{SAMPLE SET}

Gen AI Engineer / Machine Learning Engineer Assignment

Part 1: Retrieval-Augmented Generation (RAG) Model for QA Bot on P&L Data

Problem Statement:

Develop a Retrieval-Augmented Generation (RAG) model for a Question Answering (QA) bot that can process financial terms and insights from a Profit & Loss (P&L) table extracted from PDF documents. The QA bot should retrieve relevant information related to income, expenses, profit margins, and other key financial metrics from the provided P&L table and generate accurate and coherent responses.

Task Requirements:

- Implement a RAG-based model to handle questions related to a P&L table extracted from PDF documents.
- Use a vector database (such as Pinecone) to store and retrieve document embeddings of financial terms and data points efficiently.
- Parse P&L data from PDF documents into a structured format, such as tables or key-value pairs, before storing embeddings.
- Test the model with several financial queries and show how accurately it retrieves and generates responses from the dataset.

Deliverables:

1. **Collab Notebook:**
    - Demonstrates the entire pipeline, including:
        - Loading and preprocessing financial data from P&L tables in PDFs.
        - Embedding financial terms using vectorization techniques.
        - Implementing the RAG model to retrieve and generate responses.
    - Includes query examples such as:
        - "What is the gross profit for Q3 2024?"
        - "How do the net income and operating expenses compare for Q1 2024?"

2. **Documentation:**
    - Explains the model architecture, approach to data extraction and preprocessing, and how the generative responses are created.
    - Details challenges encountered and the solutions implemented.
3. **Example Queries and Outputs:**
    - Provide a range of financial queries alongside corresponding responses from the QA bot.

Part 2: Interactive QA Bot Interface for Financial Data

Problem Statement:

Develop an interactive interface for the QA bot from Part 1, allowing users to upload PDF documents containing P&L tables and ask questions about the financial data. The interface should facilitate real-time queries and display retrieved information alongside generated responses.

Task Requirements:

1. **Frontend Interface:**
    - Use **Streamlit** or **Gradio** to build a user-friendly interface.
    - Enable users to upload PDF documents containing financial data, such as P&L statements.
    - Allow users to input financial queries (e.g., "What is the total revenue for the year?") and retrieve contextually relevant answers.
2. **Backend Integration:**
    - Process uploaded PDFs to extract and preprocess P&L data, converting it into a structured format.
    - Store document embeddings in a vector database for efficient retrieval.
    - Use the backend RAG model to provide real-time, accurate answers to user queries.
    - Display the retrieved financial data segments (e.g., rows or cells from the P&L table) alongside the generated answer.
3. **Performance:**
    - Ensure the system handles large P&L documents and multiple queries efficiently without significant performance drops.

Deliverables:

1. **Deployed QA Bot Interface:**
    - A working, interactive frontend where users can:
        - Upload PDF documents with P&L tables.
        - Ask financial questions and view answers alongside the relevant table segments.

2. **Documentation:**
    - Guide users on how to upload documents, ask questions, and interpret the bot's responses.
    - Include examples of interactions demonstrating the bot's financial query capabilities.
3. **Example Interaction Scenarios:**
    - Example queries:
        - "What are the total expenses for Q2 2023?"
        - "Show the operating margin for the past 6 months."

General Guidelines

1. **Containerization:**
    - Use Docker to containerize the application for seamless deployment.
2. **Code and Documentation:**
    - Modular and scalable code with best practices for frontend and backend development.
    - Include a detailed README file in the GitHub repository with setup and usage instructions.
3. **Performance Considerations:**
    - Optimize the pipeline to handle large financial documents and multiple queries without latency issues.
    - Ensure accurate parsing and retrieval of P&L terms for precise question answering.

Submission Checklist:

- Source code for the Colab notebook and interactive interface.
- Fully functional Colab notebook demonstrating the QA pipeline.
- Complete documentation explaining the approach, deployment, and usage instructions.
- Example queries and outputs showcasing the system's accuracy with financial data.