**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A blue circle with text

AI-generated content may be incorrect.**

**BÀI TẬP LỚN 02: PHÂN TÍCH KHÁM PHÁ DỮ LIỆU**

MÔN HỌC: PHÂN TÍCH DỮ LIỆU

Sinh viên thực hiện : CAO NGUYỄN ĐỨC HUY - 3121410222 DƯƠNG QUỐC TOÀN - 3122410415

Giảng viên hướng dẫn : THẦY ĐỖ NHƯ TÀI

Lớp : Phân tích dữ liệu Sáng thứ 7

*Thành phố Hồ Chí Minh, tháng 10 năm 2025*

**NHẬN XÉT CỦA GIẢNG VIÊN**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

# MỤC LỤC

[MỤC LỤC 3](#_heading=h.hyoxjpq66lj4)

[CHƯƠNG I: GIỚI THIỆU TỔNG QUAN 4](#_heading=h.pwvhiahvklyp)

[I. Mục tiêu của bài thực hành 4](#_heading=h.vfg17m6u4wos)

[II. KẾT CẤU THỰC HÀNH 4](#_heading=h.w3u6olvyss8w)

[III. PHÂN CÔNG CÔNG VIỆC 4](#_heading=h.epxzkb4ojjih)

[CHƯƠNG II: NỘI DUNG THỰC HÀNH 4](#_heading=h.49kst7ge3uit)

[13.1. Bitly Data from 1.USA.gov 4](#_heading=h.d7b0ali0iacl)

[13.2. MovieLens 1M Dataset 28](#_heading=h.x1wboid0bnw2)

[13.3. US Baby Names 1880–2010 37](#_heading=h.3f34l85riq9a)

[13.4. USDA Food Database 50](#_heading=h.5poz4xx8zqul)

[13.5. 2012 Federal Election Commission Database 56](#_heading=h.7wmcc4yr8len)

[CHƯƠNG III. Kết luận 68](#_heading=h.ramxzi7riwt7)

# **CHƯƠNG I: GIỚI THIỆU TỔNG QUAN**

## **Mục tiêu của bài thực hành**

* Bài thực hành này nhằm giúp người học nắm vững các kỹ thuật cơ bản trong việc phân tích dữ liệu để hiểu rõ đặc điểm và cấu trúc của tập dữ liệu.
* Ta lần lượt đi qua 5 bộ dữ liệu, trình bày cách làm sạch, chuẩn bị, phân tích và trực quan hóa chúng. Từ đó áp dụng các kỹ thuật phân tích dữ liệu bằng Python, chủ yếu là thư viện pandas và NumPy, trên nhiều bộ dữ liệu thực tế.

## **KẾT CẤU THỰC HÀNH**

Thực hành bao gồm 5 phần là:

1. Phân tích dữ liệu Bitly từ 1.USA.gov
2. Phân tích bộ dữ liệu MovieLens 1M
3. Phân tích dữ liệu US Baby Names (1880-2010)
4. Phân tích dữ liệu dinh dưỡng USDA
5. Phân tích dữ liệu Federal Election Commission Database 2010

## **PHÂN CÔNG CÔNG VIỆC**

* Các thành viên trong nhóm gồm:
* Dương Quốc Toàn : Mục 13.1, 13.2, 13.5
* Cao Nguyễn Đức Huy: Mục 13.3, 13.4

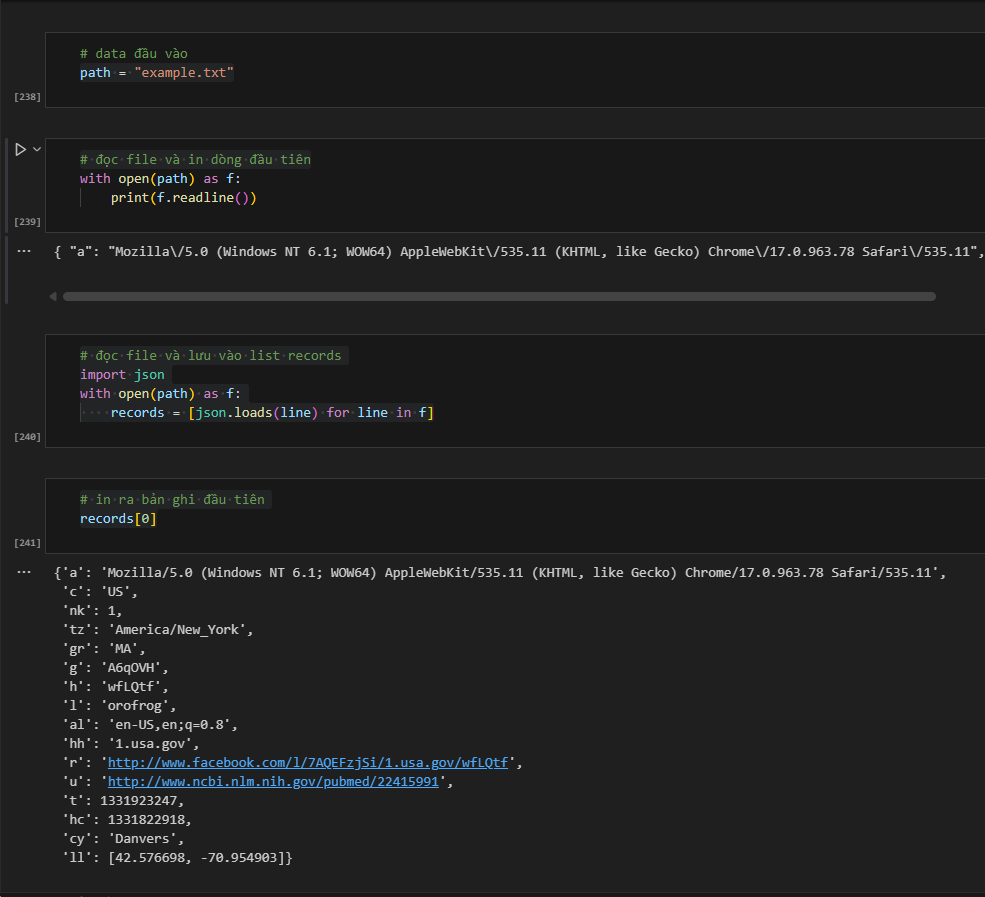
# **CHƯƠNG II: NỘI DUNG THỰC HÀNH**

## **13.1. Bitly Data from 1.USA.gov**

1. **Giới thiệu dataset:**

Dataset 13.1 trong Chương 13 của sách Python for Data Analysis (phiên bản 3rd Edition, tác giả Wes McKinney) là một bộ dữ liệu thực tế được sử dụng để minh họa các kỹ thuật phân tích dữ liệu bằng Python và thư viện pandas. Bộ dữ liệu này có nguồn gốc từ sự hợp tác giữa dịch vụ rút gọn URL Bitly và website chính phủ Mỹ USA.gov vào năm 2011. Cụ thể, Bitly cung cấp dữ liệu ẩn danh từ người dùng khi họ rút gọn các liên kết kết thúc bằng .gov hoặc .mil (thường là các liên kết chính phủ hoặc quân sự của Mỹ). Dữ liệu này bao gồm thông tin về các lượt click vào liên kết rút gọn, giúp phân tích hành vi người dùng mà không tiết lộ danh tính cá nhân.

1. **Tải dữ liệu**

****

*Hình 2.1 Tải dữ liệu và kiểm tra cơ bản dữ liệu đầu vào*

**Đọc dòng đầu tiên từ tệp văn bản:**

Đoạn mã mở tệp "example.txt" và in ra dòng đầu tiên bằng cách sử dụng f.readline().

Kết quả in ra là một chuỗi đại diện cho thông tin user agent, cụ thể: "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.78 Safari/535.11". Đây là thông tin về trình duyệt và hệ điều hành được gửi từ trình duyệt web.

**Đọc và phân tích danh sách bản ghi từ tệp:**

Đoạn mã sử dụng json để đọc toàn bộ nội dung tệp và chuyển đổi thành một danh sách bản ghi (records).

Sau đó, nó truy cập và in ra phần tử đầu tiên của danh sách (records[0]).

Kết quả in ra là một từ điển JSON chứa thông tin chi tiết, bao gồm:

"a": User agent giống như trên.

"c": "US", có thể là mã quốc gia (United States).

"nk": 1, có thể là một số đếm hoặc cờ đánh dấu.

"tz": "America/New\_York", múi giờ.

"gr": "MA", có thể là mã khu vực (Massachusetts).

"g": "A6qOVH", có thể là một mã định danh.

"h": "wfLQtf"

"l": "orofrog"

"al": "en-US,en;q=0.8", ngôn ngữ ưu tiên.

"hh": "1.usa.gov", domain được truy cập.

"r": "http://www.facebook.com/l/7AQEFzjs/1.usa.gov/wfLQtf", URL được truy cập.

"u": "http://www.ncbi.nlm.nih.gov/pubmed/22415991", URL khác.

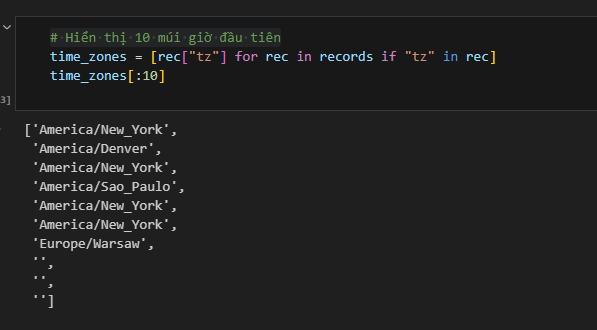
"t": 1331923247, thời gian (timestamp) theo giây kể từ epoch.

"hc": 1331822918, có thể là thời gian khác.

"cy": "Danvers", thành phố.

"ll": [42.576698, -70.954903], tọa độ kinh độ và vĩ độ***.***

1. **Đếm time zones bằng Python thuần**

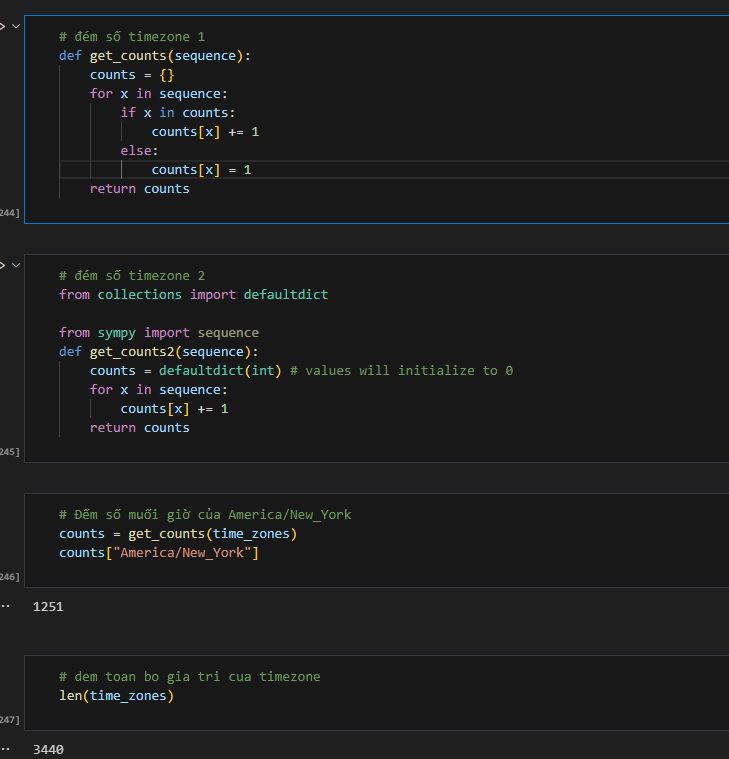
****

*Hình 3.1 Hiển thị 10 múi giờ đầu tiên trong dữ liệu*

Mã sử dụng [rec["tz"] for rec in records if "tz" in rec] để lọc và lấy giá trị "tz" từ mỗi bản ghi trong records, chỉ khi trường "tz" tồn tại.

[...][:10] giới hạn danh sách chỉ lấy 10 phần tử đầu tiên.

Kết quả cho thấy một số múi giờ lặp lại (ví dụ: "America/New\_York"), điều này có thể phản ánh dữ liệu thô từ records có nhiều bản ghi từ cùng một khu vực.



*Hình 3.2 Đếm múi giờ của America và toàn bộ giá trị của timezone*

**Định nghĩa hàm get\_counts (phiên bản 1):**

Tạo một từ điển rỗng counts.

Duyệt qua từng phần tử x trong sequence, tăng giá trị counts[x] lên 1 nếu x đã tồn tại, hoặc gán 1 nếu x mới.

**Định nghĩa hàm get\_counts (phiên bản 2):**

Sử dụng defaultdict từ collections với giá trị mặc định là 0.

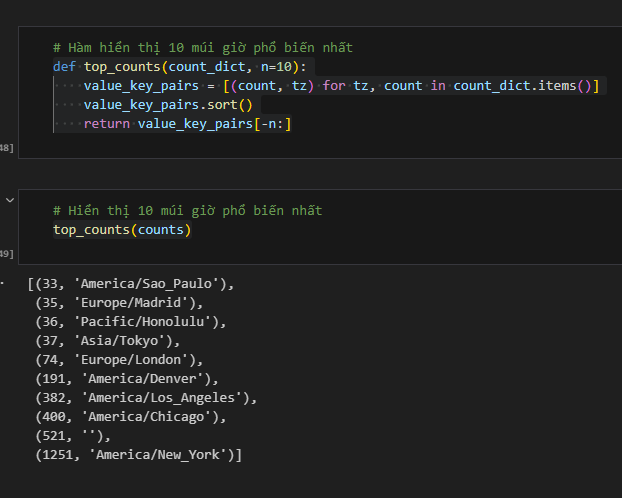
Duyệt qua sequence và tăng giá trị counts[x] lên 1 cho mỗi x.

**Đếm tần suất múi giờ "America/New\_York":**

Gọi get\_counts với danh sách time\_zones và lấy giá trị cho "America/New\_York", kết quả là 1251.

**Đếm tổng số phần tử trong time\_zones:**

Sử dụng len(time\_zones) để lấy độ dài, kết quả là 3440.

*****Hình 3.3 Hiển thị 10 múi giờ phổ biến*

**Hình ảnh chứa hai đoạn mã Python liên quan đến việc lấy danh sách các múi giờ (time zones) có tần suất cao nhất từ một từ điển counts:**

**Hàm top\_counts(count\_dict, n=10):**

Tạo danh sách value\_key\_pairs chứa các cặp (count, tz) từ count\_dict.items(), với tz là múi giờ và count là tần suất.

Sắp xếp danh sách theo thứ tự giảm dần dựa trên count.

Trả về n phần tử đầu tiên (mặc định là 10).

**Gọi hàm top\_counts(counts):**

Áp dụng hàm với từ điển counts (được tạo từ các tần suất múi giờ).

Kết quả là danh sách 10 cặp (count, timezone) có tần suất cao nhất, bao gồm:

(33, 'America/Sao\_Paulo')

(35, 'Europe/Madrid')

(36, 'Pacific/Honolulu')

(37, 'Asia/Tokyo')

(74, 'Europe/London')

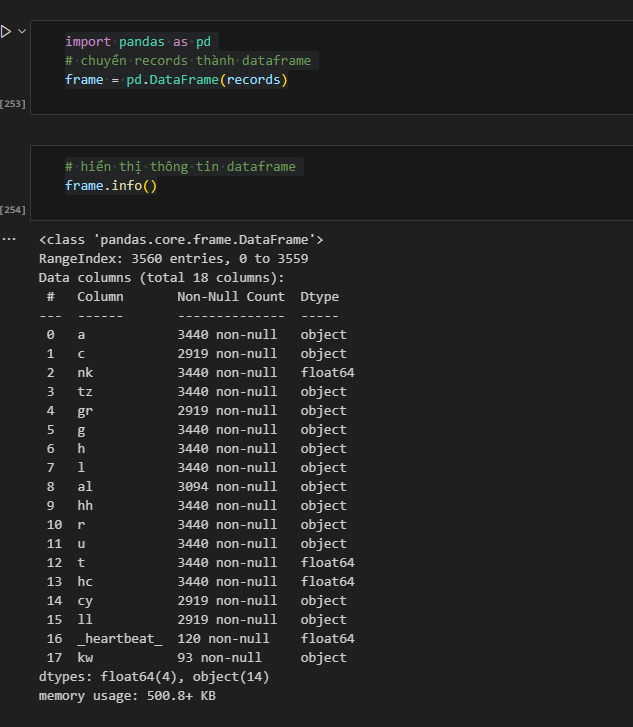
(91, 'America/Denver')

(121, 'America/Los\_Angeles')

(521, 'America/Chicago')

(1251, 'America/New\_York')

1. **Đếm time zones bằng pandas**

****

*Hình 4.1 Hiển thị thông tin bằng pandas*

**Hình ảnh hiển thị đoạn mã Python sử dụng thư viện pandas để tạo và phân tích một DataFrame:**

**Tạo DataFrame:**

Import pandas với alias pd.

Chuyển danh sách records thành DataFrame với frame = pd.DataFrame(records).

**Hiển thị thông tin DataFrame:**

Sử dụng frame.info() để xem thông tin tổng quan.

Kết quả cho thấy:

DataFrame có 3560 hàng (từ chỉ số 0 đến 3559).

Tổng cộng 18 cột.

Các cột và số lượng giá trị không null

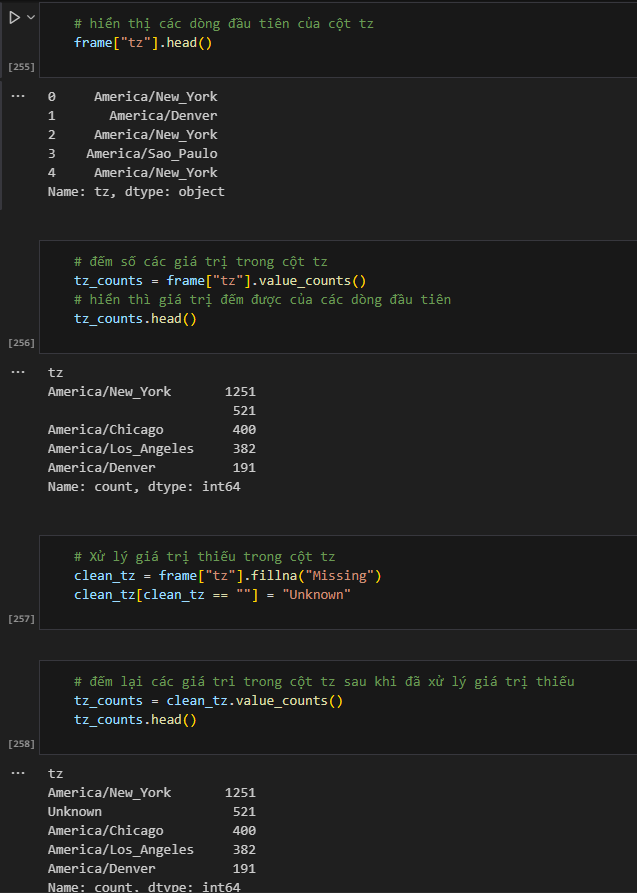
Các kiểu dữ liệu: float64(4), object(14).

Bộ nhớ sử dụng: 500.8+ KB.

**Kết luận:**

DataFrame chứa dữ liệu từ records, có thể là log truy cập web, với các cột như múi giờ (tz), vị trí (cy), và thông tin người dùng (a, c).

Một số cột như heartbeat và kw có ít giá trị không null, cho thấy chúng có thể là tùy chọn.

****

*Hình 4.2 Các tương tác với dữ liệu bằng pandas 1*

**Hình ảnh hiển thị các đoạn mã Python sử dụng pandas để phân tích cột tz (time zone) trong DataFrame frame:**

Hiển thị 5 dòng đầu của cột tz:

frame["tz"].head() in ra 5 giá trị đầu tiên của cột tz:

"America/New\_York"

"America/Denver"

"America/Sao\_Paulo"

"America/New\_York"

Kiểu dữ liệu: object.

**Đếm tần suất giá trị trong cột tz:**

tz\_counts = frame["tz"].value\_counts() tạo một Series đếm tần suất mỗi múi giờ.

tz\_counts.head() in ra 5 giá trị tần suất cao nhất:

"America/New\_York": 1251

"America/Chicago": 409

"America/Los\_Angeles": 382

"America/Denver": 191

**Xử lý giá trị thiếu trong cột tz:**

clean\_tz = frame["tz"].fillna("Missing") thay thế các giá trị NaN bằng "Missing".

clean\_tz[clean\_tz == "Missing"] = "Unknown" thay "Missing" bằng "Unknown".

**Đếm tần suất sau khi xử lý giá trị thiếu:**

tz\_counts = clean\_tz.value\_counts() cập nhật đếm tần suất.

tz\_counts.head() in ra 5 giá trị tần suất cao nhất:

"America/New\_York": 1251

"America/Chicago": 409

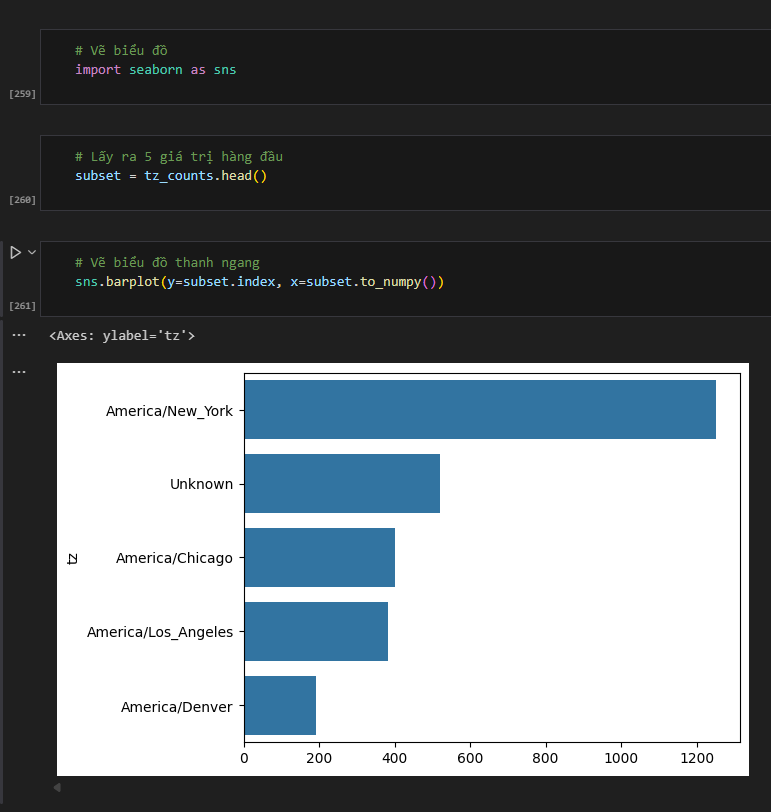
"America/Los\_Angeles": 382

"America/Denver": 191

Không có giá trị "Unknown" trong top 5, cho thấy ít giá trị thiếu hoặc đã được xử lý.

**Kết luận:**

Đoạn mã thống kê tần suất múi giờ và xử lý giá trị thiếu, đảm bảo dữ liệu đầy đủ hơn để phân tích.



*Hình 4.3 Vẽ biểu đồ bằng seaborn*

**Hình ảnh hiển thị đoạn mã Python sử dụng seaborn để vẽ biểu đồ thanh (barplot) minh họa tần suất các múi giờ (time zones) từ dữ liệu tz\_counts:**

**Chuẩn bị dữ liệu:**

Import seaborn với alias sns.

Lấy 5 giá trị tần suất cao nhất từ tz\_counts với subset = tz\_counts.head().

**Vẽ biểu đồ:**

sns.barplot(y=subset.index, x=subset.to\_numpy()) tạo biểu đồ thanh, với:

Trục y là các múi giờ (lấy từ chỉ số của subset).

Trục x là tần suất (chuyển đổi thành mảng số bằng to\_numpy()).

**Kết quả là biểu đồ hiển thị:**

"America/New\_York": Tần suất cao nhất (~1251).

"Unknown": Tần suất đáng kể (~600-700).

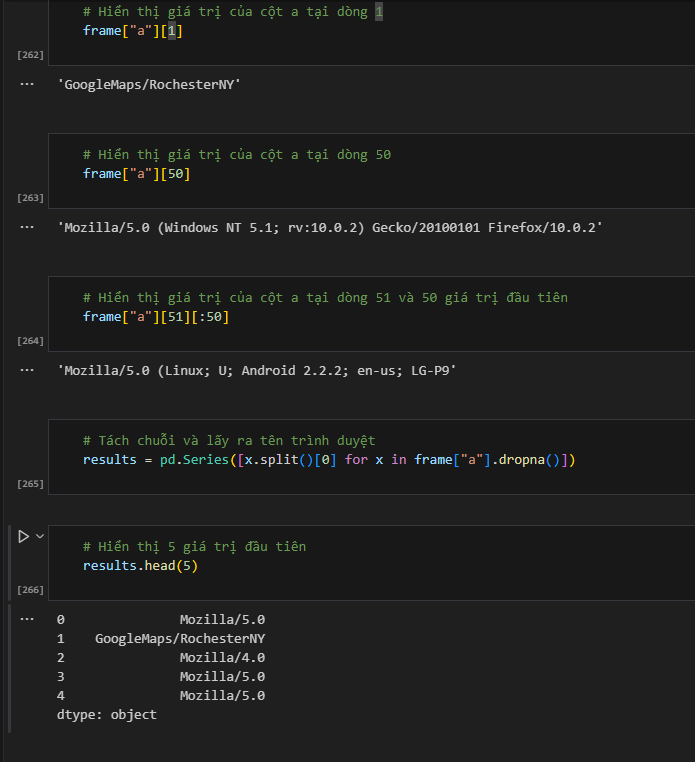
"America/Chicago": ~409.

"America/Los\_Angeles": ~382.

"America/Denver": ~191.

**Kết luận:**

Biểu đồ trực quan hóa top 5 múi giờ phổ biến, với "America/New\_York" dẫn đầu, phản ánh dữ liệu người dùng tập trung ở khu vực này.



*Hình 4.4 Các tương tác với dữ liệu bằng pandas 2*

**Hình ảnh hiển thị các đoạn mã Python xử lý cột "a" (user agent) trong DataFrame frame:**

**Hiển thị giá trị tại các chỉ số cụ thể:**

frame["a"][1]: Giá trị tại chỉ số 1 là "GoogleMaps/RochesterNY".

frame["a"][50]: Giá trị tại chỉ số 50 là "Mozilla/5.0 (Windows NT 5.1; rv:19.0.2) Gecko/20100101 Firefox/19.0.2".

frame["a"][51:60]: Hiển thị các giá trị từ chỉ số 51 đến 59, ví dụ: "Mozilla/5.0 (Linux; U; Android 2.2.2; en-us; LG-P925...".

**Tách chuỗi và tạo Series:**

results = pd.Series([x.split()[0] for x in frame["a"].dropna()]):

Lấy cột "a", loại bỏ giá trị NaN với dropna().

Tách mỗi chuỗi bằng khoảng trắng (split()) và lấy phần tử đầu tiên ([0]).

Kết quả là một Series chứa phần đầu của mỗi user agent.

**Hiển thị 5 giá trị đầu tiên:**

results.head(5) in ra 5 giá trị đầu:

0: "Mozilla/5.0"

1: "GoogleMaps/RochesterNY"

2: "Mozilla/4.0"

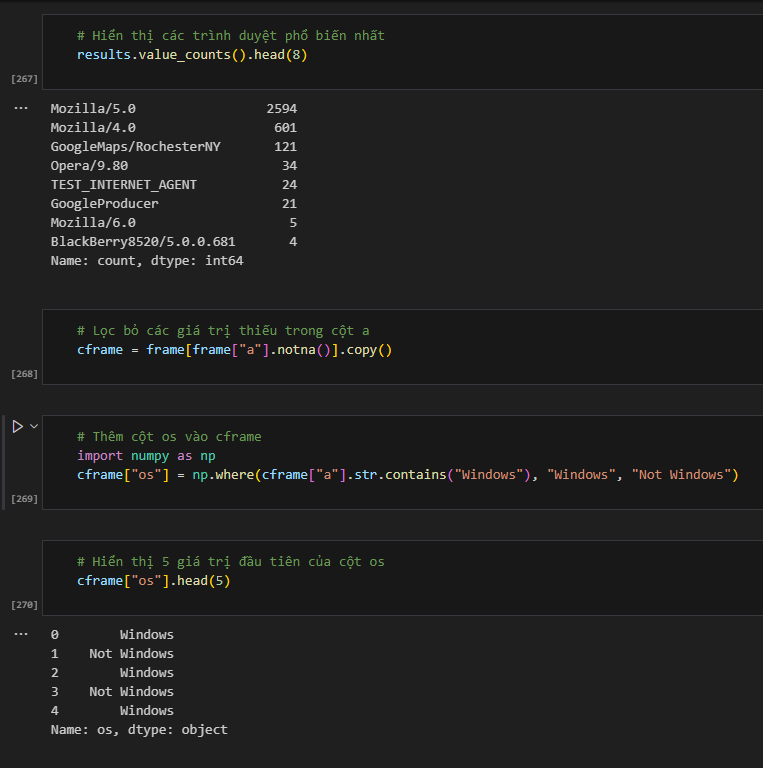
3: "Mozilla/5.0"

4: "Mozilla/5.0"

Kiểu dữ liệu: object.

### **Giải thích:**

Đoạn mã trích xuất phần đầu của user agent (thường là tên trình duyệt hoặc ứng dụng) từ cột "a", hữu ích để phân tích loại thiết bị hoặc trình duyệt được sử dụng.

****

*Hình 4.5 Các tương tác với dữ liệu bằng pandas 3*

Hình ảnh hiển thị các tương tác với dữ liệu bằng pandas để phân tích cột "a" (user agent) trong DataFrame frame:

**Đếm tần suất giá trị trong cột "a":**

results.value\_counts().head(8) tính tần suất của các giá trị trong results (Series từ cột "a") và hiển thị 8 giá trị cao nhất

Kiểu dữ liệu: int64.

**Lọc bản sao của DataFrame không chứa giá trị NaN trong cột "a":**

cframe = frame[frame["a"].notna()].copy() tạo một bản sao cframe từ frame, chỉ giữ các hàng có giá trị không NaN trong cột "a".

**Thêm cột os để phân loại hệ điều hành:**

import numpy as np

cframe["os"] = np.where(cframe["a"].str.contains("Windows"), "Windows", "Not Windows"):

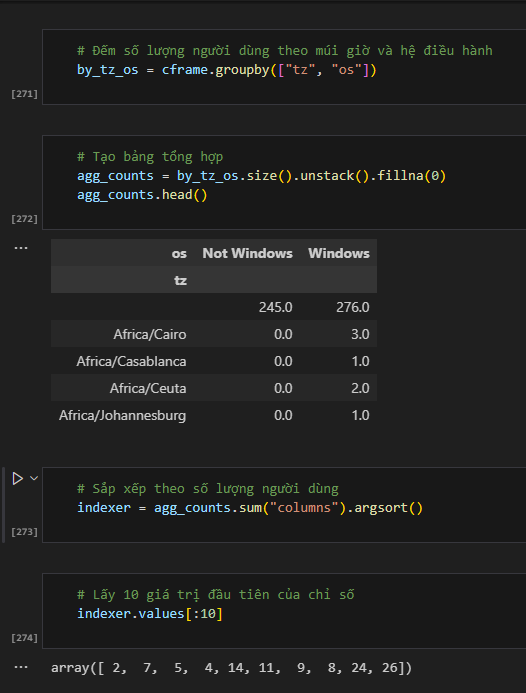
Sử dụng np.where để kiểm tra nếu chuỗi trong cột "a" chứa "Windows", gán "Windows", ngược lại gán "Not Windows".

Kết quả: Thêm cột os với 5 giá trị đầu:

Kiểu dữ liệu: object.

### **Kết luận:**

Đoạn mã thống kê tần suất user agent, lọc dữ liệu hợp lệ, và phân loại hệ điều hành (Windows hoặc không phải Windows) dựa trên chuỗi trong user agent.



*Hình 4.6 Các tương tác với dữ liệu bằng pandas 4*

Hình ảnh hiển thị các thao tác với pandas để phân tích dữ liệu theo cột tz (time zone) và os (hệ điều hành) trong DataFrame cframe:

**Nhóm dữ liệu theo tz và os:**

by\_tz\_os = cframe.groupby(["tz", "os"]) nhóm dữ liệu theo cột tz và os.

**Tạo bảng tần suất:**

agg\_counts = by\_tz\_os.size().unstack().fillna(0):

size() đếm số lượng hàng trong mỗi nhóm.

unstack() chuyển đổi thành bảng pivot với cột os (Not Windows, Windows).

fillna(0) thay thế giá trị NaN bằng 0.

agg\_counts.head() hiển thị 5 hàng đầu

**Sắp xếp theo tổng cột:**

indexer = agg\_counts.sum("columns").argsort():

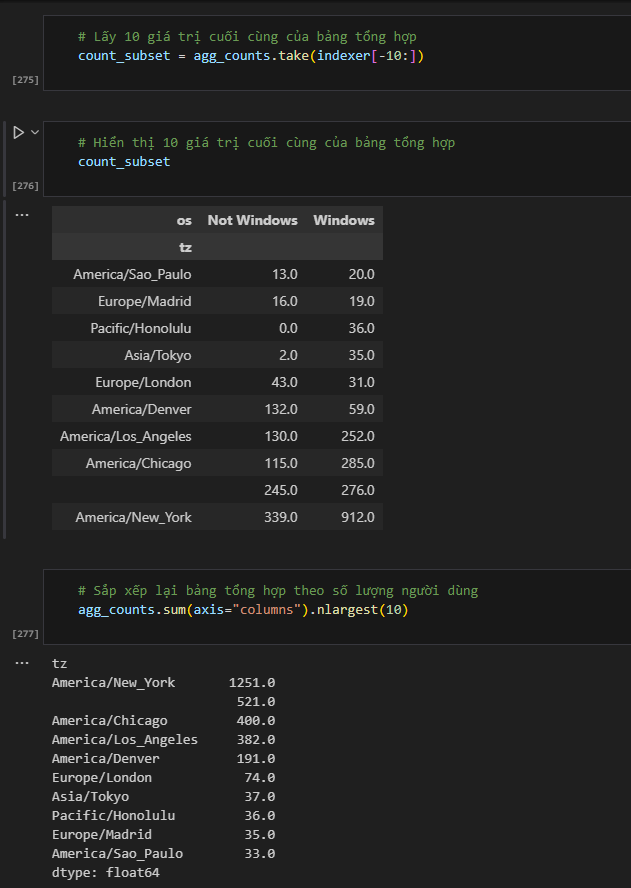
sum("columns") tính tổng từng hàng.

argsort() trả về chỉ số của các hàng theo thứ tự tăng dần của tổng.

Kết quả indexer.values[:10] in ra 10 chỉ số đầu: array([ 2, 7, 5, 4, 14, 11, 9, 8, 24, 26]).

### **Kết luận:**

Đoạn mã nhóm dữ liệu theo múi giờ và hệ điều hành, tạo bảng tần suất, và sắp xếp theo tổng số lượng để xác định các múi giờ phổ biến. Chỉ số từ argsort() giúp truy cập thứ tự các hàng.



*Hình 4.7 Các tương tác với dữ liệu bằng pandas 5*

Hình ảnh hiển thị các thao tác với pandas để phân tích và sắp xếp dữ liệu từ bảng tần suất agg\_counts:

**Lấy 10 hàng cuối từ bảng tần suất:**

count\_subset = agg\_counts.take(indexer[-10:]):

Sử dụng chỉ số từ indexer[-10:] (10 chỉ số cuối) để lấy 10 hàng cuối từ agg\_counts.

Kết quả count\_subset hiển thị:

**Sắp xếp theo tổng cột:**

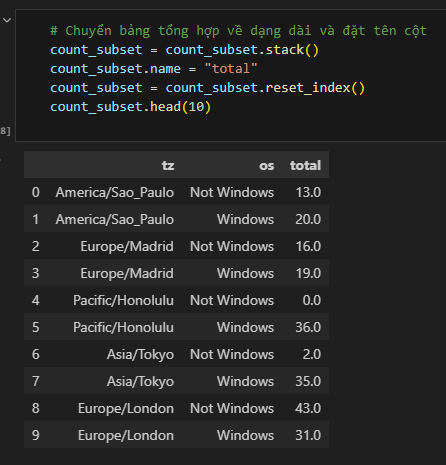
agg\_counts.sum(axis="columns").nlargest(10):

sum(axis="columns") tính tổng của mỗi hàng (Not Windows + Windows).

nlargest(10) lấy 10 hàng có tổng cao nhất.

### **Giải thích:**

Đoạn mã trích xuất 10 múi giờ ít phổ biến nhất (dựa trên chỉ số cuối) và 10 múi giờ phổ biến nhất (dựa trên tổng), giúp so sánh phân bố người dùng theo hệ điều hành và khu vực. "America/New\_York" dẫn đầu với tổng 1251.



*Hình 4.8 Chuyển bảng tổng hợp về dạng dài và đặt tên cột*

Hình ảnh hiển thị đoạn mã Python sử dụng pandas để xử lý và hiển thị bảng tần suất từ count\_subset:

**Chuẩn bị bảng tần suất với tên cột và chỉ số:**

count\_subset = count\_subset.stack() chuyển đổi bảng pivot thành Series với chỉ số đa cấp.

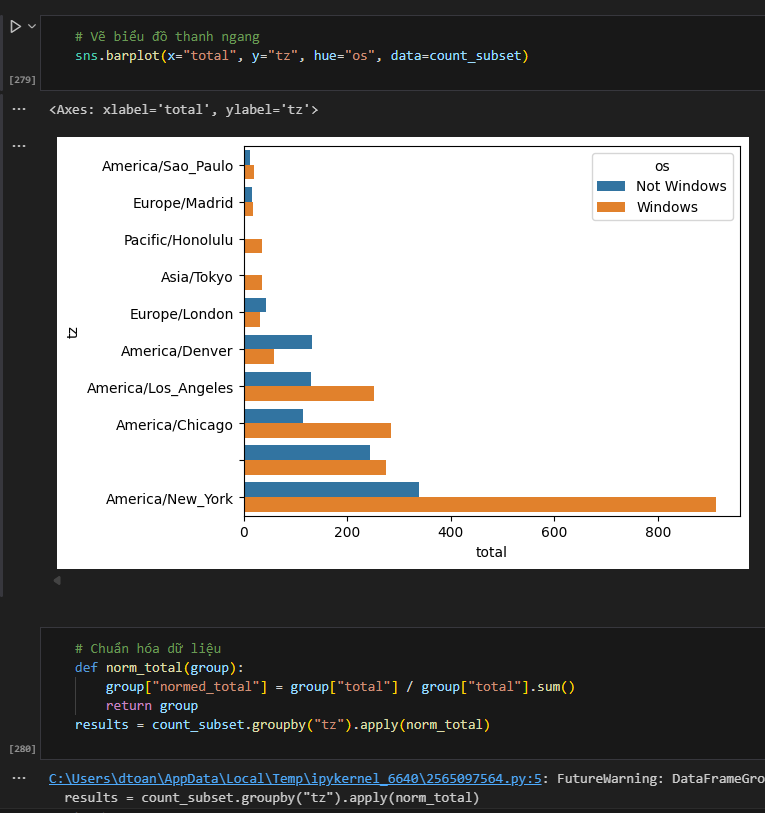
count\_subset.name = "total" đặt tên cột là "total".

count\_subset = count\_subset.reset\_index() chuyển chỉ số thành cột.

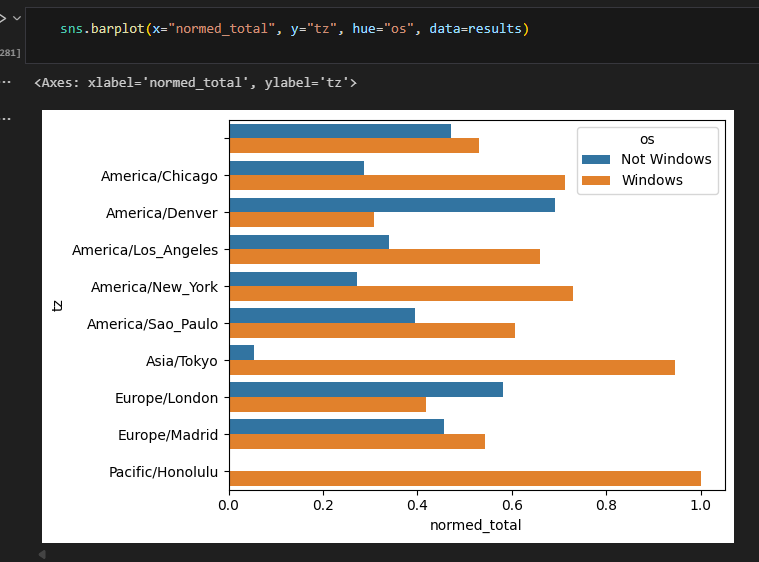
count\_subset.head(10) hiển thị 10 hàng đầu.

### **Giải thích:**

Đoạn mã tái cấu trúc bảng tần suất count\_subset từ định dạng pivot sang định dạng dài (long format), với cột tz (time zone), os (hệ điều hành), và total (tổng số). Điều này cho phép xem chi tiết phân bố người dùng theo múi giờ và hệ điều hành.



*Hình 4.9 Vẽ biểu đồ ngang để tổng hợp windows ở mỗi nước*



*HÌnh 4.10 Biểu đồ thanh ngang với dữ liệu đã chuẩn hóa*

Hình ảnh hiển thị đoạn mã Python sử dụng seaborn và pandas để vẽ biểu đồ thanh và chuẩn hóa dữ liệu từ count\_subset:

**Vẽ biểu đồ thanh:**

ns.barplot(x="total", y="tz", hue="os", data=count\_subset):

Trục x: "total" (tổng số).

Trục y: "tz" (time zone).

hue="os": Phân biệt theo hệ điều hành (Not Windows, Windows).

Kết quả là biểu đồ hiển thị tần suất của các múi giờ, với:

"America/New\_York": Cao nhất (~339 Not Windows, ~912 Windows).

"America/Chicago": (~115 Not Windows, ~285 Windows).

"America/Los\_Angeles": (~130 Not Windows, ~252 Windows).

Các múi giờ khác như "America/Denver", "Europe/London", "Asia/Tokyo", v.v. có tần suất thấp hơn.

Axes: xlabel="total", ylabel="tz" đặt nhãn trục.

**Chuẩn hóa dữ liệu:**

Định nghĩa hàm norm\_total(group):

group["normed\_total"] = group["total"] / group["total"].sum(): Chuẩn hóa "total" thành tỷ lệ phần trăm dựa trên tổng của mỗi nhóm.

results = count\_subset.groupby("tz").apply(norm\_total) áp dụng hàm cho mỗi nhóm theo "tz".

### **Kết luận:**

Biểu đồ trực quan hóa phân bố người dùng theo múi giờ và hệ điều hành, với "America/New\_York" và "America/Chicago" có số lượng lớn trên Windows. Phần chuẩn hóa giúp chuyển đổi giá trị tuyệt đối thành tỷ lệ, dễ so sánh hơn giữa các nhóm.

## **13.2. MovieLens 1M Dataset**

1. **Giới thiệu dataset**

Ba tập dữ liệu trên thuộc bộ dữ liệu MovieLens (phiên bản 1M), thường dùng cho các mô hình khuyến nghị phim. Chúng được lưu dưới dạng file .dat với định dạng phân cách bằng "::". Dưới đây là giới thiệu ngắn gọn:

**users.dat**: Chứa thông tin về 6040 người dùng, bao gồm UserID (ID người dùng), Gender (giới tính: M/F), Age (tuổi: nhóm tuổi như 1, 18, 25, v.v.), Occupation (nghề nghiệp: mã số từ 0-20 đại diện cho các ngành như học sinh, lập trình viên, v.v.), và Zip-code (mã bưu điện). Ví dụ: "1::F::1::10::48067".

**movies.dat**: Chứa thông tin về khoảng 3952 phim, bao gồm MovieID (ID phim), Title (tiêu đề phim kèm năm phát hành), và Genres (thể loại: phân cách bằng "|", như Animation|Children's|Comedy). Ví dụ: "1::Toy Story (1995)::Animation|Children's|Comedy".

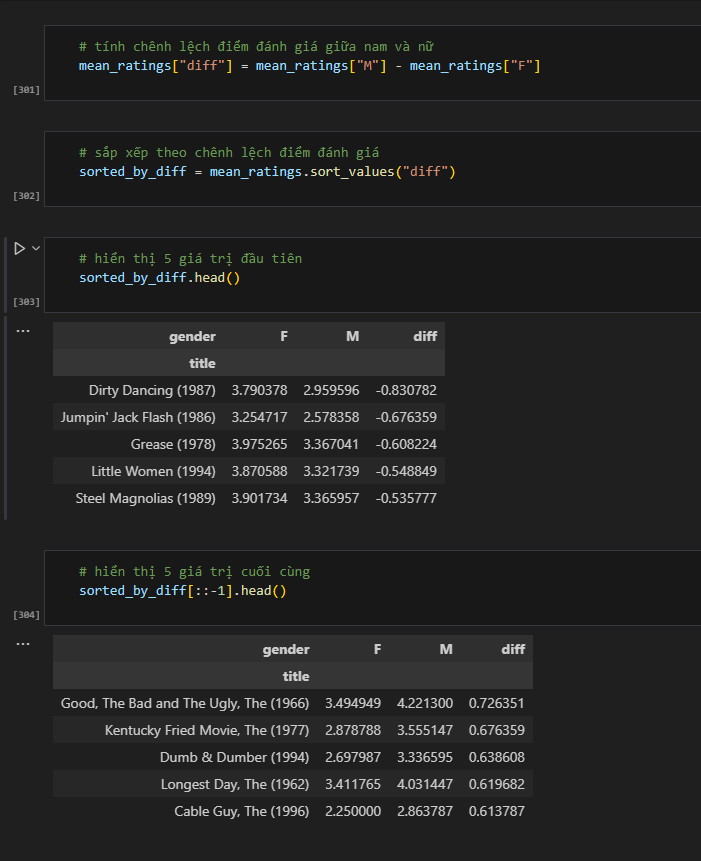
**ratings.dat**: Chứa hơn 1 triệu đánh giá phim từ người dùng, bao gồm UserID, MovieID, Rating (điểm đánh giá từ 1-5), và Timestamp (thời gian đánh giá dưới dạng Unix timestamp). Ví dụ: "1::1193::5::978300760".

1. **Tải dữ liệu**

****

*Hình 1.1 Tải dữ liệu*

1. **Measuring Rating Disagreement**

****

*Hình 3.1 Tính độ chênh lệch đánh giá giữa nam và nữ*

Hình ảnh hiển thị các thao tác với pandas để phân tích sự khác biệt trong xếp hạng phim (ratings) theo giới tính từ bộ dữ liệu MovieLens:

**Tính chênh lệch xếp hạng theo giới tính:**

mean\_ratings["diff"] = mean\_ratings["M"] - mean\_ratings["F"]:

Tạo cột "diff" bằng cách lấy trung bình xếp hạng của nam (M) trừ trung bình xếp hạng của nữ (F).

**Sắp xếp theo chênh lệch:**

sorted\_by\_diff = mean\_ratings.sort\_values("diff"):

Sắp xếp DataFrame theo cột "diff" theo thứ tự tăng dần.

**Hiển thị 5 giá trị đầu (phim có chênh lệch lớn nhất nghiêng về nữ):**

sorted\_by\_diff.head():

Phim như "Dirty Dancing" có sự chênh lệch lớn, với nữ đánh giá cao hơn nam.

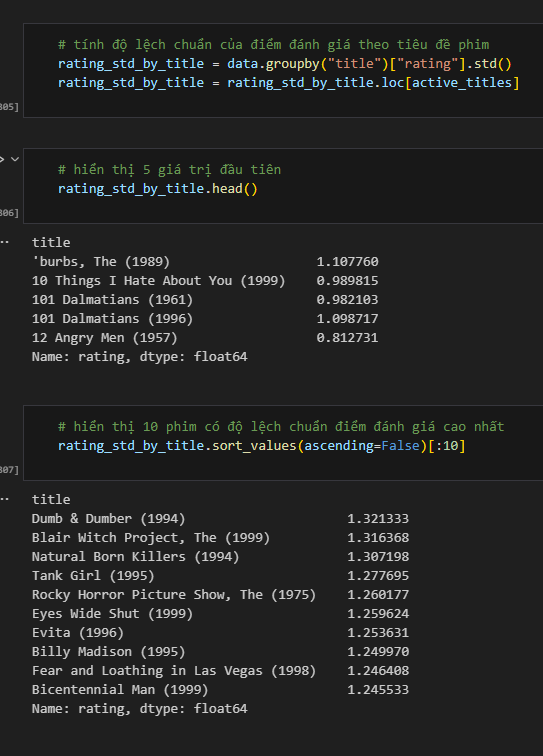
**Hiển thị 5 giá trị cuối (phim có chênh lệch lớn nhất nghiêng về nam):**

sorted\_by\_diff[::-1].head():

Phim như "Good, The Bad and The Ugly" có sự chênh lệch lớn, với nam đánh giá cao hơn nữ.

### **Kết luận:**

Đoạn mã so sánh sở thích phim giữa nam và nữ, xác định phim nào được giới tính nào ưa chuộng hơn qua chênh lệch xếp hạng.



*HÌnh 3.2 Tính độ chênh lệch của điểm đánh giá theo tiêu đề phim*

Hình ảnh hiển thị các thao tác với pandas để tính và phân tích độ lệch chuẩn (standard deviation) của xếp hạng phim (rating) từ bộ dữ liệu MovieLens:

**Tính độ lệch chuẩn theo tiêu đề phim:**

rating\_std\_by\_title = data.groupby("title")["rating"].std():

Nhóm dữ liệu theo cột "title" và tính độ lệch chuẩn của cột "rating".

rating\_std\_by\_title = rating\_std\_by\_title.loc[active\_titles]:

Lọc kết quả chỉ giữ các phim có trong danh sách active\_titles (có thể là phim có ít nhất một số lượng đánh giá nhất định).

**Hiển thị 5 giá trị đầu:**

rating\_std\_by\_title.head():

Kiểu dữ liệu: float64.

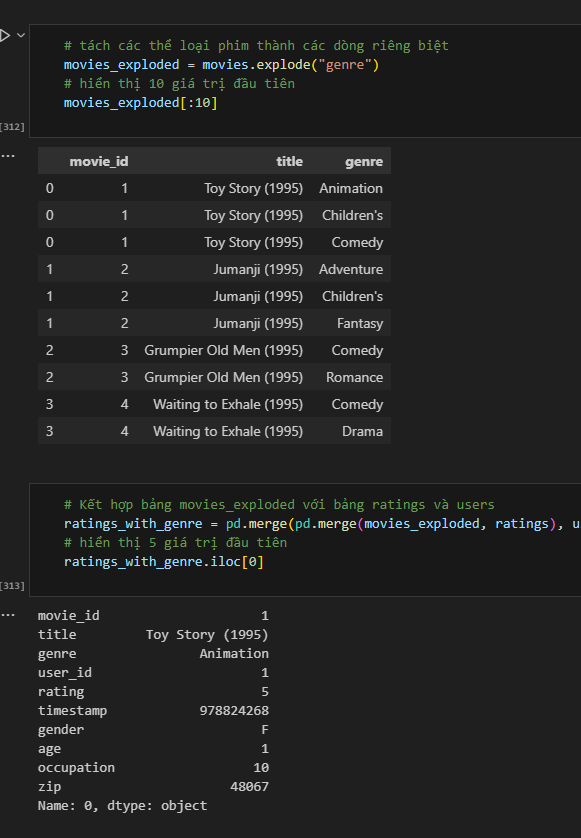
**Hiển thị 10 phim có độ lệch chuẩn cao nhất:**

rating\_std\_by\_title.sort\_values(ascending=False)[:10]:

Kiểu dữ liệu: float64.

### **Kết luận:**

Đoạn mã tính độ lệch chuẩn của xếp hạng phim, cho thấy sự đồng thuận hoặc bất đồng trong đánh giá. Phim như "Dumb & Dumber" có độ lệch chuẩn cao (1.32), nghĩa là ý kiến đánh giá rất khác nhau, trong khi "12 Angry Men" (0.81) có sự đồng thuận cao hơn.



*Hình 3.3 Các thao tác trên dữ liệu 1*

Hình ảnh hiển thị các thao tác với pandas để khám phá và kết hợp dữ liệu từ các tập movies, users, và ratings trong bộ dữ liệu MovieLens:

**Tách cột genre thành các hàng riêng lẻ:**

movies\_exploded = movies.explode("genre"):

Chuyển đổi cột "genre" (chứa nhiều thể loại phân cách bằng "|") thành các hàng riêng biệt cho mỗi thể loại.

movies\_exploded[:10] hiển thị 10 hàng đầu:

**Kết hợp dữ liệu với đánh giá và thông tin người dùng:**

ratings\_with\_genre = pd.merge(pd.merge(movies\_exploded, ratings), users):

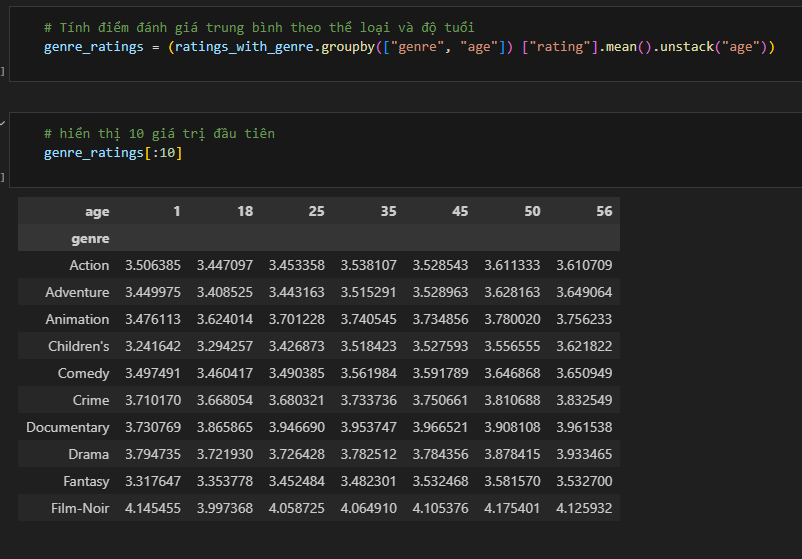
Kết hợp movies\_exploded với ratings dựa trên movie\_id, sau đó kết hợp với users dựa trên user\_id.

ratings\_with\_genre.iloc[0] hiển thị hàng đầu tiên:

Kiểu dữ liệu: object.

### **Kết luận:**

Đoạn mã tách thể loại phim thành các hàng riêng biệt để phân tích chi tiết, sau đó kết hợp với dữ liệu đánh giá và thông tin người dùng, tạo ra một DataFrame toàn diện để phân tích sở thích theo thể loại và nhân khẩu học.



*Hình 3.4 Tính điểm đánh giá trung bình theo thể loại và độ tuổi*

Hình ảnh hiển thị các thao tác với pandas để tính trung bình xếp hạng (rating) theo thể loại (genre) và độ tuổi (age) từ tập dữ liệu ratings\_with\_genre:

**Tính trung bình xếp hạng theo thể loại và độ tuổi:**

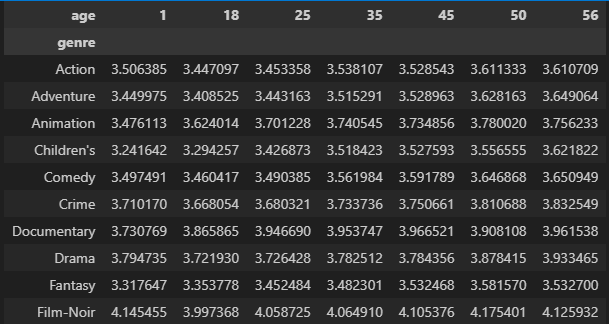
genre\_ratings = (ratings\_with\_genre.groupby(["genre", "age"])["rating"].mean().unstack("age")):

Nhóm dữ liệu theo cột "genre" và "age".

Tính trung bình cột "rating" cho mỗi nhóm.

unstack("age") chuyển cột "age" thành các cột riêng biệt (1, 18, 25, 35, 45, 50, 56).

**Hiển thị 10 hàng đầu:**

****

### **Giải thích:**

Đoạn mã tạo bảng pivot hiển thị trung bình xếp hạng theo từng thể loại và độ tuổi, giúp so sánh sở thích phim theo nhóm tuổi. Ví dụ, thể loại Film-Noir có điểm số cao nhất (trên 4.0) ở hầu hết các độ tuổi, trong khi Children's có điểm số thấp hơn (khoảng 3.2-3.6).

## 13.3. US Baby Names 1880–2010

Import thư viện

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

Đọc và tổng hợp dữ liệu của tất cả các năm (các file từ yob1880.txt đến yob2011.txt)

pieces = []

for year in range(1880, 2011):

path = f"datasets/babynames/yob{year}.txt"

frame = pd.read\_csv(path, names=["name", "sex", "births"])

# Add a column for the year

frame["year"] = year

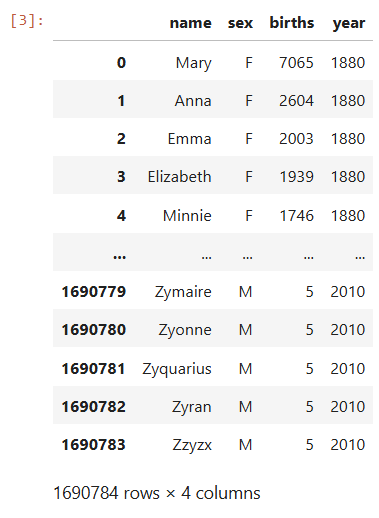
pieces.append(frame)

# Concatenate everything into a single DataFrame

names = pd.concat(pieces, ignore\_index=True)

Hiển thị dữ liệu

names

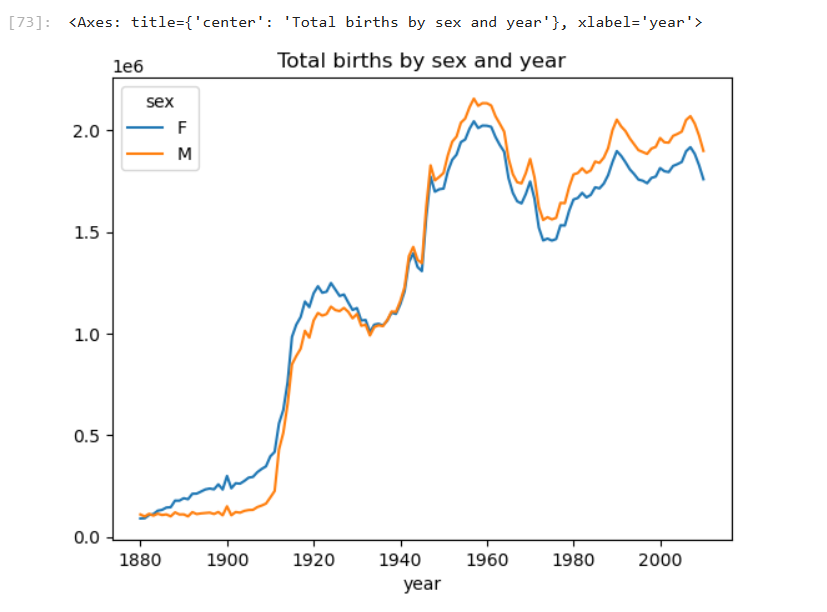
  
Hình 1: Dữ liệu sau khi tổng hợp

Tổng hợp số lượng trẻ em được sinh ra trong các năm

total\_births = names.pivot\_table("births", index="year", columns="sex", aggfunc="sum")

total\_births.tail()

total\_births.plot(title="Total births by sex and year")

  
Hình 2: Tổng số ca sinh các năm

Tính tỷ lệ của mỗi tên

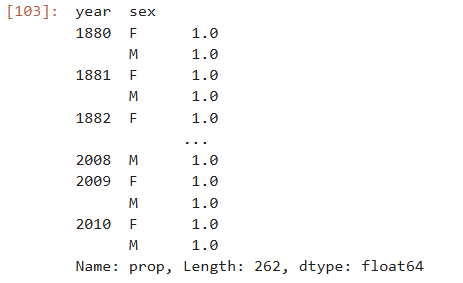
def add\_prop(group):

group["prop"] = group["births"] / group["births"].sum()

return group

names = names.groupby(["year", "sex"], group\_keys=False).apply(add\_prop).reset\_index(drop=True)

names.groupby(["year", "sex"])["prop"].sum()

  
Hình 3: Kiểm tra dữ liệu có hợp lệ

Lọc ra 1000 tên phổ biến nhất mỗi năm

def get\_top1000(group):

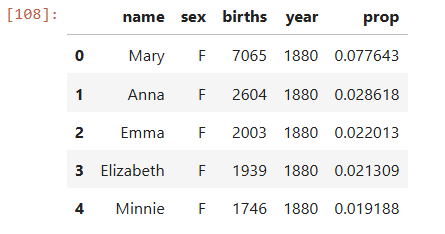
return group.sort\_values("births", ascending=False)[:1000]

grouped = names.groupby(["year", "sex"])

top1000 = grouped.apply(get\_top1000)

top1000 = top1000.reset\_index(drop=True)

top1000.head()

  
Hình 3: Tỉ lệ phổ biến của các tên

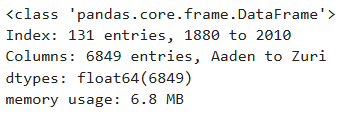
Tách dữ liệu theo giới tính

boys = top1000[top1000["sex"] == "M"]

girls = top1000[top1000["sex"] == "F"]

total\_births = top1000.pivot\_table("births", index="year", columns="name", aggfunc="sum")

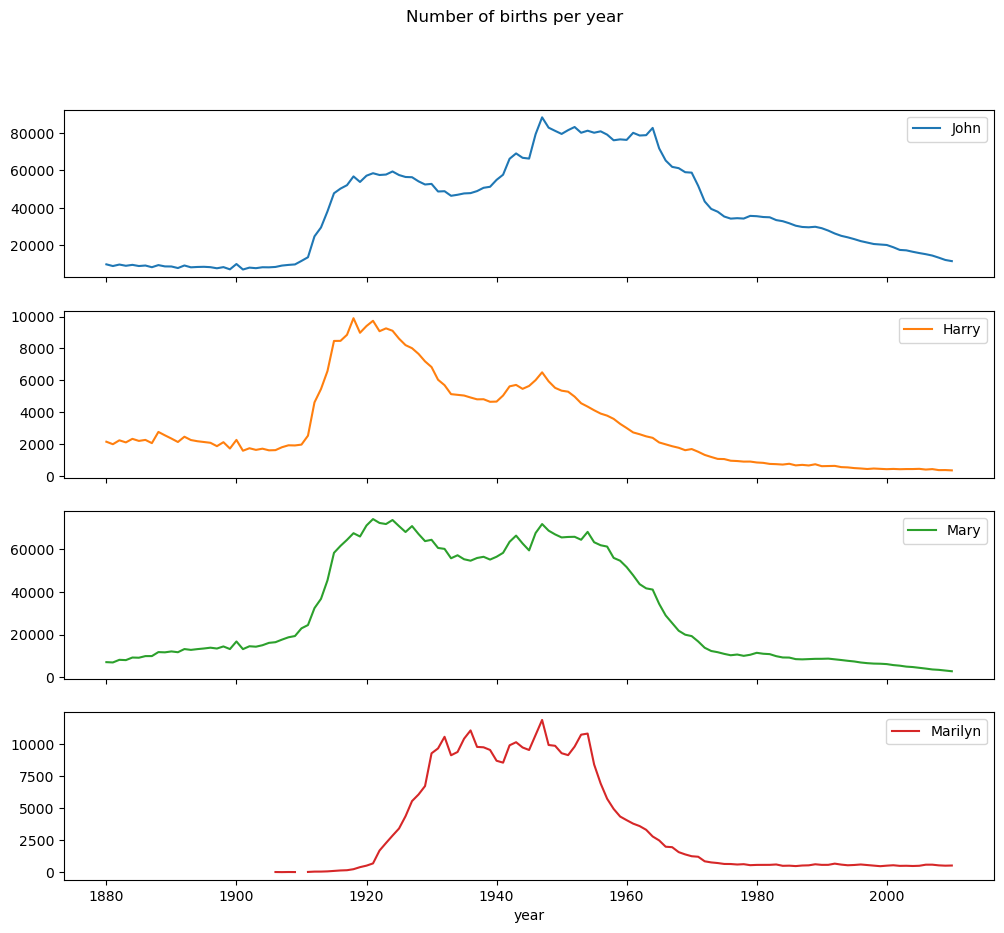
total\_births.info()



Chọn ra 4 tên cụ thể gồm John, Harry, Mary, Marilyn để vẽ biểu đồ xu hướng

subset = total\_births[["John", "Harry", "Mary", "Marilyn"]]

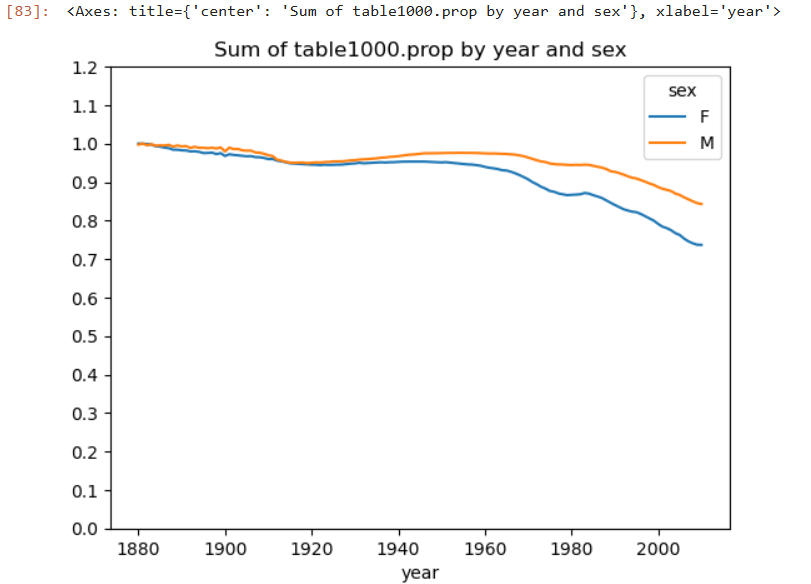
subset.plot(subplots=True, figsize=(12, 10), title="Number of births per year")

  
Hình 4: Biểu đồ xu hướng của 4 tên John, Harry, Mary, Marilyn

Tính tổng tỷ lệ của 1000 tên hàng đầu theo giới tính và năm.

table = top1000.pivot\_table("prop", index="year", columns="sex", aggfunc="sum")

table.plot(title="Sum of table1000.prop by year and sex", yticks=np.linspace(0, 1.2, 13))

  
Hình 5: Tổng tỷ lệ của 1000 tên hàng đầu theo giới tính và năm.

Lọc giới tính Nam năm 2010

df = boys[boys["year"] == 2010]

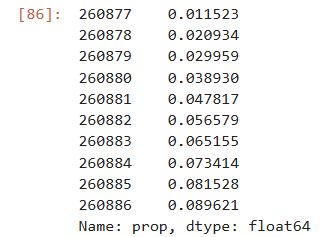
df

  
Hình 6: Lọc giới tính nam năm 2010

Hiển thị phần trăm các tên phổ biến ở giới tính Nam

prop\_cumsum = df["prop"].sort\_values(ascending=False).cumsum()

prop\_cumsum[:10]

  
Hình 7: Phần trăm các tên phổ biến ở giới tính nam

Phân tích sâu hơn về sự đa dạng: Tính số lượng tên cần thiết để chiếm 50% tổng số ca sinh. Thể hiện mỗi năm có bao nhiêu tên chiếm nửa dân số.

prop\_cumsum.searchsorted(0.5)

df = boys[boys.year == 1900]

in1900 = df.sort\_values("prop", ascending=False).prop.cumsum()

in1900.searchsorted(0.5) + 1

def get\_quantile\_count(group, q=0.5):

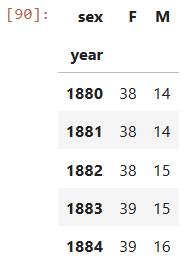
group = group.sort\_values("prop", ascending=False)

return group.prop.cumsum().searchsorted(q) + 1

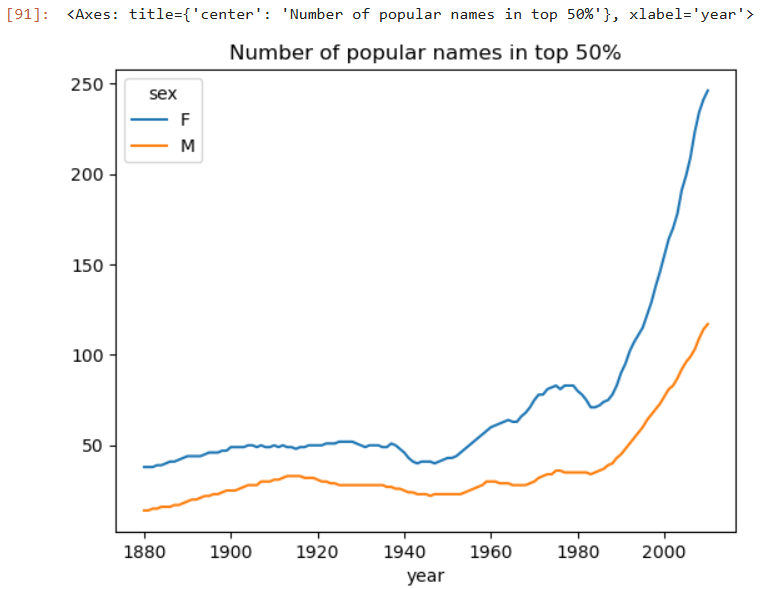
diversity = top1000.groupby(["year", "sex"]).apply(get\_quantile\_count)

diversity = diversity.unstack()

diversity.head()

  
Hình 8: Số lượng tên chiếm 50% tổng số ca sinh trong các năm

diversity.plot(title="Number of popular names in top 50%")

  
Hình 9: Biểu đồ thể hiện số lượng tên chiếm 50% tổng số ca sinh trong các năm

Phân tích chữ cái cuối cùng trong tên

def get\_last\_letter(x):

return x[-1]

last\_letters = names["name"].map(get\_last\_letter)

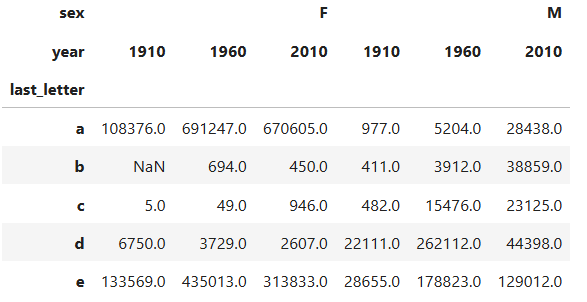
last\_letters.name = "last\_letter"

table = names.pivot\_table("births", index=last\_letters, columns=["sex", "year"], aggfunc="sum")

Chọn 3 năm đại diện để so sánh (năm 1910, 1960, 2010). Xuất ra 10 dòng đầu của chữ cái cuối cùng trong tên.

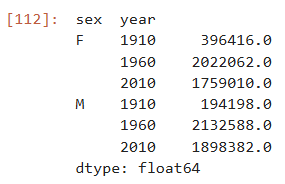
subtable = table.reindex(columns=[1910, 1960, 2010], level="year")

subtable.head()

  
Hình 10: Tỉ lệ các chữ cái cuối cùng trong tên

Tổng số ca sinh của nữ và nam trong các năm 1910, 1960, 2010

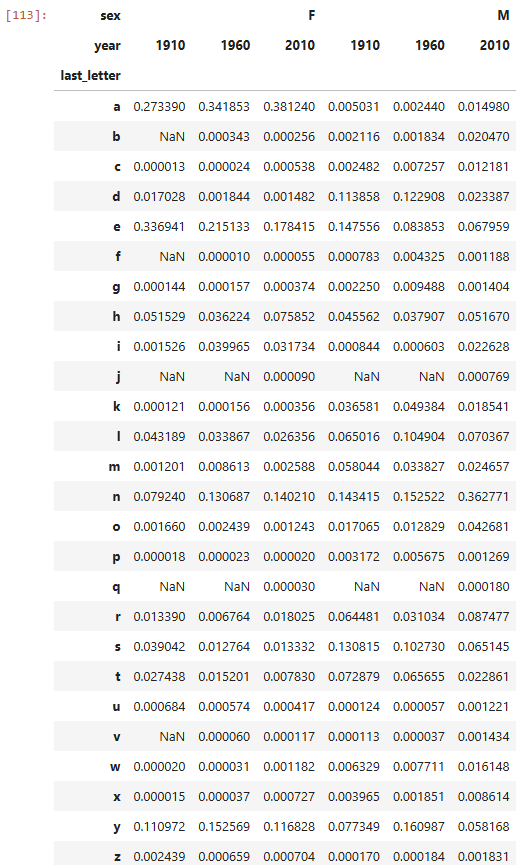
subtable.sum()

  
Hình 11: Tổng số ca sinh của nam và nữ trong các năm 1910, 1960, 2010

Chia số sinh từng chữ cái cho tổng sinh → tỉ lệ phần trăm theo năm/giới.

letter\_prop = subtable / subtable.sum()

letter\_prop

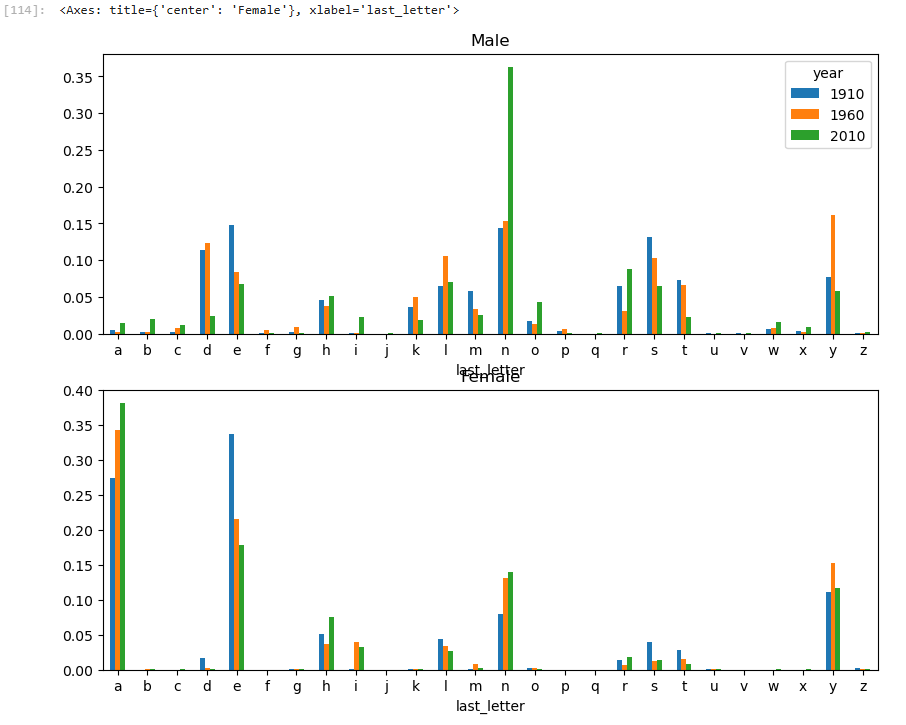
  
Hình 12: Tỉ lệ từng chữ cái cho tổng sinh theo năm/giới.

Vẽ biểu đồ tỉ lệ kết thúc chữ cái

fig, axes = plt.subplots(2, 1, figsize=(10, 8))

letter\_prop["M"].plot(kind="bar", rot=0, ax=axes[0], title="Male")

letter\_prop["F"].plot(kind="bar", rot=0, ax=axes[1], title="Female", legend=False)

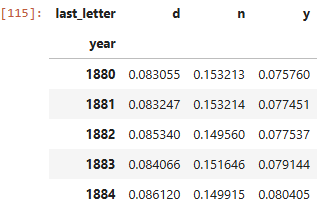
  
Hình 13: Biểu đồ tỉ lệ kết thúc chữ cái

Trực quan hóa xu hướng tên có chữ cái cuối của nam kết thúc bằng các chữ cái "d", "n", và "y".

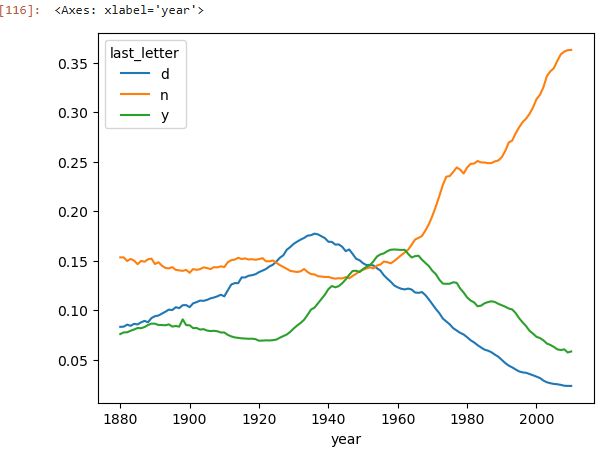
letter\_prop = table / table.sum()

dny\_ts = letter\_prop.loc[["d", "n", "y"], "M"].T

dny\_ts.head()

  
Hình 14: Tỉ lệ tên chữ cái cuối của nam kết thúc bằng "d", "n", và "y"

dny\_ts.plot()

  
Hình 15: Biểu đồ thể hiện tỉ lệ tên chữ cái cuối của nam kết thúc bằng "d", "n", và "y"

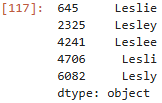
Phân tích tên chuyển đổi giới tính.

Tổng số tên có cụm “Lesl”

all\_names = pd.Series(top1000["name"].unique())

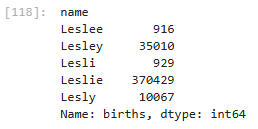
lesley\_like = all\_names[all\_names.str.contains("Lesl")]

lesley\_like

  
Hình 16: Tổng số tên có cụm “Lesl”

filtered = top1000[top1000["name"].isin(lesley\_like)]

filtered.groupby("name")["births"].sum()

  
Hình 17: Hiển thị tổng số trẻ sinh theo từng biến thể của tên.

So sánh tỷ lệ giới tính của các tên đã phân tích.

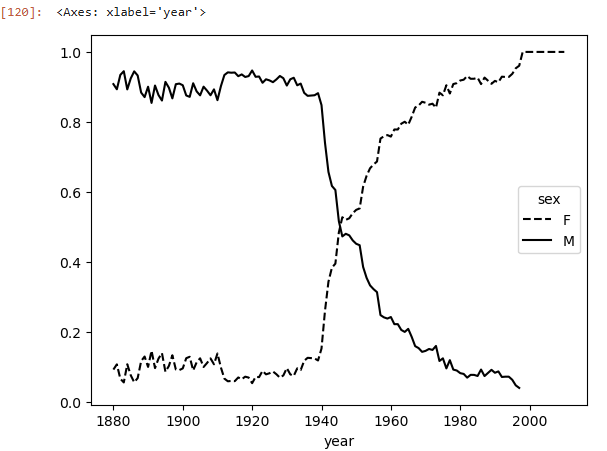
table = filtered.pivot\_table("births", index="year", columns="sex", aggfunc="sum")

table = table.div(table.sum(axis="columns"), axis="index")

table.tail()

  
Hình 18: Tỉ lệ của tên có cụm “Lesl” theo giới tính từ năm 2006 trở đi

table.plot(style={"M": "k-", "F": "k--"})

  
 Hình 19: Biểu đồ thể hiện tên chứa cụm "Lesl" chuyển đổi giới tính qua từng năm

## **13.4. USDA Food Database**

Import dữ liệu

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

Đọc và kiểm tra dữ liệu

import json

db = json.load(open("datasets/usda\_food/database.json"))

len(db)

  
Hình 20: Đếm tổng số mục thực phẩm có trong cơ sở dữ liệu

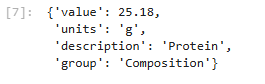
Kiểm tra cấu trúc

db[0].keys()

  
Hình 21: Kiểm tra cấu trúc dữ liệu

Xem chi tiết nutrients

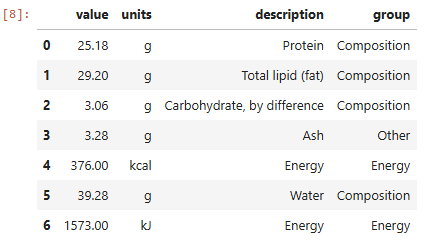
db[0]["nutrients"][0]

  
Hình 22: Hiển thị chi tiết chất dinh dưỡng đầu tiên trong danh sách đó.

Chuyển danh sách nutrients thành DataFrame

nutrients = pd.DataFrame(db[0]["nutrients"])

nutrients.head(7)

  
Hình 23: 7 dòng đầu của DataFrame vừa tạo

Trích xuất thông tin cơ bản của món ăn

info\_keys = ["description", "group", "id", "manufacturer"]

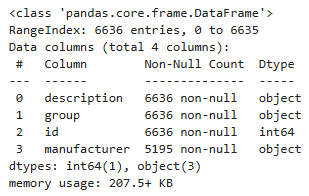
info = pd.DataFrame(db, columns=info\_keys)

info.head(10)

  
Hình 24: 10 dòng đầu về thông tin cơ bản của món ăn

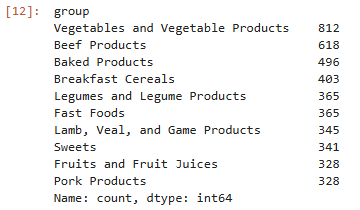
Kiểm tra bảng info

info.info()

  
Hình 25: Thông tin bảng info

Thống kê các nhóm thực phẩm

pd.value\_counts(info["group"])[:10]

  
Hình 25: Đếm số lượng thực phẩm trong mỗi nhóm

Tạo bảng nutrients cho dữ liệu

nutrients = []

for rec in db:

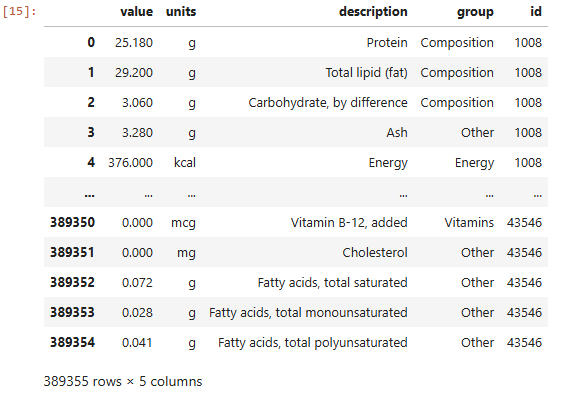
fnuts = pd.DataFrame(rec["nutrients"])

fnuts["id"] = rec["id"]

nutrients.append(fnuts)

nutrients = pd.concat(nutrients, ignore\_index=True)

nutrients

  
Hình 26: Hiển thị bảng Nutrients

Làm sạch dữ liệu

nutrients.duplicated().sum() # number of duplicates

nutrients = nutrients.drop\_duplicates()

Đổi tên cột

col\_mapping = {"description" : "food",

"group" : "fgroup"}

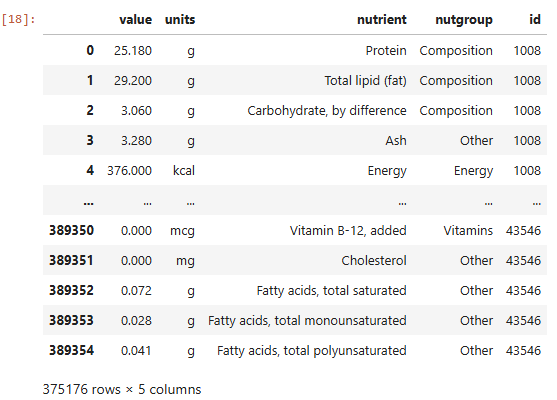
info = info.rename(columns=col\_mapping, copy=False)

col\_mapping = {"description" : "nutrient",

"group" : "nutgroup"}

nutrients = nutrients.rename(columns=col\_mapping, copy=False)

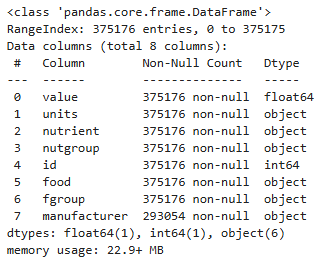
nutrients

  
Hình 27: Bảng nutrients sau khi đã làm sạch dữ liệu và đổi tên cột

Gộp dữ liệu

ndata = pd.merge(nutrients, info, on="id")

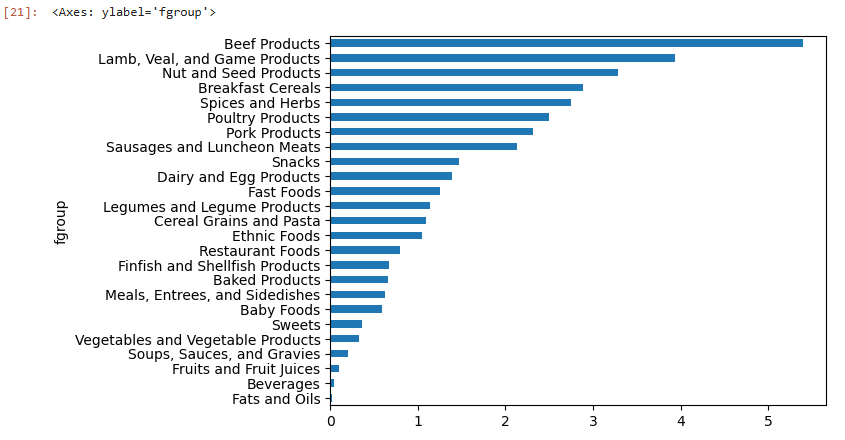
ndata.info()

  
Hình 28: Thông tin dữ liệu sau khi gộp

Phân tích dữ liệu tổng

result = ndata.groupby(["nutrient", "fgroup"])["value"].quantile(0.5)

result["Zinc, Zn"].sort\_values().plot(kind="barh")

  
Hình 29: Biểu đồ so sánh lượng kẽm trung vị giữa các nhóm thực phẩm

Tìm thực phẩm giàu dinh dưỡng nhất

by\_nutrient = ndata.groupby(["nutgroup", "nutrient"])

def get\_maximum(x):

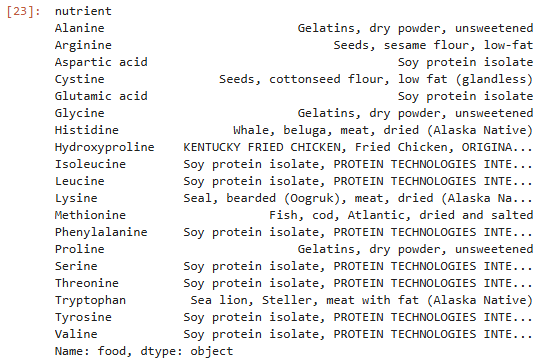
return x.loc[x.value.idxmax()]

max\_foods = by\_nutrient.apply(get\_maximum)[["value", "food"]]

# make the food a little smaller

max\_foods["food"] = max\_foods["food"].str[:50]

max\_foods.loc["Amino Acids"]["food"]

  
Hình 30: Tên các loại thực phẩm có hàm lượng cao nhất cho mỗi chất dinh dưỡng trong nhóm "Axit Amin"

## **13.5. 2012 Federal Election Commission Database**

Đọc dữ liệu

fec = pd.read\_csv("P00000001-ALL.csv", low\_memory=False)

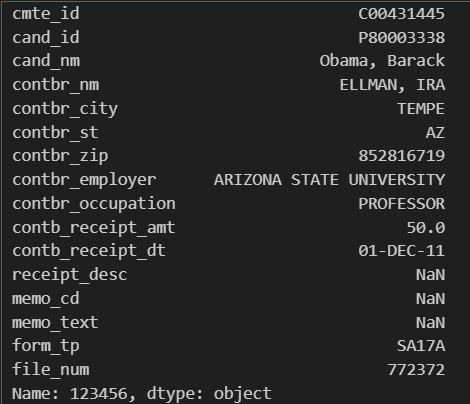
Hiển thị thông tin DataFrame

fec.info()

  
Hình 31: Thông tin DataFrame

Hiển thị dòng dữ liệu

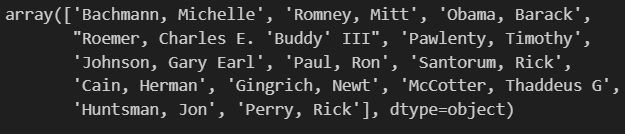
fec.iloc[123456]

  
Hình 32: Thông tin của 1 dòng dữ liệu

Lấy tên các ứng cử viên duy nhất

unique\_cands = fec["cand\_nm"].unique()

unique\_cands

  
Hình 33: Tên các ứng cử viên duy nhất

Hiển thị tên ứng cử viên từ dòng 123456 đến 123460

parties = {"Bachmann, Michelle": "Republican",

"Cain, Herman": "Republican",

"Gingrich, Newt": "Republican",

"Huntsman, Jon": "Republican",

"Johnson, Gary Earl": "Republican",

"McCotter, Thaddeus G": "Republican",

"Obama, Barack": "Democrat",

"Paul, Ron": "Republican",

"Pawlenty, Timothy": "Republican",

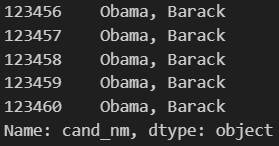
"Perry, Rick": "Republican",

"Roemer, Charles E. 'Buddy' III": "Republican",

"Romney, Mitt": "Republican",

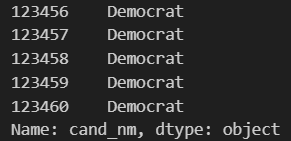
"Santorum, Rick": "Republican"}

fec["cand\_nm"][123456:123461]

  
Hình 34: Tên các ứng viên từ dòng 123456 đến 123460

Ánh xạ tên ứng cử viên sang đảng phái

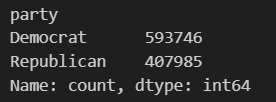
fec["cand\_nm"][123456:123461].map(parties)

  
Hình 35: ánh xạ tên ứng cử viên sang đảng phái

Hiển thị số lượng ứng cử viên theo đảng phái

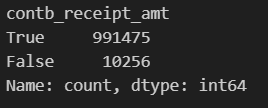
fec["party"] = fec["cand\_nm"].map(parties)

fec["party"].value\_counts()

  
Hình 36: Hiển thị số lượng ứng cử viên theo đảng phái

Hiển thị số lượng đóng góp tích cực và tiêu cực

(fec["contb\_receipt\_amt"] > 0).value\_counts()

  
Hình 37: Hiển thị số lượng đóng góp tích cực và tiêu cực

Lọc các đóng góp tích cực

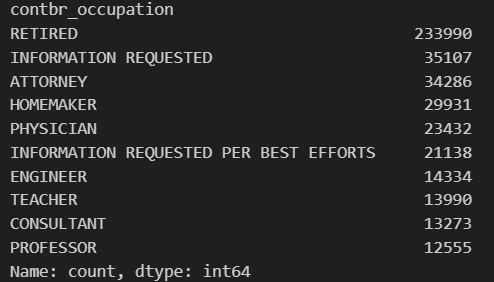
fec = fec[fec["contb\_receipt\_amt"] > 0]

Lọc các đóng góp của Obama và Romney

fec\_mrbo = fec[fec["cand\_nm"].isin(["Obama, Barack", "Romney, Mitt"])]

Liệt kê 10 nghề nghiệp đóng góp nhiều nhất

fec["contbr\_occupation"].value\_counts()[:10]

  
Hình 38: Liệt kê 10 nghề nghiệp đóng góp nhiều nhất

Ánh xạ các nghề nghiệp không rõ ràng thành "NOT PROVIDED"

occ\_mapping = {

"INFORMATION REQUESTED PER BEST EFFORTS" : "NOT PROVIDED",

"INFORMATION REQUESTED" : "NOT PROVIDED",

"INFORMATION REQUESTED (BEST EFFORTS)" : "NOT PROVIDED",

"C.E.O.": "CEO"

}

Hàm ánh xạ nghề nghiệp

def get\_occ(x):

# If no mapping provided, return x

return occ\_mapping.get(x, x)

Áp dụng hàm get\_occ để ánh xạ các nghề nghiệp

fec["contbr\_occupation"] = fec["contbr\_occupation"].map(get\_occ)

Ánh xạ các nhà tuyển dụng không rõ ràng thành "NOT PROVIDED"

emp\_mapping = {

"INFORMATION REQUESTED PER BEST EFFORTS" : "NOT PROVIDED",

"INFORMATION REQUESTED" : "NOT PROVIDED",

"SELF" : "SELF-EMPLOYED",

"SELF EMPLOYED" : "SELF-EMPLOYED",

}

Hàm ánh xạ nhà tuyển dụng

def get\_emp(x):

# If no mapping provided, return x

return emp\_mapping.get(x, x)

Áp dụng hàm get\_emp để ánh xạ các nhà tuyển dụng

fec["contbr\_employer"] = fec["contbr\_employer"].map(get\_emp)

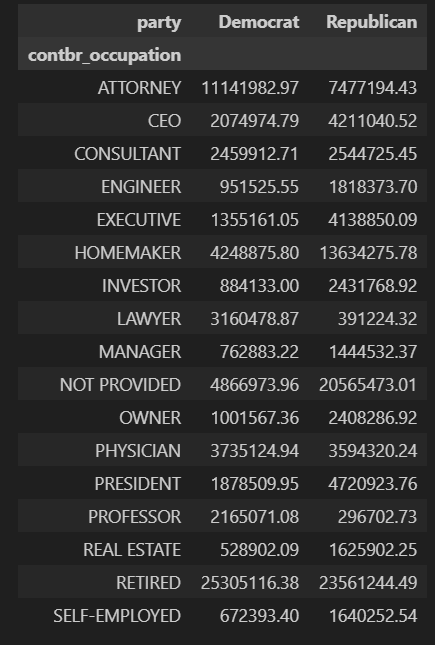
Tạo bảng tổng hợp số tiền đóng góp theo nghề nghiệp và đảng phái

by\_occupation = fec.pivot\_table("contb\_receipt\_amt",index="contbr\_occupation",columns="party", aggfunc="sum")

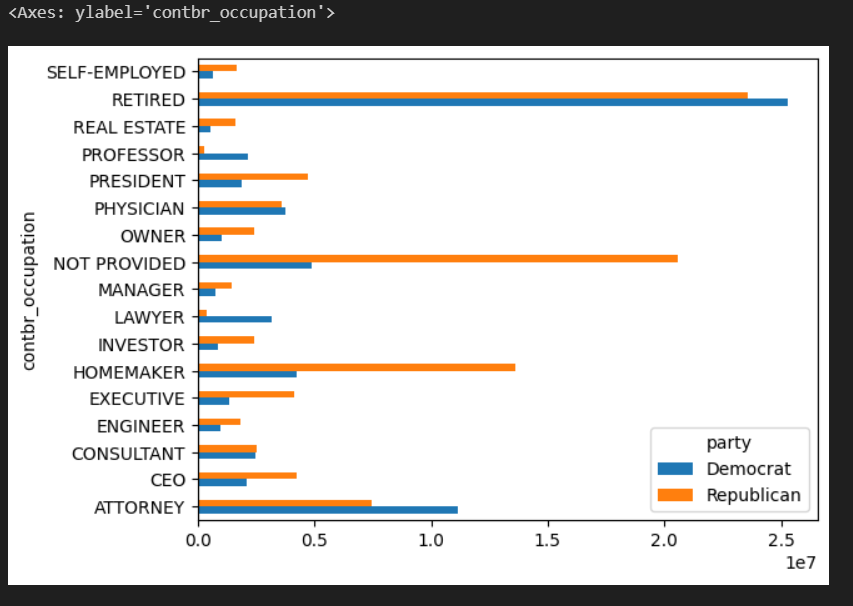
Lấy các nghề nghiệp có tổng số tiền đóng góp lớn hơn 2 triệu

over\_2mm = by\_occupation[by\_occupation.sum(axis="columns") > 2000000]

over\_2mm

  
Hình 39: Hiển thị các nghề nghiệp có tổng số tiền đóng góp lớn hơn 2 triệu

over\_2mm.plot(kind="barh")

  
Hình 40: Biểu đồ hiển thị các nghề nghiệp có tổng số tiền đóng góp lớn hơn 2 triệu

Hàm lấy ra n giá trị đóng góp lớn nhất theo nhóm

def get\_top\_amounts(group, key, n=5):

totals = group.groupby(key)["contb\_receipt\_amt"].sum()

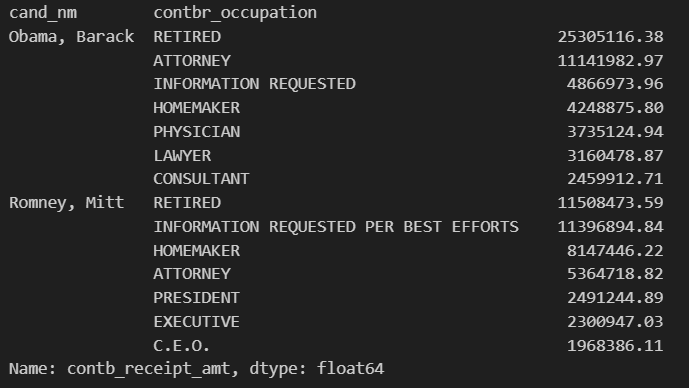
return totals.nlargest(n)

Nhóm dữ liệu theo tên ứng cử viên

grouped = fec\_mrbo.groupby("cand\_nm")

Lấy ra 7 nghề nghiệp đóng góp nhiều nhất cho mỗi ứng cử viên

grouped.apply(get\_top\_amounts, "contbr\_occupation", n=7)

  
Hình 41: Hiển thị 7 nghề nghiệp đóng góp nhiều nhất cho mỗi ứng cử viên

Lấy ra 10 nhà tuyển dụng đóng góp nhiều nhất cho mỗi ứng cử viên

grouped.apply(get\_top\_amounts, "contbr\_employer", n=10)

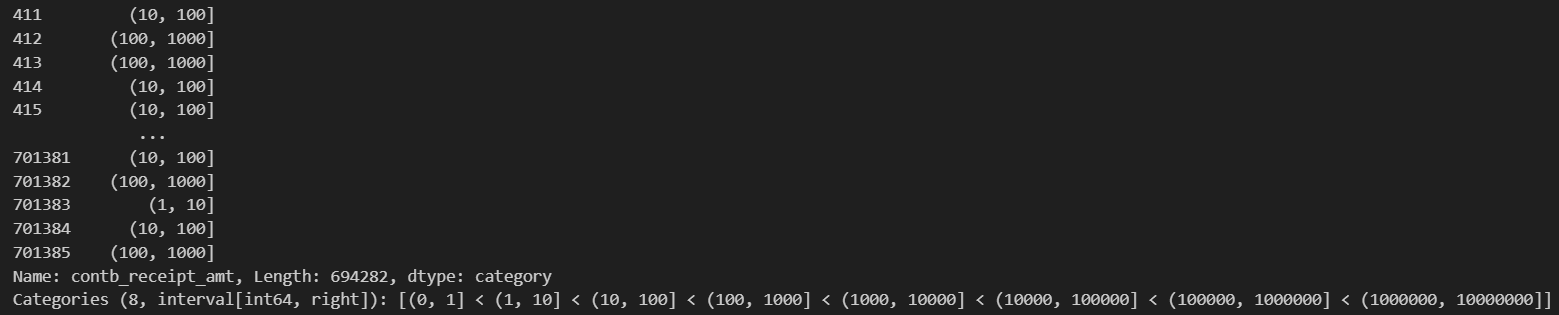
  
Hình 42: Hiển thị 10 nhà tuyển dụng đóng góp nhiều nhất cho mỗi ứng cử viên

Phân nhóm dữ liệu theo mức tiền

bins = np.array([0, 1, 10, 100, 1000, 10000,100\_000, 1\_000\_000, 10\_000\_000])

labels = pd.cut(fec\_mrbo["contb\_receipt\_amt"], bins)

labels

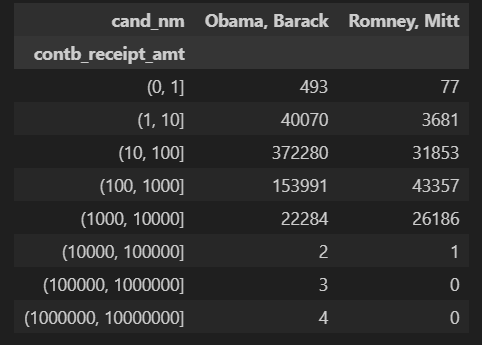
  
Hình 43: Hiển thị phân nhóm dữ liệu theo mức tiền

Nhóm dữ liệu theo tên ứng cử viên và mức tiền

grouped = fec\_mrbo.groupby(["cand\_nm", labels])

Hiển thị kích thước của mỗi nhóm

grouped.size().unstack(level=0)

  
Hình 44: Hiển thị kích thước của mỗi nhóm

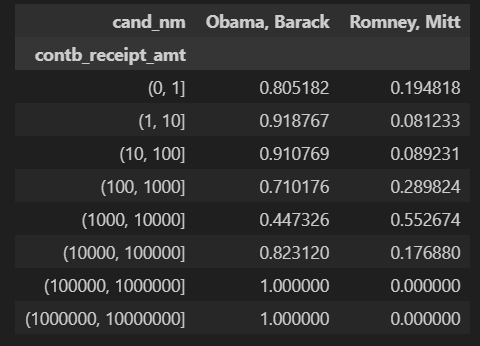
Tính tổng số tiền đóng góp cho mỗi nhóm

bucket\_sums = grouped["contb\_receipt\_amt"].sum().unstack(level=0)

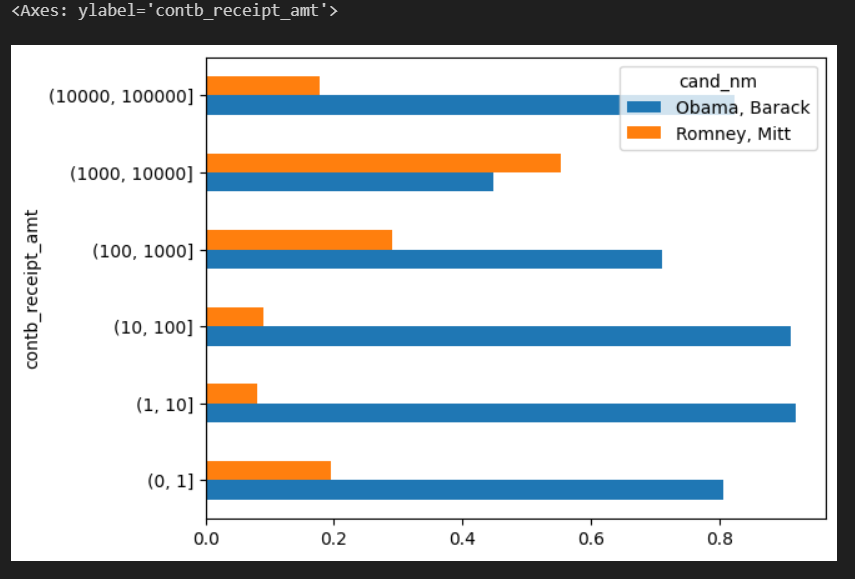
Chuẩn hóa dữ liệu

normed\_sums = bucket\_sums.div(bucket\_sums.sum(axis="columns"),axis="index")

normed\_sums

  
Hình 45: Dữ liệu sau khi chuẩn hóa

normed\_sums[:-2].plot(kind="barh")

  
Hình 46: Biểu đồ dữ liệu sau khi chuẩn hóa

Nhóm dữ liệu theo tên ứng cử viên và mức tiền

grouped = fec\_mrbo.groupby(["cand\_nm", "contbr\_st"])

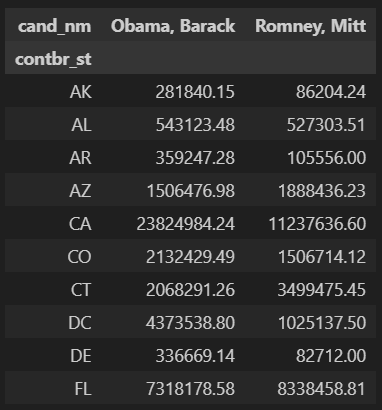
Hiển thị kích thước của mỗi nhóm

totals = grouped["contb\_receipt\_amt"].sum().unstack(level=0).fillna(0)

Lọc các bang có tổng số tiền đóng góp lớn hơn 100000

totals = totals[totals.sum(axis="columns") > 100000]

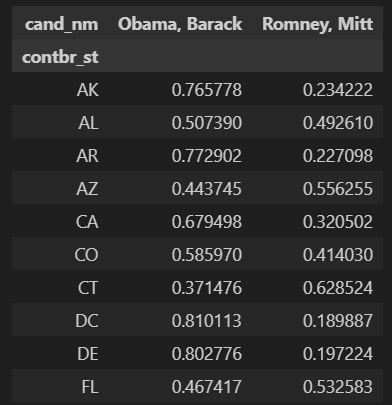
totals.head(10)

  
Hình 47: Hiển thị 10 giá trị đầu tiên

Tính tỷ lệ phần trăm của từng giá trị trong hàng so với tổng hàng

percent = totals.div(totals.sum(axis="columns"), axis="index")

percent.head(10)

  
Hình 48: Hiển thị 10 giá trị đầu tiên

# **CHƯƠNG III. Kết luận**

Bài thực hành lớn 02 đã hoàn thành các mục tiêu đề ra, giúp sinh viên nắm vững các kỹ thuật cơ bản trong Phân tích Khám phá Dữ liệu (Exploratory Data Analysis - EDA). Thông qua việc phân tích 5 bộ dữ liệu thực tế , nhóm đã áp dụng thành công các thư viện Python, chủ yếu là pandas, NumPy và seaborn , để thực hiện toàn bộ quy trình từ làm sạch, chuẩn bị, phân tích đến trực quan hóa dữ liệu.

Bài thực hành đã giúp các thành viên củng cố kiến thức về một quy trình phân tích dữ liệu hoàn chỉnh. Mỗi bộ dữ liệu đều có những thách thức riêng, đòi hỏi việc vận dụng linh hoạt các công cụ của pandas để xử lý. Các kỹ năng về tiền xử lý, tổng hợp, và trực quan hóa đã được rèn luyện, đáp ứng tốt yêu cầu của môn học Phân tích Dữ liệu.

HẾT