1. Movies.get\_dtype\_counts() -> 속성 없다면서 실행안됨
2. .ge(.15)
3. stiringIO는 입력값 자체를 실체는 문자열이지만 파일처럼 취급되도록 하는 객체를 만들어 주는 것

from io import StringIO

fout = StringIO()

beatles.to\_csv(fout)

fout.getvalue()

1. chunksize=200

def process(df):

return f'processed {df.size} items'

for chunk in diamonds\_iter:

process(chunk)

1. diamonds4.to\_feature('./d.arr')

diamonds4.to\_parquet('/tmp/d.pqt')

1. pandas의 객체에 다른 라이브러리의 함수를 적용하는 방법

pipe() : 테이블형태로 정리해주는 함수 어플리케이션

apply() : 행 혹은 열로 정리해주는 함수 어플리케이션

agg(), transform() : 집합 API

applymap() : 요소별 적용 함수

1. .pipe(func, \*args, \*\*kwargs) 와 .transform()의 차이점

.pipe()를 사용하면 func(func(func())) 이렇게 되는 걸 보기 쉽게 사용할 수 있음

1. .transform(func, axis-0, args, \*\*kwargs)

1. np.iinfo(np.int16)
2. np.finfo()
3. .nlargest(n, columns, keep='first') 는 null값을 제거하는가?

keep='first', keep='last' 는 중복될 때 첫 번째 값을 우선순위에 둘 지, 마지막 값을 우선순위에 둘 지 설정

1. .nsmallest()는 null값을 제거하는가?

→

1. .where() Series와 DataFrame 모두 어떻게 동작하는지 확인

np.where(배열에 대한 조건문, 참일때 값, 거짓일때 값)

pd.where() / Series.where(시리즈객체에 대한 조건문, 거짓 값에 대한 대체 값)

1. college.loc['E':'B']
2. 8장

employee['RANDOM\_SALARY'] = random\_salary['BASE\_SALARY']

(college\_n

.eq(college\_n.max())

.cumsum()

.cumsum()

1. 10장

pd.wide\_to\_long(

df.rename(columns = {'a1':'group1\_a1',

'b2':'group1\_b2',

'd':'group2\_a1', 'e':'group2\_b2'}),

stubnames=['group1', 'group2'],

i=['State', 'Country', 'Test'],

j='Label',

suffix='.+',

sep='\_')

1. .equals()는 정렬순서도 같아야 같은 것으로 나오는가?
2. Pandas에서 아직 개발이 되지 않아 피봇되지 않은 열이 두 개 이상인 경우는 구현이 안된다

inspections.pivot(index=['Name', 'Date'],

columns='Info', values='Value')

1. Squeeze()는 뭐지? 설명으론 1차원인 축만 제거한다고 함

Inspections

.set\_index(['Name','Date', 'Info'])

.squeeze()

1. .pivot과 .pivot\_table의 차이점

p1.pivot(index=컬럼, columns=컬럼, values=컬럼)

p1.pivot('year', 'mon', 'latte') == p1.pivot('year', 'mon')['latte'] #멀티인덱스로 생성

data.pivot\_table(index=컬럼, columns=컬럼, aggfunc=sum, margins=True)

#aggfunc의 default는 mean

1. .mask()는 where와 반대로써 입력받은 조건식에 False에 해당하는 데이터들을 살려둔다. 나머지 값은 NaN으로 만듬
2. +를 할때 카티션 곱이 발생할 수 있으므로 인덱스의 순서, 갯수를 정확히 맞추고 진행해야 한다.
3. .add()도 카티션 곱이 발생할 수 있나?
4. s1.add(s2 , fill\_value=5)를 하면 둘 다 NaN인 것을 빼고 NaN이 하나인 곳에 어디라도 상관없이 5를 넣는다.
5. .transform() / pipe() / apply()

.transform()은 transform(['exp', lambda x:np.exp(-x]) 같이 다중 함수를 입력할 수 있다.

.transform({'A':'exp', 'B':lambda x:np.exp(-x)})

**이거 .transform('max') 와 .max() 랑 왜 다른지 이해가 안됨**

1. .merge() 에서 left\_on이나 right\_on 컬럼 대신 인덱스로도 겹치는 걸 뽑아올 수 있는데 그게 right\_index=Ture, left\_index=Ture 임
2. 8장에서 [lambda college\_n : college\_n] 대신 True만 뽑을 수 있는 방법이 없나?

[college\_n.values == True] 이거는 안된

(college\_n

.eq(college\_n.max())

.cumsum()

.cumsum()

.eq(1)

.any(axis='columns')

[lambda college\_n : college\_n]

)

1. 9장 이거 안되는데 그럼 어떻게 해야되나?

(flights

.groupby('AIRLINE')

.agg(np.sqrt)

)

1. 9장 res\_1 = ['\_'.join(x) for x in res.columns.to\_flat\_index()]

여기서 to\_flat\_index()를 안써도 괜찮던데 왜 쓰는 걸까?

1. 12장 First가 어떤 역할을 하는지 모르겠음

crime\_sort.first(pd.offsets.MonthBegin(6, normalize=True))

1. 9장 .nhn이 뭔지 모르겠음

grouped.nth([1, -1])

1. 14장 np.where 과 df['Q3].where의 차이점을 모르겠음

df.Q3.where(df.Q3.isin(values), 'Another') 와

pd.Series(np.where(df.Q3.isin(values), df.Q3, 'Another'),

index=df.index)

1. Import dask 사용법(램이 아닌 cpu로 사용해서 데이터 처리를 한다고 함)
2. Skt 전처리에서 불린 처리가 빠른가? 아니면 함수? 가 빠른가?

(df['마케팅본부코드']!='A03977')

1. .first()
2. Skt 관련 전처리

* 데이터를 전처리해서 프로디스커버리에 올렸을 떄, 데이터 특성상 분석이 제한되었습니다. 여러 방식으로 분석하기 위해 3개의 데이터셋을 만들었습니다.
  1. v1(빈도 표현용 데이터셋) - 거의 원본 데이터셋
  2. v2(플로우 표현용 데이터셋) - 중복되어 수행된 작업들을 제외한 데이터 셋
  3. v3(작업시간 표현용 데이터셋) - 유니크한 작업만 남겨두고 작업시간의 합을 계산한 데이터 셋
* BIZ 주제가 6개였기 떄문에, 모든 주제에 대해 3개 종류의 데이터셋을 만든 것입니다.

1. 9장 div() 가 뭔지 모르겠음 → 나누기임

(flights

.groupby(cuts)

['AIR\_TIME']

.quantile(q=[.25, .5, .75])

.div(60)

.round(2)

)

1. Find는 인덱스를 찾는 것 같음

fb\_idx = col\_name.find('facebook')

1. 10장 Change\_col\_name 함수에서 어떻게 실행되는지 이해가 안됨(함수를 풀고하면 안됨) → replace와 find는 list는 안되고 for문으로 풀어서 하나하나 돌리고 append로 다시 list로 넣어야 함

def change\_col\_name(col\_name):

col\_name = col\_name.replace('\_name', '')

if 'facebook' in col\_name:

fb\_idx = col\_name.find('facebook')

col\_name = (col\_name[:5] + col\_name[fb\_idx - 1:]

+ col\_name[5:fb\_idx-1])

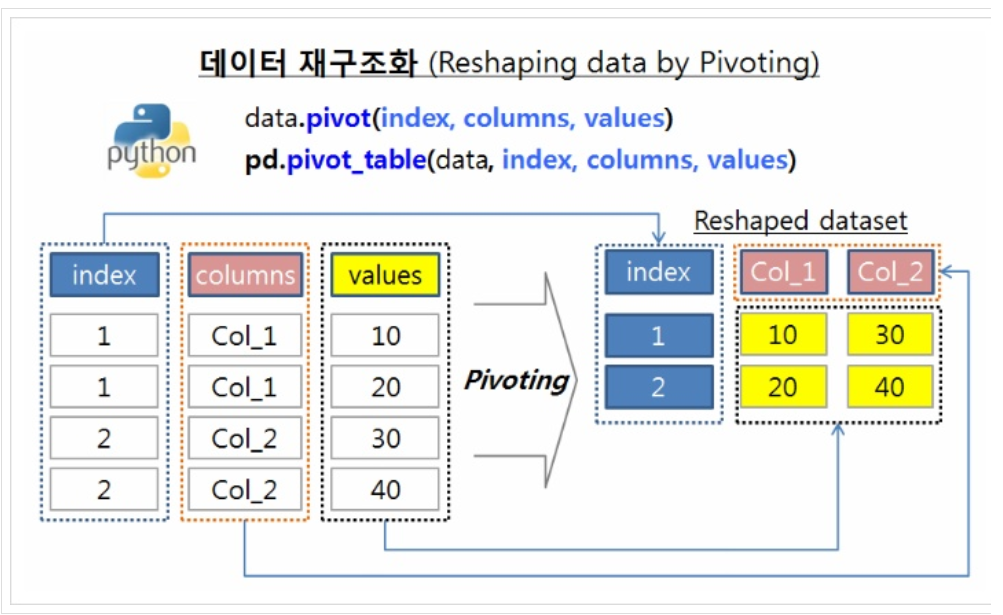
return col\_name

actor2 = actor.rename(columns=change\_col\_name)

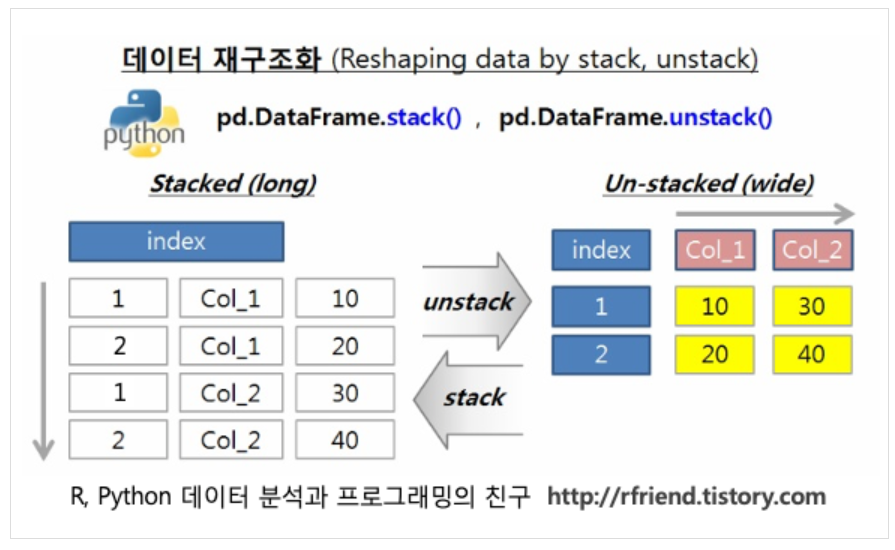
actor2

1. 데이터 재구조화(reshaping data)

1) pivot(), pd.pivot\_table()



2) stack(), unstack()



3) pd.melt() / data.melt()

테이블이(가) 표시된 사진

자동 생성된 설명

4) wide\_to\_long()

pd.wide\_to\_long(data, ["col\_prefix\_1", "col\_prefix\_2"], i="idx\_1", j="idx\_2")

wide\_to\_long()은 pivot() 이나 stack() 과는 다르게 "칼럼 이름의 앞부분"과 나머지 "칼럼 이름의 뒷부분"을 구분해서, 칼럼 이름의 앞부분을 칼럼 이름으로, 칼럼 이름의 나머지 뒷부분을 행(row)의 원소로 해서 세로로 길게(long~) 쌓아 줍니다.  말로 설명해주

10장

stubs = ['actor', 'actor\_facebook\_likes']

actor2\_tidy = pd.wide\_to\_long(actor2,

stubnames=stubs,

i=['movie\_title'],

j='actor\_num',

sep='\_')

테이블이(가) 표시된 사진

자동 생성된 설명

5) pd.crosstab()

테이블이(가) 표시된 사진

자동 생성된 설명

1. 10장 aggfunc이 없으면 맨 처음걸로 나오나? → default가 mean으로 설정되어 있음

margins=True 을 사용해서 행, 열별 총합을 구할 수 있음(All)

flights.pivot\_table(index='AIRLINE',

columns='ORG\_AIR',

values='CANCELLED',

aggfunc='sum')

1. 10장 인덱스나 컬럼의 상하위 레벨을 바꾼다.

.swaplevel(0,1,axis='columns')

1. 10장 정규표현식 .str.extract()

tidy2 = (weightlifting

.melt(id\_vars='Weight Category',

var\_name='sex\_age',

value\_name='Qual Total')

.assign(Sex=lambda df\_:df\_.sex\_age.str[0],

\*\*{'Age Group':(lambda df\_: (df\_

.sex\_age

.str.extract(r'(\d{2}[-+](?:\d{2})?)',

expand=False)))})

.drop(columns='sex\_age')

)

1. 10장 pivot에서 열 2개는 피벗되도록 개발되지 않았음

inspections.pivot(index=['Name', 'Date'],

columns='Info', values='Value')

1. 10장 reset\_index에서 col\_level에 값을 넣으면 컬럼 레벨을 조정할 수 있음

(inspections

.set\_index(['Name', 'Date', 'Info'])

.unstack('Info')

.reset\_index(col\_level=-1))

1. 10장 .droplevel()을 하면 해당 레벨의 컬럼이 없어짐

def flatten0(df\_):

df\_.columns = df\_.columns.droplevel(0).rename(None)

return df\_

1. 10장 np.squeeze(배열, 차원)은 배열에서 1차원인 축을 제거, np.squeeze(배열) 1차원인 축을 전부 제거

(inspections

.set\_index(['Name','Date', 'Info'])

.squeeze()

.unstack('Info')

.reset\_index()

.rename\_axis(None, axis='columns')

)

1. 10장 aggfunc='first' 는 값(values) 으로 사용하려는 변수가 집계가 불가능한 문자형 데이터이기 때문에 'DataError : No numeric types to aggregate' 에러가 발생하는데 그래서 디폴트인 'mean'을 사용해서 집계하는 것이 아니라 'first'를 사용해서 재구조화하는 기준의 테이블의 각 cell의 첫번째 값('first')을 그냥 가져오게끔 하는 것임

(inspections

.pivot\_table(index=['Name', 'Date'],

columns='Info',

values='Value',

aggfunc='first')

.reset\_index()

.rename\_axis(None, axis='columns')

)

1. 11장 np.concatenate, pd.concat(), merge(), join(), append()
   1. Concat
      1. Pandas 함수
      2. 둘 이상의 pandas 객체를 수직이나 수평으로 병합
      3. 인덱스에 대해서만 정렬
      4. 인덱스에 중복이 있으면 오류 발생
      5. 기본 설정은 외부 조인이고, 내부 조인은 옵션
   2. Join
      1. DataFrame 메서드
      2. 둘 이상의 pandas 객체를 수평으로 병합
      3. 호출 DataFrame의 열(들) 또는 인덱스를 다른 객체의 인덱스(열이 아님)에 대해 정렬
      4. 합쳐지는 열/인덱스에 중복 값이 있으면 카테션 곱을 생성
      5. 기본 설정은 좌측 조인이며, 내부, 외부, 우측 조인이 옵셕으로 있음
   3. Merge
      1. DataFrame 메서드
      2. 정확히 두 DataFrame을 수평으로 조인
      3. 호출 DataFrame의 열(들) 또는 인덱스를 다른 DataFrame의 열(들) 또는 인덱스와 정렬
      4. 합쳐지는 열/인덱스에 중복 값이 있으면 카티션 곱을 생성
      5. 기본 설정은 내부 조인이며, 좌측, 외부, 우측 조인이 옵션으로 있음

1. 11장 행이름 없이 딕셔너리를 올바르게 추가하려면 ignore\_index=True를 해야함

names.append({'Name':'Aria', 'Age':1}, ignore\_index=True)

1. 11장 isinstance()가 뭔가?

new\_data\_dict = {k: '' if isinstance(v, str) else

np.nan for k,v in data\_dict.items()}

1. 11장 다시 함 해보기

random\_data = []

for i in range(1000):

d = dict()

for k, v in data\_dict.items():

if isinstance(v, str):

d[k] = np.random.choice(list('abcde'))

else:

d[k] = np.random.randint(10)

random\_data.append(pd.Series(d, name=i + len(ball\_16)))

random\_data[0]

1. 11장 keys랑 names가 뭔가?

pd.concat(s\_list, join='outer', keys=['2016', '2017'],

axis='columns', names=['Year', None])

1. 11장 Dict와 zip을 같이 사용하면 어떻게 되는건가?

pd.concat(dict(zip(years, stock\_tables)), axis='columns')

1. 11장 lsuffx? rsuffix?

stocks\_2016.join(stocks\_2017, lsuffix='\_2016', rsuffix='\_2017', how='outer')

1. 11장

other = [stocks\_2017.add\_suffix('\_2017'),

stocks\_2018.add\_suffix('\_2018')]

stocks\_2016.add\_suffix('\_2016').join(other, how='outer')

1. 11장 get\_level\_values는 컬럼이 층으로 쌓였을때 해당 층을 가져온다.

stock\_concat.columns.get\_level\_values(1)

1. 11장 .merge() 함수에서 left\_index=True, right\_index=True로 설정하면 인덱스에 정렬하게 선택할 수 있음

stocks\_2016.merge(stocks\_2017, left\_index=True, right\_index=True)

1. 11장 .merge()에서 suffixes를 사용하면 stocks\_2016에 \_2016을 접미사로, stocks\_2017에 \_2017을 접미사로 붙여서 나오게 함

stocks\_2016.merge(stocks\_2017, left\_index=True, right\_index=True, how='outer',

suffixes=('\_2016', '\_2017'))

1. 11장 .merge() 에서 on을 사용하면 해당 컬럼의 동일한 값으로 합침

food\_transactions.merge(food\_prices, on=['item', 'store'])

1. 11장 .merge() 에서 .query를 사용하여 데이터를 추출함

food\_transactions.merge(food\_prices.query('Date == 2017'), how='left')

1. 11장 중복값이 있으면 아래와 같은 오류가 발생함

ValueError: cannot handle a non-unique multi-index!

1. 12장 파이썬에서 사용하는 시간의 종류

1) datetime.timedelta = datetime.timedelta

2) pd.Timestamp() = Timestamp

3) pd.to\_datetime() = Timestamp

4) pd.to\_datetime(pd.Series([10, 100]), unit='D') = datetime64[ns]

5) pd.to\_datetime(['Aug 3 1999 3:45:56', '10/31/2017'])

= DatetimeIndex(['1999-08-03 03:45:56', '2017-10-31 00:00:00'], dtype='datetime64[ns]', freq=None)

6) pd.Timedelta() = Timedelta

7) pd.to\_timedelta(pd.Series([10,100]), unit='s') = timedelta64[ns]

8) pd.to\_timedelta(['2 days 24 minutes 89.67 seconds', '00:45:23.6'])

= TimedeltaIndex(['2 days 00:25:29.670000', '0 days 00:45:23.600000'], dtype='timedelta64[ns]', freq=None)

9) ts.to\_pydatetime() =datetime.datetime



1. 12장 StringIO은 문자열을 파일처럼 사용됨 이게 뭔소리인가?

buf = StringIO()

1. 12장 to\_string()은 어떻게 되는 건가?

df.to\_string(buf=buf, max\_cols=numcols, max\_rows=rows, line\_width=width)

out = buf.getvalue()

1. 12장 endswith(끝나는 문자, 문자열의 시작, 문자열의 끝) 는 문자열이 특정문자로 끝나는지 여부를 알려줌 True나 False로 반환

line.strip().endswith('\\'):

1. 12장 startswith(시작하는문자, 시작시점) 특정문자로 시작하는지 여부를 알려줌 True나 False로 반환
2. 12장 find(찾을문자, 찾기시작할 위치) 문자열중에 특정문자를 찾고 위치를 반환, 없을경우 -1을 리턴
3. 12장 pd.Timestamp에서 unit='D'로하면 해당 숫자를 day로 인식함

pd.Timestamp(5000, unit='D')

1. 12장 pd.to\_datetime에서 dayfirst=True로 하면 연/일/월로 인식함,

pd.to\_datetime('2015-13-5', dayfirst=True, unit=D)

1. 12장 origin에 날짜를 입력하면 시작하는 날짜임

s = pd.Series([10, 100, 1000, 10000])

pd.to\_datetime(s, unit='D', origin='2013-1-1')

1. 12장 pd.to\_datetime에서 errors='coerce'를 하면 에러나는 행은 NaT로 바꿔줌

s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])

pd.to\_datetime(s, dayfirst=True, errors='coerce')

1. 12장 pd.Timestamp에서 .ceil('h') 을 하면 시간으로 올림을 함, .round('min') 하면 분으로 반올림을 함

ts = pd.Timestamp('2016-10-1 4:23:23.9')

ts.ceil('h')

ts.round('min')

1. ts.year, ts.month, ts.day, ts.hour, ts.minute, ts.second,

ts.dayofweek, ts.dayofyear, ts.daysinmonth

1. 12장 componets는 구성요소를 보여줌

td.components

1. 12장 .total\_seconds()는 모든 시간을 초로 계산함

td.total\_seconds()

1. 12장 loc과 sort\_index()를 사용하면 원하는 날짜를 인덱싱하고 정렬할 수 있음

crime.loc['2015-3-4 22':'2016-1-1 11:22:00'].sort\_index()

1. 12장 crime.memory\_usage()는 컬럼별로 메모리를 얼마나 사용하는지를 보여줌
2. 12장 원하는 컬럼에서 시간단위가 아닌 날짜단위로 인덱싱 하려면 .between()을 사용해야 함

(crime

[crime.REPORTED\_DATE == '2016-05-12 16:45:00']

) → 이렇게 딱 떨어지는 것만 ==으로 인덱싱할 수 있음

(crime

[crime.REPORTED\_DATE.dt.date == '2016-05-12']

) → 이렇게는 안됨

(crime

[crime.REPORTED\_DATE.between('2016-05-12', '2016-05-13')]

)

(crime

[crime.REPORTED\_DATE.between('2016', '2017')]

)

(crime

[crime.REPORTED\_DATE.between(

'21st October 2014 05', '21st October 2014 06')]

)

(crime

[crime.REPORTED\_DATE.between(

'2015-3-4 22','2016-1-1 23:59:59')]

)

lmask = crime.REPORTED\_DATE >= '2015-3-4 22'

rmask = crime.REPORTED\_DATE <= '2016-1-1 11:22:00'

crime[lmask & rmask]

1. 12장 .between\_time()을 써서 날짜와 상관없이 시간사이의 데이터를 찾을 수 있으며, include\_end=False를 사용해 끝의 숫자를 포함을 안하는건가??

crime.between\_time('2:00', '5:00', include\_end=False)

crime.between\_time(datetime.time(2,0), datetime.time(5,0), include\_end=False)

1. 12장 .at\_time()을 사용해 정확한 데이터를 찾을 수 있음

crime.at\_time('5:47')

1. 12장 .first()는 시간에 대해 첫 n 세그먼트(처음부터 원하는 날짜까지)를 선택할 수 있음

crime\_sort.first(pd.offsets.MonthBegin(6))

1. 12장 pd.offsets()은 ???한 메서드 이고, 이 메서드를 사용하려면 정렬이 된 상태에서 사용할 수 있음

crime\_sort.first(pd.offsets.MonthBegin(6)) → 맨마지막에 2012-07-01 00:01:00 7월 첫째날 데이터 하나가 나옴

crime\_sort.first(pd.offsets.MonthEnd(6)) → 맨마지막에 2012-06-30 00:04:00 6월 마지막날 데이터 하나가 나옴

<https://tariat.tistory.com/631>

1. 12장 pd.offsets()에서 normalize 매개변수를 True로 설정하면 모든 시간 구성 요소가 0으로 됨(2012-07-01 00:00:00으로 이동 함)

crime\_sort.first(pd.offsets.MonthBegin(6, normalize=True)) → 맨마지막에 2012-06-30 23:54:00 6월 마지막날 데이터 하나가 나옴

crime\_sort.loc[:'2012-06'] → 이렇게 사용하면 위에꺼랑 같은 결과를 출력할 수 있음

1. 12장 .first('5D') 같이 인수 alias를 사용할 수 있음

<http://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset-aliases>

3QS = 3분기 시작, A = 1년 끝

1. 12장 원하는 날짜나 시간을 늘릴때 아래처럼 더해서 사용할 수도 있음

crime\_sort.first(pd.offsets.MonthEnd(6)) == crime\_sort.index[0] + pd.offsets.MonthEnd(6)

dt = pd.Timestamp('2012-1-16 13:40')

dt + pd.DateOffset(months=1)

do = pd.DateOffset(years=2, months=5, days=3, hours=8, seconds=10)

pd.Timestamp('2012-1-22 03:22') + do

1. 12장 .resample() 는 DateOffset 객체나 별칭을 사용해 모든 그룹에서 조치를 수행할 준비가 된 객체임

(crime\_sort

.resample('W', on = 'REPORTED\_DATE')

.size()

)

1. 12장 fig.savefig()를 사용해서 그래프 그림을 저장할 수 있음

import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(16,4))

weekly\_crimes.plot(title='All Denver Crimes', ax=ax)

fig.savefig('../c12-crimes.png', dpi=300)

1. 12장 아래처럼도 사용할 수 있음

(crime

.loc['2012-4-1':'2012-6-30', ['IS\_CRIME', 'IS\_TRAFFIC']]

.sum()

)

(crime

.groupby(pd.Grouper(freq='Q'))

['IS\_CRIME', 'IS\_TRAFFIC']

.sum()

)

1. 12장 .dt.dayofyear 에서 dt가 datetime.dt 인가? → import datetime 이 아니라 그냥 파이썬이나 판다스에 있는 기능인것 같음

criteria = crime['REPORTED\_DATE'].dt.year == 2017

crime.loc[criteria, 'REPORTED\_DATE'].dt.dayofyear.max()

1. 12장 .weekday\_name이 DatetimeProperties의 속성에 없다고 나옴 왜?

(crime

.groupby([crime['REPORTED\_DATE'].dt.year.rename('year'),

crime['REPORTED\_DATE'].dt.weekday\_name.rename('day')])

.size()

1. 12장 .sub() 는 subtract의 약자로 빼기를 의미함
2. 비교연산

테이블이(가) 표시된 사진

자동 생성된 설명

1. reindex(), div() 같은 경우 아래처럼 사용할 수 있음

.reindex(columns=days)

.div(den\_100k, axis='index')

1. 12장 pd.read\_csv()할 경우 parse\_dates=[]를 사용하면 해당 컬럼을 datetime64[ns]으로 파일 형식을 바꿈
2. 12장 .groupby()를 사용하면 원래 있던 인덱스가 없어짐. 여러개를 그룹할땐 .groupby()를 먼저 사용하여 .resample()을 사용해야 함

(employee

.groupby('GENDER')

['BASE\_SALARY']

.mean()

.round(-2)

)

(employee

.groupby('GENDER')

.resample('10AS')

['BASE\_SALARY']

.mean()

.round(-2)

)

1. 12장 groupby에서 날짜를 그룹하려면 pd.Grouper()를 사용해야 함

(employee

.groupby(['GENDER', pd.Grouper(freq='10AS')])

['BASE\_SALARY']

.mean()

.round(-2)

.unstack('GENDER')

)

1. 12장 pd.cut()에서 precition은 몇번째 소수점 자리를 기준으로 할지 정함
2. Json 일기

Import json

From pandas.io.json import json\_normalize

1. Json\_normalize(Json.loads(~~~))
2. Raw\_df['column'].str.contains('찾을|문자', na=True, case=False)

찾을 문자랑 완전히 일치하는 값이 아닌 부분 일치하는 값을 찾음

na로 null값을 True로 할 것인지 False로 할 것인지 정할 수 있음

case로 대소문자 구별여부를 지정할 수 있음

| or로 문자열을 찾고 싶을때 사용

1. list.append() 는 리스트 자체를 넣는 것이고, list.expend()는 원소를 넣는 것임
2. raw\_process\_df.groupby('등록번호').last()['activity'].reset\_index().groupby('activity').count()
3. Timestamp 시간차 구해서 초로 바꾸기

1) consult\_time

→ 10203 2021-06-13 22:56:00

2) consult\_time.values[0]

→ numpy.datetime64('2021-06-13T22:59:00.000000000')

3) lead\_time = first\_contect\_time.values[0] - consult\_time.values[0]

lead\_time

→ numpy.timedelta64(180000000000,'ns')

4) pd.Timedelta(lead\_time, 'm') / pd.Timedelta(lead\_time) / pd.to\_timedelta(lead\_time)

→ Timedelta('0 days 00:03:00')

5) pd.Timedelta(lead\_time).total\_seconds()

→ 180.0

1. Timestamp 시간차 구해서 값으로 뽑기

1) lead\_time = first\_contect\_time.values[0] - consult\_time.values[0]

lead\_time

→ numpy.timedelta64(180000000000,'ns')

2) pd.Timedelta(lead\_time)

→ Timedelta('0 days 00:03:00')

3) pd.Timedelta(lead\_time).components

→ Components(days=0, hours=0, minutes=3, seconds=0, milliseconds=0, microseconds=0, nanoseconds=0)

4) pd.Timedelta(lead\_time).components[2]

→ 3

1. 시계열 데이터 분리(time series cross-validation) <https://gmnam.tistory.com/230>

from sklearn.model\_selection import TimeSeriesSplit

from matplotlib.patches import Patch

import matplotlib.pyplot as plt

XX = np.arange(100)

n\_split = 6

tscv = TimeSeriesSplit(n\_splits=n\_split)

plot\_cv\_indices(tscv, XX, n\_splits=n\_split)

1. 시계열 데이터 분리(blocked cross-validation) <https://gmnam.tistory.com/230>

class BlockingTimeSeriesSplit():

def \_\_init\_\_(self, n\_splits):

self.n\_splits = n\_splits

def get\_n\_splits(self, groups):

return self.n\_splits

def split(self, X, y=None, groups=None):

n\_samples = len(X)

k\_fold\_size = n\_samples // self.n\_splits

indices = np.arange(n\_samples)

margin = 0

for i in range(self.n\_splits):

start = i \* k\_fold\_size

stop = start + k\_fold\_size

mid = int(0.8 \* (stop - start)) + start

yield indices[start: mid], indices[mid + margin: stop]

btscv = BlockingTimeSeriesSplit(n\_splits=n\_split)

plot\_cv\_indices(btscv, XX, n\_splits=n\_split)

1. D
2. D