I built this project as a personal portfolio application to learn how AI can be used for real data analysis tasks. The idea of the project is simple: a user uploads a CSV file and then asks questions about the data in plain English. The application uses an AI model to understand the question and automatically generate Python or SQL code to analyze the data. That code is executed inside the app, and the results like tables, numbers, or charts are shown directly in a chat interface.

When the app starts, it initializes the OpenAI client and Streamlit session state. I use session state to store chat messages, the uploaded dataset, data summary, column profiling information, and the selected analysis mode. This is important because Streamlit reruns the app frequently, and session state helps maintain continuity.

After the user uploads a CSV file, I load it using pandas and store it as a DataFrame. I also create a basic data summary with column names, data types, sample rows, and statistics. In addition, the app automatically profiles each column to detect data types, missing values, possible ID columns, and potential date columns. This profiling is shown in the sidebar as a Data Health Report so users can understand their data quality before asking questions.

The user can choose between two analysis engines: Python mode or SQL mode. Python mode is useful for flexible analysis and visualizations, while SQL mode uses DuckDB to run structured queries. This makes the app more versatile and shows different ways to analyze data.

When a user asks a question, the app builds a system prompt that includes information about the dataset, such as schema and sample values. The prompt also contains strict instructions so the AI generates executable code. Only the recent chat messages are sent to the model to control token usage.

If the AI returns Python code, the app extracts the code and executes it. It captures printed output, returned results, warnings, and any charts created using matplotlib, and displays everything directly below the AI response. If the user is in SQL mode, the generated query is validated to allow only safe SELECT statements and then executed using DuckDB.

All questions and responses are stored in session state so the full conversation is preserved. The app also allows exporting the entire analysis as an HTML report, which includes the dataset information, questions, and analysis results.

I built this project to learn prompt engineering, AI integration, and end-to-end application design. This is a learning and portfolio project, not a production system, but it clearly demonstrates how AI, data analysis, and software engineering can work together in a real application.