동국대학교
dongguk university

데이터 처리/분석-2

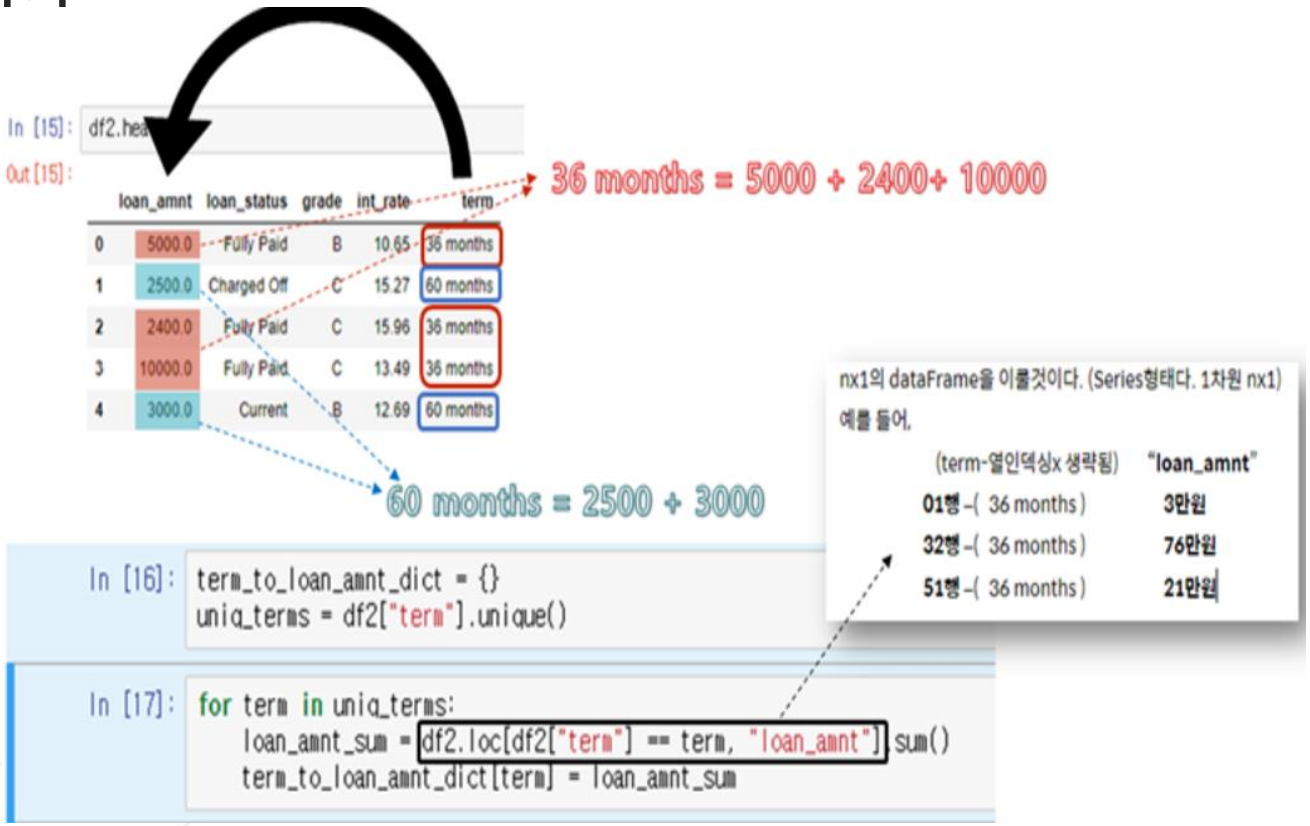
엄진영

파일 읽고 기본 분석하기

- 파일 읽기(.read_csv)→ shape/head → .columns 확인 → 필요한 열만 뽑아낸 df 만들기 → head/tail → 각 열의 데이터 타입 확인 → 문자열 데이터 범주 확인 → 결측값(NaN) 제거 후 shape 확인

파일 읽고 기본 분석하기

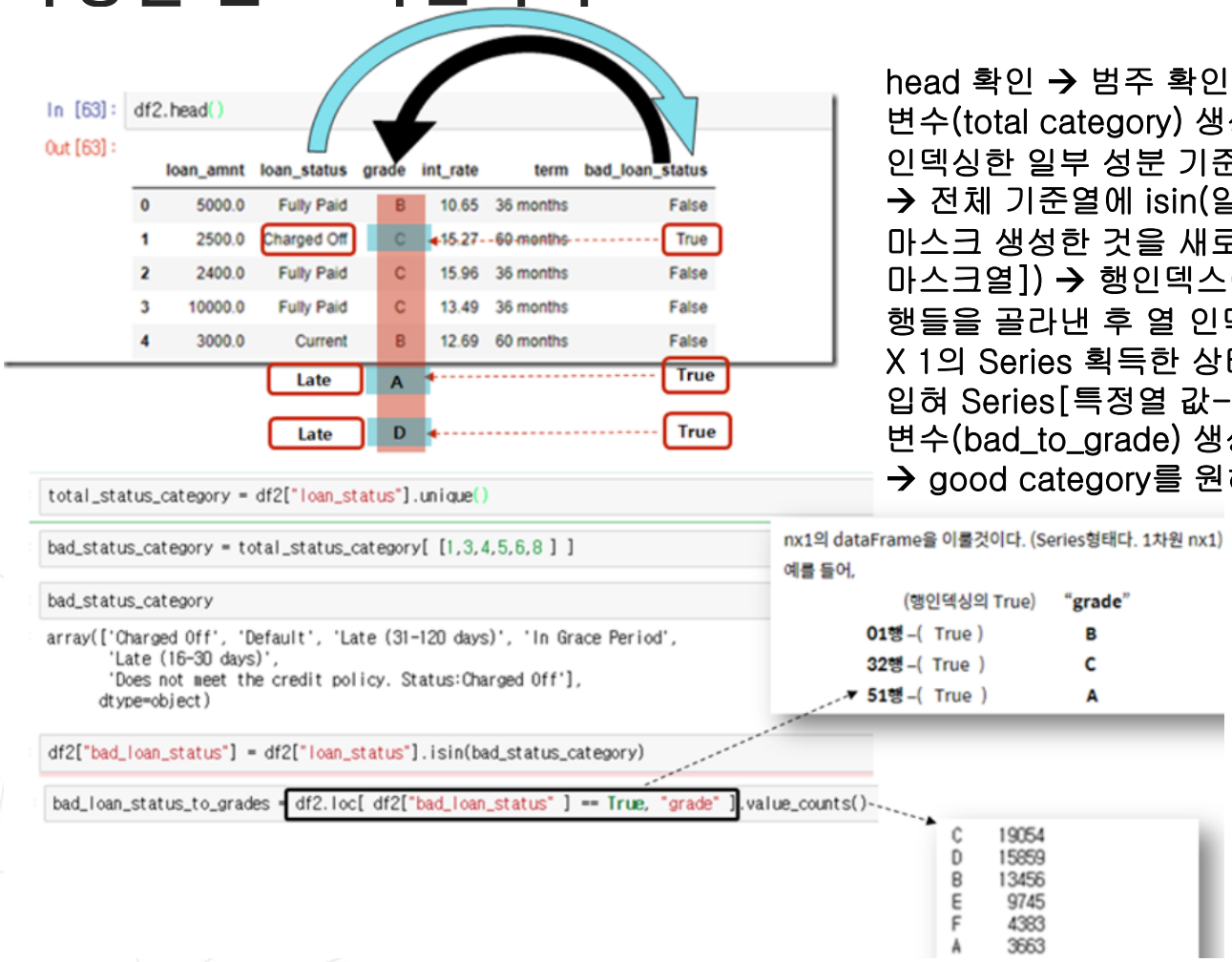
- 기준열에 특정 성분에 따른(term-2가지 문자열) 특정합(loan_anmt 총합) 파악하기



빈 딕셔너리{} 생성
→ 중복 없는 기준열 변수 생성
→ for문안에서, 마스크로 중복 없는 기준열의 각 성분에 따른 특정열만 뽑고, 가상으로 만들어진 n X 1의 총합 계산 후 변수에 담기
→ 각 성분을 딕셔너리[열이름]으로 인덱싱하여 딕셔너리의 키값으로 생성 = 우향에 키 값에 대한 총합 변수 대입으로 value 생성
→ 생성된 {key1:value1, key2:value2…}으로 구성된 1 X n 1차원 딕셔너리를 1차원인 Series에 담기

파일 읽고 기본 분석하기

- 기준열의 일부 성분들을 가진 행들을 골라내어 그 행들의 특정열 분포 확인하기



head 확인 → 범주 확인 후 중복 없는 기준열 변수(total category) 생성 → 일부 성분만 index로 열 인덱싱한 일부 성분 기준열 변수 생성(bad category) → 전체 기준열에 isin(일부 성분 기준열)을 통해, 마스크 생성한 것을 새로운 열로 추가(df[새로운 마스크열]) → 행인덱스에 마스크 == True를 통해 행들을 골라낸 후 열 인덱스에 특정열만 골라내어, n X 1의 Series 획득한 상태에서 .value.counts()를 입혀 Series[특정열 값-행 수]의 변수(bad_to_grade) 생성 → Series를 오름차순 정렬 → good category를 원하면 마스크 == False로 구함

그룹화하기

In [1]:
```
import numpy as np
import pandas as pd

df = pd.DataFrame({'key1':['a','a','b','b','a'],
                   'key2':['one','two','one','two','one'],
                   'data1':np.random.randn(5),
                   'data2':np.random.randn(5)})
df
```

Out[1]:

	key1	key2	data1	data2
0	a	one	0.084204	0.255810
1	a	two	-0.519531	0.141098
2	b	one	-2.047994	0.238742
3	b	two	-0.770977	0.147164
4	a	one	1.276054	-0.947729

key1과 key2에는 중복된 값이 있음
key1열이 a 인 행들에 대해 data1의 평균은?

1) key1의 유일한 값으로 마스크 후 반복문(df.loc(마스크, data1열))을 통해 평균 계산
2) 데이터 그룹화를 통해 계산

- groupby()
  - 통계량을 계산할 열(Series).groupby(기준이 될 열(Series)) 형식으로 실행

In [3]:
```
grouped = df['data1'].groupby(df['key1'])
grouped
```

Out[3]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x000001C66BAB1BE0>
```

값을 확인할 수 없는 이유는 key1열을 기준으로 data1열을 그룹화만 했을 뿐 통계함수를 사용하지 않았기 때문
(groupe에는 key1의 값인 a를 포함하는 행과 b를 포함하는 행을 각각 그룹화하여 동시에 가지고 있음

그룹화하기

- key1열을 데이터 그룹화한 다음 data1열의 통계량을 계산

```
In [4]: grouped.mean()

Out[4]: key1
        a     0.280242
        b    -1.409485
        Name: data1, dtype: float64
```

- **groupby() 함수 절차를 [split – apply – combine]이라고 함**
  - 기준열을 지정하여 특정열을 그룹으로 나누고 – 각 그룹에 통계함수를 적용하고–최종적인 통계량이 산출된 것은 통합하여 표시

그룹화하기

- **그룹화의 기준이 될 열을 2개 이상 지정**
  - 2개 열의 성분이 모두 같은 것만 하나의 그룹이 됨
  - 그룹화의 기준열이 2개이면, 계층적 인덱스로 적용된 Series가 나옴

	key1	key2	data1	data2
0	a	one	0.084204	0.255810
1	a	two	-0.519531	0.141098
2	b	one	-2.047994	0.238742
3	b	two	-0.770977	0.147164
4	a	one	1.276054	-0.947729

```
In [8]: means = df['data1'].groupby([df['key1'],df['key2']]).mean()
        means
```

```
Out[8]: key1  key2
        a     one     0.680129
              two    -0.519531
        b     one    -2.047994
              two    -0.770977
        Name: data1, dtype: float64
```

```
In [9]: means.unstack()
```

Out[9]:

key2	one	two
key1		
a	0.680129	-0.519531
b	-2.047994	-0.770977

DataFrame을 데이터 그룹화하기

- **Series의 데이터 그룹화**
  - 특정열 인덱싱.groupby(기준열 인덱싱)
- **DataFrame의 데이터 그룹화**
  - df.groupby("기준이 될 컬럼명")
  - 특정열이 아닌 df의 모든 열에 대해 통계량이 계산

	key1	key2	data1	data2
0	a	one	0.084204	0.255810
1	a	two	-0.519531	0.141098
2	b	one	-2.047994	0.238742
3	b	two	-0.770977	0.147164
4	a	one	1.276054	-0.947729

```
In [10]: df.groupby('key1').mean()
```

Out [10]:

key1	data1	data2
a	0.280242	-0.183607
b	-1.409485	0.192953

```
In [11]: df.groupby('key1').count()
```

Out [11]:

key1	key2	data1	data2
a	3	3	3
b	2	2	2

DataFrame을 데이터 그룹화하기


```
In [12]: df.groupby(['key1','key2']).mean()
```

Out[12]:

		data1	data2
key1	key2		
a	one	0.680129	-0.345960
	two	-0.519531	0.141098
b	one	-2.047994	0.238742
	two	-0.770977	0.147164

- DataFrame을 데이터 그룹화해서 특정 열 X 전체 열을 그룹화하더라도, 특정열에 대한 통계량만 산출 가능
- 통계함수를 적용하기 전에 [‘컬럼명’]으로 추출

```
In [13]: df.groupby(['key1','key2'])['data2'].mean()
```

```
Out[13]: key1  key2
         a     one    -0.345960
               two     0.141098
         b     one     0.238742
               two     0.147164
         Name: data2, dtype: float64
```

반복문을 이용하여 그룹화 직후의 결과물 확인해보기

- 그룹화를 수행한 직후의 결과물을 확인할 수 없었으나 반복문을 통해 그룹화에 대한 결과물 확인이 가능

```
In [14]: df.groupby('key1')
Out[14]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001C669742518>
```

- key라는 열에 대해 name, group변수로 매 반복문마다 값을 받아서 각각을 print 해보자.

```
In [15]: for name, group in df.groupby('key1') :
             print(name)
             print(group)
```

```
a
  key1 key2     data1     data2
0    a  one  0.084204  0.255810
1    a  two -0.519531  0.141098
4    a  one  1.276054 -0.947729
b
  key1 key2     data1     data2
2    b  one -2.047994  0.238742
3    b  two -0.770977  0.147164
```

```
In [20]: for name, group in df.groupby('key1') :
             print(name)
```

```
a
b
```

```
In [21]: for name, group in df.groupby('key1') :
          #   print(name)
             print(group)
```

```
  key1 key2     data1     data2
0    a  one  0.084204  0.255810
1    a  two -0.519531  0.141098
4    a  one  1.276054 -0.947729
  key1 key2     data1     data2
2    b  one -2.047994  0.238742
3    b  two -0.770977  0.147164
```

반복문을 이용하여 그룹화 직후의 결과물 확인해보기

- 기준열이 2개라면, name변수 자리에 소괄호()를 이용해서 2개의 변수에 각 기준열의 성분을 받음

In [17]:
```
for (k1, k2), group in df.groupby(['key1','key2']) :
    print(k1, k2)
    print(group)
```

```
a one
  key1 key2     data1     data2
0    a  one  0.084204  0.255810
4    a  one  1.276054 -0.947729
a two
  key1 key2     data1     data2
1    a  two -0.519531  0.141098
b one
  key1 key2     data1     data2
2    b  one -2.047994  0.238742
b two
  key1 key2     data1     data2
3    b  two -0.770977  0.147164
```

In [22]:
```
for (k1, k2), group in df.groupby(['key1','key2']) :
    print(k1)
```

```
a
a
b
b
```

In [24]:
```
for (k1, k2), group in df.groupby(['key1','key2']) :
    print(k2)
```

```
one
two
one
two
```

In [25]:
```
for (k1, k2), group in df.groupby(['key1','key2']) :
    print(group)
```

```
  key1 key2     data1     data2
0    a  one  0.084204  0.255810
4    a  one  1.276054 -0.947729
  key1 key2     data1     data2
1    a  two -0.519531  0.141098
  key1 key2     data1     data2
2    b  one -2.047994  0.238742
  key1 key2     data1     data2
```

딕셔너리를 이용해서 그룹화 직후의 결과물 보기

- df.groupby('key1')의 결과물은 2개의 변수로 받을 수 있음

➔list로 씌워서 순서대로 슬라이스해서 저장하고, dict()함수를 이용해서 딕셔너리 형태로 저장

  - 그룹화된 결과물에 list()함수를 이용하면, list 형태가 [(성분1.group1),(성분2, group2),…]의 형태로 변환
  - dict() 함수를 이용하면 {성분1:grouop1, 성분2:group2,…}의 형태의 딕셔너리 형태로 변환

```
In [18]: pieces = dict(list(df.groupby('key1')))
         pieces
```

```
Out[18]: {'a':   key1 key2     data1     data2
          0    a  one  0.084204  0.255810
          1    a  two -0.519531  0.141098
          4    a  one  1.276054 -0.947729, 'b':   key1 key2     data1     data2
          2    b  one -2.047994  0.238742
          3    b  two -0.770977  0.147164}
```

```
In [19]: pieces['b']
```

Out[19]:

	key1	key2	data1	data2
2	b	one	-2.047994	0.238742
3	b	two	-0.770977	0.147164

b에 대한 그룹화 결과만 확인 가능

별도로 정의된 Series나 딕셔너리를 이용하여 데이터 그룹화

In [26]:
```
df2 = pd.DataFrame(np.random.randn(5,5),
                   columns = ['a','b','c','d','e'],
                   index = ['Joe', 'Steve', 'Wes', 'Jim', 'travis'])
df2
```

Out[26]:

	a	b	c	d	e
Joe	0.192751	0.839385	0.156864	-0.928672	0.967479
Steve	-0.559498	0.376856	0.300423	-0.210633	0.359183
Wes	-1.440741	-1.380753	-0.370139	0.400788	1.937127
Jim	-0.640154	0.721061	0.925890	-0.478810	0.824843
travis	-0.325366	0.459341	-0.998344	-0.549238	0.175785

In [28]:
```
map_dict = {'a':'red','b':'red','c':'blue',
            'd':'blue', 'e':'red', 'f':'orange'}
map_dict
```

df2의 열을 다른 값으로 매핑하기 위한 딕셔너리 생성

Out[28]:
```
{'a': 'red', 'b': 'red', 'c': 'blue', 'd': 'blue', 'e': 'red', 'f': 'orange'}
```

In [30]:
```
dict(list(df2.groupby(map_dict, axis = 1)))
```

Out[30]:
```
{'blue':                c         d
 Joe     0.156864 -0.928672
 Steve   0.300423 -0.210633
 Wes    -0.370139  0.400788
 Jim     0.925890 -0.478810
 travis -0.998344 -0.549238, 'red':                a         b         e
 Joe     0.192751  0.839385  0.967479
 Steve  -0.559498  0.376856  0.359183
 Wes    -1.440741 -1.380753  1.937127
 Jim    -0.640154  0.721061  0.824843
 travis -0.325366  0.459341  0.175785}
```

별도로 정의된 Series나 딕셔너리를 이용하여 데이터 그룹화


```
In [31]: df2.groupby(map_dict, axis = 1).sum()
```

Out [31]:

	blue	red
Joe	-0.771807	1.999616
Steve	0.089790	0.176541
Wes	0.030649	-0.884367
Jim	0.447081	0.905750
travis	-1.547581	0.309760

a부터 e의 열을 red, blue로 매핑한 뒤 각 red, blue에 해당하는 열들을 그룹화하고 나서 합을 구함

```
In [33]: map_s = pd.Series(map_dict)
         df2.groupby(map_s, axis = 1).count()
```

Out [33]:

	blue	red
Joe	2	3
Steve	2	3
Wes	2	3
Jim	2	3
travis	2	3

Series로 매핑한 후 그룹화 가능
그룹화된 열의 개수 확인

그룹화된 데이터에 적용할 수 있는 통계함수

함수	설명
count	각 그룹내의 (NaN이 아닌) 값들의 개수를 계산
sum	각 그룹내의 (NaN이 아닌) 값들의 합을 계산
mean	각 그룹내의 (NaN이 아닌) 값들의 평균을 계산
median	각 그룹내의 (NaN이 아닌) 값들 중 중간값을 반환
std, var	각 그룹내의 값들의 표준편차, 분산을 계산
min, max	각 그룹내의 값들 중 최소값, 최대값을 반환
prod	각 그룹내의 (NaN이 아닌) 값들의 전체의 곱을 계산
first, last	각 그룹내의 (NaN이 아닌) 값들 중 첫번째, 마지막 값들을 반환

데이터 그룹화 결과물에 사용자 정의 함수 적용하기

- 그룹화된 결과물에 agg()함수를 적용하게 되면, 그룹화된 각 그룹마다 사용자 정의함수를 적용 가능

```
In [35]: grouped = df.groupby('key1')
         def peak_to_peak(arr):
             return arr.max()-arr.min()
         grouped.agg(peak_to_peak)
```

Out [35]:

각 열의 최소-최대

key1	data1	data2
a	1.795585	1.203539
b	1.277017	0.091579

표준편차

```
In [36]: grouped.agg('std')
```

Out [36]:

key1	data1	data2
a	0.913704	0.664230
b	0.902987	0.064756

그룹화된 결과물에 대한 전체 통계량 보기


```
In [37]: grouped.describe()
```

Out [37]:

	data1								data2	
	count	mean	std	min	25%	50%	75%	max	count	me
key1										
a	3.0	0.280242	0.913704	-0.519531	-0.217663	0.084204	0.680129	1.276054	3.0	-0.
b	2.0	-1.409485	0.902987	-2.047994	-1.728740	-1.409485	-1.090231	-0.770977	2.0	0.

DataFrame 멀티 색인

- xs() : 계층적 색인

df2

		apple		banana	
		price	qty	price	qty
1	s1	0	1	2	3
	s2	4	5	6	7
2	s1	8	9	10	11
	s2	12	13	14	15

df2.xs('apple', axis=1)

df2['apple'] # 컬럼색인 우선

		price	qty
1	s1	0	1
	s2	4	5
2	s1	8	9
	s2	12	13

df2['apple','price'] # 상위, 하위 level 색인

```
1 s1   0
  s2   4
2 s1   8
  s2  12
Name: (apple, price), dtype: int32
```

df2['apple','price'][1] # 추가 색인을 통해 index를 색인 가능

```
s1   0
s2   4
Name: (apple, price), dtype: int32
```

df2.xs('apple', level = 0, axis = 1).xs(2, level=0)

	price	qty
s1	8	9
s2	12	13

apple 컬럼의 2번째 레벨

엑셀파일 다루기

- **엑셀 파일의 데이터 읽기**

  df = pd.read_excel('excel_file.xlsx' [, sheet_name = number 혹은 '시트이름', index_col = number 혹은 '열이름'])

  - excel_file.xlsx을 읽어와서 엑셀파일의 내용을 pandas의 DataFrame 데이터로 변환 후에 반환

- **데이터를 엑셀 파일로 쓰기**

```
# (1) Pandas의 ExcelWriter 객체 생성
excel_writer = pd.ExcelWriter('excel_output.xlsx',engine='xlsxwriter')

#(2) DataFrame 데이터를 지정된 엑셀 시트에 쓰기
df1.to_excel(excel_writer[, index=True 혹은 False, sheet_name='시트이름1'])
df2.to_excel(excel_writer[, index=True 혹은 False, sheet_name='시트이름2'])

#(3) ExcelWriter 객체를 닫고, 지정된 엑셀 파일 생성
excel_writer.save()
```

아나콘다를 설치할때 설치된 xlsxwriter모듈 이용

다르게 지정해 수행하면 하나의 엑셀 파일에 여러 개의 시트를 생성할 수 있음

pandas를 이용해 엑셀 파일을 생성할때 같은 이름의 엑셀파일이 있으면 사용자에게 확인하지 않고 덮어쓰므로 주의!!

실습


```
import pandas as pd
import numpy as np
df14 = pd.read_csv("baseball14.csv",encoding='utf-8')
df15 = pd.read_csv("baseball15.csv",encoding='utf-8')
df16 = pd.read_csv("baseball16.csv",encoding='utf-8')
```

- **3년간 총 안타 개수를 알아본다.**
  - 'H'열에 선수별 안타 수 가 들어 있다.
  - 3년간 선수별 총 안타 개수에 누락 값(NaN)이 있으면 0으로 채운다.
- **df14.head(10) 실행해서 데이터 타입을 확인 후 df14, df15, df16의 데이터를 합쳐 통합 표를 만드시오.**
  - 'playerID'를 기준으로 정렬하시오.
  - 'playerID'를 level 0인 인덱스로, 'yearID'를 level 1인 인덱스로 하시오.
  - 'stint', 'teamID', 'lgID ' 를 삭제하시오.
  - 'Hit Rate' 열을 만들고 안타수/타석수를 소수점 2자리까지 계산 후 저장한다.('H'/'AB')

실습

```
import pandas as pd
import numpy as np
data = pd.read_csv("multi_index_ex1.csv",encoding='cp949',sep =',')
```

1) 각 인덱스와 컬럼을 multi-index를 갖도록 설정
   - ① 지역, 지역.1을 index로 바꿈
   - ② index name을 '품목', '상세품목 ' 으로 설정
   - ③ 0행의 값들 중 A,B,C를 행으로 바꿈
   - ④ 0행들을 삭제하고 다시 data에 저장
   - ⑤ columns name을 '지역','지점'으로 바꿈
   - ⑥ data를 숫자타입으로 변환 data = data.astype('int')

   >>결과 화면

	지역	서울			경기			강원		
	지점	A	B	C	A	B	C	A	B	C
품목	상세품목									
컴퓨터	노트북	1000	1109	1290	1284	1938	948	892	948	876
	데스크탑	1500	1390	1840	1010	1290	1028	839	893	819
가전	TV	2001	2019	1839	1028	1039	1948	980	899	897
	냉장고	1111	1938	1993	1932	1827	1173	1827	1172	1100
모바일	핸드폰	5000	5827	9481	8493	8732	8193	7371	7491	8749
	태블릿	2019	1290	1928	2928	1928	1928	1918	900	899

2) 컴퓨터의 서울지역 판매량 출력

>> 결과 화면

지점	A	B	C
상세품목			
노트북	1000	1109	1290
데스크탑	1500	1390	1840

3) 서울 지역의 컴퓨터의 각 세부항목별 판매량의 합계출력

>> 결과 화면

```
상세품목
노트북      3399
데스크탑     4730
dtype: int64
```

4) 각 지역의 A지점의 TV 판매량 출력

>> 결과화면

지역	서울	경기	강원
품목			
가전	2001	1028	980

5) 각 지역 C지점의 모바일의 각 세부 항목별 판매량 평균 출력 (소수점 2째자리까지)

>> 결과화면

```
상세품목
핸드폰     8807.67
태블릿     1585.00
dtype: float64
```

6) 서울 지역의 A지점의 노트북 판매량 출력

```
품목
컴퓨터    1000
Name: (서울, A), dtype: int32
```

실습

- **'담당자별_판매량_Andy사원.xlsx'을 불러와서 작업**
  - '제품명'은 '벨트', '담당자'는 'A', '지역'은 '가', '1분기' 100, '2분기' 150, '3분기' 200, '4분기' 250을 추가하시오.
  - '담당자'를 A에서 Andy로 수정하시오.
  - '담당자별_판매량_Andy사원_new.xlsx'로 저장하시오.
- **'담당자별_판매량_Becky사원.xlsx'을 불러와서 작업**
  - '담당자'를 'B'에서 Becky로 수정하시오.
- **'담당자별_판매량_Chris사원.xlsx'을 불러와서 작업**
  - '담당자'를 'C'에서 Chris 로 수정하시오.
- **담당자를 수정한 3개의 표를 합치시오.**
  - '총합' 열을 추가하고 '1사분기','2사분기','3사분기','4사분기'의 합을 저장하시오. (뒷장에 출력화면과 같은지 확인 후 아래의 문제를 푸세요.)
  - 담당자 기준으로 '1사분기','2사분기','3사분기','4사분기','총합'의 평균을 소수점 2자리까지 나타낸 표를 manager_outcome에 저장하시오.
  - 제품 별 분기별 판매량 합 추출하시오.
  - 각 분기별 가장 많이 팔린 상품을 찾으시오.
  - 각 분기별 가장 높은 판매량을 낸 담당자를 찾으시오.

- 결과화면

	제품명	담당자	지역	1분기	2분기	3분기	4분기	총합
0	시계	Andy	가	198	123	120	137	578
1	구두	Andy	가	273	241	296	217	1027
2	핸드백	Andy	가	385	316	355	331	1387
3	벨트	Andy	가	100	150	200	250	700
4	시계	Becky	나	154	108	155	114	531
5	구두	Becky	나	200	223	213	202	838
6	핸드백	Becky	나	350	340	377	392	1459
7	시계	Chris	다	168	102	149	174	593
8	구두	Chris	다	231	279	277	292	1079
9	핸드백	Chris	다	365	383	308	323	1379