Irving Fisher Committee on
Central Bank Statistics

BIS

IFC-Bank of Italy Workshop on "Data science in central banking"

18-20 February 2025

Leveraging generative AI for granular credit data utilization: a multi-agent approach[1]

Nontawit Cheewaruangroj, Kawinwish Laobundit, Peranut Nimitsurachat, Supachai Saengthong, Skunpoj Thanarojsophon and Anak Yodpinyanee, Bank of Thailand

[1] This contribution was prepared for the workshop. The views expressed in this publication are those of the authors and do not necessarily represent the official views of the Committee, its members, or the BIS.

Leveraging generative AI for granular credit data utilization: a multi-agent approach

Nontawit Cheewaruangroj, Kawinwish Laobundit, Peranut Nimitsurachat, Supachai Saengthong, Skunpoj Thanarojsophon, Anak Yodpinyanee[1]

Abstract

The Bank of Thailand's Regulatory Data Transformation (RDT) program introduces a new standard for granular credit data reporting by financial institutions. The data's complexity poses analytical challenges for central bank examiners. A generative AI model has been leveraged to aid examiners in utilizing this dataset. Given examiners' questions in plain text, these tools streamline the data analysis process by breaking down examiners' questions into several actionable steps (RDT Brainstormer), searching for relevant fields (RDT Metadata Copilot), and providing SQL code to query from the RDT database (RDT SQL Coder) using a customized Retrieval-Augmented Generation (RAG) procedure. Due to the complexity and large context of each task, a multi-agent framework has been adopted to orchestrate multiple specialized agents to collaborate on the same task. Finally, we deploy our application to production and measure the validity, relevancy and usefulness of the generated output via both automated evaluation and users' feedback. We find a substantial improvement over a single-agent approach.

Keywords: LLM, GenAI, multi-agents, text-to-SQL, Chatbot, granular data

[1] The opinions expressed in this discussion paper are those of the authors and should not be attributed to the Bank of Thailand.

Contents

1. Introduction and Motivation

The Bank of Thailand (BOT) utilizes data extensively for both supervision and monetary policy. To enhance regulatory oversight and well-informed policymaking, the Regulatory Data Transformation (RDT) project was initiated in 2020 to incorporate granular credit data from financial institutions. The initiative aims to replace the current Data Management System (DMS), which primarily relies on fixed reports, thereby restricting the BOT's capability to comprehensively analyze data from multiple perspectives. The granularity of the credit data in the RDT project enables more detailed analysis in various dimensions, including credit lines, interest rate plans, loan movements, and collaterals. However, the RDT project contains a large volume of data as well as intricate data fields and metadata, with more than 1000 fields and over 25 million loan accounts data per month. This complexity hinders its widespread use within the bank, particularly for analysts and examiners who lack familiarity with big data techniques such as SQL. In addition, translating business problems into data querying in RDT poses a challenge. To overcome these challenges, we explore generative AI approaches that can help facilitate usage of the data.

Generative AI (GenAI) has recently gained popularity, especially as assistants for users (Brynjolfsson, Li, & Raymond, 2023). Many central banks have explored using generative AI to assist with their tasks (Aldasoro, et al., 2024). For example, central banks like the European Central Bank (ECB) and Monetary Authority of Singapore (MAS) are utilizing generative AI technologies to enhance their supervisory and analytical capabilities: the ECB's tool Athena uses GenAI to analyze documents and identify trends (ECB, 2023), while the MAS uses GenAI to assist with data collection processes (MAS, 2023). Many GenAI-based chatbots have been developed to help with data utilization, such as SQL coding assistant (Gao, et al., 2024). The ECB has leveraged generative artificial intelligence to translate queries written in plain language into code to find specific data points. This has led to several applications and platforms serving many ECB and EU members' supervisory authorities to streamline their day-to-day operations (ECB, 2024). These initiatives demonstrate how central banks are leveraging AI to improve their operations and decision-making processes.

While these solutions could assist with the mentioned tasks, there are still some obstacles to adopting them in RDT. Firstly, credit data is aggregated into various data cubes, each with different levels of granularity. An SQL coding assistant may fail to select the data cube with the correct granularity because it does not have a comprehensive view of the entire database. Secondly, datasets in RDT contain more than a thousand data fields, making it difficult to identify and verify the correct ones to use. Lastly, posing the right questions to ask the AI may not be straightforward, given the complexity of the data. One way to overcome these challenges is by allowing users to interact with multiple AIs, each with a different specialization such as problem identification, metadata discovery and SQL coding. This framework is known as the multi-agent framework.

The multi-agent framework has been developed to enhance the capabilities of generative AI by allowing multiple AI agents to converse with each other to complete their tasks. It can be employed for various tasks such as retrieval-augmented code generation and question answering (Wu, et al., 2023), improving reasoning (Du, Li, Torralba, Tenenbaum, & Mordatch, 2023) and evaluating the performance of GenAI (Liu, et al., 2023). Using the multi-agent framework would allow the AI agents to assist users in using RDT by breaking down user input questions into different tasks and

assigning different agents for each task. For example, one agent could help with enhancing user's business question, whereas another could retrieve and filter relevant fields. Then, another agent can use the information from the previous agents to craft the SQL query that retrieves the correct data.

In this study, we explore the capabilities of the multi-agent framework to assist data users via three AI modules: *Brainstormer*, *Metadata Copilot* and *SQL Coder*. The *Brainstormer* module refines users' input questions, allowing them to gain better insights. For example, inputs might lack necessary information such as a timeframe and need to be scoped down to specific categories. *Metadata Copilot* assist users in finding the data fields and data cubes relevant to the input questions. Lastly, *SQL Coder* helps users write SQL queries related to the questions. Each module consists of multiple agents that will interact with one another, the user, and a customized retrieval-augmented generation (RAG) that retrieves information from the RDT datasets. The final outputs of these modules will be evaluated by an automated test, as well as our analysts and examiners, in terms of validity, relevancy and usefulness, compared against the outputs from single-agent GenAI as benchmarks.

The remainder of the paper is organized as follows. Section 2 provides background knowledge on generative AI, and the multi-agent framework. Section 3 covers our methodology and architecture. Section 4 details the experiments for evaluating the frameworks. Section 5 discusses the experimental results. Section 6 concludes our study and suggests future work.

2. Background Knowledge

Using Generative AI for Data Querying

Large Language Models (LLMs) can be used to assist with data utilization. One promising capability is to convert user input texts (natural language) to SQL queries, also known as Text-to-SQL. This is often done with zero-shot prompting[2], as LLMs already have sufficient knowledge of SQL (Liu, Hu, Wen, & Yu, 2023). Metadata, such as names and descriptions of tables or fields, need to be provided to LLMs via contexts or using retrieval techniques such as RAG. Few-shot prompting, which includes examples in the context, can further enhance Text-to-SQL capabilities (Rajkumar, Li, & Bahdanau, 2022).

Text Embedding

Text embedding is a technique used in natural language processing (NLP) to convert texts into numerical vectors that capture the semantic meaning of the words or phrases. These embeddings allow AI to understand and process textual data more effectively by representing words in a continuous vector space, where similar words are positioned closer together. Text can be embedded at different levels such as words, sentences or paragraphs (Le & Mikolov, 2014). In LLMs and GenAI applications, texts are often embedded at the level of tokens (Nie, Zhang, & Wu, 2024). This is often done automatically using commercial or open-source LLM services during the inference process. However, for other processes such as document retrieval (see

[2] https://platform.openai.com/docs/examples/default-sql-translate

Retrieval-augmented generation), embeddings need to be applied to the source documents and input questions in order to identify relevant documents. Many pretrained models can be used for embedding tasks such as OpenAI's text-ada-002 (Greene, Sanders, Weng, & Neelakantan, 2022) and BGE-M3 (Chen, et al., 2024).

Retrieval-augmented Generation (RAG)

LLMs used in GenAI are pretrained with fixed knowledge, but do not have access to external knowledge. This external information, such as the organization's data, specific domain knowledge or user preferences, needs to be supplied to LLMs as context. The task is commonly handled with the retrieval-augmented generation (RAG) process (Lewis, et al., 2020). First, candidates of relevant chunks or entries of information are selected. This may be done by finding similarities between the vector of the input questions and the vectorized chunks (Asai, Min, Zhong, & Chen, 2023). Top candidates are added to the context of the LLMs. LLMs can then generate responses with the knowledge of this relevant information.

Multi-agent Framework

Using multiple agents of LLMs can expand the capabilities of generative AI applications, especially for complex tasks. This is because multiple agents can provide feedback to one another, be configured for different capabilities and break down complex problems into smaller tasks (Wu, et al., 2023). Several frameworks exist such as AutoGen[3], LangGraph[4], and CrewAI[5].

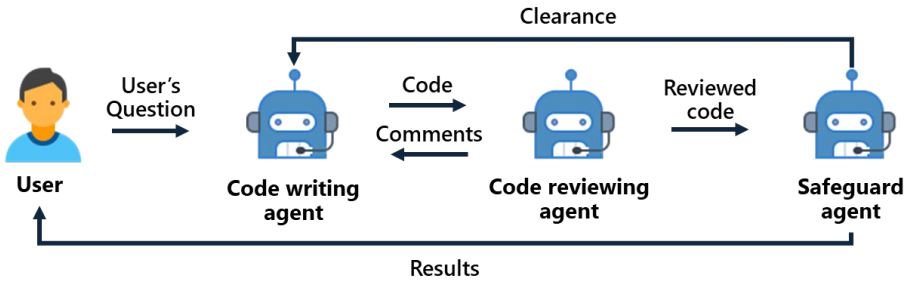
In multi-agent generative AI applications, the task is broken down into smaller tasks, each handled by agents with different configurations. For example, in a coding assistant, the task can be broken down into code writing, code reviewing and safeguarding, as illustrated in Figure 1. First, the code writing agent, prompted to interpret user questions and write codes (e.g., in Python) based on instructions. Then, the output is sent to the code reviewing agent, which is instructed to review the code following some standards. The code reviewing agent may send the comments back to the code writing agent for revisions or pass it to the safeguard agent, which will check for security and compliance. This workflow ensures that all the steps are taken care of. Another advantage of this approach is that only the relevant instructions and context are provided to each agent, reducing the problems of exceeding the context limits of LLMs. In addition, users can also see intermediate exchanges between agents and may also participate in the process.

[3] https://github.com/microsoft/autogen

[4] https://www.langchain.com/langgraph

[5] https://github.com/crewAIInc/crewAI

Example of a multi-agent workflow for a coding assistant AI Figure 1



An example of a multi-agent workflow for a coding assistant. The workflow involves three distinct agents: the Code Writing Agent, responsible for generating code; the Code Reviewing Agent, tasked with evaluating and providing feedback on the generated code; and the Safeguard Agent, ensuring compliance with safety and security standards. This collaborative approach enhances the efficiency and reliability of the coding process.

Evaluating Generative AI Response

Evaluating responses from GenAI can be both objective and subjective. For example, retrieving the correct data field is objective, whereas producing useful suggestions can be subjective. In our context, objective evaluation involves information retrieval problems. Several metrics can be used as in information retrieval systems, such as precision and recall (Saracevic, 1995).

Precision measures the proportion of relevant instances from all the retrieved instances. This metric can be used to identify the relevancy of the retrieved information. **Recall** measures the proportion of the relevant instances that were retrieved from all relevant instances. It can be used to determine whether all the relevant data fields, for instance, are retrieved. Mathematically, precision and recall are defined as:

$$\text{Precision} = \frac{\text{Relevant retrieved instances}}{\text{All retrieved instances}},$$

$$\text{Recall} = \frac{\text{Relevant retrieved instances}}{\text{All relevant instances}}.$$

Precision and recall values range from 0 to 1, where a precision of 1 means all the retrieved information is relevant and a recall of 1 means the algorithm was able to retrieve all the relevant information. In the context of GenAI responses, precision and/or recall can be used to determine the **relevancy** of the responses.

For coding tasks, such as SQL generation, the validity of the code generated can also be evaluated (Chen, et al., 2024). **Validity** may be defined as the percentage of code that is valid in terms of compilability, type-checked syntax or in successfully performing desired tasks. This may be done automatically or by human evaluation.

Some evaluations need to be subjective: for example, the usefulness of the output is best determined by the users of the GenAI (Baek, Chandrasekaran, Cucerzan, Herring, & Jauhar, 2024). **Usefulness** determines whether the outputs to the users' input questions are beneficial to them. Such a metric can be directly evaluated by users, such as through a survey using a 7-point Likert scale, where scores range from 1 (strongly disagree) to 7 (strongly agree). Alternatively, these metrics can be evaluated automatically by using other LLMs as evaluating agents (Gao, et al., 2023).

However, their assessment might not capture the opinions of actual users with diverse expertise and expectations, so evaluating agents are not leveraged in this study.

3. Methodology

We develop three complementary modules—Brainstormer, Metadata Copilot, and SQL Coder—that leverage customized RAG, LLM and Structured Data Validation. While these modules share core components, each has a distinct agent workflow implemented via AutoGen multi-agent framework. The details and structure of each core component are outlined below.

Customized RAG

RAG can be utilized to enhance LLM by retrieving relevant information from external database before generating responses to users. This architecture is particularly useful for our application because it provides LLM with necessary details of RDT metadata, especially the definition of each field, enabling LLM to assess the situations and provide information more accurately.

The RAG system in our implementation utilizes a hybrid retrieval approach that combines dense and sparse vector representations to improve accuracy and efficiency of the system. The system processes user queries through parallel tracks, leveraging both embedding-based techniques and traditional information retrieval methods to capture both of semantic and keyword relationships.

In the first track, user's queries are passed through the BGE-M3 embedding model (Chen, et al., 2024) and compared against those of our documents stored in a vector database called Qdrant[6]. This approach captures semantic relationships and contextual nuances that might be missed by traditional keyword matching.

Simultaneously, in the second track, we also use Term Frequency-Inverse Document Frequency (TF-IDF) and the BM25 ranking algorithm (Robertson, Walker, Hancock-Beaulieu, Gatford, & Payne, 1996) to process users' query. This combination allows us to capture keyword matches and term importance effectively. Finally, we also implement the negative Matrix Factorization (NMF) to identify latent topics and improve retrieval relevance. This multi-track approach allows our system to maintain high retrieval accuracy across various query types, from specific technical inquiries to more general, contextual questions about RDT dataset. The results from both tracks are then aggregated to select the most relevant previous cases from our dataset gathered in the first step; these reference cases will be used as a context for the LLM later.

LLMs and Structured Data Validation

In this step, we integrate LLMs with structured data handling mechanisms. The system processes questions from users; these questions can be classified into three main categories based on the nature of each application: high-level analytical questions about business insights (Brainstormer), metadata field definitions (Metadata Copilot), and SQL code for specific data retrieval problems (SQL Coder). Through a RAG

[6] https://qdrant.tech/

pipeline, the system pulls relevant information from our RDT metadata database to address these queries. Our private Azure OpenAI subscription is prompted to analyze the RAG results and answer users' questions accordingly using *gpt-4o-mini*[7] model.

To ensure consistency in the output format, we implement a data validation framework using Pydantic[8], combined with JSON formatting. This framework enforces a predefined schema on output from LLMs to include all necessary components such as relevant metadata fields, detailed analysis of users' questions, and comprehensive explanations. The structured output schema plays a crucial role in maintaining response quality. Each query response generated by the system must include a clear breakdown of the problem, relevant metadata field descriptions, or SQL codes depending on the application.

Application Deployment

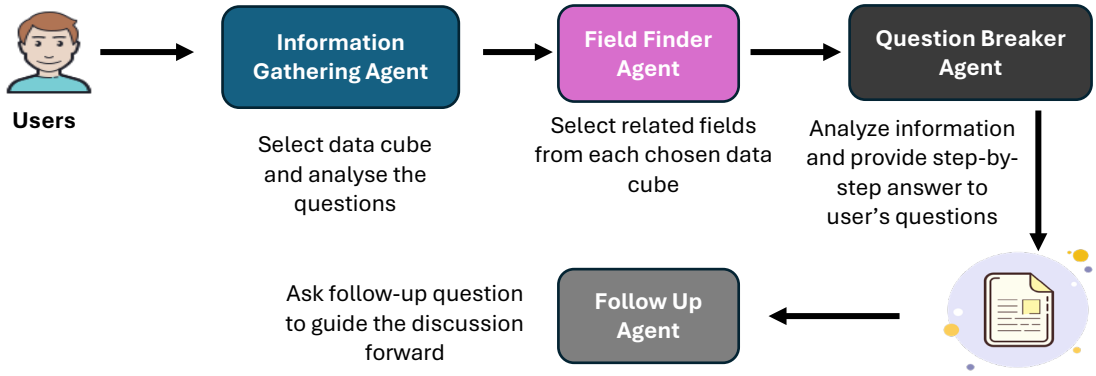
The deployment phase of the system utilizes Streamlit[9] as the primary frontend framework, offering a user-friendly web interface for accessing the service. The application is hosted on an internal server. To manage access control and maintain user session data, we implement a secure user authentication system for all users. Additionally, we also implement comprehensive logging mechanisms that capture user interactions, including both query inputs and system responses, to facilitate future improvement and system optimization. This logging system stores detailed interaction data, including timestamps, user identifiers, query content, RAG retrieval results, as well as the final outputs from each application. The collected data serves multiple purposes: enabling performance monitoring, identifying common questions about RDT, and evaluating the accuracy of application outputs.

Multi-agent Workflow

This subsection outlines the RDT Copilots agents' workflow.

Brainstormer workflow

Figure 2



Multi-agent workflow for Brainstormer module. Each box corresponds to an LLM agent.

7 https://www.openai.com/

8 https://pydantic.dev/

9 https://streamlit.io/

Brainstormer

Brainstormer serves as a chatbot interface designed to decompose complex, high-level queries into manageable sub-questions that can be answered with RDT dataset. The system operates through a sequence of specialized LLM agents as shown in Figure 2. Initially, the Information Gathering agent analyzes user queries alongside detailed cube descriptions in its prompt to identify and select the most suitable data cubes for further processing. These selected cubes are then passed to the Field Finder agent, which extracts relevant fields based on the user's query. The Question Breaker agent utilizes these selected fields as context to decompose the original query and provide guidance for RDT-based solutions. To propose new ways of problem solving, the Follow Up agent generates targeted questions that guide the conversation toward more insightful outcomes. This multi-agent architecture ensures a systematic and thorough approach to query processing, while maintaining user engagement throughout the interaction. The modular design not only enhances the system's efficiency but also allows for future scalability and improvements. By breaking down high-level queries into smaller, manageable components, Brainstormer significantly reduces the complexity while maximizing the utility of the RDT dataset.

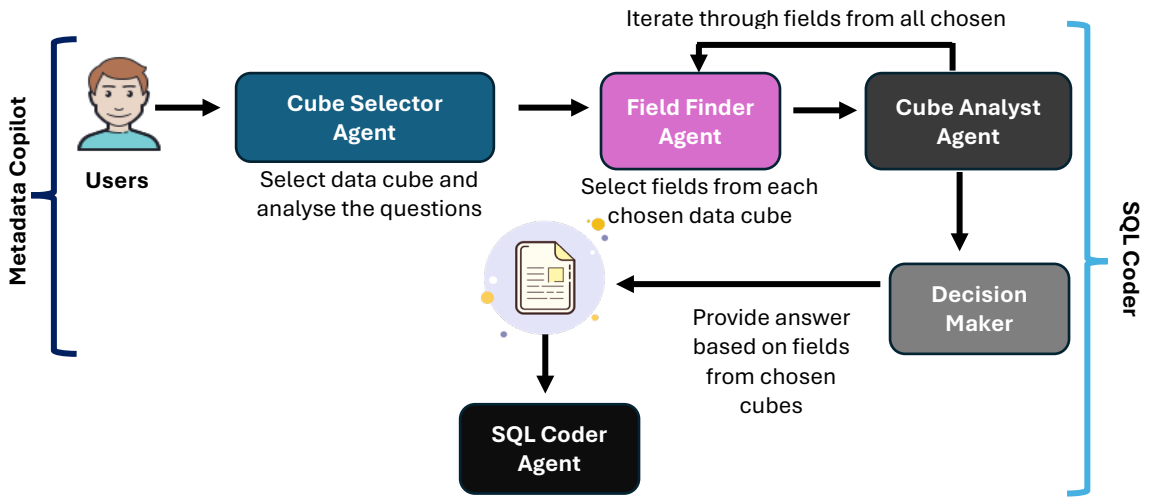
Metadata Copilot

Metadata Copilot is a specialized chatbot system focused on retrieving fields from the RDT dataset through our customized RAG pipeline. The system utilizes a system of agents to process user queries and provide useful fields effectively as shown in Figure 3; this will reduce the time users need to spend on manually searching for fields from RDT's complicated data dictionary. Beginning with the Cube Selector agent, the system analyzes user questions to identify and select the most relevant data cubes from the available collection. These selected cubes are then processed by the Field Finder agent, which extracts pertinent fields based on the query context. The Cube Analyst agent performs a critical review of the extracted fields, initiating an iterative feedback loop with the Field Finder if the initial field selection is inaccurate. Finally, the Decision Maker agent evaluates the collective findings and generates a definitive list of fields to provide to users. This multi-agent approach is beneficial in this context as it distributes tasks across several specialized agents, reducing the workload assigned to each individual agent while improving accuracy and minimizing hallucination in the output. Furthermore, this distributed architecture leverages the combined context windows of multiple agents, enabling the entire system to process more information from data dictionary than would be possible with a single agent.

SQL Coder

SQL Coder serves as a query generation system that transforms field-level metadata into SQL queries for retrieving data from the RDT dataset. Building upon Metadata's field selection capabilities, the system extends the processing pipeline with the addition of the specialized SQL Coder agent. Once the Decision Maker agent from Metadata finalizes the field list, SQL Coder agent analyzes the selected fields along with user query and selected cubes to determine SQL operations such as table joins, filtering conditions, and aggregation requirements. Finally, SQL Coder agent will write the executable SQL code that satisfies this requirement and provide to users. With SQL Coder, users can paste this provided code directly into their data portal and quickly retrieve the RDT data.

Metadata Copilot & SQL Coder workflow Figure 3



Multi-agent workflow for Metadata Copilot and SQL Coder modules. Each box corresponds to an LLM agent.

4. Experiment

In this section, we describe the experimental framework designed to evaluate the performance of our proposed multi-agent workflow in comparison to the baseline single-agent algorithm in copilot tasks aimed at assisting non-technical users in leveraging the RDT dataset. Recall that the multi-agent method outperforms its single-agent counterpart in terms of **context efficiency**: due to its extensive resource consumption, the single-agent algorithm is rendered impractical for deployment on this database. In this experiment, we aim to demonstrate that the multi-agent's task decomposition scheme also delivers superior or at least comparable **output quality** on these copilot tasks to its highly accommodated single-agent implementation in our simplified testing environment.

Tasks and Evaluation Methods

While all tasks are inherently interactive, for ease of performance comparison between the two algorithms, we consider the simplified one-round output-only scenario, thereby reducing to the question-answering setting. To this end, we create two types of questions: 62 **quantitative** questions aimed at generating specific reports (values or tables), and 30 **qualitative** questions seeking comprehensive guidance on solving analytical problems faced by central bankers, such as policy-related issues or enforcement of supervision criteria. We include 10 (translated) questions of each type in Appendix 1.

We evaluate our algorithms on the three following tasks.

Brainstormer. Given a *qualitative* question, the algorithm must suggest plausible solutions by decomposing the problem into sub-questions that can be addressed through database queries in a step-by-step fashion, while identifying essential data fields to demonstrate feasibility. It is not required to provide actual

answers or queries to solve the problem at this point: by design, the users must decide to pursue these steps themselves and will be further assisted by Metadata Copilot.

Metadata Copilot. Given a *quantitative* question, the algorithm must retrieve columns from one or more tables that are necessary for computing the answer (including intermediate fields required for filtering data, deduplicating records, and joining tables). This task requires an understanding of the data model, background concepts in central banking and related subjects, as well as abbreviations and jargons used internally by our bank examiners.

SQL Coder. Given a *quantitative* question, the algorithm must provide an SQL code that, once inputted to an SQL query execution engine, generates the desired report. While the input is simply the free-text question, our internal prompt includes separated instructions for short-listing columns (also featured in Metadata Copilot) and generating a query: the algorithm's performance can consequently be overshadowed by the quality of its former step.

To measure our algorithms' performance, we consider two evaluation methods: the automated evaluation with preset grading schemes, and the user evaluation that relies on human scoring. Their differences lie in grading objectives, processes, metrics, and underlying data tables, as summarized in Table 1.

Comparison between Automated Evaluation and User Evaluation Table 1

	Automated Evaluation			**User Evaluation (all tasks)**
	Brainstormer	**Metadata Copilot**	**SQL Coder**	
objective	relevancy	relevancy	validity	usefulness
metric	recall	recall	0/1	7-point Likert scale
dataset	multiple tables with ≤ 300 columns each (join queries and intermediate calculations may be required)			one table with ≈ 1,000 columns (feature-engineered columns included)

Automated Evaluation. This method assesses the quality of the output using pre-determined sets of criteria, allowing an automated testing on a larger number of tests, thus providing a more thorough coverage of the data fields. We measure different aspects of the algorithms' performance for our tasks.

- **Relevancy.** For the Brainstormer and Metadata Copilot tasks, we measure their ability to retrieve related columns by comparing their output columns against our answer keys and compute the recall rate. We design our answer keys for Metadata Copilot to cover *attribute groups* that should appear in the resulting SQL query, whereas for Brainstormer we focus on *concepts* that should be touched upon when solving the qualitative question. Rather than matching specific columns, we award a true positive point when they cover any fields in the required groups (e.g., business type/group/ISIC) or concepts (e.g., DPD or stage). We do not explicitly set a limit on the number of allowed output columns—with our prompts, both algorithms return a reasonable number of columns for their respective tasks and questions (up to 9 and 21 columns for Metadata Copilot and Brainstormer, respectively).
- **Validity.** For the SQL Coder task, we measure the SQL codes solely for their compilability, for two reasons. First, we aim to isolate the coding component from the business aspect, for it has already been evaluated above during the retrieval subtask, from which the coder subtask receives its short-listed

columns. Second, given the colloquialism of questions and the redundancy of data fields, the desired reports are generally underspecified (which can be resolved via a multi-turn conversation, in practice). As the correctness evaluation requires human judgement, we delegate it to the user evaluation.

The dataset for automated evaluation consists of smaller, low-level tables that often require complex join operations and intermediate computations. It tests the algorithms' capabilities in handling fundamental database structures where technical complexities are prevalent.

User Evaluation. In all three tasks, we conducted a user evaluation to gauge the perceived **usefulness** of the algorithms' outputs. Participants are presented with the outputs and asked to rate their agreement with the statement "*The algorithm's output is useful for solving the given problem*" on a 7-point Likert scale, from strongly disagree (score = 1) to strongly agree (score = 7). We intentionally let the participants decide on their own notion of "useful". At the same time, we prompt our algorithms to include explanations to their answers to ensure that the participants understand and are convinced of their correctness. Due to the participants' limited availability, we only select 10 questions of each type (precisely listed in Appendix 1) for user evaluation. We provide translated answers to some questions in Appendix 2.

The dataset for user evaluation is a single table indexed by loan account and data period as unique keys. Designed as a table for non-technical users, it contains many feature-engineered business-oriented additional fields, such as period-over-period computations, cumulative or window functions, and grouping-aggregation into different credit-line levels, amounting to roughly 1,000 features. This dataset focuses on evaluating the algorithms' ability to handle large metadata that requires business acumen, as well as the ability to eloquently express their answers to users who have limited familiarity with the data fields.

Accommodating Single-Agent Algorithm

Recall that the single-agent approach fails to execute on the RDT dataset because its limited context window size is insufficient for our metadata. As we aim to measure output qualities, we accommodate for the single-agent framework as follows. We let the single-agent algorithm skip earlier subtasks that otherwise requires large context and provide these subtasks' outputs directly. As described in Table 2, the single-agent algorithm is given relevant data cubes and shortlisted data fields, thereby bypassing the context-intensive cube selection and RAG steps altogether. In contrast, the multi-agent algorithm handles RDT metadata by distributing the relevant parts to its agents, which then perform their respective subtasks and passing their small outputs around according to our predefined workflow.

Comparison between Implementation of Multi- and Single-Agent Algorithms Table 2

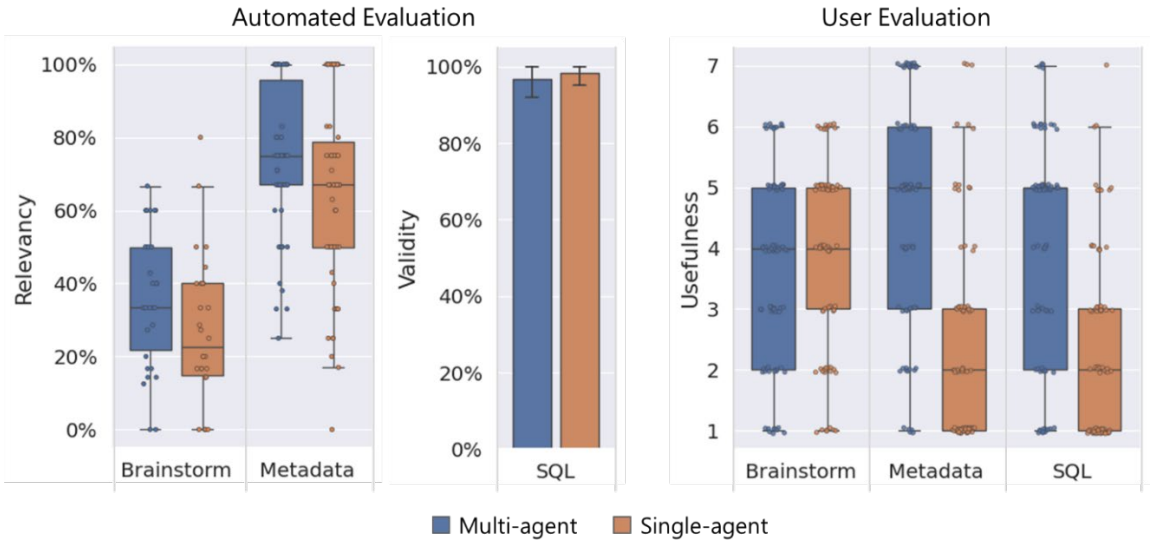
Subtask	Multi-Agent	Single-Agent
Data cube selection	general description of up to three cubes is given; the algorithm chooses relevant cubes	general descriptions of strictly required cubes are given; no cube selection required
RAG	field description of chosen cubes is given; the algorithm performs RAG	a retrieved list of fields along with their description is given; no RAG required
Brainstormer /Metadata Copilot / SQL Coder	agents solve the remaining subtasks using the data computed by other agents	one agent solves the remaining subtasks from the given prepared data

While the single-agent algorithm seemingly receives a huge handicap in this setup, we mitigate this advantage for a fairer comparison as follows. Firstly, we prompt the single-agent algorithm to reproduce the multi-agent workflow by guiding it through the same intermediate subtasks. Secondly, for the automated evaluation which involves smaller low-level cubes (tables), we restrict the questions to simpler ones that can reasonably be solved with at most two cubes, and always provide only three total cubes for selection. In essence, while the multi-agent scheme is vital for enabling our AI assistant to handle the large RDT metadata, this simplified experiment makes the single-agent framework feasible while keeping the overall task's difficulty on a rather even playing field, suitable for evaluating output quality.

5. Result & Discussion

Evaluation Results

Figure 4



This chart shows results from automated and user evaluations, using relevancy, validity and usefulness as metrics. For relevancy, each dot represents a recall score for each question in retrieving expected fields. For validity, the bars show the percentage of valid SQL queries. The error bars represent 95% CI. For usefulness, each dot represents a usefulness score (1 = strongly disagree, 7 = strongly agree) for each response, with jitter added to improve visibility.

Summary of evaluation results

Table 3

Evaluation	Module	Metric	Sample Size	Mean score		p-value	Significance
				Multi-agent	Single-agent		
Automated	Brainstormer	Relevancy	30 questions	**73.7%**	64.8%	0.006	**
	Metadata Copilot	Relevancy	62 questions	96.8%	**98.4%**	0.564	
	SQL Coder	Validity	62 questions	**35.9%**	26.9%	0.011	*
Human	Brainstormer	Usefulness	10 questions × 8 users = 80	**4.763**	2.363	9.87E-12	***
	Metadata Copilot	Usefulness	10 questions × 8 users = 80	**4.025**	2.313	5.74E-09	***
	SQL Coder	Usefulness	10 questions × 8 users = 80	3.675	**3.913**	0.100	

The framework with superior score is bold for each row. P-values and significance levels are calculated using Wilcoxon signed-rank test to determine whether there is a significant difference between multi-agent and single-agent frameworks. Asterisks refer to significance level (* for $p < 0.05$, ** for $p < 0.01$, *** for $p < 0.001$).

The evaluation results are summarized in Figure 4 and Table 3. For the automated evaluation, the results show that RDT copilots can perform objective tasks, such as field retrieval and SQL generation for data querying, quite reliably. Metadata Copilot with the multi-agent framework achieves an average relevancy score of 73.7%, improving upon the single-agent framework's (64.8%). This margin suggests that using multiple agents can enhance GenAI's data retrieval capabilities. SQL Coder obtains a high validity score, but there are no significant differences between multi-agent (96.8%) and single-agent (98.4%) frameworks. One plausible explanation is that the current LLM excels at SQL generation under the single-agent framework: adopting multiple agents here would yield negligible benefit. However, subjective tasks such as Brainstormer show an unsatisfactory performance, with average relevancy scores of 35.9% for the multi-agent framework and 26.9% for the single-agent framework, suggesting that the task of answering open-ended questions in a single turn may be overly ambitious.

For user evaluation, our analysts and examiners found multi-agent Metadata Copilot and SQL Coder slightly useful for answering quantitative questions, with average usefulness scores of 4.76 and 4.02, respectively, markedly surpassing the single-agent framework's scores of 2.36 and 2.31. For Metadata Copilot, users tend to rate more highly when the relevant fields are selected, and the multi-agent implementation seemingly outperforms the single-agent counterpart under user evaluation's dataset. As for SQL Coder, the multi-agent version reportedly provides better coding structures and clearer explanation. However, they slightly disagree that the multi-agent Brainstormer is useful for answering qualitative questions (3.68); this score is non-negligibly worse than the single-agent Brainstormer's (3.91), apparently due to the responses' writing styles: some users mentioned that the answers of multi-agent Brainstormer are too general or incoherent.

There are significant variations in relevancy and usefulness in both evaluations, indicating that some questions are harder for these agents. Some users mentioned that GenAI performed worse for questions that require domain expertise. This is likely because the information in gpt-4o-mini's training data and our data dictionary

combined may not be sufficient for performing such tasks. User ratings for usefulness also vary greatly even within the same question, likely due to users' different coding skills and expectations. Designing fair and relevant experiments, test questions, and evaluations for GenAI applications remains a big challenge.

6. Conclusion and Further Work

Conclusion

In this paper, we explore the application of generative AI in the context of the Bank of Thailand's Regulatory Data Transformation (RDT) program. Our approach leverages a multi-agent framework to enhance the utilization of granular credit data by central bank analysts and examiners. By breaking down complex tasks into manageable steps, our RDT Copilots—comprising Brainstormer, Metadata Copilot, and SQL Coder—streamline the data analysis process and improves the accuracy and efficiency of responses.

For RDT Copilots, the multi-agent framework significantly improves the performance of GenAI applications compared to the single-agent approach, especially for retrieving relevant fields and producing useful output for data querying. These tasks benefit from the multi-agents' larger context sizes and abilities to break down problems into smaller tasks. The framework can be applied to different tasks, including more objective task as well, allowing more flexible workflows for GenAI applications.

User evaluation indicates that RDT Copilots can assist users in solving objective or quantitative problems such as searching and querying data for examining tasks. Most of participants' responses suggest that the outputs generated by Metadata Copilot and SQL Coder are generally useful, aside from certain questions that require deeper domain expertise. Subjective or qualitative tasks such as suggesting analytical frameworks for solving open-ended questions, however, remain challenging for our Brainstormer module. Overall, we believe that RDT Copilots have the potential to mitigate the challenges associated with RDT data utilization and accelerate the success of the transformation program.

Further Work

Future work will focus on improving the performance of RDT Copilots. One gap is to improve the quality of the metadata and enrich the domain knowledge for RDT Copilots in systematic ways, which will enable more accurate and relevant outputs. In addition, Brainstormer needs to be revised to produce more useful outputs. This may include designing a more detailed workflow, improving prompt engineering techniques or adopting LLMs with reasoning capabilities.

We are also investigating other potential use cases that will help with utilizing RDT data, such as adding the capabilities to generate graphs from the query. Other use cases of multi-agent GenAI applications are also being explored, including but not limited to Knowledge Management Chatbot and Supervision Assistant Agents.

References

Aldasoro, I., Doerr, S., Gambacorta, L., Notra, S., Oliviero, T., & Whyte, D. (2024). Generative artificial intelligence and cyber security in central. *BIS Papers*(No 145).

Asai, A., Min, S., Zhong, Z., & Chen, D. (2023). Retrieval-based Language Models and Applications. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, (pp. 41-46).

Baek, J., Chandrasekaran, N., Cucerzan, S., Herring, A., & Jauhar, S. K. (2024). Knowledge-Augmented Large Language Models for Personalized Contextual Query Suggestion. *Proceedings of the ACM Web Conference 2024*. doi:10.1145/3589334.3645404

Brynjolfsson, E., Li, D., & Raymond, L. (2023). Generative AI at Work. *NBER Working Papers*(No 31161).

Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., & Liu, Z. (2024). BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv*. doi:10.48550/arXiv.2402.03216

Chen, L., Guo, Q., Jia, H., Zeng, Z., Wang, X., Xu, Y., . . . Zhang, S. (2024). A Survey on Evaluating Large Language Models in Code Generation Tasks. *arXiv*. doi:arXiv:2408.16498v1

Du, Y., Li, S., Torralba, A., Tenenbaum, J., & Mordatch, I. (2023). Improving Factuality and Reasoning in Language Models through Multiagent Debate. *Arxiv*. doi:10.48550/arXiv.2305.14325

ECB. (2023, November 15). Suptech: thriving in the digital age. *Supervision Newsletter*.

ECB. (2024, September 18). The future of European banking supervision – connecting people and technology. *Keynote Speech*.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2024). Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proceedings of the VLDB Endowment*, *17*, pp. 1132-1145.

Gao, M., Ruan, J., Sun, R., Yin, X., Yang, S., & Wan, X. (2023). Human-like Summarization Evaluation with ChatGPT. *ArXiv*. doi:10.48550/arXiv.2304.02554

Greene, R., Sanders, T., Weng, L., & Neelakantan, A. (2022, Dec 15). *New and improved embedding model*. Retrieved from OpenAI: https://openai.com/blog/new-and-improved-embedding-model

Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *ArXiv*. doi:10.48550/arXiv.1405.4053

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Proceedings of the 34th International Conference on Neural Information Processing Systems*, (pp. 9549-9474).

Liu, A., Hu, X., Wen, L., & Yu, P. S. (2023). A comprehensive evaluation of ChatGPT's zero-shot Text-to-SQL capability. *arXiv*. doi:10.48550/arXiv.2303.13547

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., . . . Tang, J. (2023). AgentBench: Evaluating LLMs as Agents. *Arxiv*. doi:10.48550/arXiv.2308.03688

MAS. (2023, November 16). MAS Launches Digital Platform for Seamless ESG Data Collection and Access. *Media Releases*.

Nie, Z., Zhang, R., & Wu, Z. (2024). A Text is Worth Several Tokens: Text Embedding from LLMs Secretly Aligns Well with The Key Tokens. *ArXiv*. doi:10.48550/arXiv.2406.17378

Rajkumar, N., Li, R. L., & Bahdanau, D. (2022). Evaluating the Text-to-SQL Capabilities of Large Language Models. *arXiv*. doi:10.48550/arXiv.2204.00498

Robertson, S., Walker, S., Hancock-Beaulieu, M., Gatford, M., & Payne, A. (1996). Okapi at TREC-4. *The Fourth Text REtrieval Conference (TREC-4)*.

Saracevic, T. (1995). Evaluation of evaluation in information retrieval. *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, (pp. 138-146).

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., . . . Wang, C. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *ArXiv*. doi:10.48550/arXiv.2308.08155

Appendix 1: Questions for Evaluation

Quantitative Questions for User Evaluation (translated)

1. Total outstanding balance in the latest month, among the debtors who had a total outstanding balance across all accounts exceeding 500 million baht as of December 31, 2023.
2. Monthly counts of debtors who have joined the debt restructuring program for each FI.
3. Highest and lowest PD of loan contracts in the month prior to changing status to SM, by type, by FI.
4. YoY growth of the outstanding balance of business loans.
5. Average income of new debtors in each data period.
6. Show the total outstanding balance of debtor groups that belong to more than five different business types.
7. Calculate the total outstanding balance for the latest month, of the debtor groups who have at least one NPL contract.
8. Show housing loans where the total collateral value does not exceed 20% of the approved loan amount.
9. Calculate the total outstanding balance per debtor, only for debtors who have ever had a maximum DPD greater than 90 days.
10. Total change in the outstanding balance of debtors who have been SM/NPL within the last 12 months.

Qualitative Questions for User Evaluation (translated)

1. Do different FIs consistently appraise the value of real estate collateral in each area?
2. Forecast the volume of NPLs in the next year.
3. In the past quarter, have FIs become more cautious or less cautious in lending to SMEs and retail customers compared to previous periods?
4. Does a reduction in policy interest rate genuinely reduce the burden on debtors?
5. To what extent has the 2024 LTV measure—which allows borrowers to borrow an additional 10% above the 100% LTV ceiling for housing loans during 2024—been utilized in practice?
6. Estimate customers' income using payment history to create a risk profile.
7. Do financial institutions use a DSR threshold when evaluating loan applications?
8. In 2024, a new measure was introduced to facilitate co-borrowers without home ownership rights to more easily access housing loans. Has this measure been effective?
9. High Interest Rates: Are They Driving Bank Profits Through the Roof?
10. What should the minimum payment rate be for credit card debt repayments?

Appendix 2: Responses for Evaluation Questions

As a reminder, these results are for the user evaluation. The only given data table is smart_cube_1, but the algorithms are still required to perform the cube selection task to illustrate their complete pipeline.

Responses for Quantitative Question 10 (translated)

Question: Total change in the outstanding balance of debtors who have been SM/NPL within the last 12 months.

Multi-agent algorithm's response for Metadata Copilot

User's Requirement Analysis:

The user wants to track the total outstanding balance changes of debtors who were previously SM/NPL over the last 12 months. This requires contract/account-level data from various financial institutions and information on the classification of credit debt.

Cube Selection and Reason:

Selected Cube: Smart_Cube_1

Reason: Smart_Cube_1 contains comprehensive data including financial institution codes, debtor account numbers, credit debt classification, data periods, and total outstanding balances. All of these are necessary for analyzing the total outstanding balance changes of debtors who were previously SM/NPL over the last 12 months.

Analysis of Relevant Columns:

Column Name	Description	Relevance to Requirement
fi_bot_unique_key	Unique code of the financial institution	Used to identify the financial institution related to the debtor data
account_id	Debtor's account number	Necessary for tracking account-level debtor data
entity_id	Unique code of the debtor group	Used to group debtors for analysis
debtor_bot_unique_key_ merge_mask	Unique code of the debtor	Used to identify individual debtors
asset_and_contingent_ class_code	Asset and contingent classification code	Used to identify the status of credit debt (e.g., SM/NPL)
period	Data period (year)	Used to identify the time period of the data to be analyzed
outstanding	Outstanding credit balance before deducting unearned revenue	Used to analyze the total outstanding balance of debtors
change_class_yoy_level1	Change in classification compared to 12 months ago	Used to analyze changes in the status of credit debt over the last 12 months

Multi-agent algorithm's response for SQL Coder

SQL Query:

```
WITH recent_periods AS (
    SELECT DISTINCT period
    FROM Smart_Cube_1
    ORDER BY period DESC
    LIMIT 12
),
sm_npl_accounts AS (
    SELECT DISTINCT account_id
    FROM Smart_Cube_1
    WHERE asset_and_contingent_class_code IN ('SM', 'NPL')
)
SELECT
    fi_bot_unique_key,
    account_id,
    SUM(outstanding) AS total_outstanding_balance,
    SUM(change_class_yoy_level1) AS total_change_in_outstanding_balance
FROM
    Smart_Cube_1
WHERE
    period IN (SELECT period FROM recent_periods)
    AND account_id IN (SELECT account_id FROM sm_npl_accounts)
GROUP BY
    fi_bot_unique_key,
    account_id
ORDER BY
    fi_bot_unique_key,
    account_id;
```

Description:

- **recent_periods:** Create a temporary table that stores the data periods for the last 12 months, ordered from the most recent to the oldest period.
- **sm_npl_accounts:** Create a temporary table that stores the account numbers of debtors who were previously SM or NPL.
- **Main Query:**
    - Select data from Smart_Cube_1 that is within the last 12 months and includes accounts that were previously SM or NPL.
    - Calculate the total outstanding balance (total_outstanding_balance) and the change in the outstanding balance (total_change_in_ outstanding_balance).
    - Group the data by financial institution code (fi_bot_unique_key) and account number (account_id).
    - Order the data by financial institution code and account number.

Single-agent algorithm's response for Metadata Copilot

User's Requirement Analysis:

The user needs information on the total outstanding balance changes of debtors who were previously SM/NPL over the last 12 months. This means tracking the changes in the outstanding balances of debtors who had the status of SM (Special Mention) or NPL (Non-Performing Loan) during the specified period.

Cube Selection and Reason:

Selected Cube: Smart_Cube_1

Reason: Smart_Cube_1 contains comprehensive data including financial institution codes, debtor account numbers, credit debt classification, data periods, and total outstanding balances. All of these are necessary for analyzing the total outstanding balance changes of debtors who were previously SM/NPL over the last 12 months.

Analysis of Relevant Columns:

Column Name	Description	Relevance to Requirement
period	Data period (year)	Used to identify the last 12 months
fi_bot_unique_key	Financial institution code	Used to identify the financial institution related to the debtor
account_id	Account number	Used to identify the debtor's account to be tracked
asset_and_contingent_class_reason_code	Asset and contingent classification reason code	Used to identify the status of the debtor as SM or NPL
npl_reentry_date_latest	Date of the latest reentry into NPL status	Used to identify changes in the debtor's status

Note: Selecting these columns helps efficiently track changes in outstanding balances and debtor status over the specified period.

Single-agent algorithm’s response for SQL Coder

To find the total outstanding balance changes for debtors who were previously classified as SM (Special Mention) or NPL (Non-Performing Loan) within the last 12 months, we need to filter the data based on the debtor's classification and the period. We will use the asset_and_contingent_class_reason_code to identify the SM/NPL status and the period to filter the last 12 months.

Here's the SQL query to achieve this:

```
WITH RecentPeriods AS (
    SELECT DISTINCT period
    FROM CUBE_smart_cube_1
    ORDER BY period DESC
    LIMIT 12
)

SELECT
    entity_id,
    SUM(other_debt_burden) AS total_outstanding_balance_change
FROM
    CUBE_smart_cube_1
WHERE
    asset_and_contingent_class_reason_code IN ('2000900005', '2000900019') --
Assuming these codes represent SM/NPL
    AND period IN (SELECT period FROM RecentPeriods)
GROUP BY
    entity_id
```

Explanation:

1. **RecentPeriods CTE:** This Common Table Expression (CTE) selects the distinct periods from the table and orders them in descending order to get the most recent periods. We limit the result to 12 to cover the last 12 months.
2. **Main Query:**
    - We select the entity_id to group the results by each debtor.
    - We sum the other_debt_burden to calculate the total outstanding balance change for each debtor.
    - We filter the records where asset_and_contingent_class_reason_code is either '2000900005' or '2000900019', which are assumed to represent SM/NPL statuses.
    - We further filter the records to include only those from the last 12 months by checking if the period is in the list of recent periods obtained from the CTE.

This query efficiently calculates the required total outstanding balance changes for the specified debtors over the last 12 months.

Responses for Qualitative Question 10 (translated)

Question: What should the minimum payment rate be for credit card debt repayments?

Multi-agent algorithm's response for Brainstormer

To answer the question about the minimum payment rate for credit card debt repayments, we can use data from Smart_Cube_1, which contains information about credit cards and debt repayments. The analysis involves the following steps:

Step 1: Gather Credit Card Information

- **Fields Involved**: credit_card_type_code, credit_card_type_name, credit_card_parents_account_id, credit_card_card_holder_counterparty_id
- **Reason**: This information helps identify the type of credit card and the cardholder, which is essential for analyzing debt repayments.

Step 2: Gather Debt Repayment Information

- **Fields Involved**: number_of_principal_payment, principal_payment_frequency, interest_payment_frequency, billing_default_interest, billing_fee
- **Reason**: This data provides details on the number of repayment installments, the frequency of payments, and associated fees, which are crucial for calculating the minimum payment rate.

Step 3: Analyze Debt Repayment Data

- **Fields Involved**: default_interest_rate_total, default_interest_rate_additional, default_interest_rate_base_interest_rate_for_calculation
- **Reason**: This information reveals the interest rates charged for defaulting on payments, which is a key factor in determining the minimum payment rate.

Step 4: Calculate the Minimum Payment Rate

- **Method**: Use the data from Steps 2 and 3 to calculate the minimum payment rate, considering the number of installments, payment frequency, and interest rates.
- **Expected Outcome**: An appropriate minimum payment rate for credit card debt repayments.

Step 5: Compare with Standards and Policies

- **Fields Involved**: payment_frequency_condition, provision_amount_undrawn
- **Reason**: This data helps compare the repayment conditions and available provisions with financial institution standards and policies.

Step 6: Summarize and Provide Recommendations

- **Method**: Summarize the analysis results and provide recommendations on the appropriate minimum payment rate for credit card debt repayments.
- **Expected Outcome**: Useful suggestions for setting an appropriate minimum payment rate.

Using data from Smart_Cube_1 in these steps will help effectively analyze and determine the minimum payment rate for credit card debt repayments.

Single-agent algorithm's response for Brainstormer

To answer the question about the minimum payment rate for credit card debt repayments, we can create sub-questions to help analyze and evaluate the appropriate rate as follows:

Analyzing the Current Debt Repayment Ability of Credit Card Holders:

- **Sub-question**: How is the current debt repayment ability of credit card holders, and how has it changed over time?
- **Data Cube**: CUBE_smart_cube_1
- **Method**: Use fields related to asset and contingent classification (e.g., asset_and_contingent_class_name) to analyze the current repayment status of credit card holders and compare it with previous periods (e.g., change_class_mom_level1) to observe trends in repayment ability.
- **Expected Outcome**: Understand the current repayment status of credit card holders and the trends in their repayment ability, which will help assess risk and determine an appropriate minimum payment rate.

Comparing Credit Card Interest Rates with Other Loan Types:

- **Sub-question**: How do current credit card interest rates compare with other types of loans?
- **Data Cube**: CUBE_smart_cube_1
- **Method**: Use fields related to loan interest rates (e.g., interest_rate) to compare credit card interest rates with other types of loans in the financial system.
- **Expected Outcome**: Understand the differences in credit card interest rates compared to other loan types, which will help evaluate the appropriateness of the minimum payment rate.

Analyzing the Impact of Unemployment Rates on Credit Card Repayment Ability:

- **Sub-question**: How does the unemployment rate affect the ability to repay credit card debt?
- **Data Cube**: CUBE_smart_cube_1
- **Method**: Use fields related to unemployment rates and asset classification (e.g., asset_and_contingent_class_name) to analyze the relationship between unemployment rates and the ability to repay credit card debt.
- **Expected Outcome**: Understand the impact of unemployment rates on the ability to repay credit card debt, which will help assess risk and determine an appropriate minimum payment rate.

These analyses will help us evaluate the appropriate minimum payment rate for credit card debt repayments by considering the repayment ability of debtors, economic conditions, and current interest rates.

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Leveraging Generative AI for Granular Credit Data Utilization: A Multi-Agent Approach

Nontawit Cheewaruangroj, Kawinwish Laobundit, Peranut Nimitsurachat, Supachai Saengthong, Skunpoj Thanarojsophon, Anak Yodpinyanee

Data Management and Analytics Department

Bank of Thailand

18-20 February 2025

Prepared for the 4th IFC workshop on "Data science in central banking", Rome, Italy

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Regulatory Data Transformation (RDT)

A transformative redesign from static Data Management System to granular-level regulatory reporting standard for credit data

Rich dataset with 1000+ fields and over 25 million loan accounts per month, including credit lines, interest rate plan, and outstanding amount

Multi-faceted data architecture with specialized cubes for diverse and thorough credit data analysis

Enable insights for analyst and examiners across many domains from bank supervision to monetary policies

Complex data requires knowledge and advanced data analytical skills

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Challenges with RDT Utilization

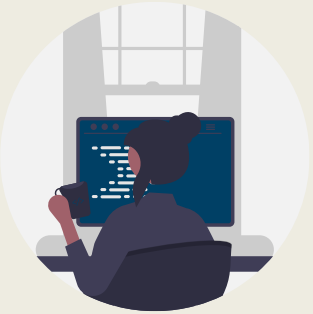
Problem translation

Users may not be able to translate business questions into data queries

Data understanding

RDT's data is complex

- Numerous data fields
- Structured into multiple "cubes" for different usages



Coding skills

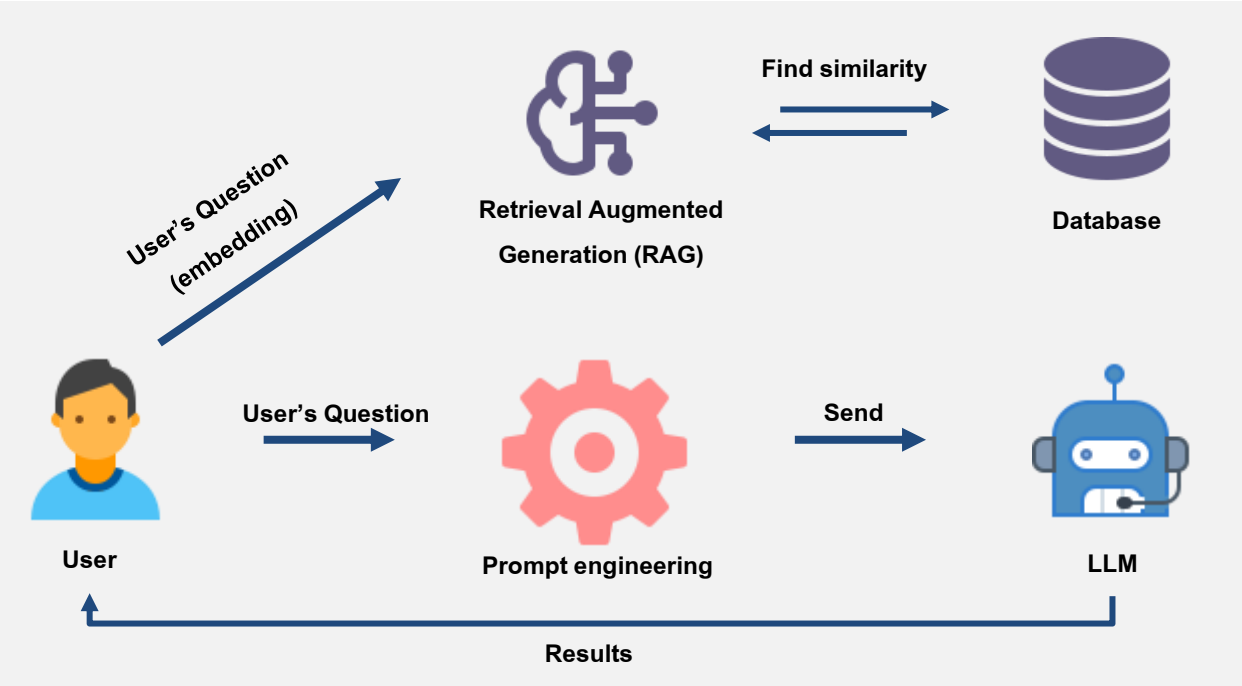
Require SQL / python knowledge to wrangle granular data effectively

- Most analysts and examiners are not proficient in coding

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Generative AI as RDT assisting agents

- **Large Language Model (LLM)** can be utilized as a copilot for answering users' questions.
- **Retreival-augmented Generation (RAG)** is used to provide internal knowledges such as metadata to GenAI .
- **Prompt Engineering** instructs GenAI to produce SQL code based on questions and retrieved metadata.



standard GenAI with RAG workflow

Ask a plain language question about RDT

Field selection

SQL query drafting

SQL

```sql
SELECT
  customers.name,
  SUM(loans.amount) AS total_loan_amount
FROM customers
INNER JOIN loans ON customers.customer_id = loans.customer_id
GROUP BY customers.name
ORDER BY total_loan_amount DESC
LIMIT 3;
```

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Challenges with GenAI workflow for RDT

	Challenges	Impacts
Complex operation	Standard workflow might struggle with multiple instructions for more complex tasks	Limit potentials for RDT utilization to basic querying
Context size	Retrieving metadata from all tables in RDT may exceed the context size of LLM	Unable to handle complex query that requires multiple tables or scale to larger data sources
Mixing LLMs	Cannot mix different models for different tasks to optimize cost	Consume more resources with running single large model for all tasks

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Multi-agent framework

Single-agent Framework

A large agent capable of executing diverse tasks comprehensively.

Analyst

Data analytics tasks
- Analyze questions
- Retrieve data
- Select fields
- Write SQL

Multi-agent Framework

A multi-agent approach decomposes the task into subtasks to be executed by different agents.

Manager
Orchestrate

Data Engineer
Retreive data

Analyst
Analyze data

Secretary
Communicate and assist user

Data Analyst
Write SQL

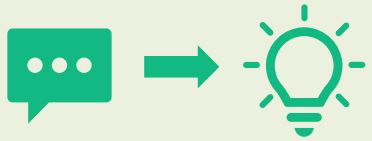
Senior Analyst
Analyze questions

Framework	Pros	Cons
Single-agent	• Easier to manage • Lower complexity to develop • Consume less resource	• Limit expertise for complex tasks • Limit on context size • Less flexibility • No steps provided to users
Multi-agent	• Able to deal with more complex tasks • Less limitation on context size • Assign different task to agents with specialities • Users can see broken-down steps	• More complexity • Consome more resource

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

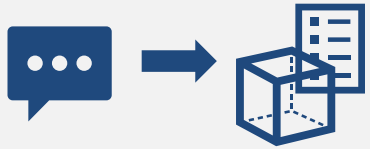
RDT Copilots: Modules

Brainstormer



Translate user's problem into tangible statements and identify relevant data sources

Metadata Copilot



Search for the best data cube and fields required to answer users' questions

SQL Coder

Assist users by converting natural language questions in to SQL query

Q: What should be the minimum payment rate for credit card debt repayment?

Brainstormer:

Find relationship between *credit_card_type*, *first_payment_amount* and *total_interest_and_fee_rate* to evaluate relationship between risks and payment.

…

Q: How many debtors have joined the debt restructuring program this year?

Metadata Copilot:

Field	Description
entity_id	Id for debtors
debt_restructuring_date	Date of debt restructuring
…	…

SQL Copilot:

```
SELECT
    COUNT(distinct entity_id) AS number_of_debtors
FROM CUBE_1
WHERE debt_restructuring_date > '2024-01-01'
```

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Brainstormer

Users

Information Gathering Agent

Select data cube and analyse the questions

Field Finder Agent

Select related fields from each chosen data cube

Question Breaker Agent

Analyze information and provide step-by-step answer to user's questions

Follow Up Agent

Ask follow-up question to guide the discussion forward

* Each agent uses GPT-4o mini

New Chat

Task:

- (•) RDT Brainstorming
- ( ) RDT Copilot - Metadata
- ( ) RDT Copilot - SQL Coder

Select Cube

Chat History

Report / Feedback

Change Password

Logout (nontawic)

● LIVE

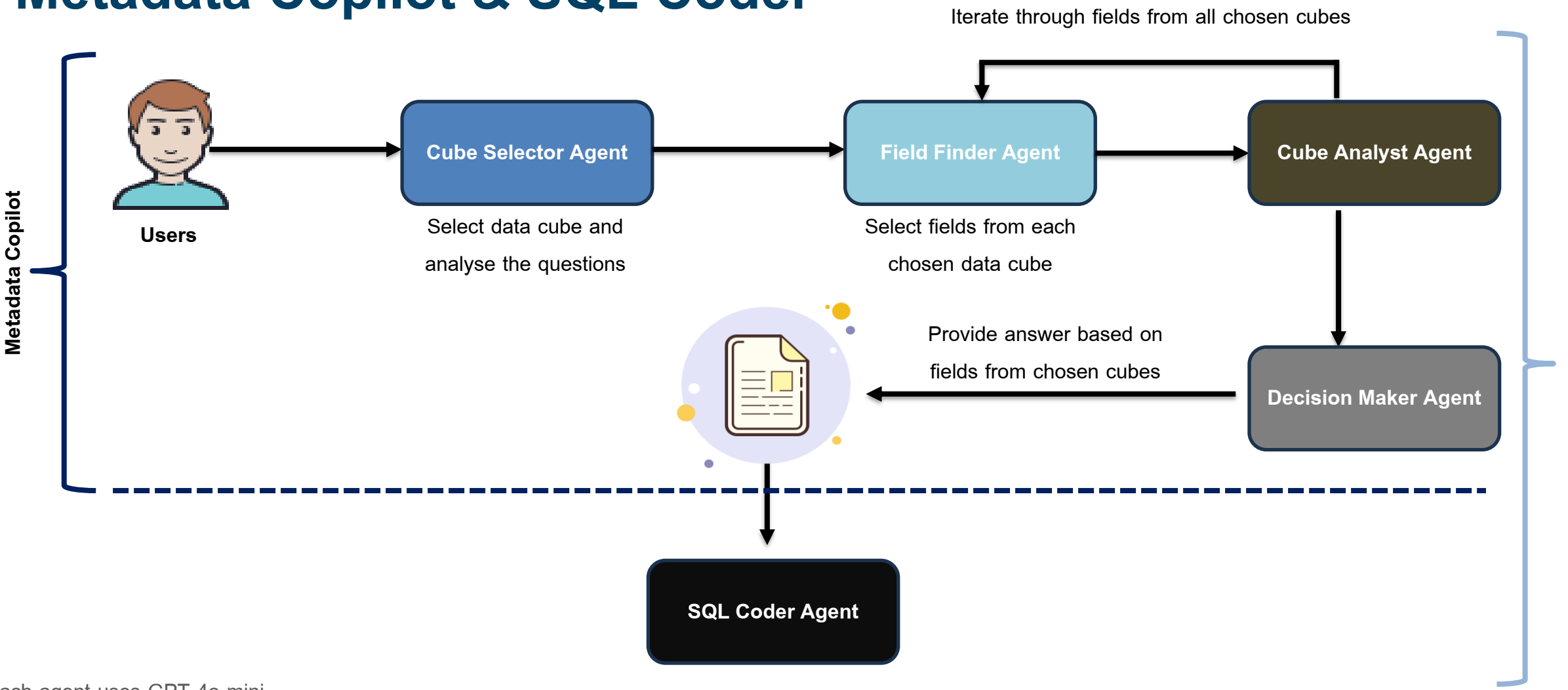
Hi Nontawit Cheewaruangroj! I am BotGPT, ready to provide assistance.

Input Question: I want to determine appropriate criterias for debt restructuring program.

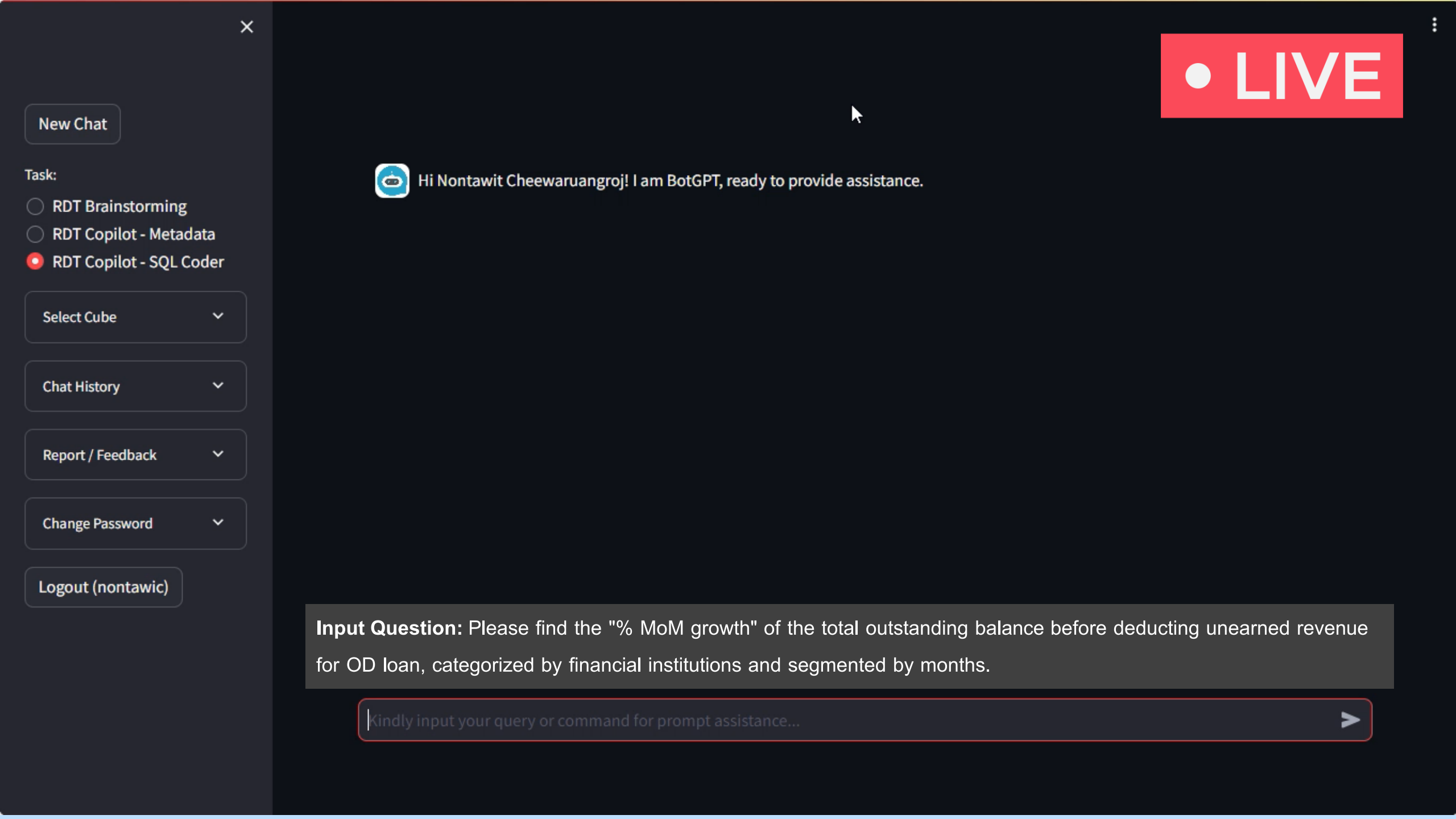
Kindly input your query or command for prompt assistance...

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Metadata Copilot & SQL Coder



* Each agent uses GPT-4o mini


New Chat
Task:
RDT Brainstorming
RDT Copilot - Metadata
RDT Copilot - SQL Coder
Select Cube
Chat History
Report / Feedback
Change Password
Logout (nontawic)
LIVE
Hi Nontawit Cheewaruangroj! I am BotGPT, ready to provide assistance.
Input Question: Please find the "% MoM growth" of the total outstanding balance before deducting unearned revenue for OD loan, categorized by financial institutions and segmented by months.
Kindly input your query or command for prompt assistance...

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Evaluation

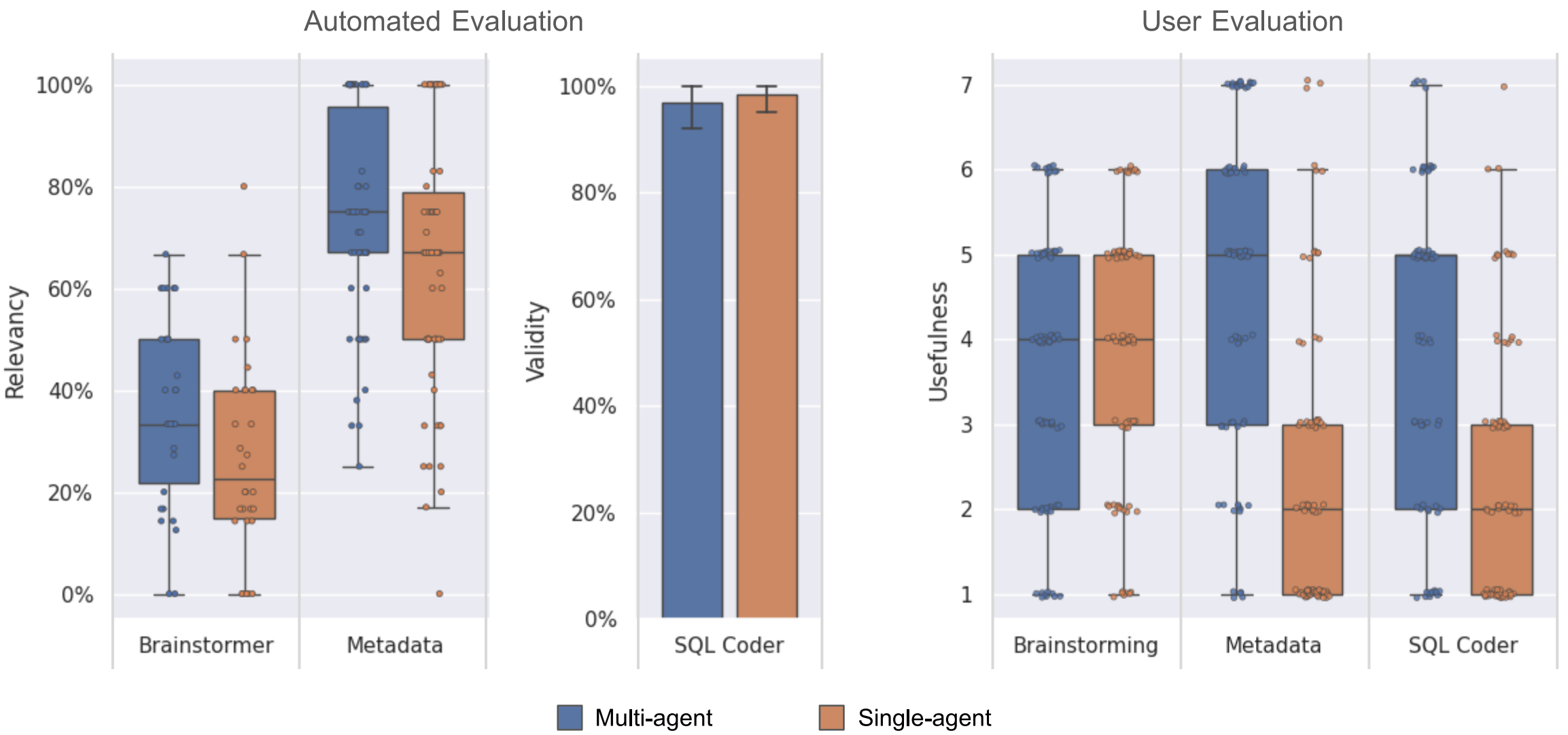
Experiment: Compare the performance of each module against the *single-agent* versions.

Metrics:

Module	Automated evaluation	Human evaluation
Brainstormer	**Relevancy** (Recall of expected fields required for business problem solving)	**Usefulness** (How useful the output for answering the question) - Likert scale of 1 to 7 - Evaluated by BOT's examiners & analysts
Metadata Copilot	**Relevancy** (Recall of expected fields required for querying data)	
SQL Coder	**Validity** (Whether the generated SQL syntax is valid, i.e., executable)	

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Results from evaluation



* For automated evaluation, Brainstormer is evaluated with 30 qualitative questions. Metadata Copilot and SQL coder are evaluated with 62 quantitative questions.
** For user evaluations, each module is evaluated with 10 questions by 8 analysts / examiners.

Discussion

Automated evaluation

- RDT Copilots can perform objective tasks, such as retrieving fields and writing SQL, very well.
- Multi-agent framework show good improvements in retrieving correct fields. For SQL generation, single-agent framework is already good.

User evaluation

- Users find multi-agent Metadata Copilot and SQL Coder useful for answering quantitative questions.
- However, users do not find Brainstormer particularly useful for qualitative questions. Multi-agents framework does not show improvements over single-agent framework.
- Users also show different preferences towards writing styles of Brainstormer versions. Results are varied for different questions for all modules.

Conclusion

GenAI Tools can be useful in helping RDT data utilization.

- RDT Copilots can help analysts and examiners find and retrieve required data.

Multi-agent framework can be applied to enhance RDT Copilots:

- Help address issues of **limited context length** for multiple tables data.
- Improve **performance** over single-agent approach for some tasks, especially in more objective tasks that require large contexts.
- Can be adopted for **variety of tasks**, both objective (coding) and subjective (brainstorming).

ธนาคารแห่งประเทศไทย
BANK OF THAILAND

Challenges and Next Steps

Challenges

- Incomplete metadata. Internal jargons.
- Some domain knowledge not in documents or LLM knowledge.
- Difficult to get users engagement in designing and adoption.
- Difficult to experiment with different LLMs due to compliance & security concerns.

Further Work

- Improve data dictionary. Enhance prompt to incorporate more domain knowledge.
- Brainstormer needs a revision to produce more useful outputs.
- Explore LLMs with improved reasoning capabilities such as OpenAI o1/o3 model.
- Incorporate ability to directly process data, e.g., plotting graph.
- Explore multi-agent solutions for problems such as KM chatbots, supervision assistant agents.

Thank You

Contact: nontawic@bot.or.th