Excel Chatbot: A Security-First Multi-Agent Framework for Querying Complex Financial Excel Sheets

Phan Trong Dai

June 22, 2025

Abstract

Financial and business data stored in Microsoft Excel spreadsheets represent a critical yet sensitive corporate asset. While Large Language Models (LLMs) offer unprecedented capabilities for natural language querying, their application in enterprise settings is severely limited by security and privacy concerns, as mainstream solutions require uploading raw, confidential data to third-party services. This paper introduces a novel, security-first framework for a dynamic Excel chatbot that circumvents this critical issue. Our core innovation lies in a multi-agent system that interacts with LLMs using only the file's structural schema (i.e., header information and layout), while the sensitive data values remain securely on-premise. The system intelligently parses complex table structures, understands user queries, and translates them into precise data lookups that are executed locally. This approach enables employees to conversationally query intricate financial reports without exposing any proprietary information, demonstrating a viable and secure pathway for deploying advanced AI analytics within the modern enterprise.

1 Introduction

Microsoft Excel stands as an indispensable tool in the modern enterprise, serving as the de facto platform for a vast array of critical business functions, from financial reporting and sales tracking to project management. Its unparalleled flexibility and intuitive, grid-based interface empower users to organize and present data in highly customized, human-readable formats. However, this same flexibility is a double-edged sword. While ideal for manual analysis and presentation, the resulting spreadsheets often contain complex, non-relational structures that create a significant barrier to automated data processing and querying. Consequently, a wealth of valuable business intelligence remains "locked" within these documents, accessible only through tedious and error-prone manual extraction.

The challenges in automating the analysis of real-world spreadsheets are multifaceted and extend far beyond the capabilities of traditional data processing tools. We identify four primary obstacles that a viable solution must overcome:

1. **Structural Complexity:** Corporate spreadsheets rarely conform to the simple, "flattened" structure of a database table, as shown in Figure 1a. They frequently employ sophisticated layouts that are visually intuitive but algorithmically ambiguous. These include headers with merged cells (Figure 1b) and complex *matrix tables* with hierarchical data relationships (Figure 1c), which are common in real-world financial reports. The full range of complexity is illustrated in Figure 1.

2. **Data Security and Privacy:** Financial and operational data are among an organization's most sensitive assets. Many contemporary AI-driven solutions, particularly those leveraging powerful cloud-based Large Language Models (LLMs), require uploading entire files to third-party servers. This

	A	B	C
1	Họ và tên	Lớp	Trường
2	ABC	1	A
3	XYZ	2	B
4	DEF	3	A
5	IJK	4	A
6	MNP	5	B
7	UVX	6	A

(a) Flattened Table

Purchase Orders

OrderDate	Region	Name	Item	Units	UnitCost	Total
1/6/2019	East	Jones Andrews	Pencil	95	1.99	189.05
1/23/2019	Central	Kivell Jardine	Binder	50	19.99	999.5
2/9/2019	Central	Jardine Jardine	Pencil	36	4.99	179.64
2/26/2019		Gill Andrews	Pen	27	19.99	539.73
4/1/2019	East	Jones Sorvino	Binder	60	4.99	299.4
4/18/2019	Central	Andrews Gill	Pencil	75	1.99	149.25
5/5/2019	Central	Jardine Sorvino	Pencil	90	4.99	449.1
5/22/2019	West	Thompson Kivell	Pencil	32	1.99	63.68
6/8/2019	East	Jones Morgan	Binder	60	8.99	539.4
6/25/2019	Central	Morgan Jones	Pencil	90	4.99	449.1
7/12/2019	East	Howard Kivell	Binder	29	1.99	57.71
7/29/2019	East	Parent Gill	Binder	81	19.99	1,619.19
8/15/2019	East	Jones Gill	Pencil	35	4.99	174.65

(b) Merged Header

	A	B	C	D	E	F	G	H
1	Cà phê	Có đá	Giá 2023			Giá 2024		
2	Cà phê thường	Không	100k	100k	100k	100k	100k	100k
3	Cà phê Đen	Không	200k	200k	200k	200k	200k	200k
4	Cà phê Đen	Có đá	300k	300k	300k	300k	300k	300k
5	Cà phê Sữa	Không	400k	400k	400k	400k	400k	400k
6	Cà phê Sữa	Có đá	500k	500k	500k	500k	500k	500k
7	Cà phê muối	Không	600k	600k	600k	600k	600k	600k

(c) Matrix Table

Figure 1: Examples of varying structural complexity in Excel spreadsheets. (a) Simple flattened tables resemble database formats. (b) Merged headers introduce structural ambiguity. (c) Complex matrix tables with hierarchical data are common in real-world financial reports.

practice introduces unacceptable security risks and violates data privacy policies in most enterprise environments.

3. **User Query Ambiguity:** Business users interact using natural language, which is inherently imprecise. Queries often contain domain-specific jargon, abbreviations (e.g., "CP" for "Chi phí" or "Cost Price"), and synonyms. A robust system must be able to interpret this nuanced language to deliver accurate results.

4. **Multi-File Context:** Strategic analysis often requires synthesizing information scattered across multiple files, such as monthly reports or annual financial statements from different years. A solution must therefore be capable of understanding queries that span several documents and intelligently routing them to the appropriate data sources.

The primary data structure targeted by our framework is the **matrix table**, a format characterized by its use of complex, multi-level hierarchies and extensive use of **merged cells** (Figure 2a). Beyond this core structure, our system is designed to confront several related challenges common in real-world financial documents, as illustrated in Figure 2. These include the presence of **undefined fields**, where feature rows are left intentionally blank for formatting (Figure 2b), and headers that are simultaneously **hierarchical and merged** (Figure 2d). Finally, the framework must handle **flexible fields**, where the structure of feature rows is not uniform across the entire table—for instance, some primary categories may have sub-categories while others do not, breaking the assumption of a consistent schema (Figure 2c).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1				Năm 2022												
2	TT	Đối tác	Nguồn	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tháng	Tổng
3	1	ALTEK														
4	1.1		Tự SX	100	150	200	250	300	350	400	450	500	550	600	650	700
5	1.2		OS	200	250	300	350	400	450	500	550	600	650	700	750	800
6	1.3		LK	300	350	400	450	500	550	600	650	700	750	800	850	900
7	1.4		Dịch vụ	400	450	500	550	600	650	700	750	800	850	900	950	1000
8	1.5		Phí QL	500	550	600	650	700	750	800	850	900	950	1000	1050	1100
9	1.6		Thương ma	600	650	700	750	800	850	900	950	1000	1050	1100	1150	1200
10	1.7		Khác	700	750	800	850	900	950	1000	1050	1100	1150	1200	1250	1300
11	2	ANT Group														
12	2.1		Tự SX	900	950	1000	1050	1100	1150	1200	1250	1300	1350	1400	1450	1500
13	2.2		OS	1000	1050	1100	1150	1200	1250	1300	1350	1400	1450	1500	1550	1600
14	2.3		LK	1100	1150	1200	1250	1300	1350	1400	1450	1500	1550	1600	1650	1700
15	2.4		Dịch vụ	1200	1250	1300	1350	1400	1450	1500	1550	1600	1650	1700	1750	1800
16	2.5		Phí QL	1300	1350	1400	1450	1500	1550	1600	1650	1700	1750	1800	1850	1900
17	2.6		Thương ma	1400	1450	1500	1550	1600	1650	1700	1750	1800	1850	1900	1950	2000
18	2.7		Khác	1500	1550	1600	1650	1700	1750	1800	1850	1900	1950	2000	2050	2100
19	3	B&C														
20	3.1		Tự SX	1700	1750	1800	1850	1900	1950	2000	2050	2100	2150	2200	2250	2300
21	3.2		OS	1800	1850	1900	1950	2000	2050	2100	2150	2200	2250	2300	2350	2400
22	3.3		LK	1900	1950	2000	2050	2100	2150	2200	2250	2300	2350	2400	2450	2500
23	3.4		Dịch vụ	2000	2050	2100	2150	2200	2250	2300	2350	2400	2450	2500	2550	2600
24	3.5		Phí QL	2100	2150	2200	2250	2300	2350	2400	2450	2500	2550	2600	2650	2700
25	3.6		Thương ma	2200	2250	2300	2350	2400	2450	2500	2550	2600	2650	2700	2750	2800
26	3.7		Khác	2300	2350	2400	2450	2500	2550	2600	2650	2700	2750	2800	2850	2900

(a) A primary example of a matrix table with hierarchical rows and columns.

Chỉ tiêu	Số liệu	Phân loại	Loại hình
DOANH THU			
		Kế hoạch (Tập đoàn)	
			Sản xuất
			Dịch vụ
			Thương mạ
			Khác
		Thực hiện	
			Sản xuất
			Dịch vụ
			Thương mạ
			Khác
		% Hoàn thành	
			Sản xuất
			Dịch vụ
			Thương mạ
			Khác
Lợi nhuận gộp			
	Tổng lợi nhuận gộp		
		Kế hoạch (Tập đoàn)	
			Sản xuất
			Dịch vụ
			Thương mạ
			Khác
		Thực hiện	
			Sản xuất
			Dịch vụ
			Thương mạ
			Khác
		% Hoàn thành	
			Sản xuất
			Dịch vụ
			Thương mạ

(b) Undefined Fields

TT	Chỉ tiêu	Số liệu	Phân loại	Loại hình
			% Hoàn thành	
12	Nợ phải trả/ Vốn chủ sở hữu			
12.1			Kế hoạch (Tập đoàn)	
12.2			Thực hiện	
12.3			% Hoàn thành	
13	Lỗ lũy kế			
14	Nộp ngân sách Nhà nước			
14.1		Tổng số phải nộp		
14.1.1			Kế hoạch (Tập đoàn)	
14.1.2			Thực hiện	
14.1.3			% Hoàn thành	
14.2		Số nộp tại Thanh Hóa		
15	Tổng tài sản			
16	Vốn chủ sở hữu			
17	Vốn vay			

(c) Flexible Fields

F	G	H	I	J	K	L	M	N	O
								Năm 2024	
6 Tháng đầu năm									
Quý I				Quý II				Cộng	
Tháng 01	Tháng 02	Tháng 03	Cộng	Tháng 04	Tháng 05	Tháng 06	Cộng		Thán

(d) Hierarchical & Merged

Figure 2: Illustrations of structural challenges in real-world spreadsheets. (a) shows the primary matrix table structure. (b), (c), and (d) highlight specific issues of undefined, complexly merged, and flexible fields, respectively.

Existing paradigms like Natural Language to SQL (NL2SQL) are ill-suited for this problem, as they presuppose a clean, pre-existing structured database, failing at the initial step of parsing the complex Excel format. While recent LLM-based tools show promise, they often falter on complex layouts and, more critically, do not address the fundamental security concerns.

In this paper, we introduce **Excel Chatbox**, a novel, security-first multi-agent framework designed to bridge this gap. Our system enables dynamic, conversational querying of complex Excel files by intelligently

interpreting their structure and the user's intent. The core contribution of our work is a methodology that leverages the semantic power of LLMs on the file's *schema* alone, without ever exposing the sensitive underlying data to an external service. By doing so, we provide a secure, intuitive, and scalable solution that unlocks the valuable information within enterprise spreadsheets for non-technical users.

2 Related Work

The challenge of making complex spreadsheets queryable via natural language sits at the intersection of data engineering, natural language processing, and system architecture. We situate our work by examining the limitations of four dominant paradigms in this domain.

2.1 Rule-Based and Library-Driven Parsing

The most direct method for programmatic spreadsheet interaction involves libraries like Pandas [4]. This approach is highly effective for files that adhere to a simple, "flattened" tabular format. However, it is fundamentally brittle when faced with the structural complexity of real-world financial reports (as illustrated in Figure 2). Rule-based systems lack the semantic awareness to interpret the context of hierarchical headers or merged cells and require bespoke, template-specific code for each new report layout, a practice that is both error-prone and unscalable.

2.2 Natural Language to SQL (NL2SQL)

To address the natural language component, one might look to the mature field of NL2SQL. Systems benchmarked on datasets like Spider [6] demonstrate impressive capabilities in translating user questions into executable SQL queries. The critical flaw in applying this paradigm here, however, is its core assumption: the data must already reside in a clean, structured, and queryable relational database. NL2SQL systems fail to construct the crucial first bridge: parsing the unstructured, matrix-like format of the Excel file into a relational schema.

2.3 End-to-End LLM Approaches

The advent of LLMs has catalyzed a new wave of tools aiming to provide end-to-end conversational analysis of spreadsheets. This category includes research prototypes designed as spreadsheet assistants, such as SheetCopilot [2] and SpreadsheetCoder [3], as well as benchmarks like InstructExcel [1]. Furthermore, advanced reasoning techniques like Chain-of-Table [5] have been proposed to improve an LLM's ability to interpret and manipulate tabular data through intermediate steps. Despite these significant advances, such approaches exhibit critical weaknesses for our target use case:

1. **Fundamental Security Concerns:** This is the most significant barrier to enterprise adoption. Even with sophisticated reasoning, the operational model for these tools requires transmitting the entire file content, including sensitive financial data, to an external API. This practice is fundamentally non-viable for most corporate environments.

2. **Reliability on Complex Structures:** While techniques like Chain-of-Table improve reasoning, they still entrust the LLM with interpreting the raw, complex structure of a matrix table. This can lead to a lack of guaranteed correctness, which is a high-risk strategy when the absolute precision of financial data is paramount. Our approach differs by first extracting a deterministic schema, which is then used to guide a more constrained and reliable query process.

2.4 Template-Based Agent Systems

A more sophisticated architectural pattern involves creating a multi-agent system where a router directs a query to one of several agents, each hard-coded for a specific file template (Figure 3). While modular, this model suffers from a critical scalability flaw, requiring labor-intensive development for every new report format.

In summary, the existing landscape of solutions falls short. Rule-based systems are too brittle, NL2SQL frameworks begin too late, end-to-end LLM approaches are fundamentally insecure and risk unreliability, and template-based agent systems are unscalable. This reveals a clear need for a framework that is simultaneously secure, intelligent, and scalable. Our work is designed to address this multifaceted challenge.

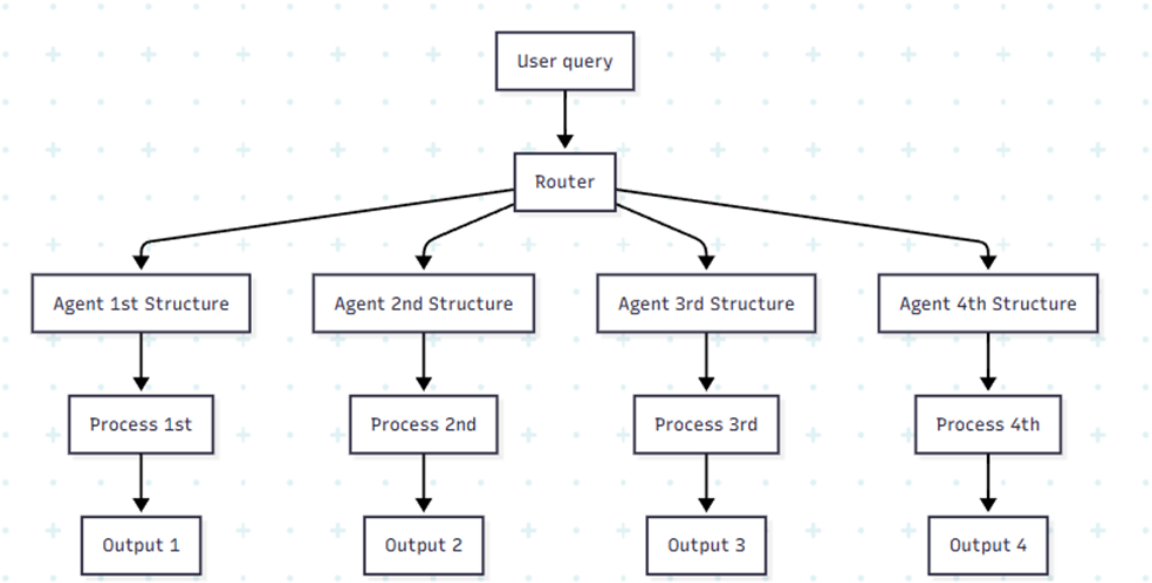

Figure 3: A baseline agent-based architecture where a router assigns queries to agents specialized for a single, pre-defined file structure. This approach is flexible but not scalable.

3 Methodology

3.1 Main Architecture

In stark contrast to the template-based agent systems discussed previously, our framework utilizes a single, unified workflow for all matrix Excel file structures. This approach achieves true code-independency, allowing the system to be extended to new and unseen file layouts without requiring any modifications to the agent's underlying code. This scalability is made possible by a design that leverages both the inherent structural patterns (*inductive bias*) of matrix tables and the powerful semantic understanding of Large Language Models (LLMs). Instead of hard-coding rules for each template, we extract a universal, machine-readable **schema** from the spreadsheet and use this as the context for an LLM to interpret user queries.

The architecture of this single-file execution engine is depicted in Figure 4. It is a pipeline of specialized agents designed to incrementally translate a natural language query into a deterministic data-retrieval command.

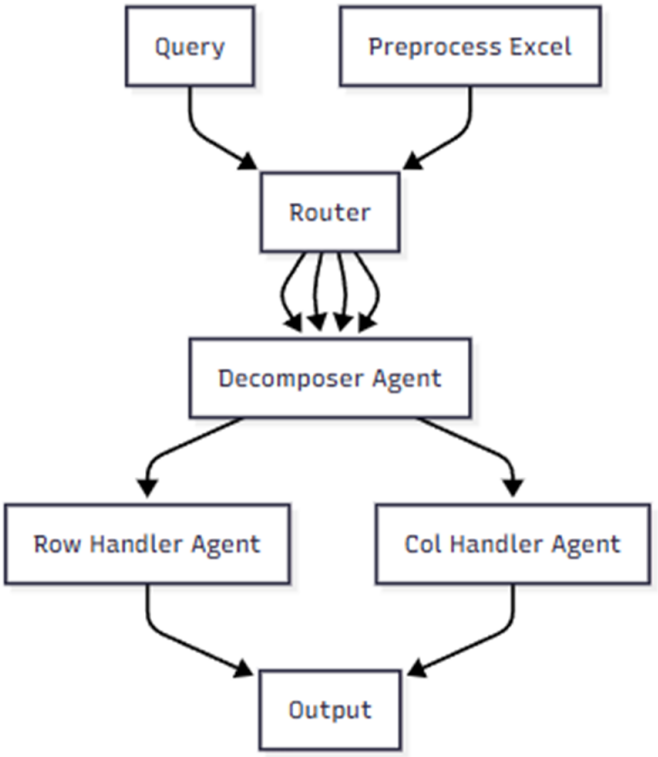

Figure 4: The Main Architecture of the single-file query execution engine, showing the flow from query and preprocessed Excel to the final output through a series of specialized agents.

3.1.1 Schema Extraction and Preprocessing

The fundamental goal of our framework is to programmatically execute the conceptual query ‘matrix[row_identifier][col_identifier]‘. To achieve this, we must first identify the parts of the spreadsheet that define the row and column identifiers. We designate these as **Feature Rows** and **Feature Columns**.

The preprocessing stage is responsible for this critical task. First, a rule-based heuristic is applied to identify the likely number of header rows, a step that is especially important for handling complex merged headers correctly. Following this, the core classification task is delegated to an LLM. As distinguishing between feature rows and feature columns requires semantic understanding of the table’s content and layout, an LLM is uniquely suited for this purpose. Figure 5 illustrates this process, showing both the conceptual division of the table into feature axes and the structured text input - output.

Once the feature axes are identified, we must extract their specific values to build a complete map of the table’s structure. A Depth-First Search (DFS) algorithm is employed to efficiently traverse the header regions and extract the full **Row Hierarchy** and **Column Hierarchy**. As shown in Figure 6, this process generates a structured text representation for both the row (Figure 6a) and column (Figure 6b) dimensions, capturing the nested relationships essential for resolving queries accurately.

3.1.2 Query Decomposition and Handling

With the schema fully extracted, the user’s query is processed by a pipeline of handler agents.

The **Decomposer Agent** is the first node in this pipeline. It takes the user’s query and the extracted

Thành Phố	Phân loại	Giới Tính	
		Nam	Nữ
TPHCM			
	Cao		
	TB		
Hà nội			
	Cao		
Cao bằng			
	TB		
Hà Giang			
	TB		
	Cao		
Vũng Tàu			
	TB		

(a) Conceptual view of Feature Rows and Feature Columns.

```
### Content
Thành Phố,          Phân loại,          Giới tính,Giới tính
Unnamed: 0_level_1,Unnamed: 1_level_1,Nam      ,Nữ

### Name of Feature Rows
Thành Phố,Phân loại,Quận,Phường

### Name of Feature Cols
Giới tính
```

(b) Classification input-output.

Figure 5: The process of classifying spreadsheet headers. (a) illustrates the concept of Feature Rows and Feature Columns that define the table's axes. (b) shows the actual structured text input-output.

```
### Feature Rows
['Cà phê', 'Loại', 'Nhập Khẩu ']

### Row Hierarchy
cà phê: cà phê thường
    loại: loại 1
        nhập khẩu : việt nam, brazil, mỹ
    loại: loại 2
        nhập khẩu : việt nam, mỹ
cà phê: cà phê đen
    loại: loại 2
        nhập khẩu : việt nam
```

(a) Extracted Row Hierarchy.

```
### Feature Column
['thời gian', 'thu nhập']

### Column Hierarchy
level_1: thời gian
    level_2: hè, đông
level_1: thu nhập
    level_2: thấp, trung bình, cao
```

(b) Extracted Column Hierarchy.

Figure 6: The DFS-based extraction process produces structured text representations of the (a) Row Hierarchy and (b) Column Hierarchy. This schema captures the nested relationships essential for query resolution.

Row and Column Hierarchies as input. Its function is to analyze the query and identify which keywords are associated with the row dimension and which are associated with the column dimension, producing a set of 'Row Keywords' and 'Col Keywords' (Figure 7).

Next, the **Row Handler** and **Col Handler** agents work in parallel to resolve the keywords provided by the Decomposer. As shown in Figure 8, each agent receives the keywords for its respective dimension and the corresponding hierarchy. Their task is to map these potentially ambiguous keywords to a final, unambiguous **identifier**—a fully-qualified path that points to a specific set of rows or columns. For example, the Row Handler (Figure 8a) resolves the keywords into a precise structural row identifier. Similarly, the Col Handler (Figure 8b) determines the final identifier for the column based on its keywords. This step is crucial for pinpointing the exact data required for the final retrieval operation.

3.1.3 Security by Design

The entire architecture is built upon a foundational security principle. The LLM agents—Decomposer, Handlers, and Classifiers—**only ever interact with the file's schema**. The actual data values within the spreadsheet cells are never sent to or processed by any external AI service. This strict separation ensures that

```
### Query
cho tôi biết cà phê đen tại việt nam có giá như thế nào vào những tháng hè

### Row Hierarchy
cà phê: cà phê thường
    loại: loại 1
        nhập khẩu : việt nam, brazil, mỹ
    loại: loại 2
        nhập khẩu : việt nam, mỹ
cà phê: cà phê đen
    loại: loại 2
        nhập khẩu : việt nam

### Column Hierarchy
level_1: thời gian
    level_2: hè, đông
level_1: thu nhập
    level_2: thấp, trung bình, cao

### Row Keywords
-   cà phê đen
-   việt nam

### Col Keywords
-   tháng hè
```

Figure 7: The Decomposer Agent maps keywords from the natural language query to the corresponding row or column dimension.

```
### Query
cho tôi biết cà phê đen tại việt nam có giá như thế nào vào những tháng hè

### Row Hierarchy
cà phê: cà phê thường
    loại: loại 1
        nhập khẩu : việt nam, brazil, mỹ
    loại: loại 2
        nhập khẩu : việt nam, mỹ
cà phê: cà phê đen
    loại: loại 2
        nhập khẩu : việt nam

### Row Keywords
['cà phê đen', 'việt nam']

### Row Identifier
cà phê: cà phê đen
    loại: Undefined
        nhập khẩu : việt nam
```

(a) The Row Handler resolves its keywords into a specific Row Identifier.

```
### Query
cho tôi biết cà phê đen tại việt nam có giá như thế nào vào những tháng hè

### Column Hierarchy
level_1: thời gian
    level_2: hè, đông
level_1: thu nhập
    level_2: thấp, trung bình, cao

### Col Keywords
['tháng hè']

### Col Identifier
level_1: thời gian
    level_2: hè
```

(b) The Col Handler resolves its keywords into a specific Col Identifier.

Figure 8: The parallel processes of the Handler agents. Each uses the keywords from the Decomposer to generate a precise, structural identifier for its respective dimension, ensuring accurate data retrieval.

sensitive internal data remains private and secure within the local environment at all times. By leveraging LLMs as powerful reasoning engines that operate exclusively on metadata, our framework solves the security problem that fundamentally disqualifies most other LLM-based approaches for enterprise use.

3.2 Multi-File Architecture for Complex Queries

While the single-file engine is core to our framework, many real-world analytical tasks require synthesizing information from multiple documents. To address this, we designed a multi-file management pipeline that intelligently routes and decomposes complex user queries. The entire process is designed to happen before the single-file execution engine is invoked.

The architecture, depicted in Figure 9, is orchestrated by two key agents: the `Agent Summarization` and the `Agent Routing`.

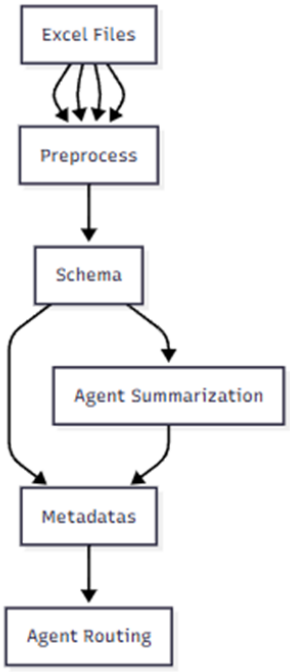

Figure 9: The high-level architecture for multi-file query processing. Each file is preprocessed and summarized, with the resulting metadata being used by a routing agent to decompose the user's query.

3.2.1 Agent Summarization: Creating Semantic Fingerprints

Before any routing can occur, the system must first understand the content of each available file. After the standard schema extraction process is run on each spreadsheet, the resulting schema is passed to a specialized `Agent Summarization`.

This agent's sole function is to generate a high-level, semantic summary—or "fingerprint"—of the file's contents. This is not a summary of the data values, but rather a structured description of the file's purpose and scope. As shown in Figure 10, the output is a set of metadata that includes the overall domain, time coverage, and primary entities and metrics contained within the file. This compact metadata is then fed to the Agent Routing.

3.2.2 Agent Routing: Intelligent Query Decomposition

The `Agent Routing` is the central dispatch node of the multi-file architecture. It receives two key inputs: the user's full, potentially complex query, and the collected metadata summaries for all available files.

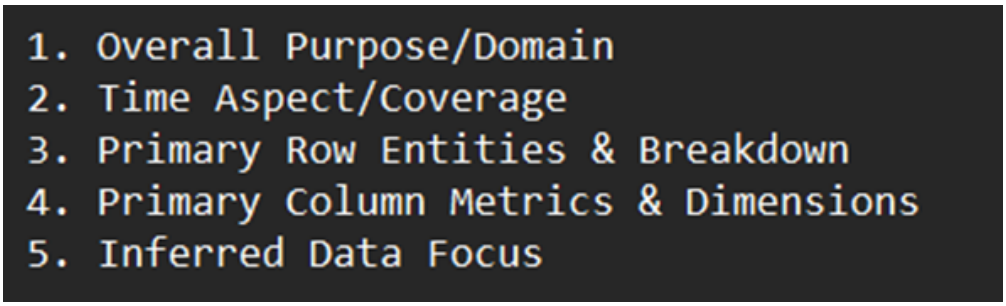

Figure 10: The structured metadata output produced by the Agent Summarization. This "fingerprint" allows the routing agent to quickly understand the contents of each file without needing to analyze its full schema.

The intelligence of this agent lies in its ability to perform **query decomposition**. Instead of simply matching a query to a single file, it breaks down a complex request that spans multiple semantic domains into several simpler, targeted sub-queries. The output is a list mapping each sub-query to the specific file it should be executed against. Figure 11 demonstrates this capability.

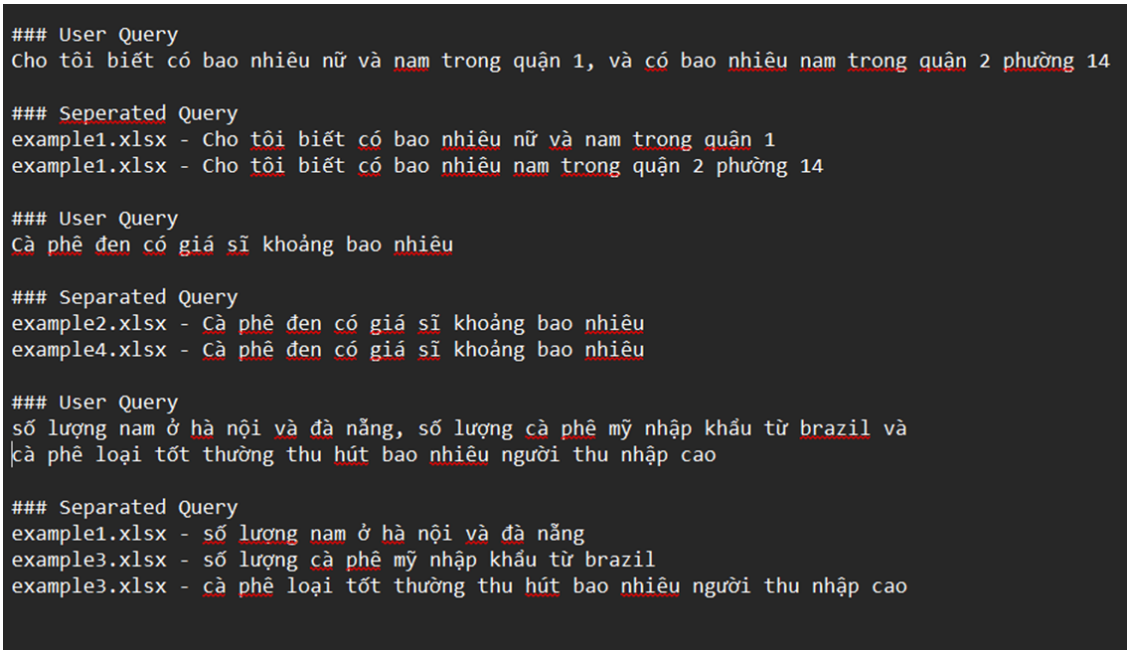

Figure 11: Examples of the Agent Routing performing query decomposition. It intelligently splits complex user queries into simpler, targeted sub-queries assigned to the most relevant files based on their metadata summaries.

3.3 Query Enrichment via Alias Management

User queries are frequently posed using informal language, including domain-specific abbreviations, acronyms, and aliases that are not explicitly present in the formal schema of a spreadsheet. To bridge this semantic gap, our framework includes a query pre-processing step for alias management, which occurs immediately before the query is passed to the multi-file routing agent.

This process is handled by an `Enrich Query` agent, which takes the user's raw query and a config-

urable alias file as input. It is crucial to note that this agent does not perform a simple search-and-replace operation. A naive replacement strategy could degrade performance or lead to errors; for example, if an alias file maps "Doanh Thu" to "DT", replacing "Doanh Thu" in the query would prevent the system from matching with a file that correctly uses the full "Doanh Thu" column header.

Instead, our agent performs query **enrichment**. It identifies terms in the query that have known aliases and appends the additional information, typically in parentheses. For instance, as shown in Figure 12, the query term ‘Doanh Thu‘ is transformed into ‘Doanh Thu(DT)‘, and ‘SkyIQ‘ becomes ‘SkyIQ(SKYIQ PTE.LTD)‘.

```
### Initial Query
CP cho sản xuất là bao nhiêu năm 2022?
### Enriched Query
CP(Chi phí) cho sản xuất là bao nhiêu năm 2022?


### Initial Query
So sánh Doanh Thu của SkyIQ và Viettel Timor Leste.
### Enriched Query
So sánh Doanh Thu(DT) của SkyIQ(SKYIQ PTE.LTD) và Viettel Timor Leste(Telemor-VTL).
```

Figure 12: The query enrichment process. The initial user query is transformed into an enriched query where known aliases and full names are provided, giving downstream agents more context to work with.

This enrichment strategy is fundamentally more robust than replacement. It preserves the user’s original intent while providing the downstream agents (like the `Agent Routing` and `Decomposer Agent`) with a richer set of keywords. These agents can then intelligently match against either the formal name or its alias, depending on what is present in the specific file’s schema or metadata summary, significantly increasing the flexibility and accuracy of the entire framework.

3.4 The Final Unified Framework

Having detailed the core components of our system—from the schema-based single-file execution engine to the multi-file routing and query enrichment pipelines—we now present the final, unified architecture in Figure 13.

This diagram illustrates the complete, end-to-end data flow of the Excel Chatbox. It shows how a user’s initial query is first enriched with aliases before being intelligently routed to the appropriate file(s) based on pre-generated semantic summaries. The resulting sub-query is then processed by the secure, schema-driven execution engine, which decomposes the request and retrieves the final data without ever exposing sensitive values to an external service.

This integrated, modular design is the key to our system’s ability to handle complex queries in a secure, scalable, and reliable manner.

4 Discussion and Future Work

The framework presented in this paper offers a robust and secure solution to the long-standing problem of querying complex, real-world spreadsheets. By architecting a system that uses LLMs to reason over a file’s metadata and schema—rather than its raw, sensitive data—we successfully circumvent the fundamental security and privacy concerns that render most other AI-based approaches non-viable for enterprise deployment. Our multi-agent pipeline, which decomposes the problem into discrete steps of schema extraction,

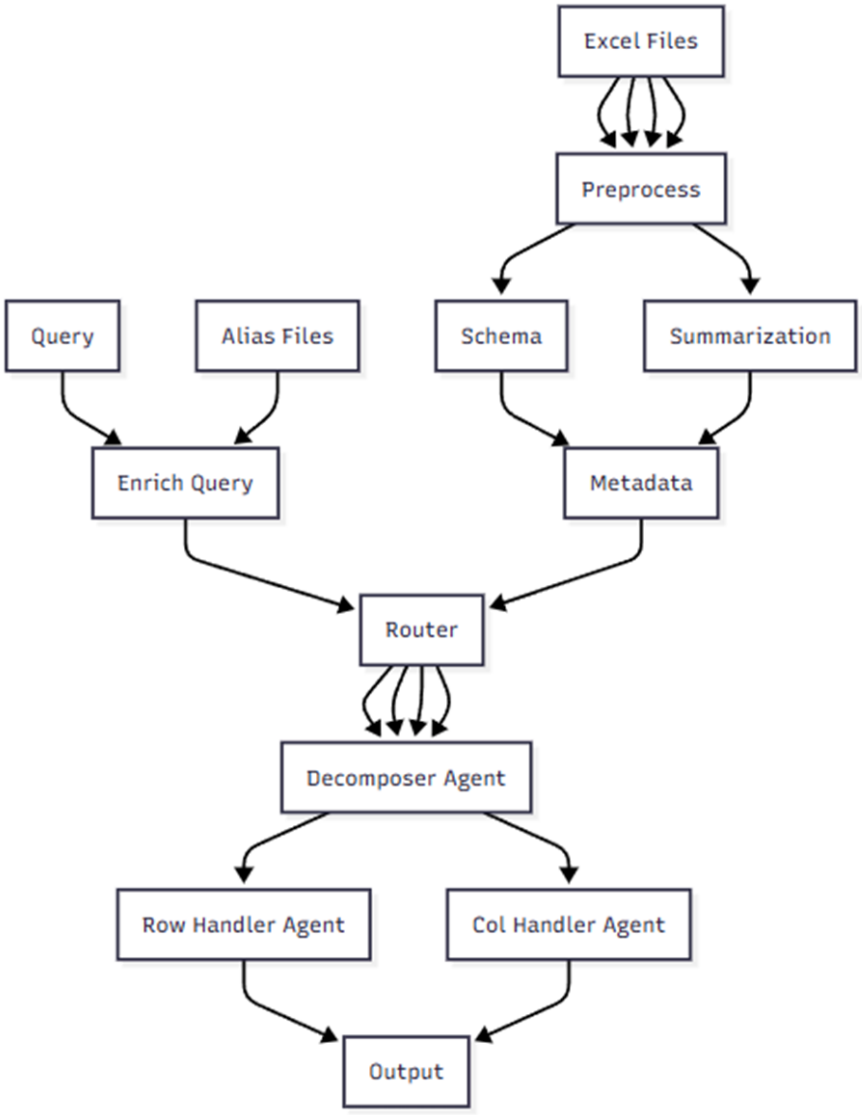

Figure 13: The final, unified architecture of the Excel Chatbox framework. It integrates the query enrichment, multi-file routing, and schema-based single-file execution engine into a cohesive, end-to-end system.

query routing, and identifier resolution, provides a scalable and reliable alternative to brittle, template-bound systems.

4.1 Discussion and Implications

A key advantage of our methodology is the direct and tangible output of the schema extraction process. Because our framework first deconstructs the spreadsheet into a structured, hierarchical schema, the final data retrieval step is a deterministic slicing operation. This has two significant benefits. First, the retrieved data subset retains its structural integrity, which allows it to be easily and accurately rendered back to the user as a clean data table. Second, this structured output can be seamlessly passed to visualization libraries to generate insightful and interactive charts on the fly, transforming our system from a mere query tool into a powerful, on-demand data analysis platform.

Furthermore, leveraging an LLM for query understanding imbues the system with a powerful capacity

for handling semantically complex requests. The framework is not limited to simple keyword matching. It can interpret qualitative or abstract concepts within the context of the extracted schema. For example, a user could ask for data from *"the odd months"* or *"the first quarter,"* and the LLM-powered agents can correctly translate this temporal concept into the specific column identifiers (‘Tháng 1‘, ‘Tháng 3‘, ‘Tháng 5‘, etc.) required for data retrieval. This capability extends to handling multi-step queries that require retrieving and implicitly combining information from different parts of a single file or even across multiple files.

4.2 Future Work

While our current framework establishes a strong foundation, there are several exciting avenues for future research and development. Our primary focus will be on enhancing the system's analytical capabilities and user experience.

One of the most promising next steps is the integration of a **Retrieval-Augmented Generation (RAG)** model. Currently, our system excels at retrieving precise data subsets. By adding a RAG component, we can empower the system to perform more advanced analytical reasoning and generate natural language summaries about the retrieved data.

5 Conclusion

In this paper, we addressed the significant challenge of enabling dynamic, conversational access to the vast amount of critical data locked within structurally complex and sensitive enterprise spreadsheets. We identified a critical gap in existing solutions, which are often too insecure for corporate data, too brittle for real-world file layouts, or too unscalable for practical deployment.

We presented the **Excel Chatbox**, a novel, security-first multi-agent framework that successfully overcomes these limitations. The core contribution of our work is an architecture that strictly separates a file's structural schema from its data values. By using Large Language Models to reason exclusively over this extracted schema, our system can interpret complex, hierarchical matrix tables and understand nuanced natural language queries without ever exposing sensitive information to an external service. Our methodology demonstrates a scalable approach that does not require hard-coded templates, and our implementation proves the viability of this framework through a functional, user-friendly web application capable of both precise data retrieval and dynamic visualization.

By decoupling semantic interpretation from local data execution, this work establishes a robust paradigm for building secure, intelligent, and user-centric data analysis tools for the enterprise. It transforms static reporting documents into active, queryable assets, empowering non-technical users to make faster, better-informed decisions. We believe this security-first, schema-driven approach represents a significant step forward in the ongoing effort to democratize data analytics in a safe and scalable manner.

References

[1] Wentao Chen, Ziqi Zhang, Yuxin Sun, Wei Luo, and William Wang. Instructexcel: A benchmark for natural language instruction on tabular data. *arXiv preprint arXiv:2309.11042*, 2023.

[2] Zhaoyuan Gao, Zirui Wang, Xing Zhang, Yican Liu, Zihan Wang, Shi Chang, Yifei Wang, Yifung Li, Haibin Sun, Wei Zhang, et al. Sheetcopilot: A general-purpose ai assistant for spreadsheets. *arXiv preprint arXiv:2305.19308*, 2023.

[3] Xinyu Kim, Hieu Le, Tuan Le, and Thien Van Nguyen. Spreadsheetcoder: Formula prediction from semi-structured context. *arXiv preprint arXiv:2106.15339*, 2021.

[4] The pandas development team. pandas-dev/pandas: Pandas. February 2020.

[5] Zilong Wang, Tao Gui, Qi Zhang, Jing Li, Wen-Bin Dai, Jipeng Wang, Ruotian Xu, and Xuan Zhang. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2310.14495*, 2023.

[6] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.