**GOVERNMENT COLLEGE OF TECHNOLOGY**

**Name : RAJESWARI P , HEMA Y**

**Reg no : 71772117134,71772117114**

**Branch : Computer Science and Engineering**

**Course : Artificial Intelligence**

**Course code : 18SPC702**

**Assignment : 2**

**Project : Excel Question-Answer System**

**Project Description:**

The **Excel Question-Answer System** is an AI-driven application that enables efficient question answering based on Excel documents. The application is designed to streamline data analysis by allowing users to ask natural language questions about data contained in Excel files, making it ideal for historical, financial, or time-series datasets. By combining data extraction with an advanced AI model, this tool allows users to receive direct, context-aware answers without the need to manually search through data or create complex queries.

**Project Approach:**

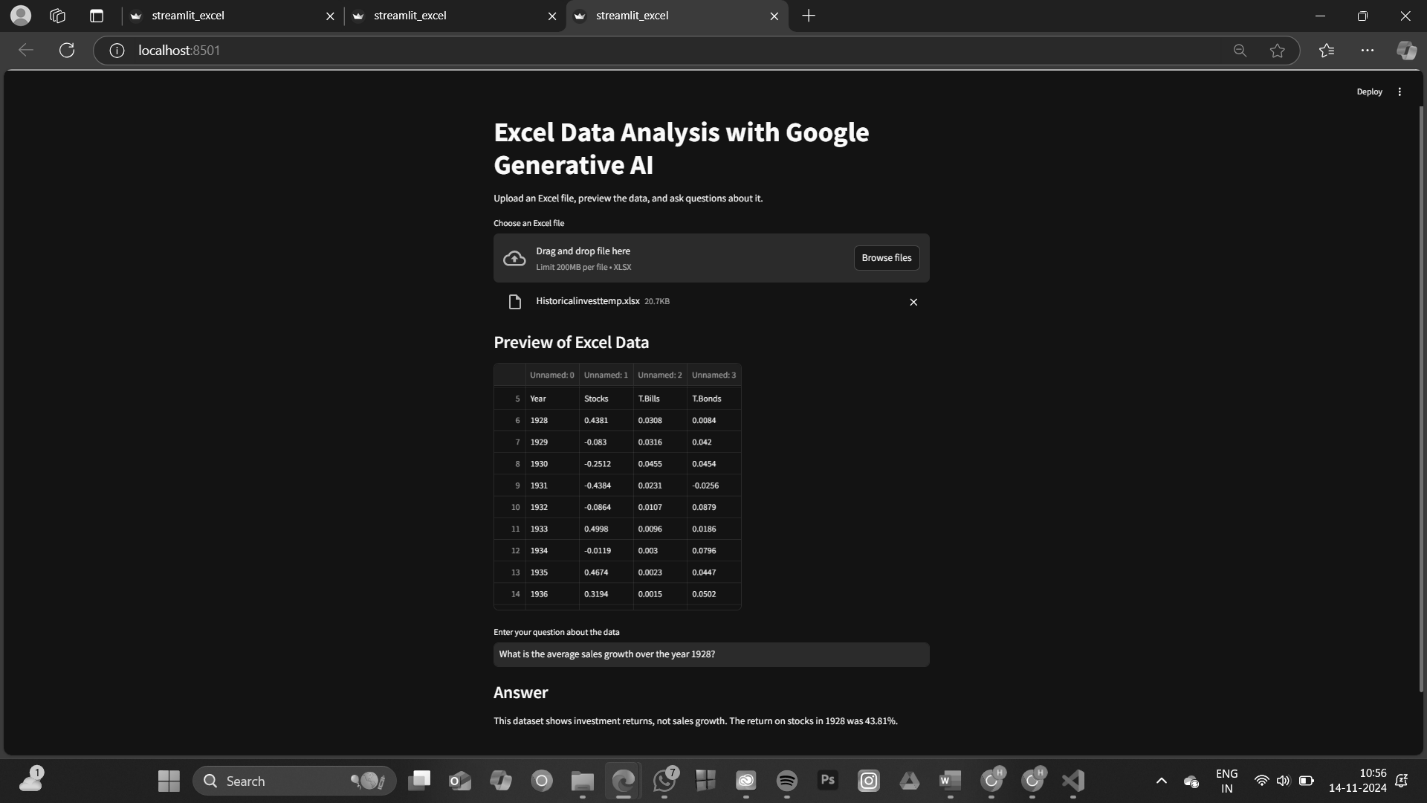
1. **Data Ingestion and Preparation:**
   * **File Upload:** Using Streamlit’s file\_uploader component, users can upload a excel file. Once uploaded, the file is stored locally and read using pandas.
   * **Data Conversion:** The Excel file is loaded into a pandas DataFrame for efficient handling and manipulation of tabular data. After loading, the DataFrame is converted to a string format using to\_string() to provide the Generative AI model with a readable context of the data. This context includes all rows and columns, giving the model a full view of the dataset.
2. **User Interaction and Query Input:**
   * **Question Input:** Streamlit’s text\_input allows users to enter questions in natural language, making it easy to ask various data-related questions, from specific queries (e.g., “What was the stock price in 1928?”) to general ones (e.g., “What is the average sales growth over the years?”).
   * **Prompt Formulation:** The question is combined with the data context into a single prompt, ensuring that the Generative AI model has both the relevant data and the question clearly presented to produce accurate, context-aware answers.
3. **Generative AI Model Integration:**
   * **Model Initialization:** The application establishes a session with Google’s gemini-1.5-pro model via the GenerativeModel API, creating a new chat instance. This allows the model to “converse” with the data context and interpret queries in the style of a human assistant.
   * **Prompt Submission and Response Handling:** The complete prompt (data context and user question) is submitted to the model using the send\_message function. The model processes this input and provides a response designed to directly answer the question based on the dataset context.
   * **Error Handling and Retry Logic:** In cases where the model request fails (e.g., network or API issues), an error message is displayed in Streamlit. A retry mechanism with a delay is included to reattempt the request if possible, ensuring a robust user experience.
4. **Response Extraction and Display:**
   * **Parsing AI Response:** Once the model returns a response, the answer text is extracted from the response’s candidates[0].content.parts[0].text. This content is a natural language response generated by the model, formatted as a human-readable answer to the question.
   * **Displaying the Answer:** The parsed answer is displayed on the Streamlit interface under an “Answer” section. This intuitive display format helps users get instant feedback on their queries, supporting an iterative and interactive question-answer experience.
   * **Session Management:** Users can ask multiple questions in a single session, viewing answers in real-time. A “Quit” button provides a clear exit from the interface.

**Model Details:**

The primary model used in this project is Google’s Generative AI gemini-1.5-pro model, known for its capabilities in natural language understanding and generation, especially in structured data querying contexts.

**Model Characteristics and Suitability:**

1. **Language Understanding and Contextual Awareness:**
   * The gemini-1.5-pro model is trained on extensive datasets to understand and respond to natural language queries. It excels at interpreting nuanced questions and providing answers grounded in the data context, even when users ask questions requiring summarizing trends or comparing values.
   * By analyzing structured tabular data in text format, the model can derive insights directly from the data without needing a predefined schema or structure.
2. **Data-Driven Response Generation:**
   * The model can deliver specific, context-aware answers by leveraging advanced language modeling and reinforcement learning. For instance, with chronological records (like stock prices by year), the model can reference specific years and observe trends.
   * The gemini-1.5-pro is particularly suitable for financial and historical datasets, as it can identify relationships and summarize quantitative data in natural language.
3. **Interactive Dialogue Capability:**
   * The chat-based architecture of gemini-1.5-pro allows it to handle sequential questions within the same session, which is useful when users need follow-up information or clarifications based on previous responses.
4. **Streamlit Integration for Enhanced Usability:**
   * Streamlit provides an intuitive web interface, making the model’s power accessible with real-time feedback and an easy-to-use platform for query and answer interactions.
5. **Output:**
6. The output of the application is a natural language answer to the user's query displayed in the Streamlit interface, under an "Answer" section for easy readability and interactive engagement.



**Challenges Faced:**

1. **Handling Large Data Contexts:**
   * **Challenge:** Generative AI models have input size limits, making it difficult to send large datasets as prompts.
   * **Solution:** The data context was formatted to provide only relevant portions or summaries of the dataset. In the future, selecting specific rows or columns based on the user’s question could further optimize data input.
2. **Ensuring Accurate Responses to Varied Queries:**
   * **Challenge:** Natural language questions vary in phrasing, specificity, and intent, posing a risk of inaccurate answers without a well-defined context.
   * **Solution:** Combining the dataset context and question in a structured prompt format helped the model focus on specific data aspects, improving response relevance and accuracy.
3. **Error Handling and API Reliability:**
   * **Challenge:** Network errors, API rate limits, or timeout issues could disrupt model responses.
   * **Solution:** A retry mechanism with error handling was incorporated to improve resilience and provide reliability even under unstable conditions.
4. **Adapting to Natural Language Complexity:**
   * **Challenge:** The model can sometimes misinterpret complex questions, particularly when multi-step reasoning is required.
   * **Solution:** Testing different prompt formulations and encouraging users to ask clear, direct questions optimized interpretive accuracy.
5. **Integration of Streamlit and API-Based Responses:**
   * **Challenge:** Coordinating data flow between Streamlit, Google’s API, and pandas required careful handling for real-time processing.
   * **Solution:** Streamlit components were designed for interactive querying, real-time response display, and user feedback, enhancing the overall user experience.