

# ODTQA-FoRe: An Open-Domain Tabular Question Answering Dataset for Future Data Forecasting and Reasoning

Zhensheng Wang<sup>1,2</sup>, Xiaole Liu<sup>3</sup>, Wenmian Yang<sup>2,†</sup>, Kun Zhou<sup>1,2</sup>, Yiquan Zhang<sup>2</sup>, Weijia Jia<sup>2,4,†</sup>

<sup>1</sup>School of Artificial Intelligence, Beijing Normal University, Beijing, PR China

<sup>2</sup>Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai, PR China

<sup>3</sup>Faculty of Arts and Sciences, Beijing Normal University, Zhuhai, PR China

<sup>4</sup>Beijing Normal-Hong Kong Baptist University, Zhuhai, PR China

{jensenwang, xiaoleliu, zhoukun}@mail.bnu.edu.cn, {wenmianyang, jiawj}@bnu.edu.cn, zhangyq987@hotmail.com

## Abstract

The rapid development of LLMs has significantly advanced tabular question answering, but most systems cannot perform future-oriented numerical prediction. To address this gap, we introduce a novel task, Open-Domain Tabular Question Answering for Future Data Forecasting and Reasoning, and propose the first dataset to cover time-series forecasting and forecast-based reasoning scenarios using real estate data. This task poses challenges in retrieving precise historical data, overcoming the forecasting limitations of LLMs, and standardizing responses for diverse queries. To solve the above challenges, we propose TimeFore, an LLM agent-based framework that decomposes the problem into three collaborative roles: a Retriever autonomously generates SQL to fetch data, a Forecaster invokes external time-series models for higher accuracy, and an Analyzer synthesizes the results to construct a precise and consistent final answer. Extensive experiments demonstrate the effectiveness of our TimeFore. The dataset and code are available at <https://github.com/jensenw1/ODTQA-FoRe>.

## 1 Introduction

With the rapid development of large language models (LLMs), question-answering systems built upon these models have achieved remarkable progress across a wide range of QA tasks (Wang et al., 2023). In particular, the integration of Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) techniques has demonstrated powerful cross-task generalization capabilities, creating new opportunities and breakthroughs for open-domain QA systems (Chen et al., 2017; Kong et al., 2024; Chen et al., 2025a). As user demands evolve, there is growing expectation for LLMs not only to accurately answer traditional knowledge-based ques-

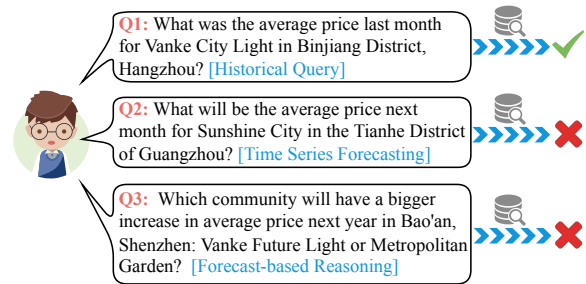


Figure 1: Example questions of future data forecasting and reasoning.

tions, but also to exhibit enhanced numerical reasoning abilities (Wang et al., 2025a; Zhang et al., 2025). As a result, considerable research attention has shifted toward open-domain numerical reasoning over tables, a direction commonly known as open-domain tabular question answering (ODTQA) (Herzig et al., 2021; Kweon et al., 2023). ODTQA methods typically leverage RAG techniques, as well as the models’ abilities in program synthesis and database query generation. These methods have proven particularly effective for numerical reasoning in vertical domains including real estate, healthcare, and finance.

However, current open-domain tabular QA studies predominantly focus on retrieval and reasoning based on data from static databases, with limited attention to questions requiring future data predictions, as shown in Figure 1. In practice, users frequently ask questions related to the forecasting and reasoning of future data, such as forecasting housing prices in the coming months (Figure 1, Q2) or comparing price trends across different communities (Figure 1, Q3), to inform investment and financial decision-making. While existing methods face significant challenges in addressing these queries for two main reasons. First, LLMs are primarily trained on static textual data and lack dedicated mechanisms for modeling dynamic time-series data, which limits their ability to make ef-

<sup>†</sup>Corresponding authors.

fective future predictions (Tan et al., 2024; Ye et al., 2025). Second, in open-domain scenarios, users typically cannot provide continuous and accurate historical data to support forecasting. Consequently, systems must autonomously generate query statements (e.g., SQL) to retrieve relevant historical data from large databases for analysis and prediction.

To address these issues, we propose a novel research task, namely Open-Domain Tabular Question Answering for Future Data Forecasting and Reasoning (ODTQA-FoRe), aiming to expand the scope of LLM-based QA systems into future data prediction. This task requires the QA system to possess cross-domain data retrieval capabilities, future-oriented time series forecasting skills, and comprehensive table data understanding and reasoning abilities to meet deeper user needs. However, this research direction is still in its infancy, lacking relevant datasets and benchmark frameworks, which restricts effective algorithm development and evaluation. To bridge this gap, we introduce the first Open-Domain Tabular Question Answering Dataset, ODTQA-FoRe, constructed using real estate data and incorporating both time series forecasting and forecast-based reasoning questions, thus addressing the absence of relevant research resources in this area.

Specifically, ODTQA-FoRe comprises 28,507 QA pairs constructed from real estate transaction records spanning 2022–2024 across 10 Chinese cities. Following the open-domain paradigm, the system must autonomously identify and retrieve relevant tables from a candidate pool of 288 tables (averaging 845 rows each), rather than relying on a pre-specified target table. We restrict the dataset to the real estate vertical to ensure dense and continuous temporal data for reliable forecasting evaluation. Despite this vertical focus, the task formulation remains generic, rendering the proposed framework readily extensible to other domains. The dataset contains 8,042 time-series forecasting questions and 20,465 forecast-based reasoning questions, posing substantial challenges for QA systems.

This new task presents three distinct challenges: First, the challenge of historical data retrieval, where systems must accurately identify relevant tables from massive databases and retrieve pertinent historical data sequences based on user queries, demanding significantly higher retrieval precision compared to traditional tabular QA tasks. Second,

the challenge of forecasting future data from historical sequences, as LLMs inherently exhibit limited predictive abilities, necessitating the development of methods to overcome these prediction limitations. Third, the challenge of problem type identification and standardized response generation, as users pose diverse question types (e.g., time series forecasting and forecast-based reasoning), each requiring different response formats and prompting methods. Thus, it is crucial for the system to accurately distinguish between question types and generate standardized responses.

To address the challenges of ODTQA-FoRe, we present TimeFore, an LLM agent-based framework that aims to establish a foundational baseline in this unique area. The framework decomposes the problem into three collaborative roles: a Retriever that tackles the data acquisition challenge by identifying relevant tables and autonomously generating SQL queries to fetch historical data; a Forecaster that addresses the LLM’s inherent predictive weaknesses by invoking external time-series models to enhance forecasting accuracy; and an Analyzer that synthesizes the retrieved data and forecasts into a standardized response format. As ODTQA-FoRe is our newly proposed benchmark without directly comparable baselines, TimeFore aims to set a performance standard for ODTQA-FoRe, further underscoring its importance in advancing this field.

The main contributions of this paper are as follows:

- We define the novel task of Open-Domain Tabular Question Answering for Future Data Forecasting and Reasoning. To support this, we construct a new benchmark dataset using real-world real estate data, incorporating diverse question types such as forecasting and forecast-based reasoning, filling a critical gap in the field.
- We propose TimeFore as the benchmark framework, effectively addressing the key challenges of historical data retrieval, future data prediction, and question type distinction and standardized response generation through its three dedicated agents.
- We conduct comprehensive experiments to validate TimeFore’s effectiveness and establish strong performance benchmarks across five mainstream LLMs. Furthermore, our detailed ablation studies dissect the framework’s

Dataset	Open Domain	# of QA pairs	# Tables	Answer format	Multi-table	TS forecasting
WikiTableQuestion	✗	22033	2108	Text	✗	✗
Spider	✗	10181	1020	SQL	✓	✗
Open-WikiTable	✓	67023	24680	Text,SQL	✗	✗
NQ-TABLES	✓	11628	169898	Text	✗	✗
RETQA	✓	20762	4932	Text,SQL	✓	✗
ODTQA-FoRe	✓	28507	288	Text,SQL	✓	✓

Table 1: Comparison with existing datasets. “Open Domain” indicates table retrieval from a large corpus, whereas “Closed Domain” (marked with ‘✗’) specifies QA on given tables. “TS forecasting” refers to time-series forecasting.

components, identify key performance bottlenecks, and provide insights for future improvements.

## 2 Related Works

### 2.1 TQA Datasets

Early table question answering (TQA) datasets, such as WikiTableQuestions (Pasupat and Liang, 2015), focused on complex queries but provided textual answers only. Spider (Yu et al., 2018), while evaluating generalization on complex SQL queries, relies on structured outputs, limiting its open-domain applicability. These datasets are primarily domain-specific.

Recent advancements extend TQA to open-domain scenarios. NQ-TABLES (Herzig et al., 2021) introduces candidate table retrieval, and Open-WikiTable (Kweon et al., 2023) enhances this with metadata. RETQA (Wang et al., 2025b) explores open-domain TQA in real estate, leveraging spoken language understanding (SLU) annotations (Xing et al., 2025; Qin et al., 2025) for improved retrieval. FutureX (Zeng et al., 2025b) addresses future-oriented QA, focusing on qualitative reasoning rather than structured data. In contrast, ODTQA-FoRe targets precise time-series forecasting over structured tables, a largely overlooked area in open-domain TQA.

### 2.2 TQA Methods

Recent approaches leverage large language models (LLMs) for TQA tasks (Fang et al., 2024). OPENTAB (Kong et al., 2024) integrates open-domain knowledge for table-based fact-checking, while SLUTQA (Wang et al., 2025b) enhances retrieval and SQL generation accuracy through SLU (Cheng et al., 2023). However, these methods mainly focus on historical query answering rather than future predictions.

Independent applications of LLMs for forecasting include TP-BERTa (Yan et al., 2024), which

discretizes numerical features but does not handle sequential predictions, and LLMTIME (Gruver et al., 2023a), which employs pretrained LLMs for time series without TQA integration.

Thus, existing methods predominantly address historical queries, non-sequential forecasting, or time series predictions, lacking a unifying framework for forecasting-oriented TQA. Our work fills this gap, representing the first effort in Open-Domain Tabular Question Answering for Future Data Forecasting and Reasoning.

## 3 Dataset Construction and Analysis

### 3.1 Data Source

The dataset for this study originates from the RETQA dataset (Wang et al., 2025b), which covers land auctions, real estate project sales, and corporate finance. Our work focuses on the real estate sales data, initially spanning eight major Chinese cities for the year 2022. We significantly expand this by leveraging the original data source (<http://www.fangdi.com.cn/>) to include two additional cities, Tianjin and Chengdu, and extend the timeframe to cover January 2022 to December 2024. This results in a comprehensive dataset of 60,183 real estate projects, each with up to 36 months of data.

To prepare the data for our task, we first define a temporal split: December 31, 2023, serves as the reference date, with data from 2022–2023 designated as historical observations and data from 2024 treated as future ground truth, which is strictly withheld during model inference. Given the inherent sparsity in real estate records, we then apply stringent filtering criteria, retaining only projects with at least six months of sales data in the historical period (2022–2023) and at least one record in the forecast period (2024). After applying these criteria, we obtain a refined dataset of 11,149 projects for a subsequent QA pair generation.

To prevent data leakage and ensure a fair evalua-

tion, we first partition the 11,149 filtered projects into training, validation, and test subsets. This project-level split is performed using a 6:2:2 ratio, resulting in 6,562 projects for the training set, 2,278 for the validation set, and 2,309 for the test set. This partitioning guarantees that there is no overlap of projects between the sets. The QA pairs for each split are subsequently generated exclusively from their corresponding project subsets.

The data corresponding to all 11,149 refined projects is then used to construct the database environments. Since the original RETQA data source is in Chinese, this process primarily involves data cleaning and localization, where all Chinese column headers, city names, district names, and project names are translated into English. The protocols for translation and schema standardization are detailed in Appendix A.1. After this unified preparation, the dataset is split by time. The historical data (2022–2023) is aggregated by city, district, and year into 288 tables. These tables are then imported into a PostgreSQL (Stonebraker et al., 2019) database to serve as the agent’s queryable knowledge base, each renamed following the format `project_price_table_{district}_{city}_{year}` (e.g., `project_price_table_zhengcheng_district_guangzhou_2022`). Separately, the future data (2024) is organized into 144 tables using the same aggregation and naming convention and stored in an isolated database named “future prices” to facilitate the automated execution of ground truth SQL queries during evaluation (see Appendix A.2 for details). Note that throughout these databases, tables do not distinguish between projects from the training, validation, or test sets, as the agent cannot access the full database contents and can only interact with it through a text-to-SQL interface.

### 3.2 Template Design

To support both time series forecasting and forecast-based reasoning tasks, we design 26 distinct sets of seed templates, comprising 7 templates specifically for forecasting tasks and 19 for reasoning tasks. Each set includes a natural language question template, a historical data SQL template, and a target (future) data SQL template. These templates primarily incorporate four variables: city, district, project name, and time. City, district, and project name variables are populated by randomly sampling from the database, while the time variable

is determined based on actual data availability for the respective projects in 2024, ensuring alignment with months that have valid price records.

The forecasting templates generate queries requiring predictions of specific statistical indicators (e.g., price, averages, or extreme values) over pre-defined periods (see Figure 1, Q2). Conversely, reasoning templates address more complex analytical tasks, including comparisons of future price trends among multiple projects or identification of projects satisfying specific criteria (see Figure 1, Q3).

Prior to large-scale generation, we initially create ten QA pairs per template set, totaling 260 pairs. These pairs undergo rigorous manual review to ensure grammatical fluency, semantic clarity, correctness, and efficiency of SQL queries, as well as accuracy and conciseness of results. Any identified issues prompt iterative revisions until all QA pairs pass validation. Subsequently, large-scale automated QA pair generation proceeds. For more details on the types of templates, please refer to Appendix A.2.

### 3.3 QA Pair Generation

The final ODTQA-FoRe dataset includes, for each QA instance: a natural language question, either a natural language answer (for reasoning queries) or a numerical answer (for forecasting queries), an SQL query to retrieve historical data, and an SQL query for retrieving future data labels. Historical data query results provide necessary context, while future data query results serve as evaluation labels. Answers are objective, uniquely determined by their corresponding questions. For numerical prediction, we provide only directly relevant values; for reasoning, only the direct conclusion. Multiple values or conclusions, if applicable, are presented as structured lists. Examples and detailed answer formats are provided in Appendix A.2. All four components (question, answer, historical data SQL, and future data SQL) are generated simultaneously, ensuring precise alignment.

As detailed in Section 3.2, each template utilizes four primary variables: city, district, project name, and time. During QA generation, cities and districts are randomly sampled first, followed by selecting project names from their corresponding dataset splits (training, validation, or test).

To handle temporal references, we create 98 natural language expressions denoting future periods (e.g., “next month,” “next year,” “next quarter,”

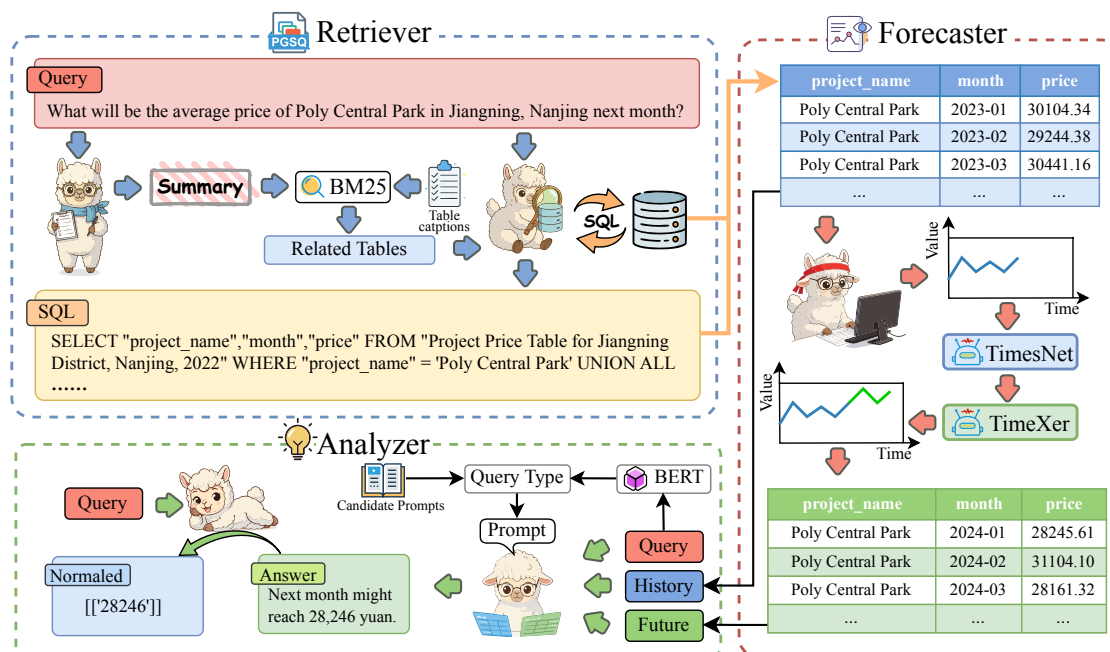


Figure 2: General framework of TimeFore.

“first half of the year”), each mapped to precise month lists (e.g., “next quarter” corresponds to [January 2024, February 2024, March 2024]). For each project, we identify all months in 2024 with available records, selecting appropriate expressions fully covered by the data. For instance, “next quarter” is used only if data for January to March 2024 is fully available. Subsequently, one eligible expression is randomly selected for template filling. Historical data SQL queries cover all months from 2022–2023, while future data SQL queries explicitly include months in 2024 with available records.

Using this template-driven approach, we initially generate 36,400 QA pairs. After removing duplicates, invalid queries, and empty results, we finalize 28,507 unique QA pairs, divided according to project assignments into 16,944 training pairs, 5,742 validation pairs, and 5,821 testing pairs. Additionally, following previous practices (Wang et al., 2025b), we employ LLMs to rewrite original questions enhancing dataset diversity and realism. Comprehensive details on the rewriting procedure and dataset statistics are provided in Appendix A.3 and A.4.

## 4 Method

### 4.1 Overview

In this section, we present TimeFore, a collaborative framework designed for Open-Domain Tabular Question Answering involving Future Data

Forecasting and Reasoning (ODTQA-FoRe). As illustrated in Figure 2, TimeFore decomposes the task across three specialized agents. The Retriever handles data acquisition by first identifying relevant data tables and then generating a SQL query to extract historical information. Subsequently, the Forecaster leverages function-calling to invoke external, specialized models for accurate time-series prediction. Finally, the Analyzer synthesizes the retrieved data with the generated forecast to construct a precise and consistently formatted final answer.

### 4.2 Retriever

In open-domain scenarios, the performance of accurately retrieving relevant data from a vast repository based on a user’s query is paramount, as this step directly governs the quality of all downstream tasks. To address this, the Retriever agent executes a two-stage process. The process commences with table retrieval, where the agent leverages an LLM’s summarization capability by employing few-shot prompting with five examples to convert the user query into a concise, canonical text in the style of a table caption. This text is first used to attempt a direct match with existing captions in the database. Should this attempt fail, the BM25 algorithm (Robertson and Walker, 1994) is then employed to retrieve the most semantically relevant table.

Upon securing the target table, the agent proceeds to generate the appropriate SQL query. This

is again accomplished through in-context learning, guided by a system prompt containing five distinct exemplars. Each exemplar consists of a natural language question, its corresponding ground-truth table caption, and the target SQL query. To ensure robustness, these examples are sampled to cover diverse query types (e.g., single and multi-item). The LLM is prompted with the user’s original question and the identified table caption to generate an initial SQL query. To ensure robustness and address potential syntax or logic errors, particularly in complex scenarios, the Retriever agent employs an execution-feedback loop, utilizing the *sqlQueryTool* to execute the SQL within the database. If execution fails, the agent refines the SQL based on the error feedback. This loop continues until success is achieved or a maximum of 25 iterations is reached, ensuring reliable data retrieval. A detailed illustration of the prompt design is available in Appendix E.

### 4.3 Forecaster

Upon receiving the SQL results, the Forecaster interprets the serialized triples formatted as [project name, year-month, price] and converts them into a numerical list for the forecasting tools (Liang et al., 2025). This historical input, limited to a 24-month window, remains compact enough to fit within the LLM’s context limits, thus avoiding token overflow issues and ensuring smooth data integration for forecasting.

While LLMs possess extensive general knowledge, their native forecasting capabilities for time-series tasks often lag behind specialized models without task-specific fine-tuning (Gruver et al., 2023b; Tan et al., 2024). The Forecaster agent is designed to bridge this gap. Consistent with the TimeFore framework’s strategy, it leverages the LLM’s function-calling capability to orchestrate specialized, high-performance time-series models, thereby addressing the LLM’s inherent predictive weaknesses.

The process begins once the Forecaster receives the historical data sequences from the Retriever. Its core action is to invoke a predefined function, the *imputationThenPredictionTool*. This function encapsulates a sophisticated forecasting pipeline: it first validates and cleans the input time series, then employs the TimesNet model (Wu et al., 2023) for robust data imputation. Based on this complete and imputed sequence, the TimeXer model (Wang et al., 2024b) is then used to generate a

price trend forecast for the subsequent 12 months. The tool concludes by returning the results as a structured three-tuple, containing the project name, year-month, and predicted price.

This architecture creates a powerful synergy: the LLM excels at orchestrating the workflow and understanding the task context, while the specialized models handle the precise, mathematical heavy-lifting of forecasting. This division of labor significantly enhances the overall forecasting accuracy and reliability.

For operational simplicity, the Forecaster is configured to generate a full 12-month forecast for the year 2024. This complete forecast is then passed to the Analyzer agent, which is responsible for reasoning upon these results to extract and deliver the specific answers pertinent to the user’s original query.

### 4.4 Analyzer

The Analyzer is the final-stage agent in the TimeFore framework, responsible for synthesizing the inputs from the preceding agents into a precise, standardized, and user-centric final answer. Its primary function is to correctly interpret the user’s ultimate goal and apply the appropriate reasoning strategy. To this end, the Analyzer first employs a BERT-based classifier to categorize the user’s query as either direct time-series forecasting (e.g., "What will the price be in June 2024?") or forecast-based reasoning (e.g., "Will the price in June exceed the price in January?"). Based on this classification, a distinct and tailored prompt, enriched with five illustrative examples via in-context learning, is selected to guide the LLM’s response generation process.

To facilitate LLM numerical reasoning, historical and forecast data are formatted as key-value pairs, each entry including a date and its corresponding average price (e.g., "[[August 2022, 10,890], [December 2022, 11,585]]"). If historical records for specific months are unavailable, these months are simply omitted from the provided data.

Despite its effectiveness in handling forecast-based reasoning queries, LLMs often provide verbose explanations rather than succinct numerical predictions for straightforward forecasting questions (Chen et al., 2025b). To mitigate this, we introduce an LLM-driven numerical extraction module, which employs in-context learning with five examples, explicitly prompting the model to return only the essential numerical results. All detailed

prompts utilized within the analysis agent are comprehensively documented in Appendix E.

## 5 Experiments

Model	MSE	MAE	MRE
TimesNet	2.77E+07	3103.52	0.1254
TimeMixer	2.78E+07	3108.29	0.1255
TimeXer	<b>2.50E+07</b>	<b>2989.55</b>	<b>0.1209</b>
WPMixer	2.75E+07	3097.81	0.1248
AutoTimes	2.93E+07	3204.13	0.1288
Time-MoE	2.95E+07	3164.47	0.1271
Qwen3 30B	6.69E+07	4344.02	0.1706
GLM 4.5 Air	7.30E+07	4824.57	0.1869

Table 2: Performance comparison of different models on time series forecasting tasks.

### 5.1 Experimental Settings and Baselines

ODTQA-FoRe consists of two subtasks: time-series forecasting and forecast-based reasoning. We evaluate time-series forecasting using Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Relative Error (MRE). For forecast-based reasoning, we use Accuracy, Precision (P), Recall (R), and F1-score (F1). To account for questions with multiple items, we compute F1 at the item level for partial credit, whereas Accuracy requires an exact match of all items within a single question.

**Data Preprocessing and Model Selection.** Real estate transaction data is often sparse, with information gaps for months without sales. To address this, we first perform data imputation. Our comparison of three leading imputation models, i.e., TimesNet (Wu et al., 2023), i-Transformer (Liu et al., 2024a), and TimeMixer (Wang et al., 2024a), shows that TimesNet achieves superior performance, demonstrating the strongest imputation capability (see Appendix C.3 for detailed results).

For the core time-series forecasting task, we evaluate eight models to select the optimal forecaster for our TimeFore framework. These include four lightweight models (TimesNet, TimeMixer, TimeXer (Wang et al., 2024b), WPMixer (Murad et al., 2025)), two LLM-based time-series models (AutoTimes (Liu et al., 2024b), Time-MoE (Shi et al., 2025)), and two general-purpose LLMs (Qwen3 30B, GLM 4.5 Air). The general-purpose LLMs perform forecasting via in-context learning (ICL) on the original data, while the six specialized

models are trained on data imputed by TimesNet.

As Table 2 shows, the specialized time-series models significantly outperform the general-purpose LLMs, which confirms the latter’s relative weakness in direct prediction tasks. Among all candidates, TimeXer achieves the best performance and is therefore selected as the forecasting component within the TimeFore framework.

**Baselines.** To validate the effectiveness of TimeFore, we compare it against five strong baseline models: two from the Qwen3 series (Yang et al., 2025) (Qwen3 30B A3B, and Qwen3 Next 80B A3B Thinking), two from the GPT-OSS series (Agarwal et al., 2025) (GPT-OSS 20B and GPT-OSS 120B), and GLM 4.5 Air (Zeng et al., 2025a).

Further details on experimental setups—including retrieval agent performance, training data specifics, and other implementation details—are provided in Appendix C due to space constraints.

### 5.2 Result and Analysis

#### Main Results

As ODTQA-FoRe is a newly proposed benchmark without directly comparable baselines, we introduce a *Vanilla* baseline, which first uses the BM25 algorithm to retrieve relevant table captions, then constructs a five-shot prompt for in-context learning (ICL), and finally relies entirely on the LLM’s intrinsic capabilities to predict future data.

As presented in Table 3, the experimental results demonstrate that our TimeFore framework significantly outperforms the vanilla baseline across all models. This outcome highlights the inherent limitations of general-purpose LLMs when applied directly to specialized time-series forecasting tasks.

These results collectively demonstrate that delegating forecasting to a specialized time-series model, rather than relying on the LLM itself, not only enhances prediction accuracy but also significantly improves performance on subsequent reasoning tasks. This finding offers a key insight for the future development of LLM-based predictive and reasoning systems: a hybrid approach that leverages specialized tools for their respective strengths yields a more robust and accurate overall framework.

### 5.3 Ablation Study

To identify performance bottlenecks and quantify the contribution of each component within our

Model	Method	Time-series Forecasting			Forecast-based Reasoning			
		MSE	MAE	MRE	Acc	P	R	F1
Qwen3 30B	Vanilla	40385720.95	3698.36	0.1627	12.19	29.01	21.76	24.87
	TimeFore	<b>31572410.36</b>	<b>2788.20</b>	<b>0.1326</b>	<b>31.59</b>	<b>61.78</b>	<b>58.80</b>	<b>60.25</b>
Qwen3 Next 80B	Vanilla	30942598.21	3406.43	0.1586	24.45	48.40	44.96	46.62
	TimeFore	<b>22442845.96</b>	<b>2588.87</b>	<b>0.1181</b>	<b>36.31</b>	<b>55.86</b>	<b>61.21</b>	<b>58.41</b>
GPT OSS 20B	Vanilla	105115117.20	4394.56	0.1838	21.52	46.79	41.98	44.25
	TimeFore	<b>29757828.58</b>	<b>2887.44</b>	<b>0.1280</b>	<b>27.80</b>	<b>53.61</b>	<b>45.31</b>	<b>49.11</b>
GPT OSS 120B	Vanilla	47324931.58	3786.48	0.1683	21.60	44.44	38.16	41.06
	TimeFore	<b>18493634.83</b>	<b>2501.18</b>	<b>0.1151</b>	<b>31.37</b>	<b>57.35</b>	<b>47.32</b>	<b>51.86</b>
GLM4.5 Air	Vanilla	139922250.13	3324.41	0.1415	23.59	49.82	47.67	48.72
	TimeFore	<b>90865440.59</b>	<b>2709.25</b>	<b>0.1172</b>	<b>35.46</b>	<b>63.73</b>	<b>58.93</b>	<b>61.24</b>

Table 3: Overall performance on the ODTQA-FoRe dataset.

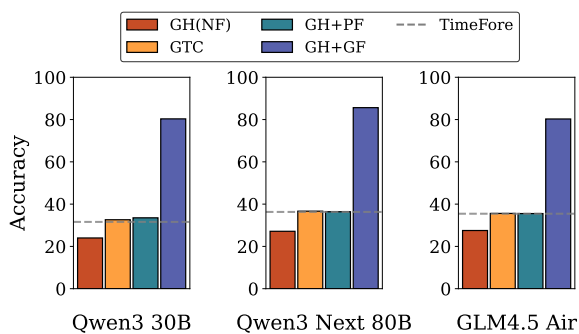


Figure 3: Ablation study on the TimeFore framework in forecast-based reasoning tasks.

multi-step TimeFore pipeline, we conduct a comprehensive ablation study. For the forecast-based reasoning task, we systematically substitute individual model-generated outputs with their ground-truth counterparts and observe the resulting performance gains. This allows us to isolate the impact of table retrieval, data retrieval (SQL generation), and time-series forecasting.

The study is designed around four specific configurations. The first, denoted as GH (NF) for Golden History (No Future), serves as a baseline where the LLM receives only ground-truth historical data and no future data. This measures the model’s reasoning ability without any forecasting input. The second configuration, +GTC, provides ground-truth table captions to the Retriever to isolate the impact of the initial table retrieval step, while the rest of the pipeline remains unchanged. The third setup, GH + PF (Golden History + Predicted Future), bypasses the Retriever by providing ground-truth historical data directly but still requires the Forecaster to predict future values, thereby isolating the performance impact of the

SQL generation stage. The final configuration, GH + GF (Golden History + Golden Future), establishes the theoretical upper bound of the LLM’s reasoning capability by providing the Analyzer with perfect historical and future data, bypassing both the Retriever and the Forecaster.

The results of this study, visualized in Figure 3, reveal a clear performance hierarchy. The most significant performance drop occurs in the GH (NF) setting, confirming that reasoning without forecasting is largely ineffective. While providing golden table captions (+GTC) or golden historical data (GH + PF) yields only marginal improvements, the most substantial performance gain by far is achieved in the GH + GF setting when perfect future data is supplied.

This unequivocally demonstrates that the accuracy of the time-series forecasting component is the primary bottleneck for the entire framework. Accurate numerical prediction is not just beneficial; it is a prerequisite for enabling the large model to perform reliable downstream reasoning.

More detailed ablation study for Analyzer’s numerical extraction module are available in Appendix C.5.

## 6 Conclusion

In this paper, we introduce a novel task—Open-Domain Tabular Question Answering for Future Data Forecasting and Reasoning—and present a pioneering dataset constructed using real estate data, addressing critical gaps in current research. To support this task, we introduce TimeFore, a comprehensive benchmark framework capable of effectively retrieving historical data, performing accurate future data predictions, and providing standard-

ized responses. Our extensive experiments establish baseline performance across multiple LLMs and highlight both the task’s inherent challenges and opportunities, laying the groundwork for future advancements in this field.

## Limitations

While our study demonstrates the effectiveness of the proposed approach, certain limitations remain that offer opportunities for future improvement.

First, regarding the forecasting mechanism, we employ TimesNet for data imputation and TimeXer for prediction. While this pipeline is effective, it has not yet incorporated a broader global context, such as regional macroeconomic statistics, environmental factors, or relevant policy information. In future work, we plan to explore models that can better integrate such exogenous variables, especially in scenarios where external factors drive significant market shifts.

Second, regarding domain coverage, the ODTQA-FoRe dataset currently focuses exclusively on the real estate domain. While the TimeFore framework is designed to be domain-agnostic, validating its generalizability across other vertical domains (e.g., finance, retail, or climate) remains a key direction. Specifically, the adaptability of the forecasting backbone (TimeXer) across diverse data characteristics with varying volatility and periodicity warrants further exploration.

Third, regarding dataset construction, while the inclusion of diverse templates may not fully encapsulate the complexity of all real-world queries, the dataset remains sufficiently comprehensive to evaluate the proposed framework. We adopted a template-based generation approach, a common practice in large-scale QA benchmarks, to ensure the availability of deterministic answers. Furthermore, to enhance linguistic diversity, we employed LLMs to rewrite the queries and conducted a human evaluation for validation. The results demonstrate that the rewritten expressions are significantly closer to natural human language than the original template-based formulations.

## Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62272050 and the grant of Beijing Normal- Hong Kong Baptist University sponsored by Guangdong Provincial Department of Educa-

tion; in part by Zhuhai Science-Tech Innovation Bureau under Grant No. 2320004002772 and the Interdisciplinary Intelligence Super Computer Center of Beijing Normal University (Zhuhai).

## References

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). *Preprint*, arXiv:2508.10925.
- Andong Chen, Kehai Chen, Yang Xiang, Xuefeng Bai, Muyun Yang, Yang Feng, Tiejun Zhao, and Min Zhang. 2025a. [LLM-based translation inference with iterative bilingual understanding](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16886–16902, Vienna, Austria. Association for Computational Linguistics.
- Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. [Reading wikipedia to answer open-domain questions](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025b. [Towards reasoning era: A survey of long chain-of-thought for reasoning large language models](#). *CoRR*, abs/2503.09567.
- Ziyang Chen, Jinzhi Liao, and Xiang Zhao. 2023. [Multi-granularity temporal question answering over knowledge graphs](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 11378–11392. Association for Computational Linguistics.
- An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. 2024. [Spatialrgpt: Grounded spatial reasoning in vision-language models](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Lizhi Cheng, Wenmian Yang, and Weijia Jia. 2023. [A scope sensitive and result attentive model for multi-intent spoken language understanding](#). In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 12691–12699. AAAI Press.

- Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. 2024. [Large language models \(llms\) on tabular data: Prediction, generation, and understanding — a survey](#). *Transactions on Machine Learning Research*.
- Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023a. [Large language models are zero-shot time series forecasters](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023b. [Large language models are zero-shot time series forecasters](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. [Open domain question answering over tables via dense retrieval](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 512–519. Association for Computational Linguistics.
- Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. [Opentab: Advancing large language models as open-domain table reasoners](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. 2023. [Open-wikitable : Dataset for open domain question answering with complex reasoning over table](#). In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8285–8297. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Xiaobo Liang, Wenjin Xie, Juntao Li, Wanfu Wang, Yibin Chen, Kehai Chen, and Min Zhang. 2025. [Tool learning via inference-time scaling and cycle verifier](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 24658–24671. Association for Computational Linguistics.
- Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024a. [itransformer: Inverted transformers are effective for time series forecasting](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. 2024b. [Autotimes: Autoregressive time series forecasters via large language models](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Md Mahmuddun Nabi Murad, Mehmet Aktukmak, and Yasin Yilmaz. 2025. [Wp mixer: Efficient multi-resolution mixing for long-term time series forecasting](#). In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 19581–19588. AAAI Press.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.
- Libo Qin, Qiguang Chen, Jingxuan Zhou, Jin Wang, Hao Fei, Wanxiang Che, and Min Li. 2025. [Divide-solve-combine: An interpretable and accurate prompting framework for zero-shot multi-intent detection](#). In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 25038–25046. AAAI Press.
- Stephen E Robertson and Steve Walker. 1994. [Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval](#). In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.
- Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. 2025. [Time-moe: Billion-scale time series foundation models with mixture of experts](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Michael Stonebraker, Lawrence A. Rowe, and Michael Hirohama. 2019. [The implementation of POSTGRES](#). In Michael L. Brodie, editor, *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker*, volume 22 of *ACM Books*, pages 519–559. ACM / Morgan & Claypool.
- Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. 2024. [Are language models actually useful for time series forecasting?](#)

- In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Cunxiang Wang, Sirui Cheng, Qipeng Guo, Yuanhao Yue, Bowen Ding, Zhikun Xu, Yidong Wang, Xiangkun Hu, Zheng Zhang, and Yue Zhang. 2023. [Evaluating open-qa evaluation](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. 2024a. [Timemixer: Decomposable multiscale mixing for time series forecasting](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Yuxiang Wang, Jianzhong Qi, and Junhao Gan. 2025a. [Accurate and regret-aware numerical problem solver for tabular question answering](#). In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA, pages 12775–12783*. AAAI Press.
- Yuxuan Wang, Haixu Wu, Jiayang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. 2024b. [Timexer: Empowering transformers for time series forecasting with exogenous variables](#). *Advances in Neural Information Processing Systems*, 37:469–498.
- Zhensheng Wang, Wenmian Yang, Kun Zhou, Yiquan Zhang, and Weijia Jia. 2025b. [RETQA: A large-scale open-domain tabular question answering dataset for real estate sector](#). In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA, pages 25452–25460*. AAAI Press.
- Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. [Timesnet: Temporal 2d-variation modeling for general time series analysis](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Bowen Xing, Libo Qin, Zhihong Zhu, Zhou Yu, and Ivor W Tsang. 2025. [Dxa-net: Dual-task cross-lingual alignment network for zero-shot cross-lingual spoken language understanding](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Jiahuan Yan, Bo Zheng, Hongxia Xu, Yiheng Zhu, Danny Z. Chen, Jimeng Sun, Jian Wu, and Jintai Chen. 2024. [Making pre-trained language models great on tabular prediction](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- Linhao Ye, Lang Yu, Zhikai Lei, Qin Chen, Jie Zhou, and Liang He. 2025. [Optimizing question semantic space for dynamic retrieval-augmented multi-hop question answering](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17814–17824, Vienna, Austria. Association for Computational Linguistics.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. 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, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.
- Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, and 80 others. 2025a. [GLM-4.5: agentic, reasoning, and coding \(ARC\) foundation models](#). *CoRR*, abs/2508.06471.
- Zhiyuan Zeng, Jiashuo Liu, Siyuan Chen, Tianci He, Yali Liao, Jinpeng Wang, Zaiyuan Wang, Yang Yang, Lingyue Yin, Mingren Yin, Zhenwei Zhu, Tianle Cai, Zehui Chen, Jiecao Chen, Yantao Du, Xiang Gao, Jiacheng Guo, Liang Hu, Jianpeng Jiao, and 11 others. 2025b. [Futurex: An advanced live benchmark for LLM agents in future prediction](#). *CoRR*, abs/2508.11987.
- Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. [A survey of table reasoning with large language models](#). *Frontiers Comput. Sci.*, 19(9):199348.

## A Datasets

### A.1 Database Translation and Standardization

Since the original data source (RETQA) is in Chinese, a rigorous protocol for translation and normalization is essential to ensure compatibility with English-based LLMs and reproducibility of the SQL-based workflows. This process is divided into handling structured schema elements and entities.

**Schema Normalization.** First, we standardized the structured architecture elements, including column headers, city names, and district names, by

mapping them to unified English terms. This manually curated mapping ensures consistency across different tables. For instance, the column representing project names is standardized as “project\_name” in all tables, and district names use standard Pinyin or official English translations. This step is a prerequisite for the Retriever agent to generate valid and executable SQL queries without ambiguity.

**Entity Translation.** Translating real estate project names presents a greater challenge due to semantic variability and the lack of a standard dictionary. To address this while maintaining semantic fidelity and consistency, we employed an LLM-driven translation strategy. As illustrated in Figure 8, we designed a specialized prompt that constrains the model’s output to follow strict translation rules.

**Quality Assurance.** To address potential concerns regarding translation errors affecting downstream retrieval or reasoning, we implemented a validation step. Specifically, we conducted a manual review of a random sample of translated project names to verify adherence to the standardized rules. This validation confirmed that the LLM, guided by the prompt, successfully generated consistent English names that align with the database schema, thereby minimizing the risk of retrieval failure due to naming mismatches.

## A.2 Template Filling

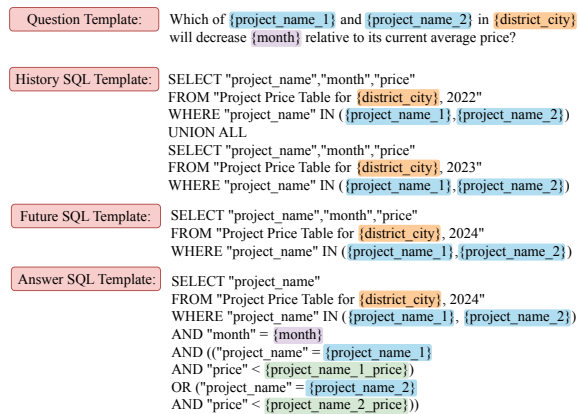


Figure 4: Examples of template filling.

Template filling is a widely adopted approach for dataset construction in question-answering research. This methodology offers two key advantages: first, it ensures that each generated question has a deterministic and verifiable answer, which is essential for rigorous evaluation; second, it mitigates human errors and subjective biases that may arise during crowd-sourced data collection. Previ-

ous works (Chen et al., 2023; Wang et al., 2025b; Cheng et al., 2024) have successfully employed template filling for dataset generation.

Based on 288 tables, we design 26 seed templates, as shown in Figure 4. These templates comprise 7 time series forecast seed templates (4 for single-project scenarios and 3 for multi-project scenarios) and 19 forecast-based reasoning seed templates (3 for single projects and 16 for multiple projects). Each seed template contains a Question Template, a History SQL Template, a Future SQL Template, and an Answer SQL Template. As illustrated in Figure 4, elements enclosed in “{ }” within the templates indicate variables to be populated, with some variables exhibiting a one-to-one correspondence across the four templates. After randomly sampling appropriate variable values from the database, these values are inserted into the corresponding positions in the templates.

In constructing this dataset, data from 2022 and 2023 serve as historical data, and the tables are organized on a yearly basis. Accordingly, the years in the History SQL Template are fixed to 2022 and 2023, while the year in both the Future SQL Template and the Answer SQL Template is set to 2024. Notably, certain variables highlighted with a green background in the Answer SQL Template (such as “project\_name\_1\_price” and “project\_name\_2\_price”) are determined dynamically during sampling. For example, when sampling a specific template, the “district\_city” variable is first set. Subsequently, all project names are extracted from the target table, from which eligible project names are randomly selected. If a question involves the “present,” the program automatically retrieves the current month’s price for the sampled project name(s) and assigns these actual values to “project\_name\_1\_price” and “project\_name\_2\_price”.

The question type is identified through fixed phrases embedded within the template. Once all variables have been populated, the SQL statement generated from the filled Answer SQL Template is executed to obtain the answer to the question. An example of a complete Question-Answer (QA) pair is provided in Figure 5.

## A.3 LLM-based Query Rewriting

Template-generated queries often exhibit monolithic syntactic structures, differing significantly from the natural language queries of real users. This discrepancy can lead to suboptimal perfor-

## Dataset Example

**Query:** What about the average price for Baoli Wutongyu in Wuqing, Tianjin, next month? Will it increase or decrease from the current price?  
**Query\_type:** "Forecast-based Reason"  
**Table\_caption:** ["Project Price Table for Wuqing District, Tianjin, 2022", "Project Price Table for Wuqing District, Tianjin, 2023"]  
**History\_SQL:** SELECT "month","price" FROM "Project Price Table for Wuqing District, Tianjin, 2022" WHERE "project\_name" = 'Baoli Wutongyu' UNION ALL SELECT "month","price" FROM "Project Price Table for Wuqing District, Tianjin, 2023" WHERE "project\_name" = 'Baoli Wutongyu';  
**Future\_SQL:** SELECT "month","price" FROM "Project Price Table for Wuqing District, Tianjin, 2024" WHERE "project\_name" = 'Baoli Wutongyu';  
**Answer\_SQL:** SELECT CASE WHEN ""price" > 9117.46 THEN 'Increase' WHEN ""price" = 9117.46 THEN 'Unchange' ELSE 'Decrease' END AS "price\_change" FROM "Project Price Table for Wuqing District, Tianjin, 2024" WHERE "project\_name" = 'Baoli Wutongyu' AND "month" = 'January 2024';  
**Answer:** [['Decrease']]

Figure 5: A sample of a QA pair.

mance when models trained on such template data process actual user queries.

To bridge the gap between template-generated rigidity and real-world linguistic diversity, we employ an LLM-based rewriting strategy. This is not merely a stylistic change; it aims to simulate the varied syntax and paraphrasing patterns found in natural user interactions, thereby reducing the structural bias inherent in fixed templates. This rewriting process, predicated on preserving the original query semantics, utilizes strategies such as synonym replacement and syntactic structure adjustments to generate queries with more diverse and natural linguistic expressions. Specifically, we utilized the API of the Qwen2.5-72B model for this query rewriting task, aiming to generate outputs that more closely align with human linguistic patterns. Details of the prompts employed for rewriting are provided in Appendix E Figure 9.

To evaluate the rewriting effect, we randomly

Statistics	Num
# Total Questions	28507
# Train	16944
# Validation	5742
# Test	5821
# Ave, Items per Question	2.3
# Ave, Length per Question	32.2
# Time Series Forecasting Questions	8042
# Numerical Reasoning Questions	20465
# Unique Tables	288
# Ave, Rows per Table	845.4

Table 4: Dataset statics

selected 50 original template queries and their corresponding LLM-rewritten versions. These 100 queries, after being randomly shuffled, were subjectively scored by seven volunteers based on their naturalness (1 to 5 points). The scoring criteria were explicit: the higher the score, the closer the sentence aligns with natural human language; conversely, a lower score indicates more pronounced template-generated characteristics. For each query, we removed the highest and lowest of the seven scores, then averaged the remaining five. We then calculated the mean score separately for the original template queries and the LLM-rewritten queries. The final average score for the original template queries was 3.324, while the LLM-rewritten queries achieved a final average score of 3.936. These results demonstrate that the LLM-rewritten queries more closely resemble natural human language expression.

Furthermore, we manually inspected a random subset of the rewritten queries to ensure semantic consistency with the original template intentions, confirming that the rewriting process improved naturalness without compromising the logical integrity of the questions.

## A.4 Dataset Statistics

The ODTQA-FoRe dataset comprises 288 tables and 28,507 question-answering pairs, offering a comprehensive benchmark for evaluating complex question-answering systems. Key characteristics include an average of approximately 2.3 query projects per question and an average of 845.4 rows per table. By question type, the dataset consists of 8,042 time-series forecasting questions (28.21%) and 20,465 numerical reasoning questions (71.79%). To ensure fair model development

and evaluation, the dataset is partitioned into training (16,944 entries), development (5,742 entries), and test sets (5,821 entries). Consequently, its diverse question types, the substantial volume of its tables, and the ambiguous nature of its temporal expressions establish ODTQA-FoRe as a challenging and comprehensive benchmark for Future Data Forecasting and Reasoning over open-domain scenarios. Detailed statistics are presented in Table 4.

## B Data Flow and Intermediate Representations

To clarify the exact mechanism of data passing between agents, we detail the serialization and constraints below:

**Data Serialization from Retriever to Forecaster:** The Retriever does not pass raw database rows directly. Instead, it formats the SQL execution results into a standardized list of textual triples: [project name, year-month, price]. This format is explicitly defined in the system prompt (see Figure 12).

**Tool Invocation by Forecaster:** The Forecaster’s LLM receives these textual triples and autonomously preprocesses them into a clean numerical list (history length = 24) to invoke the `imputationThenPredictionTool`. Missing values are represented as placeholders. The tool returns a 12-month forecast, which the agent formats back into the triple structure.

**Constraints:** Each query instance involves a fixed input size: 24 historical months + 12 forecast months + instructions. This results in a total token count significantly lower than the context window limits of the employed LLMs. Therefore, no chunking or truncation strategies are required for the time-series data.

## C Supplementary experiments

### C.1 Table retrieval

Traditional open-domain tabular question answering typically relies on BM25 to directly match target table captions with the input question. However, our approach first leverages the inductive reasoning and summarization capabilities of large language models (LLMs) to generate a table-caption summary based on the query; if this summary does not correspond to any caption in the database, we then apply BM25 to identify the most relevant table caption. This two-stage strategy not only addresses

the uncertainty associated with table count in multi-table scenarios, but also enhances the accuracy of table retrieval. In our dataset, historical queries tend to select the complete sequence of time steps, which results in a fixed number of target tables for each query (namely two, corresponding to two years). As shown in Table 5, our proposed two-stage method, i.e., summarization-then-BM25, significantly outperforms the traditional BM25 direct matching approach, thereby validating the effectiveness of our strategy in this specific scenario.

### C.2 SQL performance

To evaluate the performance of SQL generation in the Retriever module, we employ two metrics: Executable Code Ratio (ECR) and Execution Accuracy (EA) (Yu et al., 2018). ECR measures the proportion of generated code that can be executed, thus reflecting the model’s ability to produce runnable SQL statements. The EA metric represents the percentage of tasks for which the first generated code successfully passes the test case. In our study, since each sample is associated with a single test case designed to query tables, EA is equivalent to accuracy.

As shown in Table 6, Qwen3 30B achieves the highest ECR for generated SQL, while Qwen3 Next 80B obtains the highest EA score.

### C.3 Time series imputation

Specialized time-series models typically outperform general large language models in time-series forecasting; however, these models rely heavily on large-scale and standardized datasets. In practical scenarios such as real estate, severe data missingness is common, posing significant challenges for time-series analysis. To address this, we utilize the imputation functionality of the TimesNet model, allowing us to handle missing values and ensure the effectiveness of subsequent prediction tasks.

Notably, both the training of TimesNet for imputation and for forecasting require well-constructed and appropriate datasets. Therefore, building on the project sequence data introduced in Section 3.1, we further construct dedicated datasets for both time-series imputation and time-series forecasting to meet the needs of our study.

For the time-series imputation task, we start by selecting project-level sequences from the training, validation, and test sets that contain at least six months of historical records between 2022 and 2023, specifically targeting historical query data

Model	Method	P	R	F1
BM25		90.45	90.45	90.45
Qwen3 30B	Summary	94.60	94.71	94.65
	Summary+BM25	97.72	97.72	97.72
Qwen3 Next 80B	Summary	95.71	95.71	95.71
	Summary+BM25	97.63	97.63	97.63
GPT OSS 20B	Summary	96.04	96.04	96.04
	Summary+BM25	98.12	98.14	98.13
GPT OSS 120B	Summary	97.84	97.84	97.84
	Summary+BM25	99.21	99.21	99.21
GLM 4.5 AIR	Summary	94.19	94.19	94.19
	Summary+BM25	97.59	97.59	97.59

Table 5: Comparison of table retrieval performance.

Model	ECR	EA
Qwen3 30B	<b>99.85</b>	71.38
Qwen3 Next 80B	93.47	<b>85.72</b>
GPT OSS 20B	97.62	72.34
GPT OSS 120B	87.84	67.93
GLM 4.5 AIR	94.30	71.59

Table 6: Comparison of LLM-generated SQL execution results of TimeFore.

Model	MSE	MAE	MRE
TimesNet	<b>1.78E+07</b>	<b>2169.53</b>	<b>0.0728</b>
i-Transformer	2.30E+07	2720.74	0.0926
TimeMixer	1.85E+07	2234.00	0.0755

Table 7: Performance of three models for time series imputation task.

with missing values. The final imputation dataset consists of 8,418 training sequences, 2,815 validation sequences, and 2,853 test sequences.

Subsequently, for the forecasting task, we again use the same project divisions and select sequences with at least nine months of historical data during 2022–2023 and at least two additional months of data in 2024, creating a dataset fit for forecasting future trends. This results in a forecasting dataset comprising 5,806 training sequences, 1,975 validation sequences, and 1,963 test sequences.

Our experimental workflow proceeds as follows: We first train a TimesNet model for time-series imputation according to the official implementation, and apply the trained model to fill in missing values within the forecasting dataset (across all data partitions). We then retrain TimesNet on the now-complete forecasting dataset for the time-series forecasting task. Model performance is evaluated

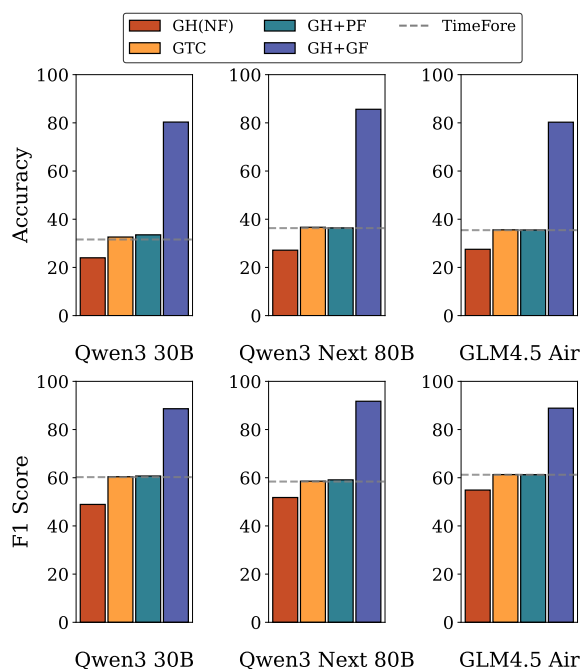


Figure 6: Ablation study on the TimeFore framework in forecast-based reasoning tasks.

on the test sets of both the imputation and forecasting datasets, as reported in Table 7. When fine-tuning time series forecasting models, we consistently use the optimal hyperparameter settings recommended in the original publications. For scenarios where LLMs perform time series forecasting via few-shot learning, we set the temperature parameter of the model to 0.8 uniformly to ensure consistent and controllable results.

#### C.4 Error Decomposition and Bottleneck Analysis

To precisely identify the source of errors within the TimeFore pipeline, we conduct a quantitative

decomposition based on our ablation study results and the performance of individual modules.

Specifically, we attribute the overall error to four main sources: (1) Table Retrieval, (2) Data Acquisition (SQL Generation), (3) Time-series Forecasting, and (4) Final Reasoning. We quantify the impact of each source by measuring the performance drop when the ground truth for that specific stage is replaced with the model’s output.

Based on Figure 6, we observe the following:

- **Data Retrieval is Robust:** The limited performance gains observed when providing golden table captions (GTC) or perfect historical data (GH+PF)—with maximum accuracy improvements of only approximately 1.02% and 1.93%, respectively—indicate that SQL generation errors are not the primary cause of overall system failures. This finding is further supported by the decoupling between SQL capability and final reasoning performance. For instance, while the Qwen3 Next 80B model achieves the highest SQL EA score (85.72%) as shown in Table 6, its Forecast-based Reasoning F1 score (58.41%) in Table 3 is lower than that of the Qwen3 30B model (60.25%), despite the latter’s inferior SQL performance. This discrepancy confirms that precise SQL generation does not necessarily translate to proportional gains in the end-to-end task, suggesting that data retrieval is a robust component and not the core bottleneck.
- **Forecasting is the Primary Bottleneck:** Quantitatively, replacing predicted future data with ground truth yields substantial accuracy gains: Qwen3-30B improves by 46.80%, Qwen3-Next 80B by 49.23%, and GLM 4.5 Air by 44.73%. These significant increases confirm that forecasting inaccuracies constitute the primary bottleneck, limiting even the strongest reasoning capabilities.
- **Retrieval and Reasoning:** While the Table Retrieval module achieves high F1 scores (above 97.59%, see Appendix Table 5), it still contributes a non-negligible portion of errors in open-domain scenarios. Furthermore, the gap between GH+GF and the theoretical upper bound (100%) reflects the limitations of the LLM in conducting complex numerical reasoning or formatting, which constitutes the smallest but still present error source.

Model	Method	P	R	F1
BERT	FT	<b>99.96</b>	<b>99.99</b>	<b>99.98</b>
Qwen3 30B	ICL	99.24	99.75	99.49
Qwen3 Next 80B	ICL	99.86	99.95	99.91
GLM 4.5 Air	ICL	94.51	82.58	86.67

Table 8: Comparison of the Analyzer module’s binary classification results on input questions, where macro-average is reported and “ICL” and “FT” refer to in-context learning and fine-tuning, respectively.

In summary, the error analysis confirms that while our Retriever and SQL generation are reliable, the forecasting accuracy of the time-series model remains the critical bottleneck limiting the end-to-end performance of ODTQA-FoRe tasks.

### C.5 Ablation Study of the Analyzer

In the Analyzer, we categorize queries by type and assign each to a corresponding prompt, thereby enhancing overall system performance. To accomplish this classification, we explore two distinct approaches. The first approach leverages in-context learning (ICL) with a large language model (LLM), enabling direct query type determination using only a handful of examples and eliminating the need for large annotated datasets. The second approach involves fine-tuning a BERT classifier<sup>1</sup> on query-type annotations drawn from our QA dataset.

As shown in Table 8, a fine-tuned BERT model achieves performance comparable to that of the Qwen3 30B and Qwen3 Next 80B models equipped with in-context learning (ICL). With traditional approaches requiring substantial amounts of labeled data, LLM-based in-context learning offers a highly effective and economical few-shot solution. This makes it particularly well-suited for scenarios where there is insufficient training data or only a few labeled examples are available.

In the evaluation, we tested the numerical extraction module, which aims to standardize answer generation for different query types. As previously discussed, large language models (LLMs) often produce lengthy explanations instead of concise numerical predictions, even for some simple forecasting tasks. To measure answer effectiveness, we introduce the Valid Completion Rate: the proportion of predictions that are both complete and structurally correct, with all target values reliably extracted as required.

For a fair comparison, other modules remain unchanged—the Retriever continues to retrieve his-

<sup>1</sup><https://huggingface.co/google-bert/bert-base-chinese>

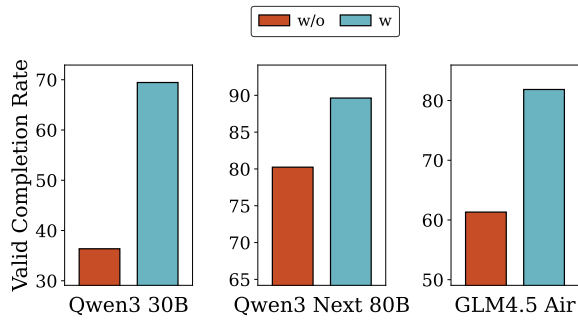


Figure 7: Ablation study of the Analyzer on time-series forecasting tasks, where “w/o” denotes TimeFore without the numerical extraction module, and “w” denotes the complete TimeFore.

torical data, and the Forecaster provides time series forecasts. The numerical extraction module in the Analyzer is removed, so the model outputs answers directly without post-processing. As shown in Figure 7, removing the numerical extraction module decreases the Valid Completion Rate for both models, with Qwen3 30B A3B exhibiting the largest drop (33.08%). This highlights the crucial role of the numerical extraction module in the Analyzer for improving the effectiveness and reliability of the TimeFore framework.

## D Computing Infrastructure Statement

All neural network models were implemented using PyTorch v2.3.1<sup>2</sup>. A single NVIDIA GeForce RTX 4090 GPU was utilized for training both the BERT model for query type classification and all time series forecasting models.

For the LLM experiments, we performed inference using the SGLang library<sup>3</sup> on a cluster of twenty NVIDIA A800-SXM4-80GB GPUs. Specifically, we allocated GPUs as follows: two for Qwen3 30B, four for GLM 4.5-air, four for Qwen3 Next 80B, two for GPT-OSS 20B, and eight for GPT-OSS 120B.

Regarding the implementation of the agent pipeline, we utilized the LangChain framework<sup>4</sup> (specifically the LangGraph components) to orchestrate the interactions between the Retriever, Forecaster, and Analyzer agents. To ensure reproducibility and robustness, we adopted the default configurations provided by the framework. Specifically, the temperature parameter for all LLM API invocations and the maximum number of iterations

for the agents (e.g., the retry mechanism during SQL generation) were set to the default values provided by LangChain, without manual fine-tuning.

## E Prompts

This section summarizes the prompts employed in data construction and the TimeFore framework, incorporating specific tool-use strategies for robust execution. The rewriting prompt used during data construction is detailed in Figure 9.

Within the TimeFore framework, the Retriever module employs a two-stage prompting strategy. First, an LLM summarizes the input query using the prompt in Figure 10. Subsequently, the SQL generation prompt (Figure 11) directs the model to formulate a query and immediately verify its validity by executing it via the sqlQueryTool (Algorithm 1). This ensures that only executable SQL statements are passed downstream.

The Forecaster module is designed to handle missing values and prediction simultaneously. It utilizes the prompt shown in Figure 12 to invoke the imputationThenPredictionTool (Algorithm 2), which performs imputation on historical data before generating future time series forecasts.

Following forecasting, the Analyzer module selects a distinct prompt based on the query type: the prompt in Figure 13 is applied for direct time series forecast queries, while forecast-based reasoning queries utilize the prompt in Figure 14. Finally, the output normalization process and numerical extraction are guided by the prompt presented in Figure 15.

<sup>2</sup><https://pytorch.org/>

<sup>3</sup><https://docs.sglang.ai/>

<sup>4</sup><https://www.langchain.com/>

---

**Algorithm 1** The sqlQueryTool for Database Interaction

---

**Require:** *sql\_statement*: A PostgreSQL-compatible SQL query string

**Ensure:** *result*: String representation of query results or error message

- 1: **Step 1: Initialization**
- 2: Initialize *PostgresQueryExecutor* with database credentials
- 3: **Step 2: Connect and Execute**
- 4: Attempt to establish database connection and create cursor
- 5: **if** Connection or Execution fails **then**
- 6:   *result*  $\leftarrow$  "Error executing SQL: " + Exception message
- 7: **else**
- 8:   Execute *sql\_statement*
- 9:   *rows*  $\leftarrow$  Fetch all results
- 10:   Commit transaction
- 11:   *result*  $\leftarrow$  Convert *rows* to string
- 12: **end if**
- 13: **Step 3: Cleanup**
- 14: Close cursor and database connection (finally)
- 15: **Step 4: Output**
- 16: **return** *result*

---

---

**Algorithm 2** The imputationThenPredictionTool for Time Series Forecasting

---

**Require:** *data*: List of 24 historical values; *project*: str; *device*: str

**Ensure:** *month\_price*: List of predicted month-price pairs

- 1: **Step 1: Validate Input**
- 2: Check  $|data| = 24$  and all values are numeric or "-"
- 3: **Step 2: Data Preprocessing**
- 4: Convert *data* to tensor, replace "-" with NaN
- 5: Create *target\_mask* for missing positions
- 6: **Step 3: Imputation with TimesNet**
- 7: *completed\_inputs*  $\leftarrow$  TimesNet(*data*, *target\_mask*)
- 8: **Step 4: Prediction with TimeXer**
- 9: Prepare decoder inputs and time features
- 10: *predictions*  $\leftarrow$  TimeXer(*completed\_inputs*)
- 11: **Step 5: Format Output**
- 12: Combine month labels with predicted values
- 13: Format as [*project*, *month*, *price*] or [*month*, *price*]
- 14: **return** *month\_price*

---

## Translation Prompt

Your input <Project Name> is a Chinese real estate project name stored in a database.

Translate it into English following these rules:

1. Output only the translated project name—no explanations, punctuation, or extra text.
2. Use the official English name if it exists; otherwise, provide a literal translation following standard English conventions.
3. Prefer literal translation; do not add words or modifiers not present in the original Chinese.
4. Preserve common renderings of proper nouns, place names, and terms (e.g., “国际” → “International”, “中心” → “Center”, “花园” → “Garden”).
5. Keep Arabic numerals and existing Latin letters as-is; convert Chinese numerals to Arabic numerals (e.g., “二期” → “Phase 2”).
6. If unsure whether an official name exists, proceed with the literal translation without speculation or commentary.

Input: <Project Name>

Output: <English Project Name>

### Examples:

Input: 万科国际花园 → Output: Vanke International Garden

Input: 招商·海上世界中心 → Output: China Merchants Sea World Center

Input: 恒大绿洲二期 → Output: Evergrande Oasis Phase 2

Figure 8: Prompt for LLM to translate project names.

## Rewriting Prompt

The input <Query>, generated through template filling with city and project names, is processed according to the following guidelines, rather than being directly answered:

1. The <Rewritten Query> is output in the specified example format.
2. The <Query> is rewritten, not answered. Rewriting methods include, but are not limited to, inversion and synonym replacement. Keywords within the query, such as project names, years, months, or time ranges, are preserved.
3. To enhance sentence diversity, creative modifications are permitted, provided that the original meaning and the keywords remain unchanged.
4. Note: The reference date is December 31, 2022. All queries pertain to future predictions.

#####Example 1#####

<Query>:What will the average price at Wanli Yazhu in Putuo District, Shanghai be next month?

<Rewritten Query>:Could you please estimate what the average price at Wanli Yazhu in Putuo District, Shanghai might be next month?

#####Example 2#####

<Query>:What will the highest average price of Greenland Suzhou ONE in Wujiang District, Suzhou be in the fourth quarter of 2024?

<Rewritten Query>:In the fourth quarter of 2024, how high will the highest average price of Greenland Suzhou ONE in Wujiang District, Suzhou reach?

.....

#####Complete the following#####

<Query>:{query}

<Rewritten Query>:

Figure 9: Prompt used for rewriting the query generated after template filling.

## Summary Prompt for Retrivers

Please complete the for the input according to the example, with the following requirements:

1. Strictly follow the example's format when completing;
2. Do not output any other content.

#####Example 1#####

<Query>:Predict the average price trend of Mingshi Jiayuan in Jianye District, Nanjing City in the first quarter of next year. Will it rise or fall?

<Summary>:['Project Price Table for Jianye District, Nanjing, 2022', 'Project Price Table for Jianye District, Nanjing, 2023']

#####Example 2#####

<Query>:Compared to the current average price, will the average price of Taoran Tingyuan in Hedong District, Tianjin in April 2024 rise or fall?

<Summary>:['Project Price Table for Hedong District, Tianjin, 2022', 'Project Price Table for Hedong District, Tianjin, 2023']

.....

#####Complete the following#####

<Query>: {query}

<Summary>:

Figure 10: Prompt used by LLMs in the Retriever module for query summarization.

## SQL Prompt for Retrivers

You are a helpful SQL assistant that generates SQL queries based on natural language questions about real estate project prices.

**\*\*Current Date\*\***: 2023-12-30

**## Task Overview**

Generate SQL queries for real estate price inquiries and validate them using the provided tool.

=====Step 1: SQL Generation=====

**### Requirements**

- Generate SQL query based on the pattern shown in examples
- Follow the exact structure from the provided examples
- Focus on extracting project names, dates, and average transaction prices

**### Examples**

#####Example 1#####

<Query>:Which of the two neighborhoods, Imperial River New Village or Oriental Famous Garden in Qinhuai District, Nanjing, is likely to have its average price decrease when comparing the average value of the average prices from March to August 2024 with the current average price?

<Table Caption>:['project\_price\_table\_qinhuai\_district\_nanjing\_2022',  
'project\_price\_table\_qinhuai\_district\_nanjing\_2023'],

<SQL>:SELECT project\_name,date,average\_transaction\_price FROM  
project\_price\_table\_qinhuai\_district\_nanjing\_2022 WHERE project\_name IN ('Imperial  
River New Village','Oriental Famous Garden') UNION ALL SELECT  
project\_name,date,average\_transaction\_price FROM  
project\_price\_table\_qinhuai\_district\_nanjing\_2023 WHERE project\_name IN ('Imperial  
River New Village','Oriental Famous Garden');

.....

=====Step 2: SQL Validation=====

**\*\*MANDATORY\*\***: You must validate the generated SQL using the `sqlQueryTool` exactly once.

**### Tool Usage**

Call `sqlQueryTool` with:

- **\*\*sql\_statement\*\***: The generated SQL query string

The tool will return the execution result or error message.

=====Step 3: Final Output=====

After executing the SQL validation tool, output ONLY the following JSON structure:

```
```\njson\n{\n  "sql": "<The validated and executable SQL statement with semicolon>"\n}\n```\n
```

**### Important Notes**

- The "sql" field must contain the complete, validated SQL statement that was successfully executed
- Include the semicolon at the end of the SQL statement
- If the SQL fails validation, revise it based on the error and validate again before final output
- Output only the JSON with the "sql" field - no additional explanations or text

Figure 11: Prompt used by LLMs in the Retriever module for SQL generation.



## Time Series Forecasting Prompt for Analyzer

Current time is December 31, 2023. Please complete the <Answer> based on the input <Query>. Requirements:

1. Output must strictly follow the example format.
2. If there is only one project in the <Query>, you do not need to include the project name when outputting numbers.
3. If there are two or more projects in the <Query>, you must include the project name when outputting numbers.
4. For <Query> items involving multiple projects, when multiple numbers need to be output for the same project, label each number with the project name and the corresponding year-month.
5. Do not output any additional content.
6. <Future Price> is the forecasted price for 2024, and <History Price> is the historical price retrieved for 2022 and 2023.
7. Based on the <Future Price> for 2024 and the requirements of the <Query>, complete the <Answer>.

#####Example 1#####

<Query>: Please forecast what the minimum average price in Suyuan Xincun North District, Wuzhong District, Suzhou, is likely to be between April and September 2024?  
<History Price>:[['October 2022', 22948.24], ['November 2022', 19332.73], ['January 2023', 18105.84], ['February 2023', 20769.33], ['March 2023', 22003.24], ['April 2023', 20690.6], ['May 2023', 21312.39], ['June 2023', 20095.24], ['July 2023', 23200.0], ['August 2023', 20984.02], ['September 2023', 22085.78], ['October 2023', 17944.77], ['November 2023', 19022.12]]  
<Future Price>:[('January 2024', 20269.29), ('March 2024', 19902.74), ('April 2024', 18216.76), ('May 2024', 16887.2), ('June 2024', 17699.11), ('July 2024', 18947.31), ('August 2024', 15327.38), ('September 2024', 14448.49), ('October 2024', 15767.29), ('November 2024', 16609.28)]  
<Answer>:[[14448.49]]

.....  
#####Complete the following#####

<Query>: {query}  
<History Price>: {history\_price}  
<Future Price>: {future\_price}  
<Answer>:

Figure 13: Prompt used by LLMs in the Analyzer module for time-series forecasting.

## Forecast-based Reasoning Prompt for Analyzer

Current date is December 31, 2023. For the given <Query>, please complete the <Answer>. You should use the 2024 <Future Price> to supplement the <Answer> according to the question. Requirements are as follows:

1. Single project: If the <Query> involves only one project, extract the relevant prices from the 2024 <Future Price> per the <Query>, derive the target data (e.g. specified price, average, maximum, minimum), then compare with the <Current Price> and output its status as increase, decrease, or no change.
  - \* [['Increase']]: target data is higher than the current price.
  - \* [['Decrease']]: target data is lower than the current price.
  - \* [['No change']]: target data equals the current price.
2. Multiple projects: If the <Query> involves multiple projects, output, in the prescribed format, the name(s) of one or more projects that meet the criteria; if none, output [['None']].
3. <Current Price> refers to the price for December 2023 (the “current” price), and <Future Price> refers to future prices.
4. Output content: only output the <Answer> part; do not add any other content.

#####Example 1#####

<Query>: Please forecast the lowest average price of Zhonghai Jia No. 3 Courtyard in Fengtai District, Beijing, from April to July 2024, and determine whether it will increase or decrease compared with the current average price.

<History Price>: [['January 2022', 110700.0], ['February 2022', 97995.04], ['March 2022', 98675.75], ['April 2022', 100153.74], ['May 2022', 104237.42], ['June 2022', 102168.49], ['July 2022', 99744.06], ['August 2022', 104785.18], ['September 2022', 107190.09], ['October 2022', 100568.12], ['November 2022', 96782.19], ['December 2022', 106161.84], ['February 2023', 97695.85], ['March 2023', 105981.15], ['April 2023', 107446.73], ['May 2023', 108248.03], ['June 2023', 106424.72], ['July 2023', 108299.13], ['August 2023', 110700.14], ['September 2023', 100720.52], ['October 2023', 93008.9], ['November 2023', 97564.06], ['December 2023', 94147.89]]

<Current Price>: [['December 2023', 94147.89]]

<Future Price>: (('January 2024', 95383.11), ('March 2024', 100582.76), ('April 2024', 100078.41), ('May 2024', 91334.97), ('June 2024', 94108.06), ('July 2024', 100575.87), ('August 2024', 102353.92), ('September 2024', 30927.6), ('October 2024', 56156.51))

<Answer>: [['Decrease']]

.....

#####Complete the following#####

<Query>: {query}

<History Price>: {history\_price}

<Current Price>: {current\_price}

<Future Price>: {future\_price}

<Answer>:

Figure 14: Prompt used by LLMs in the Analyzer module for forecast-based reasoning.

## Normalization Prompt for Analyzer

For the given <Query>, please normalize the <Predicted Answer>. Using the information provided in the <Predicted Answer>, compile the <Normalized Answer>. Requirements are as follows:

1. Single project: If the <Query> involves only one project, do not output the project name—output only the final numeric answer from the <Predicted Answer>.
2. Multiple projects: If the <Query> involves multiple projects, output each project name followed by its corresponding price, in the prescribed format.
3. The <Predicted Answer> is the output of a large model and may contain many irregularities.
4. Do not answer the question—output only the processed <Normalized Answer>.
5. Preserve all numbers exactly as they appear in the <Predicted Answer>.

##### Example 1 #####

<Query>: What will be the lowest average price of Huayudao in Xinjin District, Chengdu, from January to November 2024?

<Predicted Answer>: [[16887.2]]

Explanation: The lowest 2024 forecast average price for Huayudao in Chengdu's Xinjin District (Jan–Nov) is 16,887.2 (rounded to one decimal place), based on January's prediction.

<Normalized Answer>: [[16887.2]]

##### Example 2 #####

<Query>: Please predict the average of all monthly average prices for Shenshan Overseas Chinese Town Chunsui'an in the Shenzhen–Shenshan Cooperation Zone from August to December 2024.

<Predicted Answer>: [['August 2024', 10868.5634765625], ['September 2024', 9006.3291015625], ['October 2024', 10681.0302734375], ['November 2024', 8360.705078125], ['December 2024', 10735.41015625], [10261.63325]]

<Normalized Answer>: [[10261.63325]]

.....

#####Complete the following#####

<Query>: {query}

<Predicted Answer>: {llm\_predict\_answer}

<Normalized Answer>:

Figure 15: Prompt used by LLMs in the Analyzer module for answer normalization.