SCoPE: Planning for Hybrid Querying over Clinical Trial Data

[1]Suparno Roy Chowdhury* [1]Manan Roy Choudhury* [1]Tejas Anvekar
[2]Muhammad Ali Khan [2]Kaneez Zahra Rubab Khakwani [2]Mohamad Bassam Sonbol
[2]Irbaz Bin Riaz† [1]Vivek Gupta†
[1]Arizona State University [2]Mayo Clinic
{srchowd3,mroycho1,tanvekar,vgupt140}@asu.edu
{khan.muhammad2,Khakwani.kaneezzahra,Sonbol.mohamad,riaz.irbaz}@mayo.edu

Project Page Project Code

Abstract

Systematic reviews of clinical trials require analysts to extract attributes that are rarely stored as ready-made columns. For example, the drug class of an immunotherapy named in a regimen, the additional agents combined with it, or whether a listed endpoint is a primary or secondary outcome. These attributes must be inferred from the visible content of other fields through normalization, classification, or structured extraction, and existing approaches such as direct LLM prompting, text-to-SQL, and agentic pipelines leave this reasoning implicit in a single generation step or pay a heavy execution cost for limited accuracy gains. We propose SCoPE (***S**tructured **C**linical hybrid **P**lanning for **E**vidence retrieval in clinical trials*), a multi-LLM planner-based framework that decomposes the task into row selection, structured planning, and execution. The planner makes the source field, reasoning rules, and output constraints explicit before answer generation, reducing ambiguity relative to direct prompting. We evaluate SCoPE on 1,500 hybrid reasoning questions over oncology clinical-trial tables against zero-shot, few-shot, chain-of-thought, TableGPT2, BlendSQL, and EHRAgent. Results show that explicit multi-LLM planning improves accuracy for reasoning-based questions while offering a stronger accuracy-efficiency tradeoff than heavier agentic baselines. Our findings position clinical trial reasoning as a distinct table understanding problem and highlight hybrid planner-based decomposition as an effective solution.

Question: For Pembrolizumab trials list **additional agents** in the treatment regimen beyond the ICI.

NCT	Treatment Regiment	Added Agents
NCT0203	Pembrolizumab+**Pemetrexed**	["pemetrexed"]
NCT0204	Pembrolizumab+**Pemetrexed +Carboplatin**	["pemetrexed", "carboplatin"]
NCT0206	Pembrolizumab+**Platinum**	["platinum"]
NCT0207	Pembrolizumab+**Pemetrexed**	["pemetrexed"]

Gold Answer: Every agent in Treatment Regimen **besides Pembrolizumab**. Eg, [Pemetrexed+ Carboplatin]

Figure 1: Sample exemplar hybrid reasoning question requiring semantic understanding of regimen structure: the model must recognize **Pembrolizumab** as the ICI named in the question and extract all other treatment agents from the visible regimen text as the held-out target field. For instance, in NCT0203, **Pembrolizumab** is the ICI and **Pemetrexed** is the additional agent used in the treatment regimen

1 Introduction

Clinicians and medical researchers often analyze tables of clinical trials with structured records where each row summarizes one study. Many questions over these tables can be answered by simple lookup: for example, what was the sample size of *Pembrolizumab drug trials*? But many questions require a deeper set of reasoning and cannot be answered immediately. For example, a reviewer studying Pembrolizumab drug trials which uses additional agents in the complete treatment regimen can surmise that, in a trial using Pembrolizumab + Pemetrexed, Pemetrexed medication is the additional agent in this treatment regimen. In these cases, the target value is not explicitly stored in a visible cell and cannot be obtained by simply selecting rows or projecting an existing column. Instead, it must typically be reasoned from row contents through normalization, classification, extraction, aggregation, or lightweight domain reasoning. We study this setting as *clinical trial reasoning*: given a question and a partially observed table, the clinicians must identify the relevant rows, ground the visible source field containing the evidence, and apply the correct row-wise transformation to produce

*These authors contributed equally.
†Corresponding authors.

Proceedings of the 25th Workshop on Biomedical Language Processing (BioNLP 2026), pages 944–954

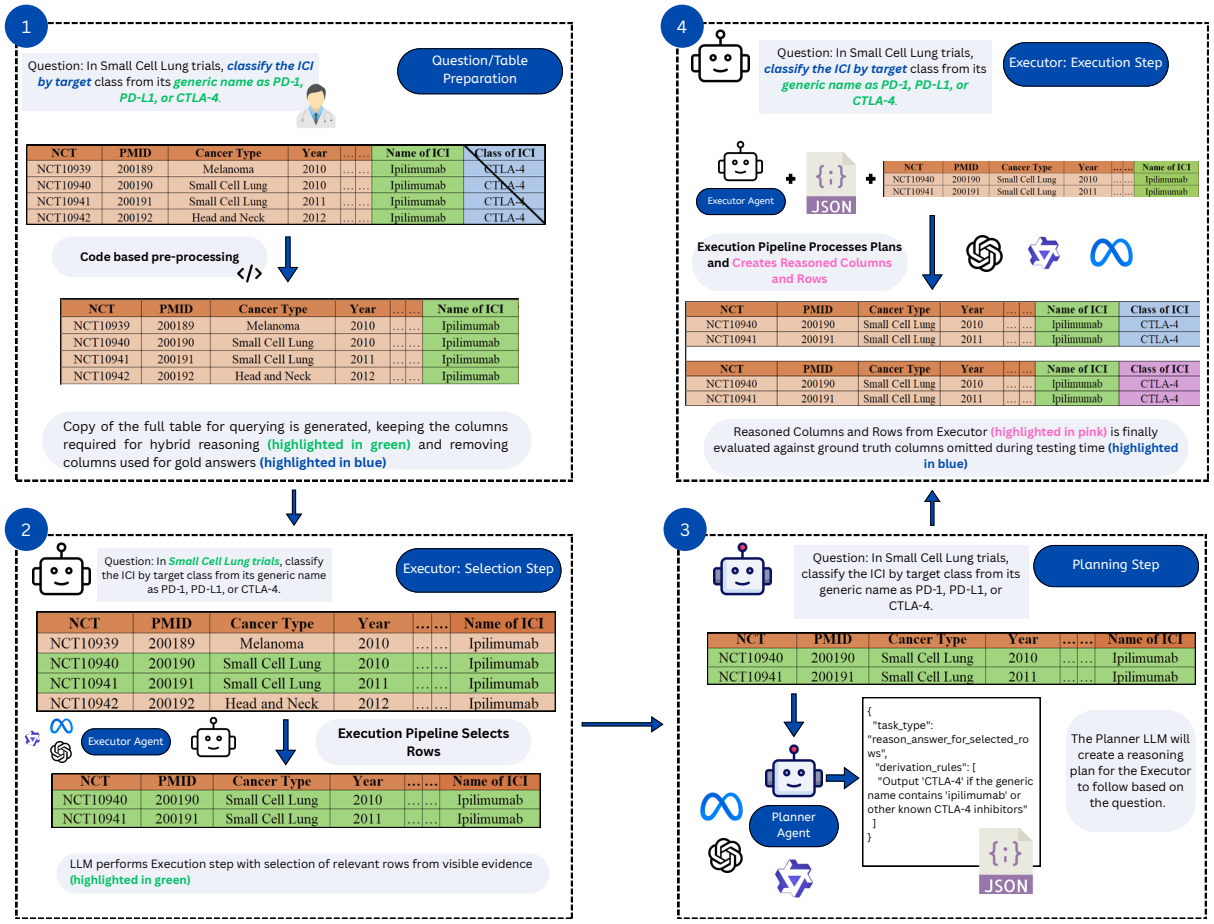

Figure 2: SCOPE for clinical tabular reasoning. Given a question and a copied visible table, SCOPE first prepares an inference-time table view by retaining the visible evidence columns and withholding the target column used for evaluation if there are any. The executor identifies the rows relevant to the question, the planner produces a structured reasoning plan over the selected table, and the executor follows this plan to generate the final row-aligned predictions. In this example, the method reasons over the *Name of ICI* column to recover the held-out *Class of ICI* field, and the resulting predictions are compared against the omitted ground-truth targets.

a target field that is often absent from the schema or from the evidence itself. This setting is related to table question answering and semantic parsing, but is only partially captured by existing table QA benchmarks and classical text-to-SQL formulations (Pasupat and Liang, 2015; Zhong et al., 2017; Yu et al., 2018; Chen et al., 2019; Nan et al., 2022).

Existing approaches do not fully address this problem. Text-to-SQL methods assume that the answer can be recovered through executable operations over explicit schema values, which is often not true when the target field is latent rather than stored (Zhong et al., 2017; Yu et al., 2018; Chowdhury et al., 2026). Prompting with large language models often under-structures the task, typically relying on a single generation step to retrieve rows, infer the evidence-bearing field, apply the transformation, and format row-aligned answers. More structured agentic and program-based approaches can improve reasoning, but can introduce additional execution overhead and brittle multi-step traces (Gao et al., 2023; Cheng et al., 2022; Yao et al., 2022; Glenn et al., 2024; Shi et al., 2024). So, the main research question we are trying to address is: *How effective is explicit hybrid planner-based decomposition in improving grounded row-level reasoning over partially observed clinical-trial tables, and do the resulting gains appear more clearly in row-level grounding than in aggregate answer recovery?*

To address these limitations, we introduce SCOPE, a planner-based framework for clinical querying that explicitly decomposes the task into row selection, structured planning, and execution. The executor grounds the question to relevant rows in the visible table and produces row-aligned outputs, while the planner interprets the question as

a structured transformation over the selected evidence. Concretely, the executor first identifies a candidate subset of rows, the planner converts the question into an explicit transformation specification, and the executor then applies that specification to generate aligned predictions. This decomposition makes the source-field choice, reasoning rules, and output constraints explicit prior to answer generation, improving interpretability and reducing ambiguity compared to direct prompting. By introducing an explicit planning interface between reasoning and execution, SCOPE better aligns the model behavior with the compositional structure of clinical trial querying.

We also construct a benchmark of 1,500 hybrid reasoning questions over oncology clinical-trial tables, where target attributes must be inferred from visible evidence rather than directly retrieved.

Unlike approaches that keep planning implicit in a single generation step or rely on heavier agentic pipelines, SCOPE introduces a lightweight planning interface that separates reasoning specification from execution while preserving grounding in the visible table. We evaluate SCOPE against zero-shot, few-shot, chain-of-thought, TableGPT2, BlendSQL, and EHRAgent, and find that explicit planning improves reasoning accuracy while offering a better accuracy-efficiency tradeoff than heavier agentic baselines.

Our main contributions are:

- We introduce SCOPE, a planner-based framework that decomposes clinical trial reasoning into row selection, source-field grounding, and structured execution.
- We formalize *clinical trial reasoning* as a distinct table understanding setting requiring inference over visible row evidence rather than direct cell retrieval.
- We construct a benchmark of 1,500 oncology clinical-trial questions and show that explicit planning outperforms prompting baselines while remaining more efficient than heavier agentic methods.

The code, scripts, and dataset are available at our project page.

Problem formulation. Given a question $\tilde{q}$ and a visible table $X = \{r_1, \ldots, r_m\}$ with a hidden target field, the goal is to recover a set of row-aligned predictions $\hat{G} = \{(r, \hat{y}_r)\}$ for the subset of rows relevant to the query. Each $\hat{y}_r$ must be derived from visible evidence in r rather than retrieved directly from a stored field. Errors may arise from incorrect row selection, incorrect source-field grounding, or incorrect transformation of visible evidence, making joint reasoning over these steps essential.

2 Related Work

We situate our work at the intersection of clinical datasets and table reasoning methods, focusing on settings where answers must be inferred rather than directly retrieved.

Related Datasets. Clinical datasets such as MIMIC III & IV (Johnson et al., 2016, 2023) and the eICU Collaborative Research Database (Pollard et al., 2018) have enabled large-scale modeling over structured and semi-structured medical records, while i2b2/n2c2 shared tasks (Uzuner et al., 2011) focus on extracting structured information from clinical text. In table reasoning, datasets such as TabFact (Chen et al., 2019) and FETAQA (Nan et al., 2022) study fact verification and free-form question answering over tables, assuming that answers are grounded in explicit table entries. Our benchmark construction is also motivated by recent domain-specific benchmarks that stress-test LLM capabilities on nuanced domain specific reasoning tasks (Choudhury et al., 2025, 2026). Given that and in contrast, our benchmark targets clinical-trial reasoning, where target attributes are often latent and must be inferred from visible row evidence through structured transformations.

Related Methods. Early table reasoning and semantic parsing approaches, such as Seq2SQL, TaBERT, and TAPAS (Zhong et al., 2017; Yin et al., 2020; Herzig et al., 2020), map natural language to executable queries or learned table representations and are strongest when the answer is explicitly represented in the schema. More recent LLM-based methods, including PAL and ReAct (Gao et al., 2023; Yao et al., 2022), introduce intermediate reasoning and action steps, while TableGPT2, BlendSQL, and EHRAgent (Li et al., 2024; Glenn et al., 2024; Shi et al., 2024) add stronger structure for reasoning over tables and clinical data. However, these methods either keep planning implicit within generation or rely on heavier agentic pipelines with iterative execution. In contrast, SCOPE explicitly decomposes the task into row selection, source-field grounding, and structured execution, making

the reasoning process more transparent and controllable while achieving a better accuracy-efficiency tradeoff.

3 Dataset

3.1 Dataset Construction

We begin with a seed set of 500 questions authored by a certified oncology researcher to reflect realistic evidence-review queries over a structured clinical-trial table containing bibliographic metadata, disease labels, treatment regimens, control-arm descriptions, endpoint text, trial phase, sample size, and follow-up information. Each seed example is paired with an executable SQLite query, which serves as a scaffold for generating new questions and aligned ground truth. We retain the subset of table-solvable questions that require hybrid reasoning rather than simple retrieval, and expand it to 1,500 questions through programmatic augmentation. For each retained pair (q, s), we generate a new pair $(\tilde{q}, \tilde{s}) = T(q, s)$ by applying a single atomic edit to the structured query, such as modifying the projection, simplifying predicates, swapping attribute values, relaxing thresholds, or changing the evidence field while preserving answerability. We keep only read-only variants that execute successfully and return non-empty results, then de-duplicate the resulting query patterns and rewrite each retained structured query into fluent natural language while preserving its exact semantics.

Ground-Truth Construction. For each final question $\tilde{q}$, the corresponding query $\tilde{s}$ identifies a set of relevant rows $R = \{r_1, \ldots, r_n\}$ together with a visible evidence field c_{src} used to derive the hidden target. We then apply a deterministic row-level reasoning function f to each row to produce the ground-truth target value $y_i = f(r_i[c_{\text{src}}])$ for every $r_i \in R$. This function implements the intended task semantics for normalization, categorization, boolean interpretation, and structured extraction, such as mapping an ICI mention to its class or brand name, determining endpoint role from endpoint text, bucketing follow-up duration or publication year, or extracting additional agents, schedules, regimen components, and control backbones from treatment text. To form the benchmark input, we expose only the visible table copy while omitting the target values $\{y_i\}_{i=1}^{n}$. The ground truth is stored separately as the row-aligned pairing $G = \{(r_i, y_i)\}_{i=1}^{n}$.

Clinical-Trial Table Statistic	Value
Rows	159
Columns	32
Unique trials (NCT)	105
Cancer types	19
ICI names	13

Table 1: Statistics of the underlying clinical-trial table used to construct the benchmark. The table covers trial-level oncology evidence, including study metadata, disease labels, treatment and control regimens, endpoint descriptions, and follow-up information, spanning 105 unique trials across 19 cancer types and 13 immune checkpoint inhibitor (ICI) names.

Crucially, each target value is derived from evidence already present in the table rather than authored independently or requiring external domain knowledge at evaluation time. This design ensures that the benchmark evaluates hybrid reasoning over visible evidence: models must infer latent attributes by interpreting existing table content, rather than generating plausible but unsupported answers.

3.2 Dataset Guidelines

To create the seed set, a certified clinician authored questions based on the IOTOX living evidence table[1]. They were asked to write realistic evidence-review questions while following three guidelines:

- Each question had to be grounded in visible table evidence.
- The expected answer had to be precise and structured, typically as a JSON value.
- The answer had to be inferable from the table itself, without requiring unsupported external knowledge.

3.3 Dataset Statistics

As shown in Tables 1 and 2, the benchmark is built from a structured clinical-trial table with 159 records and 32 columns, spanning 105 unique NCT identifiers, 131 unique PubMed IDs, 95 trial names, 19 cancer types, and 13 ICI names from 2010 to 2021. Although compact, the table is information-dense: many target outputs are not exposed as ready-made columns, but must be inferred from regimen text, endpoint descriptions, follow-up fields, and bibliographic metadata. At the benchmark level, the dataset contains 1,500 questions with

[1] https://iotox.living-evidence.com/

Question Statistic	Value
Total questions	1,500
Mean question length (tokens)	21.2
Target fields	31
Answer Types	**Value**
String	957
List	241
Boolean	224
Null-only	78

Table 2: Question and answer-level benchmark statistics. *Total questions* gives the size of the benchmark; *Mean question length* reports the average number of tokens per natural-language question; *Target fields* gives the number of distinct hidden target types covered by the dataset for reasoning . On the answer side, we report the number of questions whose gold supervision is primarily *string-valued*, *list-valued*, *boolean-valued*, or *null-only*, where null-only questions are those whose held-out target values are absent for all aligned rows.

an average length of 21.2 tokens and covers 31 distinct target fields. The answer space is also diverse, including string-valued, list-valued, boolean, and null-only outputs, highlighting that the benchmark evaluates hybrid reasoning across multiple target types rather than a single narrow form of table question answering.

4 The SCOPE System & Flow

Given a question $\tilde{q}$ and a visible table copy

$$X = \{r_1, \ldots, r_m\},$$

where the answer-bearing target field is omitted but row identifiers and visible evidence columns are preserved, SCOPE uses two LLM components: an *executor* and a *planner*. The executor grounds the question to relevant rows and produces the final row-level outputs, while the planner interprets the selected evidence as a structured reasoning problem. The planner may use the same backbone as the executor or a different one. We provide the row-selection, planning, and final-execution prompts in Appendix A.1.

The method has three steps. First, the executor identifies a candidate subset of relevant rows from the full visible table. Second, the planner receives the question and this candidate table and produces an explicit reasoning plan. Third, the executor follows that plan to generate the final row-aligned output table. In short, the planner determines *what* to derive and from *which* visible evidence, while the executor determines *where* that evidence occurs and returns the final answers.

Executor: Row Selection. The first step grounds the question $\tilde{q}$ to the visible table X. The executor selects a candidate subset of rows

$$\hat{R} = E_{\text{sel}}(\tilde{q}, X) \subseteq X.$$

At this stage, the executor is used only to identify relevant rows; it does not attempt to derive the final answer.

Planner: Structured Reasoning. The planner then receives the question $\tilde{q}$ and the candidate table $\hat{R}$ selected by the executor, and predicts a structured plan

$$\pi = P(\tilde{q}, \hat{R}) = \big(c_{\text{src}}, C_{\text{rel}}, \rho, o\big),$$

where c_{src} denotes the inferred visible source column that contains the supporting evidence, C_{rel} denotes the subset of visible columns retained for downstream execution, ρ denotes the reasoning rules needed to transform the source evidence into the target field, and o denotes the output constraints that specify the expected answer format.

Executor: Final Table Generation. Given the planner output π, the executor operates on the focused candidate table

$$X_\pi = \hat{R}[C_{\text{rel}}],$$

and produces the final row-aligned prediction table

$$\hat{G} = E_{\text{ans}}(\tilde{q}, \pi, X_\pi) = \{(r, \hat{y}_r) \mid r \in \hat{R}\}.$$

Each predicted value $\hat{y}_r$ is keyed by its row r, so the output remains aligned to the rows in the candidate table rather than collapsing the question into a single free-form answer.

Why Planning Helps. A key advantage of planning is that it explicitly separates decision points that are otherwise entangled in single-step generation. In particular, SCOPE decomposes (i) row grounding, (ii) source-field identification, and (iii) transformation into distinct stages, each of which can fail independently in direct prompting. By externalizing these decisions into a structured plan, the model avoids implicit assumptions about where evidence resides and how it should be transformed. This leads to more stable row alignment and reduces cascading errors compared to approaches where these steps are implicitly interleaved.

5 Experimental Setup

5.1 Baselines

We compare SCoPE against three baseline families under the same input setting. Every method receives the same question $\tilde{q}$ and visible table copy X, where the target field is omitted at inference time. When a method produces free text, SQL, or other intermediate outputs, we normalize its result programmatically into the shared row-aligned prediction format

$$\hat{G} = \{(\hat{r}_i, \hat{y}_i)\}_{i=1}^{k},$$

so that all systems are evaluated against the same held-out ground truth G.

The first family consists of zero-shot, few-shot, and chain-of-thought prompting (Brown et al., 2020; Wei et al., 2022), where the model is given the full visible table X and asked to recover the omitted target field directly without an explicit planning stage. The second includes BlendSQL (Glenn et al., 2024) and EHRAgent (Shi et al., 2024), which introduce more intermediate tabular structure reasoning through SQL-like compositional reasoning or heavier agentic execution. The third includes TableGPT2 (Li et al., 2024), a dedicated tabular reasoning model that jointly consumes the table and question to predict the hidden target field directly from the table representation.

5.2 Models and Hyperparameters

We evaluate three LLMs, chosen to span distinct model families, parameter scales, and post-training recipes while remaining fully reproducible: `Qwen3-30B-A3B-Instruct-2507` (Yang et al., 2025), `gpt-oss-20b` (Agarwal et al., 2025), and `Llama-3.3-70B-Instruct` (Grattafiori et al., 2024). All three are open-weight, general-purpose instruction-tuned models that are not table-specialized, which lets us test whether SCoPE's gains generalize across model lineages and isolate the contribution of explicit planning from any table-specific pretraining. All generations use deterministic decoding with temperature 0.0 and top-p 1.0. Direct prompting baselines are allowed up to 2048 generated tokens per question. For SCoPE, the planner and executor are chosen from the same model pool, enabling both same-model and cross-model configurations. The row-selection step uses 768 tokens, the planner 1024, and the final executor step 2048, keeping execution lightweight while reserving more capacity for structured planning.

5.3 Evaluation Metrics

We evaluate each predicted table against the held-out ground-truth table using three grounded metrics after one-to-one row alignment. **Table F1** is our primary metric and measures how well a method recovers the correct hidden target values for the correct rows, balancing missed and spurious predictions. **Grounded Row Jaccard** gives a stricter overlap-based view by measuring the intersection-over-union between predicted and gold row-level outputs. **Grounded Fowlkes-Mallows** (Fowlkes and Mallows, 1983) complements these metrics by summarizing the balance between grounded precision and recall. Because the target field is omitted at inference time and restored only in held-out supervision, all three metrics evaluate recovery of hidden row-level attributes from visible evidence rather than direct copying.

Impact of row selection errors. Incorrect row selection directly affects all three metrics: missing relevant rows lowers recall, while selecting spurious rows lowers precision and overlap. As a result, the metrics jointly capture both row grounding and answer correctness.

6 Results and Discussion

6.1 Main Results

Across all backbone families, the most important result is that SCoPE consistently achieves the strongest grounded performance while remaining lightweight relative to heavier structured baselines. The framework is best overall on GPT-OSS and Qwen3, and tied on Table F1 while stronger on grounding metrics for Llama-3.3. The clearest gain appears for Qwen3, where explicit planning substantially improves row-aligned reasoning over direct prompting.

RQ1: Does explicit planning improve performance over direct prompting and tabular baselines? Yes. Table 3 shows that SCoPE is strongest or tied for strongest across all three model families on the grounded metrics. It consistently outperforms BlendSQL, EHRAgent, and TableGPT2, and also improves over the strongest prompting baselines in most settings. This supports the central claim of the paper: explicit planning is more effective than leaving row grounding and reasoning implicit inside a single generation step.

RQ2: Are these gains consistent across backbone models? Largely yes, although the size of

Method	Qwen3			Llama-3.3			GPT-OSS		
	F1 (%)	RJ (%)	FM (%)	F1 (%)	RJ (%)	FM (%)	F1 (%)	RJ (%)	FM (%)
Tabular Reasoning Methods									
BlendSQL	11.56	6.52	20.63	5.60	5.15	5.81	7.30	6.48	7.71
EHRAgent	32.99	29.79	33.74	30.99	28.07	31.69	34.85	31.23	35.65
Prompting Technique Methods									
Zero Shot	56.32	44.95	62.73	66.96	54.55	72.04	73.50	61.05	77.47
CoT	55.37	44.65	61.93	**70.87**	57.83	75.15	74.17	61.77	78.05
Few-Shot	54.74	44.09	61.48	69.38	56.56	74.05	73.99	61.55	77.85
Table Model Method									
TableGPT2	**F1:** 44.03			**RJ:** 33.78			**FM:** 50.81		
Ours: SCoPE									
SCoPE	**63.19**	**52.07**	**69.45**	**70.87**	**60.66**	**76.12**	**74.31**	**62.48**	**78.27**

Table 3: Per-model comparison across baselines and same-backbone SCoPE, where each listed model is used as *both* the planner and the executor. We report **F1** (Table F1), **RJ** (Row Jaccard), and **FM** (Fowlkes-Mallows). Higher is better for all metrics. Best values are shown in bold green. TableGPT2 is a table based model which does not have a Qwen3, Llama-3.3, or GPT-OSS backbone so it is tested individually.

the gain varies by model. For GPT-OSS, SCoPE improves over the strongest prompting baseline from 74.17 to 74.31 Table F1 while also yielding the best grounded overlap scores. For Llama-3.3, it matches the best prompting Table F1 (70.87) but improves Row Jaccard and Fowlkes-Mallows, indicating better row alignment. For Qwen3, the gains are largest, showing that the framework generalizes across backbones while being especially useful for weaker implicit reasoners.

RQ3: Which models benefit most from explicit planning? Qwen3 benefits the most. Its Table F1 increases from 56.32 under the best prompting baseline to 63.19 with SCoPE, with corresponding improvements in Row Jaccard and Fowlkes-Mallows. This suggests that explicit planning is particularly helpful when the underlying model is less reliable at performing row grounding and structured reasoning implicitly.

Note. In addition to Table F1, grounded Row Jaccard (RJ) complements evaluation by measuring intersection-over-union over row-aligned outputs, directly capturing joint correctness of row selection and target recovery. This is especially relevant here, where answers must be derived from the correct subset of rows rather than retrieved from explicit cells. Empirically, RJ shows stronger grounding quality: for Llama-3.3, SCoPE matches the best Table F1 (70.87) while increasing RJ from 57.83 to 60.66; for Qwen3, F1 improves by +6.87 (56.32 $\rightarrow$ 63.19) with a larger RJ gain of +7.12 (44.95 $\rightarrow$ 52.07); and for GPT-OSS, nearly unchanged F1 (74.17 $\rightarrow$ 74.31) still yields a consistent RJ increase (61.77 $\rightarrow$ 62.48).

Executor	Planner	F1(%)	RJ(%)	FM(%)
GPT-OSS	Qwen3	75.07	63.74	79.26
Qwen3	GPT-OSS	59.59	48.26	66.32
Qwen3	Llama-3.3	62.47	51.40	68.88
GPT-OSS	Llama-3.3	**75.12**	**63.88**	**79.28**
Llama-3.3	GPT-OSS	68.01	57.37	73.64
Llama-3.3	Qwen3	71.43	61.33	76.64

Table 4: Cross-model ablation of SCoPE. Rows report configurations where the executor and planner use different backbones. Higher is better. The best cross-model result in each metric is shown in bold green.

6.2 Cross-Model Ablation

We vary the planner and executor backbones to test whether planning and execution benefit from different model strengths. In particular, this ablation asks whether a stronger planner can improve performance even when the execution model is held fixed, and whether these two roles are better treated as complementary rather than identical.

Table 4 shows that changing the planner can indeed improve performance. The clearest example is GPT-OSS: when used as both planner and executor in Table 3, SCoPE reaches 74.31 Table F1, but replacing the planner with Llama-3.3 increases performance to 75.12, with similar gains in Row Jaccard and Fowlkes-Mallows. This suggests that the planner benefits from a stronger reasoning

Coder Model	F1(%)	RJ(%)	FM(%)
GPT-OSS	14.56	11.67	23.39
Qwen3	48.92	37.53	56.01
Llama-3.3	**63.40**	**51.55**	**69.10**

Table 5: Results for the planner-coder baseline across coder backbones. The same models are used for the planning and code generation execution stage. Best results are shown in bold green.

model even when execution remains fixed.

Another example appears in the Llama-3.3 executor family. In the same-backbone setting, Llama-3.3 reaches 70.87 Table F1, but replacing the planner with Qwen3 improves performance to 71.43, with corresponding gains in Row Jaccard and Fowlkes-Mallows. Together with the GPT-OSS results, this suggests that changing the planner can improve performance even when the executor is held fixed.

6.3 Planner-Coder Baseline

We include the planner-coder setting as an ablation of whether execution should be made more explicit through programmatic synthesis. The idea is that, if the planner already produces a structured reasoning trace, then compiling that trace into deterministic Python code could make execution more transparent and reproducible. This tests an alternative to grounded LLM execution: executing the plan by first translating it into code.

Table 5 shows that this formulation performs substantially worse than SCOPE across all backbones. The drop is especially large for GPT-OSS (from 74.21 to 14.56 Table F1) and Qwen3 (from 63.19 to 48.92), while even the best planner-coder result with Llama-3.3 (63.40) remains below the strongest same-backbone SCOPE systems. This suggests that code-generation-based execution is more brittle: the model must infer the procedure, translate it into executable code, and ensure that the code still matches the table context and output format. Overall, hybrid clinical table reasoning appears to benefit more from constrained grounded execution than from open-ended code synthesis.

7 Model Cost Effectiveness Analysis

Figure 3 compares answer quality against token budget in the Qwen-based setting. Although TableGPT2 is not Qwen-based, we include it as a reference point for cost-effectiveness against a dedicated table model. SCOPE lies on the strongest

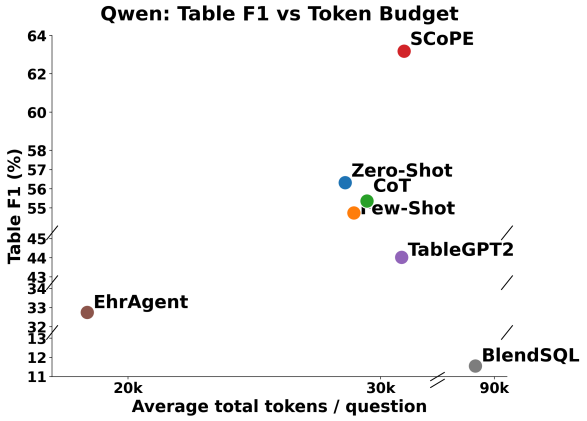

Figure 3: Cost-effectiveness comparison for Qwen-based methods. The x-axis shows average total tokens per question and the y-axis shows **Table F1**.

accuracy-cost frontier, achieving the highest Table F1 at roughly 63% with only a modestly larger token budget than direct prompting baselines. Zero-shot, few-shot, and chain-of-thought prompting cluster in the 28k-30k token range but remain in the mid-50% F1 range, while EHRAgent is cheaper but much less accurate and BlendSQL is both the most expensive and the weakest. Overall, SCOPE adds limited planning overhead while yielding the best accuracy, making it the most favorable cost-effective choice in this comparison.

8 Conclusion

We introduced SCOPE, a planner-based framework for clinical trial table reasoning that separates row grounding, structured planning, & row-level execution over visible evidence. Across same-backbone & cross-model settings, explicit planning consistently improves grounded reasoning performance over direct prompting & stronger structured baselines such as BlendSQL, EHRAgent, & TableGPT2. The gains are especially pronounced for weaker implicit reasoners while remaining competitive for stronger backbones. These findings position clinical trial reasoning as a distinct table understanding problem & suggest that lightweight planner-executor decomposition is an effective and cost-conscious approach for recovering latent clinical attributes from partially observed oncology tables.

Limitations

Our study has four main limitations. First, we evaluate strong open-weight/open-access models but do not include the latest frontier proprietary systems, so the measured gains from explicit planning may

change under stronger closed models. Second, we analyze same-backbone, cross-model, and planner-coder settings, but we do not fine-tune planner and executor roles separately; role-specialized training could alter the observed tradeoffs. Third, we do not run a dedicated memorization analysis. Because the benchmark is derived from a compact table and programmatically expanded from a seed set, some patterns may be partially recoverable from prior exposure rather than purely from visible evidence. Fourth, the benchmark is limited to a single oncology clinical-trial table, so broader validation across additional clinical domains, schemas, and larger real-world evidence tables is still needed.

Ethics Statement

This work supports oncology evidence review and clinical-trial exploration, not patient-specific diagnosis or treatment decisions. SCOPE uses public trial-level records only and does not use patient notes, PHI, or PII. No human subjects were recruited, and no new patient data were collected or released.

Because LLM outputs can be incorrect or overconfident, we emphasize auditability and grounding in visible table evidence. Evaluation is based on row-aligned recovery of omitted fields, and the pipeline exposes planning and execution steps for inspection. We also restrict execution to read-only operations and apply schema validation and timeout safeguards. Our benchmark construction keeps only valid read-only variants, which improves consistency but may bias coverage toward frequent query patterns.

While the seed questions are authored by verified clinicians, he benchmark reflects a constrained setting: a single oncology clinical-trial table, without longitudinal patient records or free-text clinical narratives.

Performance on this benchmark should not be interpreted as clinical reliability. Outputs from SCOPE require expert review before practical use. Real-world deployment should include access controls, logging, and clear warnings about model limitations and possible hallucinations.

Acknowledgment

This research was supported by the Mayo Clinic and Arizona State University Alliance for Health Care Collaborative Research Seed Grant Program (Award ID: AWD00041508; Sponsor Award ID: ARI-358187) for the project: 'Artificial intelligence-assisted digital, living, interactive clinical practice guidelines for cancer providers and patients

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, and 1 others. 2022. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*.

Manan Roy Choudhury, Adithya Chandramouli, Mannan Anand, and Vivek Gupta. 2026. Better call CLAUSE: A discrepancy benchmark for auditing LLMs legal reasoning capabilities. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5776–5818, Rabat, Morocco. Association for Computational Linguistics.

Manan Roy Choudhury, Anirudh Iyengar Kaniyar Narayana Iyengar, Shikhhar Siingh, Sugeeth Puranam, and Vivek Gupta. 2025. TABARD: A novel benchmark for tabular anomaly analysis, reasoning and detection. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21783–21817, Suzhou, China. Association for Computational Linguistics.

Suparno Roy Chowdhury, Tejas Anvekar, Manan Roy Choudhury, Muhammad Ali Khan, Kaneez Zahra Rubab Khakwani, Mohamad Bassam Sonbol, Irbaz Bin Riaz, and Vivek Gupta. 2026. Fd-nl2sql: Feedback-driven clinical nl2sql that improves with use. *arXiv preprint arXiv:2604.15646*.

Edward B Fowlkes and Colin L Mallows. 1983. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language

models. In *International conference on machine learning*, pages 10764–10799. PMLR.

Parker Glenn, Parag Dakle, Liang Wang, and Preethi Raghavan. 2024. Blendsql: A scalable dialect for unifying hybrid question answering in relational algebra. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 453–466.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4320–4333.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, and 1 others. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024. Table-gpt: Table fine-tuned gpt for diverse table tasks. *Proceedings of the ACM on Management of Data*, 2(3):1–28.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, and 1 others. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):180178.

Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C Ho, Carl Yang, and May Dongmei Wang. 2024. Ehragent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22315–22339.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8413–8426.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Supplementary Material

A.1 Planner Prompts

We provide the three inference-time prompt variants used by SCOPE below. Together, they implement the core stages of the pipeline: the selector-executor prompt identifies the candidate rows relevant to the question, the planner prompt converts the question and candidate table into an explicit reasoning plan, and the planner-executor prompt uses that plan to produce the final row-aligned predictions. Including these prompts makes clear how SCOPE externalizes row grounding, source-field selection, and answer generation rather than leaving them implicit in a single generation step.

More information on error type analsysi and prompts utilized for the baseline The code are available at our project code base underneath the *appendices* tab.

Prompt A: Plan Executor

```
You are the final executor stage of a clinical-trials
    ↪ table reasoning system.
Use the planner plan and the visible candidate table to
    ↪ derive the answer for each row.

Return ONLY valid JSON in this exact shape:
{"predictions":[{"table_row_id": 1, "answer": ...}]}

Rules:
- Return one prediction for each row in the candidate
    ↪ table.
- Use the integer "rowid" values from the candidate table
    ↪ as table row id.
- Use the planner's inferred source column and derivation
    ↪ rules when helpful.
- Derive the answer from visible values and visible row
    ↪ context.
- Do not assume the answer already appears verbatim in the
    ↪  table.
- Follow the planner's normalization rules whenever
    ↪ possible.
- Do not add explanations outside the JSON.
```

Prompt B: Planner

```
You are the planner stage of a clinical-trials hybrid
    ↪ querying system.

Overall context:
{{overall_context}}

Your job is to understand the essence of the question and
    ↪ help another model derive the answer implied by
    ↪ the question for each selected row.
The answer may need classification, normalization,
    ↪ extraction, boolean inference, or simple
    ↪ transformation.
Focus on what must be derived from visible source values
    ↪ and row context.

Return ONLY valid JSON in this exact shape:
{
  "task_type": "derive_answer_for_selected_rows",
  "inferred_source_column": "...",
  "derived_field_description": "...",
  "answer_type": "string|boolean|number|list|object",
  "relevant_columns": ["__rowid__", "..."],
  "row_filter_restatement": "...",
  "derivation_rules": ["...", "..."],
  "normalization_rules": ["...", "..."],
  "output_constraints": ["...", "..."],
  "notes": "..."
}

Guidance:
- 'inferred_source_column' should name the visible column
    ↪ that most likely contains the source evidence the
    ↪  executor should derive from.
- 'relevant_columns' should be a focused subset of the
    ↪ visible headers that the executor needs in the
    ↪ final step.
- 'derivation_rules' should explain how to derive the
    ↪ answer from visible source values and context.
- 'normalization_rules' should specify canonical output
    ↪ formatting when relevant.

Question:
{{question}}

Visible headers:
{{headers_json}}

Candidate row count:
{{candidate_row_count}}

Candidate preview JSON:
{{candidate_preview_json}}
```

Prompt C: Selector-Executor

```
You are the executor stage of a clinical-trials table
    ↪ reasoning system.
Your current job is only to identify which visible rows
    ↪ satisfy the question.
Do not derive the final answer yet.

Return ONLY valid JSON in this exact shape:
{"selected_row_ids":[1,2,3],"selection_reason":"...","
    ↪ needs_derivation":true}

Rules:
- Use only the visible table content.
- Use the integer "__rowid__" values from the table.
- If no rows satisfy the question, return {"
    ↪ selected_row_ids":[],"selection_reason":"...","
    ↪ needs_derivation":true}.
- Do not add explanations outside the JSON.

Question: {{question}}

Visible headers:
{{headers_json}}

Visible table (CSV):
{{table_csv_text}}
```