# Project Report: Conversational Data Analysis Agent (Streamlit & NVIDIA NIM)

## 1. Executive Summary

This project implements a sophisticated, conversational Exploratory Data Analysis (EDA) tool leveraging the power of Large Language Models (LLMs) to interact with user-uploaded data. Built on **Streamlit** for the frontend and powered by the **NVIDIA NIM Llama 3 8B Instruct** model via LangChain, the application enables non-technical users to perform deep data analysis, generate summaries, and produce high-quality visualizations (Matplotlib, Seaborn, Plotly) simply by asking questions in natural language.

## 2. Technical Architecture

The architecture is designed for modularity, safety, and efficiency:

| **Component** | **Technology** | **Role** |
| --- | --- | --- |
| **Frontend** | Streamlit | Provides the web interface, file uploading, chat history, and visualization rendering. |
| **LLM Backend** | NVIDIA NIM (Llama 3 8B) | Generates the analytical text and, critically, the executable Python plotting code based on the user's data context. |
| **LLM Interface** | langchain-nvidia-ai-endpoints | Manages the connection, communication, and prompt formatting for the LLM. |
| **Data Handling** | Pandas & openpyxl | Reads multi-format data (CSV, Excel, JSON) and manages the DataFrame in Streamlit's session state. |
| **Code Execution** | Python's exec() (Wrapped) | Safely extracts LLM-generated Python code, injects the DataFrame (data), and executes it to produce visuals. Robust error handling prevents common execution failures. |
| **Configuration** | python-dotenv | Securely loads the NVIDIA\_API\_KEY from the local .env file. |

## 3. Key LLM Prompt Engineering & Safety Measures

The core intelligence of the agent relies on a highly structured system prompt sent to the LLM:

1. **Context Injection:** The prompt includes the DataFrame's head() and dtypes to ground the LLM's response in the specific data.
2. **Output Enforcement:** The LLM is strictly instructed to use a **markdown table** for statistical outputs (e.g., descriptive statistics) and a **python code block** for visualizations.
3. **Code Safety & Robustness:** The prompt contains an **enforcement rule** that dictates safe coding practices, such as explicitly using the data.select\_dtypes(include=['number']).columns.tolist() method for dynamic column selection. This rule was critical for solving the "truth value of an array is ambiguous" error, ensuring the LLM's generated code is highly functional with Pandas.

## 4. Analytical Capabilities

* **Structural Summary:** Provides a rapid overview of the dataset shape, data types, and non-null values.
* **Deep Analytical Insights:** Generates initial hypotheses, identifies potential feature relationships, and suggests areas for further investigation (e.g., "The strong negative correlation between X and Y suggests Z").
* **Dynamic Visualization:** Can generate complex plots (scatterplots, histograms, pairplots, box plots, etc.) on demand using the most appropriate visualization library based on the request.