Structured Reasoning with LLMs for SemEval-2025 Task 8: QA over Tabular Data

Prajakta Kini, Neha Kolambe, Pratik Bhirud
Department of Computer Science
University of Colorado Boulder
{prajakta.kini,neha.kolambe,pratik.bhirud}@colorado.edu

Abstract

This paper presents our completed system for SemEval-2025 Task 8: Question Answering over Tabular Data using the DataBench benchmark. The task involved answering natural language questions based on diverse real-world tabular datasets, where answers may be boolean, numeric, categorical, or lists. We investigated a hybrid strategy that leverages large language models (LLMs) for structured reasoning. Specifically, we explored multiple LLM-based approaches, including direct question answering, retrieval-augmented generation, and column-aware input filtering. Our evaluation assessed these methods in terms of their effectiveness across varying table sizes, question types, and reasoning complexity.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in a wide range of natural language processing tasks, including open-domain question answering, summarization, and reasoning. Nevertheless, tabular data continues to be a fundamental format for storing structured information in a variety of domains, including financial reports, sports statistics, scientific research, and government datasets. Recent surveys, such as (Fang et al., 2024) offer a comprehensive review of recent progress in applying Large Language Models (LLMs) to various tabular data tasks, including prediction, data synthesis, and table understanding.

Answering questions on tabular data requires an LLM to understand the structure of the table, including row and column structure, column types, etc. Input length constraints for LLMs make it impractical to feed them an entire table and also prevent full-table encoding, especially for wide tables. Thus, there is a need for systems that can understand and answer questions on tables.

Models such as TAPAS (Herzig et al., 2020) and TaBERT (Yin et al., 2020) made early attempts to adapt pre-trained language models for reasoning over tabular data using specialized embeddings and pretraining objectives. TAPAS extends the BERT architecture with specific embeddings to capture tabular structure and is pre-trained on a large dataset of text and tables from Wikipedia using a masked language modelling objective adapted for structured data.

We explored three prompting-based strategies for question answering over tabular data using LLMs. The first approach serves as our baseline, where the full table is passed to the model using either CSV-style or key-value (KV) format, thus allowing us to assess how input structure affects performance. The second approach followed a retrieval-augmented generation (RAG) framework where we embed each column using an embedder and index them with FAISS, thus allowing the system to focus only on the columns most relevant to the question before generating an answer. The third approach is a two-phase prompting strategy. In Phase 1, the LLM receives the question along with column names and data types, and returns a list of relevant columns. In Phase 2, the model is prompted with the filtered columns and the original question to produce the final answer. These three methods allowed us to compare the effects of table size, input filtering, and structured prompting on overall performance.

2 Related Work

The TableQuery system proposed by (Abraham et al., 2022) translates natural language questions into SQL queries to retrieve answers from the large tabular dataset, thus improving efficiency by avoiding full-table loading. While our baseline passed the entire table to a language model, TableQuery used structured query generation for greater scalability. In contrast, we explored column selection and generation-based approaches to reduce input size while maintaining accuracy.

Retrieval-augmented generation (RAG) has also been adapted for table QA to better handle large and complex inputs. T-RAG, introduced by (Pan et al., 2022), retrieved relevant table cells and used BART for answer generation.

(Hei et al., 2024) further enhanced this with DR-RAG, which incorporated document-level relevance scoring. We built on these ideas by experimenting with RAG at different granularities and comparing it with a plain LLM baseline and a two-phase prompting method to assess input filtering strategies.

(Evangelatos et al., 2025) took a different approach, translating natural language queries into executable Python code for reasoning over tabular data. Their method emphasized interpretability through structured intermediate steps and performed well on the DataBench benchmark. In contrast, we focused on end-to-end LLM-based answer generation, comparing full-table prompting, retrieval-augmented input, and column-aware filtering without relying on program induction.

3 Methodology

3.1 Dataset

We have used the Databench benchmark (Osés Grijalba et al., 2024), which contains 65 tables derived from real-world data, spanning multiple domains. Each table includes roughly 20 human-authored natural language questions, leading to a total of 1300 questions. For each question, the dataset provides gold answers, gold column set and the expected answer type, among other details.

3.2 LLM Models

All of our experiments rely on LLM instruction prompting as the core reasoning component, with some auxiliary components being used in the RAG approach. Specifically, we used the **Llama-3-8b-8192** and **Llama-3-70b-8192** models for all our experiments, which is accessible through the Groq API. These models were selected for their strong general performance and ease of integration. Both models were evaluated on the same questions across all approaches to assess the impact of model size on performance.

3.3 Tabular Data Representation

We explored two formats for presenting tabular data to the LLM: a CSV-style format and a key-value (KV) format. The CSV format is based on the DataBench benchmark (Osés Grijalba et al., 2024). It flattens each row into a comma-separated string. The KV format is inspired by (Wu and Hou, 2025). It maps each column name (key) to its corresponding value, which improves clarity and helps the model reason over individual records. In KV format, each table row is serialized in the following format:

$$s(x_i) = \{k_1 : v_1, k_2 : v_2, \ldots, k_M : v_M\} \quad (1)$$

3.4 Prompting Strategies

All three approaches use a shared prompting template: we serialize the table as key-value pairs and follow it with a natural language question. We instruct the model to respond with a JSON object using valid Python-native types, making the output both readable and machine-parsable. Appendix A shows the full prompt format. The baseline approach uses the full table as context, while RAG and two-phase prompting use filtered tables. Two-phase prompting adds a first step where the model selects relevant columns from a list of column names and types; we then pass the selected subset to the main prompt for answer generation.

3.5 Approach 1: Baseline

The DataBench paper (Osés Grijalba et al., 2024) provides strong baselines using LLaMA-2 models in a CSV-style input format. However, we generated our baselines using LLaMA-3-8B and LLaMA-3-70B since our work is based on them. These baselines serve as reference points to evaluate our other approaches, including RAG and two-phase prompting. In addition to using a key-value (KV) input style, we also created a CSV-style baseline for the 8B model. This allows us to compare performance across input formats and decide which structure better supports table-based QA.

3.6 Approach 2: RAG with Column-Type Metadata

We used the RAG strategy, where we embed each column independently and store the embeddings in a FAISS index. We used the `BAAI/bge-small-en-v1.5` model as our embedder due to its strong performance in open-domain retrieval tasks. To make the embeddings more informative, we formatted each column as a descriptive string that includes its name, data type, and two sample values. We indexed each dataset separately and stored the column metadata alongside the FAISS index for efficient retrieval. During inference, we retrieved the top-k most relevant columns for a given question using the FAISS index. We set $k = 4$ based on validation performance. Instead of prompting the model with all columns, we used only the retrieved column subset to construct the input prompt. This reduced the prompt length and improved answer relevance.

3.7 Approach 3: Two-phase LLM prompting

We also utilized a two-phase LLM-prompting approach that separates column selection from answer generation. In the **first phase**, the LLM is prompted with the question and column headers with data types, and is asked to identify the most relevant columns. The output of this phase is a set of predicted relevant columns. In the **second phase**, we extract only the data from the predicted columns and embed it into the LLM prompt alongside the original question. The model is then tasked with generating the final answer using this filtered table input.

3.8 LLM Response Processing

To ensure fair evaluation, LLM outputs are parsed and normalized based on the expected answer type. We adopted a relaxed matching scheme to accommodate minor formatting variations: numerical values are cast to floats, categorical responses are stripped and lowercased, booleans account for common variants such as "yes", "true", or "1", and list-type answers are converted to sets to allow for order-agnostic comparison. This approach helps ensure models aren’t penalized for minor formatting differences when their answers are otherwise correct in meaning.

3.9 Evaluation Metrics

We evaluate model performance on a per-dataset basis by measuring the accuracy of both predicted answers and columns against the ground truth. Metrics are collected per table by question type and by the number of columns involved (single vs. multiple). For column selection, we also track counts of correct and incorrect selections, as well as formatting errors. Accuracy is computed within each category, and an overall mean average is reported across all datasets to summarize model performance.

4 Experiments and results

4.1 Datasets

All our experiments, across all approaches, were conducted on the Databench QA benchmark (Osés Grijalba et al., 2024). While the original benchmark comprises 65 tables and approximately 1,300 questions, we excluded 6 tables due to resource constraints, resulting in a final evaluation set of 59 tables and roughly 1,180 questions.

4.2 LLM model variants

We conducted all experiments using two variants of the LLaMA-3 family of language models: LLaMA-3-8B-8192 and LLaMA-3-70B-8192. Both of these models have a context window of size 8192 tokens.

4.3 Baseline models

We evaluate three baseline models: LLaMA-3-8B-CSV, LLaMA-3-8B-KV, and LLaMA-3-70B-KV. These allow us to assess the impact of input representation (CSV vs key-value) on question answering over tabular data. Each model is prompted with the full table and question, without any retrieval or filtering. Prompt construction details are provided in Appendix A.

4.4 RAG-based models

We evaluate two RAG-based models: LLaMA-3-8B-KV and LLaMA-3-70B-KV. This setup helps us compare the performance of smaller and larger models under retrieval-augmented prompting. Instead of passing the full table, we first retrieve the most relevant columns using FAISS and append only those columns to the prompt. This reduces input length and encourages the model to focus on relevant context for answering the question.

4.5 Two-phase Prompting Models

We evaluate two models, LLaMA-3-8B-KV and LLaMA-3-70B-KV, using a two-phase prompting setup. In Phase 1, we ask the model to select the most relevant columns by providing only the question and the column headers with their data types. In Phase 2, we give the model only the selected columns and the question, so it can focus entirely on generating the answer. This approach helps reduce noise in the input, separates the reasoning process into clear steps, and lets us evaluate how well the model selects relevant columns.

Approach	Model	avg	boolean	number	category	list[category]	list[number]	single col	multiple cols
Databench Baseline	Llama2-7B-CSV	0.144	0.380	0.105	0.194	0.031	0.008	0.162	0.118
	Llama2-13B-CSV	0.193	0.566	0.136	0.217	0.039	0.008	0.205	0.177
Our Baseline	Llama3-8B-CSV	0.267	0.451	0.314	0.311	0.139	0.118	0.326	0.181
	Llama3-8B-KV	0.322	0.515	0.309	0.392	0.198	0.198	0.385	0.218
	Llama3-70B-KV	**0.497**	**0.738**	0.444	0.572	0.353	0.380	0.563	0.366
RAG	Llama3-8B-KV	0.310	0.557	0.322	0.311	0.146	0.211	0.403	0.190
	Llama3-70B-KV	0.489	0.646	**0.457**	0.557	**0.374**	0.414	**0.569**	0.370
2-Phase prompting	Llama3-8B-KV	0.184	0.207	0.215	0.265	0.118	0.114	0.200	0.175
	Llama3-70B-KV	0.491	0.658	0.376	**0.614**	0.367	**0.443**	0.473	**0.459**

Table 1: Mean accuracy across answer types and column configurations under different prompting approaches.

Approach	Model	Right	Wrong	Error
DB Baseline	Llama2-7B-CSV	18.5	63.4	18.2
	Llama2-13B-CSV	22.7	51.2	26.1
Baseline	Llama3-8B-CSV	57.0	34.2	8.7
	Llama3-8B-KV	76.6	20.5	2.9
	Llama3-70B-KV	**87.3**	**11.8**	**0.9**
RAG	Llama3-8B-KV	66.1	30.3	3.6
	Llama3-70B-KV	75.6	22.9	1.5
2-Phase	Llama3-8B-KV	50.5	44.4	5.1
	Llama3-70B-KV	78.0	20.7	1.4

Table 2: Column prediction quality (%): Right, Wrong, and Format Error across 59 datasets.

4.6 Results

Table 1 shows the mean accuracy for each answer type and column configuration, across the different approaches. The Llama-3-70B-KV model gave the best results, with an overall accuracy of **49.7%**. Table 2 shows that it also selected the correct columns in **87.3%** of cases. All models had relatively low format-error rates compared to the Databench baseline.

5 Discussion and Findings

5.1 Type of Answer

Among all models and approaches, the Llama-3-70B-KV baseline achieved the highest overall accuracy, particularly excelling in boolean questions. As expected, larger models consistently outperformed smaller ones across all answer types. Both RAG and two-phase prompting approaches showed strong performance on more complex question types, especially lists, numbers, and categorical answers. While Llama-3 represents a significant improvement over Llama-2, the comparison still remains meaningful. Our Llama-3-8B-CSV baseline outperformed the databench baseline by 12.3%. Notably, the RAG LLama-3-70B-KV model performed best on questions involving numbers and lists of categories, while the two-phase prompting version of the same model did best on category-based questions and lists of numbers.

5.2 Answers using single vs multiple Columns

Both the RAG and two-phase prompting models showed improvements over the baseline on single and multi-column questions. For single-column questions, the RAG Llama-3-70B-KV model reached 56.9% accuracy, slightly better than the baseline. On multi-column questions, the two-phase prompting model with Llama-3-70B-KV performed best at 45.9%, showing it handled more complex queries more effectively. These results suggest that filtering the input to only relevant columns helps improve performance across question types.

5.3 Columns used

We also looked at whether the models selected the right columns and followed the expected output format. In multi-column questions, we noticed that models often included the correct columns but added a few extras that weren't needed. The selections weren't always minimal. Additionally, all of our approaches have improved column selection accuracy over the Databench baseline.

5.4 Formatting errors

In all our approaches, JSON format errors, where the output couldn't be parsed were rare. Larger models handled formatting more reliably. All our models significantly reduced JSON format errors compared to the Databench baseline.

6 Conclusions and Future Work

In this work, we explored strategies to improve question answering over tabular data using large language models. Our experiments show that filtering table inputs through RAG or a 2-phase prompting setup, can lead to more accurate answers, particularly when used with larger models like Llama-3

70B. While the two-phase approach offered more interpretable outputs, it sometimes struggled with precise column selection. Retrieval-based filtering proved to be more stable, especially for simpler, single-column questions. In future work, we aim to refine column selection with lightweight learning-based methods and extend our framework to handle more complex settings, including multi-table and multi-hop reasoning.

7 Limitations

Due to compute and memory constraints, we were unable to self-host Llama models and instead relied on external Groq APIs. This introduced additional limitations, such as strict token limits, which affected the amount of table data we could include in each prompt. As a result, we had to exclude some of the larger tables and carefully choose which parts of the data to include. This may have unintentionally biased our evaluation.

References

Abhijith Neil Abraham, Fariz Rahman, and Damanpreet Kaur. 2022. Tablequery: Querying tabular data with natural language. *Preprint*, arXiv:2202.00454.

Andreas Evangelatos, Giorgos Filandrianos, Maria Lymperaiou, Athanasios Voulodimos, and Giorgos Stamou. 2025. Ails-ntua at semeval-2025 task 8: Language-to-code prompting and error fixing for tabular question answering. *Preprint*, arXiv:2503.00435.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models(llms) on tabular data: Prediction, generation, and understanding – a survey. *Preprint*, arXiv:2402.17944.

Zijian Hei, Weiling Liu, Wenjie Ou, Juyi Qiao, Junming Jiao, Guowen Song, Ting Tian, and Yi Lin. 2024. Dr-rag: Applying dynamic document relevance to retrieval-augmented generation for question-answering. *Preprint*, arXiv:2406.07348.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13471–13488, Torino, Italia. ELRA and ICCL.

Feifei Pan, Mustafa Canim, Michael Glass, Alfio Gliozzo, and James Hendler. 2022. End-to-end table question answering via retrieval-augmented generation. *Preprint*, arXiv:2203.16714.

Jie Wu and Mengshu Hou. 2025. An efficient retrieval-based method for tabular prediction with LLM. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9917–9925, Abu Dhabi, UAE. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *Preprint*, arXiv:2005.08314.

A LLM Prompts

```
=== Baseline Prompt ===
[INST]
You are an assistant tasked with
    answering questions asked of a
    given dataset in JSON format.
You must answer in a single JSON with
    one field:
* "answer": your final answer based on
    the records.
Requirements:
* Only respond with the JSON. Do not
    include explanations or full
    objects.
* Your answer must use valid Python
    data types:
  - Use ‘True‘ or ‘False‘ (capitalized)
     for boolean values.
  - Use numbers as Python ‘int‘ or
     ‘float‘ (e.g., ‘3‘, ‘3.14‘).
  - Use double-quoted Python strings
     for categorical values (e.g.,
     "USA").
  - Use Python lists for answers
     involving multiple values:
    - For list[category], return a list
       of strings.
    - For list[number], return a list
       of ints or floats.
    - Ensure all inner values match the
       correct type.

In the following key-value formatted
    data:
‘‘‘kv
{'name': 'Bulbasaur', 'type1': 'Grass',
    'type2': 'Poison', 'hp': 45,
    'attack': 49, 'legendary': False}
{'name': 'Charmander', 'type1': 'Fire',
    'type2': None, 'hp': 39, 'attack':
    52, 'legendary': False}
{'name': 'Pikachu', 'type1':
    'Electric', 'type2': None, 'hp':
    35, 'attack': 55, 'legendary':
    False}
```

~~~
{'name': 'Zapdos', 'type1': 'Electric',
    'type2': 'Flying', 'hp': 90,
    'attack': 90, 'legendary': True}
USER: Which Pok mon have an attack
    value greater than 50?

ASSISTANT:
[/INST]

=== RAG Prompt ===
[INST]
You are an assistant tasked with
    answering questions asked of a
    given dataset in JSON format.
You must answer in a single JSON with
    two fields:
* "answer": your final answer based on
    the records.
* "columns_used": list of relevant
    columns.
Requirements:
* Only respond with the JSON. Do not
    include explanations or full
    objects.
* Your answer must use valid Python
    data types:
  - Use `True` or `False` (capitalized)
      for boolean values.
  - Use numbers as Python `int` or
      `float` (e.g., `3`, `3.14`).
  - Use double-quoted Python strings
      for categorical values (e.g.,
      "USA").
  - Use Python lists for answers
      involving multiple values:
    - For list[category], return a list
        of strings.
    - For list[number], return a list
        of ints or floats.
    - Ensure all inner values match the
        correct type.

In the following key-value formatted
    data:
```kv
{'name': 'Bulbasaur', 'type1': 'Grass',
    'type2': 'Poison', 'hp': 45,
    'attack': 49, 'legendary': False}
{'name': 'Charmander', 'type1': 'Fire',
    'type2': None, 'hp': 39, 'attack':
    52, 'legendary': False}
{'name': 'Pikachu', 'type1':
    'Electric', 'type2': None, 'hp':
    35, 'attack': 55, 'legendary':
    False}
{'name': 'Zapdos', 'type1': 'Electric',
    'type2': 'Flying', 'hp': 90,
    'attack': 90, 'legendary': True}
USER: Which Pok mon have an attack
    value greater than 50?
ASSISTANT:
[/INST]

=== 2-Phase Prompt (Phase 1)===
You are given a natural language
    question and a dictionary of table
    columns with their data types.
Your task is to select the names of the
    columns that are most relevant for
    answering the question.
Select only the minimal set of columns
    needed to answer the question
    accurately.
Return the result as a Python list of
    strings. Do not include any
    explanation.
The question is delimited by <question>
    and </question>.

<question>
Which countries had a GDP higher than
    1000 in 2020?
</question>

Columns:
{
  "Country": "string",
  "GDP_2020": "float",
  "Population": "int",
  "Region": "string",
  "Unemployment_Rate": "float"
}

=== 2-Phase Prompt (Phase 2)===
[INST]
You are an assistant tasked with
    answering questions asked of a
    given dataset in JSON format.
You must answer in a single JSON with
    one field:
* "answer": your final answer based on
    the records.
Requirements:
* Only respond with the JSON. Do not
    include explanations or full
    objects.
* Your answer must use valid Python
    data types:
  - Use `True` or `False` (capitalized)
      for boolean values.
  - Use numbers as Python `int` or
      `float` (e.g., `3`, `3.14`).
  - Use double-quoted Python strings
      for categorical values (e.g.,
      "USA").
  - Use Python lists for answers
      involving multiple values:
    - For list[category], return a list
        of strings.
    - For list[number], return a list
        of ints or floats.
    - Ensure all inner values match the
        correct type.

In the following key-value formatted
    data:
```kv
{'Country': 'USA', 'GDP_2020': 21000}
{'Country': 'India', 'GDP_2020': 2900}
{'Country': 'Nepal', 'GDP_2020': 300}
USER: Which countries had a GDP higher
    than 1000 in 2020?
ASSISTANT:
[/INST]
~~~

B Column Selection Accuracy

Table 3 – Table 9 present a comparison of column selection quality across different approaches. Each

table reports the percentage of wrongly and correctly selected columns, along with any formatting errors, for all 59 datasets.

Table 3: Baseline - Llama3-8B-CSV

Model	Wrong	Right	Error
001_Forbes	16.0	84.0	0.0
002_Titanic	10.0	90.0	0.0
004_Taxi	10.0	65.0	25.0
005_NYC	5.0	85.0	10.0
006_London	20.0	75.0	5.0
007_Fifa	5.0	85.0	10.0
008_Tornados	5.0	85.0	10.0
009_Central	5.0	90.0	5.0
010_ECommerce	5.0	75.0	20.0
011_SF	10.0	75.0	15.0
012_Heart	0.0	85.0	15.0
013_Roller	30.0	60.0	10.0
015_Food	15.0	80.0	5.0
016_Holiday	10.0	80.0	10.0
017_Hacker	15.0	75.0	10.0
018_Staff	5.0	90.0	5.0
019_Aircraft	25.0	55.0	20.0
021_Telco	15.0	80.0	5.0
022_Airbnbs	5.0	90.0	5.0
023_Climate	50.0	35.0	15.0
024_Salary	10.0	75.0	15.0
025_Data	40.0	55.0	5.0
026_Predicting	0.0	100.0	0.0
027_Supermarket	20.0	75.0	5.0
028_Predict	5.0	90.0	5.0
029_NYTimes	20.0	70.0	10.0
030_Professionals	20.0	55.0	25.0
031_Trustpilot	10.0	70.0	20.0
032_Delicatessen	10.0	90.0	0.0
033_Employee	25.0	70.0	5.0
034_World	25.0	60.0	15.0
036_US	30.0	65.0	5.0
037_Ted	10.0	70.0	20.0
038_Stroke	0.0	100.0	0.0
039_Happy	35.0	55.0	10.0
040_Speed	25.0	55.0	20.0
041_Airline	5.0	90.0	5.0
042_Predict	10.0	75.0	15.0
043_Predict	10.0	90.0	0.0
044_IMDb	10.0	85.0	5.0
045_Predict	15.0	75.0	10.0
046_120	20.0	80.0	0.0
047_Bank	0.0	100.0	0.0
048_Data	10.0	85.0	5.0
050_ING	20.0	65.0	15.0
051_Pokemon	35.0	55.0	10.0
052_Professional	20.0	45.0	35.0
053_Patents	55.0	40.0	5.0
055_German	0.0	90.0	10.0
056_Emoji	35.0	60.0	5.0
057_Spain	20.0	60.0	20.0
058_US	35.0	65.0	0.0
059_Second	10.0	75.0	15.0
060_Bakery	35.0	45.0	20.0
061_Disneyland	10.0	85.0	5.0
062_Trump	40.0	55.0	5.0
063_Influencers	40.0	50.0	10.0
064_Clustering	15.0	65.0	20.0
065_RFM	30.0	70.0	0.0

Table 4: Baseline - Llama3-8B-KV

Model	**Wrong**	**Right**	**Error**
001_Forbes	16.0	76.0	8.0
002_Titanic	25.0	75.0	0.0
004_Taxi	5.0	95.0	0.0
005_NYC	10.0	90.0	0.0
006_London	15.0	85.0	0.0
007_Fifa	5.0	95.0	0.0
008_Tornados	5.0	85.0	10.0
009_Central	0.0	100.0	0.0
010_ECommerce	0.0	100.0	0.0
011_SF	20.0	75.0	5.0
012_Heart	0.0	95.0	5.0
013_Roller	30.0	65.0	5.0
015_Food	10.0	85.0	5.0
016_Holiday	15.0	85.0	0.0
017_Hacker	10.0	65.0	25.0
018_Staff	0.0	95.0	5.0
019_Aircraft	20.0	75.0	5.0
021_Telco	15.0	85.0	0.0
022_Airbnbs	10.0	90.0	0.0
023_Climate	45.0	55.0	0.0
024_Salary	0.0	100.0	0.0
025_Data	35.0	65.0	0.0
026_Predicting	5.0	75.0	20.0
027_Supermarket	50.0	50.0	0.0
028_Predict	5.0	90.0	5.0
029_NYTimes	10.0	80.0	10.0
030_Professionals	40.0	50.0	10.0
031_Trustpilot	10.0	85.0	5.0
032_Delicatessen	15.0	85.0	0.0
033_Employee	15.0	75.0	10.0
034_World	25.0	65.0	10.0
036_US	25.0	75.0	0.0
037_Ted	0.0	95.0	5.0
038_Stroke	5.0	85.0	10.0
039_Happy	20.0	75.0	5.0
040_Speed	30.0	55.0	15.0
041_Airline	10.0	85.0	5.0
042_Predict	10.0	85.0	5.0
043_Predict	0.0	100.0	0.0
044_IMDb	45.0	55.0	0.0
045_Predict	5.0	90.0	5.0
046_120	15.0	85.0	0.0
047_Bank	5.0	90.0	5.0
048_Data	15.0	85.0	0.0
050_ING	20.0	70.0	10.0
051_Pokemon	20.0	80.0	0.0
052_Professional	10.0	75.0	15.0
053_Patents	45.0	45.0	10.0
055_German	0.0	95.0	5.0
056_Emoji	5.0	90.0	5.0
057_Spain	0.0	95.0	5.0
058_US	25.0	60.0	15.0
059_Second	10.0	80.0	10.0
060_Bakery	35.0	55.0	10.0
061_Disneyland	20.0	75.0	5.0
062_Trump	10.0	75.0	15.0
063_Influencers	10.0	80.0	10.0
064_Clustering	25.0	75.0	0.0
065_RFM	15.0	80.0	5.0

Table 5: Baseline - Llama3-70B-KV

Model	**Wrong**	**Right**	**Error**
001_Forbes	12.0	88.0	0.0
002_Titanic	0.0	100.0	0.0
004_Taxi	15.0	85.0	0.0
005_NYC	0.0	100.0	0.0
006_London	15.0	85.0	0.0
007_Fifa	0.0	95.0	5.0
008_Tornados	0.0	95.0	5.0
009_Central	5.0	90.0	5.0
010_ECommerce	10.0	85.0	5.0
011_SF	5.0	95.0	0.0
012_Heart	0.0	100.0	0.0
013_Roller	15.0	80.0	5.0
015_Food	5.0	90.0	5.0
016_Holiday	5.0	95.0	0.0
017_Hacker	5.0	95.0	0.0
018_Staff	0.0	100.0	0.0
019_Aircraft	5.0	95.0	0.0
021_Telco	15.0	85.0	0.0
022_Airbnbs	10.0	90.0	0.0
023_Climate	5.0	95.0	0.0
024_Salary	10.0	85.0	5.0
025_Data	45.0	55.0	0.0
026_Predicting	0.0	95.0	5.0
027_Supermarket	0.0	100.0	0.0
028_Predict	0.0	100.0	0.0
029_NYTimes	0.0	100.0	0.0
030_Professionals	10.0	90.0	0.0
031_Trustpilot	5.0	95.0	0.0
032_Delicatessen	0.0	100.0	0.0
033_Employee	10.0	90.0	0.0
034_World	15.0	75.0	10.0
036_US	5.0	95.0	0.0
037_Ted	5.0	95.0	0.0
038_Stroke	0.0	100.0	0.0
039_Happy	10.0	90.0	0.0
040_Speed	5.0	95.0	0.0
041_Airline	5.0	95.0	0.0
042_Predict	0.0	100.0	0.0
043_Predict	0.0	95.0	5.0
044_IMDb	35.0	65.0	0.0
045_Predict	0.0	100.0	0.0
046_120	30.0	70.0	0.0
047_Bank	5.0	85.0	10.0
048_Data	20.0	75.0	5.0
050_ING	5.0	90.0	5.0
051_Pokemon	10.0	90.0	0.0
052_Professional	20.0	80.0	0.0
053_Patents	30.0	70.0	0.0
055_German	0.0	95.0	5.0
056_Emoji	5.0	90.0	5.0
057_Spain	0.0	95.0	5.0
058_US	5.0	90.0	5.0
059_Second	0.0	100.0	0.0
060_Bakery	15.0	80.0	5.0
061_Disneyland	5.0	95.0	0.0
062_Trump	0.0	100.0	0.0
063_Influencers	5.0	95.0	0.0
064_Clustering	15.0	80.0	5.0
065_RFM	10.0	85.0	5.0

Table 6: RAG - Llama3-8B-KV

Model	Wrong	Right	Error
001_Forbes	36.0	60.0	4.0
002_Titanic	25.0	70.0	5.0
004_Taxi	10.0	90.0	0.0
005_NYC	10.0	90.0	0.0
006_London	20.0	75.0	5.0
007_Fifa	0.0	100.0	0.0
008_Tornados	45.0	50.0	5.0
009_Central	10.0	90.0	0.0
010_ECommerce	10.0	90.0	0.0
011_SF	40.0	45.0	15.0
012_Heart	0.0	100.0	0.0
013_Roller	30.0	50.0	20.0
015_Food	10.0	90.0	0.0
016_Holiday	5.0	95.0	0.0
017_Hacker	25.0	70.0	5.0
018_Staff	15.0	85.0	0.0
019_Aircraft	35.0	55.0	10.0
021_Telco	20.0	75.0	5.0
022_Airbnbs	15.0	80.0	5.0
023_Climate	80.0	20.0	0.0
024_Salary	5.0	95.0	0.0
025_Data	45.0	50.0	5.0
026_Predicting	20.0	75.0	5.0
027_Supermarket	65.0	35.0	0.0
028_Predict	50.0	50.0	0.0
029_NYTimes	20.0	55.0	25.0
030_Professionals	40.0	45.0	15.0
031_Trustpilot	5.0	95.0	0.0
032_Delicatessen	15.0	85.0	0.0
033_Employee	20.0	70.0	10.0
034_World	30.0	60.0	10.0
036_US	50.0	45.0	5.0
037_Ted	5.0	90.0	5.0
038_Stroke	5.0	95.0	0.0
039_Happy	15.0	85.0	0.0
040_Speed	25.0	60.0	15.0
041_Airline	15.0	75.0	10.0
042_Predict	25.0	65.0	10.0
043_Predict	25.0	70.0	5.0
044_IMDb	35.0	55.0	10.0
045_Predict	15.0	80.0	5.0
046_120	30.0	65.0	5.0
047_Bank	10.0	90.0	0.0
048_Data	20.0	80.0	0.0
050_ING	5.0	80.0	15.0
051_Pokemon	45.0	55.0	0.0
052_Professional	15.0	75.0	10.0
053_Patents	20.0	70.0	10.0
055_German	25.0	70.0	5.0
056_Emoji	10.0	85.0	5.0
057_Spain	0.0	95.0	5.0
058_US	30.0	65.0	5.0
059_Second	15.0	85.0	0.0
060_Bakery	20.0	75.0	5.0
061_Disneyland	15.0	85.0	0.0
062_Trump	40.0	55.0	5.0
063_Influencers	35.0	60.0	5.0
064_Clustering	10.0	85.0	5.0
065_RFM	15.0	80.0	5.0

Table 7: RAG - Llama3-70B-KV

Model	Wrong	Right	Error
001_Forbes	28.0	72.0	0.0
002_Titanic	10.0	90.0	0.0
004_Taxi	5.0	95.0	0.0
005_NYC	5.0	95.0	0.0
006_London	30.0	70.0	0.0
007_Fifa	0.0	95.0	5.0
008_Tornados	40.0	50.0	10.0
009_Central	10.0	90.0	0.0
010_ECommerce	15.0	85.0	0.0
011_SF	30.0	70.0	0.0
012_Heart	0.0	100.0	0.0
013_Roller	55.0	45.0	0.0
015_Food	10.0	85.0	5.0
016_Holiday	5.0	95.0	0.0
017_Hacker	10.0	90.0	0.0
018_Staff	5.0	90.0	5.0
019_Aircraft	20.0	75.0	5.0
021_Telco	20.0	80.0	0.0
022_Airbnbs	0.0	100.0	0.0
023_Climate	80.0	20.0	0.0
024_Salary	10.0	90.0	0.0
025_Data	40.0	55.0	5.0
026_Predicting	20.0	80.0	0.0
027_Supermarket	45.0	55.0	0.0
028_Predict	35.0	60.0	5.0
029_NYTimes	0.0	100.0	0.0
030_Professionals	15.0	85.0	0.0
031_Trustpilot	5.0	95.0	0.0
032_Delicatessen	15.0	85.0	0.0
033_Employee	20.0	70.0	10.0
034_World	20.0	75.0	5.0
036_US	45.0	55.0	0.0
037_Ted	10.0	90.0	0.0
038_Stroke	0.0	100.0	0.0
039_Happy	15.0	85.0	0.0
040_Speed	20.0	80.0	0.0
041_Airline	15.0	85.0	0.0
042_Predict	25.0	75.0	0.0
043_Predict	20.0	75.0	5.0
044_IMDb	45.0	55.0	0.0
045_Predict	0.0	100.0	0.0
046_120	40.0	60.0	0.0
047_Bank	5.0	90.0	5.0
048_Data	20.0	65.0	15.0
050_ING	10.0	90.0	0.0
051_Pokemon	45.0	55.0	0.0
052_Professional	5.0	95.0	0.0
053_Patents	20.0	80.0	0.0
055_German	30.0	70.0	0.0
056_Emoji	10.0	85.0	5.0
057_Spain	0.0	95.0	5.0
058_US	10.0	80.0	10.0
059_Second	15.0	85.0	0.0
060_Bakery	25.0	75.0	0.0
061_Disneyland	5.0	95.0	0.0
062_Trump	20.0	75.0	5.0
063_Influencers	25.0	75.0	0.0
064_Clustering	0.0	95.0	5.0
065_RFM	5.0	95.0	0.0

Table 8: 2-Phase - Llama3-8B-KV

Model	**Wrong**	**Right**	**Error**
001_Forbes	72.0	24.0	4.0
002_Titanic	30.0	70.0	0.0
004_Taxi	80.0	20.0	0.0
005_NYC	70.0	30.0	0.0
006_London	55.0	45.0	0.0
007_Fifa	35.0	60.0	5.0
008_Tornados	70.0	30.0	0.0
009_Central	45.0	55.0	0.0
010_ECommerce	85.0	15.0	0.0
011_SF	60.0	35.0	5.0
012_Heart	25.0	75.0	0.0
013_Roller	75.0	25.0	0.0
015_Food	45.0	55.0	0.0
016_Holiday	55.0	45.0	0.0
017_Hacker	60.0	35.0	5.0
018_Staff	60.0	40.0	0.0
019_Aircraft	75.0	25.0	0.0
021_Telco	70.0	30.0	0.0
022_Airbnbs	35.0	65.0	0.0
023_Climate	60.0	40.0	0.0
024_Salary	35.0	65.0	0.0
025_Data	60.0	35.0	5.0
026_Predicting	5.0	90.0	5.0
027_Supermarket	65.0	35.0	0.0
028_Predict	25.0	75.0	0.0
029_NYTimes	70.0	30.0	0.0
030_Professionals	85.0	10.0	5.0
031_Trustpilot	50.0	50.0	0.0
032_Delicatessen	45.0	55.0	0.0
033_Employee	50.0	50.0	0.0
034_World	40.0	60.0	0.0
036_US	20.0	80.0	0.0
037_Ted	35.0	60.0	5.0
038_Stroke	5.0	95.0	0.0
039_Happy	80.0	20.0	0.0
040_Speed	45.0	50.0	5.0
041_Airline	25.0	65.0	10.0
042_Predict	30.0	70.0	0.0
043_Predict	10.0	90.0	0.0
044_IMDb	25.0	70.0	5.0
045_Predict	25.0	75.0	0.0
046_120	50.0	40.0	10.0
047_Bank	25.0	70.0	5.0
048_Data	55.0	45.0	0.0
050_ING	60.0	40.0	0.0
051_Pokemon	50.0	50.0	0.0
052_Professional	20.0	75.0	5.0
053_Patents	65.0	35.0	0.0
055_German	15.0	85.0	0.0
056_Emoji	35.0	65.0	0.0
057_Spain	45.0	55.0	0.0
058_US	85.0	15.0	0.0
059_Second	45.0	55.0	0.0
060_Bakery	80.0	20.0	0.0
061_Disneyland	35.0	65.0	0.0
062_Trump	65.0	35.0	0.0
063_Influencers	50.0	50.0	0.0
064_Clustering	35.0	60.0	5.0
065_RFM	65.0	35.0	0.0

Table 9: 2-Phase - Llama3-70B-KV

Model	**Wrong**	**Right**	**Error**
001_Forbes	36.0	64.0	0.0
002_Titanic	0.0	100.0	0.0
004_Taxi	10.0	90.0	0.0
005_NYC	40.0	60.0	0.0
006_London	30.0	70.0	0.0
007_Fifa	5.0	95.0	0.0
008_Tornados	15.0	85.0	0.0
009_Central	50.0	50.0	0.0
010_ECommerce	35.0	65.0	0.0
011_SF	45.0	55.0	0.0
012_Heart	0.0	100.0	0.0
013_Roller	50.0	50.0	0.0
015_Food	20.0	80.0	0.0
016_Holiday	10.0	90.0	0.0
017_Hacker	35.0	65.0	0.0
018_Staff	25.0	75.0	0.0
019_Aircraft	60.0	40.0	0.0
021_Telco	25.0	75.0	0.0
022_Airbnbs	35.0	65.0	0.0
023_Climate	15.0	85.0	0.0
024_Salary	0.0	95.0	5.0
025_Data	45.0	55.0	0.0
026_Predicting	0.0	100.0	0.0
027_Supermarket	20.0	80.0	0.0
028_Predict	5.0	95.0	0.0
029_NYTimes	0.0	100.0	0.0
030_Professionals	15.0	85.0	0.0
031_Trustpilot	10.0	90.0	0.0
032_Delicatessen	25.0	75.0	0.0
033_Employee	15.0	85.0	0.0
034_World	10.0	90.0	0.0
036_US	10.0	90.0	0.0
037_Ted	15.0	85.0	0.0
038_Stroke	0.0	100.0	0.0
039_Happy	25.0	75.0	0.0
040_Speed	20.0	80.0	0.0
041_Airline	40.0	60.0	0.0
042_Predict	0.0	100.0	0.0
043_Predict	0.0	100.0	0.0
044_IMDb	0.0	100.0	0.0
045_Predict	5.0	95.0	0.0
046_120	20.0	80.0	0.0
047_Bank	15.0	85.0	0.0
048_Data	45.0	55.0	0.0
050_ING	20.0	80.0	0.0
051_Pokemon	15.0	85.0	0.0
052_Professional	25.0	75.0	0.0
053_Patents	15.0	85.0	0.0
055_German	0.0	100.0	0.0
056_Emoji	60.0	40.0	0.0
057_Spain	5.0	95.0	0.0
058_US	35.0	65.0	0.0
059_Second	15.0	85.0	0.0
060_Bakery	45.0	55.0	0.0
061_Disneyland	20.0	80.0	0.0
062_Trump	30.0	70.0	0.0
063_Influencers	15.0	85.0	0.0
064_Clustering	50.0	50.0	0.0
065_RFM	35.0	65.0	0.0