### **Task 3: Creating a Custom Aggregation Function**

**Objective & Rationale** pandas.groupby() is a cornerstone of data analysis, but it's often limited to standard aggregations like sum or mean. This application was designed to demonstrate how to extend this functionality by defining and applying a completely custom aggregation function. For this example, we chose to calculate the product of values, a non-standard aggregation useful in specialized scenarios.

**Our Approach** The project was designed with flexibility and user transparency in mind.

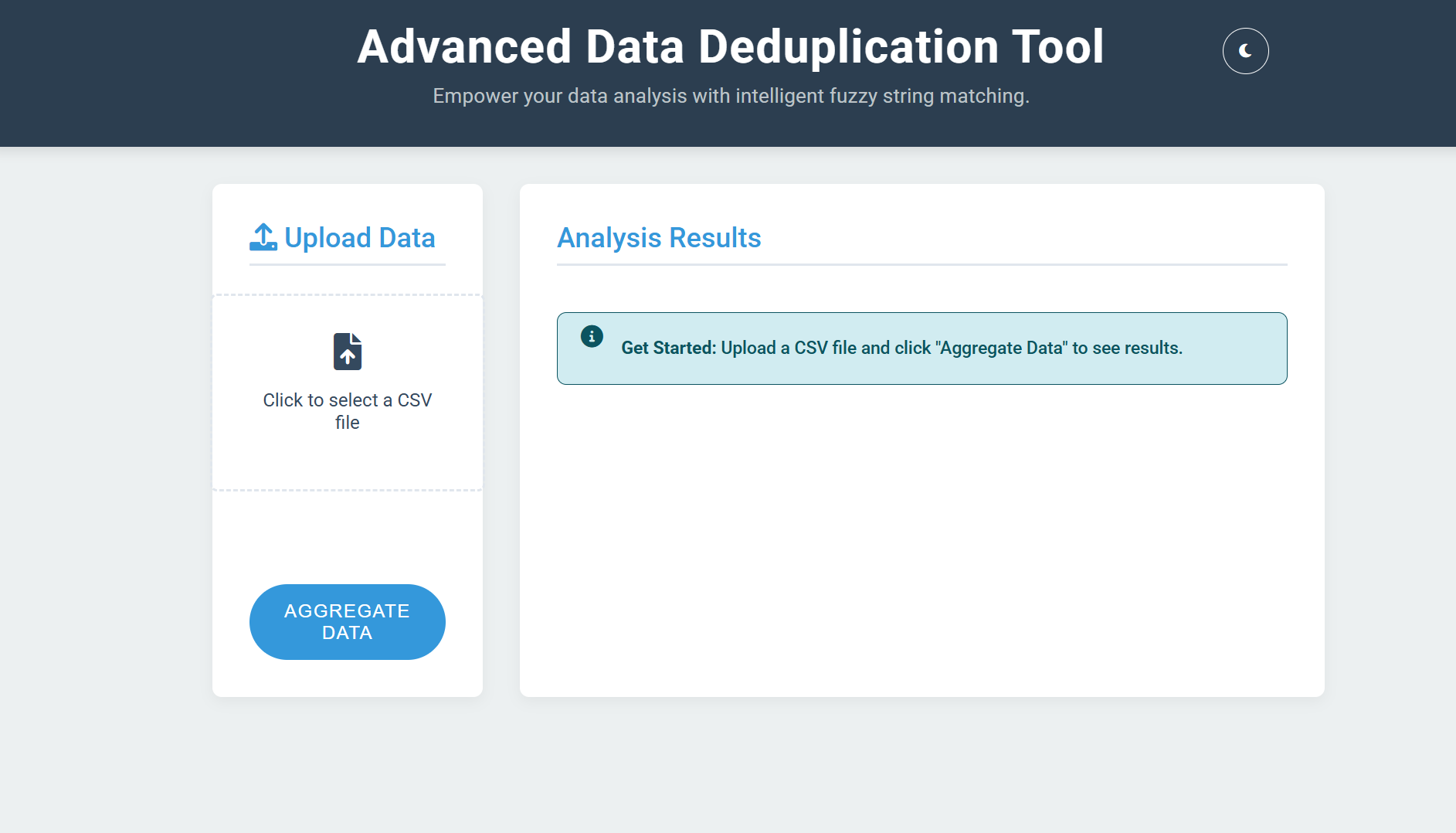
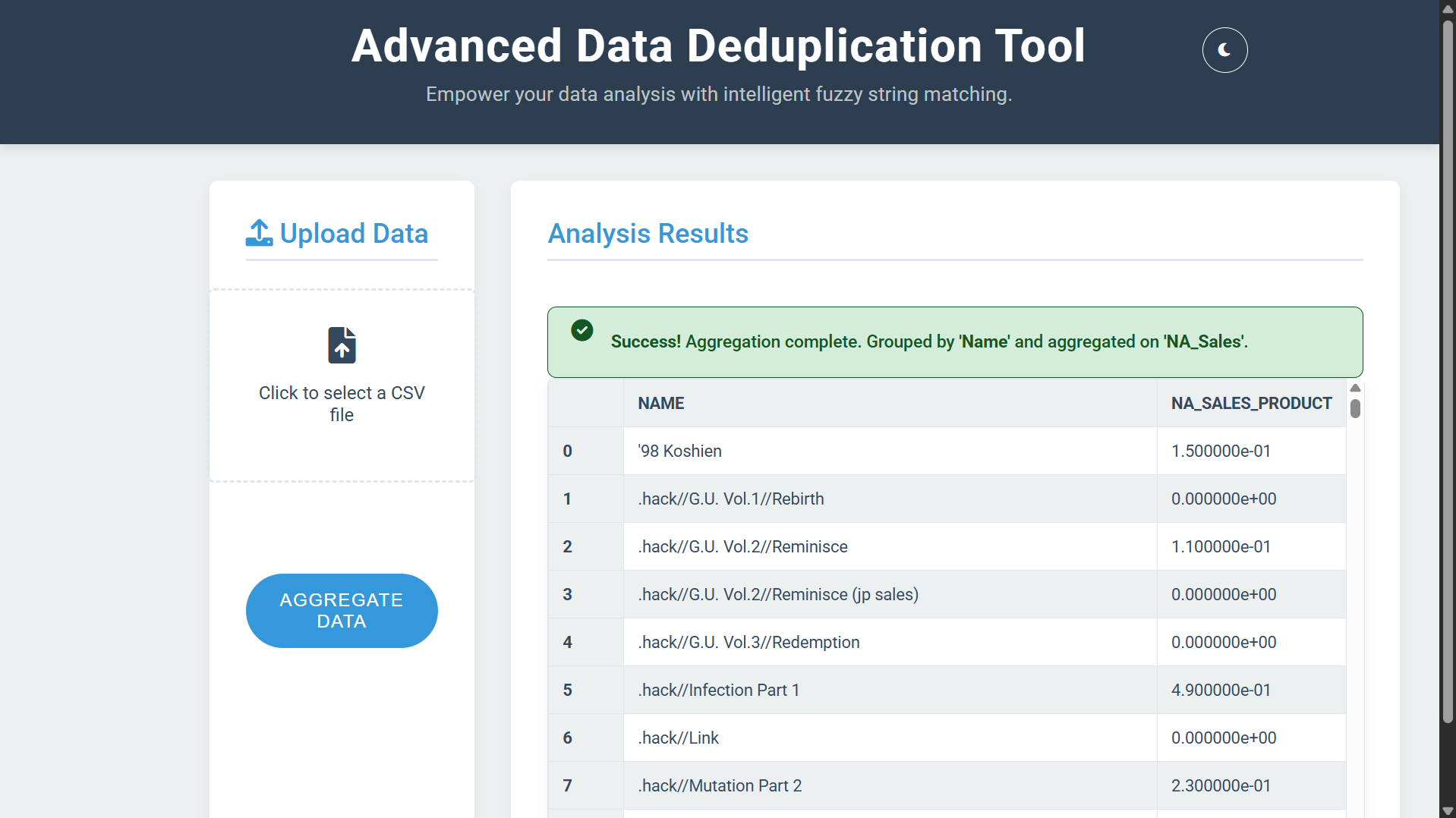
1. **Intelligent Column Detection**: When a user uploads a CSV, the backend automatically detects a suitable categorical column for grouping (e.g., 'Publisher' or 'Category') and a numeric column for aggregation (e.g., 'Global\_Sales'). This removes the need for manual configuration and makes the tool compatible with a wide variety of datasets.
2. **The Custom product\_agg Function**: We defined a custom function, product\_agg, which uses np.prod() to efficiently multiply all the elements in a given data series. This showcases how custom logic can be seamlessly integrated into the groupby pipeline.
3. **Clear Results**: The output is an HTML table that clearly displays the aggregated data. A success message confirms which columns were used, providing clarity and building user confidence in the results.

**Key Takeaway** This project is a powerful example of how to extend the core functionality of pandas to meet a specific analytical need. It highlights the flexibility of Python's data stack for solving unique and complex data problems.

**GitHub-Repo:**[**https://github.com/Kuldeep-Tapodhan/pandas-custom-aggregator**](https://github.com/Kuldeep-Tapodhan/pandas-custom-aggregator)

**Sample Output** Aggregating the Global\_Sales of the vgsales.csv file by Publisher would produce a table like this:

**Output ScreenShots:**

****