Fast, Accurate, and Privacy-Compliant Table Summarization via Offline Template Generation

Ye Yuan[1, 2]* **Mohammad Amin Shabani**[3] **Siqi Liu**[3]

[1] McGill University, [2] Mila - Quebec AI Institute, [3] RBC Borealis

Abstract

Query-focused table summarization aims to generate concise textual summaries of tabular data conditioned on a specific user query, enabling users to access relevant insights over large tables. However, existing approaches struggle to meet real-world requirements: fine-tuned LLMs, retrieval-augmented generation (RAG) pipelines, and direct LLM summarization face token-limit constraints, high computational costs, and privacy risks, while decomposition-based strategies and manual template design are labor-intensive and non-scalable. To address these challenges, we introduce **FACTS**, *a **F**ast, **A**ccurate, and Privacy-**C**ompliant **T**able **S**ummarization approach via Offline Template Generation*. FACTS leverages LLM-based agentic workflows to automatically generate *offline templates*, where each offline template consists of schema-aware SQL queries and a Jinja2 template derived from a given user query and table schema. Once generated, these offline templates can be efficiently reused across new tables, making summarization both scalable and cost-effective. Our framework achieves three advantages: fast summarization through lightweight SQL execution and reusable offline templates which avoid repeated LLM inference, accurate summaries by grounding outputs in precise query results rather than free-form text generation, and privacy-compliant deployment since only table schema is exposed to the LLM. We benchmark FACTS on QTSumm and QFMTS, demonstrating promising improvements over selected baselines.

1 Introduction

Query-focused table summarization generates concise textual summaries of tabular data conditioned on a specific query, enabling users to extract insights from large or multi-table datasets [1]. This task is especially critical in domains such as finance, healthcare, and law. For instance, a financial analyst may wish to track *gross income summaries for the past ten years*, requiring query-specific summaries over multiple yearly tables.

Recent advances have explored diverse strategies, including fine-tuning LLMs for tabular input [2, 3, 4], integrating table summarization into retrieval systems [5, 6], developing code-centric agents such as Data-Copilot [7], and using fact-based reasoning or decomposition methods to improve interpretability and manage token limits [8, 1, 9]. These approaches demonstrate the potential of LLMs but still face key limitations. In practice, deployment is hindered by three challenges: (1) privacy regulations, e.g., HIPAA, GDPR, that restrict exposing sensitive records to external LLMs, (2) prohibitive cost and latency of real-time inference, particularly for multi-call pipelines, and (3) the non-scalability of manual template design. Recent works [10] propose to use an LLM serves as a schema-aware query planner to generate SQL queries over structured data such as financial tables,

*Work done while doing an internship at RBC Borealis. Corresponding to ye.yuan3@mail.mcgill.ca.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: GenAI in Finance.

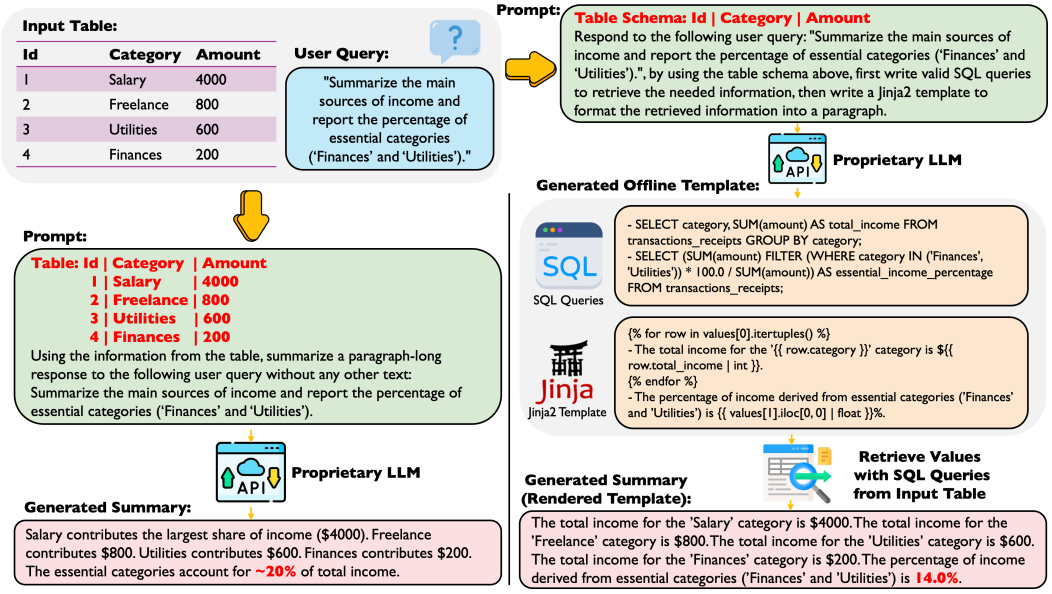

Figure 1: Comparison between Direct LLM Summarization (left) and our proposed FACTS framework (right). Direct LLM summarization prompts the model with the full table and query, which may lead to hallucinated percentages (as shown above), exposes all table records to external LLMs, and requires regeneration for each new table even under the same schema and user query. In contrast, FACTS produces a reusable offline template of SQL queries and a Jinja2 template, which directly retrieves values and renders accurate, privacy-compliant summaries.

which extracts aggregate statistics via these queries and performs reasoning at inference time on top of these statistics rather than raw data, offering a privacy-preserving but still online approach.

To address these issues, we propose **FACTS**, *a **F**ast, **A**ccurate, and Privacy-**C**ompliant **T**able **S**ummarization Framework via Offline Template Generation*. FACTS employs an agentic workflow where LLMs iteratively generate schema-aware SQL queries and Jinja2 templates, validated by an ensemble of models termed as LLM Council. The resulting composite artifact of SQL queries and a Jinja2 template, termed as offline template, is reusable across tables with the same schema. Returning to the finance example, FACTS enables a single template to summarize the gross income for each of the ten years, using the corresponding tables, whereas Direct LLM Summarization requires ten separate prompts, since as part of the prompt, the table would be different for each year, incurring higher costs, token-limit issues, and privacy risks. We illustrate this comparison in Figure 1.

We evaluate FACTS on two benchmarks: QTSumm [1], a large-scale dataset of query-focused table summarization, and QFMTS [11], a multi-table benchmark requiring reasoning across heterogeneous schemas. Experimental results demonstrate that FACTS achieves promising performance compared to the baselines. In summary, our contributions are threefold: (1) introducing offline template generation for query-focused summarization, (2) designing FACTS with schema-aware SQL and Jinja2 templates, and (3) empirically demonstrating promising improvements over selected baselines.

2 Methodology

To avoid ambiguity, we clarify the terms used throughout this section. A `user query` denotes the natural language input provided by the user, which specifies an information need over one or more tables and can potentially include a large context with many details. An `SQL query` refers to a piece of executable SQL code generated by the agent, which retrieves information needed to answer the user query. A `Jinja2 template` is the rendering program that verbalizes SQL results using natural languages. An `offline template`, as used in the paper title and throughout, is the composite artifact that bundles (1) one or more SQL queries and (2) a Jinja2 template; see Appendix A.1 for an example. Unless otherwise specified, the term `schema` refers to the structural metadata of the table, e.g., column names, data types, rather than raw values. Finally, a `summary` is the natural language output presented to the user after executing SQL queries and rendering the Jinja2 template.

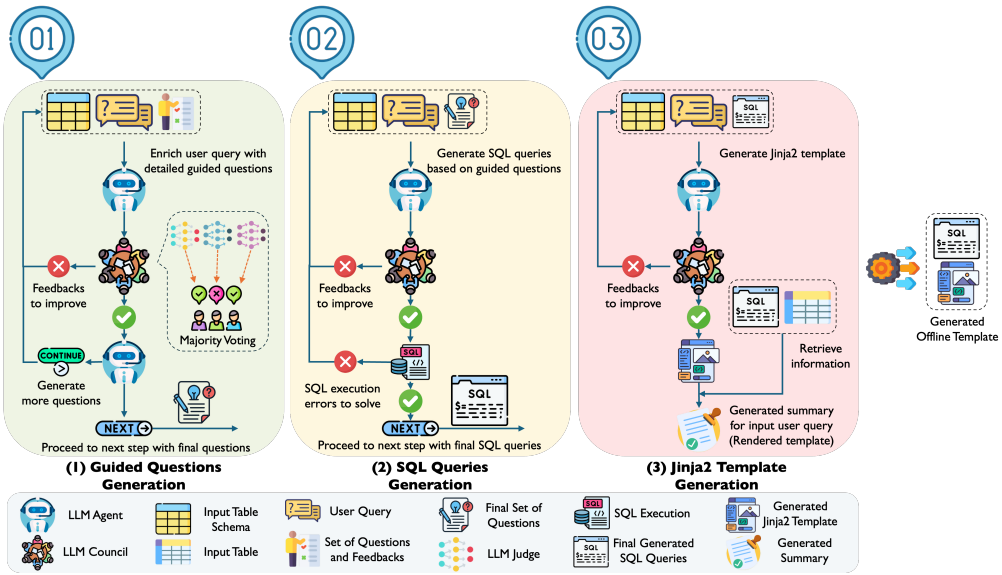

Figure 2: The FACTS framework for query-focused table summarization via Offline Template Generation. (1) Guided Questions Generation: the agent enriches the user query with schema-aware clarifications, validated by the LLM Council which is an ensemble of different LLMs. (2) SQL Queries Generation: the agent synthesizes and iteratively improves SQL queries using execution feedback and LLM Council validation. (3) Jinja2 Template Generation: a Jinja2 template verbalizes SQL outputs into natural language, with LLM Council checks ensuring alignment. The final output is a reusable *offline template* that combines SQL queries with a Jinja2 template.

LLM Council. The LLM Council is an ensemble of different large language models that collaboratively evaluate intermediate outputs at each stage of the framework. For every proposed question, SQL query, or Jinja2 template, each model in the Council independently returns a structured judgment, and a majority-voting scheme determines whether the candidate is accepted or rejected. In addition, brief feedback is aggregated into a consensus explanation, which guides iterative improvements when necessary. This mechanism reduces reliance on a single model, mitigates individual model errors, and provides lightweight quality assurance.

FACTS Framework. The FACTS framework is composed of three interconnected components, which are illustrated in Figure 2. **(1)** Guided Questions Generation: Given the user query and table schema, the agent generates clarifying questions that specify which columns, relationships, and operations are relevant. These questions are validated by the LLM Council, and only accepted questions are retained to refine the original user query into a set of schema-aware specifications. **(2)** SQL Queries Generation: Using the refined specifications, the agent proposes one or more SQL queries against tables following the target schema. Execution feedback, e.g., errors, empty results, and Council judgments are used to iteratively refine these queries until they are both executable and aligned with the user's intent. **(3)** Jinja2 Template Generation: Once the SQL queries are validated, the agent produces a Jinja2 template to render the query results into a natural language summary. The Council checks for alignment between SQL outputs and Jinja2 template references. If inconsistencies are detected, the Jinja2 template can be refined. The validated combination of SQL queries and Jinja2 template constitutes the final *offline template*, which can be reused across new tables with the same schema, enabling efficient, accurate, and privacy-compliant summarization.

Together, these three components ensure that FACTS achieves its three core objectives: fast summarization through reusable offline templates and lightweight SQL execution, accurate outputs grounded in query results rather than free-form LLM generations, and privacy-compliant deployment since only schemas, not raw table values, are ever exposed to the LLMs. Algorithm 1 in Appendix A.2 demonstrates the entire process of our FACTS framework.

3 Experimental Results

In this section, we present the experimental setup and results for evaluating our proposed **FACTS** framework. We first describe the datasets, baselines, and evaluation metrics used, followed by a

Table 1: Results on QTSumm (top) and QFMTS (bottom).

Method	BLEU	ROUGE-L	METEOR	BERTScore	Length / SQL Pass Rate
Direct Summarization	13.75	37.43	43.00	90.06	81.74 / –
Single-Call	14.01	37.90	40.60	89.49	67.83 / 77%
FACTS	**19.60**	**40.70**	**48.11**	**92.34**	82.45 / **100%**
Direct Summarization	14.28	38.45	43.70	90.16	81.87 / –
Single-Call	20.51	47.63	50.79	91.38	57.26 / 93%
FACTS	**25.65**	**50.53**	**52.30**	**94.75**	49.39 / **100%**

discussion of the results and analysis. Additional implementation details, including prompt design and model configurations, are provided in Appendix A.3.

3.1 Dataset and Evaluation

Datasets. We evaluate FACTS on two datasets. The first is QTSumm [1], a large-scale dataset of query-focused table summarization. The second is QFMTS [11], which contains multi-table inputs requiring reasoning across heterogeneous schemas. We randomly sample 500 examples from each dataset to form our evaluation set.

Evaluation. We compare FACTS against two baselines. The first is *Direct LLM Summarization* (direct summarization), where the full table content and user query are passed directly to an LLM to generate a summary in free text. The second is *Single-Call Template Generation* (single-call), where the LLM is prompted once to generate the entire offline template, SQL queries + Jinja2 template, in a single step, without iterative refinement. These baselines allow us to examine the effectiveness of FACTS in comparison to both naive direct prompting and non-agentic template generation. We report standard text generation metrics including BLEU, ROUGE-L, METEOR, and BERTScore, measuring the fidelity of generated summaries against references. To further analyze output quality, we report the average summary length as an indicator of verbosity and the SQL pass rate, defined as the proportion of generated SQL queries that execute successfully without error. The latter is particularly relevant for comparing FACTS with Single-Call Template Generation, as both approaches rely on SQL query generation for grounding the summaries.

3.2 Results and Analysis

As shown in Table 1, FACTS consistently outperforms other methods on both QTSumm and QFMTS. These gains indicate that grounding summaries in executed SQL and iteratively validating intermediate artifacts is more effective than free-form generation or one-shot template synthesis.

Robustness is another differentiator. Single-Call Template Generation attains only 77% on QTSumm and 93% on QFMTS SQL pass rates, reflecting frequent invalid generated SQL queries. In contrast, FACTS achieves a 100% pass rate on both datasets, attributable to its iterative refinement loop with LLM Council feedback and SQL execution checks. Practically, this leads to deterministic, reproducible summaries and reduces failure cases such as hallucinated aggregates or schema mismatches observed in Direct Summarization.

Overall, the results show that FACTS delivers more accurate and reliable query-focused summaries than both direct prompting and single-shot template generation, while maintaining the privacy benefits of schema-only LLM interaction during offline template generation.

4 Conclusion

We present FACTS, a fast, accurate, and privacy-compliant table summarization approach via offline template generation. By leveraging an agentic workflow, FACTS transforms table summarization from costly online inference into reusable offline templates. Experiments on QTSumm and QFMTS demonstrate that FACTS yields more accurate, reliable, and privacy-preserving summaries than direct summarization and single-call template generation. This work highlights the promise of offline template generation as a practical approach for deploying LLMs in sensitive, data-intensive domains such as finance and healthcare.

References

[1] Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen, Xiangru Tang, Yumo Xu, et al. QTSumm: Query-focused summarization over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023.

[2] Lanrui Wang, Mingyu Zheng, Hongyin Tang, Zheng Lin, Yanan Cao, Jingang Wang, Xunliang Cai, and Weiping Wang. Needleinatable: Exploring long-context capability of large language models towards long-structured tables. *ArXiv preprint*, abs/2504.06560, 2025.

[3] Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, et al. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *ArXiv preprint*, abs/2307.08674, 2023.

[4] Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2022.

[5] Haoxiang Zhang, Yurong Liu, Aécio Santos, Juliana Freire, et al. Autoddg: Automated dataset description generation using large language models. *ArXiv preprint*, abs/2502.01050, 2025.

[6] Jiaru Zou, Dongqi Fu, Sirui Chen, Xinrui He, Zihao Li, Yada Zhu, Jiawei Han, and Jingrui He. Gtr: Graph-table-rag for cross-table question answering. *ArXiv preprint*, abs/2504.01346, 2025.

[7] Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Data-copilot: Bridging billions of data and humans with autonomous workflow. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

[8] Kwangwook Seo, Jinyoung Yeo, and Dongha Lee. Unveiling implicit table knowledge with question-then-pinpoint reasoner for insightful table summarization. *ArXiv preprint*, abs/2406.12269, 2024.

[9] Yasir Khan, Xinlei Wu, Sangpil Youm, Justin Ho, Aryaan Shaikh, Jairo Garciga, Rohan Sharma, and Bonnie J Dorr. Detqus: Decomposition-enhanced transformers for query-focused summarization. *ArXiv preprint*, abs/2503.05935, 2025.

[10] Josefa Lia Stoisser, Marc Boubnovski Martell, Kaspar Märtens, Lawrence Phillips, Stephen Michael Town, Rory Donovan-Maiye, and Julien Fauqueur. Query, don't train: Privacy-preserving tabular prediction from ehr data via sql queries. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025.

[11] Weijia Zhang, Vaishali Pal, Jia-Hong Huang, Evangelos Kanoulas, and Maarten de Rijke. Qfmts: Generating query-focused summaries over multi-table inputs, 2024.

[12] OpenAI. Gpt-5 system card, 2025.

[13] Anthropic. System card: Claude opus 4 & claude sonnet 4, 2025.

[14] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, et al. Deepseek-v3 technical report, 2025.

[15] Mark Raasveldt and Hannes Mühleisen. Duckdb: an embeddable analytical database. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference*. Acm, 2019.

Example 1: An offline template generated by FACTS on the QFMTS dataset. The SQL query retrieves the top three accounts by savings balance, and the Jinja2 template renders the results into natural language.

```
SQL Queries:
  - SELECT a."name", s."balance"
    FROM "ACCOUNTS" a
    JOIN "SAVINGS" s
      ON CAST(a."custid" AS DOUBLE) = s."custid"
    ORDER BY s."balance" DESC, a."name" ASC
    LIMIT 3;
Jinja2 Template:
    {% if values and values|length > 0 %}
        The three accounts with the highest savings balances are:
        {% for row in values %}
            - {{ row["name"] }} with a savings balance of {{ row["balance"] }}.
        {% endfor %}
        Overall, these represent the top savers by balance in the dataset.
    {% else %}
        No results were found for the requested top savings accounts.
    {% endif %}
```

A Appendix

A.1 Example of Offline Template

An offline template in FACTS is a composite artifact consisting of (1) one or more SQL queries and (2) a Jinja2 template that verbalizes the SQL outputs into natural language. Once generated, the same offline template can be applied to any table with the same schema, making it reusable across datasets that share structural metadata but differ in values, e.g., multiple years of financial records. This reusability allows FACTS to avoid repeated LLM inference, while ensuring summaries remain accurate and privacy-compliant. Example 1 shows a real template generated by FACTS on the QFMTS benchmark for identifying the top three accounts with the highest savings balances. The SQL query retrieves account names and balances from the `ACCOUNTS` and `SAVINGS` tables, while the Jinja2 template renders this output into a coherent summary. This demonstrates how FACTS produces executable and reusable artifacts that generalize across tables with the same schema.

A.2 Pseudo Code of FACTS

Algorithm 1 summarizes the FACTS workflow. The process begins with Guided Questions Generation, where the agent proposes schema-aware clarifying questions based on the user query q and table schema $\mathcal{S}$. Each question is vetted by the LLM Council, and accepted ones are accumulated in $\mathcal{U}$ to progressively refine the query intent. Next, in SQL Queries Generation, candidate SQL queries are synthesized using $(q, \mathcal{U}, \mathcal{S})$, executed locally against the schema, and validated both by execution feedback and the LLM Council. Invalid queries are iteratively revised until a correct and executable query $\mathcal{Q}$ is obtained. Finally, in Jinja2 Template Generation and Alignment, the agent generates a Jinja2 template $\mathcal{J}$ to render SQL results into natural language. The LLM Council checks alignment between template references and SQL outputs; if misalignments are detected, the template is refined until valid. The resulting offline template $\mathcal{T} = (\mathcal{Q}, \mathcal{J})$ can then be reused across any table with the same schema, enabling fast, accurate, and privacy-compliant summarization without exposing raw table values to the LLMs.

A.3 Implementation Details

We provide key implementation details to facilitate reproducibility. Our proposed FACTS method employs an agentic workflow with GPT-5 [12] as the main LLM agent. At each stage of template generation, intermediate artifacts, including guiding questions, SQL queries, and Jinja2 templates, are validated by an ensemble of models, which we term the LLM Council. The Council consists of GPT-5 [12], Claude-4 Sonnet [13], and DeepSeek v3 [14]. Each model independently provides

Algorithm 1 FACTS Framework

Input: user query q, table schema $\mathcal{S}$, LLMs Council $\mathcal{C}$, max number of guiding questions K_q, patience $P_q, P_{\text{sql}}, P_{\text{tpl}}$
Output: offline template $\mathcal{T} = (\mathcal{Q}, \mathcal{J})$ where $\mathcal{Q}$ is a set of SQL queries and $\mathcal{J}$ is a Jinja2 template

```
/* Component 1: Guided Questions Generation */
U ← ∅                                                   ▷ accepted guiding questions
for k = 1 to K_q do                                     ▷ how many guiding questions we can generate
    u ← GenQuestion(q, S, U)
    (vote, fb) ← CouncilJudge(C, u)
    if vote = YES then
        U ← U ∪ {u}                                     ▷ added to the accepted set of guiding questions
    else
        t ← 0
        while vote ≠ YES and t < P_q do                 ▷ refine until Council satisfied or reach patience
            u ← ReviseQuestion(u, fb, q, S, U)
            (vote, fb) ← CouncilJudge(C, u)
            t ← t + 1
        if vote = YES then
            U ← U ∪ {u}
    if Sufficient(U) then break                         ▷ final set of guiding questions
/* Component 2: SQL Queries Generation */
Q ← ∅; vote ← false
t ← 0
while not vote and t < P_sql do
    Q̃ ← GenSQL(q, U, S)
    exec ← ExecuteSQL(Q̃, S)
    (vote, fb) ← CouncilJudge(C, (Q̃, exec))
    if vote = YES and Valid(exec) then
        Q ← Q̃; vote ← true
    else
        Q̃ ← ReviseSQL(Q̃, exec, fb)                      ▷ handle errors, empties, shape mismatches
        t ← t + 1
/* Component 3: Jinja2 Template Generation and Alignment */
vote ← false; t ← 0
while not vote and t < P_tpl do
    J ← GenJinja2(q, Q, S)
    (vote, fb) ← CouncilJudge(C, (J, Q, S))
    vote ← (vote = YES) and Aligned(J, Q, S)            ▷ fields match SQL outputs
    if not vote then
        (Q, J) ← Refine(Q, J, fb)                       ▷ fix unknown fields, shapes
        t ← t + 1
return T = (Q, J)                                       ▷ reusable offline template
```

structured feedback, and majority voting determines whether an artifact is accepted or rejected. We allow a fixed upper bound on the number of guiding questions generated, which is 10 in our implementation, and set the patience for revising questions, SQL queries, and Jinja2 templates to 3 iterations. For SQL execution within Python, we use DuckDB [15], which provides efficient in-memory querying and seamless integration with pandas DataFrames. For fair comparisons, we implement baseline methods using GPT-5 [12] as the backbone.