong logical table, join fields that do not represent the intended entities, or treat a layout artifact as valid data. Similarly, a fluent reasoning trace is not the same as an externally checkable account of the dataset: recent work shows that chain-of-thought rationales can diverge from a model's actual computational process [2]. For data analysis, these capabilities therefore need to be complemented by an explicit representation of dataset understanding.

Specialized LLM-based tools for data analysis. A growing ecosystem of LLM-based tools targets data analysis specifically. Systems such as PandasAI [17], SheetCopilot [11], and ChatGPT's data-analysis workflow [15] let users load or upload data and interact with it through natural-language requests. In these systems, dataset understanding is typically folded into the process of answering the user's question and executing analysis code, rather than treated as a distinct stage of analysis. DeepAnalyze [22], one of the most recent purpose-built autonomous data science agents, makes planning more explicit before acting on the environment. However, that planning is still directed toward solving the downstream task, rather than producing a separable description of dataset structure that stands on its own as an object of inspection, correction, or scoring. Across this landscape, a consistent pattern emerges: *prior work rarely makes dataset understanding an explicit, standalone, benchmarked object.* This motivates evaluating dataset understanding separately from downstream task execution.

What real data analysts do first. Human data analysts typically treat dataset comprehension as a necessary early part of analysis [8, 9]. Wongsuphasawat et al. [19] interviewed 18 professional data analysts and found that *profiling*—assessing data quality, identifying structural properties, and building a mental model of the dataset—is the one exploration goal shared across *all* analyses. Unlike discovery (finding new insights), which mainly characterizes open-ended analyses, profiling appears across analytical settings, including those with a known downstream question. Data science guides reflect the same view: exploratory analysis is iterative [18]. Even when the main question is already given, analysts still need to investigate what the data contains and whether its quality matches expectations [19].

In real analytical work, a dataset's logical structure is often not identical to its physical layout, especially in spreadsheets and multi-sheet workbooks. The visible organization of cells may only partially reflect the analytical objects that matter downstream, so determining what the data actually contains requires inference beyond surface reading.

We therefore formalize this neglected step as Data Exploration.

3 DATA EXPLORATION

Data Exploration is the pre-analysis phase in which a system turns the physical layout of uploaded workbooks into an explicit logical description that downstream analysis can rely on. In our setting, this description summarizes the workbook as analysis-ready metadata: the logical data objects behind sheets or blocks, their columns and semantic roles, primary and foreign keys, relationships, and lightweight quality/profiling signals. Section 4.1 formalizes these categories as the schema-fixed artifact used in our benchmark. It is not equivalent to reading sheets or extracting cell values. A workbook may be readable at the cell level while still being logically misunderstood: the relevant analytical objects may be distributed across sheets, embedded in repeated blocks, or implicit in layout conventions rather than explicitly named as database tables.

For example, in our real Vitamin D study workbook, reliable downstream analysis requires more than locating measurement cells. The system must recognize participant identifiers, summer and winter timepoints, triplicate skin-reflectance measurements, non-data rows, and measurement-quality structure before it can produce cleaned features. Similarly, in messy DSBench workbooks, the analytical objects needed for a downstream question may be latent in spreadsheet layout rather than explicitly named as database tables.

We operationalize Data Exploration as a *schema-fixed structured artifact*: a single JSON object with a fixed schema that explicitly describes the dataset before downstream reasoning begins. This artifact makes dataset understanding (i) *comparable* across systems, (ii) *auditable* and *correctable* by domain experts before downstream execution begins, and (iii) *automatically scorable* without requiring access to hidden internal states. It also makes the "walk before you run" principle concrete: before solving downstream tasks, the system should already have identified the dataset objects, keys, relations, and profiling that downstream analysis will depend on.

4 DATA EXPLORATION BENCHMARK

Building on datasets from DSBench and a real-world Vitamin D study workbook acquired from the anthropology department in our institution, we directly measure dataset understanding by asking systems to produce a schema-fixed Data Exploration artifact. For the selected DSBench tasks, Figure 2 summarizes the extension: we keep the raw workbooks and original downstream questions

unchanged, but add a Data Exploration prompt, an output template, ground-truth metadata, and an automatic evaluator.

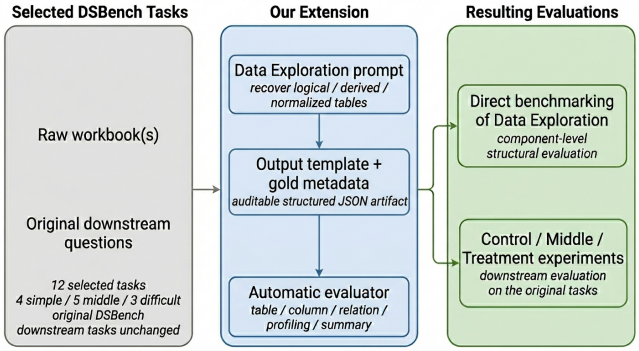

Figure 2: We build on raw workbooks and the original downstream questions, and add an explicit Data Exploration layer.

4.1 Specification-driven Data Exploration task

Formalizing the schema-fixed structured artifact introduced in Section 3, the benchmark task asks the system to produce a conforming JSON output given one or more raw Excel workbooks and a prompt, *before* any downstream analysis begins. The target of inference is the dataset's *logical* structure rather than the superficial workbook layout; the system is explicitly encouraged to recover normalized or derived tables when that better reflects the underlying data organization. The required output includes:

- **Dataset structure:** logical tables, their names, and row counts;
- **Schema and semantics:** columns with both `physical_name` (the workbook-grounded field label) and `logical_name` (the normalized canonical name), together with data types, semantic roles, and dictionary fields;
- **Relational structure:** primary keys, foreign keys, and inter-table relations over those logical tables;
- **Workbook grounding:** explicit table provenance through `source_ref`, which records where the evidence came from in the workbook (e.g., sheet, nearby anchor text, source block, and approximate cell region);
- **Lightweight quantitative characterization:** table/column profiling fields, including density, parse success, type consistency, numeric summaries, date-format consistency when applicable, and robust outlier statistics for numeric columns;
- **Documentation:** a dataset-level structured summary.

This design makes Data Exploration a schema-fixed inference task over *messy workbook blocks*: systems are free to infer normalized logical structure, but must report it in an auditable format.

4.2 Gold metadata format

For each dataset, we maintain a compact human-editable ground-truth JSON in the same overall format as the required output. The gold artifact records the expected logical tables, columns, relations, provenance fields, and profiling-relevant structure needed for evaluation. The gold schema is written at the level of *logical dataset entities*, not spreadsheet tabs: a wide matrix, repeated-period block, or parameter section may correspond to one or more derived relational tables.

4.3 Multi-step evaluation pipeline

After JSON validation, predicted and gold objects are aligned as equivalent logical objects may differ in naming or normalization. The table score s_T estimates whether two tables represent the same logical object from structural, semantic, and provenance cues:

$$s_T = 0.42 s_{\text{col}} + 0.10 s_{\text{row}} + 0.08(s_{\text{sample}} + s_{\text{role}}) + \alpha s_{T,\text{name}} + \beta s_{\text{src}},$$

where $(\alpha, \beta) = (0.08, 0.24)$ with provenance, prioritizing grounding, and $(0.32, 0)$ without it, falling back on names. Tables are maximum-weight one-to-one matched and retained at $s_T \geq 0.58$.

Within matched tables, column identity is scored by

$$s_C = 0.70 s_{C,\text{name}} + 0.10 \mathbb{I}_{\text{dtype}} + 0.10 \mathbb{I}_{\text{role}} + 0.05 \mathbb{I}_{\text{nullable}} + 0.05 \mathbb{I}_{\text{unique}},$$

weighting names as the primary identity signal and the remaining properties as supporting cues. Pairs with $s_C \geq 0.55$ are retained.

Both scores are used only for alignment, and retained pairs receive unit credit. For $x \in \{T, C, R\}$, P_x, R_x, and $F1_x$ denote precision, recall, and F1:

$$P_x = |\mathcal{M}_x|/|\mathcal{P}_x|, \quad R_x = |\mathcal{M}_x|/|\mathcal{G}_x|, \quad F1_x = 2P_xR_x/(P_x + R_x),$$

where $\mathcal{G}_x$, $\mathcal{P}_x$, and $\mathcal{M}_x$ are the gold, predicted, and matched sets; relations are projected through alignment. Unmatched objects count as false positives or negatives. Columns are scored within matched tables and relations are direction-normalized. Exact fields use canonicalized matching; profiling uses tolerance-based numeric scores.

The raw score is the weighted mean of table/column/relation F1, data-type accuracy, and role accuracy (weight 1.0 each); table/column logical-name accuracy (0.5 each); row-count accuracy (0.4); source similarity/all-fields accuracy (0.7/0.3); and profiling/outlier quality (0.3/0.2), omitting inapplicable terms.

$$S = S_{\text{raw}}\big[1 - 0.3(1 - q)\big],$$

where $q \in [0, 1]$ is the mean LLM-judged summary coverage and faithfulness, capping the summary penalty at 30%.

5 BENCHMARK TESTING RESULTS

5.1 Experimental setup

We evaluate Data Exploration on one real multi-sheet Vitamin D workbook and a targeted subset of 12 DSBench data-analysis tasks, selected because their workbooks contain nontrivial multi-sheet and spreadsheet-specific structures (e.g., multiple logical blocks, matrix/repeated layouts, and cross-sheet dependencies). The subset targets this setting rather than representing the full DSBench.

We group them into 4 simple tasks (05, 19, 29, and 38), 5 middle tasks (01, 18, 22, 25, and 27), and 3 difficult tasks (08, 10, and 43), based on the original downstream requirements and workbook reasoning involved: *simple* tasks are mostly multiple-choice questions over relatively direct workbook structure; *middle* tasks require combining messier multi-sheet structures or intermediate workbook logic; and difficult tasks require recovering implicit multi-entity or matrix structures (08 and 10), or progressively repairing linking, dragging, and circular-reference errors (43). We use the same split for both the direct Data Exploration benchmark and the

downstream experiments. The Vitamin D workbook provides a real-world setting with repeated measurements, multiple timepoints, non-data rows, and domain-specific measurement structure.

All systems are given the raw workbook files, the same Data Exploration prompt, and the same schema-fixed output template. They are asked to produce a structured JSON artifact before any downstream task execution. This section evaluates the quality of that artifact directly; the control/middle/treatment downstream experiments are introduced separately in Section 6.

We evaluate Gemini 3.1 Pro [5], Claude Opus 4.6 [1], and GPT 5.4 [14] on the selected benchmark settings. We enable the highest thinking/adaptive capabilities when possible. We also evaluate GPT's agent mode [13] where applicable. Because agent mode has usage limitations and is less suitable for large batch evaluation, we run it only on the difficult DSBench group and on the Vitamin D workbook. All non-agent systems are run in the same direct Data Exploration generation mode: they receive the workbook input and produce the required JSON artifact without access to gold metadata or downstream answers.

Quality is measured with the automatic evaluator described in Section 4. The reported component scores cover table, column, relation, profiling, grounding, and summary quality, together with an overall structural score. For DSBench, we report component scores averaged within each difficulty group. For the Vitamin D workbook, we report component scores on the single real-world workbook instance.

5.2 Results on selected DSBench tasks

Figure 3 reports component-level Data Exploration scores on the 12 selected DSBench tasks, averaged within the simple, middle, and difficult groups. The difficult panel additionally includes GPT's agent mode on the same three difficult tasks.

Key findings. First, Data Exploration remains nontrivial even on benchmark tasks that are ultimately designed for downstream analysis: across systems, the largest and most persistent gaps appear in *logical* understanding and *relation* recovery, rather than in surface-level table or column extraction. Second, system quality is clearly component-dependent. Some systems are relatively strong on structural recovery or summaries, while still leaving substantial errors in latent schema recovery, especially when workbook regions encode multiple entities or implicit objects. Third, as task difficulty increases, Data Exploration weaknesses become more varied and more structurally meaningful, motivating the following case studies, which show how these failures arise in realistic workbook layouts.

Case study: missed decomposition in a multi-entity workbook block. This workbook is the dataset for task 08 in DSBench, where the block *Vehicles for service and hub groupings* encodes several logical entities in one visible spreadsheet region. Because the prompt asks systems to recover *logical / derived / normalized* tables rather than simply transcribe sheets, the gold Data Exploration decomposes this region into `Scenarios`, `Hubs`, and `ScenarioDepotInputs`. This separates reusable scenario definitions and hub entities from the scenario–depot facts that drive downstream allocation.

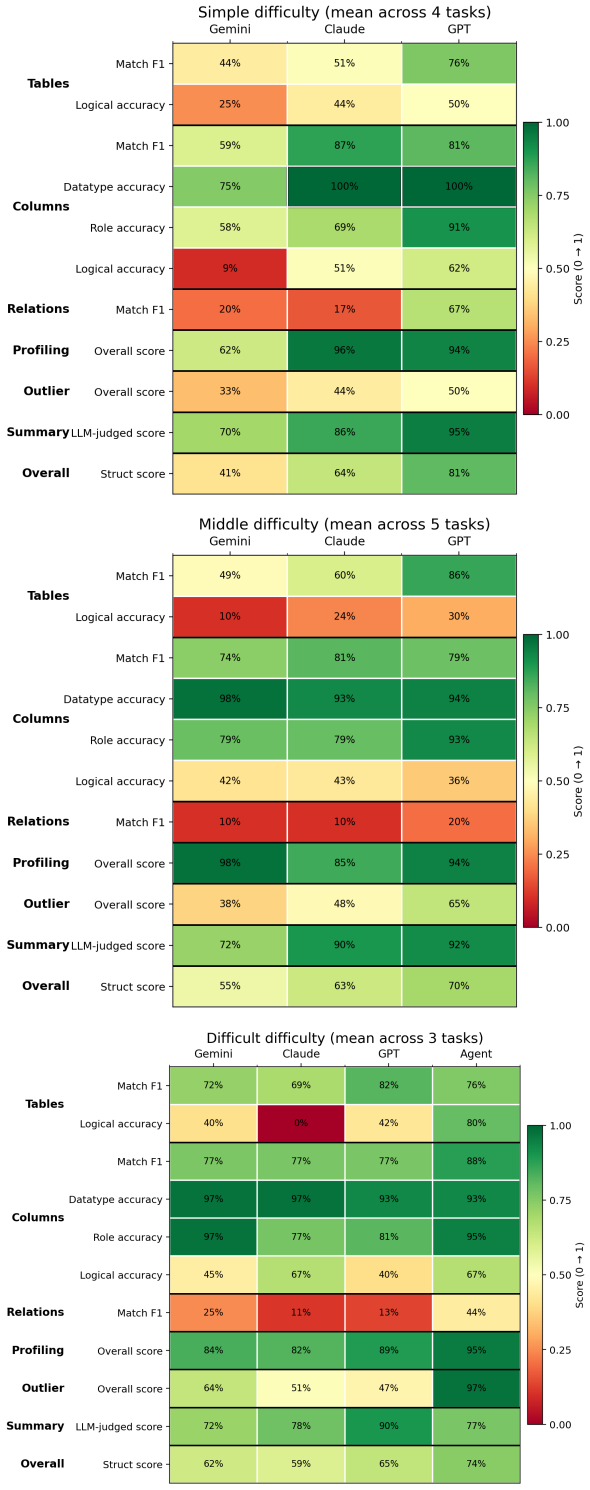

Figure 3: Component-level Data Exploration benchmark scores on selected DSBench datasets, including table/column/relation matching and logical accuracy, profiling, outlier handling, LLM-judged summary quality, and overall structural score. Each panel reports means within one difficulty group.

Claude partially reconstructs this structure. It correctly identifies a normalized scenario–depot fact table and also recovers several explicit blocks, including depot capacity, depot distance, and parameter tables. However, it collapses the scenario–hub region into one table, `depot_scenario_vehicles`, with fields such as `scenario_id`, `depot`, `vehicles_for_service`, and `hub_allocation`. This is not just a naming or formatting difference. In the gold schema, `Scenarios` and `Hubs` are separate logical objects that anchor the intended foreign-key structure, whereas Claude's prediction treats hub membership as attributes inside a single fact-like table.

This mistake explains the relation-level failure: Claude predicts only one relation, while the gold Data Exploration contains five. The model reads many visible workbook blocks correctly, but misses part of the latent relational decomposition needed to represent the workbook as an analytical dataset.

Case study: missing an implicit table hidden behind a matrix block. Task 10 in DSBench uses a football-season workbook with a match-results table, auxiliary team-level sheets, and a bonus matrix over team pairs. The key challenge is that one analytically important object is not presented as a conventional row-wise table. Instead, the `Bonus` sheet encodes pair-level information through a matrix-style layout, where the logical rows should correspond to unordered team pairs and their series-level outcomes.

GPT recovers several row-wise structures. It identifies `Matches` from the `Data` sheet, `Teams` from `Check_Sum`, and the per-team stoptime table from `Q16`, showing that it can read ordinary table-like workbook regions. However, it does not recover the `Bonus` sheet as a separate pair-level table. Instead, it predicts `Team_Primes` while missing the more consequential table linking two teams with bonus and winner information.

This omission affects both table and relation scoring. In the gold Data Exploration, the pair-level table should link each bonus record to the two participating teams and to the winner field. Because GPT misses the table, these relations are absent as well. The error is therefore not a local column mistake or a superficial naming mismatch; it is a failure to infer an implicit logical object from a non-row-wise workbook layout. This case complements the previous one: even when visible tables are recovered accurately, Data Exploration can still fail if the system does not reconstruct latent objects encoded by spreadsheet-specific structure.

5.3 Real-world Vitamin D workbook

Figure 4 reports component-level Data Exploration scores for the real Vitamin D workbook. Low relation-recovery and lightweight quantitative-characterization scores show that Data Exploration difficulty is not unique to the DSBench extension: consequential failures persist in real analytical workbooks. We next test whether stronger Data Exploration support is associated with better downstream performance.

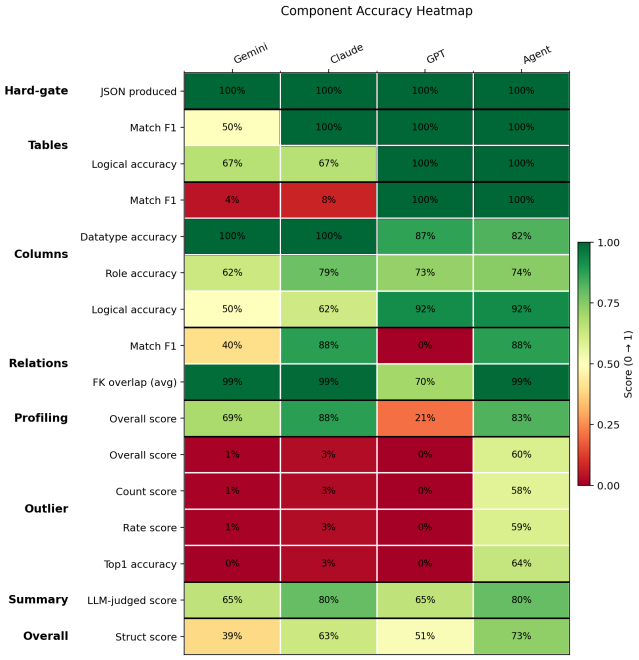

Figure 4: Component-level Data Exploration benchmark scores on the real-world workbook. Scores include table/column/relation matching and logical accuracy, profiling, outlier handling, LLM-judged summary quality, and overall structural score.

6 DOES DATA EXPLORATION IMPROVE DOWNSTREAM ANALYSIS?

6.1 Experimental setup

We evaluate whether explicit Data Exploration support improves downstream data-analysis performance on the same two families of workbook settings used in Section 5: the real Vitamin D study workbook and the 12 selected DSBench data-analysis tasks. Unlike Section 5, which scores the Data Exploration artifact itself, this section evaluates final downstream task correctness.

For each dataset and downstream task, we compare three experimental conditions summarized in Table 1. All conditions share the same downstream questions and scoring; they differ only in the Data Exploration support available before answering. The oracle artifact used in Treatment contains only pre-analysis metadata, not downstream answers, task-specific derivations, or solution hints. The goal is not to hold token usage constant across conditions, but to test whether inserting an explicit Data Exploration step or artifact improves later analysis.

Condition	What the system gets before downstream questions
Control	Raw files only
Middle	Raw files + the system's own Data Exploration JSON (generated from the files)
Treatment	Raw files + oracle Data Exploration JSON (ground truth)

Table 1: Data Exploration as an intervention: three conditions used in our downstream experiments.

For the Vitamin D workbook, we solicit real-world analysis questions from its provider. The downstream task requires systems to consolidate triplicate skin-reflectance measurements into cleaned

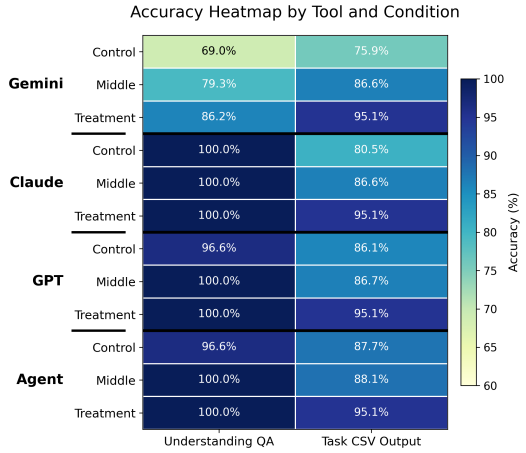

Figure 5: Accuracy on the real Vitamin D workbook experiments. Rows show tools and prompting conditions; the left column reports accuracy on dataset-understanding questions, and the right column reports accuracy on the downstream feature-table task (outputs evaluated as CSV files).

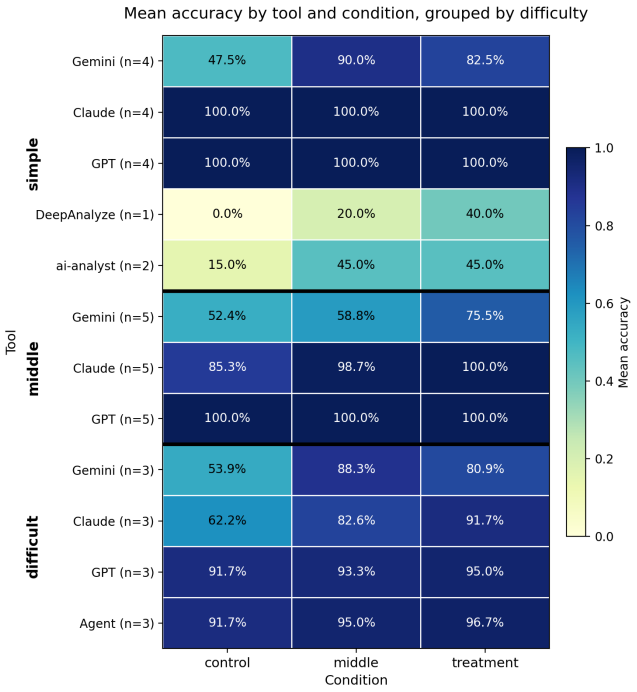

Figure 6: Mean accuracy of different tools in different conditions, grouped by difficulty.

participant–timepoint feature tables for *summer* and *winter*. We report accuracy on objective dataset-understanding questions and generated feature-table accuracy against the expert reference.

For DSBench, we keep the original downstream questions unchanged and report answer accuracy under the three conditions, using the same 4/5/3 difficulty split defined in Section 5.1. The Data Exploration artifact is auxiliary metadata for downstream reasoning, not a disguised answer key.

We evaluate the same main model versions as in the direct Data Exploration benchmark: Gemini 3.1 Pro [5], Claude Opus 4.6 [1], and GPT 5.4 [14]. We also evaluate GPT's agent mode [13] on the difficult DSBench group and on the Vitamin D workbook. For the simple DSBench group, we additionally include DeepAnalyze and ai-analyst [4, 22]; due to tool limitations, these systems are not tested on the middle or difficult groups.

6.2 Experimental results

Figures 5 and 6 summarize the downstream results on the real workbook workflow and the selected DSBench tasks. Across both settings, stronger Data Exploration support is generally associated with better downstream correctness.

Real-world workbook task. On the real Vitamin D workbook, performance increases from CONTROL to MIDDLE and then to TREATMENT, especially on the downstream feature-table output. The gains are most pronounced for Gemini, while others are already strong on the understanding questions and still benefit from stronger Data Exploration support on the downstream task. Even when a system can answer many schema-level questions correctly, explicit Data Exploration support still helps translate that understanding into more reliable execution on the real analysis workflow.

Selected DSBench tasks. On DSBench, we observe the same qualitative pattern but with substantially larger variation across task difficulty. Data Exploration support is most helpful on the middle and difficult groups, where success depends less on surface extraction and more on recovering latent structure, relations, and intermediate spreadsheet logic before answering the final questions. Although TREATMENT is usually the strongest condition, we still observe occasional cases where MIDDLE slightly outperforms TREATMENT, suggesting that externally provided Data Exploration information is not always fully leveraged unless the model first constructs the representation itself.

Case study: a difficult financial-model task. This task is built on a multi-sheet financial workbook with deliberate linking, dragging, and circular-reference errors. The instructions explicitly require the solver to progressively fix each error identified before answering the final downstream questions, so later answers depend on reconstructing cross-sheet dependencies rather than checking isolated formulas. GPT improves from **15/20** in CONTROL to **16/20** in MIDDLE and **17/20** in TREATMENT.

The clearest CONTROL→MIDDLE gain appears on Q18 and Q20, the final downstream questions. In CONTROL, GPT treats them as uncertain end-of-chain questions and selects incorrect options. In MIDDLE, after first reconstructing the workbook's dependency structure, GPT recognizes that the debt schedule depends on beginning cash, operating and investing cash flow, and debt repayment logic, and that debt issuance/repayment should not itself enter "cash available for debt repayment" because that would create circularity. This changes Q18 to b and Q20 to a.

TREATMENT adds a more localized benefit on Q1, where GPT uses the oracle artifact to identify Debt row 6—the "Cash flow available

for debt repayment" line, which links to row 26—as the source of circularity. GPT still misses Q3 and Q6, which shows that Data Exploration support improves dependency reasoning and error localization, but does not eliminate all formula-level errors.

The same task also separates Data Exploration availability from Data Exploration use. On Q8, GPT thinking mode answers b in both Middle and Treatment, defaulting to the generic accounting rule that EBIT should explicitly deduct D&A. GPT agent mode instead answers the gold option a, recognizing that in this workbook depreciation and amortization are already embedded in operating costs. Thus the remaining challenge is not only recovering workbook structure, but using it for workbook-specific semantic judgments.

Case study: a depot-allocation task. This task asks the model to simulate MVC's depot-allocation procedure across scenarios. It requires combining several workbook structures: scenarios alter hub partitions, hubs contain different depots, depots have service capacities, and overflow vehicles are greedily reassigned to the shortest feasible trip. Correct answers thus require reconstructing the latent procedure encoded across these blocks.

Claude improves monotonically, from **6/9** in Control to **7/9** in Middle and **9/9** in Treatment. The main Control→Middle gain is on Q30, which asks for Hub 1's total transport cost in Scenario 1b. In Control, Claude recognizes that Hub 1 expands to depots A–D but does not complete the reassignment chain and selects H. In Middle, after reconstructing the scenario and hub structure, it follows the correct overflow pattern: $D \to A$, $C \to A$, and $B \to A$; 56 vehicles are penalized, yielding the correct option I. The gain is therefore not a simple arithmetic correction, but comes from recovering how scenario membership, depot capacity, distance ordering, and greedy reassignment interact.

The Middle→Treatment gain is different. On Q27 and Q33, Claude in Middle computes the correct quantities but maps them to the wrong multiple-choice letters. With the oracle Data Exploration artifact, it retains the computations but grounds the final selections correctly. This case thus separates two benefits of stronger Data Exploration support: Middle improves procedural decomposition, while Treatment reduces residual answer-grounding errors.

The same task provides a counterexample: Gemini rises from **6/9** in Control to **9/9** in Middle, then falls to **7/9** in Treatment. The reversal is on Q31 and Q32 for Hub 2 in Scenario 1b. In Middle, Gemini correctly applies $I \to H$, $E \to F$, then $I \to F$ with F's remaining capacity, leaving 20 vehicles unserviced and yielding the correct Q31 revenue of $5,202,000. In Treatment, it instead uses $I \to H$, $E \to G$, and $I \to F$, producing the error and carrying it into Q32.

Receiving a Data Exploration artifact does not ensure that every downstream step is re-grounded in the workbook. Producing it in Middle may force closer inspection of the table and reassignment rule, whereas Treatment can fail under incomplete procedural verification. The effect is not uniformly negative—Treatment still repairs Q29 by correcting a local transport-cost discrepancy—but shows that Data Exploration support must be actively used.

7 DISCUSSION AND FUTURE WORK

Our evidence suggests Data Exploration is both practically important and currently under-evaluated, motivating it as an additional benchmark axis alongside downstream question answering. Future work includes expanding dataset diversity, more directly quantifying how specific Data Exploration components relate to downstream gains and failure modes, and exploring how Data Exploration artifacts can be cached, edited, and reused as persistent structured memories. Although an explicit Data Exploration step adds upfront cost, that cost can be amortized: once a dataset-specific artifact has been produced or verified, multiple downstream tasks can reuse the same structured context, potentially reducing repeated schema re-inference, token cost, and hallucinated joins.

Broader implications: from pipelines to human-in-the-loop analysis. Beyond benchmarking, Data Exploration surfaces a deeper point about the nature of data analysis itself. Most data analysis tasks do not have a single correct answer or a single correct solution path—the space of possible questions to ask, and the interpretations to draw, depends critically on what the data actually contains. If we frame AI-assisted analysis as a pipeline that begins from a pre-specified question, we leave little room for human agency: the human poses a query, the system executes it. But if a Data Exploration artifact is produced and made available before final downstream execution, the interaction changes fundamentally. Humans—including domain experts who understand what the data represents but may lack programming fluency—can engage at the stage where their knowledge matters most: deciding what to look for, identifying which variables are meaningful, flagging quality issues that a schema-level summary makes visible. Data analysis then shifts from a question-answering pipeline toward a collaborative process in which human creativity and domain expertise are not bypassed but actively engaged. In this sense, Data Exploration is not only a prerequisite for accurate downstream computation; it is also an auditable entry point through which domain experts can re-enter the analytical loop.

8 CONCLUSION

Reliable LLM data-analysis tools should "walk before they run": before answering downstream questions, they should form a faithful, dataset-grounded understanding of the workbook. This paper makes that pre-analysis step explicit as Data Exploration. We develop benchmark settings for evaluating Data Exploration directly, including a real multi-sheet Vitamin D workbook and an extension of selected DSBench tasks with schema-fixed Data Exploration artifacts and automatic component-level evaluation. Our results show that current LLMs and data-analysis agents still have measurable Data Exploration gaps, particularly in recovering implicit logical structure and relationships from messy workbooks. We further show through downstream experiments that stronger Data Exploration support often improves task correctness, while also revealing that models must actively use and re-ground the artifact for the benefit to appear. By making Data Exploration explicit, inspectable, and evaluable, we aim to move AI-assisted data analysis from a closed query-to-answer pipeline toward a workflow in which both model reliability and human oversight can improve.

ACKNOWLEDGMENTS

This work was supported (in part) by the Google ML and Systems Junior Faculty Award, the Google JAX AI Stack Research Award, and the NVIDIA Academic Grant Program Award.

REFERENCES

[1] Anthropic. 2026. Introducing Claude Opus 4.6. https://www.anthropic.com/news/claude-opus-4-6. Accessed 2026-04-28.

[2] Fazl Barez, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, Nicolas Collignon, Clement Neo, Isabelle Lee, Alasdair Paren, Adel Bibi, Robert Trager, Damiano Fornasiere, John Yan, Yanai Elazar, and Yoshua Bengio. 2025. Chain-of-Thought Is Not Explainability. ICLR 2025 Workshop on Foundation Models in the Wild. https://aigi.ox.ac.uk/publications/chain-of-thought-is-not-explainability/

[3] DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 https://arxiv.org/abs/2501.12948

[4] e2b-dev. 2026. AI Analyst by E2B. https://github.com/e2b-dev/ai-analyst. GitHub repository, accessed 2026-03-03.

[5] Google DeepMind. 2026. Gemini 3.1 Pro: A smarter model for your most complex tasks. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/. Accessed 2026-04-28.

[6] Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. InfiAgent-DABench: Evaluating Agents on Data Analysis Tasks. arXiv:2401.05507 [cs.CL] https://arxiv.org/abs/2401.05507

[7] Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2024. DSBench: How Far Are Data Science Agents to Becoming Data Science Experts? arXiv:2409.07703 [cs.AI] https://arxiv.org/abs/2409.07703

[8] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. 2011. Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data. *Information Visualization* 10, 4 (2011), 271–288. https://journals.sagepub.com/doi/10.1177/1473871611415994

[9] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2917–2926. https://ieeexplore.ieee.org/document/6327298

[10] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. *arXiv preprint arXiv:2211.11501* (2022). arXiv:2211.11501 https://arxiv.org/abs/2211.11501

[11] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023. SheetCopilot: Bringing Software Productivity to the Next Level through Large Language Models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. https://proceedings.neurips.cc/paper_files/paper/2023/hash/0ff30c4bf31db0119a6219e0d250e037-Abstract-Conference.html

[12] OpenAI. 2024. OpenAI o1 System Card. arXiv:2412.16720 https://arxiv.org/abs/2412.16720

[13] OpenAI. 2025. Introducing ChatGPT agent. https://openai.com/index/introducing-chatgpt-agent/. Accessed 2026-04-07.

[14] OpenAI. 2026. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/. Accessed 2026-04-28.

[15] OpenAI Help Center. 2025. Data analysis with ChatGPT. https://help.openai.com/en/articles/8437071-data-analysis-with-chatgpt. Accessed 2026-03-03.

[16] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*. arXiv:2302.04761 https://arxiv.org/abs/2302.04761

[17] Sinaptik-AI. 2026. PandasAI: Chat with your database or your datalake. https://github.com/sinaptik-ai/pandas-ai. GitHub repository, accessed 2026-03-03.

[18] Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data* (2nd ed.). O'Reilly Media. https://r4ds.hadley.nz/

[19] Kanit Wongsuphasawat, Yang Liu, and Jeffrey Heer. 2019. Goals, Process, and Challenges of Exploratory Data Analysis: An Interview Study. arXiv:1911.00568 [cs.HC] https://arxiv.org/abs/1911.00568

[20] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. arXiv:2210.03629 https://arxiv.org/abs/2210.03629

[21] Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. 2025. DataSciBench: An LLM Agent Benchmark for Data Science. *arXiv preprint arXiv:2502.13897* (2025). arXiv:2502.13897 https://arxiv.org/abs/2502.13897

[22] Shaolei Zhang, Ju Fan, Meihao Fan, Guoliang Li, and Xiaoyong Du. 2025. DeepAnalyze: Agentic Large Language Models for Autonomous Data Science. arXiv:2510.16872 [cs.AI] https://arxiv.org/abs/2510.16872