FinStat2SQL: A Text2SQL Pipeline for Financial Statement Analysis

Quang Hung Nguyen, Phuong Anh Trinh, Phan Quoc Hung Mai, Tuan Phong Trinh

Faculty of Mathematical Economics,
College of Technology, National Economics University, Vietnam
{11222618,11220655,11222607}@st.neu.edu.vn, ttphong@neu.edu.vn
Correspondence: maphquochung@gmail.com

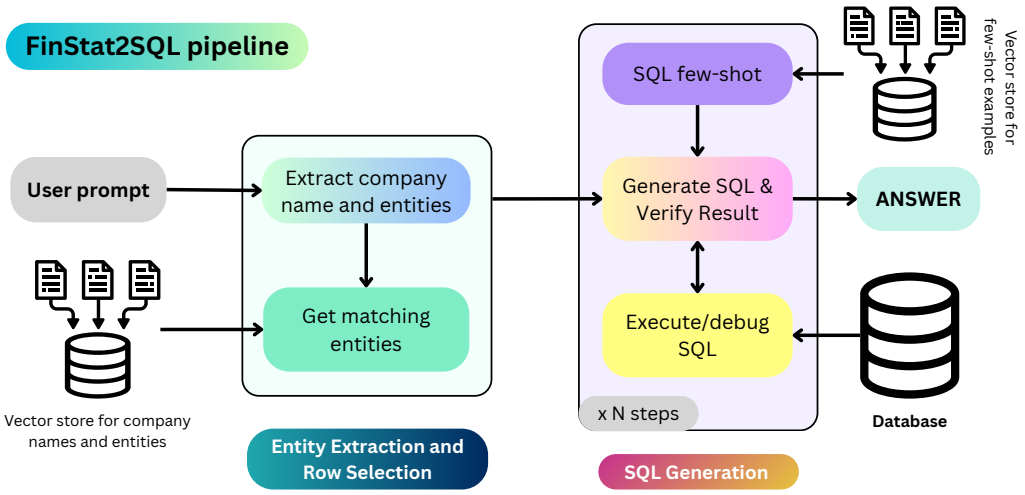

Figure 1: Diagram of the FinStat2SQL Pipeline (components with gradient color demonstrates generative model). The language model first analyzes the user's prompt, extract and retrieves relevant existing entities, and then iteratively executes and debugs SQL queries until a correct result is obtained, which is returned as the final answer. More information can be found in **Sec. 5**.

Abstract

Despite recent advances in LLMs, Text2SQL remains challenging for complex, domain-specific queries such as finance, where database designs and reporting standards vary widely. We introduce FinStat2SQL, a lightweight pipeline enabling natural language queries over financial statements, tailored to local standards like VAS. Our multi-agent setup combines large and small language models for entity extraction, SQL generation, and self-correction, and includes a fully automatic pipeline for synthetic data generation. Leveraging this synthetic data, we fine-tuned a 7B model that achieves 61.33% accuracy with sub-4s latency on consumer hardware, outperforming GPT-4o-mini on SQL generation. FinStat2SQL provides a scalable, cost-efficient solution for financial analysis. We made our source code publicly available at: Our Github Repository.

***Keywords*—** Text2SQL, Financial Analysis, Language Models

1 Introduction

Generating accurate SQL from users' natural language questions (text2sql) is a long-standing challenge (Hong et al., 2024), especially in the scenario that both the database system and user queries are growing more complex. For years, many attempts have been carried out (Katsogiannis-Meimarakis and Koutrika, 2023), introducing more sophisticated methods: from deep neural networks (Lei et al., 2020; Li et al., 2023a) to Generative Language Models (Li et al., 2023b; Pourreza and Rafiei, 2023). The results of Language Models (LMs) are very promising, however, many challenges persist when dealing with domain-specific tasks, which require further studies and refinement into each task. Database systems continue to grow rapidly and support for more languages becomes increasingly necessary, the potential applications of text2sql are broad, particularly in financial database systems, where structured querying via natural language can greatly enhance accessibility and efficiency. Applications of text2sql are wide, for example, in-

Proceedings of the 18th International Natural Language Generation Conference, pages 449–464
October 29–November 2, 2025.

tegrating into systems such as financial reporting automation, customer support chatbots, regulatory monitoring, as well as healthcare record querying, academic research databases, supply chain tracking, HR analytics (Singh et al., 2025).

Financial reports are essential for communicating a company's financial health and informing the decisions of investors, regulators, and analysts. However, manual analysis is often time-consuming, prone to human error, and inefficient for professionals who require fast, accurate insights from large volumes of financial data. While many companies globally follow the International Financial Reporting Standards (IFRS) issued by the International Accounting Standards Board (IASB) (Wagenhofer, 2009), national adaptations can vary significantly (Albu et al., 2014). For instance, Nguyen and Gong (2014) highlights that Vietnam Accounting Standards (VAS) only partially converge with IFRS, resulting in structural and interpretational differences. This diversity adds complexity to querying financial indicators across jurisdictions. To address these challenges, many text2sql techniques have gained traction as a promising solution for automating and streamlining financial analysis (Li et al., 2024; Zhang et al., 2024).

In this research, we design a pipeline for generating SQL from financial statistics (**Figure 1**), and evaluate the SQL query generation capabilities of various LMs within the context of the Vietnamese Companies Financial Statement System. Our study encompasses both Large Language Models (LLMs) and Small Language Models (SLMs), assessing their effectiveness in translating natural language questions into executable SQL queries. We particularly focus on SLMs since they are increasingly trending in agent design due to being more economical and flexible, while not requiring the full capacity of LLMs for many domain-specific tasks (Belcak et al., 2025). In addition, we propose a synthetic data generation pipeline for training SLMs and benchmark them with multiple training strategies to determine which approaches yield the best performance for domain-specific tasks.

2 Literature Review

In this section, we outline the challenges of querying financial indexes, provide a brief overview of recent advances in text2sql methods, and present key techniques for tuning LMs.

2.1 Problem in Financial Standards convergence

Financial statements are important and efficient for users, whether they are accountants, executive boards, government officials, or equity holders. Financial report standard convergence refers to the process of aligning national accounting standards with IFRS to reduce differences and enhance the comparability of financial statements across countries (Bahadir and Valev, 2015; Van den Berghe and Verweire, 2001). Convergence aims to harmonize recognition, measurement, and disclosure practices while allowing for gradual adaptation to international principles. Most accounting systems worldwide follow IFRS in preparing financial statements (IFRS Foundation, 2025). IFRS (IFRS Foundation, 2014) sets principles for recognizing, measuring, and disclosing financial elements. IFRS 15 governs revenue recognition through a five-step model, IFRS 16 addresses lease accounting by capitalizing most leases, and IFRS 9 regulates financial instruments with an expected credit loss model, promoting transparency and comparability.

Some countries, however, do not fully apply IFRS. In the case of Vietnam, this country applies a national standard, called VAS, which is not completely identical to IFRS. In particular, the main differences between VAS and IFRS are account codes, ratio definitions, and reporting formats. While IFRS is principle-based, VAS is rule-based. Although Phan et al. (2014) and Phan et al. (2018) discussed Vietnam's intentions to converge with IFRS, Nguyen and Gong (2014) further showed that VAS and IFRS only achieve mid-level convergence. When these conflicts still exist, it means that financial reporting systems are still recorded differently (Fontes et al., 2005; Larson and Street, 2004), causing many difficulties in querying.

2.2 Text2SQL

Published Datasets. In recent years, numerous datasets have been introduced to support the development of text2sql systems. The Spider dataset (Yu et al., 2018), for instance, spans 138 diverse domains to challenge cross-domain generalization. WikiSQL (Zhong et al., 2017) offers a large scale resource with over 24,000 tables derived from Wikipedia. Other datasets like Squall (Shi et al., 2020) and KaggleDBQA (Lee et al., 2021) further explore model generalization on unseen schemas. In contrast, several domain-specific datasets have

been curated for more focused evaluation, including those based on Yelp and IMDB reviews (Yaghmazadeh et al., 2017), the Advising dataset SEDE (Hazoom et al., 2021), and datasets targeting the Restaurants (Tang and Mooney, 2001) and Academic (Li and Jagadish, 2014) domains. These datasets aim to assess model performance within narrow domains, often prioritizing precision over generalization. However, these datasets still do not fully satisfy the demands of the industry, which are often more complex and challenging (Zhang et al., 2024).

Text2SQL methods. Prior to LLMs, text2sql relied on finetuning encoder decoder models. To improve representation, graph-relational neural networks, such as LGESQL (Cao et al., 2021) or RAT-SQL (Wang et al., 2019), were used to capture structural relationships among query tokens, tables, and columns. With the rise of LMs like T5 and LLaMA, fine-tuning methods have significantly improved text2sql performance on benchmarks like Spider. Recent advances include Graphix (Gan et al., 2021b), which enables multi-hop reasoning, Picard (Iyer et al., 2017), which supports constrained decoding, and RESDSQL (Gan et al., 2021a), the current state-of-the-art (SoTA) fine-tuning method on the Spider leaderboard. To improve SQL generation, MAC-SQL (Multi-Agent Collaborative Framework for Text-to-SQL) (Wang et al., 2023) and E-SQL (Question Enrichment in Text-to-SQL) (Caferoğlu and Ulusoy, 2024) decompose queries into sub-steps, aiding data understanding and reducing logical errors. However, this multi-step approach increases processing time due to added computational overhead.

Text2SQL in Finance. Despite its potential application, very little research has been conducted on the application of text2sql in finance. FinSQL (Zhang et al., 2024) is a model agnostic LLM-based text2sql framework tailored for financial analysis, addressing challenges like wide tables and limited domain specific datasets. Kumar et al. (2024) introduces BookSQL, a large-scale text2sql dataset focused on the accounting and financial domain, featuring 100k NL-SQL pairs and databases with over 1 million records. It highlights the challenges current models face in this domain, emphasizing the need for specialized approaches to support non-technical users in querying accounting databases.

2.3 Language model tuning methods

Recent research has explored parameter-efficient fine-tuning (PEFT) as a lightweight alternative to full model adaptation. Techniques such as Adapters (Houlsby et al., 2019), Prompt Tuning (Lester et al., 2021), Prefix-Tuning (Li and Liang, 2021), and LoRA (Hu et al., 2022) update less than 1% of model parameters while maintaining competitive performance, making them well-suited for rapid deployment under limited hardware. In parallel, advances in model alignment have focused on incorporating human feedback into optimization. Direct Preference Optimization (DPO) (Rafailov et al., 2023) simplifies reinforcement learning from human feedback by reframing it as a classification task, while Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024) extends this approach by modeling human cognitive biases through human-aware losses. Together, these works highlight the dual directions of improving efficiency in fine-tuning and ensuring alignment with human preferences.

3 Preliminaries

3.1 Database Construction

We constructed a financial database covering 200 major Vietnamese companies from 2016 to Q3 2024, including VN30, HNX30, and other industry leaders, with data collected from FiinPro[1]. Raw Excel files were processed and stored in SQL with a schema consisting of company details, financial statements, financial ratios, and vector databases for entity matching. Since Vietnam employs three reporting formats (banks, corporations, securities), we implemented a universal mapping table to standardize account codes and resolve structural inconsistencies (**Appendix B**). To evaluate scalability, two checkpoints were created: a training database with 102 companies and limited features, and a comprehensive testing database with 200 companies and expanded accounts, simulating full-scale deployment. For efficient querying and historical analysis, we adopted a Star schema design (Dedić and Stanier, 2016; Iqbal et al., 2019), centralizing financial figures in fact tables connected to descriptive dimensions (e.g., company, industry, category). This design reduces complex joins, ensures consistency across formats, and provides a scalable foundation for financial statement analysis.

[1]www.fiinpro.com

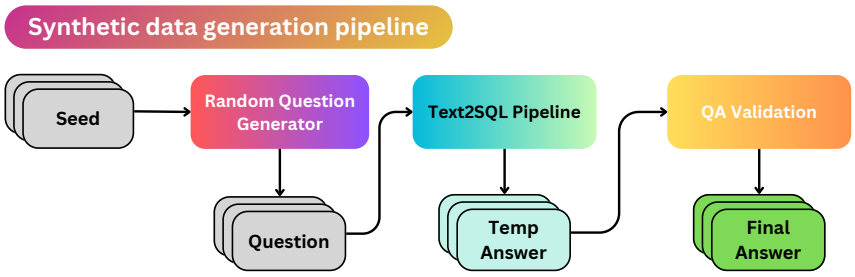

Figure 2: Synthetic data generation pipeline. Synthetic data generation involves using diverse seed attributes to create varied questions, generating temporary answers via a Text2SQL pipeline, and refining them through QA validation. This process produces high quality question–answer pairs for fine-tuning.

3.2 Evaluation Dataset

To comprehensively evaluate the FinStat2SQL pipeline in real-world scenarios, we curated a dataset of around 300 financial analysis questions. These were sourced from actual financial reports by stock exchanges, brokerage firms, and investment analysts, covering company performance, market sectors, and industry trends. Although many were originally presented in chart format, we reformulated them as SQL-based tasks to align with our system's capabilities. This evaluation dataset aims to evaluate the pipeline as a whole, not a single component.

To further strengthen the evaluation, we augmented the dataset with multiple-choice questions (MCQs). For each financial task, we created one to five MCQs to assess the model's reasoning, contextual understanding, and retrieval accuracy. In total, we have 821 different MCQs. This approach ensures a more thorough and realistic test of the system's ability to interpret and analyze structured financial data. Detail for this MCQs will be discussed in **Sec. 6.2**.

4 Synthetic Data

To train the LMs effectively for financial analysis, we created a large-scale synthetic question-answer dataset focused on exploring, interpreting, and analyzing financial statements. Inspired by Shirgaonkar et al. (2024), the dataset was generated through an automated pipeline designed to cover a broad range of tasks, including fundamental, technical, and comparative financial analysis. We used gemini-2.0-flash-thinking-exp-01-21[2] and GPT-4o mini[3] as the main generators for this synthetic data. The example of golden query and result is represented at **Appendix A**, and the synthetic pipeline is in **Figure 2**.

Description	Count
Primary scope of analysis	3
Secondary operations or subtasks	7
Types of financial analysis perspectives	11
Temporal scope for analysis	2
The persona or role context	4
Difficulty of the question	2

Table 1: Seed for random question generation.

4.1 Random Question Generator

In the synthetic question generation step, we begin by randomly generating questions. However, this approach presents challenges, as LLMs tend to be poor at producing diverse random outputs, often restricting themselves to a narrow range of topics and repeating patterns. To address this issue, we introduce the concept of "seed" to guide generation. Each seed specifies a particular combination of attributes, allowing the LLM to generate questions and thereby reduce duplication, details in **Table 1**. We then enumerate all seed combinations, and for each seed we generate only 10–15 questions. In total, this yields 3,696 unique seeds, from which we randomly select half for actual question generation.

4.2 Generator Pipeline

Next, in the synthetic answer generation step, we use the previously generated synthetic questions to produce temporary answers through the proposed pipeline described in Section 4. We set the number of few-shot examples to four, as this configuration provided a good balance and yielded the most reliable answers.

[2] ai.google.dev/gemini-api/docs/models#gemini-2.0-flash
[3] platform.openai.com/docs/models/gpt-4o-mini

Task Type	Count
Financial & Accounting Knowledge	1,800
Entity Extraction Tasks	4,000
SQL Generation Conversations	12,400
DPO training samples	2,000

Table 2: Synthetic training dataset composition. Entity Extraction Tasks is the sub-task in FinStat2SQL Pipeline

4.3 QA Validation

To ensure dataset quality, we used the LLM-as-a-Judge framework (Zheng et al., 2023), where multiple evaluators classified each generated pair as correct (1) or incorrect (0); only correct pairs were retained, yielding 12,400 samples for SFT. We then shortened system prompts, removed few-shot examples, and collected suboptimal outputs (solved by only one model) for alignment training. Each correct/incorrect pair forms a DPO training pair, while KTO training uses the True/False flags (Ethayarajh et al., 2024).

In addition to SQL-related samples, we generated 1,800 QA pairs on Vietnamese accounting knowledge, following a simplified version of Yuen et al. (2025). Given the factual contexts of documents, laws, and regulations, an LLM produced questions and answers, which form the final synthetic corpus (**Table 2**).

4.4 Overlap between synthetic training and evaluation data

Our evaluation validates the end-to-end chatbot pipeline, from synthetic data generation to inference. Since the application inherently involves self-generated data, synthetic QA pairs may overlap between training and evaluation, emphasizing adaptability and stability rather than strict out-of-distribution generalization (Hancock et al., 2019). We also analyze the degree of overlap, measuring semantic similarity between synthetic and real data using bge-large-en-v1.5[4] (Xiao et al., 2024) for embeddings and cosine similarity, inspired by Chim et al. (2025) and Zamzmi et al. (2025). For each sample, we take the maximum cosine similarity result between each synthetic sample as score, and calculate the similarity on them. This method will flag any leakage if the score is too high. Mathematically, let $Q = \{q_1, q_2, \ldots, q_m\}$ be the set of evaluation questions and $S = \{s_1, s_2, \ldots, s_n\}$ the set of synthetic questions. Let ϕ be the embedding function and sim the cosine similarity measure. Then, for each $q_i \in Q$, the representation score is defined as:

$$\text{score}(q_i) = \max_{s \in S} \text{sim}\big(\phi(q_i), \phi(s)\big)$$

The set of all scores is

$$\text{Score}(Q, S) = \left\{ \max_{s \in S} \text{sim}\big(\phi(q), \phi(s)\big) \;\middle|\; q \in Q \right\}$$

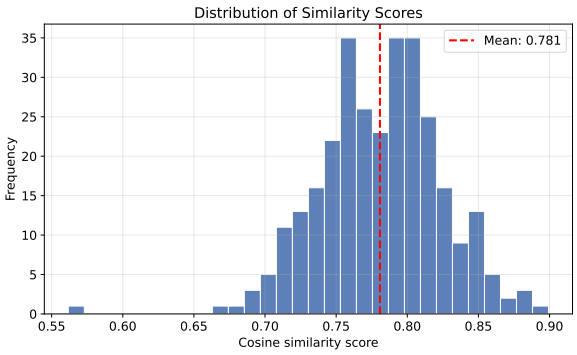

Figure 3: Distribution of Score between evaluation set Q and synthetic samples S

Figure 3 presents a histogram of Score between evaluation set Q and synthetic samples S. The mean score is 0.78, with the 95th percentile at 0.8496. While the scores are generally high, they suggest that evaluation questions do not excessively overlap with synthetic training data. **Table 3** provides illustrative examples of question pairs at two thresholds: mean and 95^{th} percentile. Intuitively, the 95^{th} percentile group shows synthetic questions that are much closer in meaning and detail to the original, while the mean group synthetic questions, though still related, are looser in focus and phrasing.

5 FinStat2SQL Pipeline

In this section, we introduce FinStat2SQL, a pipeline that converts noisy financial queries into accurate SQL using entity extraction, code generation, and self-correction (**Figure 1**). The pipeline is specifically adapted to the structure of VAS, which differs from IFRS, ensuring accurate handling of local financial data. For reference, we include the prompt template of each component in the pipeline in **Appendix C**

5.1 Text2SQL Pipeline

Entity Extraction. The Entity Extraction step uses LLMs to identify key elements in user queries for

[4]huggingface.co/BAAI/bge-large-en-v1.5

Group	Score	Question
p95	original	Non-performing loan (NPL) ratio of the banking sector in 2023
	0.8496	For the banking sector listed on HOSE, compare the ratio of Non-Performing Loans (NPLs) to total loans for the year 2023. How does this ratio vary across different banks, and what factors might explain the differences?
Mean	original	Bad debt ratio of banking industry from Q1 2020 to Q1 2024
	0.7800	Analyze the trend of the Non-Performing Loan ratio for companies in the 'Banking' industry listed on the VN30 index from 2020 to Q3 2024.

Table 3: Example of original evaluation question and its synthetic examples. The table demonstrates the evaluation questions and similarity closest samples of synthetics samples at the mean and 95^{th} percentile.

generating accurate SQL, focusing on four fields: Industry, Company Name, Financial Statement Account, and Financial Ratio. To achieve this, the agent utilizes a prompt-based approach with LM. This prompt-based approach minimizes ambiguity, ensures precise parsing, and allows the system to infer additional relevant metrics, offering more flexibility and accuracy than traditional NER methods (see **Figure 4**).

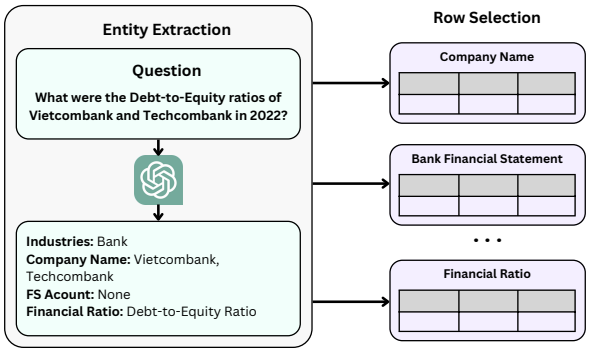

Figure 4: Diagram of Entity Extraction and Row Selection process

Row Selection. After entity extraction, matching candidates are retrieved from vector databases, with a selection mechanism used to narrow down relevant results, especially when entities are ambiguous. The selected candidates are then passed to the Code-Generation LLM to determine the best match based on context.

Similarity Search. Full-text search is insufficient for extracting relevant entities in specialized domains like financial statements under VAS due to semantic nuances, such as the difference between "Net Income" (IFRS) and "Profit After Tax" (VAS), or colloquial term "Bad debt" and industry-standard term "Non-Performing Loan". We leverage similarity search using vector databases to map entities based on their semantic meaning, ensuring accurate alignment even with different phrasing. By using similarity search, we bridge the gap between commonly trained LLMs and exact VAS-specific terminology in database, enhancing entity matching and improving query results.

From this paragraph, we describe three SQL Generation process, see **Figure 5** for more illustrated details.

Few-shots. Without examples, LLM-generated SQL queries tend to be overly complex or inefficient. Providing well-designed few-shot examples helps the LLM produce more accurate and optimized queries by guiding it toward standard structures. These examples can be retrieved from a database, such as via a retrieval-augmented generation pipeline. In our design, we employ 71 examples to handle common user queries and cover issues encountered during the development process. Although higher-quality few-shot examples are known to improve generation performance, optimizing this aspect falls outside the scope of this study.

Self-correction. In the query generation process, errors often occur, requiring mechanisms like self-debugging or self-correction to make the output executable. These errors are typically syntax errors, where the SQL does not follow proper syntax, and logical errors, where the query does not capture the user's intent. Li et al. (2024) noted that self-correction is particularly effective for syntax errors, but has limited impact on logical inconsistencies. Based on this, our methodology uses self-correction as a fallback to fix any remaining syntax issues or incorrect data during the query cleaning process, ensuring the system's reliability and that the resulting query answers the user's problem accurately.

Decomposition and Multistep generation. To improve SQL generation quality, frameworks like MAC-SQL and E-SQL (Wang et al., 2023; Caferoğlu and Ulusoy, 2024) break down queries into

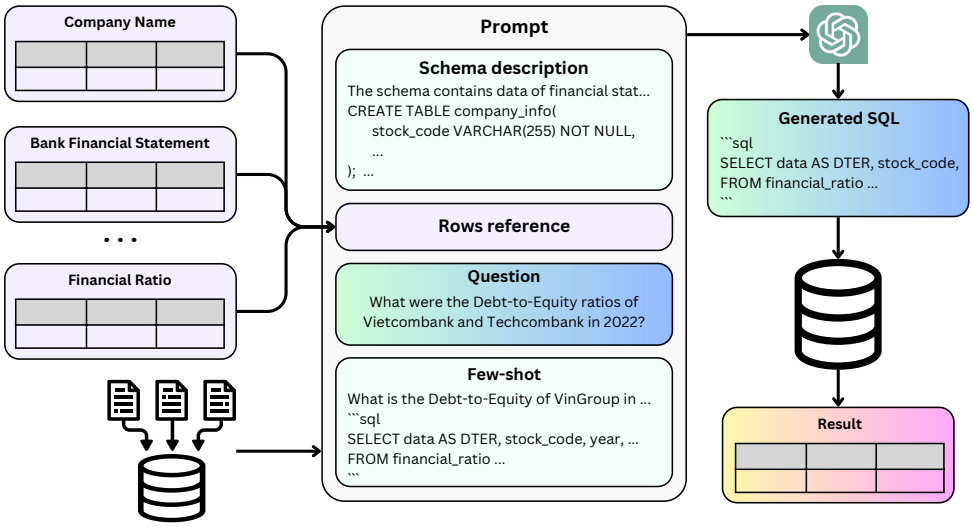

Figure 5: SQL generation process. The system integrates schema descriptions, row references, and user questions into prompts, enhanced with few-shot examples, to generate SQL queries.

independent sub-queries, allowing the model to better understand the data structure, similar to an exploratory data analysis (EDA) phase, which reduces logical and data-related errors. However, this multi-step reasoning approach increases processing time due to the additional computational overhead from each sub-query.

5.2 Fine-tuning

We fine-tune SLMs on our QA dataset using the `LLaMAFactory` framework with the LoRA technique (Hu et al., 2022). In this setup, we prepare the QA dataset in the required instruction–response format and use `LLaMAFactory` (Zheng et al., 2024) to manage the training process efficiently. Instead of updating all model parameters, LoRA injects trainable low-rank matrices into specific layers of the LLM, which significantly reduces memory and computational requirements.

Further, we experiment with the effectiveness of alignment training methods, specifically Direct Preference Optimization (DPO) (Rafailov et al., 2023) and Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024), to align the model's outputs with human preferences and cognitive biases. These alignment techniques enable the finetuned model to generate SQL queries that are not only correct but also aligned with human evaluative behavior.

6 Experiment

6.1 Setup

To evaluate our pipeline's effectiveness in processing Vietnamese financial statements, we tested a range of LM across two categories (see **Table 4**): Commercial LLMs (denoted LLM in this session) and SLMs. DeepSeek-V3 has 671B params (37B activated each token), Qwen 2.5 has 2 model versions: 3B and 7B. LLMs are large, closed-source models accessed via API due to high resource demands or require expensive infrastructure, while smaller, open-source SLMs offer easier deployment and can achieve competitive performance with fine-tuning.

Category	Model Name	Params
LLM	LLaMA 3.3 Instruct	70B
	Qwen 2.5 Coder	32B
	Gemini 2.0 Flash	-
	Gemini 2.0 Flash Think	-
	GPT-4o Mini	-
	Deepseek V3	37B (671B)
SLM	LLaMA 3.1 Instruct	8B
	LLaMA 3.2 Instruct	8B
	Qwen 2.5 Instruct	3B – 7B
	Qwen 2.5 Coder Instruct	3B – 7B

Table 4: Summary of Models Used in the Experiments.

We finetune SLMs using Llama Factory (Zheng et al., 2024), and we illustrate our hardware specifications and hyperparameters for SLM finetuning and techniques in **Appendix D**. In our experiment, DPO and KTO were used to align SLMs by leveraging GPT-4o-mini's incorrect outputs and Gemini's improved responses as training pairs, aiming to enhance reasoning in the text2sql pipeline.

In general, our **finstat2sql** is a fine-tuned version

of Qwen 2.5 Coder for each model size, integrated with our proposed pipeline for enhanced performance.

6.2 Evaluation Metrics

In Text2SQL, evaluation metrics are typically divided into content-matching and execution-based approaches (Mohammadjafari et al., 2024). While effective for fixed schemas, these methods penalize valid outputs expressed in alternative formats, making them less suitable for QA-oriented tasks. LLM-as-a-Judge has emerged as an alternative, but even state-of-the-art models often introduce bias by misclassifying results, overlooking data correctness, or overemphasizing table structure. To address this, we propose a **hybrid evaluation** that integrates structural accuracy with contextual correctness. Our method leverages an LLM to answer multiple-choice questions derived from the ground truth, selecting the correct option or "I don't know" (IDK) when uncertain. This framing simplifies the evaluation task, reduces hallucination risks (Kalai et al., 2025), and enables partial credit in measuring table completeness. We use gemini-2.0-flash as the judge LLM.

For each question q, we construct a set of related MCQs, which can only be correctly answered if the table result for q is accurate. The accuracy of q is then defined as the mean score across its MCQs, where each MCQ contributes 1 if answered correctly and 0 otherwise. Formally, let $M_q = \{m_1, m_2, \ldots, m_{|M_q|}\}$ denote the set of MCQs associated with question q, and let $\delta(m_i)$ be an indicator function such that:

$$\delta(m_i) = \begin{cases} 1 & \text{if the LM answers } m_i \text{ correctly,} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the accuracy for question q is:

$$\text{Acc}(q) = \frac{1}{|M_q|} \sum_{i=1}^{|M_q|} \delta(m_i).$$

6.3 Result

Proprietary models (**Table 5**), particularly the Gemini family from Google, outperform all other models in the text2sql evaluation, with the "thinking" variant achieving the highest accuracy (72.03%). Despite both Gemini 2.0 Flash and GPT-4o Mini being optimized for speed, Gemini demonstrates significantly better performance, suggesting its architecture or training data is better suited for this task. Among open-source models, DeepSeek-V3 performs competitively, likely due to its efficient MoE-based architecture using 37B active parameters. In contrast, dense models like Qwen2.5 32B Coder and LLaMA 3.3 70B Instruct show similar performance around 66%, indicating a possible plateau at this scale. Overall, the evaluation highlights Gemini's strength in both speed and reasoning.

Model	Acc.
gemini-2.0-flash-thinking	**0.7203**
gemini-2.0-flash	*0.6969*
deepseek-v3	0.6929
qwen2.5-coder-32B-instruct	0.6590
llama-3.3-instruct	0.6569
gpt-4o-mini	0.5914

Table 5: Performance of LLMs on Evaluation set

Params	Model	Acc.
7–8B	qwen2.5-coder-instruct	0.4793
	qwen2.5-instruct	0.4571
	llama3.1-instruct	0.3804
	finstat2sql	**0.6133**
3B	qwen2.5-coder-instruct	0.3019
	llama-3.2-instruct	0.2126
	finstat2sql	**0.5558**
1.5B	qwen2.5-coder-instruct	0.2299
	finstat2sql	**0.5301**

Table 6: Evaluation on SLMs

Table 6 demonstrates the significant impact of task-specific fine-tuning on SLMs for the text2sql task. Across all sizes, the finstat2sql models (7B: 0.6133, 3B: 0.5558, 1.5B: 0.5301) consistently outperform their corresponding base models from the Qwen2.5 and Llama3 families, highlighting the benefits of specialization. Although the Qwen2.5 coder models show stronger baseline performance than Llama3, fine-tuning proves to be the key differentiator, most notably, the 7B finstat2sql model surpasses even the much larger 70B Llama and GPT-4o-mini in accuracy. This supports the conclusion that, with targeted fine-tuning, smaller models can rival or even exceed the performance of much larger or proprietary alternatives, offering a cost-efficient solution for domain-specific tasks.

Base model	Training type	Validation test	Evaluation Acc.
qwen2.5-coder-3b	SFT	0.7195	**0.5558**
	SFT+KTO	**0.7257**	0.5204
	SFT+DPO	0.6761	0.4644
qwen2.5-coder-1.5b	SFT	0.6780	**0.5301**
	SFT+KTO	**0.6879**	0.4419
	SFT+DPO	0.6288	0.4615

Table 7: Evaluation on Model Alignment Methods.

During the training process, we sub-sample the total training data to get a validation set and use the same QA Validation module in the Synthetic Data Generation Pipeline for measuring accuracy. **Table 7** shows that alignment methods, especially DPO, significantly reduced performance, with both 3B and 1.5B models experiencing major accuracy drops on the private test set; hence, further evaluation was skipped. This behavior is expected, as DPO is sensitive and only performs well if both the data preparation and the initial SFT steps are carefully calibrated (Feng et al., 2024). While KTO is designed to fix the DPO's drawback and slightly better than SFT private test accuracy, it has lower performance on the main evaluation set. This indicates that the alignment method are sensitive to the dataset and does not as robust as SFT.

7 Conclusion

7.1 Discussion & Conclusion

This research demonstrates that FinStat2SQL allows users to query complex financial data using natural language, significantly lowering the barrier for non-experts. The pipeline balances accuracy and efficiency effectively, with simpler architectures often outperforming more complex ones. While proprietary models like Gemini-2.0-Flash-Thinking achieved the highest accuracy (72.03%), open-source and fine-tuned models, especially DeepSeek V3 and Qwen2.5-Coder proved highly competitive. The 7B finstat2sql model achieved performance comparable to, and in some cases surpassing, that of larger models, suggesting that fine-tuned SLMs can offer a promising balance between capability and efficiency. Moreover, the integration of a synthetic data generation pipeline provided scalable supervision for model training, while the explicit focus on Vietnamese financial statements (FS) ensured domain-specific robustness. Finally, by adopting an agentic design—where entity extraction, SQL generation, and self-correction operate as coordinated agents—the system enhances reliability and adaptability. Model alignment techniques were found to be unnecessary in many cases, and the system has been deployed as a financial chatbot with nearly 70% query success and sub-4-second response times, addressing a critical gap in Vietnam's financial automation landscape.

7.2 Limitation & Future Works

This study has several limitations, including the focus on VN30 and HNX30-listed firms while excluding SMEs, challenges in handling Vietnamese financial terminology variations, and reliance on VAS rather than broader frameworks like US GAAP, which restricts cross-national applicability. In addition, dependence on closed-source models such as GPT-4o-mini raises cost and infrastructure concerns for real-time deployment. To address these issues, future work will expand the dataset to include SMEs and unlisted firms, adapt the pipeline to international standards, and integrate predictive analytics for tasks like trend forecasting and risk assessment. We also plan to refine training methods, explore data augmentation, and investigate improved alignment strategies to enhance the robustness, scalability, and global utility of FinStat2SQL.

Acknowledgments

We extend our gratitude to the professors, lecturers, and colleagues at the National Economics University for their consultation in conducting this research. We appreciate INLG 2025 reviewers for their valuable insights and feedback on this work.

Supplementary Materials Availability Statement: We publish our source code for reference and further studies at: **this Github repository**. The dataset used for fine-tuning is publicly available under the license *'Creative Commons Attribution Non Commercial 4.0'* at: doi.org/10.57967/hf/6485.

Ethics Statement: We confirm that this work adheres to the ACL Code of Ethics

(https://www.aclweb.org/portal/content/acl-code-ethics). No ethical concerns beyond those common to standard research in this area have been identified.

References

Cătălin Nicolae Albu, Nadia Albu, and David Alexander. 2014. When global accounting standards meet the local context—insights from an emerging economy. *Critical perspectives on accounting*, 25(6):489–510.

Berrak Bahadir and Neven Valev. 2015. Financial development convergence. *Journal of Banking & Finance*, 56:61–71.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *Preprint*, arXiv:2506.02153.

Hasan Alp Caferoğlu and Özgür Ulusoy. 2024. E-sql: Direct schema linking via question enrichment in text-to-sql. *arXiv preprint arXiv:2409.16751*.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: line graph enhanced text-to-sql model with mixed local and non-local relations. *arXiv preprint arXiv:2106.01093*.

Jenny Chim, Julia Ive, and Maria Liakata. 2025. Evaluating synthetic data generation from user generated text. *Computational Linguistics*, 51(1):191–233.

Nedim Dedić and Clare Stanier. 2016. An evaluation of the challenges of multilingualism in data warehouse development.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Duanyu Feng, Bowen Qin, Chen Huang, Zheng Zhang, and Wenqiang Lei. 2024. Towards analyzing and understanding the limitations of dpo: A theoretical perspective. *Preprint*, arXiv:2404.04626.

Alexandra Fontes, Lúcia Lima Rodrigues, and Russell Craig. 2005. Measuring convergence of national accounting standards with international financial reporting standards. In *Accounting forum*, volume 29, pages 415–436. Elsevier.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-sql models against synonym substitution. *arXiv preprint arXiv:2106.01065*.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-sql generalization. *arXiv preprint arXiv:2109.05157*.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-SQL in the wild: A naturally-occurring dataset based on stack exchange data. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, pages 77–87, Online. Association for Computational Linguistics.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

IFRS Foundation. 2014. *IFRS 15 Revenue from Contracts with Customers*. International Accounting Standards Board (IASB), London, United Kingdom.

IFRS Foundation. 2025. *Who uses IFRS Accounting Standards?* International Accounting Standards Board (IASB), London, United Kingdom. Retrieved from https://www.ifrs.org/use-around-the-world/use-of-ifrs-standards-by-jurisdiction/.

M Zafar Iqbal, Ghulam Mustafa, Nadeem Sarwar, Syed Hamza Wajid, Junaid Nasir, and Shaista Siddque. 2019. A review of star schema and snowflakes schema. In *International Conference on Intelligent Technologies and Applications*, pages 129–140. Springer.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. 2025. Why language models hallucinate. *Preprint*, arXiv:2509.04664.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32(4):905–936.

Rahul Kumar, Amar Raja Dibbu, Shrutendra Harsola, Vignesh Subrahmaniam, and Ashutosh Modi. 2024. Booksql: A large scale text-to-sql dataset for accounting domain. *arXiv preprint arXiv:2406.07860*.

Robert K Larson and Donna L Street. 2004. Convergence with ifrs in an expanding europe: progress and obstacles identified by large accounting firms' survey. *Journal of international accounting, auditing and taxation*, 13(2):89–119.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-sql. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8(1):73–84.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data*, 2(3):1–28.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Ma Chenhao, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*, volume 36, pages 42330–42357. Curran Associates, Inc.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Ali Mohammadjafari, Anthony S Maida, and Raju Gottumukkala. 2024. From natural language to sql: Review of llm-based text-to-sql systems. *arXiv preprint arXiv:2410.01066*.

Anh Tuan Nguyen and Guangming Gong. 2014. Measurement of formal convergence of vietnamese accounting standards with ifrs. *Australian Accounting Review*, 24(2):182–197.

Duc Phan, Mahesh Joshi, and Bruno Mascitelli. 2018. What influences the willingness of vietnamese accountants to adopt international financial reporting standards (ifrs) by 2025? *Asian Review of Accounting*, 26(2):225–247.

Duc Phan, Bruno Mascitelli, and Meropy Barut. 2014. Perceptions towards international financial reporting standards (ifrs): The case of vietnam. *Global Review of Accounting and Finance Journal*, 5(1):132–152.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348.

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

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1849–1864, Online. Association for Computational Linguistics.

Anup Shirgaonkar, Nikhil Pandey, Nazmiye Ceren Abay, Tolga Aktas, and Vijay Aski. 2024. Knowledge distillation using frontier open-source llms: Generalizability and the role of synthetic data. *Preprint*, arXiv:2410.18588.

Aditi Singh, Akash Shetty, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. A survey of large language model-based generative ai for text-to-sql: Benchmarks, applications, use cases, and challenges. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 00015–00021. IEEE.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477, Freiburg, Germany.

Lutgart Van den Berghe and Kurt Verweire. 2001. Convergence in the financial services industry. *The Geneva Papers on Risk and Insurance. Issues and Practice*, 26(2):173–183.

Alfred Wagenhofer. 2009. Global accounting standards: reality and ambitions. *Accounting Research Journal*, 22(1):68–80.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and 1 others. 2023. Mac-sql: A multi-agent collaborative framework for text-to-sql. *arXiv preprint arXiv:2312.11242*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 641–649, New York, NY, USA. Association for Computing Machinery.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. 1(OOPSLA).

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 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, Brussels, Belgium. Association for Computational Linguistics.

Sizhe Yuen, Ting Su, Ziyang Wang, Yali Du, and Adam J. Sobey. 2025. Automatic dataset generation for knowledge intensive question answering tasks. *Preprint*, arXiv:2505.14212.

Ghada Zamzmi, Adarsh Subbaswamy, Elena Sizikova, Edward Margerrison, Jana G Delfino, and Aldo Badano. 2025. Scorecard for synthetic medical data evaluation. *Communications Engineering*, 4(1):130.

Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. 2024. Finsql: model-agnostic llms-based text-to-sql framework for financial analysis. In *Companion of the 2024 International Conference on Management of Data*, pages 93–105.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

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

A Example of Evaluation Dataset

Figure 6 shows a golden query for the question 'Banks with credit growth higher than average in Q3 2023?'. The golden query is generated by the LLM and is designed to work reliably on the database and in real contexts. Its result is presented in **Table 8**. To construct the evaluation dataset, the LLM is asked to generate multiple-choice questions based on this result table (**Table 9**). In addition to four options, an extra choice **"E. I don't know"** is included. The evaluation LLM is encouraged to select option **E** whenever uncertain, in order to reduce hallucination.

B Database Design

Figure 7 represents the financial data database design, integrating company-level and industry-level financial information. The company_info table is the core, storing company identity, industry, and exchange details, and linking to financial statements, ratios, and ownership structures. Financial_statement and financial_ratio tables capture quarterly and yearly performance data, with mapping tables providing standardized codes and descriptions for consistency. Parallel industry-level tables (industry_financial_statement and industry_financial_ratio) store aggregated statistics such as sums and means for benchmarking across industries. Additional explanation tables enrich the dataset by linking financial categories and ratios to human-readable descriptions in multiple languages.

C Prompt Detail

C.1 Entity extraction and Row selection Agent Prompt

As shown in **Figure 8**, this prompt is designed to guide the pipeline in extracting key entities and relevant financial data sources from a user's question. It ensures the system identifies company names, industries, financial statement accounts, and ratios that can answer the query. The results are structured in a standardized JSON format, making them easy to use in downstream SQL generation or reasoning steps.

C.2 Schema Description Prompt

Figure 9 shows the Schema Description Prompt, which guides the pipeline in interacting with the Vietnamese financial database. It outlines the scope (VAS-based quarterly and annual data), key tables (company info, statements, industry aggregates, ratios, explanatory data), and notes for interpretation. The prompt enforces strict rules: use mapping tables for codes, never compute ratios manually, always specify a quarter (defaulting to annual), and include LIMIT for efficient queries.

C.3 Self Correction Prompt

Figure 10 shows self self-correction prompt. This prompt facilitates the pipeline to validate whether a SQL query result correctly answers the user's original task. It guides the system to check the output, confirm correctness, or generate a corrected query with reasoning if the result is missing or unsuitable.

D Training Config

Table 10 specifies the config during fine-tuning SLMs, including hardware and trainer configurations.

Configuration	Value/Details
Training steps	8,000 - 10,000 steps
Batch size	8
Optimizer	AdamW
Learning rate	2e-5
Warm-up rate	0.1
LoRA rank	64
GPU (1.5B 3B models)	NVIDIA RTX 4090
GPU (7B model)	NVIDIA A100
Training duration (7B)	16 hours
Alignment Methods	KTO & DPO
DPO Samples	2,000
KTO Samples	4,000

Table 10: Set up for training SLMs

Golden Query

```
Question: 'Banks with credit growth higher than average in Q3 2023'

Query:
WITH bank_credit_growth AS (
 SELECT
 fr.stock_code,
 ci.industry,
 fr.year,
 fr.quarter,
 fr.data AS credit_growth_yoy
 FROM financial_ratio fr
 JOIN company_info ci ON fr.stock_code = ci.stock_code
 WHERE fr.ratio_code = 'CDGYoY'
 AND ci.is_bank = TRUE
 AND fr.year = 2023
 AND fr.quarter = 3
),
industry_avg_growth AS (
 SELECT
 industry,
 year,
 quarter,
 data_mean AS industry_credit_growth
 FROM industry_financial_ratio
 WHERE ratio_code = 'CDGYoY'
 AND industry = 'Banking'
 AND year = 2023
 AND quarter = 3
)
SELECT
 b.stock_code,
 b.year,
 b.quarter,
 b.credit_growth_yoy,
 i.industry_credit_growth
FROM bank_credit_growth b
JOIN industry_avg_growth i ON b.industry = i.industry
WHERE b.credit_growth_yoy > i.industry_credit_growth
ORDER BY b.credit_growth_yoy DESC
```

Figure 6: Golden query example. A golden query for question 'Banks with credit growth higher than average in Q3 2023?'

Stock code	Year	Quarter	Credit-YoY	Industry YoY
HDB	2023	3	0.64	0.2395
VPB	2023	3	0.52	0.2395
MSB	2023	3	0.35	0.2395
KLB	2023	3	0.34	0.2395
...	...	...	...	...

Table 8: Example Golden Query result table

Question	A	B	C	D	E
Which bank among the listed ones demonstrated the highest credit growth year over year in Q3 2023?	VPB	**HDB**	MSB	TCB	IDK
How many banks achieved a credit growth rate exceeding the industry average in Q3 2023?	10	12	14	8	**IDK**
What was the industry average credit growth in Q3 2023?	0.2475	0.6445	**0.2395**	0.2825	IDK

Table 9: Multiple Choice Questions. Each MCQ will have additional choice "E. I don't know" (IDK).

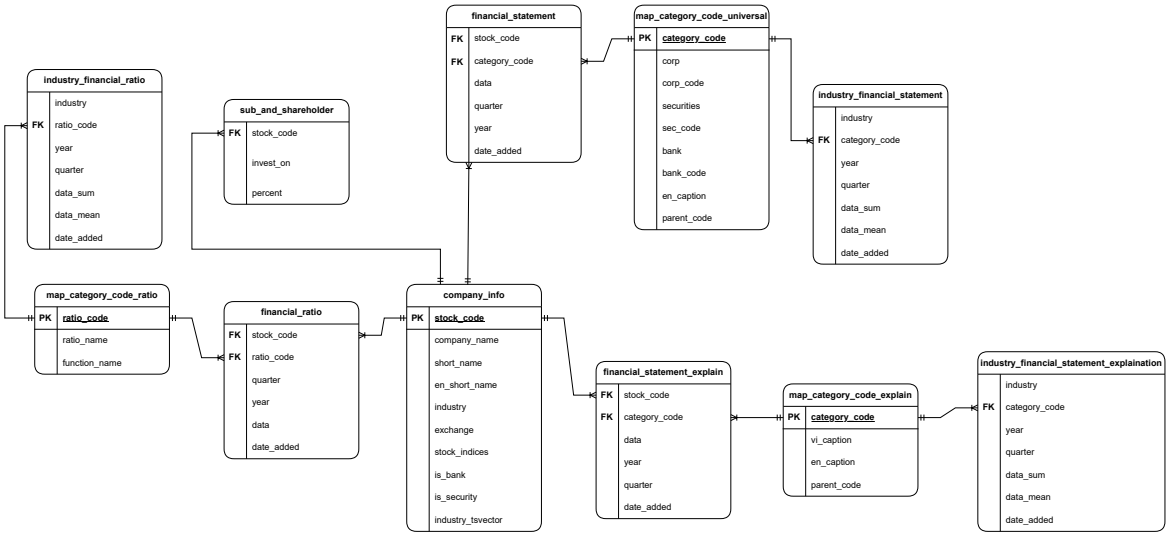

Figure 7: ERD database design for text2sql experiments

User Prompt

```
<task>
Based on given question, analyze and suggest the suitable accounts in the financial
statement and/or financial ratios that can be used to answer the question.
Extract the company name and/or the industry that positively mentioned based on the
given question.
</task>
<question>
{task}
</question>
Analyze and return the accounts and entities that useful to answer the question.
Return in JSON format, followed by this schema.
‘‘‘
{{
 "industry": list[str],
 "company_name": list[str],
 "financial_statement_account": list[str],
 "financial_ratio": list[str]
}}

‘‘‘
Note:
- Return an empty list if no related data is found.
- If the question include tag "4 nearest quarter", "trailing twelve months
(TTM)","YoY" or "QoQ", include the name and tag into ‘financial_ratio‘ and/or
‘financial_statement_account‘
- If "YoY" and "QoQ" ratio are mentioned, include the original account in
‘financial_statement_account‘ and the ratio in ‘financial_ratio‘.
<example>
### Question:
Net Income YoY and ROE 4 nearest quarter of HPG in 2023
### Response:
‘‘‘json
 {{
 "industry": [],
 "company_name": ["HPG"],
 "financial_statement_account": ["Net Income"],
 "financial_ratio": ["Net Income YoY", "ROE 4 nearest quarter"]
 }}
‘‘‘
</example>
```

Figure 8: Entity extraction and Row selection prompt

System Prompt

```
<overall_description>
The database conatains financial statments of Vietnamese firms, followed by the
regulation of Vietnamese Accounting Standard (VAS). The database includes two
reporting periods: quarterly (1, 2, 3, 4) and annually (quarter = 0).
</overall_description>
### PostgreSQL tables in OpenAI Template
<schema>
- **company_info**(stock_code, industry, exchange, stock_indices, is_bank,
is_securities)
- **sub_and_shareholder**(stock_code, invest_on)
- **financial_statement**(stock_code, year, quarter, category_code, data,
date_added)
- **industry_financial_statement**(industry, year, quarter, category_code,
data_mean, data_sun, date_added)
- **financial_ratio**(ratio_code, stock_code, year, quarter, data, date_added)
- **industry_financial_ratio**(industry, ratio_code, year, quarter, data_mean,
date_added)
- **financial_statement_explaination**(category_code, stock_code, year, quarter,
data, date_added)
</schema>
### Note on schema description:
- For industry tables, column ‘data_mean‘ is average data of all firms in that
industry, while ‘data_sum‘ is the sum of them.
- Table ‘financial_statement_explaination‘ contains information which is not covered
in 3 main reports, usually about type of loans, debt, cash, investments and
real-estate ownerships.
- With YoY ratio in ‘financial_ratio‘, you should recalculate the ratio if the time
window is not 1 year.
### Note on query:
- You will be provided a mapping table for ‘category_code‘ and ‘ratio_code‘ to select
suitable code.
- For any financial ratio, it must be selected from the database rather than being
calculated manually.
- Always include a ‘quarter‘ condition in your query. If not specified, assume using
annual reports (‘quarter‘ = 0).
- Always include LIMIT.
```

Figure 9: Schema description prompt

User Prompt

```
<result>
{sql_result}
</result>

<correction>
Based on the SQL table result in <result> tag, do you think the SQL queries is correct
and can fully answer the original task? If there is no SQL Result table on <result>
tag, it means the preivous queries return nothing, which is incorrect.
If the result of SQL query is correct and the table is suitable for <task> request,
you only need to return YES under *Decision* heading. You must not provide the SQL
query again.
Otherwise, return No under *Decision* heading, think step-by-step under
*Reasoning* heading again and generate the correct SQL query under *SQL Query*.
Return in the following format (### SQL Query is optional):
### Decision:
{{Your decision}}
### Reasoning:
{{Your reasoning}}
### SQL Query:
{{Corrected SQL query}}
</correction>
```

Figure 10: Self correction prompt