**Private GPT For Personalized Finance Management (PFM)**

**Introduction**

Efficient data retrieval is essential in financial technology, where timely and accurate information can significantly impact decision-making. This project focuses on the integrating a large language model with Lang Chain to translate natural language queries into SQL commands, allowing users to interact seamlessly with complex databases. By harnessing advanced natural language processing capabilities, the system enhances user accessibility and improves the accuracy of financial data retrieval.

**Objective**

To develop an AI-powered chatbot, PrivateGPT, that integrates local and public datasets to provide personalized finance management services.

### **Experimental Setup**

#### Platforms Used:

* **Google Colab**: A cloud-based platform providing free access to computational resources.
  + **Google Colab Free Tier Specifications**:
    - RAM: Approximately 12.68 GB
    - Disk: Approximately 67.2 GB
    - **GPU**: Intel Xeon CPU @2.20 GHz
      * GPU Memory: 15 GB GDDR5
      * Compute Capability: 7.5
* **VS Code**: An integrated development environment (IDE) used for code development and debugging.

#### Local Machine Specifications:

* **RAM**: 12 GB
* **Disk**: 512 GB SSD

**Database Used:**

* **MySQL Workbench**: An integrated tool for MySQL database administration and management, used for storing and managing the bank transaction data.

**Methodology**

**1. Data Collection and Preparation**

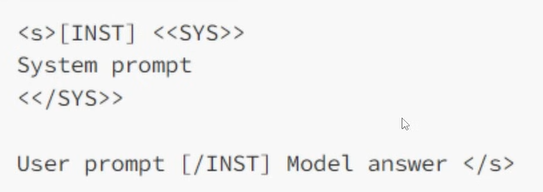
The objective was to acquire, clean, and preprocess data for subsequent analysis and model training. A sample dataset of 116,000 bank transactions in CSV format served as the data source. The CSV file underwent a cleaning and preprocessing stage to ensure data quality and consistency. The cleaned dataset was then loaded into MySQL Workbench for efficient data management.

**2. Large Language Model (LLM) Setup**

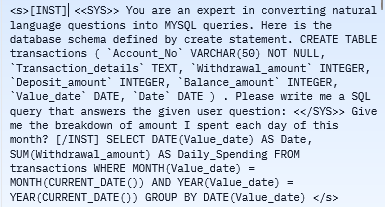
The Llama-2-7b ("meta-llama/Llama-2-7b-hf”,[link](https://huggingface.co/meta-llama/Llama-2-7b-hf)) model from Hugging Face was used for the project. The model is specifically designed for natural language understanding and generation as it excels in various task such as question answering, summarization and text generation. The model utilizes the capabilities of the Hugging Face Transformer library, allowing for seamless integration and fine tuning within the Hugging Face ecosystem.

**3. Training Dataset Generation**

To train the model, a custom dataset was created ([link](https://huggingface.co/datasets/prabal123/nl-sql-llama-2format-2/viewer?row=27)) which included SQL schema, instructions, and corresponding SQL queries. The Standard data format used for fine tunning the llama 2 model is of the following format.



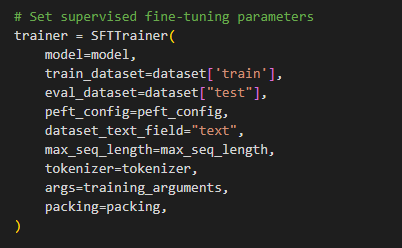
Considering this standard format, the custom data was also made accordance with the same format.Here is a sample of the data format .



**3. Training the Model**

Directly fine-tuning all parameters of large models (all 7 billion parameters in Llama-2) is costly and resource-intensive. Therefore, Parameter-Efficient Fine-Tuning (PEFT) strategies are used for optimal fine-tuning with fewer parameters. This technique involves freezing the pre-trained model weights (Llama-2 7B) and fine-tuning with a smaller set of parameters.

**Low-Rank Adaptation (LoRA)** is a novel technique for fine-tuning large language models (LLMs) that significantly reduces the number of trainable parameters while maintaining or even improving their performance. This is achieved by injecting smaller, trainable matrices into each layer of the LLM’s architecture, rather than directly modifying the original weights.



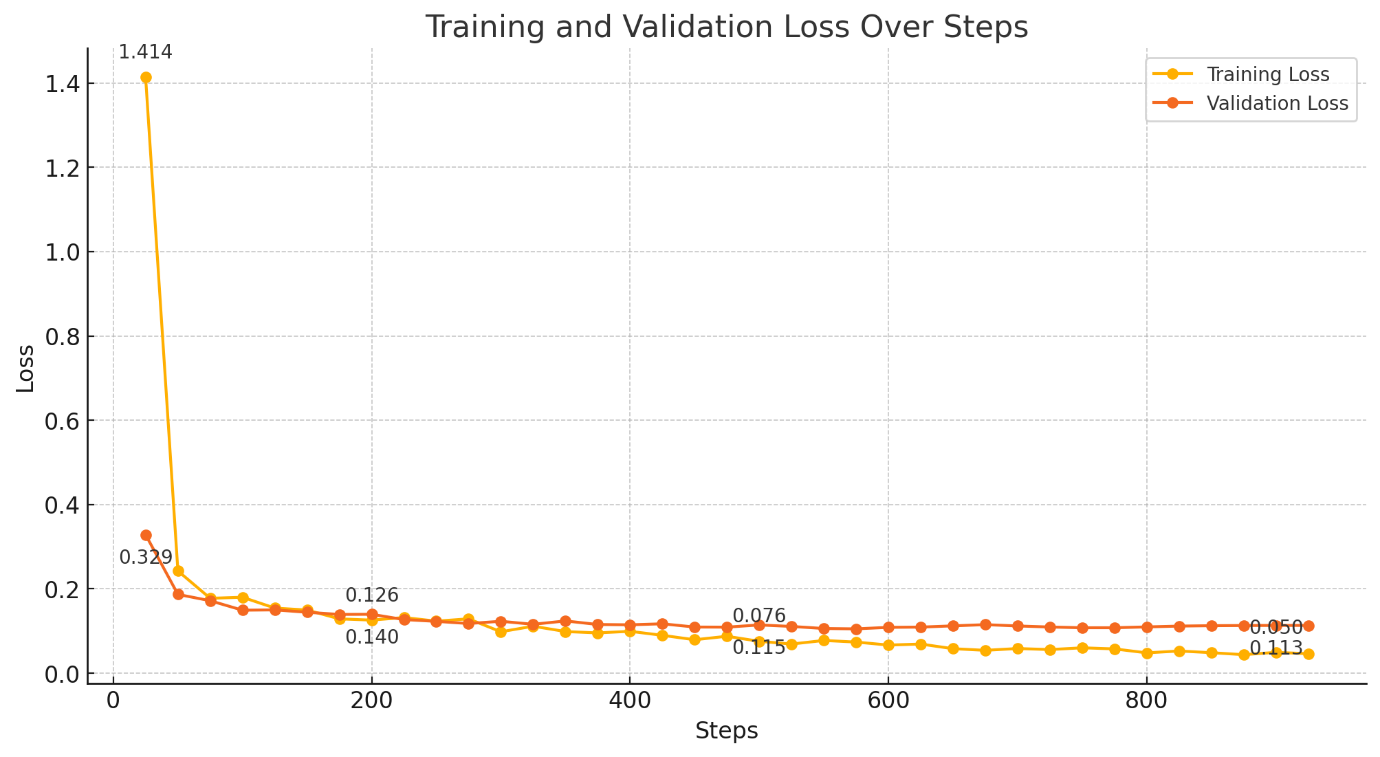
The model was trained on two different dataset. The first dataset consists of custom created user natural language queries related to finance and their corresponding SQL queries. This dataset was crucial for helping the model accurately respond to user queries, thereby minimizing LLM hallucinations.

Since the data consisted of a relatively small number of data (around 350), determining the optimal number of epochs for training the model was challenging. Higher Epoch could cause the LLM model to memorize the data rather then learn patterns, while a lower epoch might not be enough for model to identify the relationships and patterns in data. So training was conducted multiple times with varying hyper parameters such and epoch, batch size, learning rate ,learning scheduler, weight decay, rank ,scaling factor etc to identify the best result for the LLM .

Ultimately, following configuration was chosen to train the model because this configuration performed better compared to other epoch settings. You can view the experiments related to varying epoch configurations on the [HuggingFace Account](https://huggingface.co/models?sort=trending&search=prabal123) .

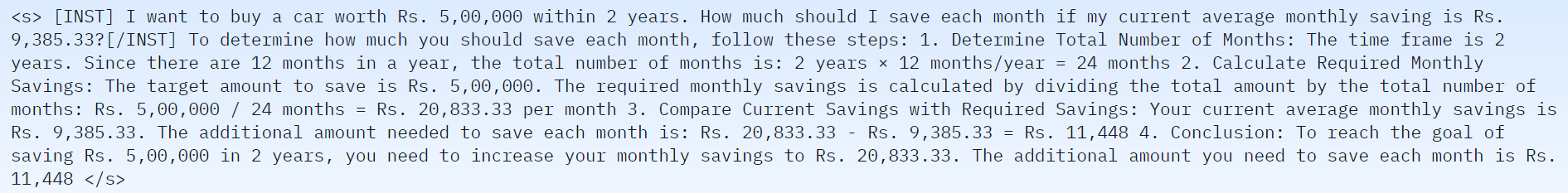
The Configuration:

* Learning Rate: 2e-4
* Epochs: 6
* Batch Size: 2
* Rank: 64
* Scaling Factor (lora\_alpha): 16
* Optimizer: paged\_adamw\_32bit
* Weight Decay: 0.001

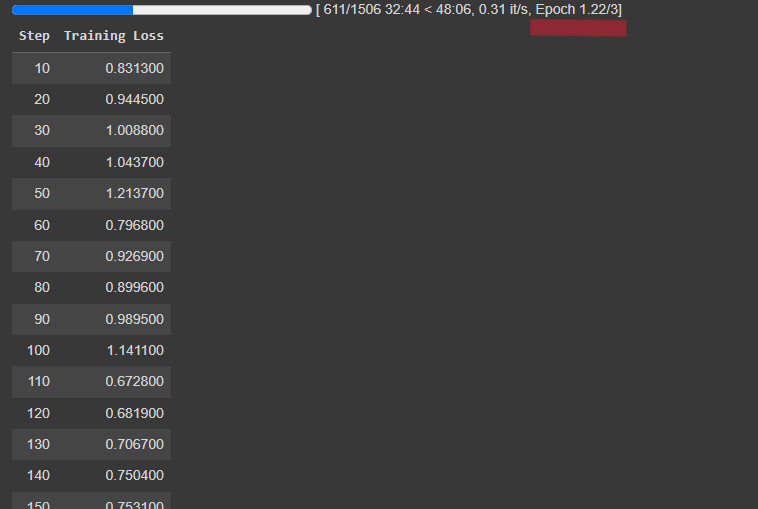


The above diagram illustrates the training and validation loss during model training. Overfitting occurs when a model learns the training data too well, capturing noise and details that do not generalize to new, unseen data. This typically results in a significant gap between training loss (which decreases) and validation loss (which increases). In contrast, the diagram shows no signs of overfitting, as the training loss and validation loss remain close to each other throughout the process, with no significant fluctuations in the validation loss. This suggests that the model generalizes well to new data and has not excessively memorized the training examples.

The second dataset, which included 1,500 records, focused on reasoning ([link](https://huggingface.co/datasets/prabal123/Logical-Reasoning-1000-Data)), designed to help the LLM break down complex questions into smaller, manageable tasks. This approach aims to enhance the model's ability to provide precise and contextually appropriate answers, reducing errors and improving overall performance. The data format was as follows:

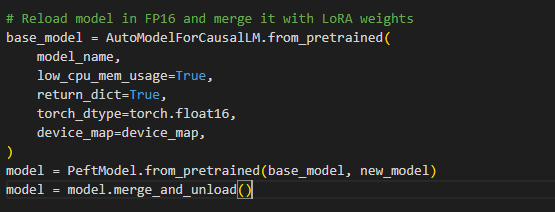


The data format included detailed problem statements, intermediate reasoning steps, and final answers, allowing the model to learn not only the solution but also the process of arriving at it. This comprehensive format supports the development of a more robust and nuanced understanding of complex queries. Each record was crafted to ensure that the model could understand and replicate the reasoning process, that helps in improving its capability to handle similar complex questions in real-world scenarios. This structured approach allows for a better grasp of logical sequences and decision-making patterns.



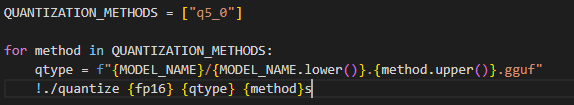
4.Merging the Trained model with Base model

After fine tunning using the PEFT technique , the next step involved merging the fine tuned model with the base model . The result is a unified model that preserves the general knowledge of the base model while incorporating specialized improvements. This approach ensures that the final model benefits from both the broad capabilities of the base model and the specific advancements obtained through targeted fine-tuning.



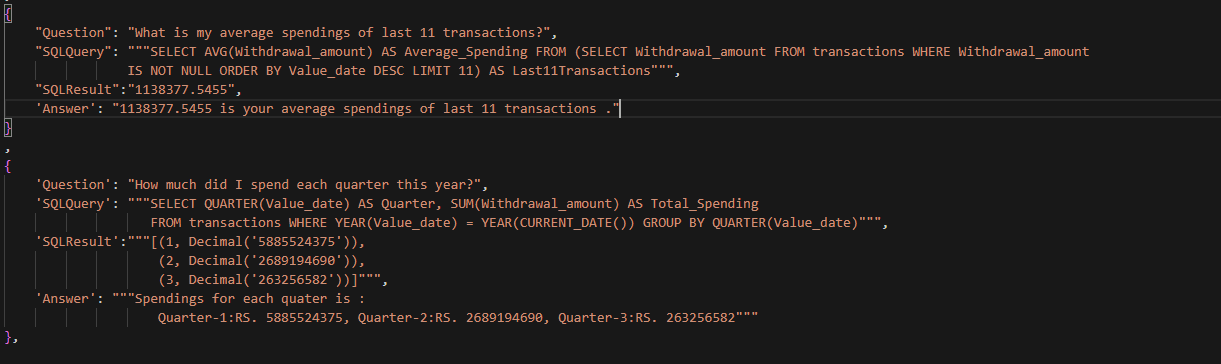
**5. Model Conversion and Storage**

The trained LLM model was not directly suitable for inference due to its original format. Therefore, it was quantized into 5-bit(q5\_0) GGUF (GPT-Generated Unified Format), which is specifically designed for efficient inference on low computational devices. Subsequently, the model was stored in the Hugging Face repository, making it available for local download and further use. ([link](https://huggingface.co/prabal123/Original-Metallama-5Epoch-Graphofloss-GGUF/tree/main))



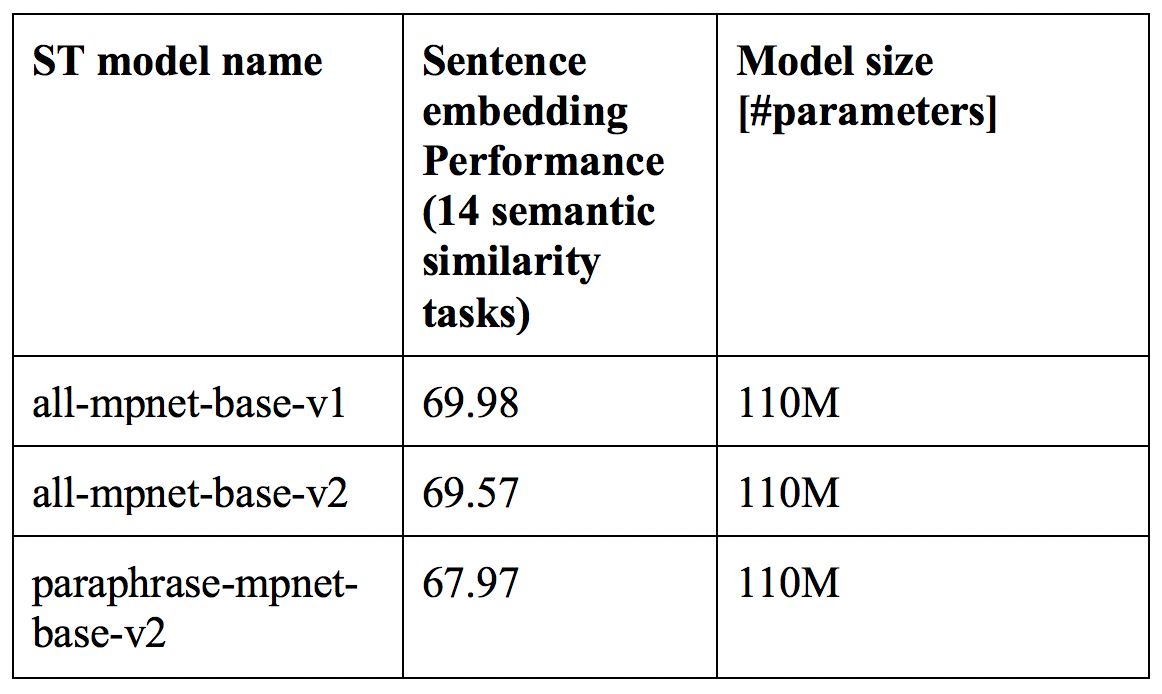
**6. Database Connection and RAG(Retrieval Augmented Generation)**

A connection was established with MySQL Workbench, where the actual transaction data was stored. During the implementation, a few-shot prompt template was created as part of the RAG (Retrieval-Augmented Generation) Architecture. This template included a diverse range of finance-related questions that users might typically ask, as well as examples of natural language questions where the LLM had previously failed to generate accurate responses. Each entry in the few-shot template comprised the user’s question, the corresponding SQL query, the result from MySQL Workbench, and the accurate answer. This comprehensive few-shot template served as a reference for the LLM when generating SQL queries for new, unseen natural language questions. By including both previously failed queries and a broad spectrum of finance-related queries, the template provided a robust foundation for improving the LLM’s response accuracy.



**7. Embedding and Vector Store Creation**

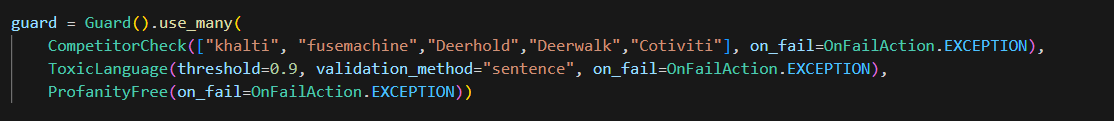
The few-shot template was embedded using the "all-mpnet-base-v2" sentence transformer from Hugging Face, which excels in semantic search compared to other embeddings. The resulting embeddings were then stored in the Chroma DB vector store. When a new user question is posed, the system performs a semantic search to identify the top 2 most relevant questions from the vector store. For each of these top 2 questions, the corresponding SQL query, result, and answer are retrieved. The LLM then generates a response for the new user’s unseen question by referencing these top 2 similar questions, their SQL queries, results, and answers, thereby improving the accuracy and relevance of the generated response.



**8. Additional Information and Guard Rails Implementation**

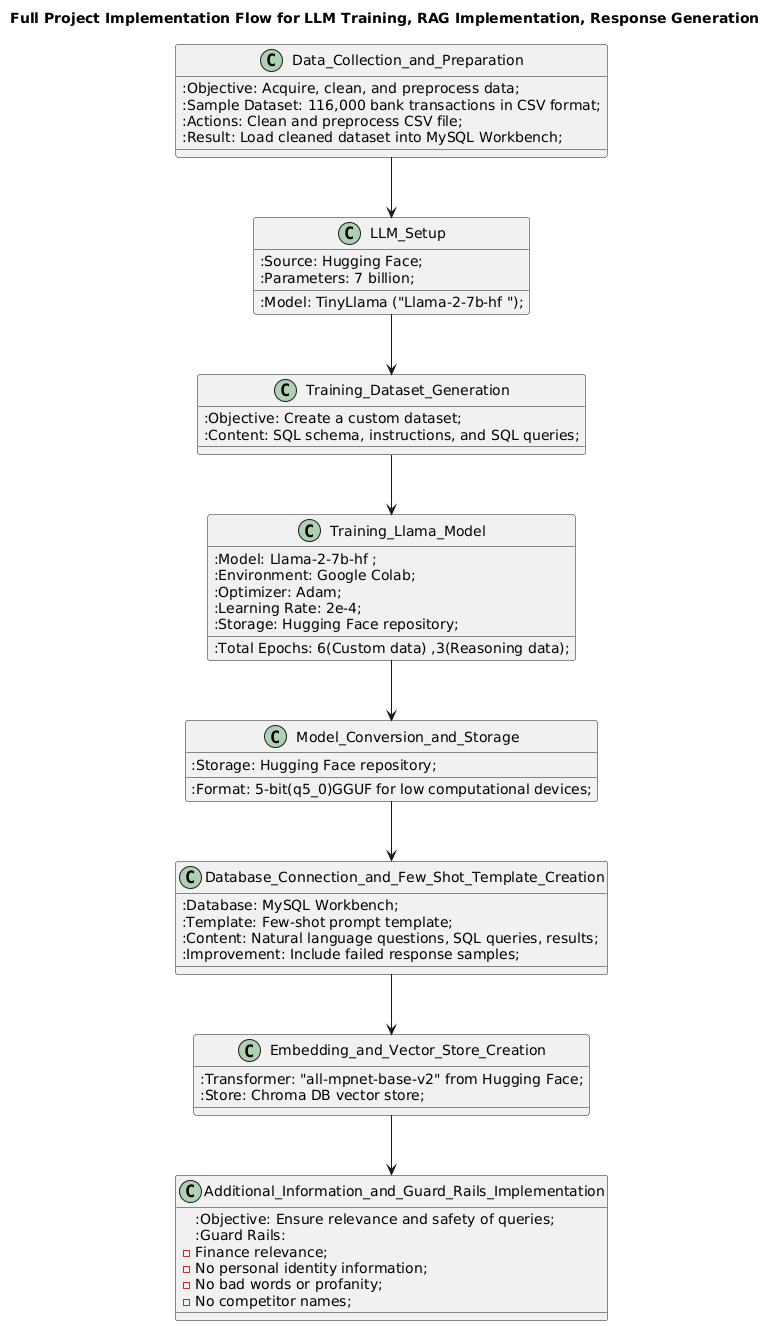
The LLM was provided with additional necessary information about the database schema to ensure that the model does not use unnecessary columns that were not present in MySQL. User questions (in natural language) were taken as input, and necessary guard rails were implemented to ensure the relevance and safety of the queries. These guard rails ensured that:

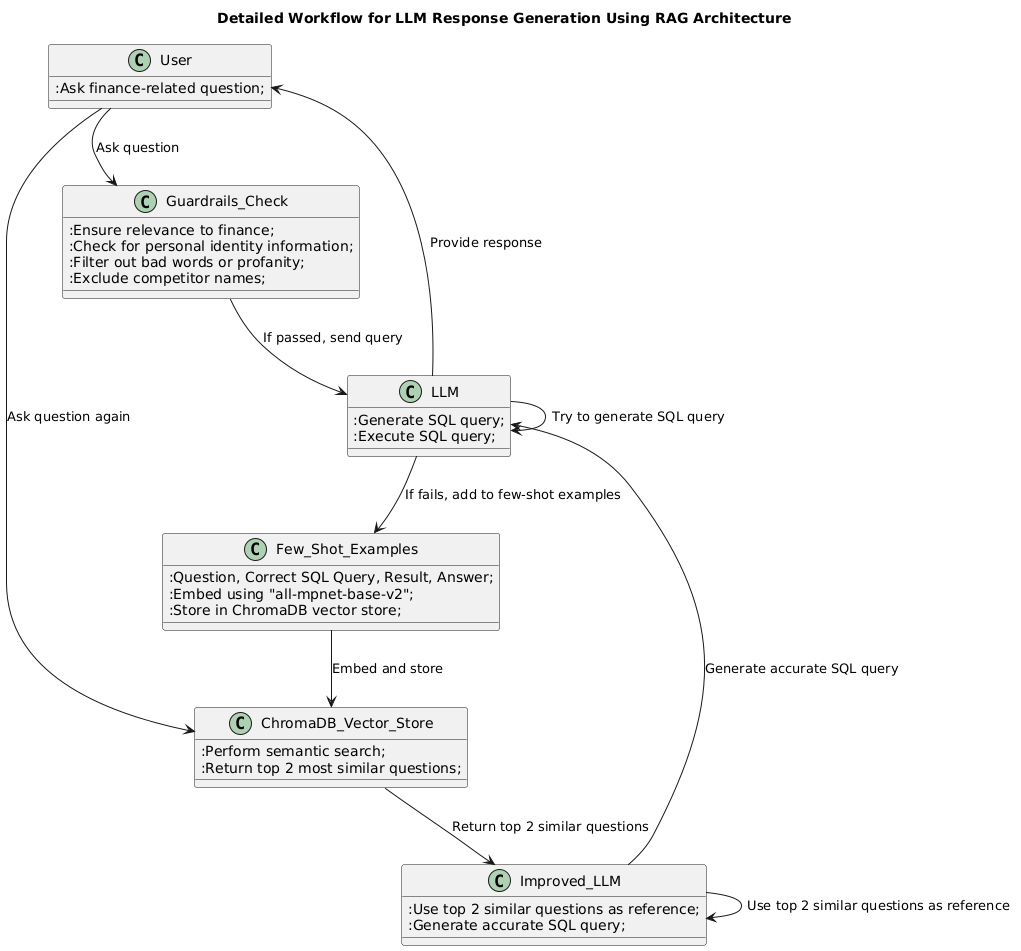
* User questions were relevant to finance.
* User questions did not contain any personal identity information such as bank account numbers.
* User questions did not contain any bad words, toxic language, or profanity.
* User questions did not involve competitor companies names .



If the user question passed the guard rails, it was given to the LLM to generate the SQL query, execute the corresponding query in MySQL Workbench to retrieve the relevant data, and respond with the accurate answer.

**Project Implementation Flow Diagrams**

****

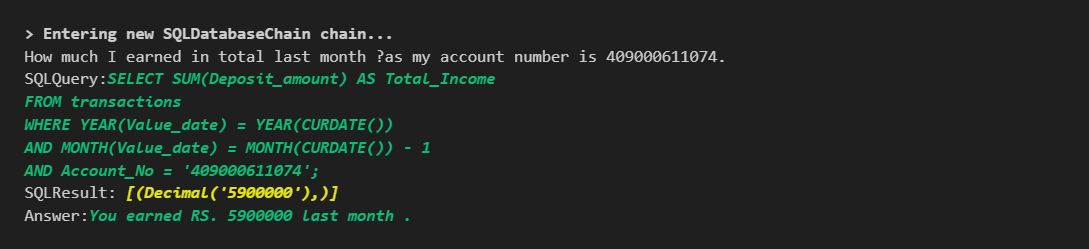


**Results**

The response of the LLM model in generating SQL queries from natural language inputs was satisfactory, effectively retrieving relevant data from the MySQL database.

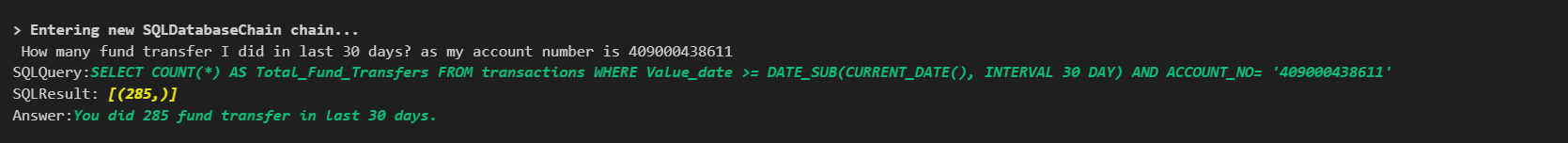
**1. Possible Descriptive Statistics to deliver :**

Question 1: How much I earned in total last month?



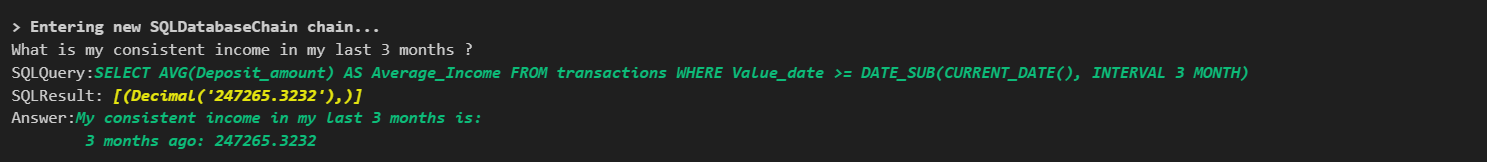
Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 2: How many fund transfer I did in last 30 days?



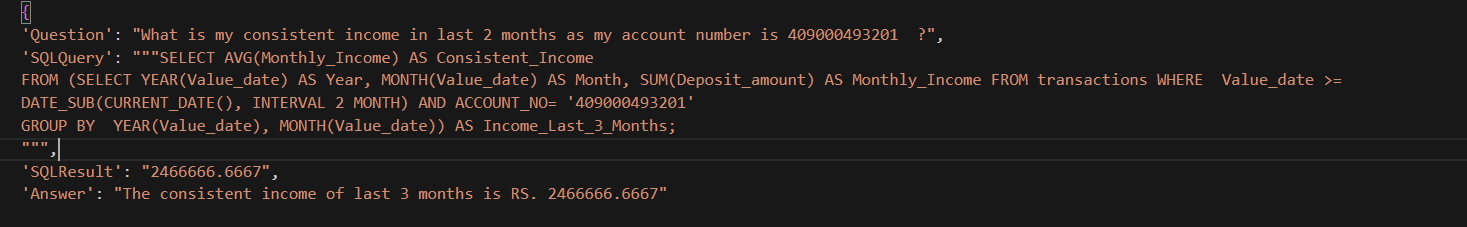
Status: Successful query generation, execution, fetching, and response with correct sentence formation.­

Question 3: What is my consi­stent income in my last 3 months?

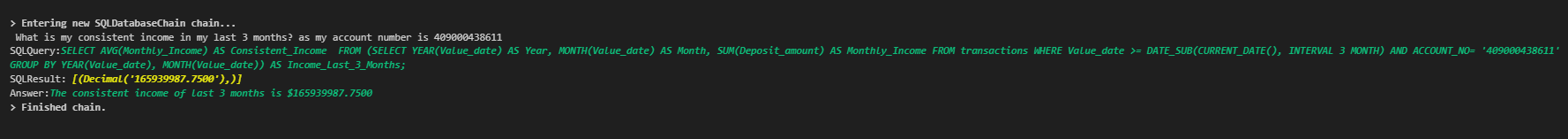


Status: Failed to generate correct SQL Query .The SQL query generated by llm calculates the average deposit amount across all transactions in the last 3 months, not the average monthly income (Assuming the "consistent income" refers to some measure of stability or average income over a period).

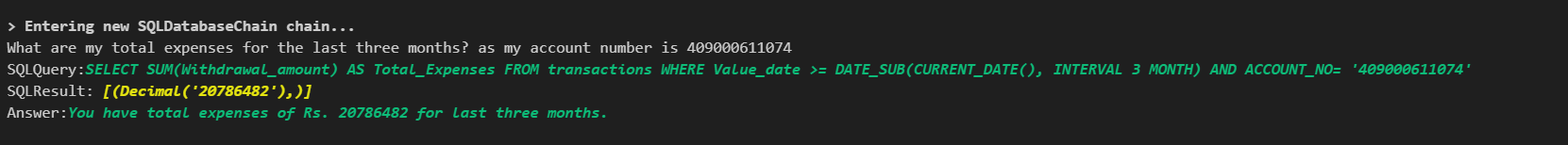
So we added the Question along with its corresponding correct SQL query, SQL Result and proper format of response in few-shot learning template.



Next time the same question was asked, the model was able to respond accurately.

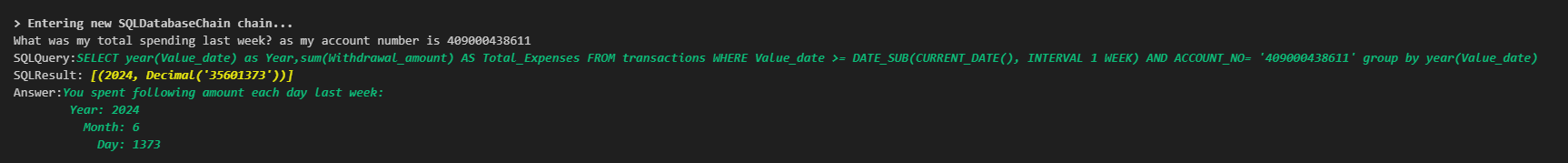


Question 4: What are my total expenses for the last three months ?



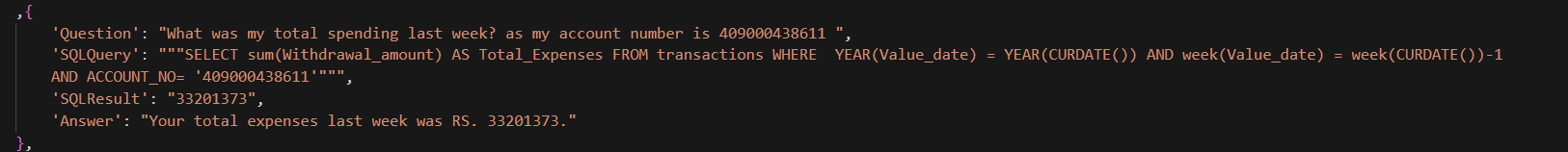
Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 5: What was my total spending last week?

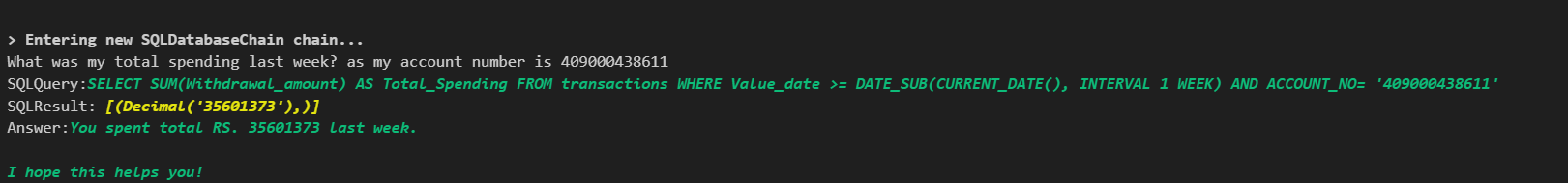


Status: Successful query generation, execution, fetching, but there was Error in sentence formation while responding.

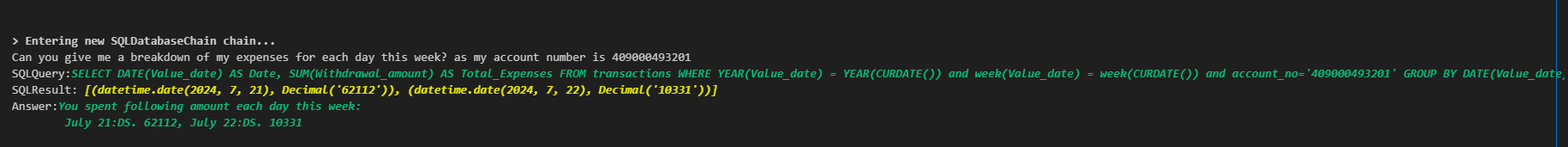
So we added the Question along with its corresponding correct SQL query, SQL Result and proper format of response in few-shot learning template.



Next time the same question was asked to the model and it was able to respond accurately.

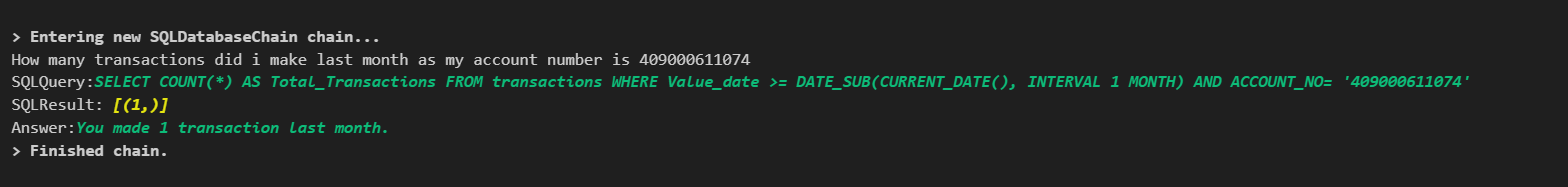


Question 6: Can you give me a breakdown of my expenses for each day this week?



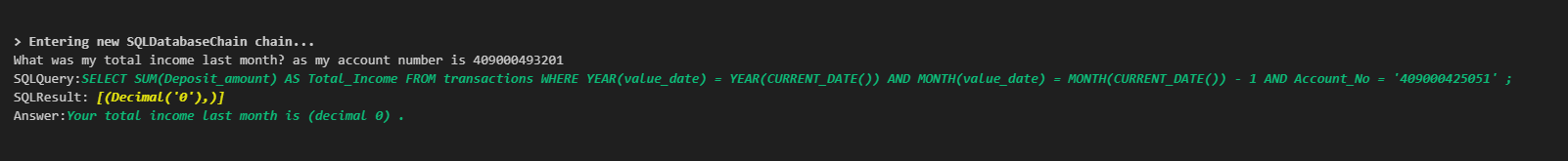
Status: Successful query generation, execution, fetching, and response with correct sentence formation. (Note : Instead of Rupees (RS) it displayed DS).

Question 7: How many transactions did I make last month?



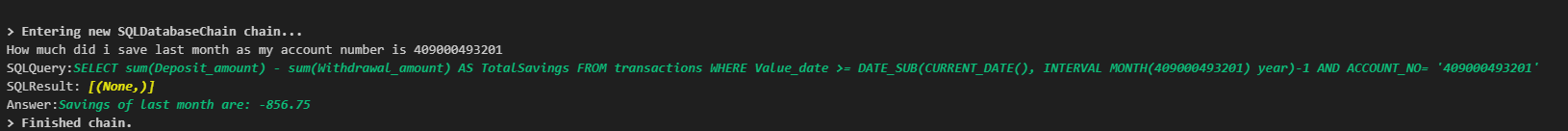
Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 8: What was my total income last month?



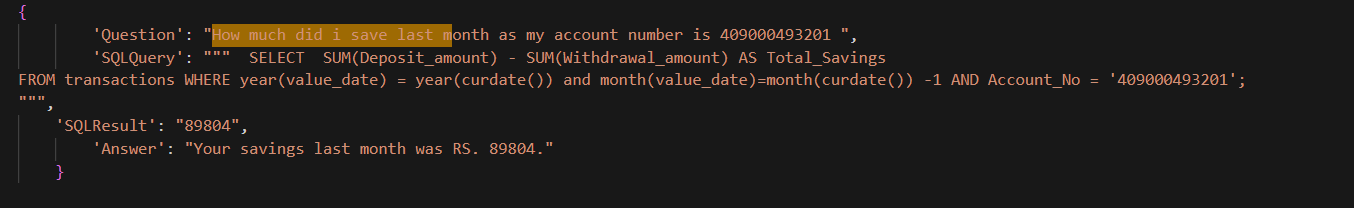
Status: Successful query generation, execution, fetching, and response with correct sentence formation. (Note : Amount is 0 since there was no income last month for the user).

Question 9: How much did I save last month?

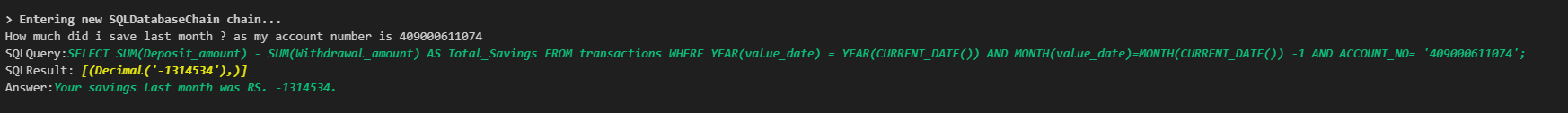


Status: Error in Generation of correct SQL query.

So we added the Question along with its corresponding correct SQL query, SQL Result and proper format of response in few-shot learning template .

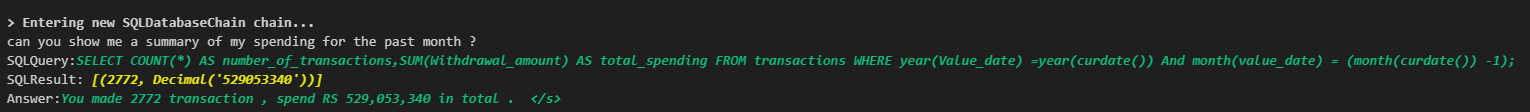


Next time the same question was asked to the model and it was able to respond accurately.



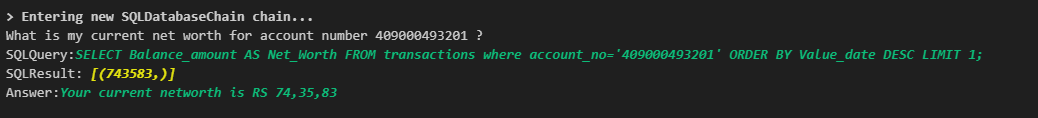
**2. Personal Finance Management**

Question 2.1: Show me a summary of my spending for the past month.



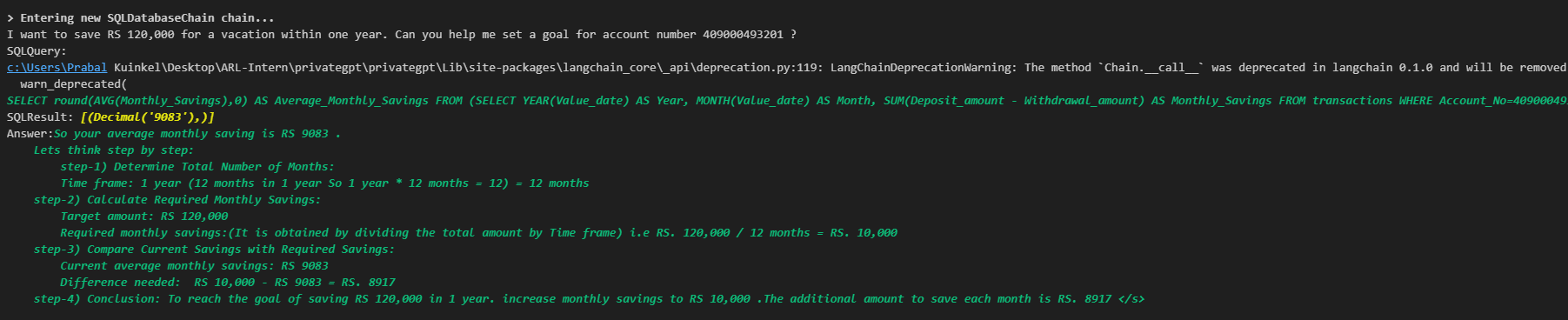
Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 2.2: What is my current Net worth?



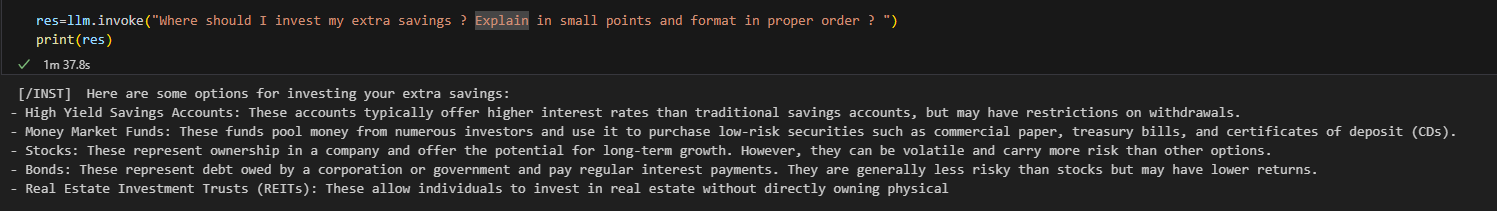
Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 3.1: I want to save RS 120,000 for a vacation within a year. Can you help me set a goal ?



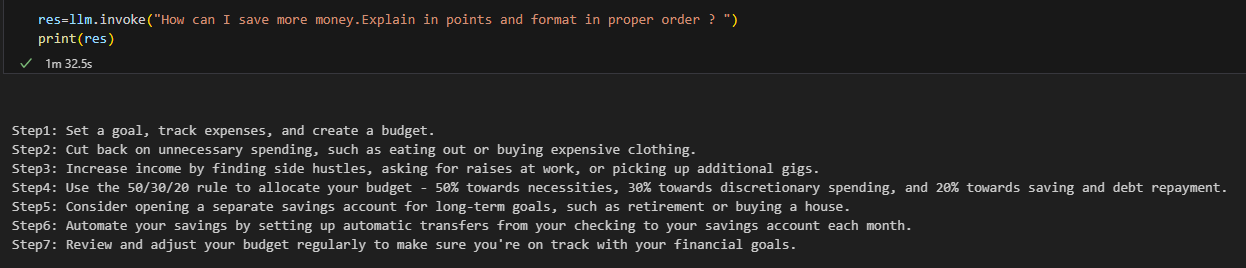
Status: Successful query generation, execution, fetching, but calculation mistake since 10,000 – 9083 is 917 (as model responded 8917)

Question 3.2 Where Should I invest my extra savings ?



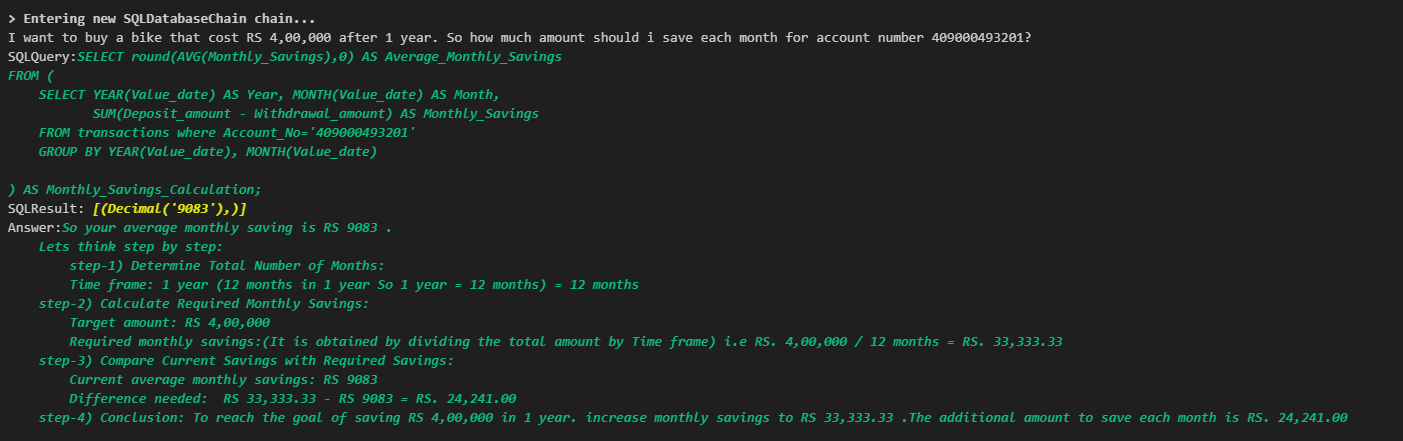
Status: Satisfactory response

Question 6.2 How can I save more money ?



Status: Satisfactory response

Question 8.1: I want to buy a bike that cost RS 4,00,000 next year. How should I plan for it ?



Status: Successful query generation, execution, fetching, and response with correct sentence formation.

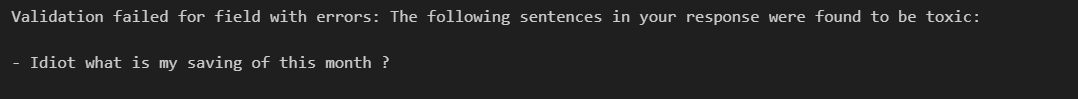
**3. Test for Guard Rails Implementation**

Question 1: Does cat bark like dog?



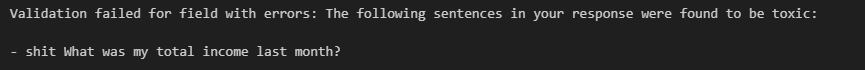
Status: Since the question asked by user was irrelevant, the question was discarded.

Question 2: Idiot what is my saving of this month?



Status: Since the question asked by user contains toxicity, the question was discarded.

Question 3: Shit What was my total income last month?



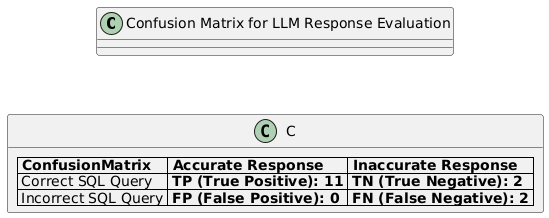
Status: Since the question asked by user contains toxicity, the question was discarded.

Question 4: Hey tell me detailed description of fusemachine.



Status: Since the question asked by user contains information about other companies, the question was discarded.

### **Evaluation metrics**



### Explanation:

### TP (True Positives): Correct SQL query generation with an accurate response (6 instances).

### TN (True Negatives): Correct SQL query generation with an inaccurate response (1 instance).

### FN (False Negatives): Incorrect SQL query generation with an inaccurate response (2 instances).

### **Conclusion**

The project successfully demonstrated the integration of a large language model with Lang Chain to facilitate efficient data retrieval from natural language inputs, enabling intuitive access to financial data. The system effectively bridged the gap between technical complexity and user accessibility, showcasing the potential for real-world applications in the finance sector.

While the model performed satisfactorily, there remains room for improvement in handling complex queries and optimizing response times. Future work could focus on refining these areas to enhance user experience further.

Overall, this project sets the foundation for developing intelligent financial data retrieval systems that empower users to make informed decisions seamlessly.

**References**

1. Guardrails AI. (n.d.). *Hub - Guardrails AI*. from <https://hub.guardrailsai.com/>

2. Substratus AI. (2023, July 15). *Converting Hugging Face models to GGUF models*. from <https://www.substratus.ai/blog/converting-hf-model-gguf-model/>