QA System using Google Palm

Goal : To build a LLM Model with a software on top of SQL Databases, so that manager can retrieve data information from SQL Databases without the knowledge of SQL in human language

Tools Used

- UI : Streamlit
- LLM : Google Palm LLM Model
- Embeddings: Hugging Face
- Framework: Langchain

MySQL Database:

T-shirts table and Discount table

Technical Architecture :

Question by human -> LLM Google Palm from langchain framework -> SQL Query (Converted)

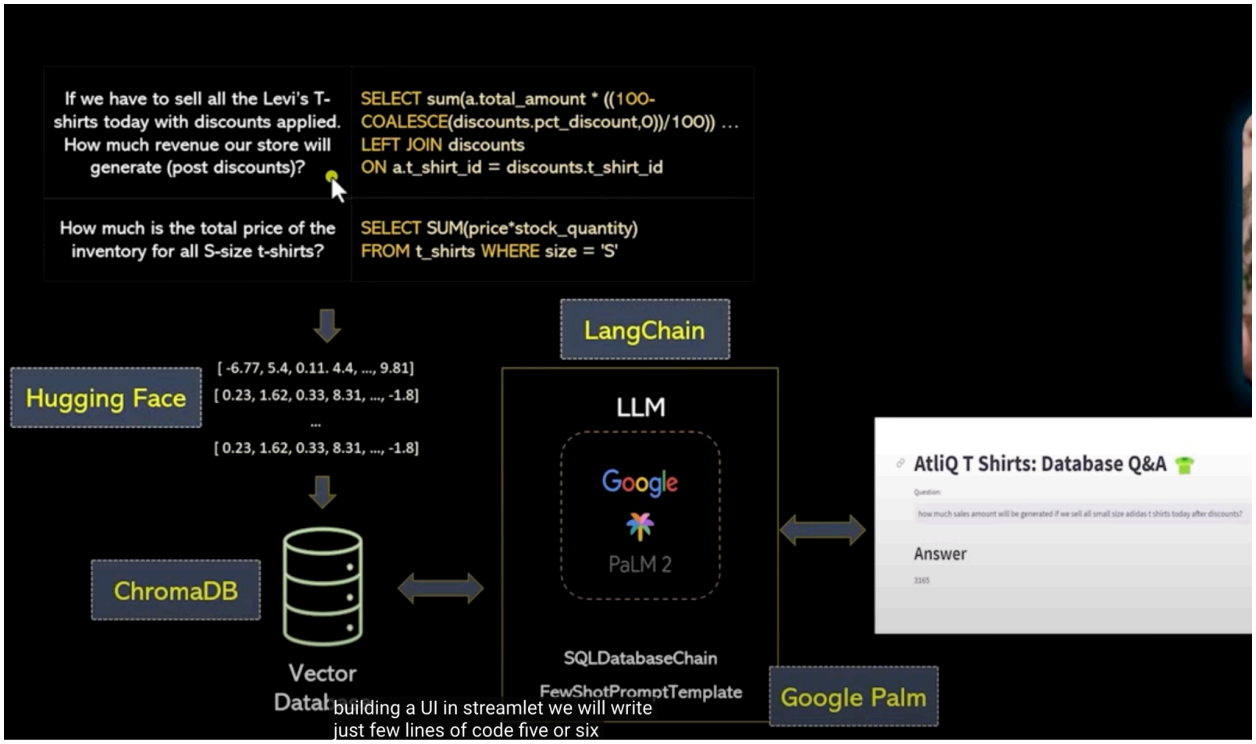
The above process works fine for some of the queries but fails in complex ones. So we need to do spacial handling using few short learning

1. Here we have to create a training dataset with sample questions and corresponding SQL query, it serves as an out of the box where google model fails. It can be prepared by Data Analyst

2. Convert this training dataset into embedding vector (use HuggingFace library for Embeddings)

3. Once embeddings are created store them in vector database - Chromadb

4. Pair the vector database with google palm LLM along with few short prompt template to create SQL Database chain atlast with a UI using streamlit



Step : 1

Login to Makersuite -> it has text prompts and others which uses architecture as Google Palm but the model used is **'Text Bison'**

Temperature - creativity parameter - closer to 1 - more creative

MySQL Workbench

Create database - with t-shirt and discount table

Step : 2

Google Colab - Initiating Google Palm

CODE:

Pip install -q -u google-generativeai
Import google.generativeai as genai

Genai.configure = (api_key = “”)
Model = genai.GenerativeModel(“gemini-1.5-flash”)
Response = model.generate_content(“write a world war script”)

Step : 3

From langchain.utilities import SQLDatabase

SQLDatabase.from_uri()

Connect SQL Database from colab to MySQL workbench

From langchain_experimental.sql import SQLDatabaseChain

Db_chain = SQLDatabaseChain(llm, db, verbose = True)

Step: 4

Few Short Learning

```
few_shots = [
   {'Question' : "How many t-shirts do we have left for Nike in XS size and white color?",
    'SQLQuery' : "SELECT sum(stock_quantity) FROM t_shirts WHERE brand = 'Nike' AND color = 'White' AND size =
'XS'",
    'SQLResult': "Result of the SQL query",
    'Answer' : qns1},
   {'Question': "How much is the total price of the inventory for all S-size t-shirts?",
    'SQLQuery':"SELECT SUM(price*stock_quantity) FROM t_shirts WHERE size = 'S'",
    'SQLResult': "Result of the SQL query",
    'Answer': qns2},
   {'Question': "If we have to sell all the Levi’s T-shirts today with discounts applied. How much revenue  will our store
generate (post discounts)?" ,
    'SQLQuery' : """SELECT sum(a.total_amount * ((100-COALESCE(discounts.pct_discount,0))/100)) as total_revenue
from
(select sum(price*stock_quantity) as total_amount, t_shirt_id from t_shirts where brand = 'Levi'
group by t_shirt_id) a left join discounts on a.t_shirt_id = discounts.t_shirt_id
 """,
    'SQLResult': "Result of the SQL query",
    'Answer': qns3} ,
    {'Question' : "If we have to sell all the Levi’s T-shirts today. How much revenue will our store generate without
discounts?" ,
     'SQLQuery': "SELECT SUM(price * stock_quantity) FROM t_shirts WHERE brand = 'Levi'",
     'SQLResult': "Result of the SQL query",
     'Answer' : qns4}
```

```
    {'Question': "How many white colored Levi's shirts do I have?",
     'SQLQuery' : "SELECT sum(stock_quantity) FROM t_shirts WHERE brand = 'Levi' AND color = 'White'",
     'SQLResult': "Result of the SQL query",
     'Answer' : qns5
     }
]
```

From langchain.embeddings import HuggingFaceEmbeddings

Embedding: It is a array of values generated from the queries provided

Store in vector DB - vectorstore

From langchain.prompts import SemanticsSimilarityMatcher

To match the vector values corresponding to semantic search using cosine similarity function in chromadb

```
Vector_store = (
        Vector_store
        K = 2
)
```

K indicates similar kind of matching values

Form a mySQL query prompt and use UI from langchain project documentation from github to make it as running application