

# UniDataBench: Evaluating Data Analytics Agents Across Structured and Unstructured Data

Han Weng<sup>1\*</sup>, Zhou Liu<sup>1,2\*</sup>, Yuanfeng Song<sup>1†</sup>,  
Xiaoming Yin<sup>1†</sup>, Xing Chen<sup>1</sup>, Wentao Zhang<sup>2</sup>

<sup>1</sup>ByteDance, China

<sup>2</sup>Peking University, China

## Abstract

In real-world business environments, data is stored in a variety of sources, including structured relational databases, semi-structured databases, and unstructured files. The ability to extract reasonable insights across these diverse sources is integral to data-driven decision-making. Existing benchmarks, however, are limited in assessing agents' capabilities across these diverse data types. To address this gap, we introduce **UniDataBench**, a multi-source benchmark designed to evaluate the performance of data analytics agents in handling diverse data sources. Specifically, UniDataBench is constructed based on real-life industry analysis reports, employing a pipeline to synthesize data that aligns with authentic analytical trends. It encompasses diverse datasets spanning relational databases, CSV files, and NoSQL stores to reflect real-world business settings, and provides a unified framework for evaluating how effectively agents can explore multiple data formats, extract insights, and generate meaningful summaries and recommendations. Based on UniDataBench, we propose a novel LLM-based agent named **ReActInsight**, an autonomous agent that performs end-to-end analysis over diverse data sources by automatically discovering cross-source linkages, decomposing goals, and generating robust, self-correcting code to extract actionable insights. Our benchmark and agent together provide a framework for facilitating the development of data analytics agents in real-world applications.

## 1 Introduction

The ultimate goal of modern enterprise analytics is to support effective decision-making by distilling vast amounts of raw data into insights (Jahns, 2013; Black, 2023; Ghazal et al., 2013; Sharma et al., 2014). This process typically begins with a high-level business objective and proceeds through

an iterative exploration of data. In contemporary business intelligence, however, the most valuable insights are seldom found within a single, well-structured dataset. Instead, they emerge from the complex synthesis of a fragmented information landscape with diverse formats (Egg et al., 2025; Hu et al., 2024).

The rapid development of large language models (LLMs) has led to the emergence of data science agents capable of automating the deconstruction of complex problems and exploring insights within data by coordinating specialized tools (Ma et al., 2023; Li et al., 2023; Zhang et al., 2024a; Sahu et al., 2025; Hong et al., 2025; Bai et al., 2025; Pérez et al., 2025). Despite extensive research in data science, current benchmarks fall short in measuring data agents' ability to perform comprehensive analysis across diverse data formats (e.g., databases, NoSQL, and text files) rather than just CSV files. Existing data science benchmarks such as Text2Analysis (He et al., 2024), DABench (Hu et al., 2024), and DASTep (Egg et al., 2025) focus on data analysis for specific problems. Furthermore, current multi-step data analysis benchmarks that start from high-level goals only focus on structured tabular data within a single enterprise management domain, with human-designed analytical conclusions further hindering their effectiveness in evaluating agents' performance in real-world scenarios (Sahu et al., 2025).

To address these challenges, we propose UniDataBench, a comprehensive data analysis benchmark tailored for assessing the performance of data agents across various data formats. This benchmark includes 100 datasets covering five business analysis scenarios, featuring diverse and varied input data file formats and difficulties. The tasks require agents to perform multi-step problem decomposition starting from a high-level goal, and to connect, query, and understand information from structured, semi-structured databases, and unstruc-

\*Equal contribution.

†Corresponding authors.

tured documents to generate insights. To ensure the value of the analytical problems and insights in the benchmark, we enlisted expert annotators to extract questions and corresponding relevant insights from real-world enterprise business data analysis reports, such as tracking product usage metrics, analyzing event-driven sales fluctuations, and monitoring changes in user feedback sentiment. Subsequently, we conducted data synthesis and visualization validation based on LLMs and manual inspection to ensure that the insights were indeed discoverable.

Building upon our benchmark, we introduce ReActInsight, an innovative autonomous agent engineered for end-to-end analysis across diverse data sources. ReActInsight initiates its workflow with a novel multi-source data exploration and linkage discovery phase. It automatically constructs a unified metadata representation (i.e., *MetaGraph*) for all available data, then performs similarity analysis to discover potential join keys, formulating them into an actionable Join-Hint. Guided by this hint, the agent employs a hierarchical planning mechanism to decompose a high-level user goal into a series of answerable sub-questions. For each sub-question, it generates executable code, leveraging a robust self-correction and debugging module to ensure reliability and adaptive visualization techniques to uncover underlying patterns. Finally, ReActInsight synthesizes the results from all steps into a coherent summary, presenting the user with key insights and the detailed analytical process.

In summary, our contributions are as follows:

- We propose UniDataBench, a benchmark designed to evaluate the capability of agents in conducting data analysis starting from high-level goal under multiple data formats.
- We propose ReActInsight, an agent that extracts insights from multiple data formats through multi-source exploration, hierarchical planning and adaptive visualization. It achieves superior performance compared to other data analysis agents.
- We conduct extensive experiments on our benchmark, demonstrating the challenges of these tasks and validating the effectiveness of our proposed agent in tackling them.

## 2 Related Work

### 2.1 Data Science Benchmarks

The evaluation of LLMs in data science has produced a rich landscape of benchmarks, initially

focused on generating code for specific, isolated steps of the analytical workflow (Yu et al., 2018; Lai et al., 2023; Zhang et al., 2024b; Song et al., 2024; Qin et al., 2025; Lu et al., 2025).

For example, benchmarks like DS-1000 (Lai et al., 2023) test the fine-grained code generation abilities of models on discrete data manipulation tasks using popular libraries. As LLMs evolved and were augmented by agentic frameworks, it became clear that evaluating isolated skills is insufficient for gauging performance in real-world applications. This led to a new generation of benchmarks designed to evaluate an agent’s ability to solve complex, end-to-end tasks by interacting with an execution environment. These benchmarks target a broader spectrum of data science activities (Huang et al., 2024; Jing et al., 2025; Hu et al., 2024; Lei et al., 2025; Egg et al., 2025; Sahu et al., 2025). For example, InsightBench (Sahu et al., 2025) assesses the entire process of planning, coding, and insight generation, pushing the evaluation towards higher-level, multi-step reasoning.

Despite these advances, a critical limitation pervades the current landscape: these benchmarks are designed to assess capabilities within siloed data environments. Each task typically operates on a single, self-contained data source. This leaves a crucial gap in evaluating an agent’s ability to perform analysis that requires synthesizing information from multiple data formats, which is a common requirement in modern business intelligence.

### 2.2 Data Analytics Agents

The proliferation of LLMs has promoted the development of numerous agents designed to automate complex tasks. In the domain of data science, a variety of specialized agents have emerged to streamline analytical workflows (Li et al., 2023; Yao et al., 2023; Majumder et al., 2025; Ma et al., 2023; Zhang et al., 2024a; Sahu et al., 2025; Hong et al., 2025; Lu et al., 2026). For instance, SheetCopilot (Li et al., 2023) is a spreadsheet agent designed to generate and execute a plan of operations on tabular data to achieve a desired outcome. Similarly, Data Copilot (Zhang et al., 2024a) can automatically perform data processing, prediction, and visualization based on questions. Other agents focus on the high-level goal of insight discovery. For example, AgentPoirot (Sahu et al., 2025) performs end-to-end data analysis to extract descriptive, diagnostic, predictive, and prescriptive insights. Concurrently, systems like Data Interpreter (Hong et al.,

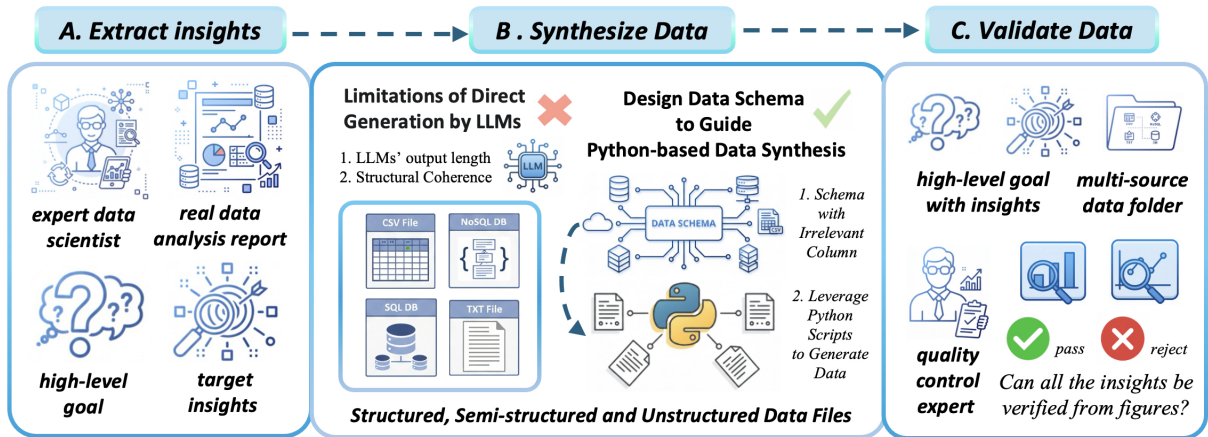


Figure 1: The three-stage generation pipeline for UniDataBench: (A) Extracting insights from real reports, (B) Designing a schema to guide Python-based data synthesis, and (C) Human validation via visualization.

2025) enhance problem-solving capabilities in data science through advanced methods like dynamic planning and logical consistency checking.

While some prominent agents, such as OpenAI’s Code Interpreter (OpenAI, 2024), Pandas Agent (LangChain, 2024) and the ReAct-style Agent of DABstep (Egg et al., 2025), demonstrate the ability to process multiple data formats and answer questions about them, their operational paradigm is typically limited to addressing these formats in isolation. Our agent is specifically designed to focus on exploring data and discovering insights that spanning structured and unstructured data sources, thereby enriching the current landscape of agents in the field of data analytics by addressing the need for integrated, multi-source reasoning.

### 3 The proposed UniDataBench

#### 3.1 Overview

To address a key limitation of existing high-level data analysis benchmarks, which are constrained to a structured data format (e.g., CSV or MySQL), we propose UniDataBench. Our benchmark is built around insights collected directly from real-world enterprise data analysis reports. Deriving these insights requires synthesizing information from diverse data formats, including structured (CSV, SQLite), semi-structured (NoSQL), and unstructured (TXT). In this section, we introduce the dataset construction pipeline illustrated in Figure 1.

##### 3.1.1 Extract Insights from Real Reports

Unlike insights that are synthetically generated by LLMs or artificially constructed from human-implanted trends (Zhang et al., 2024b; He et al., 2024; Hu et al., 2024; Sahu et al., 2025), the in-

sights in UniDataBench stem from real-world business reports analyzing areas like product metrics, sales figures, and market competition. Specifically, experts first extract core business conclusions from raw reports containing contextual background and data information. These findings are then abstracted into high-level analytical objectives; for example, distilling “Sales decline in the Northern Region due to logistics delays” into a broader goal such as “Analyzing the root causes of sales fluctuations.” To ensure that the mapping logic is aligned with practical business logic, the entire process undergoes cross-validation by an independent group of senior data scientists. Details regarding experts are provided in Appendix A.6.

**Types of Insights** To facilitate a structured extraction, we adopt a taxonomy adapted from the formulations in (Ding et al., 2019), identifying four distinct insight types refined to suit the objectives of high-level multi-source analysis: (i) *Trend*: identifying patterns or directional movements in data over time or sequence. (ii) *Comparison*: segmenting data to compare the performance or behavior of different groups. (iii) *Extreme Value*: pinpointing maximum or minimum values to identify outliers or key performers. (iv) *Attribution*: uncovering causal or correlational relationships between variables to explain why something is happening. Based on these categories, the raw insights are then systematically filtered, combined, and mapped to high-level analytical goals.

##### 3.1.2 Data Synthesis and Validation

While real-world insights are obtained through the preceding step, the corresponding raw data is often inaccessible. This is because such underlying data

Dataset	Analysis Source	Data Formats	Task Size	File Count	Avg. Rows
<i>Specific Data Analysis - A specific answer provided for Question: What are the applicable fee IDs for Rafa_AI in 2023?</i>					
DS-1000 (Lai et al., 2023)	StackOverflow Extracted	data.frame	1000	-	-
DSEval (Zhang et al., 2024b)	LLM Generation & Human	csv	825	44	23630
Text2Analysis (He et al., 2024)	LLM Generation & Human	csv	2249	347	-
DABench (Hu et al., 2024)	LLM Generation & Human	csv	257	52	2024
DABstep (Egg et al., 2025)	Real-world Data Analysis	3 types	450	7	3121
<i>High Level Data Analysis - Insights are extracted for Goal: Find the discrepancy and imbalance in incidents assigned</i>					
InsightBench (Sahu et al., 2025)	LLM Generation & Human	csv	100	112	545
<b>UniDataBench (Ours)</b>	Real-world Data Insights	4 types	100	223	17945

Table 1: Comparison of UniDataBench with other benchmarks. UniDataBench is distinguished by its emphasis on high-level data analysis, which extends beyond the limitations of CSV-only data to support multiple file formats.

are usually sensitive, and current business reports only release conclusive insights. To address this issue, this step focuses on automatically constructing underlying data that aligns with the trends of these insights.

As illustrated in Figure 1, direct synthesis of large-scale, complex data by LLMs is constrained by their inherent length limitations and the challenge of maintaining long-range structural coherence (Tang et al., 2024). To address this, we adopt a three-step approach for data generation:

First, leveraging the extracted high-level questions and corresponding insights, we construct comprehensive instruction prompts for LLMs by incorporating targeted scenario-specific context. For question-insight pairs requiring domain knowledge, we manually curate this context on a per-case basis to ensure logical rigor. This background information includes domain-specific logic, such as calculation formulas (e.g.,  $Lead\ Rate = CTR \times CVR$ ). These prompts guide LLMs to design multi-source data schemas, whose outputs then undergo manual inspection to ensure coherence and correctness. Second, we proactively introduce distractor columns that are irrelevant to the analytical objectives but related to the scenario to increase analytical complexity and better reflect real-world scenarios.

Finally, we instruct the LLM to generate a Python script for data synthesis with specified information. The generation of unstructured data follows a separate process. We use the required analytical information as a core basis, prompting an LLM to expand upon it and generate a simulated document. A detailed example of data synthesis and validation is provided in Appendix A.3.

During the data validation process, we utilized Python scripts to generate visualizations related to insights. These charts are then subjected to

Insights (Total: 439)		Files (Total: 223)	
Category	Count (%)	Type	Count (%)
Trend	112 (25.5%)	CSV	110 (49.3%)
Extreme	126 (28.7%)	DB	49 (22.0%)
Comparison	110 (25.1%)	NoSQL	32 (14.3%)
Attribution	91 (20.7%)	TXT	32 (14.3%)

Table 2: Insights and File Types in UniDataBench.

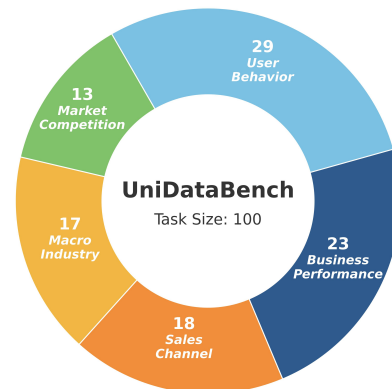


Figure 2: Domain Distribution in UniDataBench

a human-in-the-loop review by data analysis experts. Their task is to confirm that every related insight is clearly discoverable from the visual evidence. To ensure transparency in the benchmark construction, we provide a detailed breakdown of the human-labeled, synthetic, and expert-validated components in Appendix A.5.

### 3.2 Statistics of UniDataBench

UniDataBench is a comprehensive benchmark for high-level data analysis. It features 100 analytical tasks designed to elicit 439 target insights from a corpus of 223 data files. The tasks are stratified into three difficulty levels based on the number of data formats required for analysis: 25 easy (single format), 50 medium (two formats), and 25 hard

(three or four formats). As illustrated in Figure 2, the tasks span five core business domains, while Table 2 provides a detailed breakdown of the insight categories and file type distributions. More comprehensive statistics are provided in Appendix A.4.

## 4 The Proposed ReActInsight

### 4.1 Motivation and Overview

Despite their proficiency in executing siloed operations on single data modalities, existing agents lack the capability to orchestrate complex analytical workflows and synthesize coherent insights from diverse data formats. To alleviate this limitation, we introduce **ReActInsight**, a novel autonomous agent designed for end-to-end data analysis across both structured and unstructured data sources. ReActInsight moves beyond simple reactive execution by implementing a sophisticated, multi-layered reasoning process inspired by the ReAct framework.

As shown in Figure 3, its workflow begins with an efficient **multi-source data exploration** phase to gain a preliminary understanding of the vast and varied data landscape. Following this, ReActInsight employs a **hierarchical planning mechanism**, decomposing the primary analytical goal into a series of manageable sub-goals and specific, answerable sub-questions. For each sub-question, the agent autonomously generates executable code, leveraging **diverse and adaptive visualization** techniques to uncover underlying patterns. Crucially, it is equipped with a robust **self-correction and debugging** module to ensure reliability. Finally, ReActInsight culminates its analysis by presenting the user with the discovered insights, a conclusive summary, and the detailed analytical steps that led to these findings.

### 4.2 Architecture of ReActInsight

The architecture of ReActInsight is designed as a multi-stage, iterative pipeline that enables comprehensive data analysis from initial exploration to final insights. Each component is engineered to handle the complexity and heterogeneity of real-world data environments.

#### 4.2.1 Multi-Source Data Exploration and Cross-Source Linkage Discovery

To overcome the limitations of analyzing data sources in isolation, ReActInsight initiates its process with an intelligent exploration and linkage discovery pipeline. This phase is critical for building

a semantic understanding of how disparate datasets relate to one another. The process is broken down into four key steps:

1. **Heterogeneous Schema Extraction:** The agent first automatically detects and processes all available data sources, including SQLite databases, CSV files, JSON documents, and unstructured TXT files. It performs lightweight sampling on each source to extract metadata. For structured data (SQL, CSV), it extracts schemas, column names, data types, and representative example values. For unstructured text, it goes a step further by performing content summarization and background knowledge expansion using an LLM to generate a “pseudo-schema” (e.g., a summary column), making the text’s content discoverable and linkable.
2. **Unified Metadata Hub (MetaGraph) Construction:** All extracted metadata is centralized into a unified **MetaGraph**. In this graph, each column from every data source is treated as a node and assigned a globally unique alias (e.g., ‘csv.sales.user\_id’, ‘sqlite.users.customer\_id’). Each node stores its properties, such as data type and example values. This creates a comprehensive, queryable catalog of the entire data ecosystem, providing a holistic data preview before any intensive processing occurs.

#### Formalization of MetaGraph Construction:

Let the set of all data sources be  $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$ . For each source  $d_i \in \mathcal{D}$ , we extract a set of columns  $\mathcal{C}_i = \{c_{i,1}, c_{i,2}, \dots, c_{i,m_i}\}$ . Each column  $c$  is formally represented as a tuple containing its unique identifier (alias), data type, and a set of sample values:

$$c = (\text{id}(c), \tau(c), \mathcal{E}(c))$$

where  $\text{id}(c)$  is the unique alias,  $\tau(c)$  is its data type (e.g., INT, TEXT, FLOAT), and  $\mathcal{E}(c)$  is a set of sampled example values. The MetaGraph,  $\mathcal{G}_M$ , is then defined as the total set of all columns from all data sources:

$$\mathcal{G}_M = \bigcup_{i=1}^N \mathcal{C}_i$$

3. **Entity-Graph Generation via Similarity Analysis:** With the MetaGraph in place, the agent proceeds to discover potential relationships between columns. It performs a pairwise comparison of all columns in the MetaGraph, applying a similarity heuristic to identify potential join keys.

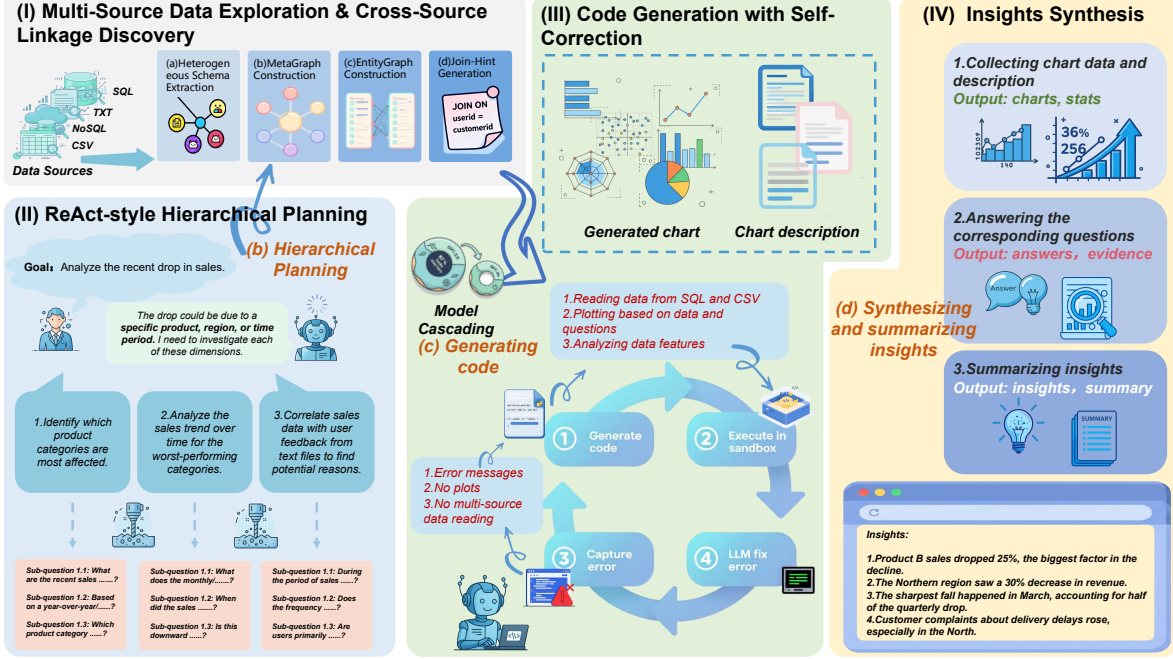


Figure 3: The workflow starts with (I) multi-source data exploration and cross-source linkage discovery, proceeds to (II) ReAct-style hierarchical planning that decomposes the analytical goal into sub-questions, continues with (III) automatic code generation augmented by an iterative self-correction loop, and culminates in (IV) insight synthesis that distills visual evidence and answers into conclusions.

**Formalization of Similarity Analysis:** We formalize the EntityGraph as a weighted, undirected graph  $\mathcal{G}_E = (V, E, W)$ , where the set of vertices  $V$  is the MetaGraph itself ( $V = \mathcal{G}_M$ ). The set of edges  $E$  represents probable entity linkages, with an edge  $c_a, c_b$  existing between any two distinct, type-compatible columns ( $c_a, c_b \in V$  with  $\tau(c_a) = \tau(c_b)$ ) if their similarity score  $\text{Sim}(c_a, c_b)$  exceeds a pre-defined threshold  $\theta$ . The weight function  $W$  assigns this similarity score to each edge, such that  $W(c_a, c_b) = \text{Sim}(c_a, c_b)$ . The similarity score is calculated as a weighted linear combination of name and value distribution similarity:

$$\text{Sim}(c_a, c_b) = w_n \cdot \text{Sim}_{\text{name}}(c_a, c_b) + w_v \cdot \text{Sim}_{\text{val}}(c_a, c_b)$$

Here,  $w_n$  and  $w_v$  are hyperparameters representing the weights for name and value similarity, respectively. Based on our empirical validation, we set  $w_n = 0.6$  and  $w_v = 0.4$  (where  $w_n + w_v = 1$ ). The name similarity component  $\text{Sim}_{\text{name}}$  is defined as a binary exact match indicator (i.e.,  $\text{Sim}_{\text{name}} \in \{0, 1\}$ ). The value similarity  $\text{Sim}_{\text{val}}$  is computed using the Jaccard similarity coefficient on the sets of sampled example values,  $\mathcal{E}(c_a)$  and  $\mathcal{E}(c_b)$ :

$$\text{Sim}_{\text{val}}(c_a, c_b) = \frac{|\mathcal{E}(c_a) \cap \mathcal{E}(c_b)|}{|\mathcal{E}(c_a) \cup \mathcal{E}(c_b)|}$$

The resulting EntityGraph,  $\mathcal{G}_E$ , thus maps potential join paths across the different data sources, with the edge weights quantifying the confidence of each linkage.

- Actionable Join-Hint Formulation:** Finally, the agent takes the highest-scoring relationships from the EntityGraph and formulates a concise, human-readable **Join-Hint**. This hint, such as "You must *JOIN ON*: `csv.sales.user_id = sqlite.users.customer_id`", serves as a direct instruction for the subsequent code generation phase.

**Formalization of Hint Selection:** Let the set of discovered edges in the EntityGraph be  $E$ . The agent constructs an ordered list of candidate join keys,  $\mathcal{L}$ , by sorting the edges in  $E$  in descending order of their weight  $W$ .

$$\mathcal{L} = \text{sorted}(E, \text{key} = W, \text{reverse}=\text{True})$$

The final set of hints,  $\mathcal{H}$ , is generated by selecting the top- $k$  edges from this list:

$$\mathcal{H} = \{e \in \mathcal{L} \mid \text{rank}(e) \leq k\}$$

This set  $\mathcal{H}$  is then translated into the textual Join-Hint provided to the code generation module.

#### 4.2.2 Hierarchical Planning with ReAct-style Reasoning

At its core, ReActInsight leverages a hierarchical planning module that operationalizes the ReAct paradigm. Given a high-level analytical objective from the user (e.g., “*Analyze the recent drop in sales*”), the agent first reasons about the potential causes and breaks the problem down into a sequence of logical sub-goals. As illustrated in Figure 3, this hierarchical decomposition transforms a broad query into a structured analytical plan. Each sub-goal is then further decomposed into specific, actionable sub-questions (e.g., “*What are the total sales per product category for the last quarter?*”), which directly guide the code generation step.

#### 4.2.3 Robust Code Generation and Adaptive Visualization

For each actionable sub-question, ReActInsight generates the necessary code (e.g., SQL queries, Python scripts using Pandas) to perform the analysis. Crucially, the prompt provided to the LLM for code generation is augmented with the **Join-Hint** derived from the EntityGraph. This explicitly guides the model to construct correct JOIN statements (e.g., SQL JOINS or Python Pandas merges) across different data sources, which is essential for answering complex, multi-faceted questions.

This process is designed to be robust. After generation, the agent executes the code in a sandboxed environment and captures all outputs, including stdout and stderr. If an error is detected, a self-correction loop is initiated. The agent feeds the original code, the error message, and the corresponding sub-goal back into itself to generate a corrected version, iterating until the code executes successfully. Upon successful execution, a key innovation of our agent becomes apparent: its adaptive visualization capability. Instead of defaulting to simple histograms or tables, the agent reasons about the nature of the data (now potentially joined from multiple sources) and the sub-question to select the most appropriate visualization type to reveal underlying patterns. This includes generating scatter plots to show correlations, time-series plots for trends, heatmaps for matrix data, and bar charts for comparisons, ensuring that the intermediate results are not only computationally correct but also intuitively presented.

#### 4.3 Insight Synthesis and Summary

ReActInsight does not simply present a series of charts; it analyzes the results from each successfully executed step. It aggregates the observations, identifies key findings, and constructs a final, coherent narrative. This summary report provides users with a high-level overview, detailed findings supported by visualizations, and actionable recommendations, fulfilling end-to-end data analysis.

#### 4.4 Optimization via Model Cascading

To balance performance with operational costs, ReActInsight implements a model cascading strategy. Routine and structurally simple tasks, such as the generation of sub-questions, are delegated to smaller, faster, and more cost-effective models (e.g., GPT-3.5-Turbo). Only when a task proves too complex, or when the self-correction module is triggered by repeated failures, does the agent escalate the task to a more powerful state-of-the-art model (e.g., GPT-4o, GPT-5). This strategy significantly optimizes both computational cost and response latency, making ReActInsight a practical and efficient solution for real-world applications.

### 5 Experiments

#### 5.1 Experimental Setup

**Baselines.** We benchmarked a diverse set of state-of-the-art data analytics agents on UniDataBench, including CodeGen (Majumder et al., 2025), ReAct (Yao et al., 2023), Data-to-Dashboard (Zhang and Elhamod, 2025), Pandas Agent (LangChain, 2024), DABstep (Egg et al., 2025), and Agent-Poirot (Sahu et al., 2025). Detailed descriptions of all baselines are provided in Appendix C.1.

To ensure that the evaluation focuses on core reasoning capabilities rather than information access, we implemented baseline adaptations to maintain strict information parity. For baselines originally designed for single-source inputs, we extended their file-reading utilities to provide programmatic access to all file formats provided. This ensures that all agents operate on the same raw information, isolating their multi-step reasoning performance as the primary variable for comparison.

**Evaluation Metrics.** Following established practices (Sahu et al., 2025), we use the G-Eval (Liu et al., 2023) metric to assess agents at both levels of insight and summary. More detailed definitions of these two levels are provided in Appendix C.2.

Agent	Insight-level G-Eval Scores				Summary-level G-Eval Scores			
	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.
ReAct (Yao et al., 2023)	0.3115	0.3024	0.2982	0.3040	0.3549	0.3430	0.3223	0.3401
CodeGen (Majumder et al., 2025)	0.3197	0.2999	0.3256	0.3151	0.3796	0.3342	0.4075	0.3738
Pandas Agent (LangChain, 2024)	0.3831	0.3978	0.3740	0.3850	0.5551	0.5332	0.4905	0.5263
D2D (Zhang and Elhamod, 2025)	0.3890	0.4346	0.3234	0.3823	0.4964	0.5235	0.4336	0.4845
DABstep (Egg et al., 2025)	0.4236	0.4326	0.3916	0.4159	0.5379	0.5377	0.4744	0.5167
AgentPoirot (Sahu et al., 2025)	0.4810	0.4448	0.3826	0.4361	0.5917	0.5708	0.5229	0.5618
ReActInsight (Ours)	<b>0.5030</b>	<b>0.4704</b>	<b>0.4517</b>	<b>0.4750</b>	<b>0.5984</b>	<b>0.5789</b>	<b>0.5605</b>	<b>0.5792</b>
Improvement (%)	+4.57%	+5.76%	+18.06%	+8.92%	+1.13%	+1.42%	+7.19%	+3.10%

Table 3: Comparison of agent performance across varying difficulty levels on UNIDATABENCH.

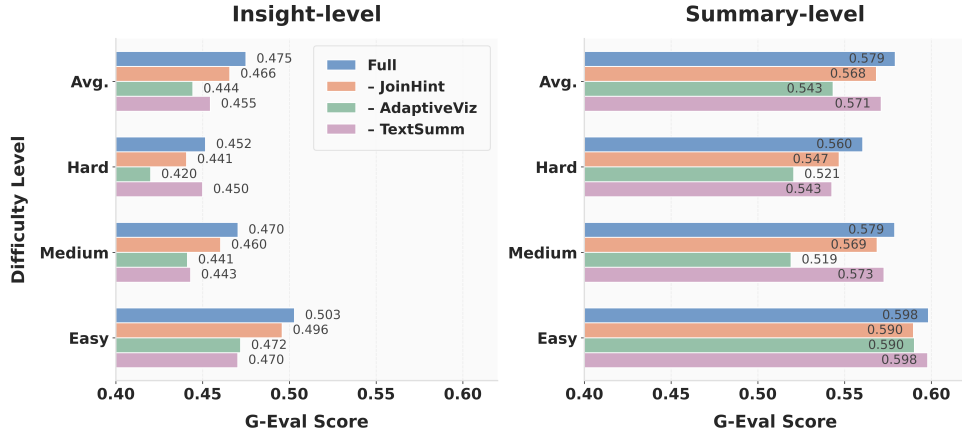


Figure 4: Ablation results of ReActInsight modules evaluated on dual performance metrics.

**Implementation Details.** We conducted a comparative analysis across a range of LLMs to serve as the backbone for our agents. For all experiments, a temperature of  $T = 0$  was applied. The model configurations, prompt references, and more hyperparameters for the four stages of ReActInsight are summarized in Appendix C.3. The full prompt templates are provided in Appendix E.

## 5.2 Performance Comparison

As detailed in Table 3, ReActInsight surpasses all baseline methods across all difficulty levels on the UniDataBench benchmark. Single-agent methods, CodeGen and ReAct demonstrate limited effectiveness, highlighting the inadequacy of generic code generation or simple action-based frameworks for complex, multi-source data analysis. The specialized focus of Data-to-Dashboard and Pandas Agent on data analysis leads to respectable scores and AgentPoirot’s multi-step sub-question approach provides further improvement. However, their performance difference is primarily on “Easy” difficulty tasks, with smaller variations on more challenging tasks, underscoring the need for optimized multi-source data processing. Furthermore, the mediocre results of DABstep, an agent designed for multi-source analysis, indicate that existing

multi-source agents are not easily adapted to high-dimensional data analysis. Our proposed method, ReActInsight, demonstrates superior performance by initially constructing a unified map of all data sources, which is then combined with hierarchical planning and self-correction. ReActInsight achieves an insight-level score of 0.4750 and a summary-level score of 0.5792, representing an 8.9% and 3.1% improvement over the baseline, respectively. Notably, the performance advantage of ReActInsight becomes more pronounced as the task difficulty increases. On “hard” difficulty tasks, our model scores 0.4517, surpassing the AgentPoirot baseline (0.3826) by a significant 18%. This trend highlights our agent’s capability in handling data analysis tasks that involve a variety of data formats. Detailed results of human evaluation are provided in Appendix D.1, followed by illustrative case studies in Appendix D.3.

## 5.3 Efficiency Analysis

Table 4 shows that the cascaded ReActInsight configuration maintains strong insight quality (0.4510) while reducing the estimated cost from  $\sim$ \$0.15 to  $\sim$ \$0.08 per task, corresponding to an approximately 47% reduction relative to the single-GPT-4o ReActInsight setting. Compared with the strongest

Configuration	Insight Avg.	Time / task	API calls / task	Cost / task
AgentPoirot (single GPT-4o)	0.4361	~1.6 min	~20	~\$0.15
ReActInsight (single GPT-4o)	0.4737	~1.2 min	~6	~\$0.15
ReActInsight (cascade: GPT-4o-mini + GPT-4o)	0.4510	~1.0 min	~6	~\$0.08

Table 4: Efficiency comparison across baseline and ReActInsight configurations.

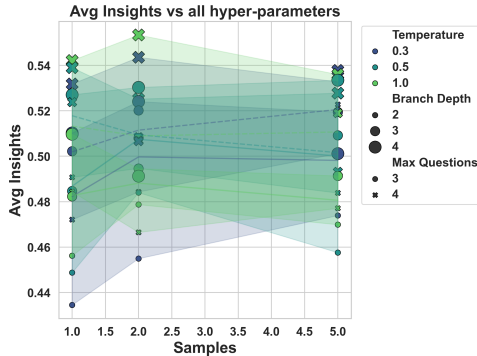


Figure 5: Impact of hyperparameters on Insight-level.

baseline, AgentPoirot, ReActInsight also reduces the average number of API calls from ~20 to ~6 and shortens the runtime from ~1.6 to ~1.0 minutes per task. These results indicate that model cascading improves deployment efficiency while preserving a competitive performance level.

#### 5.4 Ablation Study

We conducted an ablation study by systematically removing the JoinHint mechanism, the Adaptive Visualization module, and the Text Summarizer from the full ReActInsight (gpt-4o). As shown in Figure 4, the results unequivocally demonstrate that the full framework outperforms all ablated variants across both Insight-level and Summary-level evaluations, confirming that each component contributes positively to the overall performance.

The removal of the adaptive visualization module caused the most significant performance degradation, with the average Summary-level score dropping from 0.5792 to 0.5434. This highlights its critical role in generating effective charts for deriving insights. Ablating the Text Summarizer also led to a notable decline, while removing the JoinHint mechanism resulted in a more modest but still consistent performance drop.

#### 5.5 Hyperparameter Study

We conducted a hyperparameter study to evaluate the impact of temperature, samples, max questions, and branch depth on the quality of generated insights and summaries. As shown in Figure 5, there is a fundamental trade-off between

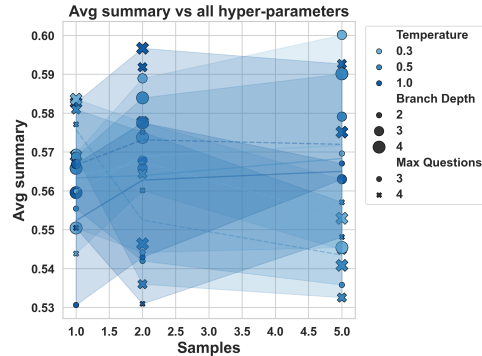


Figure 6: Impact of hyperparameters on Summary-level.

the two objectives, demonstrating that no single configuration is optimal for both. Specifically, the highest insight score (0.5534) was achieved with a configuration favoring creativity and deep exploration: a high temperature (1.0), max questions of 4, and a branch depth of 4. Conversely, in Figure 6, the peak summary score (0.6001) required a setting that prioritizes precision and focus: a low temperature (0.3), a high sample count (samples=5), and a more moderate exploration depth (max questions=3, branch depth=3). This clear divergence in optimal settings validates the approach of employing task-specific hyperparameter configurations to maximize performance for each respective goal. We further evaluate the impact of backbone LLMs and schema matching hyperparameters in Appendix D.2. These results validate our framework’s design and parameter robustness.

## 6 Conclusion

In this paper, we presented UniDataBench, a new benchmark for evaluating data analytics agents’ ability to work with structured and unstructured data. UniDataBench offers a diverse set of datasets and tasks reflecting real-world business analytics challenges. We also proposed ReActInsight, an LLM-based agent designed to handle multiple data sources. Our experiments show UniDataBench effectively assesses agent performance, highlighting its potential to drive innovation in data analytics. Overall, our work provides a robust foundation for advancing data analytics agents in handling complex, real-world data environments.

## Limitations

The current data construction approach relies heavily on manual construction, which is labor-intensive and costly. UniDataBench currently consists of 100 high-level analytical tasks. The current sample size may present challenges in establishing robust statistical significance when performance margins between agents are narrow. Therefore, future efforts will explore automated construction methods to overcome these constraints and expand the dataset size efficiently.

Regarding scope, UniDataBench prioritizes pattern rediscovery over open-ended exploration. While this focus precludes evaluation of open-ended exploratory analysis where targets are not pre-defined, it enables objective and verifiable performance measurement against a stable ground truth, which is essential for the rigorous evaluation of analytical accuracy.

Furthermore, while UniDataBench supports multiple data formats, it does not yet encompass multimodal data (e.g., images or audio) or the full diversity of industrial domains. Future efforts will focus on expanding both modal and domain coverage to enhance the benchmark's generalizability and better reflect the multifaceted complexity of real-world environments.

## Ethical Statement

Our agent is primarily designed to assist users in automatically generating data analysis reports and uncovering valuable insights from diverse data sources. However, like all AI systems, it is not infallible. For instance, LLMs and agents may occasionally generate hallucinations or produce misleading information. Therefore, the reports generated by our system should only be used as an auxiliary tool. We strongly recommend that professionals thoroughly review and verify the results before making any critical decisions.

## Acknowledgements

This project is supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM113), National Natural Science Foundation of China (92470121, 62402016), National Key R&D Program of China (2024YFA1014003), Zhongguancun Academy (C20250204, C20250602), Beijing Major Science and Technology Project (Z251100008125043,

Z251100008425023), and High-performance Computing Platform of Peking University.

## References

- Jincheng Bai, Zhenyu Zhang, Jennifer Zhang, and Jason Zhu. 2025. Insight agents: An LLM-based multi-agent system for data insights. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4335–4339.
- Ken Black. 2023. *Business statistics: for contemporary decision making*. John Wiley & Sons.
- Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. [Quickinsights: Quick and automatic discovery of insights from multi-dimensional data](#). In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD '19*, page 317–332, New York, NY, USA. Association for Computing Machinery.
- Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. 2025. [Dabstep: Data agent benchmark for multi-step reasoning](#). *Preprint*, arXiv:2506.23719.
- Ahmad Ghazal, Tilmann Rabl, Mingqing Hu, Francois Raab, Meikel Poess, Alain Crolette, and Hans-Arno Jacobsen. 2013. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*, pages 1197–1208.
- Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, Zejian Yuan, and Dongmei Zhang. 2024. [Text2analysis: a benchmark of table question answering with advanced data analysis and unclear queries](#). In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'24/IAAI'24/EAAI'24*. AAAI Press.
- Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Bin-hao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, and 9 others. 2025. [Data interpreter: An LLM agent for data science](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821, Vienna, Austria. Association for Computational Linguistics.
- 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. In *The Twelfth International Conference on Machine Learning (ICML)*.

- Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, and 1 others. 2024. Da-code: Agent data science code generation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13487–13521.
- Veit Jahns. 2013. [Data insights: new ways to visualize and make sense of data by hunter whitney](#). *SIGSOFT Softw. Eng. Notes*, 38(6):45–46.
- Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2025. [DSBench: How far are data science agents from becoming data science experts?](#) In *The Thirteenth International Conference on Learning Representations*.
- Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: a natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning, ICML'23*. JMLR.org.
- LangChain. 2024. Pandas dataframe. <https://python.langchain.com/v0.2/docs/integrations/toolkits/pandas/>. Accessed: 2024-06-03.
- Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin SU, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. [Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows](#). In *The Thirteenth International Conference on Learning Representations*.
- 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 *Thirty-seventh Conference on Neural Information Processing Systems*.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. [G-eval: NLG evaluation using gpt-4 with better human alignment](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.
- Jinwei Lu, Yuanfeng Song, Zhiqian Qin, Haodi Zhang, Chen Zhang, and Raymond Chi-Wing Wong. 2025. [Bridging the gap: Enabling natural language queries for nosql databases through text-to-nosql translation](#). *arXiv preprint arXiv:2502.11201*.
- Jinwei Lu, Yuanfeng Song, Chen Zhang, and Raymond Chi-Wing Wong. 2026. [Multivis-agent: A multi-agent framework with logic rules for reliable and comprehensive cross-modal data visualization](#). *Proc. ACM Manag. Data*, 4(1).
- Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. [InsightPilot: An LLM-empowered automated data exploration system](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 346–352, Singapore. Association for Computational Linguistics.
- Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. 2025. [Discoverybench: Towards data-driven discovery with large language models](#). In *International Conference on Learning Representations (ICLR)*.
- OpenAI. 2024. Code interpreter. <https://platform.openai.com/docs/assistants/tools/code-interpreter>. Accessed: 2024-06-03.
- Alberto Sánchez Pérez, Alaa Boukhary, Paolo Papotti, Luis Castejón Lozano, and Adam Elwood. 2025. [An LLM-based approach for insight generation in data analysis](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 562–582, Albuquerque, New Mexico. Association for Computational Linguistics.
- Zhiqian Qin, Yuanfeng Song, Jinwei Lu, Yuanwei Song, Shuaimin Li, and Chen Jason Zhang. 2025. [MultiTEND: A multilingual benchmark for natural language to NoSQL query translation](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24632–24657, Vienna, Austria. Association for Computational Linguistics.
- Gaurav Sahu, Abhay Puri, Juan A. Rodriguez, Amirhossein Abaskohi, Mohammad Chegini, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vázquez, Nicolas Chapados, Christopher Pal, Sai Rajeswar, and Issam H. Laradji. 2025. [Insightbench: Evaluating business analytics agents through multi-step insight generation](#). In *ICLR*.
- Rajeev Sharma, Sunil Mithas, and Atreyi Kankanhalli. 2014. Transforming decision-making processes: a research agenda for understanding the impact of business analytics on organisations. *European Journal of Information Systems*, 23(4):433–441.
- Yuanfeng Song, Xuefang Zhao, and Raymond Chi-Wing Wong. 2024. [Marrying dialogue systems with data visualization: Interactive data visualization generation from natural language conversations](#). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2733–2744.
- Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2024. [Struc-bench: Are large language models good at generating complex structured tabular data?](#)

- In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 12–34, Mexico City, Mexico. Association for Computational Linguistics.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.
- Ran Zhang and Mohannad Elhamod. 2025. Data-to-dashboard: Multi-agent llm framework for insightful visualization in enterprise analytics. *arXiv preprint arXiv:2505.23695*.
- Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2024a. [Data-copilot: Bridging billions of data and humans with autonomous workflow](#). In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.
- Yuge Zhang, Qiyang Jiang, XingyuHan XingyuHan, Nan Chen, Yuqing Yang, and Kan Ren. 2024b. [Benchmarking data science agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5677–5700, Bangkok, Thailand. Association for Computational Linguistics.

## A UniDataBench Details

### A.1 Task Illustration

As illustrated in the top of Figure 7, existing benchmarks often rely on idealized scenarios with data confined to a single format, such as CSV files. In contrast, UniDataBench provides a “real-world” situation where data is stored in multiple formats (e.g., CSV, TXT, NoSQL, SQLite), compelling the agent to integrate information to fulfill a user’s analytical goal. Each problem instance in UniDataBench is structured to test this capability and consists of three key components:

- **Goal:** A high-level, open-ended question or objective posed by the user.
- **Data Files and Info:** A collection of data files in multiple sources.
- **Ground-Truth Insights:** A set of key insights that an agent is expected to discover.

### A.2 Typical Examples in UniDataBench

To provide a concrete understanding of our benchmark’s design, we illustrate its structure through two examples, as shown in Figure 7.

**Example 1: Medium-Difficulty Level** This task simulates a typical business intelligence query requiring trend analysis and attribution.

- **Goal:** Analyze the trend and reasons for the ‘DAU’ (Daily Active Users) of the ‘StoryLoom’ feature within one month after 2022-02-14.
- **Data Files:** The task involves three data files across two formats:
  - CSV: `user_activity`, `covid_cases`
  - NoSQL: `features_log`
- **Analytical Challenge:** To successfully answer the query, an agent cannot rely on a single file. It must first identify the DAU trend from `user_activity` data. Then, to explain the reasons for this trend, it must integrate external factors from `covid_cases` and internal product changes from the `features_log` (which contains the launch date of a new ‘AI Removing’ function). The expected insights involve correlating the DAU growth with both the rising COVID-19 cases and the launch of the new feature.

**Example 2: Hard-Difficulty Level** This task presents a more complex scenario involving a deeper causal analysis across a wider array of data types, including unstructured text and a relational database.

- **Goal:** Analyze the changes and reasons of the core user metric ‘lead rate’ after halving the advertising budget in August 2024.
- **Data Files:** This problem includes data from four different formats:
  - CSV: `user_activity`
  - NoSQL: `exp_periods`
  - SQLite: `marketing`
  - TXT: `definitions`
- **Analytical Challenge:** The complexity is significantly increased. The agent must first consult the unstructured `definitions.txt` file to understand the key metric. It then needs to query the SQLite database (`marketing`) and join this information with user behavior data from CSV (`user_activity`) and event timelines from NoSQL (`exp_periods`). The core challenge lies in disentangling the effects of the budget cut on different metrics (click-through rate vs. conversion rate) and segmenting the analysis by user lifecycle to uncover nuanced insights. This requires the agent to not only handle more data formats but also to construct a more sophisticated analytical workflow.

Through these carefully constructed tasks, UniDataBench provides a robust framework for evaluating an agent’s ability to perform integrated, multi-faceted data analysis that is representative of practical, real-world requirements.

### A.3 Example of Synthesizing and Validating a Multi-Source Dataset

We outline a practical case of how we created a multi-source dataset for a complex analytical scenario. This example start from following goal:

**Goal:** Analyze the combined impact of the COVID-19 pandemic and advertising budget cuts on the lead conversion rate (CVR) in Q2 2022.

**Insights we aim to embed within the data:**

- The lead conversion rate (CVR) in Q2 2022 showed an overall downward trend.
- The CVR is negatively correlated with the severity of the pandemic.
- After implementing an advertising budget cut, the rate of CVR decline slowed down.
- The improvement in user quality from reduced ad investment mitigated the overall CVR decline.

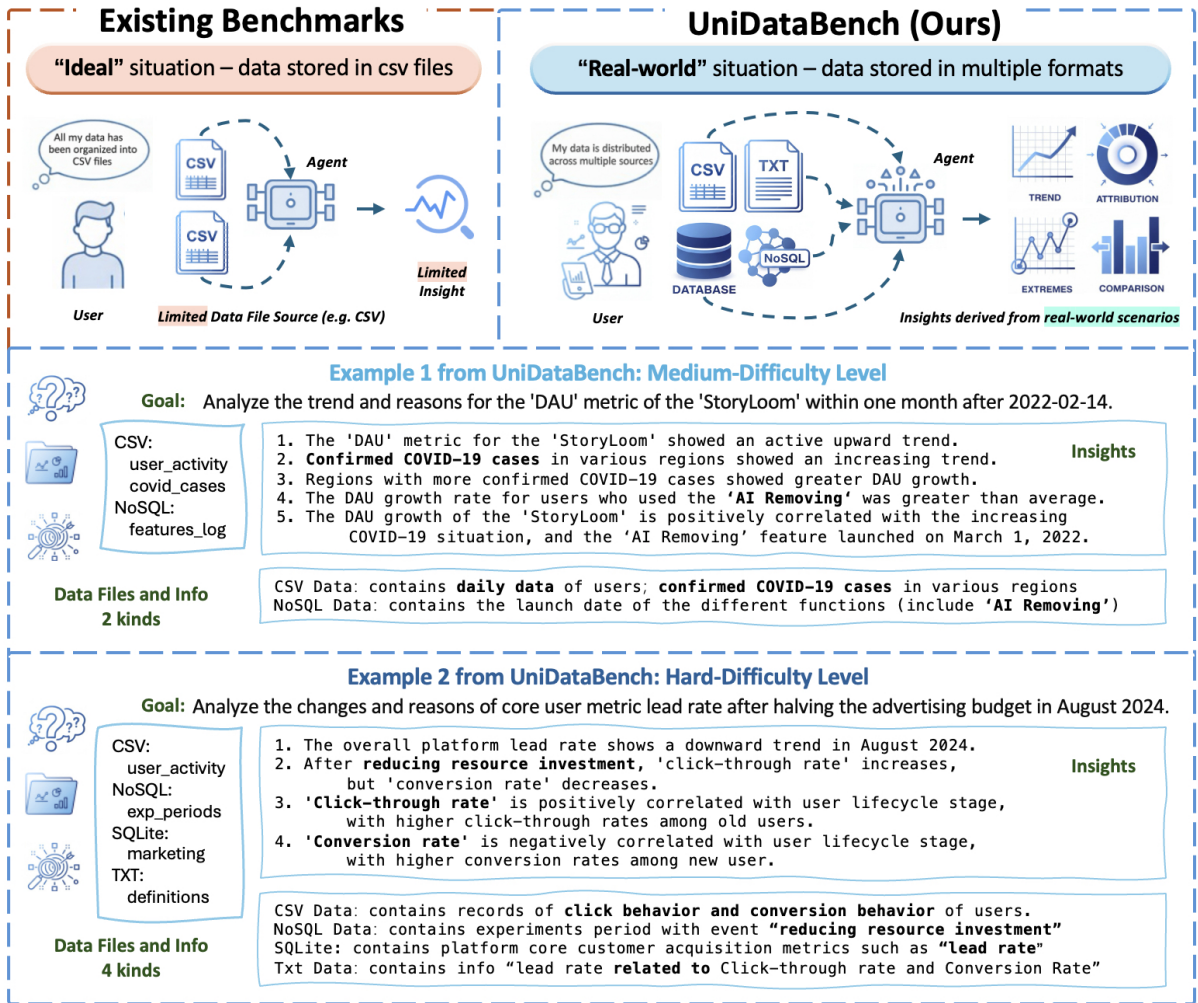


Figure 7: Existing benchmarks (left) that evaluate agents on isolated data sources. This represents an idealized yet limited scenario for data analysis, which leads to incomplete insights. UniDataBench (right), designed to simulate practical data analysis requirements. User pose a high-level goal, then will get comprehensive insights about trends, extreme values, comparisons and attribution.

### A.3.1 Schema Design and Standardization

Based on the goal and ground-truth insights, we first design the schemas for the required multi-source data. This involves defining core fields that directly enable the analysis and ensuring relationships are maintained across files.

(a) **Core Analysis Dataset (CSV):** This file contains the primary time-series data. The schema is designed to directly support the core analysis by linking CVR to the pandemic and ad spend.

Field Name	Type	Description
log_date	string	Log date for Q2 2022
region_id	string	Unique region identifier
daily_cvr	float	Daily conversion rate
ad_spend	float	Daily advertising spend in USD
...	...	(12 other columns)

Table 5: Schema for the core 'daily\_summary.csv' file.

(b) **Event Period Data (NoSQL):** The time boundaries for the advertising strategy change are stored in NoSQL. This simulates a real-world scenario where company-wide events are logged in a semi-structured format. The agent must parse this file to find the relevant periods—"Investment Period" and "Budget Cut Period"—among other irrelevant entries, and extract their start and end dates to correctly segment the analysis.

Listing 1: Excerpt from 'company\_events.json'. The agent must identify the relevant financial periods among other distracting entries.

```
{
  "company_periods": [
    {
      "period_id": "p2021q4-sales",
      "period_name": "Q4 2021 Holiday
      ↪ Sale",
      "type": "Marketing",
      ...
    }
  ]
}
```

```

    },
    {
      "period_id": "p2022q1-invest",
      "period_name": "Investment
↪ Period",
      "type": "Financial",
      "start_date": "2022-03-01",
      "end_date": "2022-05-31",
      "description": "Period of
↪ increased spending..."
    },
    {
      "period_id": "p2022q2-budget",
      "period_name": "Budget Cut
↪ Period",
      "type": "Financial",
      "start_date": "2022-06-01",
      "end_date": "2022-06-30",
      "description": "A short period of
↪ reduced expenditure..."
    }
  ]
}

```

### A.3.2 Distractor Column Introduction

To enhance analytical complexity, we introduce more than 10 distractor columns into the ‘csv’ file (e.g., `campaign_id`, `weather_index`). Similarly, there are some irrelevant event periods (e.g., “Q4 2021 Holiday Sale”) stored in ‘nosql’ file. This multi-source noise challenges the agent to perform effective feature selection and information extraction, a common task for data analysts.

### A.3.3 Synthetic Data Creation

The entire data generation pipeline is implemented using Python scripts. We instruct an LLM to generate a script that adheres to the designed schemas and embeds the intended patterns. For example, the script defines region-specific parameters to control the base CVR and its sensitivity to factors like COVID-19 severity and ad spend.

Listing 2: Python snippet for generating region-specific metadata used in data synthesis.

```

# Create region metadata
region_meta = []
for i in range(NUM_REGIONS):
    region_id = f"R{1000 + i}"
    region_name = f"Region-{chr(65 + (i
% 26))}{(i // 26) + 1}"
    # base cvr for region (between 40%
    and 60%)
    base_cvr = random.uniform(0.4, 0.60)
    # base daily ad spend (mean) in
    dollars
    base_ad_spend = random.uniform(800,
    4500)
    # region-level sensitivity to covid
    severity (coeff)
    covid_sensitivity = random.uniform
    (0.0009, 0.0022)

```

```

# ad spend effect on CVR (higher
    spend, lower quality)
    ad_spend_effect = random.uniform
    (0.00002, 0.00005)
    region_meta.append({
        "region_id": region_id,
        "region_name": region_name,
        "base_cvr": base_cvr,
        "base_ad_spend": base_ad_spend,
        "covid_sensitivity":
        covid_sensitivity,
        "ad_spend_effect":
        ad_spend_effect
    })

```

### A.3.4 Data Validation and Visualization

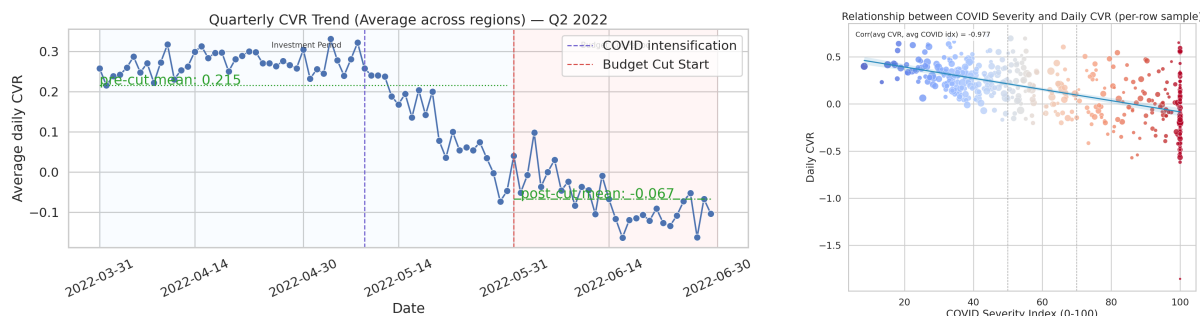
After generation, we perform a crucial validation step to ensure the ground-truth insights are discoverable. Visualization is a primary method for this confirmation. As shown in Figure 8, the plot serves as visual proof that the intended analytical patterns are present and accessible. Specifically, the visualization confirms: (1) the overall downward trend of CVR, (2) its negative correlation with the pandemic’s severity, and (3) a clear change in the CVR’s decline rate after the budget cut was implemented in June.

### A.4 Task Across Analysis Theme

To ensure the practical relevance and comprehensive coverage of our benchmark, the tasks in UniDataBench are sourced from authentic, real-world data analysis scenarios. These scenarios are thoughtfully categorized into five core business themes, each representing a common area of analytical inquiry in industry. The analytical tasks are structured across five core business domains: User Behavior Analysis (29 tasks), Business Performance Analysis (23 tasks), Market Competition Analysis (13 tasks), Sales Channel Analysis (18 tasks), and Macro-Industry Analysis (17 tasks). An illustrative example for each of these five themes is provided in Table 7, showcasing a representative goal, the expected insights, and the necessary data files for each task.

### A.5 Human-Labeled, Synthetic, and Validated Components

To ensure the integrity and real-world applicability of UniDataBench, we adopt a “human-in-the-loop” construction strategy that balances the authenticity of business logic with the scalability of data synthesis. Table 6 provides a granular breakdown of the seven primary stages in our pipeline, distinguishing between components that are human-labeled,



(a) Time series analysis of conversion rate.

(b) Rate versus COVID severity.

Figure 8: Validation plot for the synthesized data, confirming that the intended CVR trends and the impact of the budget cut are successfully embedded.

synthetically generated, and expert-validated.

Specifically, to reduce potential model hallucinations in complex business reasoning, we anchor foundational analytical goals and ground-truth insights in expert knowledge. With this human-labeled foundation, we then employ LLMs and automated scripts for scalable data generation. Multi-source schemas and distractor columns are designed to replicate the structural complexity of real-world databases, while underlying records are produced via Python scripts rather than direct LLM text generation. This decoupled design is intended to maintain numerical consistency and structural coherence across thousands of records, addressing the token-length constraints and reasoning-drift issues often associated with naive LLM-based generation.

To verify the alignment between the synthesized data and the original insights, experts perform a two-stage inspection. First, a structural audit of the data schemas is conducted to ensure they support the intended analytical objectives. Following data population, a visual verification phase is implemented where experts utilize visualization tools to confirm that the ground-truth insights are discoverable within the datasets. Only instances where the generated patterns are empirically confirmed to match the original human-labeled insights are included in the final benchmark.

#### A.6 Expert Qualifications and Annotation Reliability

All annotators participated voluntarily after receiving comprehensive task training, and were compensated at a rate exceeding the standards mandated by local labor laws. During dataset construction, the extraction and mapping process was conducted

by three senior data scientists, each holding an academic degree in Data Science and over three years of industry experience in Business Intelligence (BI). We held a formal calibration session to align the understanding of evaluation rubrics. To assess reliability, we calculated the Intraclass Correlation Coefficient (ICC 2, 1), yielding a value of 0.79, which indicates “Good” inter-annotator agreement based on established standards.

### B Benchmark Quality Assessment

To validate the quality and robustness of UniDataBench, we conducted a rigorous evaluation from multiple perspectives. We employed both Large Language Models (LLMs) and human experts to assess the dataset, ensuring a comprehensive and objective analysis. This process was facilitated through a specifically developed GUI, as shown in Figure 9, which guided the expert reviewers through a structured review of the ground-truth included in the benchmark.

#### B.1 Evaluation Metrics

Our quality assessment is centered on four key dimensions that reflect the benchmark’s core characteristics. These metrics are designed to be direct and clearly address the fundamental qualities of a data analysis task.

- **Data Suitability:** Assesses the overall fitness of the provided data by evaluating both its **content quality** and **structural integrity**. **Content quality** involves verifying the presence of necessary attributes, appropriate data granularity, and the absence of critical flaws (e.g., an age of -1, excessive missing values). **Structural integrity** evaluates the realism and logical coherence of the schema design, particularly how different data

#	Stage	Pipeline Component	Nature	Method / Source
1	Extraction	High-level Goals	Human-Labeled	Expert-extracted from real-world reports
2	Extraction	Ground-truth Insights	Human-Labeled	Expert-extracted from real-world reports
3	Synthesis	Multi-source Schema	Synthetic	LLM-generated based on domain context
4	Synthesis	Distractors	Synthetic	Proactive injection of irrelevant columns
5	Validation	Schema Review	Validated	Expert inspection for structural coherence
6	Synthesis	Underlying Data	Synthetic	Python-generated to align with target trends
7	Validation	Insight & Data Verification	Validated	Experts verified via generated visualizations

Table 6: Step-by-step breakdown of the UniDataBench pipeline components and their respective natures.

Goal	Insights	Data Files
<b>Task Domain:</b> User Behavior Analysis (Tasks Size: 29)		
<b>Task No.:</b> 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 21, 22, 23, 24, 25, 42, 48, 49, 53, 54, 55, 56, 85, 92		
Analyze the impact of features launched in Q2 2024 on the 'DAU' metric for the video editing product 'StoryLoom'	<ol style="list-style-type: none"> <li>The overall DAU metric showed a rising trend in Q2 2024.</li> <li>The DAU of users who used the 'Fill Light Filter' feature accounts for a high proportion of the overall DAU, while the DAU of users who used the 'Vintage Filter' feature accounts for a very low proportion.</li> <li>The DAU for users of the 'Fill Light Filter' feature showed a rising trend.</li> <li>The DAU for users of the 'Vintage Filter' feature showed no significant change.</li> <li>The increase in DAU this quarter is mainly attributed to the positive impact of the 'Fill Light Filter' feature, whereas the 'Vintage Filter' feature had no significant impact on the overall DAU growth.</li> </ol>	features_metadata.json storyloom_usage_q2.csv
<b>Task Domain:</b> Business Performance Analysis (Tasks Size: 23)		
<b>Task No.:</b> 7, 26, 27, 28, 32, 33, 34, 37, 38, 39, 40, 43, 44, 45, 46, 47, 51, 73, 75, 91, 93, 94, 99		
Analyze the changes in rental business transaction volume and the reasons from February to July 2023	<ol style="list-style-type: none"> <li>The transaction volume of the rental business exhibited an upward trend from February to July.</li> <li>The total exposure of the rental business has maintained a continuous upward trend since March 2023.</li> <li>The transaction volume of all housing types showed an upward trend.</li> <li>The new housing type 'Branded Apartments' launched in April 2023 boosted the transaction volume of the rental business.</li> <li>The upward trend of trading volume in June and July 2023 was significantly higher than before.</li> <li>The increase in the transaction volume of the rental business was jointly influenced by three factors: the rise in exposure since March, the launch of the new 'Branded Apartments' in April, and the peak rental season in June and July.</li> </ol>	Break_Information_Report.txt daily_transactions.csv housing_type_daily.csv
<b>Task Domain:</b> Market Competition Analysis (Tasks Size: 13)		
<b>Task No.:</b> 19, 20, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 84		
Analyze Changes in Public Opinion on 'LucidPic' Membership Benefits in 2024 Based on NSR and Its Components	<ol style="list-style-type: none"> <li>The NSR (Net Sentiment Ratio) of public opinion related to 'LucidPic' membership benefits improved from negative in Q3 to positive in Q4.</li> <li>The number of positive feedback in Q3 was the lowest of the year, while the number of negative feedback was the highest of the year.</li> <li>The number of positive feedback in Q4 was the highest of the year, while the number of negative feedback was the lowest of the year.</li> <li>Among all quarters in 2024, only Q4 saw the number of positive feedback exceed that of negative feedback.</li> <li>The increase in the number of positive feedback in Q4 was mainly concentrated in the aspect of 'Paid Features Valuable'.</li> <li>The growth of positive feedback in Q4 was related to the paid feature 'AI removal' launched in Q4.</li> </ol>	lucidpic_knowledge.txt feature_launch.json membership_feedback.db
<b>Task Domain:</b> Sales Channel Analysis (Tasks Size: 18)		
<b>Task No.:</b> 17, 18, 29, 30, 31, 35, 36, 41, 50, 52, 83, 86, 87, 95, 96, 97, 98, 100		
Comparison of the conversion funnel efficiency between the 'QuickMart' platform APP and Mini-Program channels, identifying their respective strengths and bottlenecks	<ol style="list-style-type: none"> <li>The total order volume of the APP channel is higher than that of the 'Mini-Program' channel.</li> <li>By comparing stages in the conversion funnel, the 'Mini-Program' has a higher 'Store_Entry' stage conversion rate.</li> <li>By comparing stages in the conversion funnel, the APP channel's 'Order_Completion' stage conversion rate is significantly higher than that of the 'Mini-Program'.</li> </ol>	daily_store_performance.csv funnel_knowledge.txt
<b>Task Domain:</b> Macro-Industry Analysis (Tasks Size: 17)		
<b>Task No.:</b> 67, 68, 69, 70, 71, 72, 74, 76, 77, 78, 79, 80, 81, 82, 88, 89, 90		
Analyze the indicator trends of PPI and CPI in developed economies from 2022 to 2024 and their impact on corporate profits	<ol style="list-style-type: none"> <li>From 2022 to 2024, both PPI and CPI showed a downward trend.</li> <li>From 2022 to 2024, the decline rate of PPI was much greater than that of CPI.</li> <li>Due to the difference in decline speeds, PPI remained significantly lower than CPI for most of the time, forming a persistent negative price spread.</li> <li>The continuous widening of the negative PPI-CPI spread indicates that the cost of corporate inputs is falling faster than product selling prices, which is favorable for corporate profit margin expansion.</li> </ol>	indicators_definition.txt monthly_inflation_data.csv

Table 7: Examples for five distinct analysis categories from the UnidataBench benchmark. Each task includes a goal, several insights and some necessary data files.

- formats (e.g., CSV for tabular, JSON for nested) are utilized to plausibly represent the dataset.
- Relevance:** Assesses whether the resulting insight is a direct and pertinent answer to the task's primary question. The focus is on ensuring all conclusions contribute purposefully to the ana-

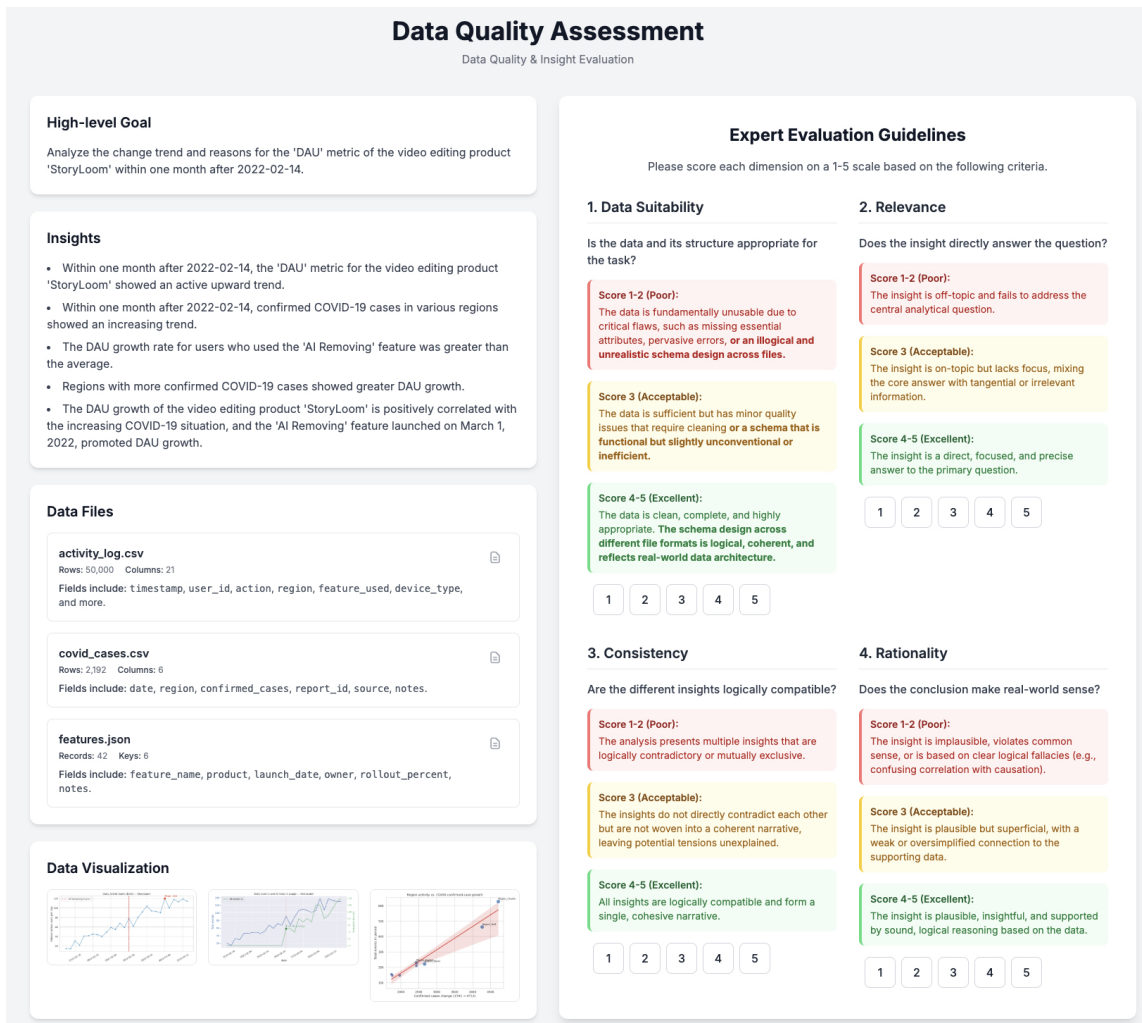


Figure 9: Screenshot of the GUI for Benchmark Quality Assessment (Human Expert Scoring)

lytical objective, rather than being sidetracked by tangential discoveries.

- **Consistency:** Evaluates the logical coherence of the analysis. This ensures that insights derived from diverse data sources (e.g., CSV, DB, NoSQL, TXT) are free from internal contradictions and are not mutually exclusive.
- **Rationality:** Assesses the real-world plausibility of the derived insights. This verifies that each conclusion is factually grounded and could manifest in reality, rather than being an artifact of flawed reasoning or representing an impossible scenario.

## B.2 Evaluation Results

We conducted a quality assessment of UniDataBench using evaluations from data analysis experts and the G-Eval (Liu et al., 2023) methodology. The same experts also evaluated tasks from InsightBench using our framework.

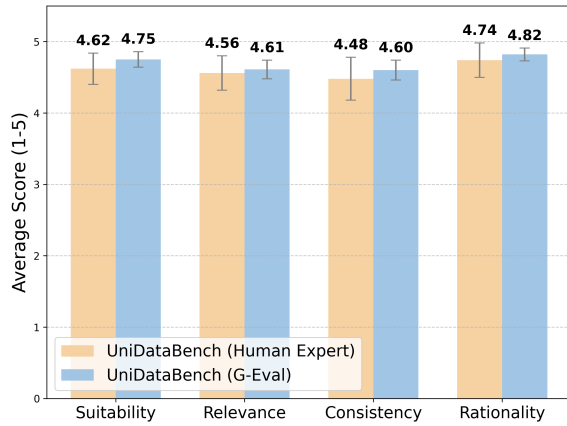
As shown in Figure 10, the evaluation confirms UniDataBench’s high quality and its superior performance against InsightBench, particularly in task relevance.

## C Detailed Experimental Setup

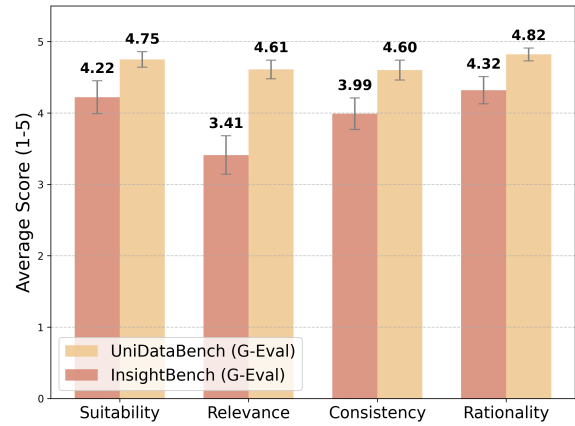
### C.1 Baselines

We benchmarked the following data analytics agents on UniDataBench:

- **CodeGen (Majumder et al., 2025):** CodeGen generates the entire analysis code in a single step based on the provided context, subsequently deriving insights and a workflow summary from the execution results.
- **ReAct (Yao et al., 2023):** ReAct solves tasks through an iterative process of multi-turn thought and action, where it incrementally generates and executes code to explore the data and uncover insights.



(a) UniDataBench Quality Assessment.



(b) Comparative Analysis with InsightBench.

Figure 10: Quality assessment of UniDataBench. (a) UniDataBench Quality Assessment comparing Human Expert and G-Eval evaluations. (b) Comparative Analysis of UniDataBench and InsightBench based on G-Eval.

- **Data-to-Dashboard** (Zhang and Elhamod, 2025): Data-to-Dashboard is a modular multi-agent system that automates the generation of dashboards to produce insights supported by visualizations.
- **Pandas Agent** (LangChain, 2024): Pandas Agent is a LangChain-based agent, generates and executes Python code over a given data frame to produce direct answers to user queries as insights.
- **DABstep Baseline Agent** (Egg et al., 2025): The baseline agent introduced with the DABstep benchmark. It first inspects files in the working directory using a Python tool to plan its analysis path, then follows the ReAct paradigm to iteratively solve the task until a final answer is produced.
- **AgentPoirot** (Sahu et al., 2025): AgentPoirot is the baseline agent from InsightBench, which adopts a question-driven paradigm that first generates root questions and then formulates follow-up questions to delve deeper for insights. Since these agents natively support only CSV files, we adapted them to handle our multiple data sources for a comprehensive comparison.
- **ReActInsight**: ReActInsight is our proposed agent for analysing diverse data sources. It performs end-to-end analysis over diverse data formats by automatically discovering cross-source linkages, decomposing goals, and generating robust, self-correcting code to extract actionable insights.

## C.2 Evaluation Metrics

Following other data analysis studies like InsightBench (Sahu et al., 2025), we evaluate agent’s performance using the G-Eval metric with summary-level and insight-level.

- **Summary-level evaluation**: we generate a comprehensive summary of insights provided by the agent. This summary is then compared against a ground-truth summary using the G-Eval score, which quantifies the overall coherence and correctness of the agent’s output.
- **Insight-level evaluation**: We conduct a more granular assessment based on individual findings. Let  $G$  be the set of ground-truth insights and  $I$  be the set of insights generated by the agent. To ensure the rigor of this evaluation and prevent performance inflation through over-generation, all agents were subject to strict output constraints. By standardizing the number of exploration steps and search depth, we ensured that each method generated an identical number of insights, thereby facilitating a fair comparison under a uniform output scale. For each ground-truth insight  $g \in G$ , we identify the agent-generated insight  $i \in I$  that maximizes the G-Eval similarity. The insight-level score is calculated as the average of these maximum scores across all ground-truth insights:

$$score = \frac{1}{|G|} \sum_{g \in G} \max_{i \in I} \text{G-Eval}(g, i)$$

where  $|G|$  represents the total number of ground-truth insights, and  $G(g, pred\_insight)$  denotes

the G-Eval score between a ground-truth insight  $g$  and an agent-generated insight  $pred\_insight$ .

This dual-level evaluation approach allows us to thoroughly assess both the broad understanding and the precise identification capabilities of our agent in complex data analysis tasks.

### C.3 Reproducibility Details

To ensure the full reproducibility of the ReActInsight framework, we provide a comprehensive summary of the configurations used across the entire pipeline. Table 8 details the model assignments, prompt references, and key hyperparameters for each stage, ranging from multi-source data exploration to final insight synthesis.

## D More Experimental Results

### D.1 Human Evaluation of ReActInsight

To supplement and validate the automatic G-Eval metrics, we also organized a small-scale yet rigorous human evaluation, the final results of which will be summarized in Table 9. During the task selection phase, we randomly sampled 30 questions from the complete UniDataBench test set, ensuring a balanced distribution of difficulty with 10 questions from each of the Easy, Medium, and Hard categories. We invited three senior data analytics practitioners to serve as independent evaluators, responsible for scoring the “Insights” and final “Summary” outputs generated by each agent. The evaluation process was structured around the following four core dimensions:

- **Correctness:** Assessing the fidelity of numbers and facts in the generated content to the source data.
- **Insightfulness:** Evaluating whether the output goes beyond surface-level information to provide deeper perspectives and analysis.
- **Coherence & Clarity:** Focusing on logical flow, readability, and the absence of internal contradictions.
- **Similarity to GT:** Measuring the semantic closeness of the generated content to the gold-standard answers provided by UniDataBench.

### D.2 Hyperparameter Study

**Backbone LLM Performance** We conducted a hyperparameter study to assess the impact of the backbone LLM on performance, comparing gpt-4o, gpt-4-turbo, gpt-3.5-turbo, and llama-4-scout. As shown in Table 13, the results confirm that the

choice of LLM is a key factor. The gpt-4o variant consistently achieved the best results, securing the highest average scores in both Insight-level (0.4750) and Summary-level (0.5792) evaluations. While performance generally scaled with model capability, we noted that llama-4-scout (0.4618) was highly competitive, outperforming gpt-4-turbo (0.4547) on the average Insight-level score. Nevertheless, the dominant performance of gpt-4o, especially on medium and hard difficulty tasks, validates its selection as the most effective backbone for ReActInsight.

**Schema Matching Parameters** We conducted ablation studies to validate our default configuration ( $\theta = 0.5, w_n = 0.6, w_v = 0.4$ ). **Threshold  $\theta$ :** As shown in Table 10, performance is stable within  $\theta \in [0.3, 0.6]$ . However, setting  $\theta = 0.7$  causes a sharp decline in Recall to 41.0%, as legitimate associations are filtered out by the overly restrictive threshold.

**Weights  $w_n/w_v$ :** Table 11 demonstrates that the system is robust when the name similarity weight  $w_n$  is  $\geq 0.6$ . When the value similarity weight  $w_v$  exceeds 0.6, Recall drops significantly to 42.4%, confirming that column names provide a more reliable signal for association than raw value overlaps.

**Name Similarity Metrics** In our methodology, we define the name similarity component  $Sim_{name}$  as a binary exact match indicator ( $\in \{0, 1\}$ ). To justify this choice, we compared it against two commonly used fuzzy matching metrics: Levenshtein and Jaro-Winkler distances. As shown in Table 12, while all three metrics yield comparable Recall, the fuzzy matching approaches introduce significant noise (False Positives), reducing Precision to 85.8% and 65.6% respectively. These results validate that the Exact Match strategy is superior for maintaining the high precision required for joint-hint generation in UniDataBench.

### D.3 Case Study

To evaluate the quality of insights generated, we conduct case study against the strongest baseline, AgentPoirot, across four benchmark tasks, as shown in Table 14. ReActInsight demonstrates superior performance in two key areas.

**Factual Accuracy and Correct Attribution.** Baseline models frequently generate factually incorrect or misattributed conclusions. For example, AgentPoirot incorrectly links peak profits to

Stage	Sub-step	Model / Engine	Prompt Ref.	Key Hyperparameters
<b>I. Exploration of Multi-Source Data and Discovery of Cross-Source Associations</b>	Hetero-Schema Extraction	Deterministic Alg.	—	Sampling $n_{rows} = 1000$
	Text Content Summary	GPT-4o-mini	—	$T = 0$
	MetaGraph Construction	Deterministic Alg.	—	Unique Alias Assignment
	EntityGraph Generation	Deterministic Alg.	—	$w_n = 0.6, w_v = 0.4, \theta = 0.5$
<b>II. Hierarchical Planning</b>	Join-Hint Generation	Deterministic Alg.	—	Top- $k$ Selection
	Sub-question Generation	GPT-4o-mini (Cascaded)	Prompt 1	$T = 0, max\_questions = 10$
<b>III. Code Generation and Execution</b>	Follow-up Generation	GPT-4o-mini (Cascaded)	Prompt 2	$T = 0, max\_questions = 10$
	Code Generation	GPT-4o	Prompt 3 & 4	$T = 0, n\_retries = 3$
<b>IV. Summary Insights</b>	Result Interpretation	GPT-4o-mini (Cascaded)	—	$T = 0$
	Summary Generation	GPT-4o-mini (Cascaded)	Prompt 5	$T = 0$

Table 8: Comprehensive implementation details for ReActInsight, including model configurations, prompt citations, and hyperparameters for each stage.

Evaluation Method	Insight-level					Summary-level				
	D1	D2	D3	D4	Avg.	D1	D2	D3	D4	Avg.
Human Evaluation	0.93	0.87	0.91	0.63	0.84	0.95	0.90	0.94	0.67	0.87
LLM Evaluation	0.95	0.97	0.94	0.48	0.84	0.96	0.92	0.95	0.58	0.85

Table 9: Human & LLM Evaluation: Individual Scores and Mean (0–1) for D1 Correctness, D2 Insightfulness, D3 Coherence and Clarity, and D4 Similarity to GT.

$\theta$	Precision	Recall	F1
0.3	94.6%	97.2%	95.9%
0.4	94.6%	96.5%	95.5%
0.5	99.3%	96.5%	97.9%
0.6	99.3%	96.5%	97.9%
0.7	100.0%	41.0%	58.1%

Table 10: Ablation of Threshold  $\theta$ .

$w_n/w_v$	Precision	Recall	F1
0.8 / 0.2	99.3%	96.5%	97.9%
0.7 / 0.3	99.3%	96.5%	97.9%
0.6 / 0.4	99.3%	96.5%	97.9%
0.5 / 0.5	96.5%	96.5%	96.5%
0.4 / 0.6	93.8%	42.4%	58.4%

Table 11: Ablation of Weights  $w_n/w_v$ .

marketing campaigns that actually drove sales in a different quarter, demonstrating a confusion between correlation and causation. It also analyzes an irrelevant metric (revenue instead of order volume) in another task. ReActInsight avoids these pitfalls. Its **Hierarchical Planning** mechanism decomposes the main objective into logically coherent sub-questions, ensuring the analysis remains focused on the correct metrics and causal relation-

Metric	Precision	Recall	F1
Exact Match	99.3%	96.5%	97.9%
Levenshtein	85.8%	96.5%	90.8%
Jaro-Winkler	65.6%	96.5%	78.1%

Table 12: Ablation of Name Similarity Metrics.

ships, thus yielding factually accurate insights.

**Granular Root Cause Analysis.** ReActInsight excels at identifying specific root causes, whereas the baseline often provides only superficial observations. In the vehicle delivery analysis, Agent-Poirot notes a drop in a conversion rate but fails to identify the specific product model responsible. This failure arises from an inability to synthesize data across multiple sources. In contrast, ReActInsight’s **Multi-Source Data Exploration** and **Entity-Graph** successfully link the sales funnel data with a separate product database. This allows it to pinpoint the issue precisely to the "VOR-TEX GT" model and quantify its conversion rate collapse from 45% to 22%. Similarly, it moves beyond a vague hypothesis in the StoryLoom analysis to identify a specific successful feature, the "Fill Light Filte", and provides detailed usage statistics across user segments, offering a far more actionable and granular insight.

Agent	Insight - level Scores (G-Eval)				Summary - level Scores (G-Eval)			
	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.
ReActInsight (gpt-4o)	<b>0.5030</b>	<b>0.4704</b>	<b>0.4517</b>	<b>0.4750</b>	0.5984	<b>0.5789</b>	<b>0.5605</b>	<b>0.5792</b>
ReActInsight (gpt-4-turbo)	0.4659	0.4688	0.4293	0.4547	<b>0.6032</b>	0.5441	0.5484	0.5652
ReActInsight (gpt-3.5-turbo)	0.4327	0.4561	0.4489	0.4459	0.5517	0.5361	0.5365	0.5414
ReActInsight (llama-4-scout)	0.4792	0.4708	0.4355	0.4618	0.5720	0.5386	0.5273	0.5460

Table 13: Comparison of our agent performance using different LLMs on UNIDATABENCH.

UnidataBench Insights (GT)	AgentPoirot’s Insights (Baseline)	ReActInsight’s Insights (Ours)
<b>Task Goal: Analyze the change in DAU metric for ‘StoryLoom’ in Q2 2024</b>		
The increase in DAU in Q2 2024 is mainly attributed to the positive impact of the ‘Fill Light Filter’ feature, while the ‘Vintage Filter’ feature had no significant effect on DAU growth.	The lack of immediate DAU increase suggests that the new filters may require more time for user adoption or additional marketing efforts.	The “Fill Light Filter” launched in Q2 2024 had the highest positive impact on the ‘DAU’ metric for the ‘StoryLoom’ product, with total usage of 77,183 among intermediate account age users, 73,535 among new users, and 54,522 among long-term users.
<b>Task Goal: Identify the peak sales quarter and peak profit month for the e-commerce ‘MarketGrid’ in 2018</b>		
The peak sales and profit periods are closely linked to targeted marketing activities: the ‘Home_Refresh’ and ‘Tech_Launch’ campaigns drove category-specific sales growth in Q1, while the ‘Winter_Style’ campaign in November was instrumental in maximizing profit through the Clothing category.	November 2018 stood out as the most profitable month for MarketGrid, achieving a total profit of \$270,000. This peak profit period was influenced by targeted marketing campaigns such as “Tech_Launch” and “Home_Refresh.”	The top-performing marketing activities during the peak sales quarter were ‘Tech_Launch_Mar’ and ‘Home_Refresh_Feb’, contributing significantly to sales. In contrast, the peak profit month (November 2018) was driven by ‘Winter_Style_Launch’.
<b>Task Goal: Analysis of MoM Changes in the Number of Leads Converted to ‘Vehicle Delivery’ Created in August</b>		
The significant drop in conversion rate from “test drive” to “order” for the “VORTEX GT” model is a key driver behind the overall decline in vehicle deliveries, suggesting targeted investigation and intervention for this model could improve outcomes.	The primary reason for this decline was a significant drop in the conversion rate from the ‘test drive’ to the ‘order’ stage [Note: There is no analysis of the car model here.], which fell from 65% to 52% month-over-month across all models.	The primary driver for the overall decline in vehicle deliveries in August was a significant drop in the conversion rate from “test drive” to “order”, specifically for the “VORTEX GT” model, whose conversion rate at this stage plummeted from 45% in July to 22% in August.
<b>Task Goal: Analyze the change in product order volume on the e-commerce platform ‘ShopNext’ between 2021 and 2020</b>		
The increase in order volume during these months is strongly correlated with a substantial rise in the number of promotions offered, suggesting that promotional activities were a key driver.	Total revenue on ShopNext grew substantially by over 14% from approximately \$70,989 in 2020 to \$81,161 in 2021, supported by a notable surge in coupon usage where orders using coupons more than doubled from 119 to 271.	Promotional activities in 2021, including increased counts and diversified types like ‘HolidayPromo’ and ‘MegaSale’, led to a 40% higher peak monthly order volume compared to 2020, with 140 orders in November and December in 2021 compared to 100 orders in December 2020.

Table 14: Comparative analysis of insights from ReActInsight and AgentPoirot. The baseline’s failures (highlighted in red), such as factual inaccuracies and superficial conclusions, stem from its **inability to synthesize information across multiple data sources**. In contrast, ReActInsight (highlighted in green) leverages its multi-source exploration and linkage discovery mechanism to **effectively integrate heterogeneous data**. This capability enables it to produce quantitatively precise, granular, and robust analytical insights, demonstrating superior analytical depth.

## E Prompts

### Prompt 1: Prompt for the agent in raising questions from multi sources data.

```
You are a senior data-analytics team lead.
Your job is to craft answerable, high-impact questions for the team,
given several heterogeneous datasets (csv/sql/nosql...).

Rules:
1. Only output what the user can copy-paste into their notebook.
2. NEVER add explanations outside the required <question> tags.
3. If the user sets question_type, bias the questions toward that type
   (e.g. descriptive / diagnostic / predictive / prescriptive).

We have SEVERAL datasets to analyse.
<context>
{context}
</context>

Our business goal is:
<goal>
{goal}
</goal>

Below is the merged schema and sample rows from all data sources:
<schema>
{schema}
</schema>

You are asked to propose insightful questions for the team. These questions should be of a {
question_type} nature.

Guidelines:
* Ask questions that help reach the stated goal.
* Consider relationships across datasets (joins, correlations, etc.).
* Each question must be realistic given ONLY the columns shown above.
* Do NOT number the questions.
* Generate at most {max_questions} questions and STOP.
* Wrap every question inside <question>...</question> tags.
### Response
```

### Prompt 2: Prompt for the agent in raising followup questions from multi sources data.

```
You the manager of a data science team whose goal is to help stakeholders within your company extract
actionable insights from their data.
You have access to a team of highly skilled data scientists that can answer complex questions about
the data.
You call the shots and they do the work.
Your ultimate deliverable is a report that summarizes the findings and makes hypothesis for any trend
or anomaly that was found.

Hi, I require the services of your team to help me reach my goal.

<context>{context}</context>

<goal>{goal}</goal>

<schema>{schema}</schema>

<question>{question}</question>

<answer>{answer}</answer>

Instructions:
* Produce a list of follow up questions explore my data and reach my goal.
* Note that we have already answered <question> and have the answer at <answer>, do not include a
question similar to the one above.
* Explore diverse aspects of the data, and ask questions that are relevant to my goal.
* You must ask the right questions to surface anything interesting (trends, anomalies, etc.)
```

- \* Make sure these can realistically be answered based on the data schema.
- \* The insights that your team will extract will be used to generate a report.
- \* Each question that you produce must be enclosed in <question>content</question> tags.
- \* Each question should only have one part, that is a single '?' at the end which only require a single answer.
- \* Do not number the questions.
- \* You can produce at most {max\_questions} questions.

### Prompt 3: Prompt for the agent in coding from multi sources data.

You are a meticulous senior data-scientist.  
 When the user asks for code, you must output **only ONE executable Python code block** that fulfils all requirements no extra commentary, markdown, or explanation.

If something would fail at run-time, handle it inside the code (raise a clear Exception). Be concise yet complete.

### Instruction

Business goal:

```
<goal>
{goal}
</goal>
```

Question to answer:

```
<question>
{question}
</question>
```

You have access to SEVERAL datasets whose schemas & samples follow:

```
<schema>
{schema}
</schema>
```

Raw file paths (csv / sqlite / json ...):

```
<database_paths>
{database_paths}
</database_paths>
```

Coding requirements:

1. Load whichever datasets / tables are needed to solve the question.  
 (pandas for csv/json, sqlite3+pandas for sqlite, json module for jsonl ...)
2. Perform the analysis that directly answers the question.
3. Create **one or more** plots (use seaborn / matplotlib).  
 Name them sequentially: plot\_1.jpg, plot\_2.jpg, ...
4. For **each** plot i save four companion files in the working directory:
  - stat\_i.json      key statistics used in your answer
  - x\_axis\_i.json    =< 50 representative x-values
  - y\_axis\_i.json    =< 50 representative y-values
  - plot\_i.jpg       the image itself (already saved in step 3)
 Every json must contain keys "name", "description", "value".
7. Make the script deterministic; set random seeds where randomness appears.
8. All files are saved under the {output\_dir} directory

Output rules (IMPORTANT):

- \* Return exactly ONE code block that starts with '''python and ends with '''.
  - \* Do NOT generate code blocks for other languages.
  - \* No text outside the single code block.
  - \* The code must execute all logic immediately at the top level; do not just define functions.

### Response

#### Prompt 4: Prompt for the agent in joining from multi sources data.

```
### Instruction

Business goal:
<goal>
{goal}
</goal>

Question to answer:
<question>
{question}
</question>

You have access to SEVERAL datasets whose schemas & samples follow:

<schema>
{schema}
</schema>

Raw file paths (csv / sqlite / json xxx):

<database_paths>
{database_paths}
</database_paths>

You must strictly follow the join_hints to perform data merging before creating any plots:
<join_hints>
{join_hints}
</join_hints>

Coding requirements:
1. Load whichever datasets / tables are needed to solve the question.
   (pandas for csv/json, sqlite3+pandas for sqlite, json module for jsonl)
   If <require_multi_join>true</require_multi_join>, you must join > 2 tables using the hints
   above.
2. Perform the analysis that directly answers the question.
3. Create one or more plots (use seaborn / matplotlib).
   Name them sequentially: plot_1.jpg, plot_2.jpg,xxx
4. For each plot i save four companion files in the working directory:
   stat_i.json      -- key statistics used in your answer
   x_axis_i.json    -- =< 50 representative x-values
   y_axis_i.json    -- =< 50 representative y-values
   plot_i.jpg       -- the image itself (already saved in step 3)
   Each json must contain keys "name", "description", "value".
5. Make the script deterministic; set random seeds where randomness appears.
6. Save all files under the directory {output_dir}.

Additional merging and processing requirements:
1. Standardize key values before merging:
   - For date keys: 'df["date"] = pd.to_datetime(df["date"], errors="coerce").dt.normalize()'.
   - For category keys (e.g., province): 'df["province"] = df["province"].astype(str).strip().lower()'.
   - Apply similar normalization (strip + lower) for other categorical keys like city or country.
2. Use a single merge result variable:
   - Use a consistent variable name (e.g., 'merged') for the merged result throughout the script. All
     statistics, plotting, and JSON exports must be based on this variable to avoid confusion.
3. Handle column name conflicts automatically:
   - When merging tables with overlapping column names, detect and resolve suffixes (e.g., '_x', '_y'
     ') using regex or column relationships.
   - Standardize critical columns (e.g., 'confirmed_cases' or 'dau') by renaming them back to their
     original names.
4. Ensure the merge result is non-empty and provide fallback:
   - After an inner join, ensure the result is non-empty using 'assert not merged.empty' or
     equivalent checks.
   - If empty, log a clear warning and automatically downgrade to an outer join to avoid data loss,
     while also providing diagnostics on key consistency.
5. Log key debugging information:
   - Log shapes of individual datasets and the merged result.
   - Log data types and sample values of keys used for joining.
```

- For outer joins, optionally enable 'indicator=True' to examine merge diagnostics.
- 6. Ensure safe paths and filenames:
  - Generate safe filenames using slugification and length limits: 'safe = re.sub(r"[^\w]", "\_", text)[:60]'.
  - Use 'pathlib.Path' for cross-platform path management.
- 7. Validate completeness of plots and JSON:
  - Ensure required columns for plotting are present and non-null before plotting. Use 'pd.to\_numeric(errors="coerce")' if needed and report missing ratios.
  - Write dates as strings in JSON ('dt.strftime("%Y-%m-%d)').
  - Ensure consistent output structure for JSON files: 'stat\_x.json', 'x\_axis\_x.json', 'y\_axis\_x.json'. All fields must have clear meanings and non-empty values.
- 8. Limit resource usage and output sizes:
  - For images: Set 'dpi =< 300', long edge =< 4000 pixels, and file size =< 1 MB (adjust 'dpi' or use 'bbox\_inches="tight"' if needed).
  - For JSON files: Limit entries and size (e.g., =< 10 KB). Use sampling or truncation if necessary.
- 9. Manage connections and file handles carefully:
  - Use 'with' statements for database and file operations to ensure automatic resource cleanup.
  - Specify necessary parameters (e.g., 'dtype', 'parse\_dates') when reading files to minimize type ambiguities.
- 10. Handle errors and provide diagnostic feedback:
  - Catch critical exceptions (e.g., 'KeyError', 'ValueError', 'TypeError', 'pd.errors.ParserError') and provide clear error messages with suggestions:
    - Missing columns: List available and expected column names.
    - Key mismatches: Indicate missing key values in each table.
    - Type issues: Recommend standardizing dates as 'datetime' and categories as lowercase strings.
  - On failure, return a non-zero exit code or log 'FAILED' to facilitate retries; include the latest exception summary in retry hints.

Optional but strongly recommended:

- Clearly label units and time ranges in legends, axis labels, and titles.
- Perform basic sanity checks on numerical columns (e.g., negative values or extreme outliers) and document handling methods (e.g., filling, truncation, or removal).
- Drop rows with NaN or use pairwise complete observations for correlation analyses; report the sample size used.

Output rules (IMPORTANT):

- \* Return exactly **one** code block that starts with 'python' and ends with ''.
- \* Do **NOT** generate code blocks for other languages.
- \* No text outside the single code block.
- \* The code must execute all logic immediately at the top level; do not just define functions.

#### Prompt 5: Prompt for the agent in summarizing multi source data.

I require the services of your team to help me reach my goal.

<context>{context}</context>

<goal>{goal}</goal>

<history>{history}</history>

Instructions:

- \* Given the context, goal, and the full list of <question\_i><answer\_i> history pairs above, generate the 3 most important insights.
- \* Present the insights as a numbered list (e.g., "1. ... 2. ... 3. ..."), with each point focusing on a key, concise conclusion in plain language.
- \* Summarize highlights and findings; do not provide action recommendations.
- \* Make each insight as quantitative as possible, aggregating the findings.
- \* Enclose each insight within the <insight></insight> tag.