TablePilot: Recommending Human-Preferred Tabular Data Analysis with Large Language Models

Deyin Yi[1*], Yihao Liu[2*], Lang Cao[3*],
Mengyu Zhou[4†], Haoyu Dong[4], Shi Han[4], Dongmei Zhang[4]
[1]Shanghai University of Finance and Economics [2]Peking University
[3]University of Illinois Urbana-Champaign [4]Microsoft

Abstract

Tabular data analysis is crucial in many scenarios, yet efficiently identifying relevant queries and results for new tables remains challenging due to data complexity, diverse analytical operations, and high-quality analysis requirements. To address these challenges, we aim to recommend query–code–result triplets tailored for new tables in tabular data analysis workflows. In this paper, we present TablePilot, a pioneering tabular data analysis framework leveraging large language models to autonomously generate comprehensive and superior analytical results without relying on user profiles or prior interactions. Additionally, we propose Rec-Align, a novel method to further improve recommendation quality and better align with human preferences. Experiments on DART, a dataset specifically designed for comprehensive tabular data analysis recommendation, demonstrate the effectiveness of our framework. Based on GPT-4o, the tuned TablePilot achieves 77.0% top-5 recommendation recall. Human evaluations further highlight its effectiveness in optimizing tabular data analysis workflows.

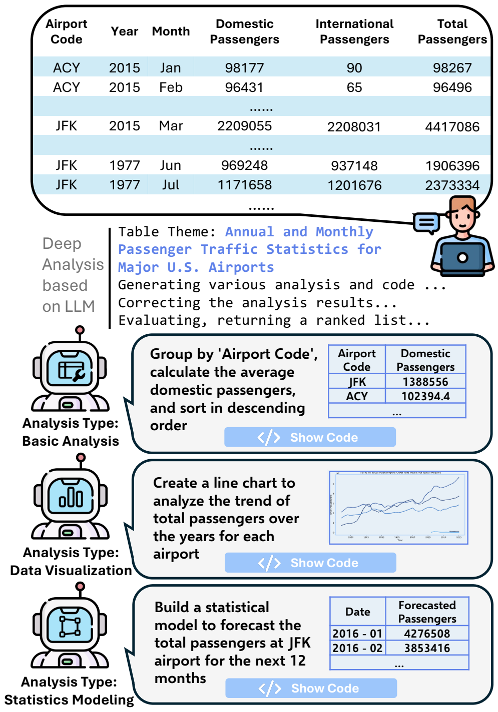


Figure 1: Overview of TablePilot. Through deep analysis based on LLM, TablePilot generates three types of analysis: basic analysis, data visualization, and statistical modeling, each presented as a <query, code, result> triplet.

1 Introduction

Tabular data is widely used in various data analysis scenarios (Ghasemi and Amyot, 2016; Li et al., 2021). However, its complexity and density (Cao, 2025; Tian et al., 2024) can make it challenging, even for professional analysts, to determine the most appropriate analysis operations for a new table. Conducting tabular data analysis is often tedious, and the analysis operations may include errors that lead to suboptimal outcomes. Therefore, automatically recommending high-quality analysis queries and results becomes essential in the data analysis workflow, particularly in zero-turn scenarios where no user profile or historical records are available.

In the task of tabular data analysis recommendation, given only a table as input, we aim to recommend query–code–result triplets to users. A query specifies the type and objective of the analysis task, while the code executes the corresponding operation on the table, serving as an intermediate step in the analysis. The result presents the execution output, which also constitutes the analysis findings.

Previous works on tabular data analysis recommendation (Zhou et al., 2020, 2021) primarily rely on traditional machine learning methods but often exhibit suboptimal performance and strong depen-

* Work during internship at Microsoft.
† Corresponding author (mezho@microsoft.com).

dence on specific datasets. Recently, large language models (LLMs) (OpenAI, 2024; Touvron et al., 2023) have made significant strides in natural language processing. With their advanced data processing, language comprehension, and generation capabilities, LLMs present new opportunities for delivering more effective tabular data analysis recommendations.

In practical data analysis scenarios, these triplets are expected to be (a) **accurate**, (b) **diverse**, and (c) **human-preferred**. Human-preferred refers to the data analysis operations that humans genuinely intend to perform, meaning the results should be meaningful, insightful and so on. Employing LLMs to recommend tabular data analyses while meeting these requirements presents several key challenges.

Challenges: (a) Tabular data is often large and data-intensive, making it difficult for LLMs to process effectively. Long-context windows can trigger hallucinations (Huang et al., 2024), leading to inaccurate results. (b) Existing approaches primarily construct workflows around single-operation scenarios, executing predefined analytical queries to obtain results (Fang et al., 2024; Zhang et al., 2025), but they lack diversity and fail to deliver comprehensive analyses. (c) Selecting and presenting analysis results in a way that aligns with human cognitive patterns is crucial (Song et al., 2024; Dai et al., 2023; Yu et al., 2024). A well-designed system should balance diversity and quality in recommending data analysis operations that match users' analytical preferences, ensuring the insights generated are interpretable, actionable, and meaningful.

Solution: To address these challenges, we propose **TablePilot**, a framework designed to tackle the zero-turn recommendation task for tabular data analysis, as illustrated in Figure 1.

To enhance the **accuracy** of analysis results, we adopt sampling techniques (Sui et al., 2024; Ye et al., 2023b; Ji et al., 2024), employing a table sampler to refine model inputs and introducing a table explanation component that incorporates world knowledge learned during the pretraining phase of LLMs. This stage of analysis preparation facilitates the generation of more contextually appropriate queries and results. At the optimization level, we utilize post-refinement techniques (Chen, 2022; He et al., 2024) to adjust outputs. However, instead of focusing solely on code refinement, we identify multiple aspects of query and result optimization.

To improve the **diversity** of our analysis, we implement a modularized approach to support various workflow operations. This modular design provides two key benefits. First, it ensures comprehensive coverage by enabling the workflow to handle a diverse range of data analysis tasks, making it more adaptable to various requirements. Second, it enhances performance by allowing each module to be trained independently for better efficiency, with improvements across modules contributing to overall effectiveness.

To ensure our analysis aligns with **human preferences**, we introduce **Rec-Align**, a method specifically designed to further enhance the quality of analysis by directly incorporating human preferences. We train a ranking model to optimize the final set of recommended operations, ensuring they align with human analytical tendencies and produce superior results.

We contribute a dataset **DART** to support and validate our framework. Experimental results demonstrate that the tuned TablePilot achieves nearly 100% execution rates, while the analysis modules show an overall recall improvement of 11.25% with GPT-4o in the dataset. Rec-Align further enhances alignment with human preferences, leading to gains of 6.8% in Recall@3 and 6.0% in Recall@5. Additionally, human evaluations confirm that the TablePilot framework provides more practical and insightful data analysis recommendations compared to baseline models. Extensive experiments validate the effectiveness of TablePilot and our training approach.

In summary, our main technical contributions are as follows:

- We propose TablePilot, a framework for zero-turn recommendation in tabular data analysis, encompassing a comprehensive set of analytical operations. We also contribute DART, a dataset to support and validate our framework.
- We introduce two additional steps to enhance the accuracy of analysis results, applied before and after core analysis. These steps incorporate sampling, explanation, and multi-faceted refinement.
- We develop Rec-Align, a method designed to align recommendations with human analytical preferences, further enhancing the quality and practical utility of the recommended results.

2 Related Work

Current tabular data analysis recommendation tasks can be categarized into three main types:

Basic Data Analysis in Tables. Basic analysis refers to simple, initial processing of a table. It involves generating tabular outputs or single-cell text entries to highlight key information or insights based on a user query. This is usually done by manipulating and aggregating tabular data. Table understanding tasks (Pasupat and Liang, 2015; Chen et al., 2020) are the most basic form of this analysis. Given a query, these tasks either provide an answer or extract a sub-table (Wang et al., 2024; Ye et al., 2023a) that contains important information. TableMaster (Cao, 2025) offers a general recipe for table understanding and basic analysis based on user queries. Text2SQL (Pourreza and Rafiei, 2024; Gao et al., 2023; Lee et al., 2024; Zhao et al., 2024) is another approach that extracts relevant parts of a table by converting user queries into SQL-based outputs. However, these methods only return results based on a given query and do not generate natural language queries automatically. Auto-Formula (Chen et al., 2024) predicts and suggests formula syntax for spreadsheet-based analysis. Table2Analysis (Zhou et al., 2020) and MetaInsight (Ma et al., 2021) automatically recommends common analysis without requiring user input.

Tabular Data Visualization. Visualizing data helps users quickly understand complex patterns and relationships. Table2Charts (Zhou et al., 2021) applies sequence token sampling and reinforcement learning to recommend different chart types. Furmanova et al. (Furmanova et al., 2019) developed a tool for automatically combining overview and details in tabular data visualizations. AdaVis (Zhang et al., 2023) uses knowledge graphs to adaptively recommend one or multiple suitable visualizations for a dataset. LLMs have further improved data visualization. Chart2VIS (Maddigan and Susnjak, 2023) leverages LLMs for natural language-to-visualization tasks by generating Python code for chart creation. ChartLlama (Han et al., 2023), a multi-modal LLM, shows strong chart generation capabilities but does not recommend charts based on existing data.

Statistical Modeling of Tabular Data Statistical modeling in tabular data focuses on building models to recognize patterns and relationships. RIM (Qin et al., 2021) enhances tabular data prediction with a retrieval module. GReaT (Borisov et al., 2022) uses a decoder-only transformer to model data distributions and generate realistic synthetic data. GTL (Wen et al., 2024) integrates LLMs with deep learning techniques for regression and classification tasks. TabDDPM (Kotelnikov et al., 2024) is a diffusion model that can handle any tabular dataset and support various feature types.

Despite these advancements, most existing methods are task-specific and do not support multiple types of analysis within a single framework. This limitation prevents users from obtaining a comprehensive view of their data. Currently, no unified system seamlessly integrates table analysis, visualization, and statistical modeling. A complete all-in-one framework would allow users to explore data more effectively from different perspectives. Moreover, existing methods primarily emphasize the accuracy of analysis results while neglecting the importance of aligning with human analytical preferences.

3 Methodology

3.1 Task Formulation

Tabular Data Analysis Recommendation. In the task of tabular data analysis recommendation, the objective is to generate a series of recommended data analysis queries q, corresponding code c, and execution results r for a given table $\mathbb{T}$ under a zero-turn setting (i.e., with no user profile or historical context). The table $\mathbb{T}_{a\times b}$ contains a rows and b columns, where $C_{i,j}$ denotes the cell in the i-th row and j-th column. For each table $\mathbb{T}$, n analysis results A is recommended in triplets:

$$A = \left\{ \left(q_i,\, c_i,\, r_i\right) \right\}_{i=1}^{n}, \tag{1}$$

where each triplet $a = (q,\, c,\, r)$ represents a single recommended analysis result.

3.2 Framework

To generate recommendation results from a given table, we propose TablePilot, a four-step analysis framework, as illustrated in Figure 2. The framework consists of Analysis Preparation, Module-based Analysis, Analysis Optimization, and Analysis Ranking. A new table T is provided as input to generate the recommended results A.

$$\text{TablePilot}(\mathbb{T}) = A. \tag{2}$$

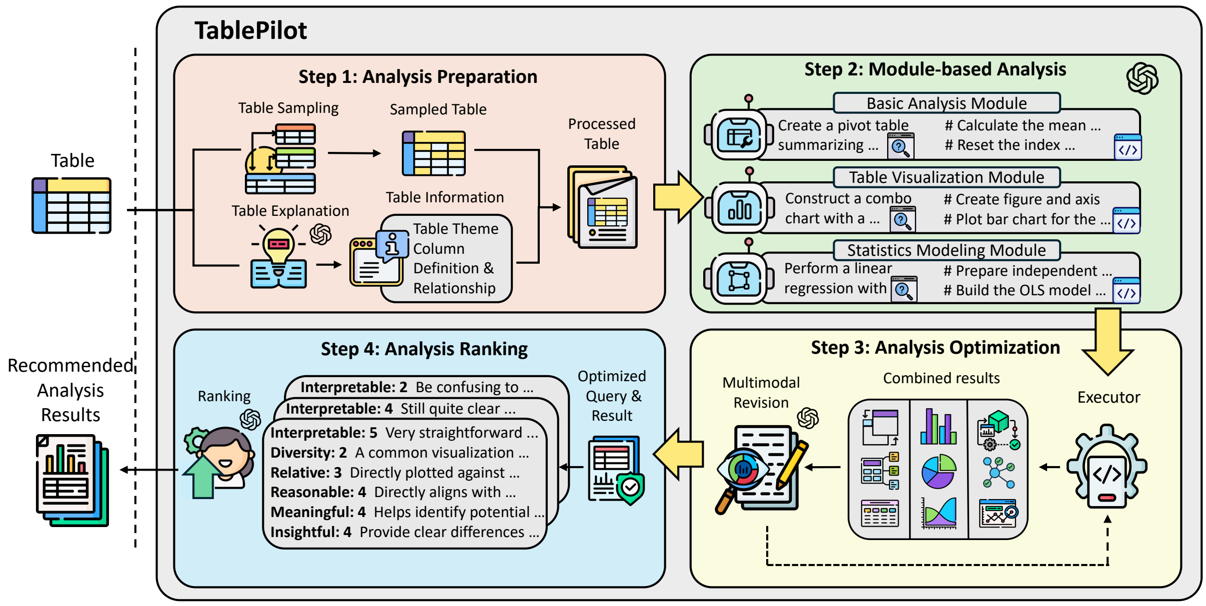


Figure 2: The TablePilot framework. Step 1: Sample the input table and generate corresponding explanations for its structure and content. Step 2: Generate query and code for modules involving basic analysis, table visualization, and statistics modeling. Step 3: Optimize the quality of <query, code, result> triplets. Step 4: Score and rank the optimized results based on multiple criteria to recommend the top-K analysis. TablePilot Case Study and Analysis Report can be seen at Appendix K and Appendix L.

Step 1: Analysis Preparation. The objective of this step is to transform raw tabular data into a more focused form that facilitates efficient analysis. This step involves two key tasks: sampling a subset of the table and generating a table explanation.

Raw tables often contain large amounts of data, much of which may not be relevant for a specific analysis task. Sampling extracts a representative subset of the table, capturing essential patterns while reducing computational load and focusing the analysis on key data points. This process involves selecting a subset of rows from the original table:

$$\text{Sampling}(\mathbb{T}_{a\times b}) = \mathbb{T}'_{a'\times b'}, \tag{3}$$

where $\mathbb{T}'$ represents the sampled table, a' denotes the number of selected rows, and b' denotes the number of selected columns.

Additionally, generating a table explanation is crucial for structuring the data, making column relationships and the table's overall theme clearer and more interpretable. This explanation includes metadata such as the table's theme, column descriptions, and relationships between different columns, all of which guide subsequent analysis. The explanation is denoted as E:

$$\text{Explanation}(\mathbb{T}) = E. \tag{4}$$

Step 2: Module-based Analysis. In this step, we perform a module-based analysis on the sampled table $\mathbb{T}'$ and its corresponding table explanation E. The goal is to generate analysis results by applying specialized modules to different aspects of the data. These modules focus on basic analysis (ba), data visualization (dv), and statistical modeling (sm). Each module takes $\mathbb{T}'$ and E as inputs to generate meaningful query-code pairs (q, c):

$$\mathcal{M}_k(\mathbb{T}', E) = (q_k, c_k), \tag{5}$$

where $k \in \{ba, dv, sm\}$ represents the three different analysis task.

The Basic Analysis module ($\mathcal{M}_{ba}$) applies fundamental yet powerful techniques to explore the data, performing operations such as filtering, grouping, sorting, and aggregation. The Data Visualization module ($\mathcal{M}_{dv}$) generates visual representations of the data to reveal patterns, trends, and relationships. The Statistical Modeling module ($\mathcal{M}_{sm}$) applies advanced statistical techniques to analyze the data and uncover deeper insights. It may involve regression analysis, hypothesis testing, or predictive modeling, depending on the analysis objectives.

Step 3: Analysis Optimization In this step, we first execute the code to obtain results r for each code c_k:

$$\text{Execution}(\mathbb{T}, c_k) = r = \begin{cases} T, & \text{if } k = ba \\ V, & \text{if } k = dv \\ M, & \text{if } k = sm \end{cases} \quad (6)$$

where T represents the sub-table after data manipulation in basic analysis, V denotes the result of data visualization, and M corresponds to the output of statistical modeling. The result of data visualization, $r = V$, is also an image $r = I$, which will be used as input for the vision module of LLMs at a later stage. We then combine the query q, code c, and result r into an analysis triplet $a = (q, c, r)$. The results $r =$ Error indicate an error in the code execution.

Next, we refine the analysis triplet a based on the results from table sampling $\mathbb{T}$ and explanation E. The optimization process utilizes LLMs to improve the alignment of queries and code with the data and analysis intent, ensuring more accurate and meaningful results. There are two different strategies for LLMs to optimize triplets, depending on whether the result contains an error. After refinement, the optimized code is executed to generate the final enhanced execution results, yielding an optimized triplet $a' = (q', c', r')$:

$$a' = \begin{cases} \text{Optimize}_A(q, c, r \mid \mathbb{T}, E), & \text{if } r \neq \text{Error} \\ \text{Optimize}_B(q, c, r \mid \mathbb{T}, E), & \text{if } r = \text{Error} \end{cases} \quad (7)$$

Step 4: Analysis Ranking In the final step, the objective is to evaluate and rank all the (q, c, r) triplets $A = a_{i\,i=1}^{\,n}$ that were generated and optimized in the previous step. To achieve this, we design a ranking module that scores each triplet based on multiple dimensions, such as relevance, diversity, and other key factors (criteria detailed in Appendix H). These scores are then aggregated to compute an overall score s. Using these scores, the triplets are ranked in descending order, allowing us to select the top k results:

$$A'_k = \text{Top}_k\Big(\text{Rank}\Big(\big\{(q', c', r')_i\big\}_{i=1}^{n}\Big)\Big) \quad (8)$$

After scoring, ranking, and selecting the top-k results A'_k, the final triplets are recommended to users.

3.3 Training

The training process in TablePilot is designed to enhance the model's ability to generate high-quality analysis results, with a focus on accurate query-code generation and human-preferred ranking of analysis triplets $a = (q, c, r)$. We primarily employ Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) (introduced in Appendix J), both widely used techniques for tuning LLMs. SFT is used to ensure that each module follows our instructions for performing tasks. Additionally, we introduce **Rec-Align**, implemented via DPO, to enhance our ranking module, further refining recommendation quality and ensuring that the selected results align more closely with human preferences.

Our training strategy consists of the following key components:

- **Analysis SFT** trains the LLMs in three analysis module ($\mathcal{M}_{ba}$, $\mathcal{M}_{dv}$, $\mathcal{M}_{sm}$) to improve their ability to follow instructions, generating relevant queries and accurate code. This enhances the accuracy of the analysis.
- **Rank SFT** trains the LLMs in the ranking module *Rank* to better follow instructions in evaluating each analysis triplet based on comprehensive criteria and assigning appropriate scores. This ensures that the ranking model adheres to our guidelines when ranking triplets..
- **Rank DPO** implements Rec-Align through DPO to refine the evaluation of analysis triplets in *Rank*, ensuring that evaluation and scoring are more closely aligned with human analytical preferences. This further enhances the quality of the recommended analysis.

4 Experiments

4.1 Experiment Settings

To support, validate the framework, and evaluate its performance, we carefully curate a dataset, **DART**. Details on the dataset can be found in Appendix C.

In our experiments, we evaluate the performance of TablePilot on three typical analysis tasks: Basic Analysis, Data Visualization, and Statistics Modeling. Additionally, we consider them collectively without distinction for the overall evaluation. We aim to evaluate the result of query–code–result triplets for a given table. To assess the quality of code generation, we use the execution rate as a metric. For the quality of the final results in recommendation, we evaluate using Recall@K. Detailed evaluation metrics can be found in Appendix B.

We selected three state-of-the-art vision-

Method	Basic Analysis			Data Visualization			Statistics Modeling			Overall		
	R@3	R@5	R@N	R@3	R@5	R@N	R@3	R@5	R@N	R@3	R@5	R@N
GPT-4o												
Baseline	13.00	20.11	42.00	17.57	26.30	53.40	15.08	27.08	56.67	38.11	52.11	80.00
Vanilla	14.05	21.07	50.67	35.84	48.81	69.37	15.48	38.91	59.58	53.51	70.90	87.67
Analysis SFT + Rank Vanilla	15.67	22.33	55.33	43.88	53.06	70.41	20.00	30.42	61.25	59.00	72.67	89.00
Analysis SFT + Rank SFT	15.67	28.00	55.33	41.84	53.06	70.41	21.25	38.33	61.25	58.00	74.33	89.00
Analysis SFT + Rank SFT-V	15.33	25.67	55.33	**44.22**	54.42	70.41	16.25	45.83	61.25	61.00	75.00	89.00
Analysis SFT + Rank SFT & DPO	**19.33**	**30.00**	55.33	42.86	52.72	70.41	20.42	42.08	61.25	61.33	76.00	89.00
Analysis SFT + Rank SFT-V & DPO	17.67	26.00	55.33	43.88	**54.78**	70.41	**22.92**	**47.08**	61.25	**63.00**	**77.00**	89.00
GPT-4o-mini												
Baseline	15.99	24.94	35.33	27.33	39.33	44.22	3.61	6.67	35.33	29.33	42.44	62.67
Vanilla	8.67	10.67	38.33	40.48	50.34	56.12	5.54	10.83	38.33	45.33	56.67	78.33
Analysis SFT + Rank Vanilla	13.00	**57.14**	46.67	44.22	25.33	64.29	1.67	10.42	59.58	52.00	68.67	85.00
Analysis SFT + Rank SFT	**24.91**	34.33	46.67	34.15	45.24	64.29	12.02	32.08	59.58	56.66	71.67	85.00
Analysis SFT + Rank SFT-V	16.00	24.33	46.67	**46.60**	**54.08**	64.29	22.50	43.33	59.58	61.00	75.00	85.00
Analysis SFT + Rank SFT & DPO	21.33	32.67	46.67	42.86	50.34	64.29	16.25	27.05	59.58	60.33	73.67	85.00
Analysis SFT + Rank SFT-V & DPO	21.00	29.00	46.67	40.14	51.02	64.29	**22.92**	**49.17**	58.58	**62.33**	**76.67**	85.00
Phi-3.5-vision												
Baseline	3.00	4.00	5.00	1.36	3.40	4.08	0.00	0.00	0.42	4.33	7.00	8.67
Vanilla	1.43	1.79	13.33	1.83	1.83	3.74	3.12	3.12	7.92	5.73	6.09	21.67
Analysis SFT + Rank Vanilla	3.77	3.77	24.00	**3.83**	**4.53**	9.52	18.45	19.31	32.50	20.89	21.58	47.67
Analysis SFT + Rank SFT	**6.85**	14.04	24.00	2.79	4.18	9.52	15.88	22.75	32.50	20.89	32.19	47.67
Analysis SFT + Rank SFT-V	5.14	13.01	24.00	1.74	3.14	9.52	**19.31**	21.89	32.50	21.23	30.14	47.67
Analysis SFT + Rank SFT & DPO	**8.90**	**15.07**	24.00	1.74	3.83	9.52	18.88	23.61	32.50	**23.97**	**32.88**	47.67
Analysis SFT + Rank SFT-V & DPO	7.53	14.38	24.00	1.74	2.09	9.52	**19.31**	**25.32**	32.50	23.63	32.19	47.67

Table 1: Recall across multiple models and experimental settings (all values in %). Experimental results demonstrate the effectiveness of TablePilot, with *Analysis SFT + Rank SFT-V & DPO* generally achieving the best performance.

language models of varying sizes and availability, including both private and open-source options, as the foundation models: *GPT-4o*, *GPT-4o-mini*, and *Phi-3.5-Vision*. These models were chosen for their strong vision-language interaction capabilities, making them well-suited for multi-modal refinement.

We conduct multiple comparative experiments to comprehensively evaluate performance. The *baseline* experiments exclude all components of our proposed framework, relying on a single prompt to generate queries and code across all three task categories, with recall computed via random ranking. In contrast, *vanilla* experiments employ TablePilot without additional model tuning. Subsequent experiments examine different components of TablePilot, incorporating tuning methods such as SFT and DPO. The definitions of *Analysis SFT*, *Rank SFT*, and *Rank DPO* are detailed in Section 3. *Rank Vanilla* represents random ranking over three rounds, while the -V notation denotes the inclusion of vision input during training.

We then compare these experimental results to assess the impact of each tuning strategy on overall performance improvements. Detailed experimental settings are provided in Appendix D, and the corresponding prompts are listed in Appendix M.

Method	ExecRate		
	Basic Analysis	Data Visualization	Statistics Modeling
GPT-4o			
Baseline	96.07	95.00	95.00
Vanilla	99.67	99.67	**99.44**
Analysis SFT	**100.00**	**99.93**	99.33
GPT-4o-mini			
Baseline	91.37	88.75	56.11
Vanilla	96.32	97.80	92.76
Analysis SFT	**99.40**	**99.66**	**98.73**
Phi-3.5-vision			
Baseline	44.17	26.65	10.83
Vanilla	77.03	57.55	65.78
Analysis SFT	**87.80**	**99.28**	**85.11**

Table 2: Execution rate across multiple models and experimental settings (all values in %)

4.2 Main Results

TablePilot Performance. As illustrated in Table 5 and Table 6, TablePilot delivers substantial performance improvements across various models without the need for fine-tuning LLMs. In particular, *GPT-4o* benefits from TablePilot, exhibiting improvements across all key metrics, with a 4.24% increase in execution rate and recall at different thresholds, especially make 18.79% gain in Recall@5 which is considered as the most balanced metric. These consistent gains across all tasks demonstrate the method's effectiveness in enhancing LLM performance without manual adjustments or additional tuning.

Notably, we observe some performance drops in certain analysis among three analysis tasks. This is due to a **diverse analysis trade-off effect**, where an excessively high recall in one task may lead to a decline in recall for others. Therefore, overall recall serves as a more reliable measure of the method's overall performance.

TablePilot Performance after Tuning. Supervised Fine-Tuning significantly improves both analysis and ranking tasks. Vision-enabled SFT further enhances ranking performance, especially when combined with DPO applied to vision components. While *GPT-4o* sees modest gains over the vanilla workflow, GPT-4o-mini improves by 10–20% on average, with some cases reaching 20 points. *Phi-3.5-vision* shows the most notable improvement, exceeding 20% on average, with rank@N increasing by 26%. These results highlight the importance of tuning in optimizing TablePilot, ensuring alignment with human values for more robust and valuable outputs.

Supervised Fine-Tuning significantly improves both analysis and ranking tasks. Vision-enabled SFT further enhances ranking performance, especially when combined with DPO applied to vision components, resulting in a 9.49% boost in Recall@3 and 6.10% in Recall@5 for *GPT-4o*. The most pronounced improvements are observed in smaller language models, as detailed in Appendix E. These results underscore the importance of fine-tuning in optimizing TablePilot, ensuring alignment with human preferences for more robust and valuable outputs.

Ablation Study. Each components of the TablePilot workflow (Sampling, Explanation, Revision, and Ranking) contributes to a consistent improvement in the overall system performance. The complete results of the ablation experiments are presented in Appendix F.

4.3 Analysis of Rec-Align

Our proposed Rec-Align, implemented via DPO, consistently improves model performance across various configurations and tasks by enhancing alignment with user expectations in ethical and qualitative aspects. *GPT-4o* benefits from a 2% increase in Recall@3 and Recall@5, while smaller models exhibit even greater performance gains after applying Rec-Align, as shown in Table 5. We also observed that in the vanilla ranking mode, some models initially exhibit scoring biases toward specific tasks like data visualization. Rec-Align mitigates this imbalance, resulting in a more diverse ranked list and guiding models to generate outputs that better reflect human preferences.

4.4 Human Evaluation

Rating	5	4	3	2	1	Avg	≥ 4	≥ 3	≤ 2
Baseline	47	71	92	46	44	3.10	118	210	90
TablePilot (Vanilla)	114	61	79	25	21	3.74	175	254	46
TablePilot (Tuned)	146	75	48	28	3	**4.11**	**221**	**269**	**31**

Table 3: Results of human evaluation ratings

Table 3 presents human evaluation results on 300 test tables from DART. TablePilot (Tuned) achieves the highest mean score, the largest proportion of high-rated outputs (ratings ≥ 3), and the lowest proportion of low-rated outputs (rating ≤ 2). The Wilcoxon signed-rank test(Wilcoxon et al., 1963) confirms significant improvements at a 95% confidence level, supporting the effectiveness of TablePilot and Rec-Align in enhancing recommendation quality. Further details on evaluation methodology and criteria are provided in Appendix I.

5 Conclusion

In this paper, we introduce TablePilot, a comprehensive data analysis recommendation framework powered by large language models. Extensive experiments demonstrate TablePilot 's superior performance, marking a new milestone in tabular data analysis recommendation.

Limitations

Our work presents an exploratory study on comprehensive tabular data analysis, with several limitations including workflow fragmentation, limited interactivity, and constraints of DPO. For further discussion on the extendability of TablePilot and future directions, please refer to Appendix A.

References

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. 2022. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*.

Lang Cao. 2025. Tablemaster: A recipe to advance table understanding with language models. *Preprint*, arXiv:2501.19378.

Sibei Chen, Yeye He, Weiwei Cui, Ju Fan, Song Ge, Haidong Zhang, Dongmei Zhang, and Surajit Chaudhuri. 2024. Auto-formula: Recommend formulas in spreadsheets using contrastive learning for table representations. *Proceedings of the ACM on Management of Data*, 2(3):1–27.

Wenhu Chen. 2022. Large language models are few (1)-shot table reasoners. *arXiv preprint arXiv:2210.06710*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*.

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

Katarina Furmanova, Samuel Gratzl, Holger Stitz, Thomas Zichner, Miroslava Jaresova, Alexander Lex, and Marc Streit. 2019. Taggle: Combining overview and details in tabular data visualizations. *Information Visualization*, 19(2):114–136.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Mahdi Ghasemi and Daniel Amyot. 2016. Process mining in healthcare: a systematised literature review. *International Journal of Electronic Healthcare*, 9(1):60–88.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*.

Xinyi He, Jiaru Zou, Yun Lin, Mengyu Zhou, Shi Han, Zejian Yuan, and Dongmei Zhang. 2024. Cocost: Automatic complex code generation with online searching and correctness testing. *arXiv preprint arXiv:2403.13583*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2024. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Deyi Ji, Lanyun Zhu, Siqi Gao, Peng Xu, Hongtao Lu, Jieping Ye, and Feng Zhao. 2024. Tree-of-table: Unleashing the power of llms for enhanced large-scale table understanding. *arXiv preprint arXiv:2411.08516*.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2024. Tabddpm: Modelling tabular data with diffusion models. *Preprint*, arXiv:2209.15421.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *arXiv preprint arXiv:2405.07467*.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763*.

Yiren Li, Zheng Huang, Junchi Yan, Yi Zhou, Fan Ye, and Xianhui Liu. 2021. Gfte: graph-based financial table extraction. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part II*, pages 644–658. Springer.

Pingchuan Ma, Rui Ding, Shi Han, and Dongmei Zhang. 2021. Metainsight: Automatic discovery of structured knowledge for exploratory data analysis. In *Proceedings of the 2021 international conference on management of data*, pages 1262–1274.

Paula Maddigan and Teo Susnjak. 2023. Chat2vis: generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *Ieee Access*, 11:45181–45193.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *Preprint*, arXiv:1508.00305.

Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36.

Jiarui Qin, Weinan Zhang, Rong Su, Zhirong Liu, Weiwen Liu, Ruiming Tang, Xiuqiang He, and Yong Yu. 2021. Retrieval & interaction machine for tabular data prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1379–1389.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Yuzhang Tian, Jianbo Zhao, Haoyu Dong, Junyu Xiong, Shiyu Xia, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, et al. 2024. Spreadsheetllm: encoding spreadsheets for large language models. *arXiv preprint arXiv:2407.09025*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *Preprint*, arXiv:2401.04398.

Xumeng Wen, Han Zhang, Shun Zheng, Wei Xu, and Jiang Bian. 2024. From supervised to generative: A novel paradigm for tabular deep learning with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3323–3333.

F. Wilcoxon, S.K. Katti, and R.A. Wilcox. 1963. *Critical Values and Probability Levels for the Wilcoxon Rank Sum Test and the Wilcoxon Signed Rank Test*. American Cyanamid.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023a. Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning. *Preprint*, arXiv:2301.13808.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023b. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184.

Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwen He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, et al. 2024. Rlhf-v: Towards trustworthy mllms via behavior alignment from fine-grained correctional human feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13807–13816.

Songheng Zhang, Haotian Li, Huamin Qu, and Yong Wang. 2023. Adavis: Adaptive and explainable visualization recommendation for tabular data. *Preprint*, arXiv:2310.11742.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. A survey of table reasoning with large language models. *Frontiers of Computer Science*, 19(9):199348.

Wei Zhao, Zhitao Hou, Siyuan Wu, Yan Gao, Haoyu Dong, Yao Wan, Hongyu Zhang, Yulei Sui, and Haidong Zhang. 2024. Nl2formula: Generating spreadsheet formulas from natural language queries. *arXiv preprint arXiv:2402.14853*.

Mengyu Zhou, Qingtao Li, Xinyi He, Yuejiang Li, Yibo Liu, Wei Ji, Shi Han, Yining Chen, Daxin Jiang, and Dongmei Zhang. 2021. Table2charts: recommending charts by learning shared table representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2389–2399.

Mengyu Zhou, Wang Tao, Ji Pengxin, Han Shi, and Zhang Dongmei. 2020. Table2analysis: Modeling and recommendation of common analysis patterns for multi-dimensional data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 320–328.

Contents of Appendix

A Extendability Analysis and Future Works 11

B Evaluation Metrics 11

C DART Dataset details 12

D Detailed Experiment Settings 13

E Complete Experiment Results 13

F Ablation Study 14

G Analysis of Incorporating Vision in Training 14

H Ranking Criteria 15

I Human Evaluation 16

J Direct Preference Optimization 17

K Case Study 18

L TablePilot Report Generation 27

M Prompt Design 31

A Extendability Analysis and Future Works

In this paper, we present an exploratory study on comprehensive tabular data analysis. Several important extensions of our proposed framework, TablePilot, remain open for future work.

Data Curation. We provide the dataset DART to support model training and to validate the performance of TablePilot. However, the current dataset has several limitations: it is relatively small in scale, lacks image-contrastive data necessary for effective multi-modal SFT and DPO, and contains limited high-quality samples. We believe that with more carefully curated data and improved data construction pipelines, TablePilot could achieve significantly better performance and enable more powerful analytical capabilities.

Multi-Modal Training. One significant direction for extending TablePilot lies in the integration of multi-modal GPT-based models, such as multi-modal SFT and DPO. As previously mentioned, higher-quality multi-modal training data is crucial for achieving better performance. In addition, current GPT-series models on the Azure platform do not yet support multi-modal DPO, limiting our ability to fully leverage visual information during optimization. Multi-modal DPO could substantially improve TablePilot 's ability to evaluate and analyze results based on figures and visualizations. Furthermore, how to design multi-stage training pipelines that balance SFT and DPO to achieve optimal model performance remains an open challenge. We believe that, with the integration of more advanced multi-modal capabilities, TablePilot can generate richer analytical insights, enhance contextual understanding, and better align with how human analysts interpret complex, heterogeneous data sources.

Analysis Modularization. The current version of TablePilot supports three types of analysis: Basic Analysis, Table Visualization, and Statistical Modeling. These analyses are implemented in a modularized manner, allowing flexible composition and extension. As these three represent some of the most classical forms of tabular data analysis, they provide a strong foundation for various use cases. In the future, more diverse or specialized analysis modules can be easily integrated into TablePilot, showcasing the flexibility of our framework. Furthermore, in different downstream application scenarios, TablePilot can adaptively select and combine specialized analysis modules to better address domain-specific needs.

System Internal Interaction. The current framework of TablePilot is unidirectional, with different analysis modules operating in parallel without internal interaction. In the future, we aim to extend TablePilot into a multi-agent system, enabling richer interactions between modules. For example, different analysis modules could complement and enrich each other's data, and the ranking module could provide feedback to guide the analysis modules. We believe that such a design would make the system more intelligent and capable of generating higher-quality analytical recommendations.

Efficiency Optimization. Our current TablePilot framework involves multiple large language model (LLM) calls, which can lead to efficiency issues. In the future, we plan to improve efficiency by replacing certain modules with smaller language models or well-trained traditional machine learning models. Additionally, optimizing and compressing prompts will help streamline the pipeline and further enhance overall efficiency.

B Evaluation Metrics

In our experiments, we adopt two primary metrics to evaluate system performance comprehensively: *Execution Rate* (abbreviated as *ExecRate*) and *Recall*.

ExecRate quantifies the accuracy and stability of generated code by measuring whether it executes without error and returns the expected output. This metric is consistently applied across all modules (Basic Analysis, Table Visualization, and Statistical Modeling) by calculating the ratio of successful executions to the total number of generated outputs.

Recall serves as our key indicator for retrieval accuracy, assessing whether the correct result appears among the top-ranked candidates. We distinguish among three variants: *Recall@All*, *Recall@5*, and *Recall@3*. *Recall@All* checks if the correct result is present anywhere in the candidate set, while *Recall@5* and *Recall@3* evaluate if it ranks within the top five and top three candidates, respectively.

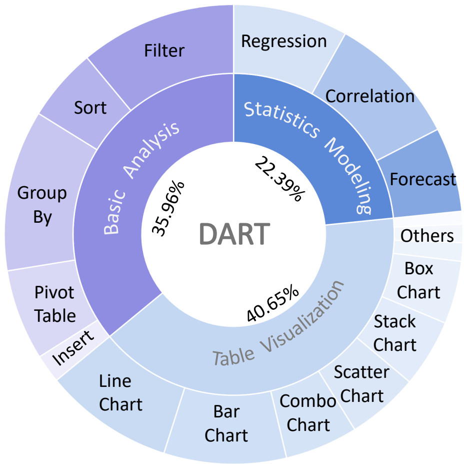


Figure 3: Statistics of the test split of the DART dataset. We can categorize data analysis tasks into three major groups: Basic Analysis (35.96%), Table Visualization (40.65%), and Statistics Modeling (22.39%). This distribution highlights the diversity of analytical tasks covered in the dataset.

For Basic Analysis, success is defined by an exact match of the output table to the expected result. In Table Visualization, the evaluation hinges on the precise match of generated chart information—including x_fields, y_fields, and chart_type. For Statistical Modeling, evaluation is further subdivided into Regression, Correlation, and Forecast tasks. Specifically, Regression is deemed successful if the *Mean Absolute Percentage Error* (*MAPE*) is ≤ 1.0, Correlation if the *p-value* is < 0.05, and Forecast if the *R-squared value* is > 0.9, with the additional requirement that the table column relationships align with the expected structure. These metrics ensures a robust assessment of both execution reliability and the system's ability to prioritize accurate results.

C DART Dataset details

To support and validate the performance of TablePilot, we conducted an investigation on the table dataset DART (representing **D**ata **A**nalysis **R**ecommendation for **T**ables). Existing datasets, such as those in the Text2SQL domain (Xu et al., 2018; Lei et al., 2024), which focus on SQL-like analytical QA tasks, and the Table2Charts domain (Han et al., 2023; Zhou et al., 2021), which specializes in table-to-chart QA and conversion, are designed for specific domains rather than comprehensive analysis. Additionally, even common analysis datasets like Text2Analysis (Zhou et al., 2020) are primarily designed for TableQA scenarios, making them misaligned with our proposed task of zero-turn data analysis recommendation. As a result, we constructed a custom dataset to better support our tasks. To the best of our knowledge, DART is the first dataset dedicated to recommending comprehensive tabular data analysis operations.

Our dataset construction process was inspired by Table2Charts (Zhou et al., 2021), which contains a collection of real-world tables. We leveraged these tables as a foundation for synthetic data generation, ensuring that the dataset retained realistic tabular structures while expanding its applicability to our target tasks. The data generation process was conducted using *o1* and consisted of three main step:

1. **Table Selection.** We filtered the tables from Table2Charts, selecting those that were most suitable for data analysis tasks with strong tabular structures. This selection process ensured that the tables contained sufficient variability in structure, numerical and categorical distributions, and contextual relevance for analytical queries.

2. **Label Generation.** For each of the three tasks, *o1* was employed to generate a diverse set of queries and their corresponding code implementations. The queries were designed to cover a range of complexity levels, from simple transformations to advanced statistical modeling tasks. The code snippets were generated in Python, incorporating libraries such as `Pandas`, `Matplotlib`, and `StatsModels`, ensuring their practical applicability. However, from all the generated queries and code, we carefully selected only those that were able to successfully produce the expected results.

3. **Human Evaluation.** We manually curated a subset of 300 tables to ensure diversity in structure and analytical needs. From the generated triplets, we selected those that met specific criteria for clarity, correctness, and so on based on human preferences. This process resulted in a test set that reflects real-world analytical tasks. The test set was then used to evaluate the model's performance, particularly through metric recall, providing a robust benchmark for TablePilot 's capabilities. Finally, DART consists of 300 data from different tables. The dataset distribution is shown in Figure 3.

Model	Parameter	Supervised Fine-Tuning (SFT)	Direct Preference Optimization (DPO)
GPT-4o / GPT-4o-mini	Learning Rate	1×10^{-6}	5×10^{-7}
	Number of Epochs	6	2
	Batch Size	64	32
Phi-3.5-Vision	Learning Rate	1×10^{-5}	5×10^{-7}
	Number of Epochs	3	2
	Batch Size	8	8
	Full-Parameter Training	Yes	No

Table 4: Training Parameters for GPT-4o, GPT-4o-mini, and Phi-3.5-Vision Models

D Detailed Experiment Settings

We use the `trl` package to fine-tune open-source models on a workstation equipped with 4 × A100 Nvidia GPUs. LoRA fine-tuning (Hu et al., 2021) is applied to train GPT-series models on the Azure platform [1], leveraging its scalable infrastructure. The models used in our experiments include *GPT-4o* (gpt-4o-08-06), *GPT-4o-mini* (gpt-4o-mini-2024-07-18), and *Phi-3.5-Vision* (microsoft/Phi-3.5-vision-instruct). OpenAI *o1* used in our study are *o1-2024-12-17*. The detailed training parameters can be found in Table 4.

For inference, the parameters are set as follows for all models, including both open-source and private models: temperature is 0, max tokens is 6000, top-p is 0.95, frequency penalty is 0, presence penalty is 0, and stop is set to None. All other settings are configured to their default values. Inference stage is also conducted in 4 × A100 Nvidia GPUs.

In the SFT phase, we used *o1* to generate a batch of data tailored to the task requirements. To ensure the quality of the data, we employed LLM-based evaluation along with manual sampling. For fine-tuning the module-based analysis, we used 800 training samples and 100 validation samples for both the basic analysis and table visualization modules. Due to the complexity of its tasks, the statistics modeling module was trained using 1,100 samples, with 100 samples reserved for validation.

In the DPO phase, we first performed an SFT run on the ranking module using 342 ranked samples generated by *o1*. Afterward, DPO training was conducted with 1,000 positive and negative samples. The positive samples consisted of ranking results generated by *o1*, which were manually adjusted based on preference calibration. The negative samples were disordered ranking results produced by *GPT-4o-mini*.

[1]https://azure.microsoft.com/en-us/

E Complete Experiment Results

This section presents the complete experimental results of TablePilot, covering Recall at different thresholds (Recall@3, Recall@5, and Recall@N) as well as the Execution Rate across the three analysis modules.

Recall. Following the application of the TablePilot framework, *GPT-4o-mini* exhibited significant improvements, achieving enhanced results across all three analysis tasks and demonstrating strong potential in overall Recall@N with a notable increase of 15.66%. Similarly, *Phi-3.5-vision* also realized comprehensive gains, securing a 13.00% improvement in overall Recall@N. After training with SFT and DPO, TablePilot further improved upon the vanilla framework. Notably, *Phi-3.5-vision* achieved increases of 15.33% in Basic Analysis, 16.19% in Data Visualization, and 24.58% in Statistical Modeling. With the integration of Rec-Align, *GPT-4o-mini* achieved peak improvements of 10.33% and 8.00% for Recall@3 and Recall@5, respectively, while *Phi-3.5-vision* showed maximum gains of 3.08% and 11.30%.

Extensive experimental results confirm that incorporating vision-based training enhances the model's performance in recall by integrating additional dimensions of information. However, after introducing vision-based training to *Phi-3.5-vision*, its ranking performance declined. Our analysis indicates that this drop is due to a gap introduced by model pretrained ability, which was validated through comparative experiments. Detailed instructions are provided in Appendix G.

Execution Rate. The execution rate of the generated query code demonstrated a steady improvement following optimization with the TablePilot framework. *GPT-4o-mini* achieved an execution

Method	Basic Analysis			Data Visualization			Statistics Modeling			Overall		
	R@3	R@5	R@N	R@3	R@5	R@N	R@3	R@5	R@N	R@3	R@5	R@N
GPT-4o												
Baseline	13.00	20.11	42.00	17.57	26.30	53.40	15.08	27.08	56.67	38.11	52.11	80.00
Vanilla	14.05	21.07	50.67	35.84	48.81	69.37	15.48	38.91	59.58	53.51	70.90	87.67
Analysis SFT + Rank Vanilla	15.67	22.33	55.33	43.88	53.06	70.41	20.00	30.42	61.25	59.00	72.67	89.00
Analysis SFT + Rank SFT	15.67	28.00	55.33	41.84	53.06	70.41	21.25	38.33	61.25	58.00	74.33	89.00
Analysis SFT + Rank SFT-V	15.33	25.67	55.33	**44.22**	54.42	70.41	16.25	45.83	61.25	61.00	75.00	89.00
Analysis SFT + Rank SFT & DPO	**19.33**	**30.00**	55.33	42.86	52.72	70.41	20.42	42.08	61.25	61.33	76.00	89.00
Analysis SFT + Rank SFT-V & DPO	17.67	26.00	55.33	43.88	**54.78**	70.41	**22.92**	**47.08**	61.25	**63.00**	**77.00**	89.00
GPT-4o-mini												
Baseline	15.99	24.94	35.33	27.33	39.33	44.22	3.61	6.67	35.33	29.33	42.44	62.67
Vanilla	8.67	10.67	38.33	40.48	50.34	56.12	5.54	10.83	38.33	45.33	56.67	78.33
Analysis SFT + Rank Vanilla	13.00	**57.14**	46.67	44.22	25.33	64.29	1.67	10.42	59.58	52.00	68.67	85.00
Analysis SFT + Rank SFT	**24.91**	34.33	46.67	34.15	45.24	64.29	12.02	32.08	59.58	56.66	71.67	85.00
Analysis SFT + Rank SFT-V	16.00	24.33	46.67	**46.60**	**54.08**	64.29	22.50	43.33	59.58	61.00	75.00	85.00
Analysis SFT + Rank SFT & DPO	21.33	32.67	46.67	42.86	50.34	64.29	16.25	27.05	59.58	60.33	73.67	85.00
Analysis SFT + Rank SFT-V & DPO	21.00	29.00	46.67	40.14	51.02	64.29	**22.92**	**49.17**	58.58	**62.33**	**76.67**	85.00
Phi-3.5-vision												
Baseline	3.00	4.00	5.00	1.36	3.40	4.08	0.00	0.00	0.42	4.33	7.00	8.67
Vanilla	1.43	1.79	13.33	1.83	1.83	3.74	3.12	3.12	7.92	5.73	6.09	21.67
Analysis SFT + Rank Vanilla	3.77	3.77	24.00	**3.83**	**4.53**	9.52	18.45	19.31	32.50	20.89	21.58	47.67
Analysis SFT + Rank SFT	**6.85**	14.04	24.00	2.79	4.18	9.52	15.88	22.75	32.50	20.89	32.19	47.67
Analysis SFT + Rank SFT-V	5.14	13.01	24.00	1.74	3.14	9.52	**19.31**	21.89	32.50	21.23	30.14	47.67
Analysis SFT + Rank SFT & DPO	**8.90**	**15.07**	24.00	1.74	3.83	9.52	18.88	23.61	32.50	**23.97**	**32.88**	47.67
Analysis SFT + Rank SFT-V & DPO	7.53	14.38	24.00	1.74	2.09	9.52	**19.31**	**25.32**	32.50	23.63	32.19	47.67

Table 5: Recall across multiple models and experimental settings (all values in %). Experimental results demonstrate the effectiveness of TablePilot, with *Analysis SFT + Rank SFT-V & DPO* generally achieving the best performance.

Method	ExecRate		
	Basic Analysis	Data Visualization	Statistics Modeling
GPT-4o			
Baseline	96.07	95.00	95.00
Vanilla	99.67	99.67	**99.44**
Analysis SFT	**100.00**	**99.93**	99.33
GPT-4o-mini			
Baseline	91.37	88.75	56.11
Vanilla	96.32	97.80	92.76
Analysis SFT	**99.40**	**99.66**	**98.73**
Phi-3.5-vision			
Baseline	44.17	26.65	10.83
Vanilla	77.03	57.55	65.78
Analysis SFT	**87.80**	**99.28**	**85.11**

Table 6: Execution rate across multiple models and experimental settings (all values in %)

rate close to 100% across all three analysis tasks, while *Phi-3.5-vision* exhibited the most significant gains among all models. Notably, its execution rate increased by 41.73% in Data Visualization and 19.33% in Statistical Modeling.

F Ablation Study

The ablation study results are presented in Table 7 and Table 8. In this experiments, we examine the contributions of key components within the TablePilot workflow, specifically assessing the impact of the Table Explanation, Revision, and Ranking modules on the quality of generated data analysis recommendations. The baseline results represent a system without any of these modules.

Experimental results indicate that nearly all designed components contribute to performance improvements in TablePilot. However, some performance drops can also be attributed to the **diverse analysis trade-off effect**.

G Analysis of Incorporating Vision in Training

Incorporating vision into the training process proves both valuable and effective. For GPT-4o and GPT-4o-mini, the addition of vision capabilities significantly enhances the ranking module. Compared to pure text-based ranking, these models show improved recall@3 and recall@5 metrics.

Method	Basic Analysis		Data Visualization		Statistics Modeling		Overall Recall@N
	ExecRate	Recall@N	ExecRate	Recall@N	ExecRate	Recall@N	
Vanilla	99.67	**50.67**	**99.67**	**69.37**	99.44	59.58	**87.67**
w/o sampling	98.04 (-1.63)	48.67 (-2.00)	98.20 (-1.47)	65.31 (-4.06)	98.53 (-0.91)	58.75 (-0.83)	86.00 (-1.67)
w/o sampling & revision	93.27 (-6.40)	39.00 (-11.67)	93.20 (-6.47)	63.61 (-5.76)	86.62 (-12.82)	53.75 (-5.83)	82.00 (-5.67)
w/o explanation	**99.93** (+0.26)	46.00 (-4.67)	99.27 (-0.40)	63.61 (-5.76)	**99.56** (+0.12)	**62.08** (+2.50)	83.67 (-4.00)
w/o explanation & revision	99.33 (-0.34)	38.33 (-12.34)	97.47 (-2.20)	62.24 (-7.13)	96.89 (-2.55)	49.58 (-10.00)	79.67 (-8.00)
w/o sampling & explanation	99.60 (-0.07)	38.67 (-12.00)	97.33 (-2.34)	62.59 (-6.78)	98.44 (-1.00)	49.17 (-10.41)	83.00 (-4.67)
w/o all	94.73 (-4.94)	39.67 (-11.00)	93.87 (-5.80)	37.76 (-31.61)	89.19 (-10.25)	45.83 (-13.75)	71.67 (-16.00)

Table 7: Impact of removing several components on ExecRate and Recall@N across different tasks (all values in %)

Method	Basic Analysis		Data Visualization		Statistics Modeling		Overall	
	Recall@5	Recall@3	Recall@5	Recall@3	Recall@5	Recall@3	Recall@5	Recall@3
ranking	21.07	14.05	48.81	35.84	28.91	15.48	70.90	53.51
w/o ranking	16.67 (-4.40)	11.56 (-2.49)	39.80 (-9.01)	23.36 (-12.48)	22.92 (-5.99)	15.00 (-0.48)	57.33 (-13.57)	40.22 (-13.29)

Table 8: Impact of removing ranking on Recall@K across different tasks (all values in %)

Specifically, in the Table Visualization Task, GPT-4o-mini demonstrates a 9% increase in recall@5 and a 12% increase in recall@3, which contributes substantially to the overall improvements of 5% in recall@3 and 4% in recall@5. Due to its smaller scale and comparatively weaker multimodal capabilities relative to GPT-4o, GPT-4o-mini benefits even more from multimodal training in enhancing its ranking ability.

Conversely, Phi-3.5-vision does not benefit from multimodal training; in fact, its performance declines. This decline is primarily attributed to the poor quality of table visualizations generated by Phi-3.5. While we trained the ranking model on high-quality ranking data generated by GPT-4o and GPT-4o-mini, which in turn produced abundant high-quality analysis data, Phi-3.5 generated relatively few examples of data with lower quality. This data disparity, coupled with the inherent limitations of Phi-3.5, makes it challenging for the model to effectively learn to rank lower-quality data, ultimately resulting in reduced performance.

To verify that Phi-3.5-vision indeed learns to rank multimodal triplets after multimodal SFT, we conducted an experiment using GPT-4o–generated triplets as the basis for ranking, as detailed in Table 9. Our evaluation indicates that employing the multimodal SFT–enhanced Phi-3.5-vision as the ranking module yields an overall recall boost of 3% to 5%. Furthermore, in multimodal scenarios—particularly in the Table Visualization task—Phi-3.5-vision achieves an average increase of 6.8% in recall@3 and recall@5. These findings suggest that while Phi-3.5-vision demonstrates robust multimodal ranking capabilities, its overall performance is nevertheless limited by the suboptimal quality of the triplets it generates.

H Ranking Criteria

TablePilot employs a structured prompt with explicit criteria to filter and rank data analysis recommendations using an LLM. The core ranking criteria include:

1. **Meaningfulness**: Recommendations must offer impactful, insightful queries rather than trivial data representations. Queries should directly facilitate actionable insights.
2. **Relevance**: Recommendations must align closely with the Table Theme, ensuring analytical coherence with the dataset's core objective.
3. **Logical Coherence**: Recommendations must follow fundamental data analysis principles, accurately reflecting logical relationships and dataset characteristics.
4. **Diversity**: Ensures a broad coverage of analytical tasks across basic operations, data visualization, and advanced analyses to maximize insight comprehensiveness.
5. **Interpretability**: Recommendations should be clear, concise, and easily implementable by analysts without ambiguity.
6. **Insightfulness**: Prioritizes queries revealing non-obvious patterns, trends, and relationships that significantly enhance understanding of the data.

Analysis	Phi-3.5-vision Rank	Basic Analysis		Table Visualization		Statistics Modeling		Overall	
		Recall@3	Recall@5	Recall@3	Recall@5	Recall@3	Recall@5	Recall@3	Recall@5
Phi-3.5-vision	Rank SFT	6.85	14.04	2.79	4.18	15.88	22.75	20.89	32.19
	Rank SFT-V	5.14 (-1.71)	13.01 (-1.03)	1.74 (-1.05)	3.14 (-1.04)	19.31 (+3.43)	21.89 (-0.86)	21.23 (+0.34)	30.14 (-2.05)
	Rank SFT & DPO	8.90	15.07	1.74	3.83	18.88	23.61	23.97	32.88
	Rank SFT-V & DPO	7.53 (-1.37)	14.38 (-0.69)	1.74 (0.00)	2.09 (-1.74)	19.31 (+0.43)	25.32 (+1.71)	23.63 (-0.34)	32.19 (-0.69)
GPT-4o	Rank SFT	14.67	24.00	20.07	28.57	13.75	20.00	39.57	52.00
	Rank SFT-V	10.76 (-3.91)	20.33 (-3.67)	26.87 (+6.80)	35.71 (+7.14)	13.75 (0.00)	23.75 (+3.75)	42.00 (+2.43)	55.33 (+3.33)
	Rank SFT & DPO	12.67	23.00	21.77	30.27	12.08	20.42	38.33	53.33
	Rank SFT-V & DPO	17.00 (+4.33)	25.33 (+2.33)	27.21 (+5.44)	38.10 (+7.83)	13.33 (+1.25)	18.75 (-1.67)	46.00 (+7.67)	60.00 (+6.67)

Table 9: Performance on Recall@3 and Recall@5 with different Phi-3.5-vision Rank

Additional task-specific constraints are applied to further refine the recommendations, eliminating redundancy, trivial analyses, and logically unsound operations. This structured ranking criteria, embedded within a unified prompt and processed through an LLM, ensures the efficient selection and prioritization of high-quality analytical queries that align with professional analytical standards.

I Human Evaluation

Automatic quantitative evaluation of tabular data analysis recommendations has inherent limitations, as it typically relies on predefined metrics that may not fully capture nuances such as practical relevance, clarity, or interpretability. Therefore, we complemented the automatic evaluation with a human evaluation study, ensuring a comprehensive assessment of recommendation quality. Specifically, we recruited domain experts and experienced data analysts to manually evaluate the recommendations produced by different variants of our method, namely, the baseline, TablePilot (Vanilla), and TablePilot (Tuned).

The evaluation was structured around three critical qualitative dimensions:

1. **Practicality** – Assesses whether recommended operations are genuinely valuable and feasible in realistic data analysis contexts, capturing the degree to which recommendations meet actual analyst needs beyond general relevance and meaningfulness. High practicality implies direct applicability to specific user workflows and domain-specific analysis scenarios, aspects not fully addressed by broader criteria like relevance or meaningfulness.

2. **Clarity** – Measures the explicitness and transparency of the recommended queries and results, ensuring analysts can effortlessly grasp their intent and execution details. This dimension emphasizes immediate understandability and user-friendly phrasing, aspects that extend beyond the logical coherence and interpretability criteria defined in automated ranking, by explicitly capturing the communicative quality and unambiguity.

3. **Interpretability** – Evaluates the ease with which analysts can explain, justify, and utilize the recommended analysis results in practice. This dimension specifically highlights the analysts' ability to contextualize insights in stakeholder communication and practical decision-making, aspects distinct from automatic criteria like insightfulness or logical coherence, which do not inherently ensure communicative ease or effective translation of insights into actionable outcomes.

Evaluators consisted of five professional data analysts, each having extensive experience in interpreting tabular data. To ensure consistency and objectivity, the evaluators were provided detailed instructions and standardized scoring criteria, assessing each recommendation independently under these three dimensions using a 5-point Likert scale (1 = Poor, 5 = Excellent).

To ensure robust comparison of results across methods, we employed the Wilcoxon signed-rank test (Wilcoxon et al., 1963), a robust non-parametric test designed to assess differences between paired observations without assuming normal data distribution. The test ranks the absolute differences between paired scores, evaluating if observed differences between methods are statistically significant or due merely to chance variations. In our evaluation, we applied the Wilcoxon test at a significance level of $\alpha = 0.05$.

The results from the Wilcoxon signed-rank test demonstrated statistically significant improvements for TablePilot (Tuned) over both the baseline and TablePilot (Vanilla), as well as for TablePilot (Vanilla) over the baseline. Specifically,

TablePilot (Tuned) showed significantly enhanced performance across all evaluation metrics, confirming the effectiveness of our tuning process based on human preferences.

J Direct Preference Optimization

Direct Preference Optimization (DPO) (Rafailov et al., 2023) is a reinforcement learning-free approach for fine-tuning large language models (LLMs) using human preferences. Given preference-labeled data pairs $\{(x, y^+, y^-)\}$, where y^+ is the preferred response and y^- is the less preferred response for input x, DPO optimizes the policy $\pi_\theta(y|x)$ by maximizing the implicit reward function derived from the Bradley-Terry model:

$$r_\theta(x, y^+) - r_\theta(x, y^-) = \log \frac{\pi_\theta(y^+|x)}{\pi_\theta(y^-|x)}$$

The loss function for DPO is formulated as:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y^+,y^-)}\left[\log \sigma \left(\beta \log \frac{\pi_\theta(y^+|x)}{\pi_\theta(y^-|x)}\right)\right]$$

where σ is the sigmoid function and β is a scaling factor that controls the sharpness of preference separation. This formulation ensures that the model directly optimizes preference probabilities while maintaining policy stability and avoiding the high variance introduced by reinforcement learning methods.

In Rec-Align, we specifically integrate Direct Preference Optimization (DPO) within the ranking module to align data analysis recommendations with human analytical preferences. By assigning higher scores to operations that effectively capture user intent and generate actionable insights, and lower scores to less useful analyses, DPO effectively reinforces outputs aligned with analyst expectations.

This targeted integration of DPO significantly enhances the quality and practical applicability of generated analyses by ensuring accurate alignment with human analytical preferences.

K Case Study

Figure 4 to Figure 11 illustrate a case study demonstrating our TablePilot framework. This case provides a detailed analysis of a real-world example, showcasing the practical applications and effectiveness of TablePilot in generating comprehensive data analysis recommendations.

TablePilot Input: A Table

Airport Code	Year	Month	Domestic Passengers	International Passengers	Total Passengers
ACY	2015	Jan	98177	90	98267
ACY	2015	Feb	96431	65	96496
ACY	2015	Mar	116493	197	116690
ACY	2015	Apr	105539	161	105700
			......		
ACY	2014	Jan	92421	0	92421
ACY	2014	Feb	90255	148	90403
			...		
ACY	2013	Jan	87375	190	87565
ACY	2013	Feb	81750	380	82130
			...		
EWR	2015	Jan	1752923	859391	2612314
EWR	2015	Feb	1672285	756925	2429210
			...		
EWR	2014	Jan	1703949	842188	2546137
EWR	2014	Feb	1516373	705458	2221831
			...		
JFK	2015	Jan	1891156	2060967	3952123
JFK	2015	Feb	1806434	1718574	3525008
			...		
SWF	2015	Jan	20952	0	20952
SWF	2015	Feb	19816	0	19816
			...		
LGA	2015	Jan	1721701	110101	1831802
LGA	2015	Feb	1719642	111968	1831610
			...		
LGA	1977	Nov	1323301	0	1323301
LGA	1977	Dec	1293393	0	1293393

Step1: Analysis Preparation

1.Table Sampling

Table:
- 'Airport Code' (Column A): ['EWR', 'JFK', 'LGA', 'LGA', 'SWF', ...]
- 'Year' (Column B, Numeric): [2011, 1995, 1981, 1999, 1978, ...]
- 'Month' (Column C): ['Dec', 'Sep', 'Nov', 'Dec', 'Mar', ...]
- 'Domestic Passengers' (Column D, Numeric): [1901523, 1904407, 1085316, 1096845, 1594280, ...]
- 'International Passengers' (Column E, Numeric): [2252910, 1102018, 104142, 136952, 14095, ...]
- 'Total Passengers' (Column F, Numeric): [775695, 2499703, 1858392, 2318716, 1756044, ...]

2.Table Explanation

{ "Table Theme": "Annual and Monthly Passenger Traffic Statistics for Major U.S. Airports",
"Columns": {
"Airport Code": "This column contains the unique three-letter codes representing different airports. It is essential for identifying the specific airport associated with the passenger data.",
"Year": "This column indicates the year in which the passenger data was recorded. It helps in analyzing trends and changes in passenger traffic over different years.",
"Month": "This column specifies the month of the year when the passenger data was collected. It allows for the examination of seasonal patterns and monthly variations in passenger numbers.",
"Domestic Passengers": "This column records the number of passengers traveling domestically through the airport. It is crucial for understanding the volume of domestic air travel and its contribution to total passenger traffic.",

Figure 4: Overview of TablePilot framework case study. Analysis Preparation-1.

"International Passengers": "This column shows the number of passengers traveling internationally through the airport. It provides insights into the international travel segment and its impact on overall airport traffic.",
"Total Passengers": "This column represents the sum of domestic and international passengers for each airport, year, and month. It is a key metric for assessing the total passenger throughput and overall airport activity." },
"Column Relationships": {
"Airport Code": "The 'Airport Code' column is linked with all other columns to provide context for the passenger data, allowing for airport-specific analysis of trends and patterns.",
"Year": "The 'Year' column works with 'Month' to create a time frame for the data, enabling the analysis of passenger trends over specific periods. It also interacts with passenger columns to observe yearly changes.",
"Month": "The 'Month' column complements 'Year' to define the exact time of data collection, facilitating the study of monthly fluctuations in passenger numbers.",
"Domestic Passengers": "The 'Domestic Passengers' column, along with 'International Passengers', contributes to the 'Total Passengers' column, providing a breakdown of passenger types.",
"International Passengers": "The 'International Passengers' column, in conjunction with 'Domestic Passengers', forms the 'Total Passengers' column, offering insights into the composition of passenger traffic.",
"Total Passengers": "The 'Total Passengers' column is derived from the sum of 'Domestic Passengers' and 'International Passengers', serving as a comprehensive measure of airport traffic for analysis." } }

Step2: Module-based Analysis

1. Basic Analysis Module: <query, code> pairs

[{ "query": "Filter the data to show only the records for the airport code 'JFK'.",
"code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\njfk_data = df[df['Airport Code'] == 'JFK']\ncsv_file_path = 'output.csv'\njfk_data.to_csv(csv_file_path, index=False)" },
{ "query": "Sort the data by 'Total Passengers' in descending order.",
"code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\nsorted_data = df.sort_values(by='Total Passengers', ascending=False)\ncsv_file_path = 'output.csv'\nsorted_data.to_csv(csv_file_path, index=False)" },
{ "query": "Group the data by 'Year' and calculate the total number of 'Domestic Passengers' for each year.",
"code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\ngrouped_data = df.groupby('Year')['Domestic Passengers'].sum().reset_index()\ncsv_file_path = 'output.csv'\ngrouped_data.to_csv(csv_file_path, index=False)" },
{ "query": "Create a pivot table to show the total number of 'International Passengers' for each 'Airport Code' and 'Year'.",
"code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\npivot_table = df.pivot_table(values='International Passengers', index='Airport Code', columns='Year', aggfunc='sum').reset_index()\ncsv_file_path = 'output.csv'\npivot_table.to_csv(csv_file_path, index=False)" },
{ "query": "Filter the data to show only the records where 'Total Passengers' is greater than 2,000,000.",
"code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\nfiltered_data = df[df['Total Passengers'] > 2000000]\ncsv_file_path = 'output.csv'\nfiltered_data.to_csv(csv_file_path, index=False)" }]

2. Table Visualization Module: <query, code> pairs

[{ "query": "Generate a line chart to show the trend of total passengers over the years for each airport.",
"code": "import matplotlib.pyplot as plt\nimport pandas as pd\nfile = 'table_name.txt'\ndf = pd.read_csv(file, sep='\\t')\nfor airport in df['Airport Code'].unique():\n airport_data = df[df['Airport Code'] == airport]\n plt.plot(airport_data['Year'], airport_data['Total Passengers'], label=airport)\nplt.xlabel('Year')\nplt.ylabel('Total Passengers')\nplt.title('Trend of Total Passengers Over the Years for Each Airport')\nplt.legend()\nplt.xticks(rotation=45)\nplt.tight_layout()\n#Chart INFO: {'x_fields': 'Year', 'y_fields': ['Total Passengers'], 'chart_type': 'lineChart'}\nplt.show()" },
{ "query": "Create a bar chart to compare the number of domestic and international passengers for each airport.",
"code": "import matplotlib.pyplot as plt\nimport pandas as pd\nfile = 'table_name.txt'\ndf = pd.read_csv(file, sep='\\t')\nlabels = df['Airport Code'].unique()\ndomestic = df.groupby('Airport Code')['Domestic Passengers'].sum()\ninternational = df.groupby('Airport Code')['International Passengers'].sum()\nx = range(len(labels))\nplt.bar(x, domestic, width=0.4, label='Domestic Passengers', align='center')\nplt.bar(x, international, width=0.4, label='International Passengers', align='edge')\nplt.xlabel('Airport Code')\nplt.ylabel('Number of Passengers')\nplt.title('Comparison of Domestic and International Passengers for Each Airport')\nplt.xticks(x, labels)\nplt.legend()\nplt.tight_layout()\n#Chart INFO: {'x_fields': 'Airport Code', 'y_fields': ['Domestic Passengers', 'International Passengers'], 'chart_type': 'barChart'}\nplt.show()" },

Figure 5: Overview of TablePilot framework case study. Module-based Analysis-1.

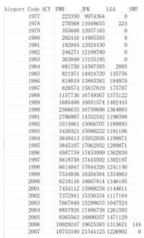
{ "query": "Generate a scatter chart to analyze the relationship between domestic and international passengers for each airport.",
"code": "import matplotlib.pyplot as plt\nimport pandas as pd\nfile = 'table_name.txt'\ndf = pd.read_csv(file, sep='\\t')\nplt.scatter(df['Domestic Passengers'], df['International Passengers'], c='blue', alpha=0.5)\nplt.xlabel('Domestic Passengers')\nplt.ylabel('International Passengers')\nplt.title('Relationship Between Domestic and International Passengers for Each Airport')\nplt.tight_layout()\n#Chart INFO: {'x_fields': 'Domestic Passengers', 'y_fields': ['International Passengers'], 'chart_type': 'scatterChart'}\nplt.show()" },
{ "query": "Create a pie chart to show the proportion of total passengers for each airport.",
"code": "import matplotlib.pyplot as plt\nimport pandas as pd\nfile = 'table_name.txt'\ndf = pd.read_csv(file, sep='\\t')\nlabels = df['Airport Code'].unique()\ntotal_passengers = df.groupby('Airport Code')['Total Passengers'].sum()\nplt.pie(total_passengers, labels=labels, autopct='%1.1f%%', startangle=140)\nplt.title('Proportion of Total Passengers for Each Airport')\nplt.tight_layout()\n#Chart INFO: {'x_fields': 'Airport Code', 'y_fields': ['Total Passengers'], 'chart_type': 'pieChart'}\nplt.show()" },
{ "query": "Generate a combo chart to show both the total passengers and the number of international passengers over the years for each airport.",
"code": "import matplotlib.pyplot as plt\nimport pandas as pd\n\nfile = 'table_name.txt'\ndf = pd.read_csv(file, sep='\\t')\n# Summing Domestic and International Passengers across all airports by year\ngrouped = df.groupby('Year')[['Domestic Passengers', 'International Passengers']].sum()\n\n# Plotting a stacked bar chart\nax = grouped.plot(kind='bar', stacked=True, figsize=(10, 6))\nax.set_xlabel('Year')\nax.set_ylabel('Number of Passengers')\nplt.title('Relative Trend of Domestic and International Passengers for All Airports')\nplt.xticks(rotation=45)\nplt.tight_layout()\n\n# Chart INFO: {'x_fields': 'Year', 'y_fields': ['Domestic Passengers', 'International Passengers'], 'chart_type': 'stackedBarChart'}\nplt.show()" }]

3. Stastics Modeling Module : <query, code> pairs

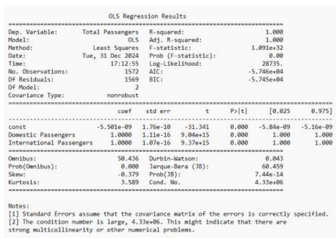
[{ "query": "Perform a trend prediction analysis to forecast the total number of passengers for the next 12 months at JFK airport using historical data.",
"code": "import pandas as pd\nfrom statsmodels.tsa.statespace.sarimax import SARIMAX\n\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\n\ndf['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' + df['Month'] + '-01')\njfk_data = df[df['Airport Code'] == 'JFK'].sort_values('Date')\njfk_data.set_index('Date', inplace=True)\n\nmodel = SARIMAX(jfk_data['Total Passengers'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))\nmodel_fit = model.fit(disp=False)\n\nforecast = model_fit.forecast(steps=12)\nforecast_df = pd.DataFrame({'Date': pd.date_range(start=jfk_data.index[-1] + pd.DateOffset(months=1), periods=12, freq='M'), 'Forecasted Total Passengers': forecast})\n\nprint(forecast_df)" },
{ "query": "Conduct a correlation test to determine the relationship between domestic and international passengers across all airports.",
"code": "import pandas as pd\nfrom scipy.stats import pearsonr\n\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\n\ncorr, p_value = pearsonr(df['Domestic Passengers'], df['International Passengers'])\n\nprint(\"Correlation Method: Pearson\")\nprint(f\"Correlation Coefficient: {corr}\")\nprint(f\"P-value: {p_value}\")" },
{ "query": "Build a regression model to predict the total number of passengers based on the number of domestic and international passengers.",
"code": "import pandas as pd\nimport statsmodels.api as sm\n\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\n\nX = df[['Domestic Passengers', 'International Passengers']]\nX = sm.add_constant(X)\nY = df['Total Passengers']\n\nmodel = sm.OLS(Y, X).fit()\n\nprint(model.summary())" }]

Step3: Analysis Optimization

1. First Round Execution Results (Part of)



Airport Code	ACY	EWR	JFK	LGA	SWF
1977		223350	9974364	0	
1978		270568	11048655	224	
1979		355608	12057183	0	
1980		202416	11985593	0	
1981		182044	12024430	0	
1982		246271	12199780	0	
1983		363048	13155195	0	
1984		691750	14587395	2095	
1985		821971	14824720	135756	
1986		819019	13893383	194970	
1987		820574	15837819	175707	
1988		1157736	16749367	1375122	
1989		1685408	16691574	1482443	
1990		2566635	16750606	1364803	
1991		2786007	14552542	1190398	
1992		3315861	15066707	1189895	
1993		3439321	15080232	1191106	
1994		3649413	15952036	1199971	
1995		3845107	17062932	1289871	
1996		4587759	17453099	1362030	
1997		5618758	17445592	1302197	
1998		6614047	17844326	1241140	
1999		7534036	18204584	1334863	
2000		8210116	18667814	1346195	
2001		7454112	15988258	1144611	
2002		7372941	15336334	1117104	
2003		7667840	15299633	1047524	
2004		8857926	17486758	1261593	
2005		9365563	18800357	1471129	
2006		10029247	19625383	1313621	144
2007		10753100	21544125	1226902	0
		...			



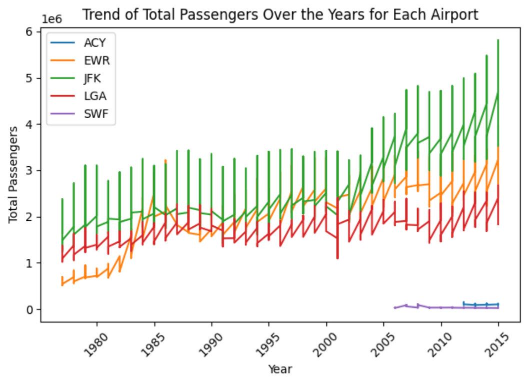
```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Total Passengers   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.091e+32
Date:                Tue, 31 Dec 2024   Prob (F-statistic):               0.00
Time:                        17:12:55   Log-Likelihood:                 28735.
No. Observations:                1572   AIC:                        -5.746e+04
Df Residuals:                    1569   BIC:                        -5.745e+04
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                     -5.501e-09   1.76e-10    -31.341      0.000   -5.84e-09   -5.16e-09
Domestic Passengers           1.0000   1.11e-16   9.04e+15      0.000       1.000       1.000
International Passengers      1.0000   1.07e-16   9.37e+15      0.000       1.000       1.000
==============================================================================
Omnibus:                       50.436   Durbin-Watson:                   0.043
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               60.459
Skew:                          -0.379   Prob(JB):                     7.44e-14
Kurtosis:                       3.589   Cond. No.                     4.33e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.33e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

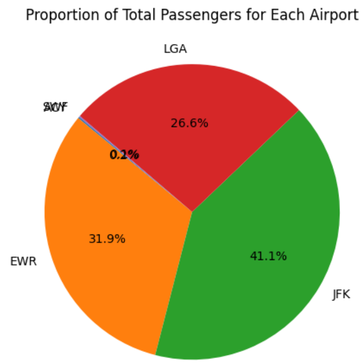
Figure 6: Overview of TablePilot framework case study. Module-based Analysis-2.

Unclear image

1.vertical axis data overlap

2.data labels overlap





2. Revision: Optimized Successful Results and Revision Error cases

(Show only the queries and codes that have undergone significant optimization.)

[{ "improvement_reason": "The original code is already well-structured and performs the task as intended. However, to enhance the functionality, I will modify the code to sort the results in descending order based on the average number of domestic passengers. This addition makes the output more informative and easier to analyze, as the busiest airports (with the highest number of passengers) will appear first.",
"original_query": **"Group the data by 'Airport Code' and calculate the average number of domestic passengers for each airport.",**
"optimized_query": **"Group by 'Airport Code', calculate the average domestic passengers, and sort in descending order.",**
"optimized_code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\n\n# Group the data by 'Airport Code' and calculate the average number of domestic passengers for each airport\n# The result is a DataFrame with 'Airport Code' as one column and the average 'Domestic Passengers' as another\ngrouped_df = df.groupby('Airport Code')['Domestic Passengers'].mean().reset_index()\n\n# Sort the result in descending order based on the average number of domestic passengers\nsorted_df = grouped_df.sort_values(by='Domestic Passengers', ascending=False)\n\n# Output the sorted result to a CSV file without the index\ncsv_file_path = 'output.csv'\nsorted_df.to_csv(csv_file_path, index=False)" }]

After the model has reviewed all the execution results, it combines the group by and sort operations, leading to further analysis with greater practicality.

[{ "improvement_reason": "The original code performs a filtering operation, but this can be improved by replacing the filter with a more insightful operation , because the filter operation is duplicated. It's important to notice that the total number of passengers has been increasing over the years. To enhance the analysis, I will add a new column that represents the year-over-year growth rate of total passengers, which provides a better insight into the changes in passenger volume rather than just filtering by the year.",
"original_query": **"Filter the data to find records from the year 2000 onwards and sort them by total passengers in ascending order.",**
"optimized_query": **"Calculate the year-over-year growth rate of total passengers and add it as a new column, allowing insights into the changes in passenger volume over time.",**
"optimized_code": "import pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\n\n# Calculate the year-over-year growth rate of total passengers\n# First, sort the data by 'Year' to ensure the correct order\ndf = df.sort_values(by='Year')\n\n ... }]

The model identified the redundant data analysis operation and replaced it with a more insightful one.

Figure 7: Overview of TablePilot framework case study. Analysis Optimization-1.

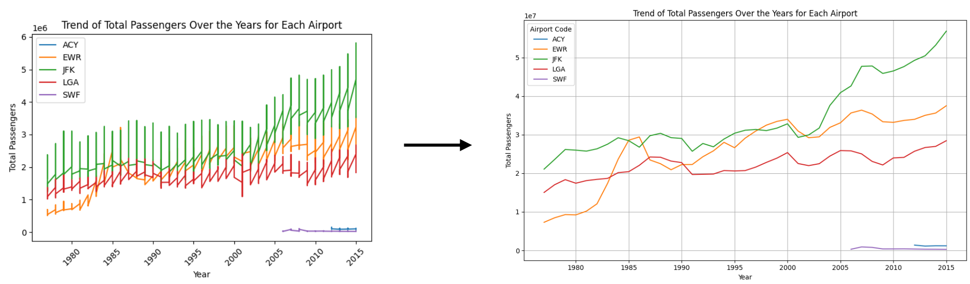
[{ "improvement_reason": "The original code suffers from overlapping y-axis labels, which can make the chart difficult to read. In the updated version, I pivot the data so that years are the index and airport codes are the columns. This approach avoids label overlap by ensuring that each airport's passenger trend is plotted separately. Additionally, the chart is improved by automatically handling missing values with `fillna(0)` and using `tight_layout()` to optimize the figure layout.",

"original_query": **"Generate a line chart to show the trend of total passengers over the years for each airport."**,

"optimized_query": **"Create a line chart to analyze the trend of total passengers over the years for each airport."**,

"optimized_code": "import matplotlib.pyplot as plt\nimport pandas as pd\nfile = 'table_name.csv'\ndf = pd.read_csv(file)\n\n# Pivot the data to have years as index and airport codes as columns\npivot_df = df.pivot_table(index='Year', columns='Airport Code', values='Total Passengers', aggfunc='sum').fillna(0)\n\n# Plotting\npivot_df.plot(kind='line', figsize=(10, 6))\nplt.title('Trend of Total Passengers Over the Years for Each Airport')\nplt.xlabel('Year')\nplt.ylabel('Total Passengers (in millions)')\nplt.legend(title='Airport Code')\nplt.grid(True)\nplt.tight_layout()\n#Chart INFO: {'x_fields': 'Year', 'y_fields': ['Total Passengers'], 'chart_type': 'lineChart'}\nplt.show()" }]

The Multimodal Revision module provides the model with the ability to recognize unclear images and can combine this with its coding capabilities to make corrections.



Multimodal Revision module makes the chart clearer

Step4: Analysis Alignment

(Show only the high score query which is recalled by our Ranking module)

[{ "query": "Group by 'Airport Code', calculate the average domestic passengers, and sort in descending order.",
"criteria":
{ "Meaningful": { "score": 5, "reason": "Identifies airports with the highest average domestic traffic, providing clear comparative insights." },
"Relative": { "score": 5, "reason": "Closely tied to the table theme, focusing on domestic passenger trends across airports." },
"Reasonableness": { "score": 5, "reason": "Grouping and averaging are standard practices for summarizing and comparing data." },
"Diversity": { "score": 4, "reason": "Introduces a grouping operation, enhancing the variety in rudimentary tasks." },
"Interpretable": { "score": 5, "reason": "Mostly straightforward, though understanding averages might require some users to apply basic statistical knowledge." },
"Insightful": { "score": 4, "reason": "Offers useful insights, but mostly confirms expected trends in domestic traffic." } } },
{ "query": "Calculate the year-over-year growth rate of total passengers and add it as a new column, allowing insights into the changes in passenger volume over time.",
"criteria":
{ "Meaningful": { "score": 5, "reason": "Offers critical insights into growth trends, essential for forecasting and strategic decision-making." },
"Relative": { "score": 5, "reason": "Highly relevant to the table's theme, focusing on growth trends in

Figure 8: Overview of TablePilot framework case study. Analysis Optimization-2.

passenger traffic." },
"Reasonableness": { "score": 5, "reason": "Calculating growth rates is a logical and insightful way to analyze time-series data." },
"Diversity": { "score": 5, "reason": "Introduces a calculation for growth rate, a more advanced rudimentary task." },
"Interpretable": { "score": 3, "reason": "The concept of growth rates might be less intuitive for some users, requiring explanation." },
"Insightful": { "score": 5, "reason": "Reveals valuable trends in passenger growth, aiding in predictive analysis." } } },
{ "query": "Create a line chart to analyze the trend of total passengers over the years for each airport.",
"criteria": {
"Meaningful": { "score": 5, "reason": "Effectively visualizes long-term trends, aiding strategic planning and capacity forecasting." },
"Relative": { "score": 5, "reason": "Directly related to the table theme, focusing on longitudinal passenger trends." },
"Reasonableness": { "score": 5, "reason": "Line charts are ideal for time-series data, providing clear trend visualization." },
"Diversity": { "score": 4, "reason": "Adds variety by introducing a line chart, though line charts are common." },
"Interpretable": { "score": 5, "reason": "Clear and easy to interpret, with well-labeled axes and legend." },
"Insightful": { "score": 5, "reason": "Uncovers trends over time, providing deep insights into passenger traffic dynamics." } } },
{ "query": "Generate a stacked bar chart to show the relative trend of Domestic Passengers and International Passengers for all airports.",
"criteria": {
"Meaningful": { "score": 5, "reason": "This query effectively visualizes the relative trend of domestic and international passengers over time, which is highly valuable for understanding the traffic distribution." },
"Relative": { "score": 5, "reason": "The query is closely tied to the dataset's theme, focusing on passenger trends across airports and time." },
"Reasonableness": { "score": 5, "reason": "The stacked bar chart is a reasonable method for visualizing the relative comparison of domestic and international passengers over time." },
"Diversity": { "score": 5, "reason": "This query introduces a more complex visualization (stacked bar chart), adding significant diversity to the analysis." },
"Interpretable": { "score": 5, "reason": "The chart is clear, with labeled axes and a legend, making it easy to interpret for users." },
"Insightful": { "score": 5, "reason": "The chart provides insightful information about the relative changes in passenger traffic, which is valuable for strategic planning." } } }
{ "query": "Perform a trend prediction analysis to forecast the total number of passengers for the next 12 months at JFK airport using historical data.",
"criteria": {
"Meaningful": { "score": 5, "reason": "Highly valuable for forecasting future passenger volumes, aiding in strategic planning." },
"Relative": { "score": 5, "reason": "Directly tied to the dataset's theme, focusing on future trends in passenger traffic." },
"Reasonableness": { "score": 5, "reason": "Trend prediction is a logical extension of time-series analysis in this context." },
"Diversity": { "score": 5, "reason": "Adds significant diversity by introducing predictive modeling and forecasting." },
"Interpretable": { "score": 4, "reason": "Results are clear, though understanding forecasting might require some statistical knowledge." },
"Insightful": { "score": 5, "reason": "Provides forward-looking insights, crucial for planning and decision-making." } } }]

Unrecalled Query Example:
The selection of the regression variables are meaningless.

[{ "query": "Build a regression model to predict the total number of passengers based on the number of domestic and international passengers.",
"criteria": {
"Meaningful": { "score": 2, "reason": "While the regression model shows a perfect fit (R-squared = 1.00), the analysis is meaningless because the total number of passengers is simply the sum of domestic and international passengers, making the model redundant."}
... } }]

Figure 9: Overview of TablePilot framework case study. Analysis Optimization-3.

TablePilot Output: Recommend Analysis Results

(The code is not displayed.)

Query1:

Group by 'Airport Code', calculate the average domestic passengers, and sort in descending order.

Result:

Airport Code	Domestic Passengers
LGA	1767834.84
EWR	1759483.09
JFK	1388555.60
ACY	102394.36
SWF	37388.93

Query2:

Calculate the year-over-year growth rate of total passengers and add it as a new column.

Result:

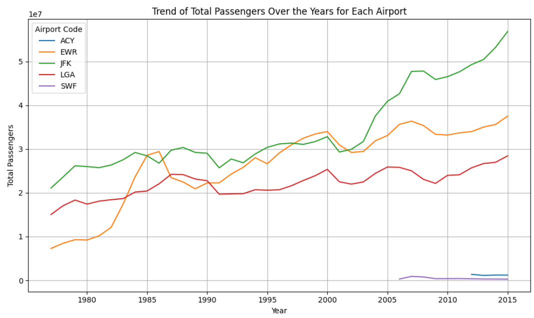
Airport Code	Year	Total Passengers	YoY Growth Rate (%)
ACY	2012	1385638	
ACY	2013	1132898	-18.24
ACY	2014	1211667	6.95
...			
EWR	1977	7301651	
EWR	1978	8468482	15.98
EWR	1979	9296742	9.78
EWR	1980	9223130	-0.79
EWR	1981	10181468	10.39
EWR	1982	12087789	18.72
EWR	1983	17402874	43.97
EWR	1984	23647301	35.88
...			
JFK	1977	21080688	
JFK	1978	23589693	11.9
JFK	1979	26171733	10.95
...			
LGA	1977	15033019	
LGA	1978	17041335	13.36
LGA	1979	18347855	7.67
...			
SWF	2006	309921	
SWF	2007	913927	194.89
SWF	2008	789307	-13.64
...			

Figure 10: Overview of TablePilot framework case study. TablePilot Output Results-1.

Query3:

Create a line chart to analyze the trend of total passengers over the years for each airport.

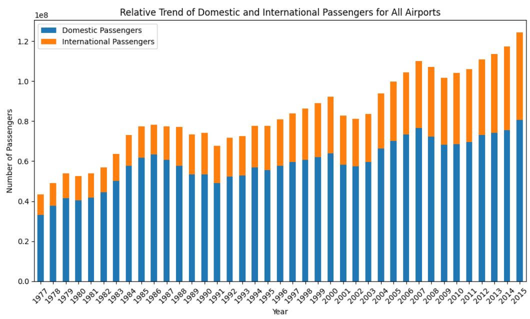
Result:



Query4:

Generate a stacked bar chart to show the relative trend of Domestic Passengers and International Passengers for all airports.

Result:



Query5:

Perform a trend prediction analysis to forecast the total number of passengers for the next 12 months at JFK airport using historical data.

Result:

Date	Forecasted Total Passengers
2016-01-01	4268876
2016-02-01	3892484
2016-03-01	4764605
2016-04-01	4777741
2016-05-01	4882515
2016-06-01	4989724
2016-07-01	5372664
2016-08-01	5442647
2016-09-01	4550673
2016-10-01	4605193
2016-11-01	4284598
2016-12-01	4709262

Figure 11: Overview of TablePilot framework case study. TablePilot Output Results-2.

L TablePilot Report Generation

Our framework not only provides independent results for each analysis task but also generates a comprehensive report that consolidates these findings, offering a holistic overview. Figure 12, Figure 13, and Figure 14 illustrate an example of the extended functionality of TablePilot in generating analysis reports. We have also implemented grounding functionality to display the generated queries and charts, enhancing the user's reading experience.

Passenger Traffic Insights of Major U.S. Airports

1. Overview of Domestic-Passenger Volumes

From **Query1** (**Group by 'Airport Code', calculate the average domestic passengers, and sort in descending order),** we see the following rank order in **descending** order (see the corresponding table in *Query1 Result*):

•**LGA** leads with **1.77 million** Domestic Passengers on average.
•**EWR** follows closely at **1.76 million**.
•**JFK** reports **1.39 million**.
•**ACY (Atlantic City)** averages about **0.10 million** (102 thousand).
•**SWF (Stewart)** has the smallest average, at around **0.04 million** (37 thousand).

Although LaGuardia (LGA) tops the list in Domestic volume, the data in *Query1* also indicate that JFK's higher International passenger counts often make it the busiest airport overall in many individual months.

2. Year-over-Year Growth Observations

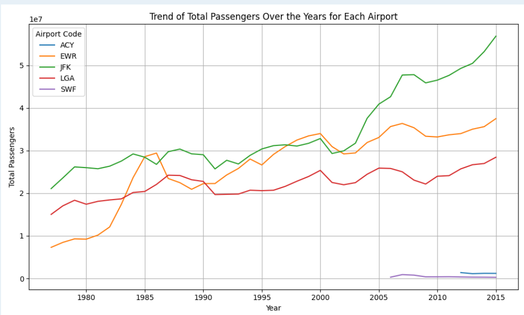
Looking at **Query2** (**Calculate the year-over-year growth rate of total passengers and add it as a new column**), we see:

•**JFK**, **EWR**, and **LGA** can show significant YoY swings—some well over 50%—especially if the prior year's traffic was unusually low (e.g., weather disruptions).
•Small airports such as **SWF** show extremely large percentage changes when the base was tiny (some results were +8000% in certain months).
•**ACY** also exhibits wide fluctuations for similar reasons (relatively small passenger volumes).

The *Query2 Result* table highlights examples like **ACY** moving from 1,385,638 passengers in 2012 to 1,132,898 in 2013, a nearly **-18.24%** drop, followed by a **+6.95%** increase in 2014.

3. Long-Term Passenger Trends

Using **Query3** (**Create a line chart to analyze the trend of total passengers over the years for each airport**), we observe:

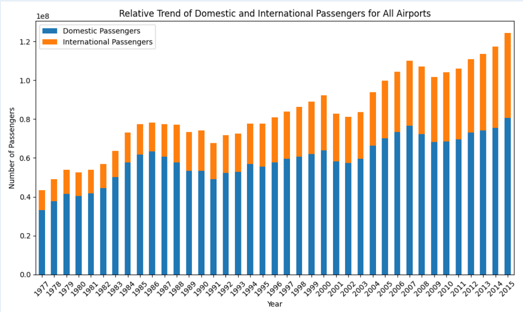


•**JFK** volumes climb steadily after the mid-1980s, eventually surpassing other airports by the early 2000s and exceeding **50 million** annually by around 2015.
•**EWR** shows robust growth in the 1980s, levels off somewhat in the 1990s, then resumes an upward trend in the 2000s, surpassing **30**

Figure 12: Overview of TablePilot framework report-1.

million by 2015.
•**LGA**, predominantly domestic, displays steady but moderate growth, frequently ranging **25–30 million** annual passengers in the 2010s.
•**ACY** and **SWF** remain relatively niche airports with well under **2 million** passengers per year, though **ACY** experiences more pronounced seasonality.

From **Query4** (**Generate a stacked bar chart to show the relative trend of Domestic Passengers and International Passengers for all airports**).



The number of passengers at all airports has shown a steady upward trend year by year, with an increasing proportion of international passengers, reflecting a growing level of airport internationalization.

4. JFK Passenger Forecast for 2016

Query5(**Perform a trend prediction analysis to forecast the total number of passengers for the next 12 months at JFK airport using historical data**) performs a trend prediction analysis (time-series modeling) to forecast **JFK**'s total monthly passengers for the next 12 months (Jan–Dec 2016). These projections align with historical seasonal patterns at **JFK**: lower winter volumes, peaks in mid- to late-summer, followed by a dip in early autumn, and a modest rebound during the holiday season.

5. Key Takeaways and Recommendations

•JFK Dominates Overall Passenger Counts
•Thanks to significant domestic and international traffic, JFK remains the busiest. *Query5* forecasts continued monthly volumes exceeding 4 million, peaking above 5 million in summer 2016. Expansion in gate, ground transport, and customs capacity may be warranted.
•LGA Tops Domestic Traffic but Has Limited International Reach
•*Query1* shows LGA having the highest average monthly domestic traffic (1.77 million). The airport can capitalize on frequent business routes. Evaluating potential to expand international service (where feasible) could be a strategic consideration.
•EWR Demonstrates Steady, High Total Volumes with Notable International Shares
•The growth figures from *Query2* show Newark's consistent rise over decades. EWR remains among the top three in total passengers, underpinned by robust domestic

Figure 13: Overview of TablePilot framework report-2.

•and transatlantic flight offerings.
•Smaller Airports (ACY, SWF) Show Volatility
•The year-over-year variations in *Query2* confirm that lower baselines magnify percentage changes at ACY and SWF. Targeted seasonal or niche routes may help manage this volatility.
•Forecast Confidence at JFK Remains Robust
•With *Query5* predicting monthly totals above 5.3 million at peak, JFK's role as an international gateway will only grow. Strategic planning for future demand surges—especially in the summer season—is essential.

6. Conclusion

Overall, the queries confirm that **JFK**, **EWR**, and **LGA** together handle the bulk of New York-area passenger traffic. Their respective trends (*Query3*) reveal:
•JFK's steadily increasing dominance,
•EWR's balanced, continued growth,
•LGA's leading domestic share.
Meanwhile, *Query2* shows the large swings that can occur at smaller airports (SWF, ACY). Lastly, the *Query5* forecast underscores JFK's projected climb toward 5.44 million monthly passengers in August 2016, reinforcing its status as the region's busiest hub.
In summary, capacity and strategic planning at **JFK**, **EWR**, and **LGA** will remain priorities, especially as New York-area passenger counts continue to climb year over year.

Figure 14: Overview of TablePilot framework report-3.

M Prompt Design

Prompt 15 to Prompt 39 illustrate the detailed prompt designs used in TablePilot.

Prompt for TablePilot – Table Explanation

```
## Table Understanding Expert
You are an experienced **Table Understanding Expert** specializing in interpreting and analyzing complex table data from a global perspective. Your task is to receive a table and, based on your expertise, provide a detailed analysis of the table’s theme, the meaning of each column, and the relationships between columns, in order to generate accurate explanations that can be used for downstream data analysis.

## Your Primary Responsibilities:
#### 1. Accurately understand the table's theme:
By analyzing the content and structure of the table, you need to identify its main purpose and core theme.  Ensure the theme is concise and briefly summarizes the main purpose of the entire table.

#### 2. Understand the role of each column in the table:
Analyze each column one by one, understanding its data type, business context, and specific function in the table.
- A description of the content of the column.
- How this column’s data contributes to understanding the overall table or supports a particular business scenario.
- If the column name is too vague or unclear, provide a reasonable inference or additional explanation to make it easier for data analysts to understand its purpose.

#### 3. Understand the relationships between columns:
Based on the structure of the table, infer any potential relationships between columns. Particularly focus on the interactions between columns during data analysis, business logic, or statistical analysis.
- One column’s value may depend on another column’s value in order to have practical significance.
- Several columns may need to be used together in certain analysis scenarios for meaningful insights.

## Table
{table}

### Output Format:
You need to generate a **JSON file** containing the following three main fields:
1. `"Table Theme"`: The overall theme of the table as you have understood it.
2. `"Column Name"`: The specific function and meaning you’ve interpreted for each column.
3. `"Column Relationships"`: The relationships between each column and others.
```

Figure 15: Prompt design in TablePilot

Prompt for TablePilot – Basic Analysis

Basic Analysis Assistant
You are an advanced data analysis assistant tasked with predicting meaningful user queries based on a given table and its explanations. Your objective is to recommend some **diverse and practical queries**, each accompanied by the corresponding **Python code** using the `pandas` library. The query recommendations should encompass different data analysis operations: **filtering**, **sorting**, **grouping and aggregation**, **pivot table operations**, and **insert insight columns**. Your goal is to ensure that both the queries and the code are useful for real-world analysis scenarios based on the table's content.

Definitions of Rudimentary Data Analysis Task:
This task involves essential data manipulation operations such as filtering rows based on specified conditions, sorting data in ascending or descending order, grouping data by one or more columns to apply aggregate functions like sum or average, and creating pivot tables to summarize data. These operations are fundamental for organizing raw data, simplifying complex datasets, and generating quick overviews. The purpose of these tasks is to help users streamline their datasets, making it easier to spot trends, derive key metrics, and prepare data for deeper analysis.

Explanations of the rudimentary data analysis operations

1. Filter (Filtering Data)
…
2. Sorting (Sorting Data)
…
3. Group by and Aggregation (Grouping & Aggregating Data)
…
4. Pivot Table (Creating a Pivot Table)
…
5. Insert Insight Columns (Calculating and Adding Insightful Data)
…

Query Generation Requirements:
1. **Diversity**: Ensure that the queries span different types of analysis operations (filter, sort, group by with aggregation, and pivot table).
2. **Variety**: Each query should involve different columns and operations, utilizing as much of the table's information as possible.
3. **Practicality**: The queries should align with real-world analysis needs, making them contextually relevant to the provided table and its explanations.

Figure 16: Prompt design in TablePilot

Prompt for TablePilot – Basic Analysis (Cont.)

Code Generation Requirements:
*1. **Accuracy**: The code must execute successfully without errors or warnings, taking into account the specific formatting of table data (e.g., date formats).*
*2. **Logical Consistency**: The code should precisely reflect the intent of the query and perform the required operation accurately.*

Integration:
*When generating both queries and their corresponding code, ensure that they are **mutually aligned**. The query should guide the generated code, and the code should fully satisfy the query's requirements. This joint generation will improve coherence and ensure that each query has a perfectly matched, executable solution in `pandas`.*

Please propose some queries along with the corresponding executable code for the following table:
Table
{table}
This table format retains all column names from the full table, with [] showing randomly sampled rows to represent part of the data. This sampling is only to help you understand the table's data structure and types. Please generate queries and code based on the complete table.

The table's explanation is provided below to guide your query and code :
Explanation
{table explanation}

To ensure the generated queries meet task requirements and are relevant, you may choose the number of queries to generate (up to a maximum of five).

DO NOT output anything other than the JSON file containing only the `query` and `code`.

Figure 17: Prompt design in TablePilot

Prompt for TablePilot – Table Visualization

Table Visualization Assistant
You are an advanced data analysis assistant specializing in chart generation based on a given table and its explanations. Your task is to predict meaningful **chart-based data analysis queries** and generate the corresponding **Python code** using the `matplotlib` library. Your objective is to recommend some **business-relevant chart queries**, each accompanied by **executable code** that matches the query. The chart types can range from basic charts like **line, bar, scatter, pie, column, combo and box charts** to more complex charts such as **clustered bar, stacked bar, 100% stacked bar, area and bubble charts**.Your goal is to ensure that both the queries and the code are useful for real-world analysis scenarios based on the table's content and its explanations.

Definitions of Chart-Based Data Analysis Task:
This task focuses on the visualization of data through various chart types, such as line, bar, scatter, pie, column, combo and box charts. Additionally, more advanced chart types like clustered bar charts, stacked bar charts, 100% stacked bar charts, area charts, and bubble charts allow for more complex comparisons and multidimensional analysis. The goal of these tasks is to enable users to visually explore patterns, relationships, and trends within their data. By making data easier to interpret, users can gain deeper insights, facilitate decision-making, and communicate findings more effectively through clear, compelling visuals.

Explanations of the Chart-Based Data Analysis Operations

1. Line Chart (Trend Analysis)
…
2. Bar Chart (Category Comparison)
…
3. Scatter Chart (Correlation and Distribution Analysis)
…
4. Pie Chart (Proportional Distribution)
…
5. Column Chart (Vertical Bar Chart)
…
6. Combo Chart (Multiple Data Series Visualization)
…
7. Box Chart (Statistical Distribution)
…

Figure 18: Prompt design in TablePilot

Prompt for TablePilot – Table Visualization(Cont.)

Advanced Chart Types (For Specific, Complex Use Cases)
***Clustered Bar Chart**, **Stacked Bar Chart**, **100% Stacked Bar Chart**, **Area Chart**, and **Bubble Chart** are advanced chart types used for more specialized data comparisons, such as showing subcategory breakdowns, proportions, and relationships across multiple dimensions. These charts should be applied when they offer additional value over simpler chart types, particularly in complex datasets.*

Chart Selection Consideration
Choose the most suitable chart type based on the structure of the table data. Ensure that each chart selected aligns with the structure and purpose of the data being analyzed, and only use complex charts if they offer distinct analytical value.

Requirements

Query Generation Requirements:
*1. **Diversity**: Ensure that the queries cover different types of charts (line, bar, scatter, combo, stacked bar, etc.).*
*2. **Contextual Relevance**: The queries should reflect meaningful data combinations based on the table's context, ensuring alignment with real-world needs and DO NOT generate irrelevant analyses that lack actionable insights.*
*3. **Advanced Analysis**: Include at least one query that uses a complex chart type (combo chart, stacked bar, bubble chart) if applicable to the table's data.*
*4. **Chart Type Specification**: The generated natural language query must explicitly specify which type of chart is to be drawn.*
*5. **Clear Data Scope**: Clearly define the specific data categories and scope in each query to ensure precise charts that accurately reflect the table's data, avoiding overly generic descriptions.*

Code Generation Requirements:
*1. **Accuracy**: The Python code must be fully executable and correctly reflect the chart type specified in the query.*
*2. **Clarity**: Ensure that the generated code includes appropriate labeling, axis formatting, and legends to enhance the readability of the chart.*
*3. **Aesthetic Quality**: Ensure the generated chart is visually appealing, clear, and easy to interpret. Achieve this by adjusting axis scales, removing redundant labels, and optimizing the overall layout through code.*

Figure 19: Prompt design in TablePilot

Prompt for TablePilot – Table Visualization(Cont.)

Please propose some queries along with the corresponding executable code for the following table:
Table
{table}
This table format retains all column names from the full table, with [] showing randomly sampled rows to represent part of the data. This sampling is only to help you understand the table's data structure and types. Please generate queries and code based on the complete table.

The table's explanation is provided below to guide your query and code :
Explanation
{table explanation}

Output Format:
*The output must be in JSON format, containing five distinct **chart-based queries** with corresponding **Python code** using the `matplotlib` library.*

Finally, generate a comment in the following format in the code:
#Chart INFO: {{'x_fields': '', 'y_fields': [], 'chart_type': ''}}.
The information inside the '' records the details of the chart being plotted. 'x_fields' stores the x-axis of the chart, 'y_fields' stores the y-axis values (which can include multiple fields), and 'chart_type' stores the type of the chart (available options include lineChart, barChart, scatterChart, pieChart, and others).

To ensure the generated queries meet task requirements and are relevant, you may choose the number of queries to generate (up to a maximum of five).

DO NOT output anything other than the JSON file containing only the `query` and `code`.

Figure 20: Prompt design in TablePilot

Prompt for TablePilot – Statistics Modeling

 # Statistics Modeling Analysis Assistant
*You are an advanced data analysis assistant specializing in **statistical modeling and time series forecasting** based on a given table and its explanations. Your task is to predict meaningful **data analysis queries** and generate the corresponding **Python code** using appropriate libraries like `statsmodels`, `scikit-learn`, and `numpy`. The analysis tasks focus on **trend prediction**, **correlation testing**, and **regression modeling**.Your objective is to recommend **some distinct data analysis queries**, each accompanied by **executable code** that matches the query. Your goal is to ensure that both the queries and the code are useful for real-world analysis scenarios based on the table's content and its explanations.*

Definitions of Advanced Data Analysis Task:
This task includes predictive and statistical analyses such as trend forecasting using historical data, correlation testing to quantify relationships between variables, and regression modeling to predict outcomes based on one or more independent variables. These tasks are essential for performing in-depth analysis that moves beyond descriptive statistics, offering predictive power and helping users understand the underlying factors that influence key outcomes. The purpose of these tasks is to support users in making data-driven predictions, identifying correlations, and building models that provide actionable insights for future planning and decision-making.

Explanations of the Data Analysis Operations

1. Trend Prediction (Time Series Analysis)
…
2. Correlation Testing (Dependency Analysis)
…
3. Regression Modeling (Predictive Analysis)
…

Requirements

Query Generation Requirements
*1. **Identify Key Columns**: Recognize which **numerical columns** are suitable for trend prediction, correlation, or regression analysis.*
*2. **Task Suitability**: Select the appropriate modeling technique based on the relationships between columns.*
*4. **Contextual Relevance**: Ensure that the queries are business-relevant and match real-world use cases.*

Figure 21: Prompt design in TablePilot

Prompt for TablePilot – Statistics Modeling(Cont.)

Code Generation Requirements

*1. **Accuracy**: The Python code must be fully executable and correctly implement the specified statistical technique.*
*2. **No Visualization**: The output should only be numerical or numerical sequences (e.g., predicted values, correlation coefficients, or regression results). No plots or visualizations are required.*
*3. **Library Usage**: Use `pandas`, `numpy`, `statsmodels`, `scikit-learn` as necessary for data loading, processing, and modeling.*
*4. **Code Structure**: The code must include proper data loading, transformation, and analysis steps, ensuring it's executable with minimal modification. **No code comments should be generated**.*

Logical Consistency

*1. **Trend Prediction**: When generating prompts for trend prediction tasks, ensure that the dataset includes historical data over a long time period to provide a solid basis for identifying trends accurately.*
*2. **Correlation Testing**: For correlation analysis, focus on examining data categories that may have subtle or non-obvious connections, rather than relationships that are immediately visible. This approach allows for the discovery of deeper insights within the data.*
*3. **Regression Forecasting**: Select data types with potential underlying correlations for regression modeling. Ensure the prompt guides the analysis toward meaningful predictors that can support accurate regression forecasts.*

Integration:
*When generating both queries and their corresponding code, ensure that they are **mutually aligned**. The query should guide the generated code, and the code should fully satisfy the query's requirements. This joint generation will improve coherence and ensure that each query has a perfectly matched, executable solution.*

Output Format:
*The output must be in JSON format, containing three distinct **data analysis queries** with corresponding **Python code** using the appropriate libraries. Each query should be accompanied by executable code that adheres to the following structure:*

Figure 22: Prompt design in TablePilot

Prompt for TablePilot – Statistics Modeling(Cont.)

Please propose some queries along with the corresponding executable code for the following table:
Table
{table}
This table format retains all column names from the full table, with [] showing randomly sampled rows to represent part of the data. This sampling is only to help you understand the table's data structure and types. Please generate queries and code based on the complete table.

The table's explanation is provided below to guide your query and code :
Explanation
{table explanation}

The output format for each specific task is as follows:

1. Trend Prediction (Time Series Analysis)
Result description:
The result returns the forecasted data for the specified time horizon, including the forecasted dates and corresponding values. The output should be in a `DataFrame` format, showing predictions for future time points. Additionaly, return the MAPE calculated between the model's predictions and the ground truth (using the last few rows of the time-series data as ground truth).
```python
# Output Format: Print the forecast DataFrame
print(forecast_df)
print(f"MAPE: {MAPE}")
```

2. Correlation Testing (Dependency Analysis)
Result description:
The result should include the name of the correlation test used (e.g., Pearson or Spearman) and the corresponding correlation coefficient and p-value. The output provides insight into the strength and significance of the relationship between the two variables.
```python
# Output Format: Print correlation method and result
print("Correlation Method: Pearson")  # Or "Spearman" based on the test used
print(f"Correlation Coefficient: {corr}")
print(f"P-value: {p_value}")
```

Figure 23: Prompt design in TablePilot

Prompt for TablePilot – Statistics Modeling(Cont.)

3. Regression Modeling (Predictive Analysis)
Result description:
The result should return the full regression model summary, detailing coefficients, p-values, R-squared, and other relevant statistics that describe the fit of the model.

```python
# Output Format: Print the regression summary
print(model.summary())
```

DO NOT output anything other than the JSON file containing only the `query` and `code`. The code should return the result using the `print()` function at the end.

Figure 24: Prompt design in TablePilot

Prompt for TablePilot – Multimodal Revision

Verifier Prompt for Code Execution Results
You are a seasoned data analyst and professional code verification expert, with extensive experience in data analysis, a deep understanding of various business contexts, and strong coding proficiency. Your role involves not only verifying outputs but also identifying potential issues in data analysis queries and uncovering limitations in the implemented code.

Overview:
You will evaluate each code snippet, whether successfully executed or encountering errors, with three main principles in mind:

*1. **General Standards**:*
*- **Relevance to Table Content**: Assess whether the data analysis code is closely related to the table content.*
*- **Clarity and Business Context Alignment**: Confirm that the code is well-connected to relevant business scenarios, providing valuable insights for actionable data analysis.*
*2. **Task-Oriented Standards**: Evaluation is split across three categories, tailored to specific types of analysis tasks. For each successfully executed code snippet, ensure that the output aligns with the specific task guidelines.*
*3. **Error Correction Standards**: For any code snippet that fails to execute successfully, you will follow a structured approach to identify and resolve issues. The goal is to diagnose the error's root cause and apply targeted corrections that ensure consistency with the intended analysis query and overall functionality.*

Task-Specific Guidelines:
*1. **Rudimentary Analysis Operations** (Filter, Sort, Aggregation and Group By, Pivot Table)*
*- **Insightfulness**: Verify if the results reveal key characteristics of the data and offer insightful observations.*
*- **User-Friendliness**: Confirm that the output is easily interpretable, and the operation aligns with common data analysis practices.*
*- **Visualization Clarity**: Ensure headers are clearly labeled, and the content is well-organized, without excessive missing values or unclear cells.*

*2. **Chart-Based Analysis**:*
*- **User Interpretability**: Check if the generated chart is clear and easy for users to understand, with a well-defined chart type.*
*- **Presentation Quality**: Assess if there are any visual issues, such as overlapping axes, overly dense data labels, or cluttered layouts that detract from readability.*

Figure 25: Prompt design in TablePilot

Prompt for TablePilot – Multimodal Revision(Cont.)

 - **Field Combinations**: Evaluate if the combination of x-axis and y-axis fields presents meaningful relationships, delivering valuable insights for data analysis. Please ensure the code's execution success rate while improving the clarity and intuitiveness of the charts, so that the user can understand them accurately.
   - **Chart Documentation**: In the modified chart code, add a comment in this format: # Chart INFO: {{'x_fields': '', 'y_fields': [], 'chart_type': ''}}. Here, x_fields specifies the x-axis field, y_fields lists y-axis values (allowing multiple fields), and chart_type defines the chart type (e.g., lineChart, barChart, scatterChart, pieChart).

3. **Statistical Modeling Tasks** (Trend Prediction, Correlation Testing, Regression Modeling)
   - **Trend Prediction**: Confirm the appropriateness of the target variable for forecasting (e.g., time series). Evaluate the prediction window setting and model suitability for the data characteristics. If NaN values occur, please correct the errors in the modeling process and generate valid forecasted values.
   - **Correlation Testing**: Check if the selected variables have a meaningful correlation worth analyzing, beyond obvious or trivial associations.
   - **Regression Modeling**: Ensure the chosen variables are suitable for modeling, with an appropriate regression model based on data linearity or non-linearity.

Code Error Correction Guidelines:
1. **Step-by-Step Diagnosis**: Carefully consider each step of the code to understand the error's root cause. Pinpoint why the code fails when executing the specific data analysis query.
2. **Query and Code Consistency**: Verify that the code accurately implements the query's requirements. Ensure consistency between the query and the code, confirming that the logic aligns with the query's intended analysis.
3. **Error Message Analysis**: Use the details from the error message to identify specific issues. Follow a logical approach, thinking through each possible cause, and apply corrections that logically address the error.

Figure 26: Prompt design in TablePilot

Prompt for TablePilot – Multimodal Revision(Cont.)

````
### Optimized the Successful  Results (This part switches conditioned on whether the execution result is successful or a failure.)

"""Please review and optimize the following content according to the guidelines:
### Table Information:
{table}

The table's explanation is provided below to guide your revision:
## Explanation
{table explanation}

```
### Query Details:
**Query**:
{query}

**Code**:
{code}
`
** Execution Results**:
{Results – text content}

{Results – image content}

Please ensure that the optimized code can produce the correct results; otherwise, do not proceed with the optimization.

### Revise the Error  Results (This part switches conditioned on whether the execution result is successful or a failure.)

The current code matched to the query is incorrect. Please analyze the reasons for the error and suggest how it can be improved. Please review and correct the following content according to the guidelines:
### Table Information:
{table}

The table's explanation is provided below to guide your revision:
## Explanation
{table explanation}

### Error Message:
{error}

Please ensure that the optimized code can produce the correct results.
````

Figure 27: Prompt design in TablePilot
````

Prompt for TablePilot – Ranking

Evaluating High-Quality Data Analysis Recommendations
Task Description:
*As the most senior data analysis manager, you bring extensive experience in identifying and recommending the most valuable tasks generated by other data analysis processes. Your task is to evaluate data analysis operations for a given table. Your input includes a sampled version of the table, relevant explanations about the table, and a set of key data analysis queries along with their execution results. **Adjust the distribution of recommendations across these task types as needed to align with the table's unique data profile.** Ensure that each selected recommendation is of high quality and insight, providing professional-level analysis that will leave users highly satisfied.*

Definitions of Data Analysis Task Categories:
*1. **Basic Data Analysis Tasks**:*
This category covers basic operations like filtering, sorting, grouping, and creating pivot tables to summarize data. These tasks help organize raw data, making it easier to identify trends, compute key metrics, and prepare for deeper analysis.

*2. **Table Visualization Data Analysis Tasks**:*
This category involves visualizing data using charts like line, bar, scatter, pie, column, combo and box charts, along with advanced types like stacked and bubble charts. These tasks allow users to explore patterns and trends, enabling clearer insights and effective decision-making.

*3. **Statistics Modeling Analysis Tasks**:*
This category includes predictive and statistical analyses like trend forecasting, correlation testing, and regression modeling. These tasks provide deeper insights by predicting outcomes, identifying relationships, and supporting data-driven decisions.

Evaluation Criteria:
*1. **Meaningful (Practical Usefulness)**:*
***Concept**: The recommendation's ability to provide practical value in real-world data analysis tasks.*
***Definition**: A meaningful recommendation should address a specific analytical need and provide actionable insights that directly support business decisions. It should offer solutions to key issues within the data and guide users in making informed choices based on the analysis.*
***Good Performance**: A high-quality recommendation effectively addresses real-world problems, aligns with the overall objectives of the analysis, and enables users to gain useful insights that drive decisions or further exploration.*

Figure 28: Prompt design in TablePilot

Prompt for TablePilot – Ranking(Cont.)

2. **Relative (Relevance to the Table Theme)**:
   Concept: The degree to which the recommendation is aligned with the core content and purpose of the dataset.
   Definition: A relevant recommendation should directly relate to the **Table Theme**—the main topic or focus of the dataset being analyzed. The closer the recommendation is to the central theme, the more relevant it becomes.
   Good Performance: A well-aligned recommendation highlights key elements of the table, such as analyzing core columns or offering insights that support the main subject of the table. It enhances the analysis by focusing on the most important data points and their relationships.

3. **Reasonableness (Logical Coherence and Suitability to Data Characteristics)**:
   Concept: The degree to which a recommendation logically aligns with the table's structure and the intrinsic characteristics of its data values.
   Definition: A reasonable recommendation should be logically coherent and grounded in sound data analysis principles that a data analyst would naturally follow. The queries generated should reflect meaningful relationships within the data, and the chosen analysis methods should perfectly match the data's properties, highlighting relevant patterns or insights.
   Good Performance: A well-reasoned recommendation is intuitive, logically structured, and tailored to the data's unique attributes, making it feel like a natural and insightful extension of the data itself. The generated data analysis content should align with the rational understanding and expectations of the data analyst.

4. **Diversity (Variety of Analysis Tasks)**:
   Concept: The extent to which the set of recommendations covers a broad range of data analysis operations.
   Definition: Diversity ensures that within the same type of task, recommendations reflect a range of different data analysis methods and data columns.
   Good Performance: A diverse set of recommendations should focus on each task type, selecting different data analysis methods within each while utilizing various combinations of data columns. For example, choose various operations for Rudimentary Operations using different column sets, different chart types for Chart-Based Data Analysis exploring different data dimensions.

Figure 29: Prompt design in TablePilot

Prompt for TablePilot – Ranking(Cont.)

*5. **Interpretable (Ease of Understanding and Implementation)**:*
***Concept**: The clarity and simplicity of the recommendation in terms of how easily it can be understood and executed by the user.*
***Definition**: An interpretable recommendation should be straightforward, with clear steps that the user can follow without ambiguity. It must be simple enough to be implemented directly and should not require excessive explanation or complex reasoning.*
***Good Performance**: A well-interpreted recommendation is concise, uses plain language, and describes the task in a way that is immediately actionable. Users should be able to quickly grasp its value and apply it without needing additional clarification.*

*6. **Insightful (Ability to Reveal New Data Insights)**:*
***Concept**: The potential of the recommendation to uncover valuable insights or new perspectives from the data.*
***Definition**: An insightful recommendation should offer more than just surface-level observations. It should reveal hidden relationships, highlight trends, or provide a fresh perspective that may not be immediately obvious from the raw data.*
***Good Performance**: A strong recommendation goes beyond basic analysis, helping users to identify significant patterns, correlations, or predictions that could lead to deeper understanding or strategic actions. It often uncovers key insights that were previously unknown or unexplored.*

Evaluation Criteria for Basic Data Analysis
The evaluation of rudimentary data analysis execution results should adhere to the same six principles outlined previously.
*1. **Sort-Type Queries**:*
The Table Data provided represents sequential samples from the original table. When a column in these samples exhibits an ordered sequence, it indicates that the corresponding column in the original table maintains the same ordering pattern. Therefore, any sorting operation on such columns would be redundant.
Exclude sort queries if the sorted results are identical to the original table, as this indicates an ineffective operation.
*2. **Empty or NaN Values**:*
Exclude queries producing results with many empty or NaN values.
*3. **Pivot Table**:*
Carefully evaluate the execution results of pivot tables and retain only those that provide truly insightful data analysis.

Figure 30: Prompt design in TablePilot

Prompt for TablePilot – Ranking(Cont.)

Evaluation Criteria for Charts
The evaluation of chart execution results should adhere to the same six principles outlined previously. However, as charts are presented in image form, additional criteria are necessary to ensure high-quality outputs.
*1. **Clarity of Scales and Labels***
Ensure that the chart includes clear scales and accurately defined data labels, making it easy to interpret the presented data.
*2. **Completeness of Content***
The chart's content must comprehensively reflect the data analysis operation intended by the query, covering all relevant aspects.
*3. **Aesthetic Quality and Richness of Meaning***
The chart should be visually appealing, well-designed, and capable of effectively conveying rich and meaningful insights.

Evaluation Criteria for Statistics Modeling Data Analysis
The evaluation of advanced data analysis execution results should adhere to the same six principles outlined previously.
*1. **Selection of Variables for Analysis**:*
Prioritize advanced modeling or correlation tests for variables with potential relationships, rather than those already exhibiting significant correlations.
*2. **Statistically Significant***
*For Statistics Modeling Data Analysis tasks, please evaluate whether the query results are statistically significant (i.e., MAPE value < 0.1, P-value < 0.05, R-squared > 0.9). Assign higher scores to queries with **statistically significant results** and lower scores to queries without statistical significance.*

Input:
1. A subset of the table obtained through a specific sampling method and table Explanation.
*2. A set of data analysis recommendation queries targeting this table, categorized into three types of tasks: **Rudimentary Data Analysis**, **Chart-Based Data Analysis**, and **Advanced Data Analysis**. Along with their corresponding execution results.*

Figure 31: Prompt design in TablePilot

Prompt for TablePilot – Ranking(Cont.)

Table Data:
{table}

Explanation
{table explanation}

Here are the queries and its results for the three task categories:
1. Basic Data Analysis Queries:
{basic analysis queries}
{basic analysis results}

2. Visualization Data Analysis Queries:
{visualization analysis queries}
{visualization analysis results – image content}

3. Statistics Modeling Data Analysis Queries:
{statistics modeling analysis queries}
{statistics modeling analysis results}

Please evaluate all the queries listed above across the three categories. Each query from these three types of tasks must be evaluated and assigned a score without omitting any.

Please evaluate all queries based on the six dimensions in the Ranking Criteria: Meaningful, Relative, Reasonableness, Diversity, Interpretable, Insightful. Assign a score to each dimension on a scale of 0 to 5, where a higher score indicates that the query result better aligns with that criterion. Additionally, provide an explanation for each score to justify the rating.

Be strict. Comprehensively consider all queries and results to ensure that the evaluation scores exhibit a certain degree of differentiation.

Retain the original query information exactly as it is, without making any modifications to its content.

Figure 32: Prompt design in TablePilot

Prompt for Baseline

Table Analysis Assistant
*You are an advanced data analysis assistant specializing in generating actionable **query and code recommendations** based on a given table and its explanations. Your objective is to create **diverse, practical, and business-relevant queries** spanning three types of tasks:*

*1. **Basic data operations**: Filtering, sorting, grouping & aggregation, pivot table creation, and insightful column insertion.*
*2. **Data Visualization analysis**: Generating various charts like line, bar, scatter, pie, combo, and advanced charts such as stacked bar and bubble charts.*
*3. ** Statistics modeling**: Conducting trend prediction, correlation testing, and regression modeling.*

*For each query, generate **Python code** that:*
- Accurately implements the query using the appropriate libraries (`pandas`, `matplotlib`, `statsmodels`, or `scikit-learn`).
- Fully aligns with the query's intent and logic.
- Outputs the analysis results in a clear and interpretable format.

*## **Query Generation Guidelines***
*1. **Diversity and Variety**: Ensure the queries cover different analysis operations, chart types, and statistical models, utilizing the table's columns comprehensively.*
*2. **Practicality**: Queries must align with real-world data analysis needs and the table's context, avoiding overly generic or irrelevant analyses.*
*3. **Specificity**: Clearly define the scope and purpose of each query to ensure precision in the generated code.*

Please propose some queries for each task along with the corresponding executable code for the following table:

Table Data:
{table}

*## **Code Input ***
- Import `pandas` for data manipulation.
- Import `pandas` and `matplotlib` for chart creation.
- Import `pandas` and the relevant statistical libraries (`statsmodels`, `scikit-learn`, or `numpy`).
```

Figure 33: Prompt design in TablePilot
```

Prompt for Baseline(Cont.)

- **Output Format**:
…

Output Requirements
The output must be a JSON object containing **queries** and corresponding **Python code** for the three task types:

1. **Rudimentary Data Operations**: Queries that involve filtering, sorting, grouping, aggregation, pivot table, or insightful column insertion.
2. **Chart-Based Analysis**: Queries that involve generating different types of charts, clearly specifying the chart type and data scope.
3. **Advanced Statistical Modeling**: Queries that involve statistical analysis tasks such as trend prediction, correlation testing, or regression modeling.

Each query must be aligned with its code, and the JSON object must strictly include only the `query` and `code` fields.

Important Notes
1. **Output Alignment**: Ensure each query's code satisfies the requirements and intent of the query.
2. **Clean Code**: Provide executable code without unnecessary comments or explanations.
3. **No Extra Information**: DO NOT include anything outside the JSON object containing the `query` and `code`.

Figure 34: Prompt design in TablePilot

Prompt for Constructing Dataset - DART

Your task is to predict meaningful user queries based on a given table and its explanations. Recommend diverse and practical queries, each accompanied by the corresponding Python code using the pandas library. The query recommendations should encompass different data analysis operations: filtering, sorting, grouping and aggregation, pivot table operations, and insert insight columns. Your goal is to ensure that both the queries and the code are useful for real-world analysis scenarios based on the table's content. Select the most appropriate operations based on the table's characteristics.

Purpose of Basic Data Analysis Task:
This task involves essential data manipulation operations for organizing raw data, simplifying complex datasets, and generating quick overviews. The purpose of these tasks is to help users streamline their datasets, making it easier to spot trends, derive key metrics, and prepare data for deeper analysis.

Please propose queries and corresponding executable code based on the table provided:
Table:
{table}
This table format is the result of sampling a portion of the original CSV file, providing an overview. Please generate data analysis recommendations for the complete table.
Explanations:
{Explanations}

Figure 35: Prompt design in TablePilot

Prompt for Constructing Dataset – DART(Cont.)

Your task is to predict meaningful chart-based data analysis queries and generate the corresponding Python code using the matplotlib library. Your objective is to recommend some business-relevant chart queries, each accompanied by executable code that matches the query. The chart types can range from basic charts like line, bar, scatter, pie, column, combo and box charts to more complex charts such as clustered bar, stacked bar, 100% stacked bar, area and bubble charts. Your goal is to ensure that both the queries and the code are useful for real-world analysis scenarios based on the table's content and its explanations.

Purpose of Chart-Based Data Analysis Task
This task focuses on visualizing data to enable users to explore patterns, relationships, and trends effectively. By creating clear and compelling visuals, users can gain deeper insights and facilitate decision-making. Chart types vary in complexity and should be selected based on the structure and purpose of the tabular data being analyzed.

Please propose queries and corresponding executable code based on the table provided:
Table:
{table}
This table format is the result of sampling a portion of the original CSV file, providing an overview. Please generate data analysis recommendations for the complete table.
Explanations:
{Explanations}

Figure 36: Prompt design in TablePilot

Prompt for Constructing Dataset – DART(Cont.)

Your task is to predict meaningful data analysis queries and generate the corresponding Python code using appropriate libraries like statsmodels, scikit-learn, and numpy. The analysis tasks focus on trend prediction, correlation testing, and regression modeling. Your objective is to recommend some distinct data analysis queries, each accompanied by executable code that matches the query. Your goal is to ensure that both the queries and the code are useful for real-world analysis scenarios based on the table's content and its explanations.

Purpose of Advanced Data Analysis Task
This task includes predictive and statistical analyses such as trend forecasting using historical data, correlation testing to quantify relationships between variables, and regression modeling to predict outcomes based on one or more independent variables. These tasks are essential for performing in-depth analysis that moves beyond descriptive statistics, offering predictive power and helping users understand the underlying factors that influence key outcomes. The purpose is to support data-driven predictions, identify correlations, and build models that provide actionable insights for future planning and decision-making.

Please propose queries and corresponding executable code based on the table provided:
Table:
{table}
This table format is the result of sampling a portion of the original CSV file, providing an overview. Please generate data analysis recommendations for the complete table.
Explanations:
{Explanations}

Figure 37: Prompt design in TablePilot

Prompt for Constructing DPO Positive Data

Your task is to evaluate data analysis operations for a given table. Your input includes a sampled version of the table, relevant explanations about the table, and a set of key data analysis queries along with their execution results. Your goal is to assess these queries from a professional data analysis perspective, assign a reasonable score and reason based on the following Evaluation Criteria:

1. Meaningful (Practical Usefulness):
- A meaningful recommendation should address a specific analytical need and provide actionable insights that directly support business decisions.
2. Relative (Relevance to the Table Theme):
- A relevant recommendation should directly relate to the "Table Theme"—the main topic or focus of the dataset being analyzed.
3. Reasonableness (Logical Coherence and Suitability to Data Characteristics):
- A reasonable recommendation should be logically coherent and grounded in sound data analysis principles that a data analyst would naturally follow.
4. Diversity (Variety of Analysis Tasks):
- Diversity ensures that within the same type of task, recommendations reflect a range of different data analysis methods and data columns.
5. Interpretable (Ease of Understanding and Implementation):
- An interpretable recommendation should be straightforward, with clear steps that the user can follow without ambiguity.
6. Insightful (Ability to Reveal New Data Insights):
- An insightful recommendation should offer more than just surface-level observations. It should reveal hidden relationships, highlight trends, or provide a fresh perspective that may not be immediately obvious from the raw data.

The above outlines the requirements of your task. Below are the corresponding data points that you need to evaluate:

Table Data:
{table}
Explanation
{table explanation}
Here are the queries and its results for the three task categories:
1. Basic Data Analysis Queries:
{basic analysis queries}
{basic analysis results}
2. Visualization Data Analysis Queries:
{visualization analysis queries}
{visualization analysis results – image content}
3. Statistics Modeling Data Analysis Queries:
{statistics modeling analysis queries}
{statistics modeling analysis results}

Please evaluate all the queries listed above across the three categories. Each query from these three types of tasks must be evaluated and assigned a score without omitting any.

Figure 38: Prompt design in TablePilot

Prompt for Constructing DPO Negative Data

You often make some erroneous judgments about phenomena in the real world and provide absurd and abstract explanations. You will receive some tables, as well as data analysis queries and corresponding results on top of these tables. Please generate random and unreasonable scores for all queries, accompanied by an extremely absurd explanation.

Please generate random scores ranging from positive 100 to negative 100.

Table Data:
{table}

1. Basic Data Analysis Queries:
{basic analysis queries}
{basic analysis results}

2. Visualization Data Analysis Queries:
{visualization analysis queries}
{visualization analysis results}

3. Statistics Modeling Data Analysis Queries:
{statistics modeling analysis queries}
{statistics modeling analysis results}

Figure 39: Prompt design in TablePilot