Bài 3
Thao tác dữ liệu

Khóa học: Phân tích dữ liệu với Python

Nội dung

1. Lọc dữ liệu
2. Gom nhóm dữ liệu
3. Sinh dữ liệu từ các cột dữ liệu đã có

1. Lọc dữ liệu

Quy mô dữ liệu của doanh nghiệp ngày càng tăng do xu hướng số hóa doanh nghiệp:

- làm gia tăng quy mô dữ liệu
- sự thiếu nhất quán
- nhiều thông tin nhiễu

 Lọc dữ liệu trả về bộ dữ liệu phù hợp với nhu cầu người dùng.

Các phương pháp lọc dữ liệu:

- Lọc trực tiếp bởi chỉ số dòng và tên cột
- Lọc bằng phương thức filter
- Lọc dữ liệu có điều kiện
- Lọc có điều kiện bằng phương thức query

Lọc trực tiếp bởi chỉ số dòng và tên cột

- Lọc các cột dữ liệu từ tập dữ liệu df: df[[‘Tên_cột_1', …,‘Tên_cột_n']]
- Lọc một dòng dữ liệu có chỉ số cs: df[cs]
- Lọc nhiều dòng dữ liệu liên tiếp nhau: df[chỉ_số_dòng_đầu : chỉ_số_dòng_cuối]
- Lọc dữ liệu theo cột và dòng:

  df[[‘Tên_cột_1', …,‘Tên_cột_n']] [chỉ_số_dòng_đầu : chỉ_số_dòng_cuối]

Lọc bằng phương thức filter

- Lọc các cột dữ liệu từ tập dữ liệu df: df.filter([‘Tên_cột_1', ...,‘Tên_cột_n'])
- Lọc các dòng dữ liệu rời rạc: df.filter([chỉ_số_dòng_1,... ,chỉ_số_dòng_2], axis=0)
- Lọc dữ liệu theo cột và dòng:

  df.filter([‘Tên_cột_1', ...,‘Tên_cột_n']) [chỉ_số_dòng_đầu : chỉ_số_dòng_cuối]

Lọc dữ liệu có điều kiện

- Lọc các bản ghi theo điều kiện đơn: df[df[‘Tên_cột’] PTSS giá_trị]

  Trong đó phép toán so sánh PTSS sử dụng các phép toán so sánh gồm: ==, <, <=, >, >=, !=

  (giá_trị cần so sánh được để trong cặp dấu nháy đơn nếu có kiểu chuỗi ký tự)
- Lọc các bản ghi theo điều kiện phức:

  df[(df[‘Tên_cột_1’] PTSS giá_trị) PTLG (df[‘Tên_cột_n’] PTSS giá_trị_n)]

  Các điều kiện được nối với nhau bởi phép toán logic PTLG gồm: & (và), | (hoặc), ~ (phủ định)
- Lọc các cột và các bản ghi theo điều kiện:

  df[df[‘Tên_cột’] PTSS giá_trị]. filter([‘Tên_cột_1', ...,‘Tên_cột_n'])

Lọc có điều kiện bằng phương thức query

- Phương thức truy vấn query giúp câu lệnh truy vấn chạy nhanh hơn
- Các điều kiện được nối với nhau bởi các phép toán logic như: and (và), or (hoặc), not (phủ định) và được để trong một cặp dấu nháy kép.
- Cú pháp: df.query("(Tên_cột_1 PTSS giá_trị_1) PTLG (Tên_cột_n PTSS giá_trị_n)")

2. Gom nhóm dữ liệu

- Nhóm – groupby là một hình thức tổng hợp dữ liệu theo chiến lược tách-áp dụng-kết hợp. Đầu tiên dữ liệu được chia thành các nhóm giống nhau rồi áp dụng các phép toán tổng hợp như mean, min, max, count, sum… trên mỗi nhóm và kết hợp các kết quả từ mỗi nhóm.
- Cú pháp cơ bản của groupby:

  dataFrame.groupby('Thuộc tính cần phân nhóm')['Thuộc tính cần tính giá trị'].phép toán()

- Ví dụ: Tính trung bình cộng ngày công theo từng phòng ban.

3. Sinh dữ liệu từ các cột dữ liệu đã có

Trong Python ta có thể dễ dàng tạo cột dữ liệu mới từ các cột dữ liệu đã có trong dataframe thông qua việc kết nối các cột dữ liệu thông các phép toán số học, logic, ghép chuỗi....

- Ví dụ: df['Thành_tiền'] = df['Số_lượng'] + df['Đơn_giá']
- Trong phép toán ghép chuỗi nếu một cột chứa giá trị không có kiểu chuỗi thì sử dụng hàm astype(str) để đổi về kiểu chuỗi.

Thêm cột mới theo điều kiện

- Điều kiện đơn: ví dụ thêm cột new có giá trị True nếu join_year = 2021 và False trong trường hợp còn lại.

```
df['new'] = df['join_year'] == 2021
```

- Điều kiện có 2 lựa chọn – sử dụng np.where

Ví dụ thêm cột rate có giá trị good nếu rating_good >= 50000, bad trong trường hợp còn lại.

```
import numpy as np
df['rate'] = np.where(df['rating_good'] >= 50000,'good','bad' )
```

Thêm cột mới theo điều kiện

- Điều kiện có nhiều lựa chọn: sử dụng np.select

  Ví dụ: Thêm cột flag tặng cờ cho các cửa hàng, flag nhận các giá trị như sau:

  blue khi rating_good >= 30000 và rating_bad <= 100

  yellow khi 10000 <= rating_good < 30000 và 100 < rating_bad <= 1000

  red khi rating_good < 10000

  black đối với các trường hợp còn lại.

```
conditions = [(df['rating_good'] >= 30000) & (df['rating_bad'] <= 100),
              (df['rating_good'] >= 10000) & (df['rating_good'] < 30000) & (df['rating_bad'] <=
              1000) & (df['rating_bad'] > 100), (df['rating_good'] < 10000)]
choices = ['blue', 'yellow','red']
df['flag'] = np.select(conditions, choices, default='black')
```

Tách dữ liệu của một cột thành nhiều cột

- Trong quá trình thu thập và xử lý dữ liệu, chúng ta thường phải thực hiện thao tác bóc tách dữ liệu từ 1 cột thành 2 hay nhiều cột để dễ dàng theo dõi và quản lý dữ liệu.
- Dữ liệu được tách thường ở dạng chuỗi và thời gian

Tách cột dữ liệu có kiểu chuỗi

- Sử dụng hàm str.split('Ký tự phân cách') có tác dụng tách chuỗi ban đầu str thành các chuỗi con được ngăn cách nhau bởi ký tự phân cách.
- Các chuỗi con được lưu vào mảng str
- Ví dụ: Tách cột Địa_chỉ có dạng ‘Chùa Bộc, Đống Đa’ thành hai cột Phường, Quận

  df[‘Phường']=df['Địa_chỉ'].str.split(',').str[0]

  df[‘Quận']=df['Địa_chỉ'].str.split(',').str[1]

Tách cột dữ liệu có kiểu Date

- Sử dụng hàm to_datetime() chuyển một chuỗi về dạng ngày tháng theo định dạng format.
- Ví dụ: Tách cột Ngày_mua có dạng ’15/08/2021’ thành ba cột Ngày, Tháng, Năm

```
df[‘Ngày']=pd.to_datetime(df[‘Ngày_mua'],format='%d/%m-%Y').dt.day
df[‘Tháng']=pd.to_datetime(df[‘Ngày_mua'],format='%d/%m-%Y').dt.month
df[‘Năm']=pd.to_datetime(df[‘Ngày_mua'],format='%d/%m-%Y').dt.year
```

Tách cột dữ liệu có kiểu Time

- Sử dụng hàm to_datetime() chuyển một chuỗi về dạng thời gian theo định dạng format.
- Ví dụ: Tách cột Thời_gian có dạng ’20:45:32’ thành ba cột Giờ, Phút, Giây

```
df[‘Giờ']=pd.to_datetime(df[‘Thời_gian'],format=‘%H:%M:%S’).dt.hour
df[‘Phút']=pd.to_datetime(df[‘Thời_gian'],format=‘%H:%M:%S’).dt.minute
df[‘Giây]=pd.to_datetime(df[‘Thời_gian'],format=‘%H:%M:%S’).dt.second
```

Tóm tắt

Qua bài học này chúng ta đã biết:

- Cách lọc các cột/dòng dữ liệu theo chỉ số, điều kiện, truy vấn (query)
- Cách tổng hợp dữ liệu theo nhóm
- Cách tạo một cột dữ liệu mới từ các cột dữ liệu đã có sử dụng các phép toán số học, phép toán nối chuỗi, phép toán logic…
- Cách phân tách dữ liệu có kiểu chuỗi, kiểu ngày tháng, kiểu thời gian