DAB RADIO DATA PROCESSING & VISUALIZATION APPLICATION

Prototype Application for
Data Cleaning, Transformation, Analysis & Visualisation

Presented by: Nicolette Woolery

Client Brief

Design and develop a prototype application that formats, cleans, and uses data to generate specific outputs.

- Develop a prototype application using Python (3.7–3.10)
- Process DAB radio station data from 3 CSV files
- Runs in Jupyter Notebook within Anaconda

- Focus: Data cleaning, transformation, GUI
- Generate outputs and visualisations

Client Specifications and Requirements

Functional Requirements

The application should provide the following functionality:
A means to load the initial data set (which consists of three CSV files) and translate it into a suitable format, either XML, or JSON or an entity relationship structure (not CSV).

A means to back up the suitable format using either files or a database. This should preserve the current state of the data when the program is closed, and make it available when the program is reopened.

A process for cleaning and preparing the initial data set, managing inconsistencies, errors, missing values and any specific changes required by the client (see below).

A graphical user interface(s) for interacting with the data that enables the user to:

- Load and clean an initial data set (from the CV format)
- Load and save a prepared data set (from its translated format)
- Use the prepared data set to generate output and visualisations
- Manipulate the range of values used to generate output and visualisations
- This program should be able to handle other sets of data generated from the same source, i.e. data with the same column row headings but containing different values and anomalies. However, the application is not required to be generic (work with multiple unknown data sets). Given this best practice regarding code reuse, encapsulation and a well-defined programming interface should be applied where applicable.

Data Cleaning and Preparation Process

- Remove DAB stations with specific ‘NGR’ values
- Extract DAB multiplex block from 'EID': C18A, C18F, C188
- Join multiplex to station metadata (Site, Site Height, Aerial height, Power)
- Rename fields: Ae Ht → Aerial height (m), ERP Total → Power (kW)

Data Analysis & Outputs

- Compute mean, mode, median for Power (kW)
- Site Height > 75
- Date from 2001 onwards
- Present results in table format

Data Visualisation

- Visualise data from multiplexes: C18A, C18F, C188
- Display: Site, Freq, Block, Service Labels 1–4 & 10
- Use bar charts or grouped visuals
- Identify correlation between Freq, Block, and Serv Labels
- Use visualisation like heat maps or correlation matrix

Technical Architecture

- Python 3.7–3.10 in Anaconda
- Jupyter Notebook interface
- Libraries: pandas, numpy, seaborn, matplotlib
- Data stored as JSON/XML or in DB
- No multithreading used

GUI Functionality Highlights

- Load raw CSV data
- Clean and convert data
- View/export processed dataset
- Filter and visualise data with feedback
- Error handling built-in

Reusability & Extensibility

- Works with similar-structured datasets
- Modular, encapsulated functions
- Reusable design, open to enhancements

Project Solution

Design Decisions

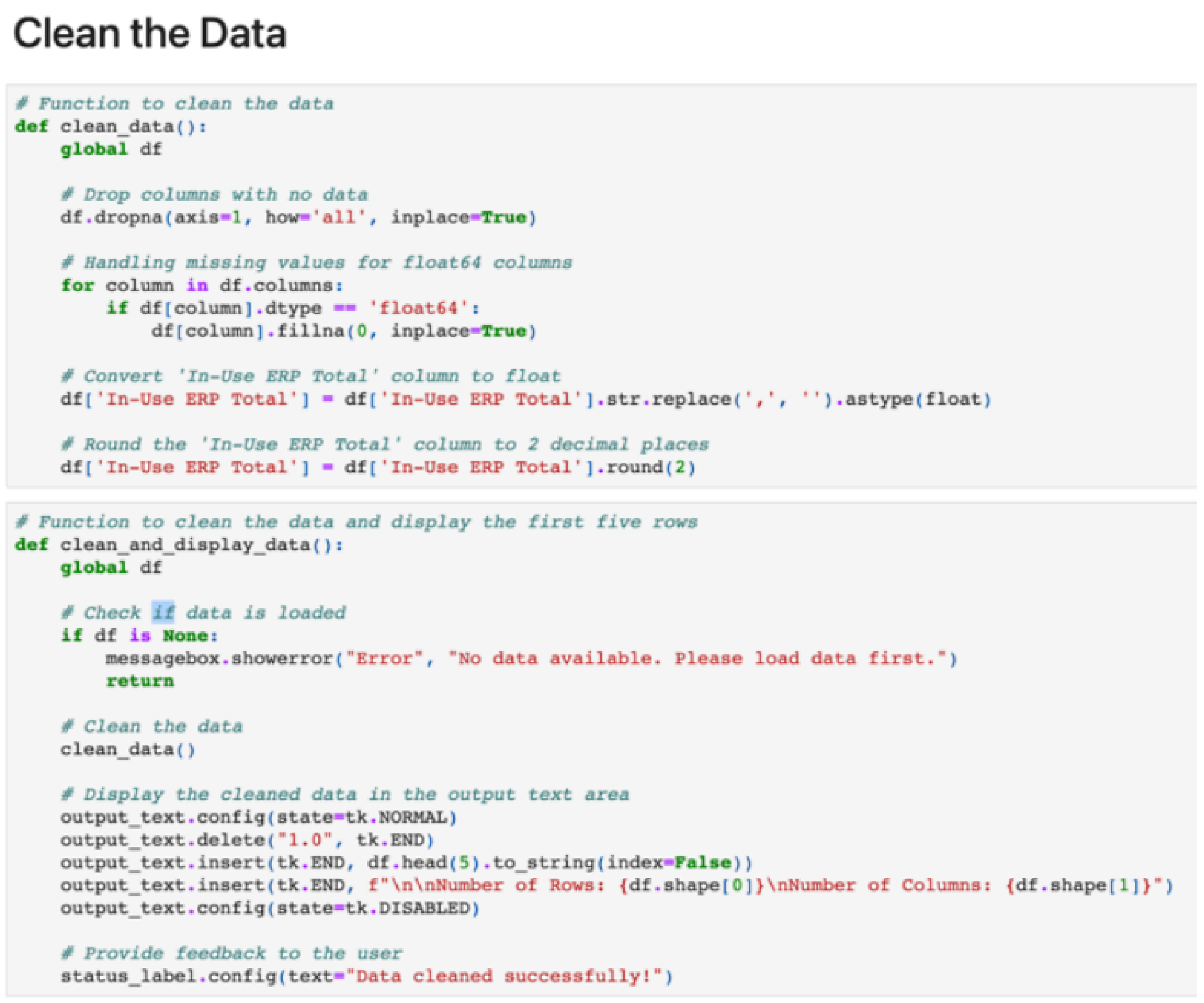
• JSON was selected as the primary format for storing cleaned data as it is lightweight, flexible, and easy to parse.
• Tkinter GUI with tabbed navigation was chosen as it improves usability.
• Pandas and Seaborn were chosen for robust data handling and insightful visualisations.

Technical Strengths

• Clean modular code with reusable functions
• Strong visualisation tools and statistical outputs
• Intuitive user interface supporting clear feedback
• Fulfills both functional and non-functional requirements

Advanced Programming Techniques & Libraries

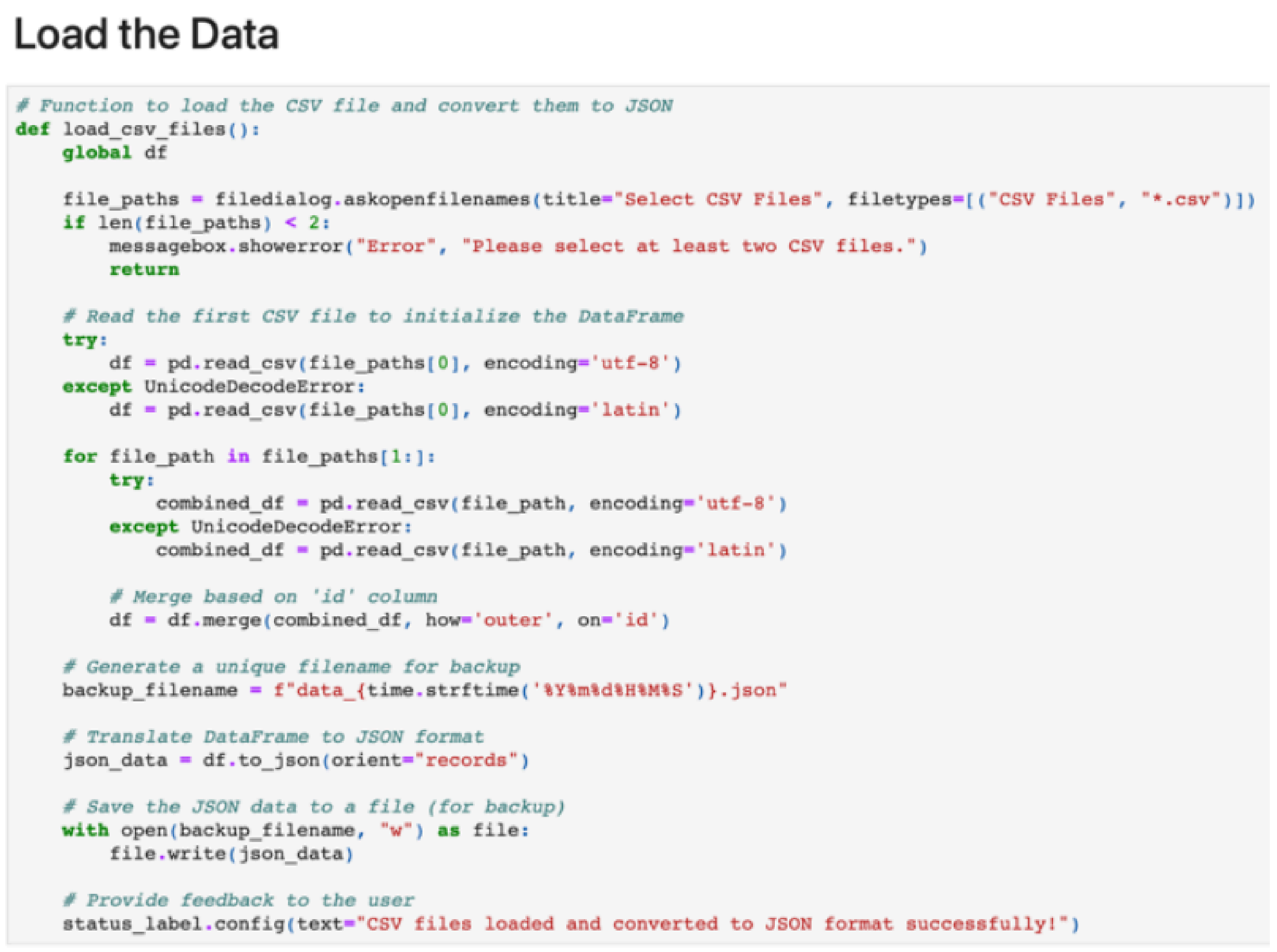
- Explored Python threading for parallelism
- Proposed threading for data loading and cleaning to improve performance
- Not implemented due to client constraints (no threading).



Clean the Data

```python
# Function to clean the data
def clean_data():
    global df

    # Drop columns with no data
    df.dropna(axis=1, how='all', inplace=True)

    # Handling missing values for float64 columns
    for column in df.columns:
        if df[column].dtype == 'float64':
            df[column].fillna(0, inplace=True)

    # Convert 'In-Use ERP Total' column to float
    df['In-Use ERP Total'] = df['In-Use ERP Total'].str.replace(',', '').astype(float)

    # Round the 'In-Use ERP Total' column to 2 decimal places
    df['In-Use ERP Total'] = df['In-Use ERP Total'].round(2)
```

```python
# Function to clean the data and display the first five rows
def clean_and_display_data():
    global df

    # Check if data is loaded
    if df is None:
        messagebox.showerror("Error", "No data available. Please load data first.")
        return

    # Clean the data
    clean_data()

    # Display the cleaned data in the output text area
    output_text.config(state=tk.NORMAL)
    output_text.delete("1.0", tk.END)
    output_text.insert(tk.END, df.head(5).to_string(index=False))
    output_text.insert(tk.END, f"\n\nNumber of Rows: {df.shape[0]}\nNumber of Columns: {df.shape[1]}")
    output_text.config(state=tk.DISABLED)

    # Provide feedback to the user
    status_label.config(text="Data cleaned successfully!")
```



Load the Data

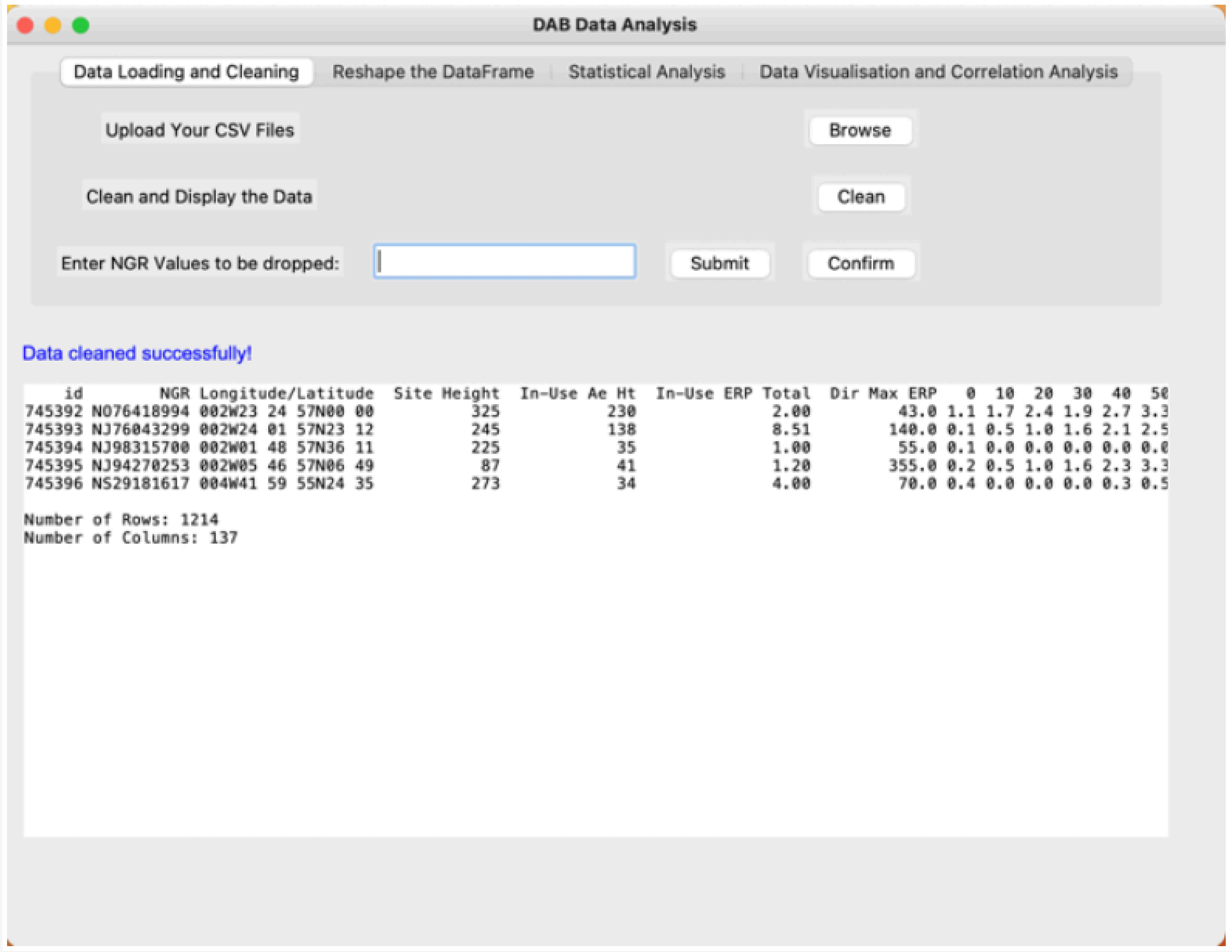
```python
# Function to load the CSV file and convert them to JSON
def load_csv_files():
    global df

    file_paths = filedialog.askopenfilenames(title="Select CSV Files", filetypes=[("CSV Files", "*.csv")])
    if len(file_paths) < 2:
        messagebox.showerror("Error", "Please select at least two CSV files.")
        return

    # Read the first CSV file to initialize the DataFrame
    try:
        df = pd.read_csv(file_paths[0], encoding='utf-8')
    except UnicodeDecodeError:
        df = pd.read_csv(file_paths[0], encoding='latin')

    for file_path in file_paths[1:]:
        try:
            combined_df = pd.read_csv(file_path, encoding='utf-8')
        except UnicodeDecodeError:
            combined_df = pd.read_csv(file_path, encoding='latin')

        # Merge based on 'id' column
        df = df.merge(combined_df, how='outer', on='id')

    # Generate a unique filename for backup
    backup_filename = f"data_{time.strftime('%Y%m%d%H%M%S')}.json"

    # Translate DataFrame to JSON format
    json_data = df.to_json(orient="records")

    # Save the JSON data to a file (for backup)
    with open(backup_filename, "w") as file:
        file.write(json_data)

    # Provide feedback to the user
    status_label.config(text="CSV files loaded and converted to JSON format successfully!")
```

Library	Category	Purpose
os, json	Core	File handling, JSON conversion and storage
csv	Core	Reading raw CSV files
datetime	Core	Handling and filtering date fields
pandas	Data Handling & Analysis	Data cleaning, transformation, statistical calculations
numpy	Data Handling & Analysis	Numerical operations (support for stats and data filtering)
statistics	Data Handling & Analysis	Additional statistical functions (e.g., mode, median)
matplotlib	Data Visualisation	Bar charts, line plots, general graphing
seaborn	Data Visualisation	Advanced plots (e.g., heatmaps, correlation matrix)
matplotlib.backends.backend_tkagg.FigureCanvasTkAgg	Data Visualisation	Embeds Matplotlib graphs in the Tkinter GUI
tkinter	GUI Development	Core GUI framework for creating the user interface
tkinter.filedialog	GUI Development	Enables file selection dialogs
tkinter.ttk.Notebook	GUI Development	Implements tabbed navigation in the interface
threading (not used)	Concurrency (Mentioned)	Mentioned for concurrent data loading/cleaning (not implemented as per brief)
sqlite3 (optional)	Storage (Optional)	Suggested for backing up data in database format instead of JSON

GUI Design & User Interaction

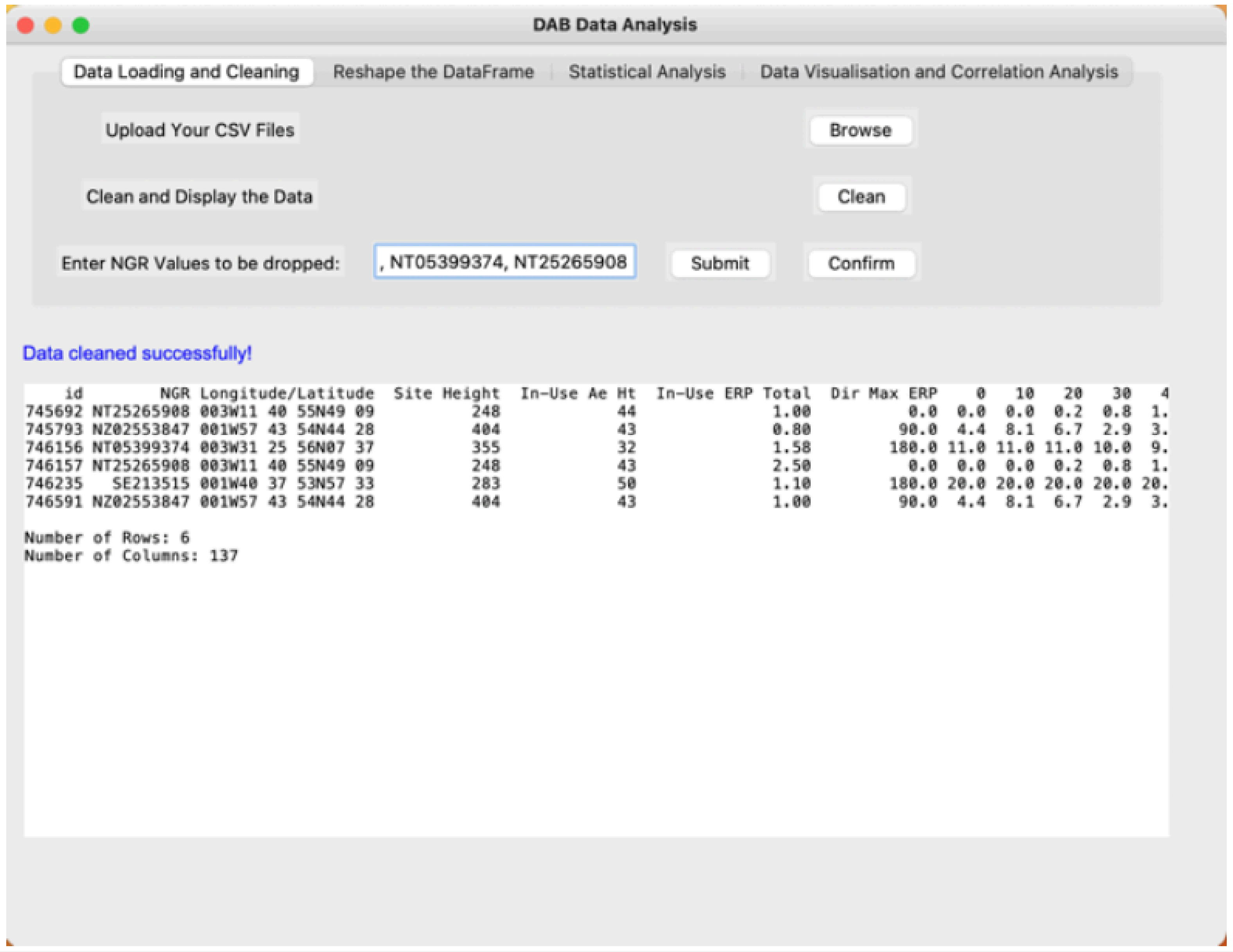
GUI Design & User Interaction

- Designed with a user-first approach using Tkinter
- Tab-based navigation using `tk.Notebook`
- Includes file dialog, data cleaning controls, output display, and status updates
- Visuals integrated using `FigureCanvasTkAgg`


DAB Data Analysis
Data Loading and Cleaning
Reshape the DataFrame
Statistical Analysis
Data Visualisation and Correlation Analysis
Upload Your CSV Files
Browse
Clean and Display the Data
Clean
Enter NGR Values to be dropped:
Submit
Confirm

Data cleaned successfully!

```
    id         NGR Longitude/Latitude  Site Height  In-Use Ae Ht  In-Use ERP Total  Dir Max ERP   0  10  20  30  40  50
745392 NO76418994 002W23 24 57N00 00          325           230              2.00         43.0 1.1 1.7 2.4 1.9 2.7 3.3
745393 NJ76043299 002W24 01 57N23 12          245           138              8.51        140.0 0.1 0.5 1.0 1.6 2.1 2.5
745394 NJ98315700 002W01 48 57N36 11          225            35              1.00         55.0 0.1 0.0 0.0 0.0 0.0 0.0
745395 NJ94270253 002W05 46 57N06 49           87            41              1.20        355.0 0.2 0.5 1.0 1.6 2.3 3.3
745396 NS29181617 004W41 59 55N24 35          273            34              4.00         70.0 0.4 0.0 0.0 0.0 0.3 0.5

Number of Rows: 1214
Number of Columns: 137
```

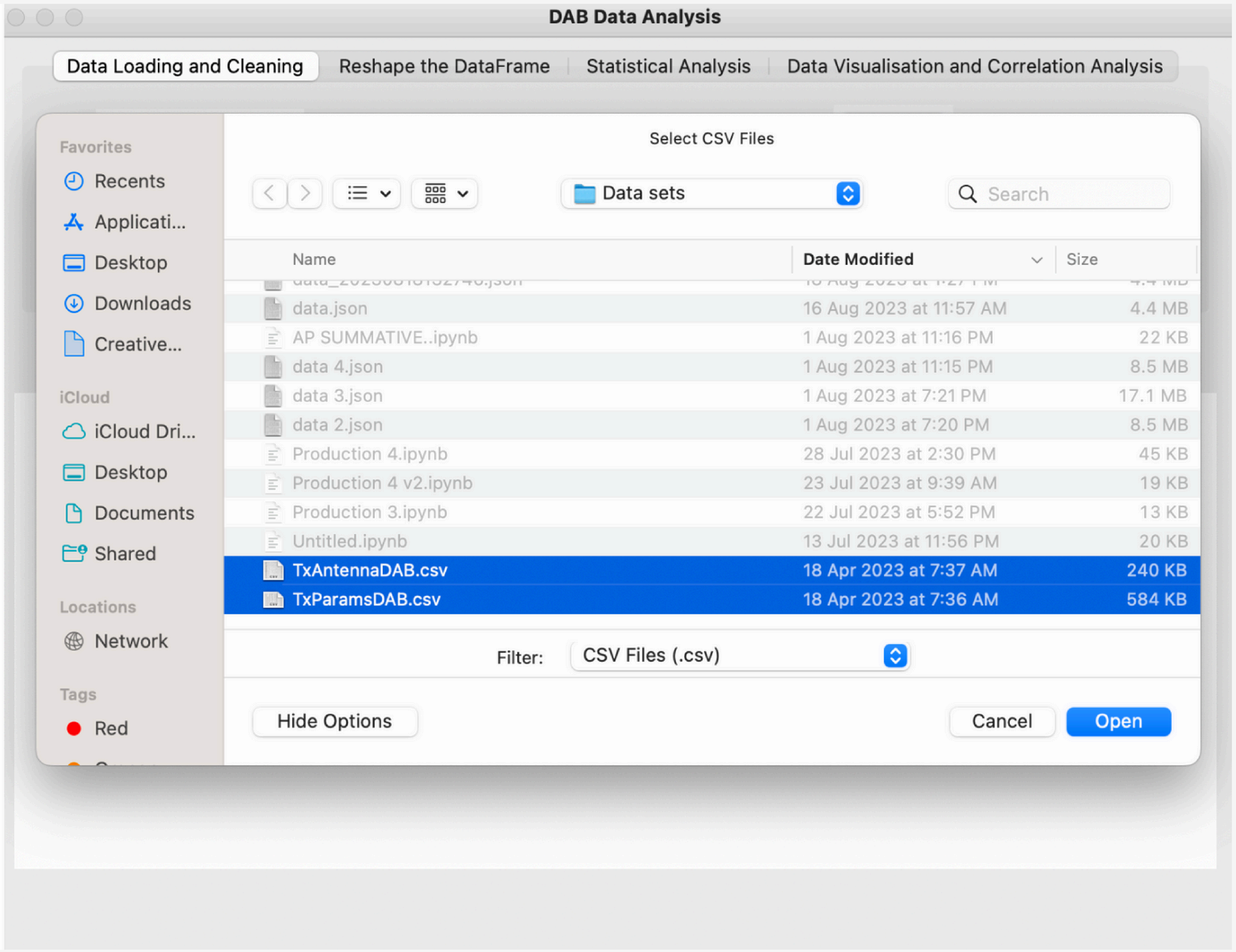

DAB Data Analysis
Data Loading and Cleaning
Reshape the DataFrame
Statistical Analysis
Data Visualisation and Correlation Analysis
Upload Your CSV Files
Browse
Clean and Display the Data
Clean
Enter NGR Values to be dropped:
Submit
Confirm

Enter NGR Values to be dropped: , NT05399374, NT25265908

Data cleaned successfully!

```
    id         NGR Longitude/Latitude  Site Height  In-Use Ae Ht  In-Use ERP Total  Dir Max ERP    0   10   20   30   4
745692 NT25265908 003W11 40 55N49 09          248            44              1.00          0.0  0.0  0.0  0.2  0.8  1.
745793 NZ02553847 001W57 43 54N44 28          404            43              0.80         90.0  4.4  8.1  6.7  2.9  3.
746156 NT05399374 003W31 25 56N07 37          355            32              1.58        180.0 11.0 11.0 11.0 10.0  9.
746157 NT25265908 003W11 40 55N49 09          248            43              2.50          0.0  0.0  0.0  0.2  0.8  1.
746235   SE213515 001W40 37 53N57 33          283            50              1.10        180.0 20.0 20.0 20.0 20.0 20.
746591 NZ02553847 001W57 43 54N44 28          404            43              1.00         90.0  4.4  8.1  6.7  2.9  3.

Number of Rows: 6
Number of Columns: 137
```

Data Cleaning Tab



DAB Data Analysis

Data Loading and Cleaning | Reshape the DataFrame | Statistical Analysis | Data Visualisation and Correlation Analysis

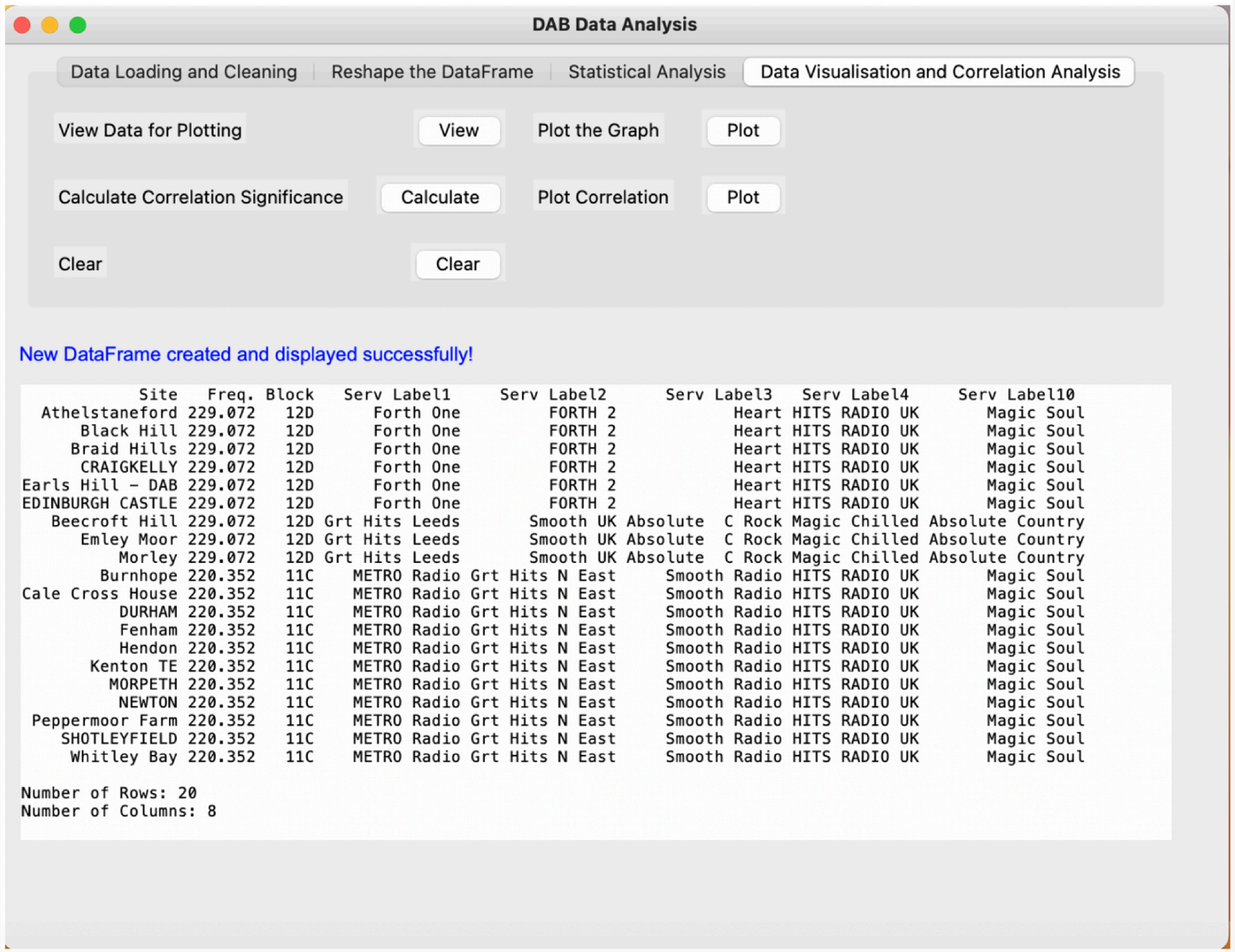
Select CSV Files

Data sets

Name	Date Modified	Size
data.json	16 Aug 2023 at 11:57 AM	4.4 MB
AP SUMMATIVE..ipynb	1 Aug 2023 at 11:16 PM	22 KB
data 4.json	1 Aug 2023 at 11:15 PM	8.5 MB
data 3.json	1 Aug 2023 at 7:21 PM	17.1 MB
data 2.json	1 Aug 2023 at 7:20 PM	8.5 MB
Production 4.ipynb	28 Jul 2023 at 2:30 PM	45 KB
Production 4 v2.ipynb	23 Jul 2023 at 9:39 AM	19 KB
Production 3.ipynb	22 Jul 2023 at 5:52 PM	13 KB
Untitled.ipynb	13 Jul 2023 at 11:56 PM	20 KB
TxAntennaDAB.csv	18 Apr 2023 at 7:37 AM	240 KB
TxParamsDAB.csv	18 Apr 2023 at 7:36 AM	584 KB

Filter: CSV Files (.csv)

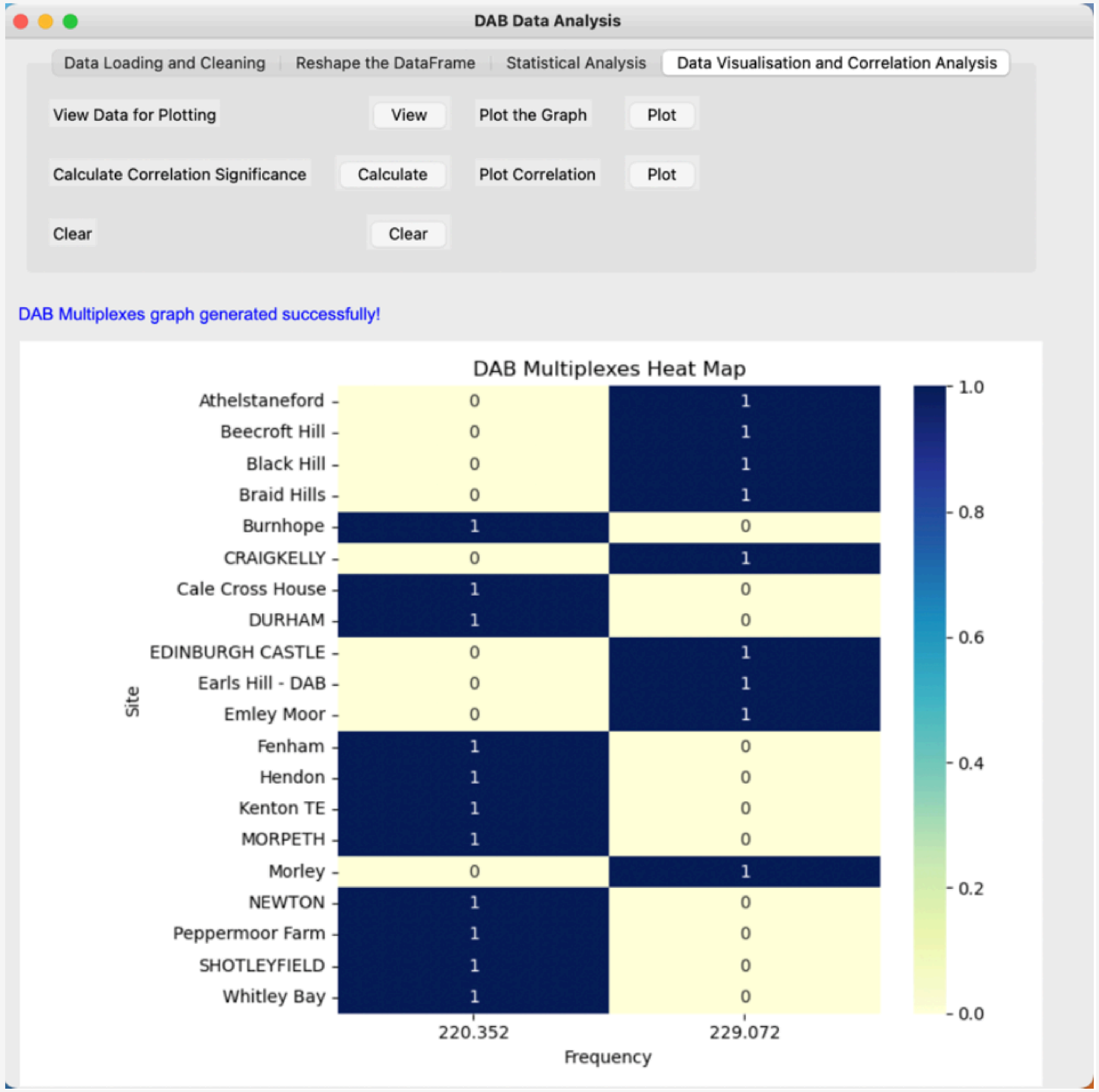
File Dialog for selecting CSV files (Data Import)



New DataFrame created and displayed successfully!

```
            Site   Freq. Block   Serv Label1     Serv Label2      Serv Label3   Serv Label4     Serv Label10
  Athelstaneford 229.072   12D      Forth One         FORTH 2            Heart HITS RADIO UK       Magic Soul
      Black Hill 229.072   12D      Forth One         FORTH 2            Heart HITS RADIO UK       Magic Soul
     Braid Hills 229.072   12D      Forth One         FORTH 2            Heart HITS RADIO UK       Magic Soul
      CRAIGKELLY 229.072   12D      Forth One         FORTH 2            Heart HITS RADIO UK       Magic Soul
Earls Hill - DAB 229.072   12D      Forth One         FORTH 2            Heart HITS RADIO UK       Magic Soul
EDINBURGH CASTLE 229.072   12D      Forth One         FORTH 2            Heart HITS RADIO UK       Magic Soul
   Beecroft Hill 229.072   12D Grt Hits Leeds       Smooth UK Absolute  C Rock Magic Chilled Absolute Country
      Emley Moor 229.072   12D Grt Hits Leeds       Smooth UK Absolute  C Rock Magic Chilled Absolute Country
          Morley 229.072   12D Grt Hits Leeds       Smooth UK Absolute  C Rock Magic Chilled Absolute Country
        Burnhope 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
Cale Cross House 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
          DURHAM 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
          Fenham 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
          Hendon 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
       Kenton TE 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
         MORPETH 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
          NEWTON 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
 Peppermoor Farm 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
    SHOTLEYFIELD 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul
     Whitley Bay 220.352   11C    METRO Radio Grt Hits N East     Smooth Radio HITS RADIO UK       Magic Soul

Number of Rows: 20
Number of Columns: 8
```

New DataFrame with Site, Freq, Block and the various Serv Labels



DAB Multiplexes graph generated successfully!

DAB Multiplexes Heat Map

Site	220.352	229.072
Athelstaneford	0	1
Beecroft Hill	0	1
Black Hill	0	1
Braid Hills	0	1
Burnhope	1	0
CRAIGKELLY	0	1
Cale Cross House	1	0
DURHAM	1	0
EDINBURGH CASTLE	0	1
Earls Hill - DAB	0	1
Emley Moor	0	1
Fenham	1	0
Hendon	1	0
Kenton TE	1	0
MORPETH	1	0
Morley	0	1
NEWTON	1	0
Peppermoor Farm	1	0
SHOTLEYFIELD	1	0
Whitley Bay	1	0

DAB Multiplex Visualisation

Solution Rundown

Key Functionalities

Feature	Status	Details
Load and translate CSV files	✅ Completed	Converts to JSON; supports multiple file selection
Data cleaning and preprocessing	✅ Completed	Handles missing values, removes specific NGRs, renames fields
Data filtering and transformation	✅ Completed	Filters DAB multiplexes (C18A, C18F, C188)
Statistical analysis	✅ Completed	Computes mean, mode, and median based on set conditions
Data visualisation	✅ Completed	Generates plots and heatmaps using Matplotlib and Seaborn
Correlation analysis	✅ Completed	Includes Pearson correlation matrix and heatmap
GUI with feedback/status messages	✅ Completed	Tabs, status bar, file dialog, embedded graphs
Data storage & backup (JSON)	✅ Completed	Allows saving and reloading cleaned datasets

Project Overview

The application allows the user to upload the three provided CSV files through the interface. Upon loading, the data is immediately parsed using pandas and translated into a structured JSON format. This format is preferred for its human-readability and compatibility with both frontend interfaces and backend storage. An option is also available to store the structured data into a SQLite database, which supports an entity-relationship structure and facilitates querying.

The system automatically stores the translated data in a local SQLite database. This serves as a persistent backup, ensuring that once the program is closed, the current cleaned and translated state of the data remains intact. Users can also export the current dataset as a .json or .db file for manual backup or transfer.

The application includes a cleaning pipeline that:

- Identifies and fills missing values using forward-fill, mean/mode imputation, or user-selected methods.
- Removes or replaces inconsistent data using predefined rules.
- Applies client-specific rules such as:
- Removing DAB stations with specified NGR values.
- Extracting DAB multiplexes (C18A, C18F, C188) from the EID column.
- Renaming fields like “In-Use Ae Ht” to “Aerial height (m)” and “In-Use ERP Total” to “Power (kW)”.

Users can view, modify, and confirm the cleaned data via the GUI.

Project Overview

A responsive graphical interface (built using Jupyter Notebook widgets / Tkinter / Flask frontend*) offers:
• File Upload Panel: to load CSVs and trigger cleaning.
• Data Cleaning Dashboard: where users can view raw data, run cleaning scripts, and inspect anomalies.
• Data Storage Panel: to save and reload processed datasets (JSON or from DB).
• Visualization Module: allowing users to select metrics and display charts (bar, boxplot, scatter, etc.).
• Range Controls: e.g. sliders or input fields, allowing users to filter data based on values such as site height, date, or frequency before generating insights.

The program is built to validate the column headers of new CSVs to confirm schema compatibility before processing. It's designed to adapt to variations in row values, including missing or malformed data entries, using flexible parsing logic. This ensures scalability for future data collected from the same source system.

The backend follows modular and reusable practices:
• DataLoader module: handles CSV-to-JSON and database insertion
• Cleaner module: implements a chain of cleaning functions
• Exporter module: for saving/retrieving translated data
• Visualizer module: generates insights based on filtered inputs
• GUI module: separates UI logic from backend processing

All major functions are encapsulated in reusable methods, and a clear API is exposed internally for GUI components to interact with the backend logic.

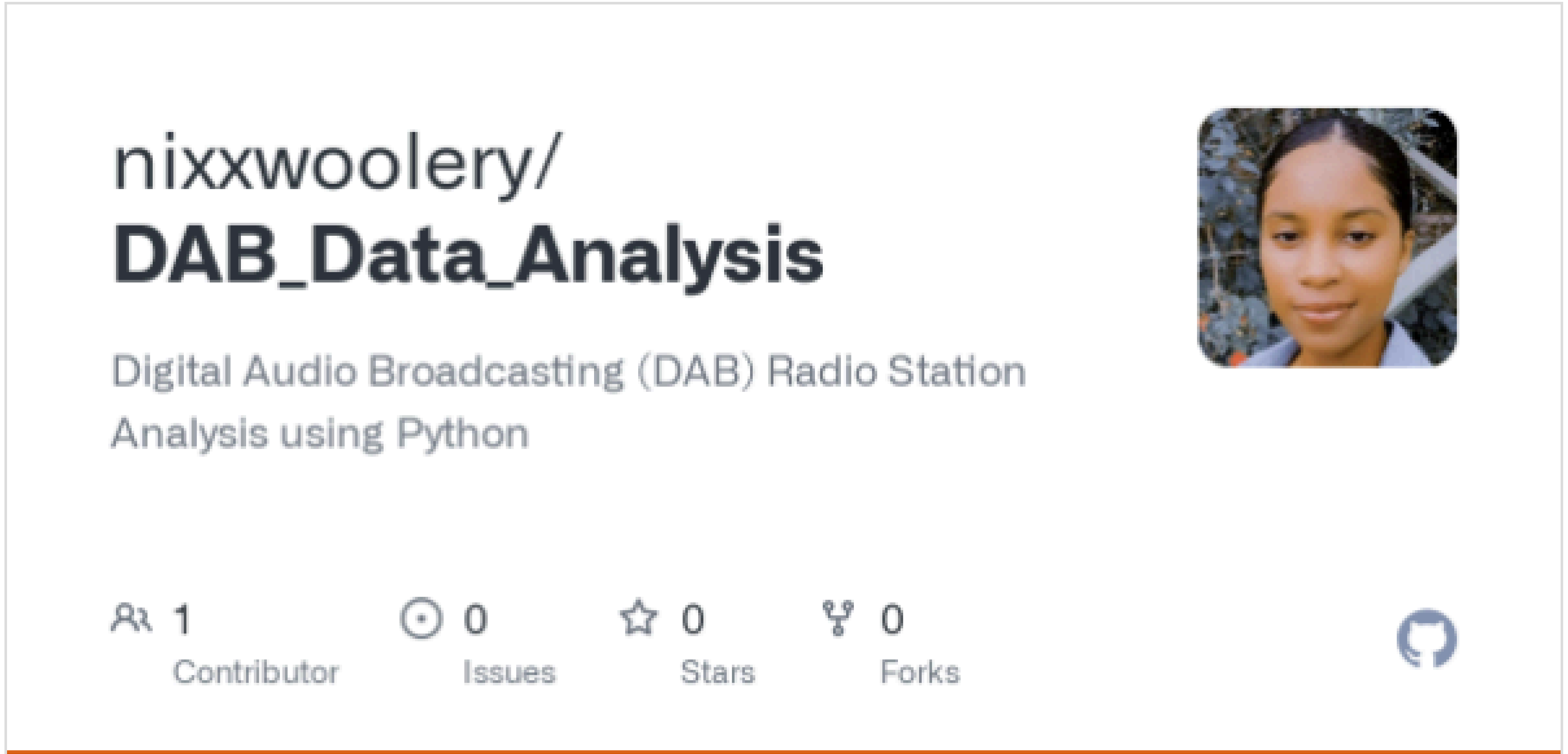
Limitations

• No threading/concurrency allowed as per technical constraints, which limits performance on large datasets.
• Only works with datasets of the same schema — not fully generic.
• Currently only supports JSON format; database backup is not implemented.

Next Steps & Future Considerations

• Implement database support (e.g., SQLite) for scalable backups.
• Add dynamic filtering and advanced visualizations.
• Convert the project into a standalone desktop application.
• Refactor code to allow threaded operations for performance (if constraints lifted).
• Add export to Excel or PDF for reports.

Project Link



nixxwoolery/
DAB_Data_Analysis

Digital Audio Broadcasting (DAB) Radio Station Analysis using Python

1 Contributor · 0 Issues · 0 Stars · 0 Forks

nixxwoolery/DAB_Data_Analysis: Digital Audio Broadcasting (DAB) Radio Station Analysis using Python

Digital Audio Broadcasting (DAB) Radio Station Analysis using Python - nixxwoolery/DAB_Data_Analysis

GitHub

How to Run

System Requirements

- Python Version: 3.7 – 3.10
- Environment: Anaconda (Recommended)
- Interface: Jupyter Notebook
- Platform: Windows, Mac, or Linux

Required Libraries

```
pip install pandas numpy matplotlib seaborn
```

Steps to Run

1. Launch Anaconda Navigator
Open Anaconda Navigator and launch Jupyter Notebook.

2. Navigate to Project Folder
In the Jupyter file browser, navigate to the folder where your notebook and data are saved.

3. Open the Notebook
Click to open dab_data_analysis.ipynb.

4. Run the Notebook
• Go to Kernel and click on “Restart and run all cells”.
• Follow on-screen instructions in the GUI (Tkinter window will open).

Steps to Run

5. Load CSV Files
• Use the Browse button in the GUI to select and load the three CSV files.
• The data will be converted to JSON, cleaned, and displayed.

6. Clean and Analyze Data
• Perform data cleaning by removing NGRs, renaming columns, and filtering.
• View statistical analysis and generate visualisations via tabs in the GUI.

7. Save and Reload Cleaned Data (Optional)
• Save cleaned data using the GUI.
• Reload it later from JSON format to continue where you left off.

Thank you.