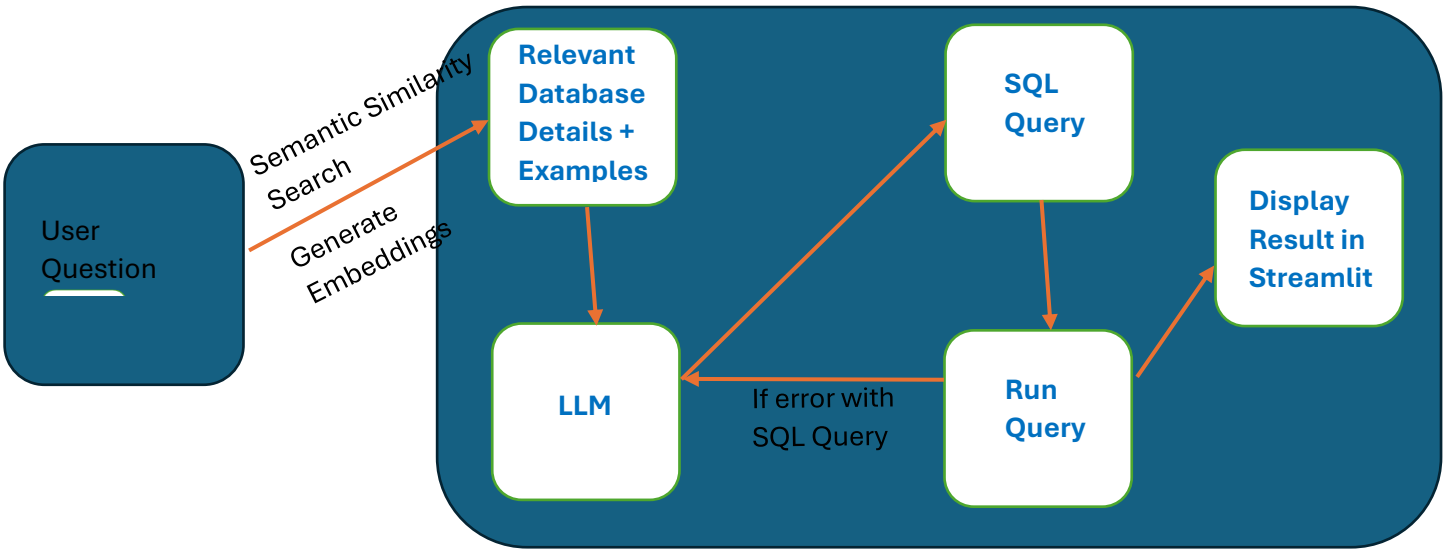
Jio Internship Report:

By Archit Lakhani

I have implemented a SQL Chat Bot that converts Natural Language Text to SQL Queries using an LLM from the Ollama Library of LLMs. In this project, I utilized Streamlit for creating the web interface, Pandas for data manipulation, LangChain Community's Ollama LLM for language model integration, SQLDatabase for database management, Pygwalker for visualizing data insights, inflect for converting numbers to words, LangChain's ChatPromptTemplate and StructuredOutputParser for prompt engineering and structured output parsing, OllamaEmbeddings for generating vector embeddings, FAISS for efficient similarity search in vector space, SemanticSimilarityExampleSelector for example-based retrieval, and finally, integrated tools like ollama and chromadb to streamline operations within this complex environment.

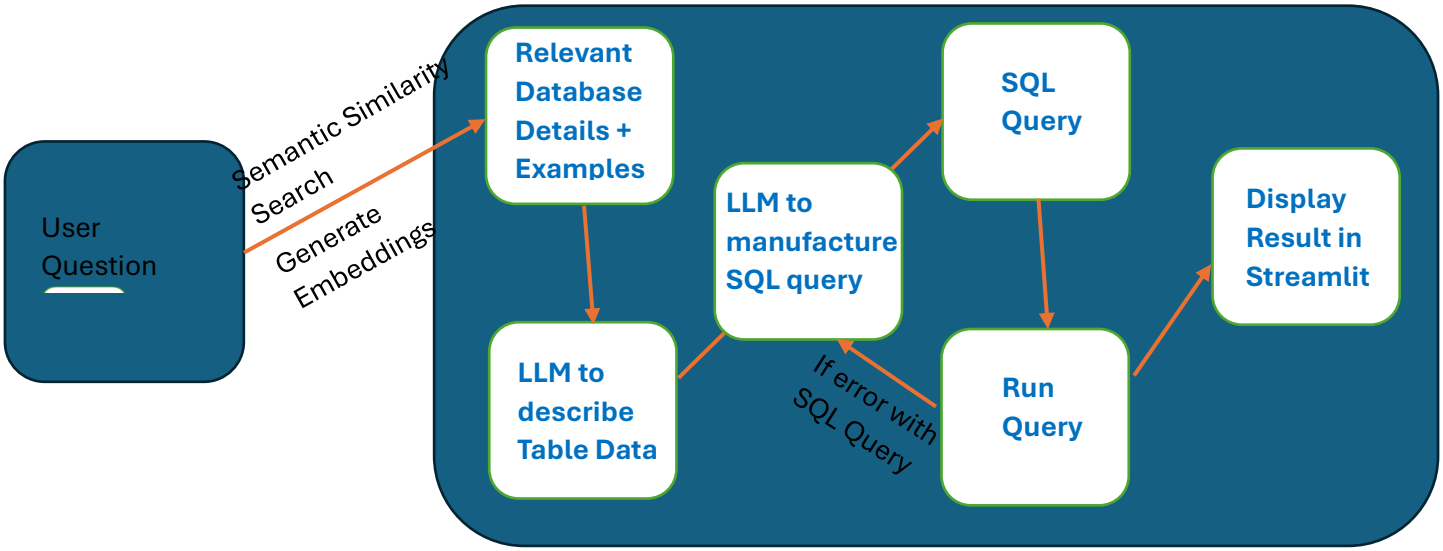
To accomplish this, I have developed 2 methods with similar functionality but an added aspect to one of them:

1. For the first code, the basic flow of my project looks something like this:



Pros of this method	Cons of this method
Quicker Runtime	Slightly Skewed Accuracy

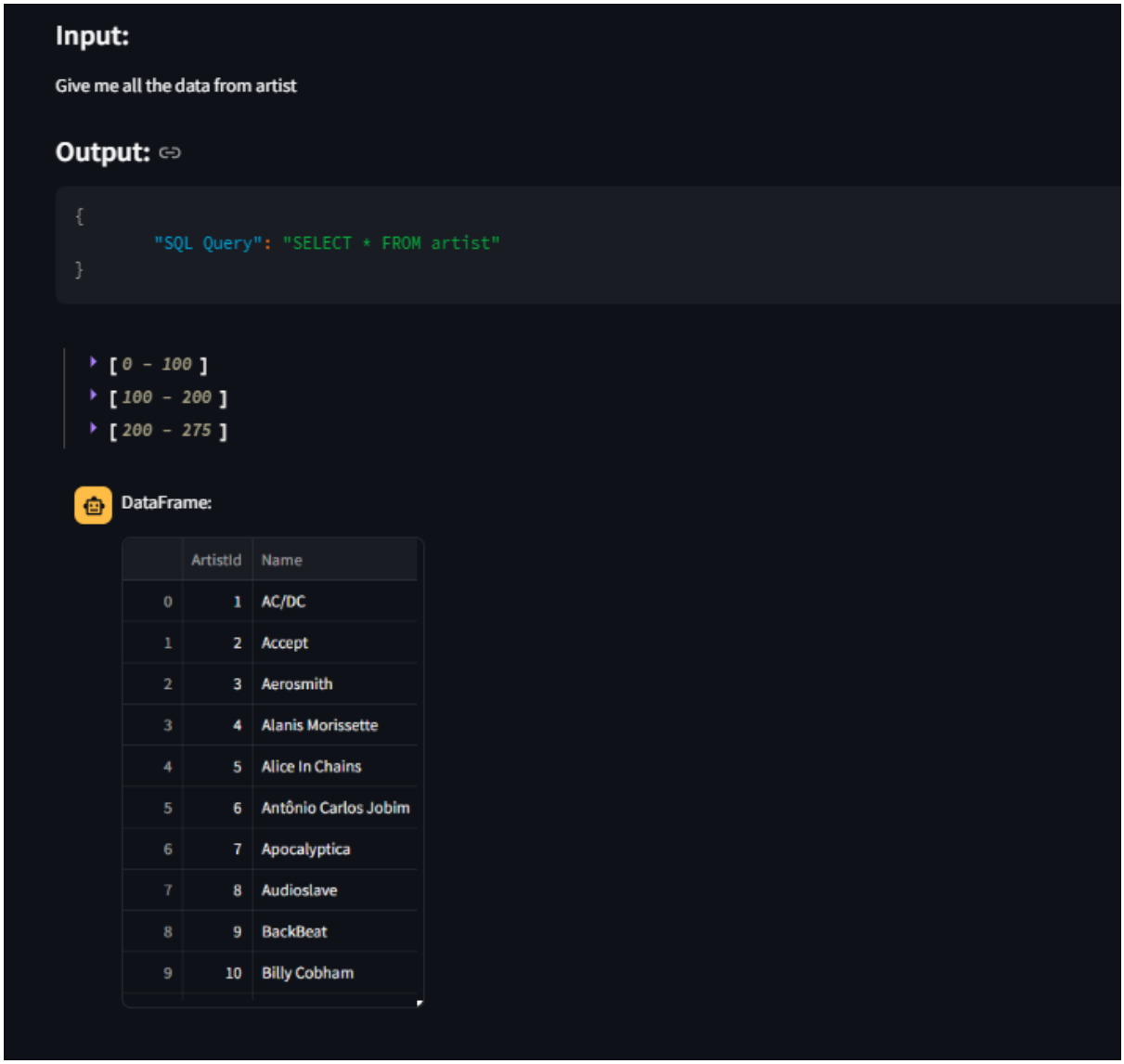
2. For the Second code, the Flow of the project is:



Pros of this method	Cons of this method
Higher Accuracy	Longer Runtime.

SQL Jio Internship Project

- Here is the Github Link where you can find the project code for a sample “Chinook” database.



Heres an example run of the code. As you can see I asked the model a question on streamlit to give all the data from the “artist” table and it generated the query in a csv downloadable dataframe.

With these 2 methods, the user can chose what they prioritize and then accordingly run the respective code.

To run these codes, all the user has to do is to enter their MySQL database name and URI and the code will automatically connect to the database, generate descriptions and examples for each table within the database,

engineer the prompt, pass it to the LLM, run the then generated SQL query and print the respective output.

Difficulty Faced	Solution
I started off knowing very little about Ollama, Huggingface, Embeddings, Prompt Engineering, connecting SQL and Streamlit with Python, Langchain.	I watched comprehensive youtube videos and learnt information from various websites to expand my knowledge.
I encountered an issue with accessing and using a valid API key for HuggingFace	I eradicated this issue by using Ollama.
I didn't know how to connect SQL with Python.	After some research I came upon the SQLDatabase Library in Langchain which helped me establish the connection
I did not know how to autogenerate examples and table descriptions for each database.	I researched on python logic and eventually set up a fully generalized code to generate complicated and simple examples for me.
I was getting a phantom error with the embeddings process while doing using the ollama.embeddings function from the ollama library.	I eliminated this error by using a different and more efficient Library called OllamaEmbeddings from Langchain.

I was not sure on how to make the table descriptions readable by the Language Learning Model instead of a cryptic python dictionary.	I used the “inflect” library in python as well as python string logic to facilitate the process of making the data readable
I didn’t know how to format the LLM response as I needed it to.	I used the “ResponseSchema” and “StructuredOutputParsers” Library from the Langchain package provided by python.

Future Steps: I am planning to apply the knowledge that I have gained of LLM’s here to create my own chatbots designed to answer questions only relevant to me.