Project Description: CSV Data Cleaner

Overview:

The primary objective of this project is to create a robust and versatile class named `CleanerCSV`. This class should be adept at processing CSV datasets, encapsulating multiple common data cleaning tasks within its arsenal.

Key Features:

1. **Initialization with Specifications** [Optional]:
    - The class should offer flexibility during instantiation, permitting the usage of a custom specification file.
    - A default specification file, named "default_specs.json", should be employed if no custom file is provided.
2. **Load Specifications from JSON**:
    - The class should incorporate a method to load cleaning parameters and instructions from a given JSON specification file.
3. **Data Reading**:
    - A method dedicated to reading a CSV dataset, preserving both its original version and a deep copy for manipulation.
4. **Comprehensive Data Cleaning**:
    - The core of the class, comprising a suite of methods to handle:
        - Deduplication of rows.
        - Exclusion of NaN value-containing rows.
        - Column removal based on specification.
        - Trimming white spaces in specific string columns.
        - Removal of rows identical to header rows.
        - Replacement of specific characters in rows.
        - Data type corrections for specified columns.
        - Outlier management based on the Interquartile Range (IQR).
5. **Data Export**:
    - A utility to save the cleaned dataset to a specified CSV file.
6. **Data Profiling**:
    - Integrating the `ydata_profiling` module, the class should be able to generate a detailed report showcasing insights, distributions, and potential issues for both the original and cleaned data.
7. **Row Counting Utility**:
    - A succinct method to return the count of rows in the cleaned dataset.

8. **Logging**:
    - Incorporate logging to track the flow and status of operations, assisting in debugging and understanding the cleaning process.

Requirements:

- **Libraries**:
    - `pandas`: For dataframe manipulation.
    - `numpy`: For mathematical operations.
    - `json`: To read the specification JSON file.
    - `logging`: To handle logging.
    - `ydata_profiling`: For data profiling.
- **Specification File**:
    - The JSON specification file should outline all cleaning instructions, such as columns to drop, columns to correct data types for, characters to replace, and so forth.

Specification Example (specs.json)

```
{
    "input_file" : "data/my_data.csv",
    "output_file" : "data/my_data_clean1.csv",
    "delimiter": ",",

    "input_file_profile": "profiles/input_file.html",
    "output_file_profile": "profiles/output_file.html",
    "summary_file": "profiles/summary.txt",



    "drop_duplicates" : true,
    "drop_na" : true,
    "clean_str_columns" : true,
    "drop_row_title": true,
    "replace_row_char": true,
    "clean_outliers": true,
    "export_output_file": true,


    "str_col": ["Product","Purchase Address"],
    "float_col": [],
    "int_col": [],
    "numeric_col": ["Order ID", "Quantity Ordered", "Price Each"],
    "datetime_col": ["Order Date"],

    "drop_col" : [],
    "col_outlier" : ["Price Each"],
```

```
    "replace_row_char_details": {"col": ["Order ID"], "change": {"@":""}},
    "statistical_outliers" :["Product","Price Each","95","15"]
}
```

Milestones:

1. **Setup & Initialization**:
    - Set up the project environment, including the installation of required libraries.
    - Draft the initialization function for the `CleanerCSV` class.
2. **Loading & Reading Data**:
    - Develop the methods to load the JSON specification and read the CSV data.
3. **Data Cleaning Functions**:
    - Implement each cleaning function one by one, ensuring that each adheres to the specifications from the JSON file.
4. **Utilities Development**:
    - Construct the data export, profiling, and row counting functionalities.
5. **Documentation**:
    - Document all methods and functionalities within the codebase using Docstrings.
    - Craft a comprehensive user manual or guide detailing the usage of the class and the structure of the specification JSON.

Deliverables:

- The finalized `CleanerCSV` class code.
- Sample specification JSON files.
- A comprehensive user guide or documentation.
- Sample datasets for testing purposes.
- A demonstration showcasing the cleaning of a dataset using the class.

By the conclusion of this project, the developer should have a versatile CSV cleaning toolkit, embodied within the `CleanerCSV` class, capable of catering to a myriad of data cleaning needs with ease and efficiency.