**Sắp xếp và lọc dữ liệu ( Sort và Filter )**

**1.Ý tưởng**

Mong muốn sẽ sắp xếp giá trị các cột từ thấp tới cao hoặc từ cao tới thấp. Nếu là các cột có giá trị là ký tự chữ thì sẽ sắp xếp xuôi hoặc ngược theo bảng chữ cái.

Đầu tiên sẽ tạo file sắp xếp và lọc các dữ liệu trong python. Rồi sẽ tinh chỉnh, sửa đổi để phù hợp trở thành file logic chứa các hàm xử lý dữ liệu độc lập.

Tiếp theo, sẽ tạo file làm cầu nối giữa giao diện và phần xử lý dữ liệu : nhận dữ liệu từ giao diện rồi gọi hàm xử lý ở file logic, rồi trả kết quả về giao diện.

**2.Thực hiện file logic**

a. Sắp xếp ( Sort ) :

\*Sau khi sửa :

Import thư viện pandas để phục vụ cho việc sắp xếp dữ liệu (DataFrame).

Hàm chính là sorted\_dt : sắp xếp dữ liệu theo cột cụ thể ( sort\_by ) và kiểu sắp xếp ( order ) : Tăng dần hoặc giảm dần

A black background with white text

Description automatically generated

Đầu tiên, xác định thứ tự sắp xếp bằng biến ascending, nếu order ( kiểu sắp xếp ) là ‘asc’ : kiểu sắp xếp tăng dần thì ascending = True còn không thì ascending = False



Sử dụng sort\_values của pandas để thực hiện sắp xếp với 2 giá trị trong ngoặc là cột muốn sắp xếp (by = sort\_by) và thứ tự sắp xếp (ascending). Hàm sort\_values() được sử dụng để sắp xếp dữ liệu trong DataFrame và hỗ trợ sắp xếp tăng dần hoặc giảm dần, hỗ trợ tốt và tối ưu hiệu suất xử lý.

Trong đó :

* by = sort\_by ( tên cột muốn sắp xếp).
* ascending : nếu là True thì xếp tăng dần, là False thì xếp giảm dần.

Sử dụng try, except xử lý lỗi nếu cột chỉ định không tồn tại, khi đó sẽ trả về dữ liệu gốc.

A black background with white text

Description automatically generated

Cuối cùng là trả về giá trị đã sắp xếp dưới dạng DataFrame.



b. Lọc ( Filter ) :

Import thư viện pandas để phục vụ việc xử lý dữ liệu và thư viện datetime để phục vụ việc xử lý dữ liệu thời gian.

Hàm chính là filter\_dataframe : lọc dữ liệu dựa trên các tiêu chí sau :

* Thời gian :

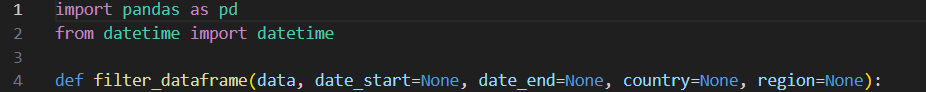
date\_start : điều kiện để lọc các dữ liệu từ mốc thời gian này

date\_end : điều kiện để lọc các dữ liệu từ mốc thời gian này trở lại.

* Quốc gia : country : lọc theo tên quốc gia ở cột ‘Country/Region’ trong bảng dữ liệu.
* Khu vực : region : lọc theo khu vực ở cột ‘WHO Region’ trong bảng dữ liệu.

Và các tham số date\_start, date\_end, country, region được đặt giá trị là None, vì để None sẽ là giá trị mặc định cho các tham số này, để khi nếu tham số không được truyền giá trị thì nó sẽ dùng None làm giá trị mặc định. Từ đó, thoả mãn việc có thể lọc với 1 hoặc nhiều điều kiện.

Ý nghĩa: biểu thị tham số đó là không bắt buộc, nếu tham số không được cung cấp, hàm sẽ bỏ qua khi thực hiện, điều này đáp ứng việc lọc với một số điều kiện nhất định.



Tiếp theo là xử lý dữ liệu cột ‘Date’ bằng cách chuyển sang định dạng datetime để so sánh chính xác các giá trị ngày tháng vì lúc đầu ở dữ liệu gốc thường sẽ ở dạng chuỗi ( string ), không thể so sánh trực tiếp.

Xử lý bằng hàm pd.to\_datetime của pandas giúp chuyển đổi và xử lý lỗi nếu dữ liệu ngày tháng không hợp lệ với trong ngoặc là 2 tham số :

* arg là data[‘Date’], là đầu vào cần chuyển đổi, ở đây đang là dữ liệu ở cột ‘Date’
* format : là chuỗi định dạng ngày tháng đầu vào, ở đây đang là định dạng kiểu ‘%d/%m/%Y’.

Hàm pd.to\_datetime được dùng để chuyển đổi giá trị ngày tháng thành đối tượng datetime của pandas và chuẩn hoá dữ liệu ngày tháng để đảm bảo dữ liệu có thể sử dụng để tính toán hoặc lọc.



Tiếp đến là lọc theo ngày bắt đầu, nếu date\_start hay date\_emd tồn tại giá trị khác None, thì thực hiện :

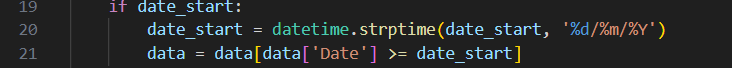
Xử lý chuyển chuỗi date\_start và date\_end thành kiểu datetime để so sánh. Ở đây sử dụng datetime.strptime vì hàm này trong pandas tối ưu hớn đối với chuyển đổi một giá trị duy nhất. Nhưng thay vào đó cần phải định dạng rõ ràng. Còn pd.to\_datetime thì không tối ưu bằng vì nó được thiết kế để xử lý dữ liệu với số lượng lớn, danh sách.

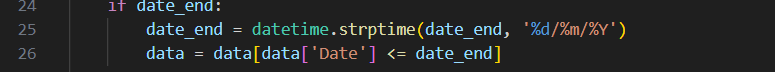
Dùng hàm datetime.strptime với 2 tham số là đối tượng muốn chuyển đổi ( là date\_start và date\_end ) và định dạng datetime đầu vào ( ‘%d/%m/%Y’ ).

Tiếp theo là lọc các dữ liệu trong cột ‘Date’ theo điều kiện :

* Nếu lọc theo date\_start thì lấy các giá trị lớn hơn hoặc bằng date\_start
* Nếu lọc theo date\_end thì lấy các giá trị nhỏ hơn hoặc bằng date\_end

Đặt điều kiện lọc này trong data[ ] để lọc lấy các hàng thoả điều kiện.





Còn lọc theo country và region thì cũng sử dụng data[ ] với điều kiện lọc trong [] là :

* Với country : lọc giá trị cột ‘Country/Region’ theo tên quốc gia ( country )
* Với region : lọc giá trị cột ‘WHO Region’ theo tên khu vực (region )





Cuối cùng là trả về dữ liệu đã lọc dưới dạng DataFrame



**3. Thực hiện file cấu nối giữa giao diện và phần xử lý logic**

a. Sắp xếp dữ liệu ( sort\_route )

Ý tưởng : Triển khai endpoint /sort cho phép người dùng sắp xếp dữ liệu theo cột mong muốn và thứ tự sắp xếp ( tăng dần hoặc giảm dần ), lấy dữ liệu người dùng chọn, nhập từ giao diện web về để xử lý. Dữ liệu đã được sắp xếp sẽ được hiển thị lên giao diện web kèm phân trang

Đầu tiên, cài các thư viện cần thiết vào :

Dòng đầu tiên :

Cài Blueprint trong flask, đây là 1 thành phần của flask được dùng tổ chức và tách biệt các nhóm chức năng (routes) của ứng dụng web. Cho phép tạo sort là một module route riêng biệt thay vì phải định nghĩa tất cả các route trong 1 file lớn.

Cài request, 1 thành phần của flask, dùng để lấy dữ liệu từ HTTP request ( yêu cầu của người dùng gửi đến Server ).

Cài render\_template, dùng để kết nối dữ liệu từ backend với giao diện frontend web.

Dòng 2 : Cài module paginate có trong routes, một module tuỳ chỉnh tự tạo trong folder routes, trong đây chứa hàm paginate\_data để phân trang dữ liệu, chia dữ liệu thành các trang nhỏ.

Dòng 3 : Cài thư viện pandas để xử lý dữ liệu

Dòng 4 : Hàm tuỳ chỉnh sorted\_dt trong file Sort\_df thực hiện logic sắp xếp dữ liệu.

A black background with multicolored text

Description automatically generated

Tiếp tục là định nghĩa đường dẫn : đường dẫn đến file dữ liệu chưa sạch ban đầu và đường dẫn đến file đầu ra lưu dữ liệu đã được sắp xếp.

Xác định số dòng dữ liệu sẽ hiển thị trên web là 12

A computer screen shot of a computer program

Description automatically generated

Tiếp theo, khởi tạo blueprint, vì blueprint trong Flask dùng để tổ chức và tách biệt các nhóm chức năng (routes) của ứng dụng web. Có thể tách các routes thành các module nhỏ (blueprint). Mỗi blueprint chứa các route cho một chức năng cụ thể : ví dụ blueprint sort\_bp quản lý các route liên quan đến sắp xếp dữ liệu (/sort).

Ở đây dùng hàm Blueprint với các tham số trong ngoặc:

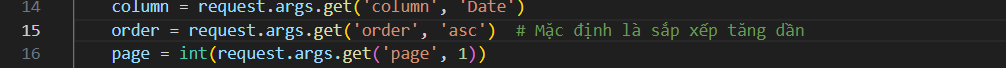
* ‘sort’ : đây là tên được gán cho blueprint này ( là định danh nội bộ, giúp Flask quản lý các blueprint )
* \_\_name\_\_ : tên của file hiên tại vì flask cần biết file nào chưa blueprint này
* url\_prefix=‘/sort’ : định nghĩa tiền tố cho các route trong blueprint này là /sort



Tiếp đến, định nghĩa route cho blueprint sort\_bp bằng hàm @sort\_bp.route với 2 tham số trong ngoặc : ‘/sort’ là url đã được định nghĩa trước đó và methods chứa 2 yêu cầu là ‘GET’ ( lấy dữ liệu đã sắp xếp ) và ‘POST’ ( yêu cầu gửi dữ liệu từ người dùng )



Tiếp theo là hàm sort\_data, trong đó lấy giá trị column, order và page từ url bằng hàm request.args.get( ), nếu không có tham số lấy được từ url thì column sẽ mặc định lấy cột ‘Date’, order thì sẽ mặc định là ‘asc’ ( tăng dần ), page mặc định là 1 (trang đầu ).



Tiếp đến thì đọc dữ liệu từ file csv, rồi sắp xếp dữ liệu theo cột, rồi lưu vào file đầu ra. Đặt số dòng dữ liệu hiển thị trên mỗi trang là 10.

Dùng paginate\_data để chia nhỏ dữ liệu của sortData thành từng trang. Khi này biến pagination lưu kết quả phân trang gồm :

* pagination["table\_data"]: Dữ liệu của trang hiện tại.
* pagination["total\_pages"]: Tổng số trang.
* pagination["has\_next"]: Có trang tiếp theo không?
* pagination["has\_prev"]: Có trang trước đó không?
* pagination["nearby\_pages"]: Các số trang gần kề.

Và cuối cùng là trả kết quả về giao diện : render\_template sẽ kết nối dữ liệu từ backed với giao diện html, ở đây, template sort.html sẽ hiển thị kết quả sắp xếp. Các tham số được truyền vào template

* table\_data: Dữ liệu của trang hiện tại (sau khi sắp xếp và phân trang).
* headers: Tên các cột trong bảng dữ liệu, được dùng để tạo tiêu đề cho bảng.
* page và total\_pages:
* Số trang hiện tại và tổng số trang. has\_next và has\_prev:
* Thông tin phân trang: Có trang tiếp theo/trang trước không?
* nearby\_pages: Danh sách các số trang gần kề, giúp người dùng điều hướng dễ dàng.
* current\_sort: Thông tin về cột và thứ tự sắp xếp hiện tại (hiển thị trạng thái sắp xếp trên giao diện).

A screen shot of a computer code

Description automatically generated

b. Lọc dữ liệu ( filter\_route )

Ý tưởng : Triển khai endpoint /filter cho phép người dùng lọc dữ liệu và hiển thị kết quả lọc lên giao diện web kèm phân trang.

Đầu tiên, cài các thư viện cần thiết vào, cũng tương tự như ở sort\_route

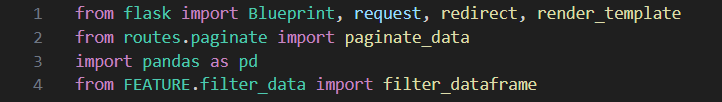
Dòng đầu tiên :

* Cài Blueprint để tạo module filter\_bp để quản lý các route liên quan đến lọc dữ liệu.
* Cài request để xử lý dữ liệu từ http request
* Cài redirect để chuyển hướng đến trang khác ( trong trường hợp này là trang hiển thị dữ liệu đã lọc )
* Cài render\_template để hiển thị giao diện html

paginate\_data : hàm dùng để phân trang dữ liệu

pandas : cài thư viện pandas để xử lý dữ liệu

filter\_dataframe : hàm tuỳ chỉnh từ file filter\_data.py, thực hiện logic chính để lọc dữ liệu theo điều kiện



Tương tự như ở sort\_route, tiếp theo là định nghĩa đường dẫn, đọc file và đặt số dòng hiển thị trên 1 trang là 12.

Tiếp đến là tạo blueprint với tiền tố là /filter. Sau đó, định nghĩa route cho blueprint filter\_bp bằng hàm @filter\_bp.route với 2 tham số trong ngoặc : ‘/filter’ là url đã được định nghĩa trước đó và methods chứa 2 yêu cầu là ‘GET’ (lấy dữ liệu đã lọc và hiển thị kết quả lên giao diện) và ‘POST’ ( xử lý dữ liệu người dùng nhập )

A computer screen with text

Description automatically generated

Khi nhấn nút lọc dữ liệu, thực hiện lấy dữ liệu từ form html :

Dùng request.form.get() để lấy dữ liệu người dùng nhập từ form:

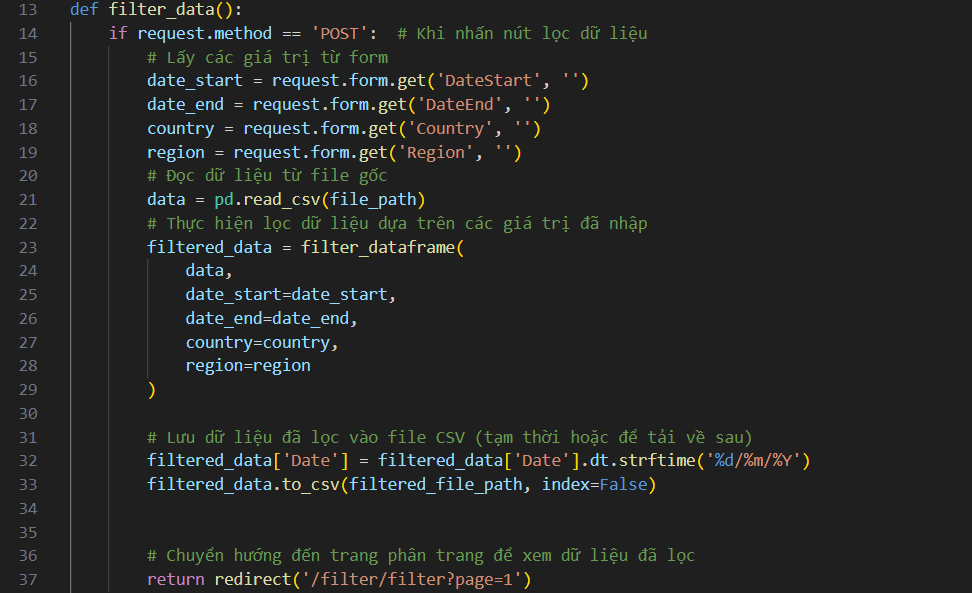
* date\_start, date\_end : Ngày bắt đầu và kết thúc để lọc dữ liệu.
* country, region : Lọc theo quốc gia hoặc vùng.

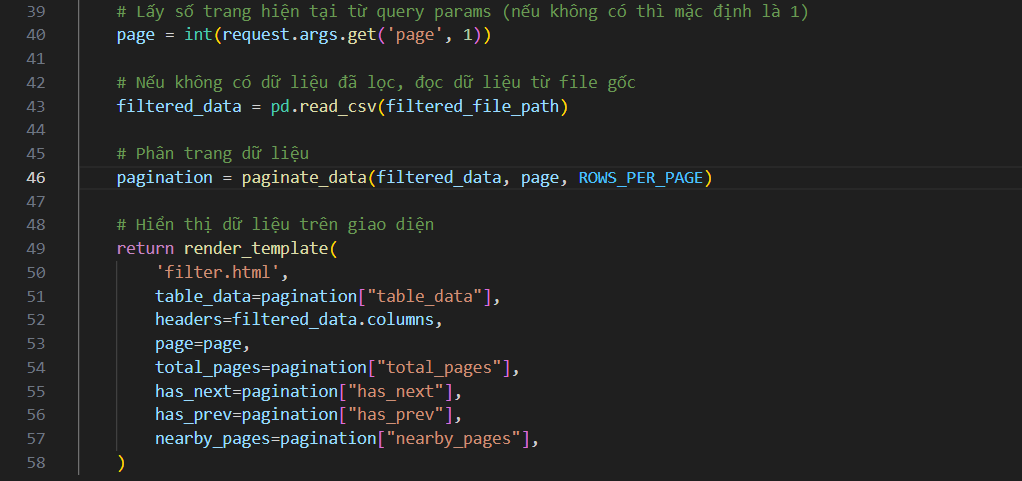
Tiếp đến, đọc dữ liệu từ file csv gốc, lưu vào data. Rồi thực hiện lọc dữ liệu dựa trên tham số được lấy dữ liệu từ người dùng nhập từ form.

Rồi trong dữ liệu đã lọc, chuyển định dạng của cột ‘Date’. Sau đó, lưu dữ liệu đã lọc vào file csv tạm thời hoặc để tải sau. Sau khi lọc, chuyển hướng người dùng đến trang đầu tiên của kết quả.

Tiếp đến là xử lý yêu cầu get ( hiển thị dữ liệu đã lọc ) :

* Lấy trang hiện tại từ URL: Lấy tham số page từ URL. Nếu không có, mặc định là trang 1.
* Đọc dữ liệu đã lọc.
* Phân trang: Gọi hàm paginate\_data để chia dữ liệu thành các trang nhỏ, chỉ trả về dữ liệu của trang hiện tại.
* Hiển thị giao diện: render\_template(): Gửi dữ liệu đã lọc và phân trang tới file HTML để hiển thị.





4. Tóm gọn

Tạo 2 file xử lý logic gồm hàm sắp xếp và lọc, tạo 2 file tương ứng để tương tác với trên web, sao cho người dùng nhập và chọn chức năng sắp xếp hoặc lọc sẽ gửi dữ liệu về file tương tác, tại đây nó sẽ gọi hàm trong file logic để xử lý dữ liệu và trả về kết quả, kết quả sẽ được hiển thị lại trên web.

**Clean trong Sort và Filter**

**1.Ý tưởng**

Ngoài việc tạo file sort và filter, còn cần có file clean để **làm sạch dữ liệu** từ một file CSV :

* Xử lý dữ liệu thiếu, không hợp lệ, hoặc không chuẩn hóa.
* Chuyển đổi và định dạng dữ liệu để đảm bảo tính nhất quán.
* Loại bỏ các giá trị trùng lặp, giữ dữ liệu gọn gàng và đáng tin cậy.

Đảm bảo dữ liệu sạch trước khi thực hiện các thao tác khác như lọc, sắp xếp, hoặc hiển thị.

**2.Thực hiện file clean**

Ở hàm delete\_emty() :

* Xóa các hàng và cột trống.
* Điền giá trị 0 cho các ô trống còn lại.

Ở dòng 4 : Xóa các hàng mà tất cả giá trị đều là null bằng hàm dropna(), trong đó how = ‘all’ là để xoá các hàng mà toàn bộ giá trị là null

Ở dòng 5 : Xóa các cột mà tất cả giá trị đều là null bằng dropna() với axis = 1 là thực hiện thao tác trên cột

Ở dòng 6 : Điền giá trị 0 vào các ô còn trống (null) với fillna( 0 )

Tác dụng:

* Loại bỏ dữ liệu thừa hoặc không hợp lệ.
* Đảm bảo các giá trị trống được thay thế bằng giá trị mặc định (0).

**A black background with colorful text

Description automatically generated**

Tiếp theo là hàm format\_number : Chuyển các cột số (float, int) trong DataFrame thành kiểu số nguyên (int) với hàm select\_dtypes cùng tham số inlcude là danh sách các cột có kiểu dữ liệu ‘float’, ‘int’ .

Dùng vòng lặp for để duyệt qua từng cột và chuyển đổi dữ liệu thành số nguyên bằng astype(int) để đảm bảo dữ liệu số không có phần thập phân, phù hợp với các phép tính yêu cầu số nguyên.

A black background with white text

Description automatically generated

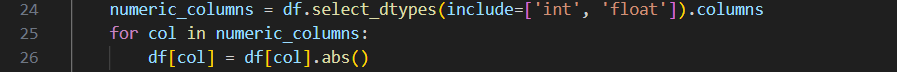
Cuối cùng là hàm cleanData : là hàm chính thực hiện làm sạch dữ liệu, kết hợp các hàm con và bổ sung thêm các bước xử lý.

Đầu tiên là đọc dữ liệu rồi xoá các phần dữ liệu bị thiếu bằng hàm delete\_empty, rồi chuyển dữ liệu số sang số nguyên bằng format\_number đã thực hiện trước đó.

A black background with green and yellow text

Description automatically generated

Tiếp theo, chuyển các giá trị nhỏ hơn 0 thành giá trị tuyệt đối. Dùng vòng lặp for để duyệt qua từng cột và thay thế giá trị nhỏ hơn 0 bằng trị tuyệt đối abs( ). Đảm bảo tất cả dữ liệu số không âm, phù hợp với dữ liệu dân số, số lượng.



Mục đích chính:

Loại bỏ tất cả ký tự đặc biệt đảm bảo dữ liệu trong các cột không chứa ký tự không mong muốn như !@#$%^&\*().

Duyệt qua từng cột (trừ 'Date'). Rồi chuyển tất cả các giá trị trong cột thành chuỗi (string) với df[column].astype(str). Cần chueyenr sang chuỗi string vì .str.replace() chỉ hoạt động trên dữ liệu kiểu chuỗi.

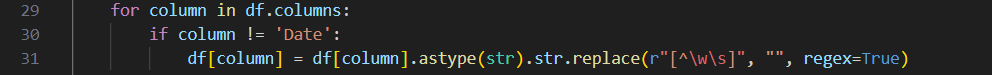
str.replace(): Là phương thức của Pandas để thay thế các ký tự hoặc chuỗi trong dữ liệu chuỗi. Ở đây ta dùng regex = True để dùng biểu thức chính quy để cho [^\w\s] ( đây là một nhóm ký tự ) :

* ^ bên trong ngoặc vuông: Là ký tự phủ định trong regex. Nó có nghĩa là "bất kỳ ký tự nào không thuộc nhóm sau đây".
* \w : Đại diện cho các ký tự chữ cái (a-z, A-Z), chữ số (0-9), và dấu gạch dưới \_
* \s: Đại diện cho các khoảng trắng (space, tab, xuống dòng).

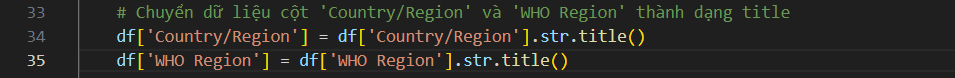
Vậy [^\w\s] có nghĩa là tìm bất kỳ ký tự nào không phải là chữ cái, số, gạch dưới, hoặc khoảng trắng.

Tác dụng:

Loại bỏ ký tự đặc biệt để đảm bảo tính nhất quán trong dữ liệu văn bản.



Tiếp theo, chuyển dữ liệu cột 'Country/Region' và 'WHO Region' thành dạng title bằng cách dùng hàm str.title() : Chuyển đổi dữ liệu chuỗi trong cột thành dạng Title Case ( chữ cái đầu của mỗi từ viết hoa, các chữ còn lại viết thường ). Việc này đảm bảo dữ liệu văn bản trong các cột 'Country/Region' và 'WHO Region' đồng nhất về mặt hiển thị.



Tiếp đến, đổi định dạng ngày thành DD/MM/YYYY, chuyển đổi cột 'Date' sang kiểu datetime. Trong đó, các tham số :

* format='mixed' cho phép xử lý các định dạng ngày tháng khác nhau trong cùng một cột.
* dayfirst=True: Pandas sẽ hiểu rằng ngày được viết trước tháng trong chuỗi thời gian
* errors='coerce': Các giá trị không thể chuyển đổi được thay bằng NaT (Not a Time).

Sau đó, định dạng lại ngày tháng bằng strftime('%d/%m/%Y') : Chuyển giá trị ngày tháng datetime thành chuỗi với định dạng DD/MM/YYYY

Từ đó giúp dữ liệu cột ‘Date’ có cùng kiểu dữ liệu datetime và cùng định dạng.



Cuối cùng, thực hiện xóa các hàng trùng lặp và giữ lại hàng đầu tiên. Ta dùng hàm df.drop\_duplicates() : Loại bỏ các hàng trùng lặp trong DataFrame, với tham số keep='first' để có thể giữ lại hàng đầu tiên trong các nhóm hàng trùng lặp.



3. Tóm gọn

Loại bỏ giá trị trống (null):

Chuẩn hóa dữ liệu số:

Làm sạch văn bản:

Định dạng ngày tháng:

Loại bỏ trùng lặp: