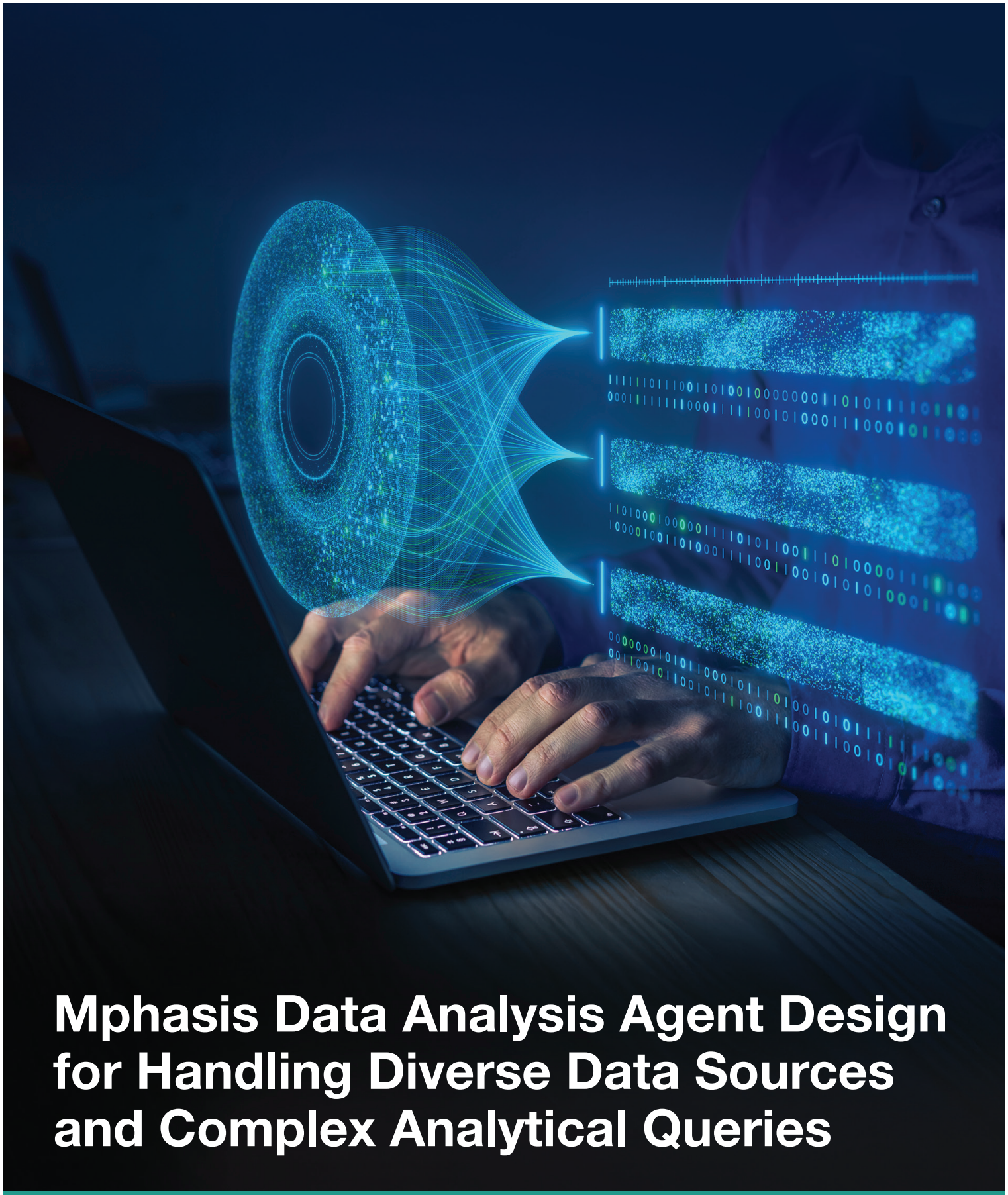


Mphasis Data Analysis Agent Design for Handling Diverse Data Sources and Complex Analytical Queries

Whitepaper by

Sai Barath Sundar, Senior Manager – Mphasis NEXT Labs | Solution Architect
Pranav Satheesan, Lead Data Scientist – Mphasis NEXT Labs | Solution Development
Dr. Udayaadithya Avadhanam, Principal & Vice President – Mphasis NEXT Labs | Advisor
Biju Mathews, Partner & Head - Mphasis NEXT Labs | Advisor

Mphasis
The Next Applied

Contents

1. Introduction

Reasoning models have been developed to address complex problems requiring multiple steps, often being employed as agents within frameworks such as ReACT [1]. However, these are typically general-purpose reasoners and when presented with the same problem, they can sample completely different approaches, leading to inconsistent and divergent results [Appendix A]. This highlights a lack of systematicity and guidance in the reasoning process, which is often a critical requirement for real-life use cases.

Our aim was to build an agent design for data analysis, recognizing that effective data analysis extends significantly beyond simple tasks like generating SQL queries or performing Retrieval Augmented Generation (RAG). We identify several challenges related to data and agentic systems.

Key data-related challenges

1. Handling heterogeneous data sources - tabular (e.g., relational database, columnar data files like CSV, parquet, etc.), non-tabular (e.g., JSONs with consistent/inconsistent schema, files with textual data).
2. Multiple intermediate calculations that must be correctly sequenced and interpreted.
3. Supporting open-ended analytical queries, which span a diverse set of objectives such as descriptive, diagnostic, counterfactual, procedural or even purely exploratory tasks.

Key agentic system challenges

1. Difficulty in planning without sufficient initial data exploration leading to incorrect data assumptions.
2. Missing to identify specific procedural rules within noisy contextual data.
3. Ensuring that execution/actions remain consistently aligned with the overall planning.

2. Our Approach

We design our agent as a workflow composed of multiple sub-tasks (or sub-agents), where each sub-agent focuses on a specific part of the overall reasoning and planning process.

2.1. Key Tenets of Our Approach

1. **Progressive abstraction of information:** At each stage, we perform controlled abstraction of information, ensuring that critical details are preserved while filtering out irrelevant or noisy data. This prevents ignoring critical details (due to over-summarization) and the inclusion of unnecessary information, maintaining focus throughout the workflow.
2. **Multi-step refinement (two planning stages):** We built a multi-step reasoning process where understanding and planning are progressively refined across stages. This allows the agent to iteratively build up more accurate and contextually relevant plans, rather than relying on a single pass.
3. **Usage of non-reasoning model:** Since each stage is a single task, we leverage a non-reasoning model – Anthropic Claude 3.5 Sonnet, which performs the task effectively.

2.2. Agent Building Blocks

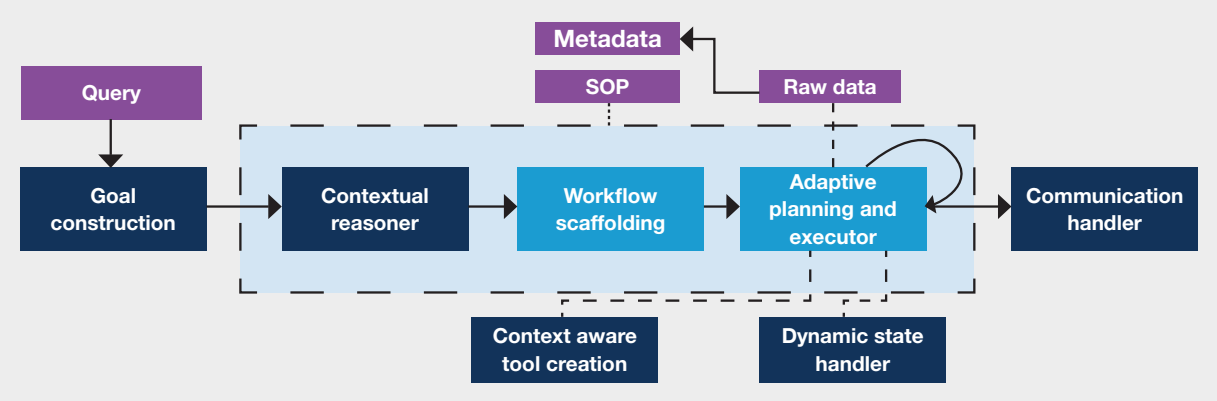


Figure 1: The workflow of tasks for our agent

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 the problem type is essential for guiding subsequent data exploration, and building belief from scratch ensures the agent considers every detail relevant to the request. The outcome of this step consists of:
   a. **Question understanding** - Understand the core intent of the user.
   b. **Entity extraction** - Identifying relevant data points, dimensions or concepts mentioned in the query.
   c. **Generic solution approach** - Outlining a preliminary high-level strategy.
   d. **Constraints** - Detailing any specific limitations or conditions provided.
2. **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 Standard Operating Procedures (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.
3. **Two planning stages:**
   a. **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.
   b. **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.
4. **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.
5. **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.
6. **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.

3. Results

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 [2]. The second benchmark dataset is DABench [3]. 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 solutions.

3.1. DABstep

The DABStep dataset, 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.

DATA SOURCE NAME	DESCRIPTION	TYPE
payments.csv	Dataset of 138k anonymized transactions with fraud and risk signals	Structured (CSV)
payments-readme.md	Documentation for the payments dataset	Unstructured
acquirer_countries.csv	Table of acquiring banks and their respective countries	Structured (CSV)
fees.json	Dataset of 1000 scheme fee structures	Structured (JSON)
merchant_category_codes.csv	Table of Merchant Category Codes (MCCs)	Structured (CSV)
merchant_data.json	Table describing merchants	Structured (JSON)
manual.md	Simplified markdown handbook distilling essential business knowledge	Unstructured

Table 1: A description of the DABstep dataset

We used Anthropic's Claude 3.5 Sonnet in our agentic workflow and at the time of publishing this article, our agent tops the leaderboard, outperforming data science agents from Google and Microsoft, with an accuracy of 80.56% on easy tasks and 28.04% on hard tasks.

AGENT	EASY LEVEL ACCURACY	HARD LEVEL ACCURACY	ORGANIZATION	MODEL FAMILY	DATE
MPHASIS-I2I-AGENTS	**80.56%**	**28.04%**	**(Ours) Mphasis Limited**	**claude-3-5-sonnet**	**10-04-2025**
DICE	75.00%	27.25%	Microsoft	o3-mini	17-04-2025
O4-MINI REASONING PROMPT BASELINE	76.39%	14.55%	Hugging Face	OpenAI o4-mini	22-04-2025
CLAUDE 3.7 SONNET REACT BASELINE	75.00%	13.76%	Hugging Face	claude-3-7-sonnet	07-04-2025
GEMINI DATA SCIENCE AGENT	61.11%	9.79%	Google	Gemini 2.0 Flash	10-02-2025
CLAUDE 3.5 SONNET REACT BASELINE	77.78%	9.26%	Adyen	claude-3-5-sonnet	23-01-2025
DEEPSEEK V3 REACT BASELINE	66.67%	5.56%	Adyen	Deepseek v3	23-01-2025
LLAMA 3.3 70B REACT BASELINE	68.06%	3.70%	Adyen	Llama 3.3 70B Instruct	23-01-2025

Table 2: DABstep leaderboard

Shown above are the top few state-of-the-art (SOTA) performing agents. To view the live leaderboard, visit: DABstep Leaderboard - a Hugging Face Space by Adyen.

Where our agent succeeds:

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

Where we see chances to improve:

- The agent seems inconsistent when applying the 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 B presents our agent’s traces 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.

3.2. DABench

The InfiAgent-DABench benchmark, introduced by Hu et al. (2024) [3], 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. Hence, we provided just the definitions of the tasks given by Hu et al. (2024) as SOP input.

AGENT	ACCURACY	MODEL FAMILY
DATA INTERPRETER (S. HONG ET AL, 2024) [4]	94.93%	GPT-4o
MPHASIS-I2I-AGENTS (OURS)	**90.27%**	**claude-3-5-sonnet**
DATAWISE AGENT (Z. YOU ET AL, 2025) [5]	85.99%	GPT-4o
DATA INTERPRETER (S. HONG ET AL, 2024) [4]	73.55%	GPT-4
AGENTPOIROT (G. SAHU ET AL, 2024) [6]	75.88%	GPT-4
DATALAB (L. WENG ET AL, 2024) [7]	75.10%	GPT-4

Table 3: Performance comparison on DABench

The accuracy metric shown 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 C presents our agent’s traces on a hard task.

4. Concluding Remarks

In this article, 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-agents, each dedicated to a distinct aspect of reasoning and planning. This multi-step 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.

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

Appendix A: Inconsistencies of Reasoning Models

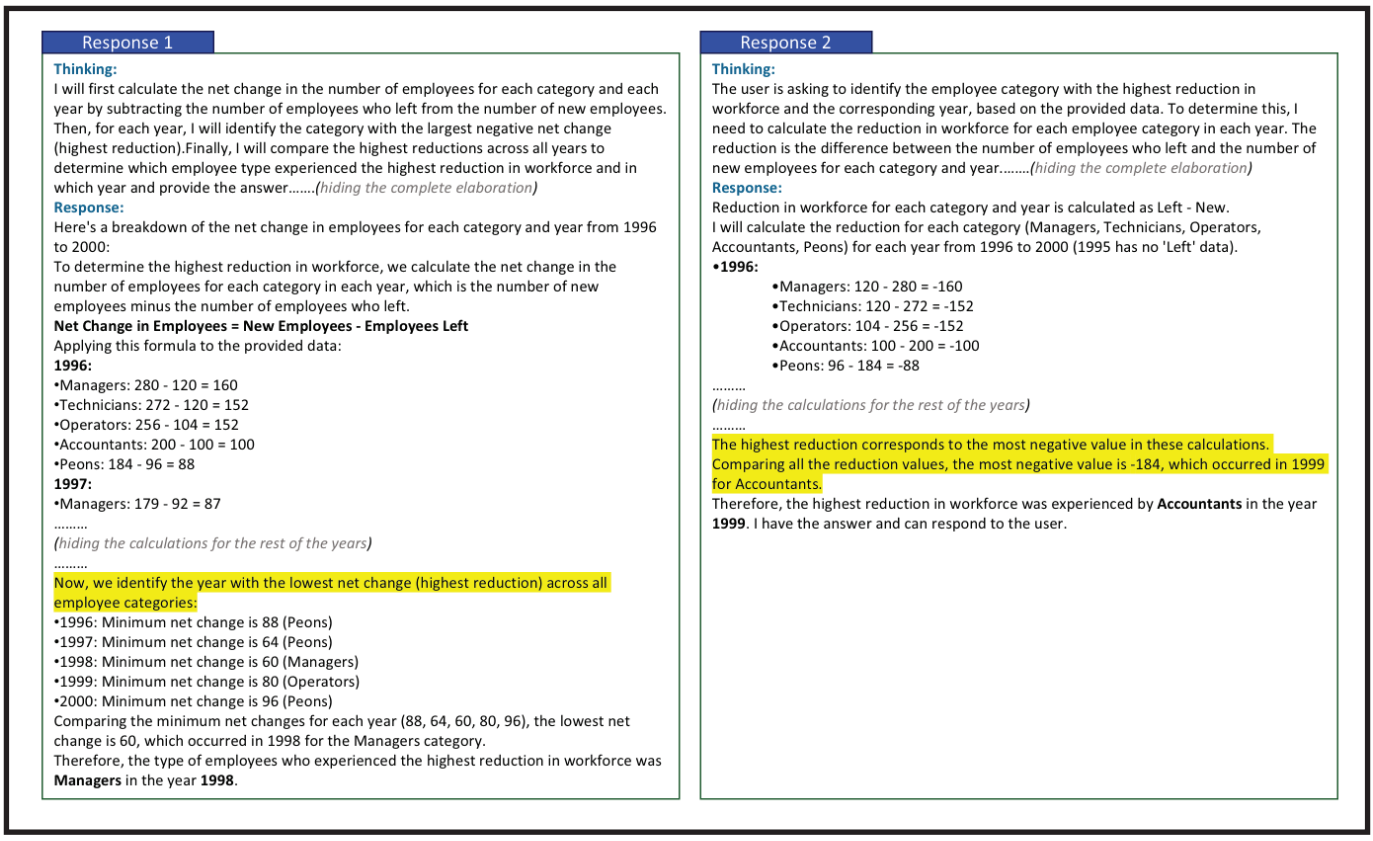
Two attempts at solving a question using Gemini 2.5 Flash with reasoning.

Problem

The following table presents the number of new employees hired and employees who left across various job categories in a company, recorded each year since the company's founding in 1995.

Q. Based on the data provided, identify which category of employees experienced the largest reduction in workforce in a particular year, and specify the year in which this occurred.

Year	Managers		Technicians		Operators		Accountants		Peons	
	New	Left	New	Left	New	Left	New	Left	New	Left
1995	760	-	1200	-	880	-	1160	-	820	-
1996	280	120	272	120	256	104	200	100	184	96
1997	179	92	240	128	240	120	224	104	152	88
1998	148	88	236	96	208	100	248	96	196	80
1999	160	72	256	100	192	112	272	88	224	120
2000	193	96	288	112	248	144	260	92	200	104



The expected answer is that in none of the categories there was any reduction in the workforce over the years.

Response 1: This shows a slight change in interpretation of the question to provide some close answer.

Response 2: This response shows an incorrect reasoning by the model.

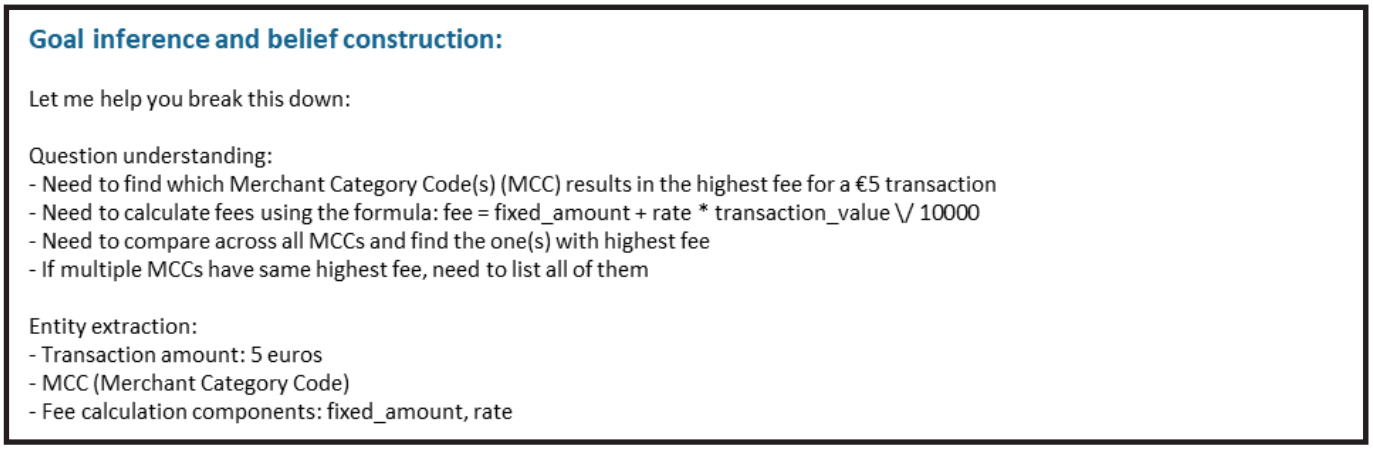
Appendix B: Example of Our Agent's Trace on DABstep

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:



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

Appendix C: Example of Our Agent's Trace 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:

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]
"

References

[1] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y., "ReAct: Synergizing Reasoning and Acting in Language Models.," in *https://arxiv.org/abs/2210.03629, 2023.*

[2] Martin Iglesias, Alex Egg, Friso Kingma, "Data Agent Benchmark for Multi-step Reasoning (DABstep)," 2025. [Online]. Available: *https://www.adyen.com/knowledge-hub/data-agent-benchmark-for-multi-step-reasoning-dabstep.*

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

[4] Sirui Hong and Yizhang Lin and Bang Liu and Bangbang Liu and Binhao Wu and Ceyao Zhang and Chenxing Wei and Danyang Li and Jiaqi Chen and Jiayi Zhang and Jinlin Wang and Li Zhang and Lingyao Zhang and Min Yang and Mingchen Zhuge and Taicheng Guo and Tuo Z, "Data Interpreter: An LLM Agent For Data Science," in *https://arxiv.org/abs/2402.18679*, 2024.

[5] Ziming You and Yumiao Zhang and Dexuan Xu and Yiwei Lou and Yandong Yan and Wei Wang and Huaming Zhang and Yu Huang, "DatawiseAgent: A Notebook-Centric LLM Agent Framework for Automated Data Science," in *https://arxiv.org/abs/2503.07044*, 2025.

[6] Gaurav Sahu and Abhay Puri and Juan Rodriguez and Amirhossein Abaskohi and Mohammad Chegini and Alexandre Drouin and Perouz Taslakian and Valentina Zantedeschi and Alexandre Lacoste and David Vazquez and Nicolas Chapados and Christopher Pal and Sai Rajesw, "InsightBench: Evaluating Business Analytics Agents Through Multi-Step Insight Generation," in *https://arxiv.org/abs/2407.06423*, 2024.

[7] Luoxuan Weng and Yinghao Tang and Yingchaojie Feng and Zhuo Chang and Ruiqin Chen and Haozhe Feng and Chen Hou and Danqing Huang and Yang Li and Huaming Rao and Haonan Wang and Canshi Wei and Xiaofeng Yang and Yuhui Zhang and Yifeng Zheng and Xiuqi Huang , "DataLab: A Unified Platform for LLM-Powered Business Intelligence," in *https://arxiv.org/abs/2412.02205*, 2024.

About Mphasis

Mphasis' purpose is to be the *"Driver in the Driverless Car"* for Global Enterprises by applying next-generation design, architecture and engineering services, to deliver scalable and sustainable software and technology solutions. Customer centricity is foundational to Mphasis, and is reflected in the Mphasis' Front2Back™ Transformation approach. Front2Back™ uses the exponential power of cloud and cognitive to provide hyper-personalized ($C = X2C^{2}_{TM} = 1$) digital experience to clients and their end customers. Mphasis' Service Transformation approach helps 'shrink the core' through the application of digital technologies across legacy environments within an enterprise, enabling businesses to stay ahead in a changing world. Mphasis' core reference architectures and tools, speed and innovation with domain expertise and specialization, combined with an integrated sustainability and purpose-led approach across its operations and solutions are key to building strong relationships with marquee clients. Click here to know more. (BSE: 526299; NSE: MPHASIS)

For more information, contact: marketinginfo.m@mphasis.com

USA
Mphasis Corporation
41 Madison Avenue
35th Floor, New York
New York 10010, USA
Tel: +1 (212) 686 6655

UK
Mphasis UK Limited
1 Ropemaker Street, London
EC2Y 9HT, United Kingdom
T : +44 020 7153 1327

INDIA
Mphasis Limited
Bagmane World Technology Center
Marathahalli Ring Road
Doddanakundhi Village, Mahadevapura
Bangalore 560 048, India
Tel.: +91 80 3352 5000

VAS 14/06/25 US LETTER BASIL 9644

www.mphasis.com