

Data Analyzer

An Interactive Web Application for Data Visualization, Advanced Querying, and AI-Powered Insights



May 2, 2024

BY ADITYA TRIPATHI

MAIB 61639

Table of Contents

[Introduction 2](#_Toc165372553)

[Technologies Used 2](#_Toc165372554)

[Features 2](#_Toc165372555)

[User Interface 2](#_Toc165372556)

[Implementation Details 3](#_Toc165372557)

[Main Function 3](#_Toc165372558)

[Data Visualization 4](#_Toc165372559)

[Advanced Query 5](#_Toc165372560)

[Claude AI Insights 5](#_Toc165372561)

[Helper Functions 6](#_Toc165372562)

[Error Handling and Robustness 7](#_Toc165372563)

[Usage 7](#_Toc165372564)

[Demo 7](#_Toc165372565)

[Conclusion 7](#_Toc165372566)

# Introduction

The CSV Data Analyzer is a web application developed as a major project for university coursework. This project aims to provide a comprehensive solution for analyzing and visualizing CSV (Comma-Separated Values) data, offering functionalities such as data visualization, advanced querying, and AI-powered data insights. The application is built using Python and leverages various libraries and APIs to enhance data analysis capabilities.

## Technologies Used

1. **Streamlit**: Used for building the user interface and creating interactive elements.
2. **Pandas**: Utilized for data manipulation, loading CSV files, and executing custom queries.
3. **Plotly** **Express**: Integrated for generating interactive charts and visualizations.
4. **Base64**: Employed for encoding and decoding binary data, particularly for file downloads.
5. **Anthropic** **API**: Integrated to provide AI-powered insights and data analysis capabilities.

# Features

The application offers the following features:

1. **Upload CSV File**: Users can upload a CSV file containing the data they wish to analyse.
2. **Data Visualization**: Users can select from various chart types (Scatter Plot, Bar Chart, Line Chart, Histogram, and Box Plot) and choose the columns to represent the x-axis, y-axis, and color. The selected chart is then displayed using Plotly Express.
3. **Advanced Query**: Users can write custom queries using Pandas syntax and execute them on the uploaded data. The query results are displayed in a tabular format, and users can download the results as a CSV file.
4. **Claude AI-powered Data Insights**: Users can ask questions about their data in natural language, and the application will leverage the Claude LLM to provide insightful and context-relevant responses based on the data and the user's question.

# User Interface

* The user interface of the CSV Data Analyzer is designed to be intuitive and user-friendly. It features:
* A visually appealing animated title using HTML and CSS styles.
* An upload section where users can upload CSV files for analysis.
* Three main tabs for Data Visualization, Advanced Querying, and Claude AI-powered Insights.
* Interactive elements such as dropdowns, text areas, buttons, and expandable sections for a seamless user experience.

# Implementation Details

## Main Function

The main() function is the entry point of the application. It sets up the Streamlit page configuration, displays the title, and handles the file upload process. Based on the user's selection from the tab menu, it calls the respective function for data visualization, advanced query, or Claude AI insights.

**def** **main**():  
 st.set\_page\_config(layout="wide", initial\_sidebar\_state="expanded", page\_icon="📊", page\_title="CSV Data Analyzer")  
 st.markdown(  
 """  
 <h1 class="animated-title" style='color: #f2d75e; font-family: "Playfair Display"; display: flex; align-items: center;'>  
 📊 <span>CSV Data Analyzer</span> 💡  
 </h1>  
 """,  
 unsafe\_allow\_html=**True**  
 )  
  
 # Initialize session state variables  
 **if** "query\_result" **not** **in** st.session\_state:  
 st.session\_state.query\_result = **None**  
 **with** st.expander("Upload a csv file"):  
 # Upload CSV file  
 uploaded\_file = st.file\_uploader("UPLOAD CSV", type=['csv'])  
  
 **if** uploaded\_file **is** **not** **None**:  
 data = pd.read\_csv(uploaded\_file)  
  
 # Main tab  
 selected\_tab = st.selectbox("Select a tab:", ["Data Visualization", "Advanced Query", "Claude Insights"], index=0)  
  
 **if** selected\_tab == "Data Visualization":  
 data\_visualization(data)  
 **elif** selected\_tab == "Advanced Query":  
 advanced\_query(data)  
 **elif** selected\_tab == "Claude Insights":  
 claude\_insights(data)

## Data Visualization

The data\_visualization(data) function allows users to select a chart type and specify the columns to represent the x-axis, y-axis, and color. It then creates the selected chart using Plotly Express and displays it within the Streamlit app.

**def** **data\_visualization**(data):  
 st.header("Data Visualization")  
 **with** st.expander("Select Chart type and Axis to represent"):  
 # Select Chart Type  
 chart\_type = st.selectbox("Select Chart Type", ["Scatter Plot", "Bar Chart", "Line Chart", "Histogram", "Box Plot"])  
  
 col1, col2, col3 = st.columns([1, 1, 1])  
  
 # Select X Axis and Y Axis  
 **with** col1:  
 x\_axis = st.selectbox("Select X Axis", data.columns)  
 **with** col2:  
 y\_axis = st.selectbox("Select Y Axis", data.columns)  
 **with** col3:  
 # Select Color By  
 color\_by = st.selectbox("Color By", data.columns)  
  
 **with** st.expander("Click here to see the visuals"):  
 # Update the chart in the session state  
 **if** chart\_type == "Scatter Plot":  
 st.plotly\_chart(px.scatter(data, x=x\_axis, y=y\_axis, color=color\_by), use\_container\_width=**True**)  
 **elif** chart\_type == "Bar Chart":  
 st.plotly\_chart(px.bar(data, x=x\_axis, y=y\_axis, color=color\_by), use\_container\_width=**True**)  
 **elif** chart\_type == "Line Chart":  
 st.plotly\_chart(px.line(data, x=x\_axis, y=y\_axis, color=color\_by), use\_container\_width=**True**)  
 **elif** chart\_type == "Histogram":  
 st.plotly\_chart(px.histogram(data, x=x\_axis), use\_container\_width=**True**)  
 **else**: # Box Plot  
 st.plotly\_chart(px.box(data, x=x\_axis, y=y\_axis, color=color\_by), use\_container\_width=**True**)

## Advanced Query

The advanced\_query(data) function provides an interface for users to write custom queries using Pandas syntax. When the user runs a query, the execute\_query(data, query) function is called, which evaluates the query and stores the result in the session state. The query result is displayed in a tabular format, and users can download the result as a CSV file.

**def** **advanced\_query**(data):  
 st.header("Advanced Query")  
 **with** st.expander("Custom Query"):  
 query = st.text\_area("Write your custom query using Pandas syntax:", "data.head(3)", height=150)  
 **if** st.button("Run Query"):  
 execute\_query(data, query)  
  
 # Query Result  
 **if** st.session\_state.query\_result **is** **not** **None**:  
 st.subheader("Query Result")  
 result\_df = st.session\_state.query\_result  
 **with** st.container():  
 st.table(result\_df)  
  
 # Download button for CSV file  
 csv = st.session\_state.query\_result.to\_csv(index=**False**)  
 b64 = base64.b64encode(csv.encode()).decode()  
 href = f'<a href="data:file/csv;base64,{b64}" download="query\_result.csv">Download CSV File</a>'  
 st.markdown(href, unsafe\_allow\_html=**True**)

## Claude AI Insights

The claude\_insights(data) function allows users to ask questions about their data. When the user submits a question, the get\_data\_insights(data, question) function is called, which sends a request to the Anthropic API with the provided data and question. The response from the API, containing the AI-powered insights, is then displayed within the Streamlit app.

**def** **claude\_insights**(data):  
 st.header("Claude AI-powered Data Insights")  
 **with** st.expander("Get Insights"):  
 question = st.text\_area("Ask questions about your data, and let Claude AI provide insights.")  
 **if** st.button("Get Insights"):  
 **with** st.spinner("Fetching insights..."):  
 insights = get\_data\_insights(data, question)  
 **with** st.container():  
 st.write(insights)

## Helper Functions

The application includes the following helper functions:

* execute\_query(data, query): Evaluates the provided query on the data and stores the result in the session state.

# Helper function to execute custom query  
**def** **execute\_query**(data, query):  
 **try**:  
 result\_df = eval(query)  
 st.session\_state.query\_result = result\_df  
 **except** Exception **as** e:  
 st.error(f"Error: {e}")

* get\_data\_insights(data, question): Sends a request to the Anthropic API with the provided data and question, and returns the AI-powered insights from the response.

# Helper function to get data insights from Anthropic API  
**def** **get\_data\_insights**(data, question):  
 **if** data.empty:  
 **return** "No data provided for analysis."  
  
 full\_question = f"Given the following tabular data:\n{data.to\_string(index=**False**)}\n\nQuestion: {question}\n\nInsights:"  
 response = client.messages.create(  
 model="claude-3-opus-20240229",  
 max\_tokens=512,  
 messages=[  
 {"role": "user", "content": full\_question}  
 ]  
 )  
  
 **if** response.content **and** isinstance(response.content, list) **and** len(response.content) > 0:  
 text\_block = response.content[0]  
 **if** hasattr(text\_block, 'text'):  
 content = text\_block.text  
 **return** content  
 **else**:  
 **return** "Error: Unexpected response format."  
 **else**:  
 **return** "Error: No content received from the API."

# Error Handling and Robustness

The application incorporates robust error-handling mechanisms to ensure smooth functioning and reliability. It checks for missing API keys, handles exceptions during query execution, and provides informative error messages to users when issues arise. This ensures a seamless user experience and prevents unexpected failures.

# Usage

* To run the application, you must install Python and the required libraries.
* Install the required libraries by running pip install streamlit pandas plotly base64 anthropic.
* set the ANTHROPIC\_API\_KEY environment variable on your local machine.
* Run the application with streamlit run csv\_data\_analyser.py.
* Upload a CSV file and explore the various features of the data analyzer.

# Demo

As part of this presentation, I would like to provide a live demonstration of the CSV Data Analyzer web application. Below is a video where I walk through the key features and functionalities of the app, showcasing how users can upload CSV files, visualize data, perform custom queries, and obtain AI-powered insights.

Please click on the link below as it will redirect you to the video uploaded on my GitHub repository, make sure you have access to the internet.

[Website Demo](https://github.com/Adineu03/MajorProject/blob/main/README.md" \l "demo)

# Conclusion

The CSV Data Analyzer is a dynamic solution that combines cutting-edge technologies and AI-driven insights to facilitate robust data analysis and visualization. Its intuitive interface, coupled with interactive features, caters to a wide range of users, from researchers and analysts to data enthusiasts across various industries. By streamlining data-driven decision-making processes and fostering a culture of innovation, the CSV Data Analyzer empowers users to extract actionable intelligence and drive strategic growth initiatives. Its versatility and adaptability make it a valuable asset in navigating the complexities of data analysis, ensuring that organizations and individuals can harness the full potential of their data resources. As we move towards a data-centric future, the CSV Data Analyzer plays a pivotal role in shaping how data is leveraged to derive meaningful insights, make informed decisions, and achieve transformative outcomes that drive progress and innovation.