The application should provide the following functionality:

- A means to load the initial data set (which consists of 2 CSV files) and translate it into a database (I used pgAdmin 4)
- A means to back up the suitable format using 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

It should be assumed that this program will 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.

Technical requirements

- The application is built using Python 3.7 - 3.10
- The application uses one or more of the advanced APIs introduced on this module such as: NumPy, pandas, Seaborn, Matplotlib. It should NOT use alternative APIs for this functionality, however Python core libraries can be used to support where applicable, such as support for a database.
- The application runs within the anaconda environment using a Jupyter notebook
- The application or its parts do not run concurrently, do NOT use Python threads

Non-functional requirements

- The GUI interface must be able to provide appropriate feedback to confirm or deny a user's actions
- The application must be able to manage internal and user-generated errors

Data manipulation and outputs

1. Outputs should not include any data from DAB Radio stations that have the following **'NGR'** : NZ02553847, SE213515, NT05399374 and NT252675908
2. The **'EID'** column contains information of the DAB multiplex block E.g C19A. Extract this out into a new column, one for each of the following DAB multiplexes:
    a. all DAB multiplexes, that are , **C18A**, **C18F**, **C188**
    b. join each category, **C18A**, **C18F**, **C188**
    to the **' NGR'** that signifies the DAB stations location to the following: **'Site',** '**Site Height, In-Use Ae Ht, In-Use ERP Total**
    c. Please note that: **In-Use Ae Ht, In-Use ERP Total** will need the following new header after extraction: Aerial height(m), Power(kW) respectively.
3. The application needs information to generate the following and output the results using appropriate representation:
    a. Produce the mean, mode and median for the **'In-Use ERP Total'** from the extracted DAB multiplexes extracted earlier: **C18A**, **C18F**, **C188**
        i. For '**Site Height'** more than 75
        ii. For **'Date'** from 2001 onwards
4. Produce a suitable graph that display the following information from the three DAB multiplexes that you extracted earlier: **C18A**, **C18F**, **C188: 'Site', 'Freq', 'Block', 'Serv Label1', 'Serv Label2', 'Serv Label3', 'Serv label4','Serv Label10'**
   You may need to consider how you group this data to make visualisation feasible
5. Determine if there is any significant correlation between the **'Freq', 'Block', 'Serv Label1', 'Serv Label2', 'Serv Label3', 'Serv label4','Serv Label10'** used by the extracted DAB stations. You will need to select an appropriate visualisation to demonstrate this.

Please provide references to this reports development and explanation .