피벗 테이블

피벗 테이블(pivot table)은 표 형태의 데이터로 작업하는 스프레이시트와 다른 프로그램에서 일반적으로 볼 수 있는 유사한 작업이다. 피벗 테이블은 입력 값으로 간단한 열 단위의 데이터를 취하고 그 데이터에 대한 다차원 요약을 제공하는 2차원 테이블로 항목을 그룹핑한다. 피벗 테이블과 GroupBy의 차이가 때때로 혼란을 일으킬 수도 있다.
피벗 테이블을 근본적으로 GroupBy집계의 다차원 버전이라고 생각하면 도움이 된다. 분할-적용-결합 작업을 하면 분할과 결합 작업이 1차원 인덱스에서 발생하는 것이 아니라 2차원 그리드에서 발생한다.

피벗 테이블(Pivot Tables) 시작¶

이번 절의 예제에서는 Seaborn라이브러리에서 제공하는 타이타닉 승객의 데이터베이스를 사용할 것이다.

```
In [1]:import numpy as np
        import pandas as pd
        import seaborn as sns
        titanic = sns.load_dataset('titanic')
In [2]:titanic.head()
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_m
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

이 데이터에는 불행하게 끝났던 여행의 각 승객에 대한 성별, 나이, 좌석 등급, 요금 등 다양한 정보가 담겨 있다.

피벗 테이블(Pivot Tables)의 등장 배경

이 데이터에 대해 더 알아보기 위해 성별이나 생존 여부, 또는 그 몇 가지 조합에 따라 분류하는 일부터 해보자.
일단, GroupBy연산을 적용하여 성별에 따른 생존율을 알아보자

```
In [3]:titanic.groupby('sex')[['survived']].mean()
```

Out[3]:

Sex	survived
Female	0.742038
Male	0.188908

이 결과는 몇가지 사실을 알려준다. 전반전으로 승선하고 있던 여성의 네 명 중 세명이 생존한 반면, 남성은 다섯 명 중 한명명 생존했다.

이 정보도 유용하지만, 한 단계 더 들어가서 성별과 좌석 등급별 생존율을 보고 싶을 수도 있다. GroupBy를 이용해 다음과 같이 진행할 수도 있다. 좌석 등급과 성별 단위로 그룹을 나누고 생존율을 선택하고 평균 집계를 적용하고 결과 그룹을 결합한 후 숨겨진 다 차원성을 드러내기 위해 계층적인 인덱스를 분할한다.

```
In [4]:titanic.groupby(['sex', 'class'])['survived'].aggregate('mean').unstack()
```

Out[4]:

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

이 코드는 성별과 좌석 등급이 생존율에 얼마나 영향을 미치는지 더 잘 이해할 수 있게 해주지만 약간 복잡해지기 시작했다. 이 과정의 각 단계가 앞에서 논의했던 도구 관점에서는 타당해 보이지만, 이렇게 긴 코드를 읽거나 사용하기는 쉽지 않다. 이 2차원 GroupBy는 아주 보편적으로 사용되고 있어서 Pandas는 이러한 유형의 다차원 집계를 간결하게 할 수 있도록 pivot_table이라는 루틴을 제공한다.

피벗 테이블(Pivot Table)구문

이번에는 DataFrame의 pivot_table 메서드를 사용해 앞의 연산을 동일하게 구현해보자.

```
In [5]:titanic.pivot_table('survived', index='sex', columns='class')
```

Out[5]:

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

이 코드는 GroupBy를 사용한 코드보다 훨씬 더 읽기 쉬우면서도 똑같은 결과를 만들어낸다. 20세기 초 대서양 횡단 크루즈를 떠올리면 예상할 수 있는 것처럼 여성이면서 좌석 등급이 높을수록 생존율이 높아지는 경향이 있다. 1등석에 탄 여성은 거의 확실히 생존했지만, 3등석에 탄 남성은 열명 중 한 명만 생존했다.

다 차원 피벗 테이블(pivot tables)

GroupBy에서와 마찬가지로 피벗 테이블의 그룹핑은 다단계로 여러 가지 옵션을 통해 지정할 수 있다. 세번째 차원으로 연령을 보고 싶을 수 있다. pd.cut함수를 사용해 연령 을 추가하자.

```
In [6]:age = pd.cut(titanic['age'], [0, 18, 80])
       titanic.pivot_table('survived', ['sex', age], 'class')
```

Out[6]:

	class	First	Second	Third

sex	age			
female	(0, 18]	0.909091	1.000000	0.511628
	(18, 80]	0.972973	0.900000	0.423729
male	(0, 18]	0.800000	0.600000	0.215686
	(18, 80]	0.375000	0.071429	0.133663

열 기준으로 동작할 때도 이와 같은 전략을 적용할 수 있다. 자동으로 분위 수를 계산하기 위해 pd.qcut를 사용해 지불된 내용에 대한 정보를 추가하자.

```
In [7]:fare = pd.qcut(titanic['fare'], 2)
       titanic.pivot_table('survived', ['sex', age], [fare, 'class'])
```

Out[7]:

	fare	[0, 14.454]			(14.454, 512.329]		
	class	First	Second	Third	First	Second	Third
sex	age						
female	(0, 18]	NaN	1.000000	0.714286	0.909091	1.000000	0.318182
	(18, 80]	NaN	0.880000	0.444444	0.972973	0.914286	0.391304
male	(0, 18]	NaN	0.000000	0.260870	0.800000	0.818182	0.178571
	(18, 80]	0.0	0.098039	0.125000	0.391304	0.030303	0.192308

결과는 값 사이의 관계를 보여주는 그리드에 나타난 계층적 인덱스를 가진 4차원 집계다.

기타(pivot table)옵션

DataFrame의 pivot_table메서드의 전체 호출 시그니처는 다음과 같다.

```
# Pandas 0.19버전기준 호출 시그너처
DataFrame.pivot_table(data, values=None, index=None, columns=None,
                      aggfunc='mean', fill_value=None, margins=False,
                      dropna=True, margins_name='All')
```

aggfunc키워드는 어떤 유형의 집계를 적용할지 제어하며 기본으로는 평균이 작용된다. GroupBy에서와 마찬가지로 적용할 집계 연산은 몇 가지 일반적인 방식 중 하나를 표현하는 문자열('sum','mean','count','min','max' 등)이나 집계를 구현하는 함수(np.sum(), min(), sum() 등)로 지정할 수 있다. 아울러 열을 원하는 집계 방식에 매핑한 딕셔너리로 지정할 수 있다.

```
In [8]:titanic.pivot_table(index='sex', columns='class',
                           aggfunc={'survived':sum, 'fare':'mean'})
```

Out[8]:

	fare			survived		
Class	First	Second	Third	First	Second	Third

Sex						
female	106.125798	21.970121	16.118810	91.0	70.0	72.0
Male	67.226127	19.741782	12.661633	45.0	17.0	47.0

여기서 values키워드는 생략했는데 aggfunc을 위한 매핑을 지정할 때 이 키워드가 자동으로 결정된다는 점도 알아두자.

때로는 그룹별 총합을 계산하는 것이 유용할 수 있다. 이 계산은 margins키워드를 통해 수행할 수 있다.

In [9]:titanic.pivot_table('survived', index='sex', columns='class', margins=**True**)

Out[9]:

class	First	Second	Third	All
sex				
female	0.968085	0.921053	0.500000	0.742038
male	0.368852	0.157407	0.135447	0.188908
All	0.629630	0.472826	0.242363	0.383838

그 결과 좌석 등급과 무관한 성별에 따른 생존율과 성별과 무관한 등급별 생존율, 그리고 38%라는 전체 생존율을 알 수 있다. 이 가장자리 열과 행의 명칭은 margins_name키워드로 지정할 수 있으며 지정하지 않으면 기본적으로 'All'로 표시한다.

예제: 출생률 데이터

더 흥미로운 예제로 질병 대책 본부(Centers for Disease Control (CDC))에서 제공하고 자유롭게 사용할 수 있는 미국의 출생률 데이터를 살펴보자. 이 데이터는 다음 URL에서 내려 받을 수 있다.

https://raw.githubusercontent.com/jakevdp/data-CDCbirths/master/births.csv

참고로 이 데이터세트는 앤드류 겔만(Andrew Gelman)과 그의 그룹이 광범위하게 분석했다.

In [10]:

```
# 데이터 다운로드를 위한 셀 명령어:
# !curl -O https://raw.githubusercontent.com/jakevdp/data-CDCbirths/master/births.csv
```

In [11]:births = pd.read_csv('data/births.csv')

데이터를 들여다보면 비교적 간단하다는 사실을 알 수 있다. 이 데이터는 날짜와 성별로 분류한 출생수를 담고 있다.

In [12]:births.head()

Out[12]:

	year	month	day	gender	births
0	1969	1	1	F	4046
1	1969	1	1	M	4440
2	1969	1	2	F	4454

3	1969	1	2	M	4548
4	1969	1	3	F	4548

피벗 테이블을 사용해 이 데이터를 좀 더 잘 이해할 수 있다. 그러면 decade 함수로 연대 열을 추가하고 연대별 남녀의 출생 수를 살펴본다.

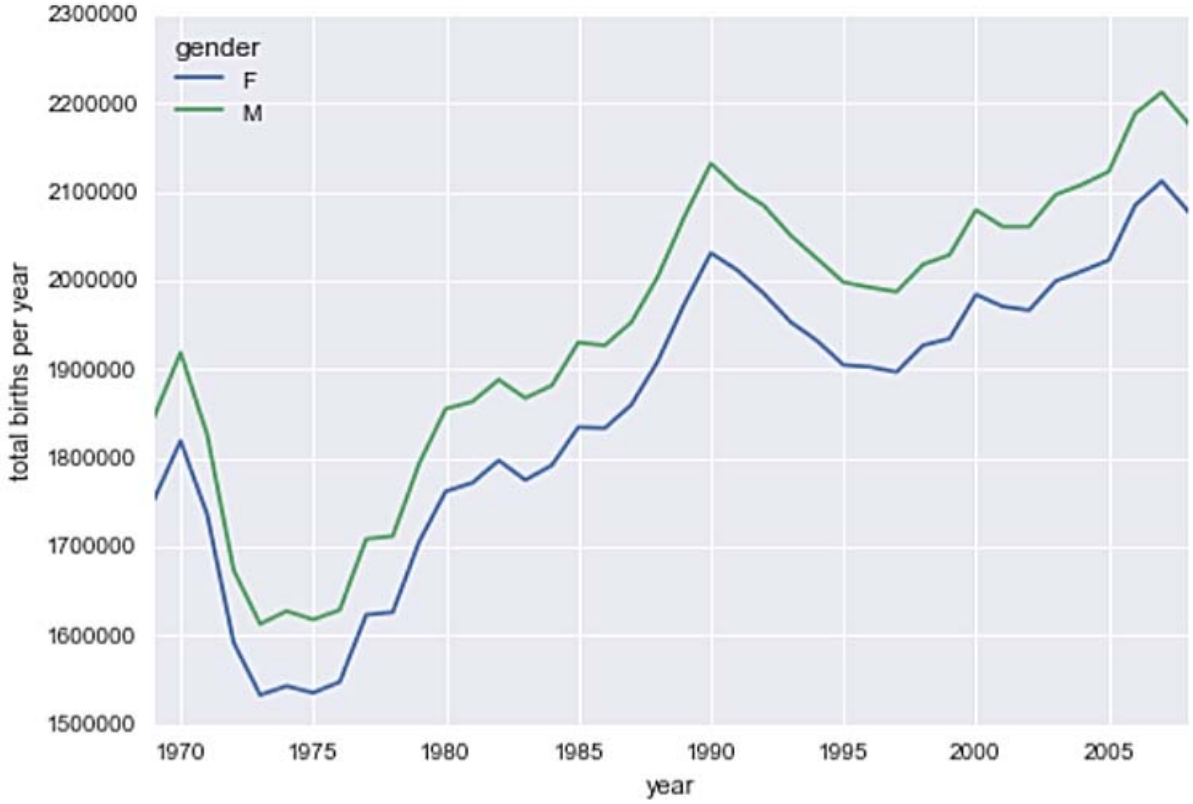
```
In [13]:births['decade'] = 10 * (births['year'] // 10)
        births.pivot_table('births', index='decade', columns='gender', aggfunc='sum')
```

Out[13]:

gender	F	M
decade		
1960	1753634	1846572
1970	16263075	17121550
1980	18310351	19243452
1990	19479454	20420553
2000	18229309	19106428

연대마다 항상 남성의 출생률이 여성보다 높았음을 바로 알 수 있다. 이 추세로 더 분명하게 확인하려면 Pandas에서 기본적으로 제공하는 플로팅 도구를 활용해 연대별 수를 시각화하면 된다.

```
In [14]:%matplotlib inline
        import matplotlib.pyplot as plt
        sns.set()   # use Seaborn styles
        births.pivot_table('births', index='year', columns='gender', aggfunc='sum').plot()
        plt.ylabel('total births per year');
```



간단한 피벗 테이블과 plot()메서드로 성별에 따른 연도별 출생 수의 추이를 바로 확인할 수 있다. 눈대중으로 보더라도 지난 50년 동안 남성의 출생 수가 여성보다 5% 정도 더 높았음을 알 수 있다.

추가 데이터 탐색

피벗 테이블과 꼭 상관있는 것은 아니지만 지금까지 다룬 Pandas도구를 사용해 이 데이터세트에서 뽑아낼 수 있는 몇 가지 흥미로운 특징이 있다. 먼저 데이터를 정제해야 하는데, 날짜 오타(예:June 31st)나 누락된 값(예: June 99th)으로 인한 이상치를 제거해야 한다. 이러한 데이터를 한번에 제거하는 쉬운 방법은 이상치를 제거하는 제거해야 한다. 이 작업을 견고한 시그마 클리핑 연산을 통해 할 것이다.

```
In [15]:quartiles = np.percentile(births['births'], [25, 50, 75])
         mu = quartiles[1]
         sig = 0.74 * (quartiles[2] - quartiles[0])
```

마지막 줄은 표본 평균의 견고한 추정치로, 0.74는 가우스 분포의 사분위 수에서 비롯한 것이다. 이와 함께 query() 메서드를 사용해 이 값에서 벗어난 출생수를 가진 행을 걸러낼 수 있다.

```
In [16]:births = births.query('(births > @mu - 5 * @sig) & (births < @mu + 5 * @sig)')
```

다음으로 day열을 정수형으로 설정한다. 이전에는 데이터세트의 일부 열이 'null'값을 포함했기 때문에 데이터타입이 문자열이었다.:

```
In [17]:# 'day'열을 정수형으로 설정 : 원래는 널 값 때문에 문자열이었음.
         births['day'] = births['day'].astype(int)
```

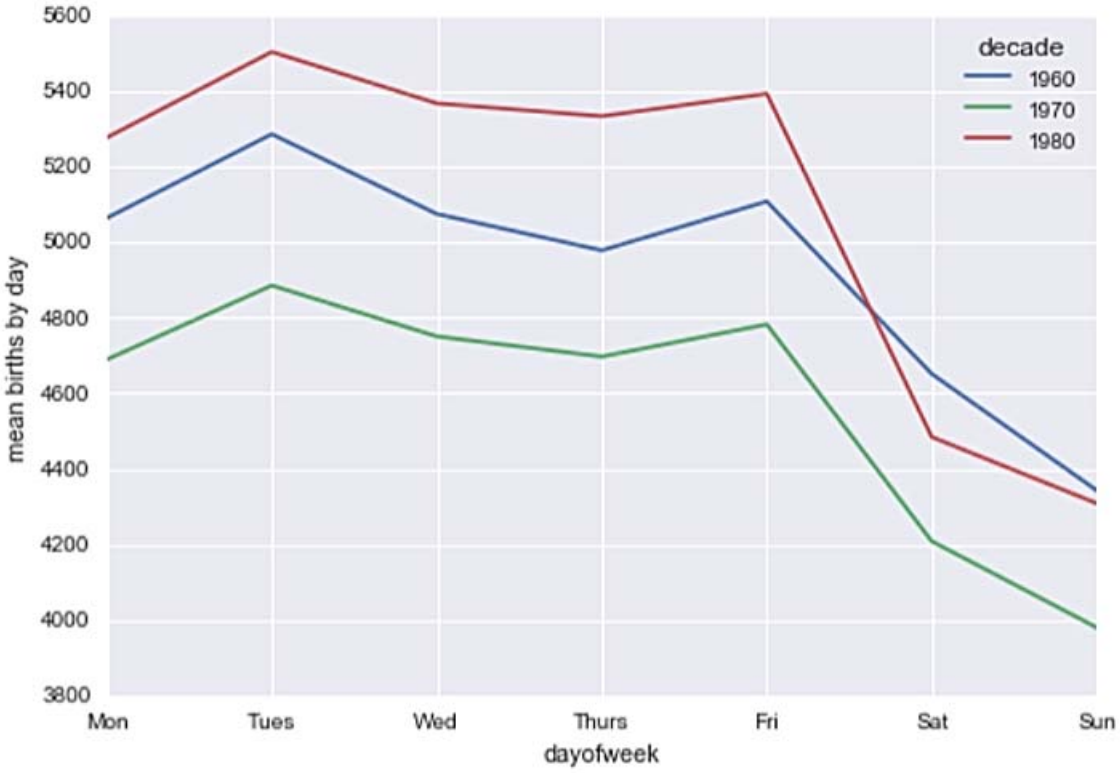
마지막으로 날짜, 월, 연도를 결합해 Date인덱스를 생성할 수 있다. 이렇게 하면 각 행에 대응하는 요일을 빠르게 계산할 수 있다.

```
In [18]:# 년 월 일로 부터 날짜 인덱싱 생성
         births.index = pd.to_datetime(10000 * births.year +
                                       100 * births.month +
                                      births.day, format='%Y%m%d')

         births['dayofweek'] = births.index.dayofweek
```

이 인덱스를 이용해 수십 년동안의 요일별 출생 수를 그래프로 그릴 수 있다.

```
In [19]:import matplotlib.pyplot as plt
         import matplotlib as mpl
         births.pivot_table('births', index='dayofweek',
                            columns='decade', aggfunc='mean').plot()
         plt.gca().set_xticklabels(['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun'])
         plt.ylabel('mean births by day');
```



출생 수가 주중보다 주말에 약간 더 낮은 것을 확인할 수 있다. 1990년대와 2000년대는 없는데, CDC데이터가 1989년을 기점으로 태어난 년월만 있기 때문이다.

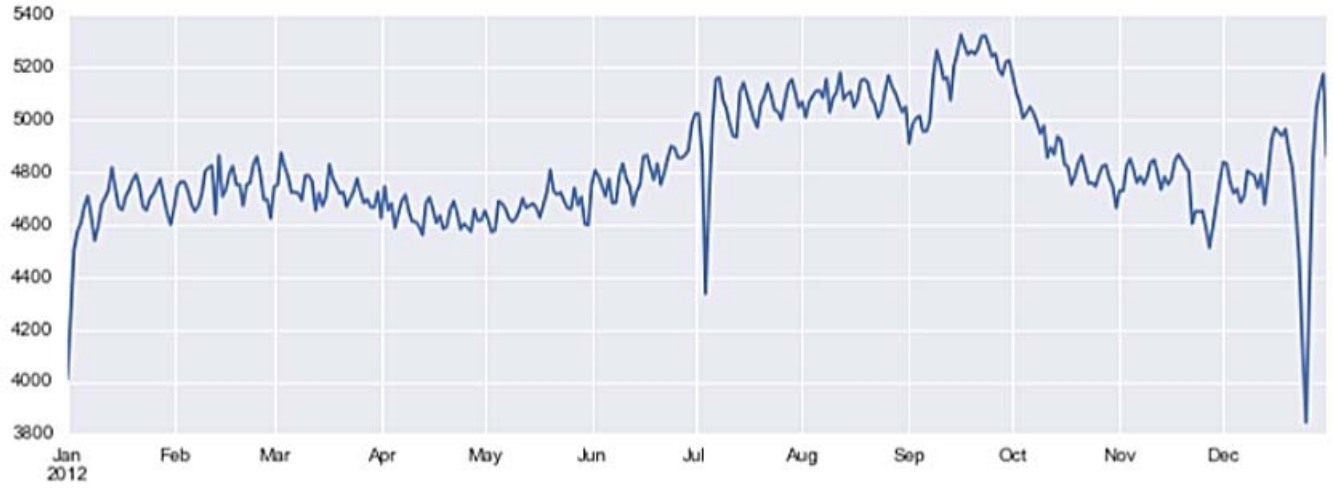
```
In [20]:births_by_date = births.pivot_table('births',
                                            [births.index.month, births.index.day])
         births_by_date.head()
Out[20]:1  1    4009.225
           2    4247.400
           3    4500.900
           4    4571.350
           5    4603.625
         Name: births, dtype: float64
```

결과로 월과 일을 인덱스로 가지는 다중 인덱스를 얻게 된다. 그래프로 쉽게 표현하기 위해 이 월과 일을 더미 연도 변수와 결하해 연월일의 날짜로 변환해 보자.(2월 29일을 올바르게 처리하도록 윤년을 선택해야 한다.

```
In [21]:births_by_date.index = [pd.datetime(2012, month, day)
                                for (month, day) in births_by_date.index]
         births_by_date.head()
Out[21]:2012-01-01    4009.225
         2012-01-02    4247.400
         2012-01-03    4500.900
         2012-01-04    4571.350
         2012-01-05    4603.625
         Name: births, dtype: float64
```

월과 일에만 초점을 맞춰 이제 해당 연도의 날짜별 평균 출생 수를 나타내는 시계열 데이터가 만들어졌다. 이로부터 데이터를 플로팅하기 위해 plot메서드를 사용할 수 있다.

```
In [22]:# 결과 그래프 그리기
         fig, ax = plt.subplots(figsize=(12, 4))
         births_by_date.plot(ax=ax);
```



특히 이 그래프에서 확인할 수 있는 놀라운 점은 미국의 휴일(예,독립 기념일, 노동절, 추수감사절, 크리스마스, 새해)에는 출생률이 급감했다는 것이다, 이 현상은 자연 분만에 대한 깊은 정신적, 신체적 효과라기보다는 예정 분만 및 유도 분만을 선호했던 경향이 반영된 것으로 보인다. !