I2I - STRADA – Information to Insights via Structured Reasoning Agent for Data Analysis

Anonymous ACL submission

Abstract

Recent advances in agentic systems for data analysis have emphasized automation of insight generation through multi-agent frameworks, and orchestration layers. While these systems effectively manage tasks like query translation, data transformation, and visualization, they often overlook the structured reasoning process underlying analytical thinking. Reasoning large language models (LLMs) used for multi-step problem solving are trained as general-purpose problem solvers. As a result, their reasoning or thinking steps do not adhere to fixed processes for specific tasks. Real-world data analysis requires a consistent cognitive workflow: interpreting vague goals, grounding them in contextual knowledge, constructing abstract plans, and adapting execution based on intermediate outcomes. We introduce I2I-STRADA (Information-to-Insight via Structured Reasoning Agent for Data Analysis), an agentic architecture designed to formalize this reasoning process. I2I-STRADA focuses on modeling how analysis unfolds via modular sub-tasks that reflect the cognitive steps of analytical reasoning. Evaluations on the DABstep and DABench benchmarks show that I2I-STRADA outperforms prior systems in planning coherence and insight alignment, highlighting the importance of structured cognitive workflows in agent design for data analysis.

1 Introduction

Real-time and ad-hoc data analysis in enterprise environments is a complex task as data tends to be heterogeneous, non-standard and lacking quality. This is due to the diversity of systems, the variability of human input, and the continuous evolution of business processes (Rozony et al., 2024). As a result, data typically undergoes pre-processing before any analytical queries can be executed. Traditionally, various data harmonization techniques have been used to address challenges arising from data in multiple formats, incompleteness, and missing values (Cheng et al., 2024). Similarly, while dealing with multiple sources, the same entity can have conflicting attributes due to naming conventions, out-of-date data etc., necessitating a truth discovery process before proceeding with any further analysis (Li et al., 2015). Furthermore, as organizational processes expand, changes to the data structures and corresponding analytical requirements result in significant re-engineering efforts (Putrama and Martinek, 2024), (Bandara et al., 2023). Thus, it is imperative that data analytics systems incorporate procedural knowledge and are knowledge-driven (Bandara et al., 2023).

LLMs are naturally suited to address these challenges, given their ability to understand unstructured data, infer context, and adapt to evolving semantics across heterogeneous sources. In Santos et al. (2025), the authors focus on developing a system for data harmonization of tabular data sources using LLMs. In Chen et al. (2023), the authors leverage representation learning techniques for multi-modal data discovery and subsequently query decomposition for planning and execution. In other similar works like Wang et al. (2025), Wang and Li (2025) the authors formalize a set of multi-modal semantic operators which are composed into execution pipelines to answer a query. These methods focus on tasks like query translation or data transformation and rely on LLM based reasoning to perform the tasks effectively. Data analysis is however a process that involves a formal set of several cognitive tasks such as understanding the query/problem, careful planning on collecting and examining the necessary data, iteratively updating the data for analysis based on examination and finally performing the most suitable statistical analysis and communicating it (Grolemund and Wickham, 2014). Relying on general purpose reasoning abilities of LLMs produces sub-optimal results. For instance, as shown in Song et al. (2025), LLMs frequently fail at basic compositional reasoning—even in relatively simple multi-hop scenarios. This limitation is especially critical in data analysis, where compositional reasoning is fundamental for tasks like integrating diverse data points, chaining logic, and deriving high-level insights. We therefore propose that effective data analysis agents must be guided by structured reasoning workflows to produce reliable and goal-aligned analysis.

We introduce **I2I – STRADA** that enables going from **I**nformation to **I**nsights via a **St**ructured **R**easoning **A**gent for **D**ata **A**nalysis. The agent follows a workflow composed of multiple specialized sub-tasks, each responsible for a distinct aspect of reasoning and planning. We discuss related work and key limitations in the next section followed by the details of our approach. We evaluate I2I-STRADA on DABstep (Egg et al., 2025) and DABench (Hu et al., 2024) data analysis benchmarks that focus on scenarios where agents must operate under procedural constraints and deliver insights. Results show significant improvements in planning quality and alignment with analytical objectives, underscoring the value of structured reasoning in agent design.

2 Related work

Recent contributions to data analysis agents can be categorized into two main streams: (1) those focused on planning, and (2) those aimed at building agents for end-to-end analytics platforms.

2.1 Planning focused approaches

DatawiseAgent (You et al., 2025), employs a (Depth First Search) DFS like planning and incremental code execution mechanism along with self-debugging capabilities. This approach is proposed to address the complexities involved in solution exploration and ensuring the result of code execution is consistent with the corresponding planning step. However, the lack of global planning can result in inconsistencies in the trajectories generated on the fly. DataInterpreter (Hong et al., 2024), aims to produce global execution steps by generating a graph of tasks for a given problem. The tasks are chosen from a list of fine-grained task definitions most seen in data processing and data science pipelines. However, both the methods above do not incorporate a data understanding step, thereby increasing the chances of erroneous interpretation of data elements and domain-specific computations.

2.2 Agents for end-to-end analytics platforms

Few approaches focus on complete business intelligence (BI) workflows and position the agents or agentic frameworks as platforms for data analysis (Weng et al., 2025, 2024; Ma et al., 2023) — combining query interfaces, tool libraries, and visualization modules. Weng et al. (2025, 2024) are broader frameworks that include offline preprocessing stages to gather metadata for data understanding and schema mapping. Hong et al. (2024); Weng et al. (2025, 2024) focus on having modules that are specific to stages of insight generation such as SQL generation, data cleaning, chart generation, etc. These platform-centric agents prioritize user workflows — handling tasks like prompt interfaces, chart rendering, and multi-modal output —while treating reasoning as a black-box module abstracted behind orchestration layers. Even in works focusing on insight generation (Weng et al., 2024; Sahu et al., 2025; Ma et al., 2023), the reasoning process is treated as a sequence of Q&As on the data. While this is strong in guiding exploration, they lack explicit structured planning and execute using flat reasoning paths.

In particular, existing methods fall short in key areas that our work aims to address: (1) insufficient data exploration during early planning, (2) failure to detect procedural constraints as per the business rules (in the vastness of the context Shi et al., 2023), and (3) misalignment between planning and execution.

3 Approach

Our design is grounded in two key tenets: (1) progressive abstraction, where we preserve critical information while filtering noise at each stage; (2) multi step refinement, using a two-stage planning process to iteratively improve reasoning quality. This structured and modular approach enables robust and interpretable agent behavior in complex analytical settings.

In this section, we present the architecture and workflow of I2I-STRADA, detailing how each component contributes to a structured reasoning pipeline (see Figure 1).

Goal construction: The initial step involves inferring the user's analytical goal directly from the given query. The agent constructs its "beliefs" about the data by extracting information solely from the query itself. This early identification of

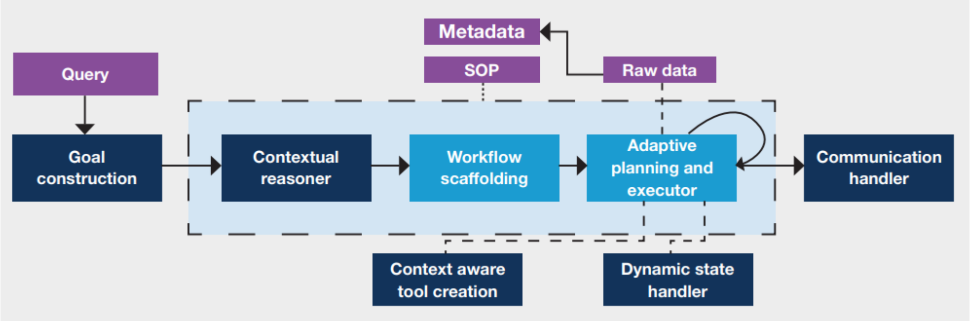


Figure 1: The workflow of sub-tasks for I2I-STRADA. A user query is first translated into a contextualized goal through a structured goal construction phase. This involves understanding the core analytical intent, identifying key entities and constraints, and outlining a preliminary solution approach—derived solely from the query. The goal is then refined and grounded using metadata and Standard Operating Procedures (SOPs) to ensure alignment with available data structures and domain-specific norms. Once contextualized, the goal enters a two-stage planning process. The workflow scaffolding module defines a high-level strategy, which guides the adaptive planner and executor—an iterative component that refines actions based on live data interaction. This core reasoning loop is supported by modules for dynamic tool creation and execution state management, while a communication handler delivers the final, user-aligned output in the required format. Refer to Algorithm 1.

the problem type is essential for guiding subsequent data exploration, and building belief from scratch ensures that the agent considers every detail relevant to the request. The outcome of this step consists of:

- Question understanding - Understand the core intent of the user
- Entity extraction - identifying relevant data points, dimensions, or concepts mentioned in the query;
- Generic solution approach - outlining a preliminary high-level strategy; and
- Constraints - detailing any specific limitations or conditions provided.

Refer to Appendix A for the prompt.

Contextual reasoner: Acting as a bridge between the initial understanding and a plan of action, the module grounds the analysis using contextual information. It references metadata of the data systems and applicable SOPs to refine the solution approach derived from the inferred goal and constructed belief. Utilizing these inputs helps ensure the resulting plan is not only aligned with the user's request but also key procedural requirements and constraints. Refer to Appendix B for the prompt.

Two Planning stages:

- **Workflow scaffolding:** The Workflow Scaffolding is the generator of a global plan of action. This plan is formulated before the agent interacts with the actual data. This high-level plan serves as the foundational workflow or 'scaffold' that guides the adaptive executor, allowing for dynamic execution while ensuring the analysis adheres to the defined overall problem-solving approach. Refer to Appendix C for the prompt.
- **Adaptive planning and executor:** It is an iterative module that generates execution-level plans aligned with the scaffolded workflow. It dynamically adjusts subsequent steps based on prior execution results, including actual data exploration and intermediate outcomes. This adaptability is necessary as complex tasks require data interaction to inform planning. The adaptive planner ensures alignment with the scaffold and tracks plan status iteratively. The execution involves writing code snippets in Python and executing them in a sandbox. The context of the execution carries through all the iterations. Refer to Appendix D, E for the prompts.

Context aware tool creation: The module utilizes metadata (types of data sources involved) and instructions (how to process the data, recommended libraries to use etc.) to dynamically create data processing tools and scripts on the fly. This is key to analyzing heterogeneous data sources effectively and extends the solution's applicability to Bring Your Own (BYO) data sources.

Dynamic State Handler: Acts as the agent's dynamic working memory, essential due to adaptive execution planning. It maintains the execution context across iterations (includes updating variables) and provides runtime debugging capabilities.

Communication Handler: Manages the presentation of results, ensuring they address user goals and conform to required formatting. It converts raw output based on guidelines or query context, making information clear and relevant.

4 Evaluation

We evaluate our solution on two recent benchmark datasets to validate the generalizability of the approach. The closest benchmark that aligned with the idea of procedural knowledge driven multi-source data analysis was DABstep (Egg et al., 2025). The second benchmark dataset is DABench (Hu et al., 2024). This dataset has a stronger focus on statistics and data science. These two datasets provide a wide spectrum of concepts to test the efficacy of agentic approach for data analysis.

4.1 Results on DABstep benchmark

The DABstep dataset (Egg et al., 2025), developed by Adyen in collaboration with Hugging Face contains tasks that test reasoning over financial and operational data. It comprises over 450 tasks that simulate real-world analytical workflows common in financial services, such as interpreting transaction records, navigating policy documentation, and reconciling structured and unstructured data sources.

We used Anthropic's Claude 3.5 Sonnet in our agentic workflow. Our agent outperforms several SOTA data science agents as well as baselines built using ReACT (Yao et al., 2023) framework with an accuracy of 80.56% on easy tasks and 28.04% on hard tasks. Refer to table 1.

Where our agent succeeds:

- Improved planning and failure handling when writing code
- Sensitive to rules mentioned in the SOP
- Planning without overthinking (Easy tasks require simple plans)

Algorithm 1 I2I-STRADA: Structured Reasoning Agent for Data Analysis

```
Require: User query $Q$, Raw data sources $D$, SOPs $S$, Instructions for handling data sources $I$
Ensure: Result for the user query in natural language $R$

    I. (Offline step): Prepare metadata from $D$, $S$ to support structured reasoning

    $M \leftarrow$ CREATEMETADATA($D$, $S$)

    Main Procedure:
    I2I-STRADA($Q, M, S, D, I$)

    II. Goal Construction
    Analyze $Q$ and build belief state $B_0$ using question understanding, entities, constraints and solution approach

    III. Contextual Grounding
    Use metadata $M$ and SOPs $S$ to update belief $B_0 \rightarrow B$

    IV. Workflow Scaffolding
    Generate high-level plan $P = \{t_1, t_2, \dots, t_n\}$ based on $B$
    Initialize execution context $C_0$

    V. Adaptive Planning and Execution
    $i \leftarrow 1$
    repeat
        Derive tool/code using $I$, $M$ and $C_{i-1} \rightarrow T_i(D)$
        Execute $T_i(D)$, observe results $r_i$
        Update execution context $C_i$

    Based on $C_i$:
        if $t_i$ complete then
            $i \leftarrow i + 1$
        else
            continue
        end if
    until $i = n + 1$

    VI. Results
    Based on $C_n$, contextualize the results to the user query $Q$ and generate response $R$
    return $R$
```

Agent	Easy Level Accuracy	Hard Level Accuracy	Model Family
I2I-STRADA (Ours)	**80.56%**	**28.04%**	**claude-3-5-sonnet**
DICE	75.00%	27.25%	o3-mini
O4-mini Reasoning Prompt Baseline	76.39%	14.55%	OpenAI o4-mini
Claude 3.7 Sonnet ReACT Baseline	75.00%	13.76%	claude-3-7-sonnet
Gemini Data Science Agent	61.11%	9.79%	Gemini 2.0 Flash
Claude 3.5 Sonnet ReACT Baseline	77.78%	9.26%	claude-3-5-sonnet
Deepseek V3 ReACT Baseline	66.67%	5.56%	Deepseek v3
Llama 3.3 70B ReACT Baseline	68.06%	3.70%	Llama 3.3 70B Instruct

Table 1: Performance comparison on DABstep benchmark

Agent	Accuracy	Model Family
Data Interpreter (Hong et al., 2024)	94.93%	GPT-4o
I2I-STRADA (Ours)	**90.27%**	**claude-3-5-sonnet**
Datawise Agent (You et al., 2025)	85.99%	GPT-4o
Data Interpreter (Hong et al., 2024)	73.55%	GPT-4
AgentPoirot (Sahu et al., 2025)	75.88%	GPT-4
DataLab (Weng et al., 2025)	75.10%	GPT-4

Table 2: Performance comparison on DABench benchmark

Where we see chances to improve:

- The agent seems inconsistent when applying SOP rule related to handling of “Null” values. It correctly interprets empty lists (i.e []) as “Null” always but on several occasions, when a field is explicitly “null”/”None”, it fails to apply this rule. This seems to be an interpretation problem with Claude 3.5 Sonnet as it focuses attention on a single example given in the SOP.

Appendix F presents our agent’s trace on one hard task. The example represents the attention to detail arising out of multi-stage refined planning. The rest of the reasoning traces are available on Hugginface DABstep submissions for reference.

4.2 Results on DABench benchmark

The InfiAgent-DABench benchmark (Hu et al., 2024), is specifically designed to evaluate large language model (LLM)-based agents on end-to-end data science tasks across a variety of real-world domains (Marketing, Finance, Energy etc.). The core of the benchmark is the DAEval dataset, comprising 257 open-ended data analysis questions associated with 52 diverse CSV files collected from public sources. The concepts covered by the tasks include - Summary Statistics, Feature Engineering, Correlation Analysis, Machine Learning, Distribution Analysis, Outlier Detection and Comprehensive Data Preprocessing. The dataset doesn’t have SOPs. We hence provided just the definitions of the tasks given by as SOP input.

The accuracy metric shown in table 2 is accuracy by question (ABQ). The numbers are as reported in the respective papers, and we haven’t attempted to replicate them. Additionally, we have picked only the best results from these papers to compare against.

Where our agent succeeds:

- Single/Multi source, the same workflow without any modifications produces consistently SOTA results.
- The exact nature of the data analysis task doesn’t affect the performance. (Domain specific or pure statistical/data science based)

Where we see chances to improve:

- When applying machine learning algorithms, the choice of hyperparameters often results in different results. This could be corrected by providing an appropriate procedure document.

Appendix G presents our agent’s traces on a hard task.

5 Conclusion

In this work, we have presented an agentic system design to address the multifaceted challenges of data analysis in real-world scenarios. Our approach leverages a structured workflow composed of specialized sub-tasks, each dedicated to a distinct aspect of reasoning and planning. The multi-step context refinement process, supported by contextual tool creation ensures that the agent can handle heterogeneous data sources, perform complex intermediate calculations, and support a wide array of analytical queries.

Our evaluation on the DABstep and DABench benchmarks demonstrates the effectiveness and generalizability of our agent. On DABstep, our agent outperforms other SOTA solutions, particularly excelling in planning and failure handling when writing code and adhering to SOPs. On DABench, our agent shows robustness across diverse domains and data analysis tasks, maintaining high accuracy without modifications to its workflow. Additionally, our approach substantially addresses the reasoning limitations of LLMs in complex analytical scenarios (Shojaee*† et al., 2025).

In conclusion, we believe that this approach can further the development of fine-tuned reasoning models to be used in agentic systems capable of performing comprehensive data analysis.

Limitations

While we have used Anthropic's Claude 3.5 Sonnet, we see that any change in model requires modifications to the prompts that best suit the model. This creates a scalability challenge when evaluating the system across newly released LLMs. Another limitation relates to the volume of metadata provided to the model. As the system scales, selecting the most relevant metadata for a given query becomes critical and requires a dedicated module for efficient and context-aware selection. Additionally, generating new reasoning for analytical paths that have already been explored is often redundant. To maintain consistency and efficiency, it would be beneficial to incorporate a reasoning cache with appropriate retrieval mechanisms.

References

Madhushi Bandara, Fethi A. Rabhi, and Muneera Bano. 2023. A knowledge-driven approach for designing data analytics platforms. *Requirements Engineering*, 28(2):195–212.

Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Samuel Madden, and Nan Tang. 2023. Symphony: Towards natural language query answering over multi-modal data lakes. In *CIDR*, pages 1–7.

Cindy Cheng, Luca Messerschmidt, Isaac Bravo, Marco Waldbauer, Rohan Bhavikatti, Caress Schenk, Vanja Grujic, Tim Model, Robert Kubinec, and Joan Barceló. 2024. A general primer for data harmonization. *Sci. Data*, 11(1):152.

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.

Garrett Grolemund and Hadley Wickham. 2014. A cognitive interpretation of data analysis. *International Statistical Review / Revue Internationale de Statistique*, 82(2):184–204.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, and 8 others. 2024. Data interpreter: An llm agent for data science. *Preprint*, arXiv:2402.18679.

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. *Preprint*, arXiv:2401.05507.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. A survey on truth discovery. *Preprint*, arXiv:1505.02463.

Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. Demonstration of insightpilot: An llm-empowered automated data exploration system. *Preprint*, arXiv:2304.00477.

I Made Putrama and Péter Martinek. 2024. Heterogeneous data integration: Challenges and opportunities. *Data in Brief*, 56:110853.

Farhana Zaman Rozony, MNA Aktar, Md Ashrafuzzaman, and A Islam. 2024. A systematic review of big data integration challenges and solutions for heterogeneous data sources. *Academic Journal on Business Administration, Innovation & Sustainability*, 4(04):1–18.

Gaurav Sahu, Abhay Puri, Juan Rodriguez, Amirhossein Abaskohi, Mohammad Chegini, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vazquez, Nicolas Chapados, Christopher Pal, Sai Rajeswar Mudumba, and Issam Hadj Laradji. 2025. Insightbench: Evaluating business analytics agents through multi-step insight generation. *Preprint*, arXiv:2407.06423.

Aécio Santos, Eduardo H. M. Pena, Roque Lopez, and Juliana Freire. 2025. Interactive data harmonization with llm agents. *Preprint*, arXiv:2502.07132.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *Preprint*, arXiv:2302.00093.

Parshin Shojaee*†, Iman Mirzadeh*, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity.

Peiyang Song, Pengrui Han, and Noah Goodman. 2025. A survey on large language model reasoning failures. In *2nd AI for Math Workshop @ ICML 2025*.

Jiayi Wang and Guoliang Li. 2025. Aop: Automated and interactive llm pipeline orchestration for answering complex queries. In *CIDR*. CIDR.

Jin Wang, Yanlin Feng, Chen Shen, Sajjadur Rahman, and Eser Kandogan. 2025. Towards operationalizing heterogeneous data discovery. *Preprint*, arXiv:2504.02059.

Luoxuan Weng, Yinghao Tang, Yingchaojie Feng, Zhuo Chang, Ruiqin Chen, Haozhe Feng, Chen Hou, Danqing Huang, Yang Li, Huaming Rao, Haonan Wang, Canshi Wei, Xiaofeng Yang, Yuhui Zhang, Yifeng Zheng, Xiuqi Huang, Minfeng Zhu, Yuxin Ma, Bin Cui, and 2 others. 2025. Datalab: A unified platform for llm-powered business intelligence. *Preprint*, arXiv:2412.02205.

Luoxuan Weng, Xingbo Wang, Junyu Lu, Yingchaojie Feng, Yihan Liu, Haozhe Feng, Danqing Huang, and Wei Chen. 2024. Insightlens: Augmenting llm-powered data analysis with interactive insight management and navigation. *Preprint*, arXiv:2404.01644.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Ziming You, Yumiao Zhang, Dexuan Xu, Yiwei Lou, Yandong Yan, Wei Wang, Huaming Zhang, and Yu Huang. 2025. Datawiseagent: A notebook-centric llm agent framework for automated data science. *Preprint*, arXiv:2503.07044.

A Appendix - Prompt for Goal construction

```
'''
You are given a user query. You have to
    extract the following things from
    the query
and context provided to you:
    Question understanding: What do you
        understand from the question.
    Entity extraction: Key entities in
        the question.
    Solution approach: How to solve the
        question in general
    Constraints: If any constraints or
        any additional details which are
          given in the context
which you have to take care while
    answering the questions
'''
```

B Appendix - Prompt for Contextual reasoner

```
'''
Relevant chunks from context: Extract
    relevant chunks(exact match) from
the context which help you get the
    answer
    The context is given by:
    <context>
    {content}
    {content2}
    </context>
The user query is given by:
    <user query>
    {query}
    </user query>
The current understanding/belief is
    given by:
    <belief>
    {belief}
    </belief>

Provide a solution approach: How to
    solve the problem using the context
    given to you
'''
```

C Appendix - Prompt for Workflow scaffolding

```
'''
You are a chatbot who has to create a
    checklist for a downstream 'plan
    executor' pipeline.
You have to create checklist to solve
    user queries based on the
    information
available in the context and the
    metadata given to you.

The context is given by:

<context>
{context_for_planner}
</context>

The metadata is given by:

<metadata>
{metadata}
</metadata>

These are the sources of the data which
    you have:
{files_list}
```

```

Follow these instructions:
<instructions>
{custom_instructions}
</instructions>

The output should be only a parsable
    JSON whose format is given by:

<output_format>
{output_format}
</output_format>

The user query is:
<user_query>
{query}
</user_query>
'''
```

D Appendix - Prompt for Adaptive planning and executor (1)

```
'''
You are an AI assistant. You have to
    execute a plan given
by a upstream 'planner agent' for
    completing a user query.
The plan you have to follow is given by:

<plan>
{plan}
</plan>

The files given to you are:
{files_list}

This is the metadata for the files which
     given to you.
When making a query make sure the column
     names, values
and everything follows the metadata.

<metadata>
{metadata}
</metadata>

Follow these instructions:
<instructions>
{instructions}
</instructions>

The output should be only a parsable
    JSON whose format is given by:
<output_format>
{output_format}
</output_format>
'''
```

E Appendix - Prompt for Adaptive planning and executor (2)

```
'''
The response for the query is given by:
{response}

Instructions:
1. Reflect on the output and take the
    next step according to the response
    as well as the plan.
2. If you get a error on your code,
    rewrite the code and make sure you
    have no undefined variables.
3. In case of writing code, every query
    is independent. \
So make sure you define variable names
    before using them directly as
    variables only have local scope.
4. You will put the plan status as "
    completed" when you have the final
    answer with yourself and you have to
     give it to user.
5. The response has to strictly follow
    the output format given to you.
'''
```

F Appendix - Example trace of I2I-STRADA on DABstep

Hard task – Task ID: 1434

Question: What is the most expensive MCC for a transaction of 5 Euros, in general? If there are many MCCs with the same value, list all of them. Provide a list as an output even if it is one element.

Guideline: Answer must be a list of values in comma-separated list, eg: A, B, C. If the answer is an empty list, reply with an empty string. If a question does not have a relevant or applicable answer for the task, please respond with 'Not Applicable'.

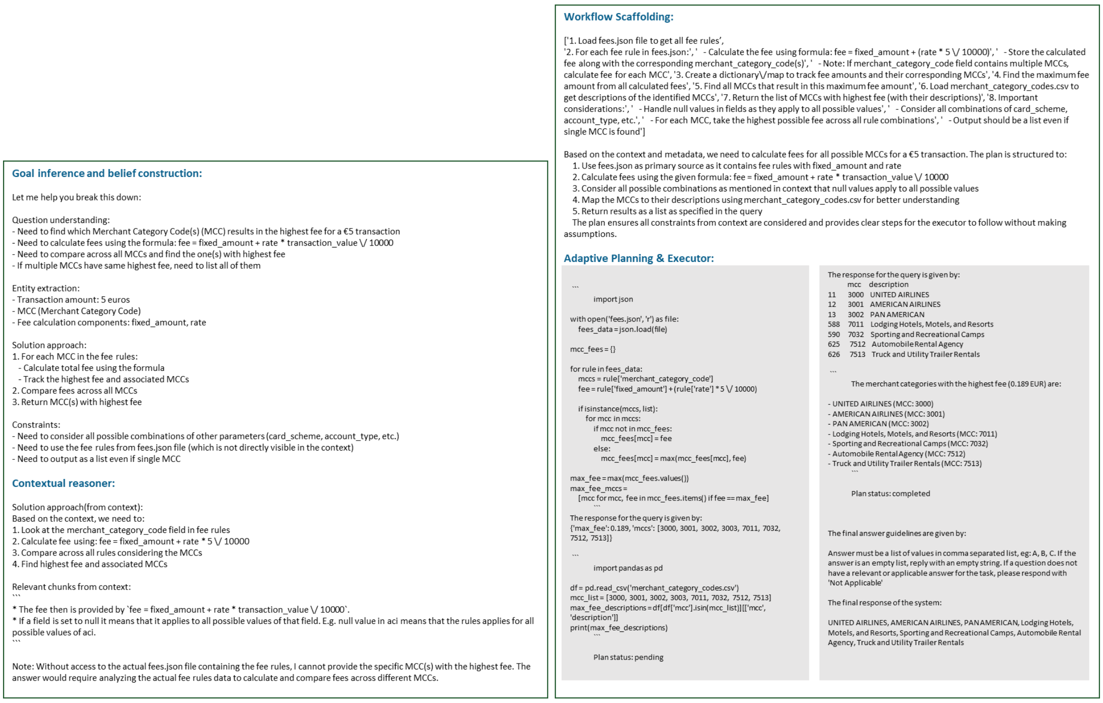
Trace: Figure 2

G Appendix - Example trace of I2I-STRADA on DABench

Hard task – Task ID - 28

Question: Perform comprehensive data preprocessing on the dataset, including cleaning, transformation, and handling of missing values. Handle the missing values in the 'age', 'sex', and 'region' columns by removing the corresponding rows. Transform the 'sex' and 'smoker' columns to binary format (0 and 1). Normalize 'age', 'bmi', 'children', and 'charges' columns. Report the mean of each column after the preprocessing.

Trace: Figure 3



Goal inference and belief construction:

Let me help you break this down:

Question understanding:
- Need to find which Merchant Category Code(s) (MCC) results in the highest fee for a €5 transaction
- Need to calculate fees using the formula: fee = fixed_amount + rate * transaction_value \/ 10000
- Need to compare across all MCCs and find the one(s) with highest fee
- If multiple MCCs have same highest fee, need to list all of them

Entity extraction:
- Transaction amount: 5 euros
- MCC (Merchant Category Code)
- Fee calculation components: fixed_amount, rate

Solution approach:
1. For each MCC in the fee rules:
   - Calculate total fee using the formula
   - Track the highest fee and associated MCCs
2. Compare fees across all MCCs
3. Return MCC(s) with highest fee

Constraints:
- Need to consider all possible combinations of other parameters (card_scheme, account_type, etc.)
- Need to use the fee rules from fees.json file (which is not directly visible in the context)
- Need to output as a list even if single MCC

Contextual reasoner:

Solution approach(from context):
Based on the context, we need to:
1. Look at the merchant_category_code field in fee rules
2. Calculate fee using: fee = fixed_amount + rate * 5 \/ 10000
3. Compare across all rules considering the MCCs
4. Find highest fee and associated MCCs

Relevant chunks from context:
~~~
```
* The fee then is provided by `fee = fixed_amount + rate * transaction_value \/ 10000`.
* If a field is set to null it means that it applies to all possible values of that field. E.g. null value in aci means that the rules applies for all possible values of aci.
```
~~~

Note: Without access to the actual fees.json file containing the fee rules, I cannot provide the specific MCC(s) with the highest fee. The answer would require analyzing the actual fee rules data to calculate and compare fees across different MCCs.

Workflow Scaffolding:

['1. Load fees.json file to get all fee rules',
'2. For each fee rule in fees.json:', '   - Calculate the fee using formula: fee = fixed_amount + (rate * 5 \/ 10000)', '   - Store the calculated fee along with the corresponding merchant_category_code(s)', '   - Note: If merchant_category_code field contains multiple MCCs, calculate fee for each MCC', '3. Create a dictionary\/map to track fee amounts and their corresponding MCCs', '4. Find the maximum fee amount from all calculated fees', '5. Find all MCCs that result in this maximum fee amount', '6. Load merchant_category_codes.csv to get descriptions of the identified MCCs', '7. Return the list of MCCs with highest fee (with their descriptions)', '8. Important considerations:', '   - Handle null values in fields as they apply to all possible values', '   - Consider all combinations of card_scheme, account_type, etc.', '   - For each MCC, take the highest possible fee across all rule combinations', '   - Output should be a list even if single MCC is found']

Based on the context and metadata, we need to calculate fees for all possible MCCs for a €5 transaction. The plan is structured to:
1. Use fees.json as primary source as it contains fee rules with fixed_amount and rate
2. Calculate fees using the given formula: fee = fixed_amount + rate * transaction_value \/ 10000
3. Consider all possible combinations as mentioned in context that null values apply to all possible values
4. Map the MCCs to their descriptions using merchant_category_codes.csv for better understanding
5. Return results as a list as specified in the query

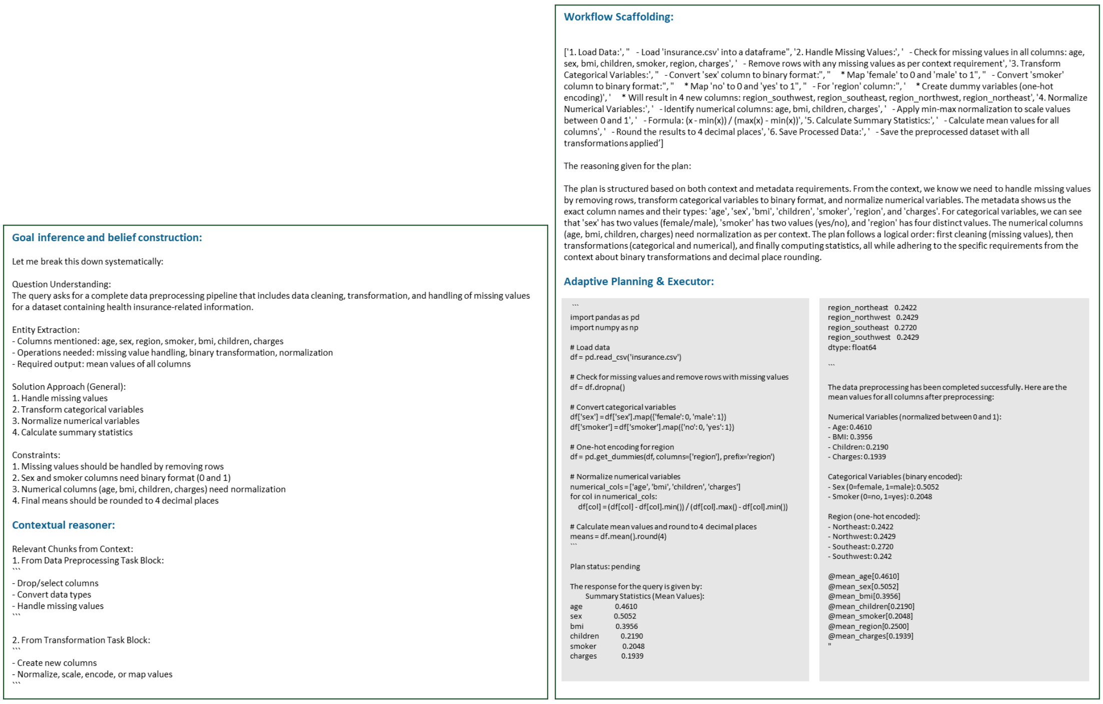
The plan ensures all constraints from context are considered and provides clear steps for the executor to follow without making assumptions.

Adaptive Planning & Executor:

~~~
```
        import json

with open('fees.json', 'r') as file:
    fees_data = json.load(file)

mcc_fees = {}

for rule in fees_data:
    mccs = rule['merchant_category_code']
    fee = rule['fixed_amount'] + (rule['rate'] * 5 \/ 10000)

    if isinstance(mccs, list):
        for mcc in mccs:
            if mcc not in mcc_fees:
                mcc_fees[mcc] = fee
            else:
                mcc_fees[mcc] = max(mcc_fees[mcc], fee)

max_fee = max(mcc_fees.values())
max_fee_mccs =
    [mcc for mcc, fee in mcc_fees.items() if fee == max_fee]
        ```
The response for the query is given by:
{'max_fee': 0.189, 'mccs': [3000, 3001, 3002, 3003, 7011, 7032,
7512, 7513]}

```
        import pandas as pd

df = pd.read_csv('merchant_category_codes.csv')
mcc_list = [3000, 3001, 3002, 3003, 7011, 7032, 7512, 7513]
max_fee_descriptions = df[df['mcc'].isin(mcc_list)][['mcc',
'description']]
print(max_fee_descriptions)
        ```

        Plan status: pending
~~~

~~~
The response for the query is given by:
      mcc   description
11    3000  UNITED AIRLINES
12    3001  AMERICAN AIRLINES
13    3002  PAN AMERICAN
588   7011  Lodging Hotels, Motels, and Resorts
590   7032  Sporting and Recreational Camps
625    7512  Automobile Rental Agency
626    7513  Truck and Utility Trailer Rentals

```
        The merchant categories with the highest fee (0.189 EUR) are:

- UNITED AIRLINES (MCC: 3000)
- AMERICAN AIRLINES (MCC: 3001)
- PAN AMERICAN (MCC: 3002)
- Lodging Hotels, Motels, and Resorts (MCC: 7011)
- Sporting and Recreational Camps (MCC: 7032)
- Automobile Rental Agency (MCC: 7512)
- Truck and Utility Trailer Rentals (MCC: 7513)
        ```

        Plan status: completed


The final answer guidelines are given by:

Answer must be a list of values in comma separated list, eg: A, B, C. If the
answer is an empty list, reply with an empty string. If a question does not
have a relevant or applicable answer for the task, please respond with
'Not Applicable'

The final response of the system:

UNITED AIRLINES, AMERICAN AIRLINES, PAN AMERICAN, Lodging Hotels,
Motels, and Resorts, Sporting and Recreational Camps, Automobile Rental
Agency, Truck and Utility Trailer Rentals
~~~

Figure 2: Trace of the agent for task 1434 from DABstep dataset



Goal inference and belief construction:

Let me break this down systematically:

Question Understanding:
The query asks for a complete data preprocessing pipeline that includes data cleaning, transformation, and handling of missing values for a dataset containing health insurance-related information.

Entity Extraction:
- Columns mentioned: age, sex, region, smoker, bmi, children, charges
- Operations needed: missing value handling, binary transformation, normalization
- Required output: mean values of all columns

Solution Approach (General):
1. Handle missing values
2. Transform categorical variables
3. Normalize numerical variables
4. Calculate summary statistics

Constraints:
1. Missing values should be handled by removing rows
2. Sex and smoker columns need binary format (0 and 1)
3. Numerical columns (age, bmi, children, charges) need normalization
4. Final means should be rounded to 4 decimal places

Contextual reasoner:

Relevant Chunks from Context:
1. From Data Preprocessing Task Block:
~~~
```
- Drop/select columns
- Convert data types
- Handle missing values
```
~~~

2. From Transformation Task Block:
~~~
```
- Create new columns
- Normalize, scale, encode, or map values
```
~~~

Workflow Scaffolding:

['1. Load Data:', "   - Load 'insurance.csv' into a dataframe", '2. Handle Missing Values:', '   - Check for missing values in all columns: age, sex, bmi, children, smoker, region, charges', '   - Remove rows with any missing values as per context requirement', '3. Transform Categorical Variables:', "   - Convert 'sex' column to binary format:", "   * Map 'female' to 0 and 'male' to 1", "   - Convert 'smoker' column to binary format:", "   * Map 'no' to 0 and 'yes' to 1", "   - For 'region' column:", '   * Create dummy variables (one-hot encoding)', '   * Will result in 4 new columns: region_southwest, region_southeast, region_northwest, region_northeast', '4. Normalize Numerical Variables:', '   - Identify numerical columns: age, bmi, children, charges', '   - Apply min-max normalization to scale values between 0 and 1', '   - Formula: (x - min(x)) / (max(x) - min(x))', '5. Calculate Summary Statistics:', '   - Calculate mean values for all columns', '   - Round the results to 4 decimal places', '6. Save Processed Data:', '   - Save the preprocessed dataset with all transformations applied']

The reasoning given for the plan:

The plan is structured based on both context and metadata requirements. From the context, we know we need to handle missing values by removing rows, transform categorical variables to binary format, and normalize numerical variables. The metadata shows us the exact column names and their types: 'age', 'sex', 'bmi', 'children', 'smoker', 'region', and 'charges'. For categorical variables, we can see that 'sex' has two values (female/male), 'smoker' has two values (yes/no), and 'region' has four distinct values. The numerical columns (age, bmi, children, charges) need normalization as per context. The plan follows a logical order: first cleaning (missing values), then transformations (categorical and numerical), and finally computing statistics, all while adhering to the specific requirements from the context about binary transformations and decimal place rounding.

Adaptive Planning & Executor:

~~~
 ```
import pandas as pd
import numpy as np

# Load data
df = pd.read_csv('insurance.csv')

# Check for missing values and remove rows with missing values
df = df.dropna()

# Convert categorical variables
df['sex'] = df['sex'].map({'female': 0, 'male': 1})
df['smoker'] = df['smoker'].map({'no': 0, 'yes': 1})

# One-hot encoding for region
df = pd.get_dummies(df, columns=['region'], prefix='region')

# Normalize numerical variables
numerical_cols = ['age', 'bmi', 'children', 'charges']
for col in numerical_cols:
    df[col] = (df[col] - df[col].min()) / (df[col].max() - df[col].min())

# Calculate mean values and round to 4 decimal places
means = df.mean().round(4)
```

Plan status: pending

The response for the query is given by:
      Summary Statistics (Mean Values):
age             0.4610
sex             0.5052
bmi             0.3956
children        0.2190
smoker          0.2048
charges         0.1939
~~~

~~~
region_northeast   0.2422
region_northwest   0.2429
region_southeast   0.2720
region_southwest   0.2429
dtype: float64

```

The data preprocessing has been completed successfully. Here are the
mean values for all columns after preprocessing:

Numerical Variables (normalized between 0 and 1):
- Age: 0.4610
- BMI: 0.3956
- Children: 0.2190
- Charges: 0.1939

Categorical Variables (binary encoded):
- Sex (0=female, 1=male): 0.5052
- Smoker (0=no, 1=yes): 0.2048

Region (one-hot encoded):
- Northeast: 0.2422
- Northwest: 0.2429
- Southeast: 0.2720
- Southwest: 0.242

@mean_age[0.4610]
@mean_sex[0.5052]
@mean_bmi[0.3956]
@mean_children[0.2190]
@mean_smoker[0.2048]
@mean_region[0.2500]
@mean_charges[0.1939]
"
~~~

Figure 3: Trace of the agent for task 28 from DABench dataset