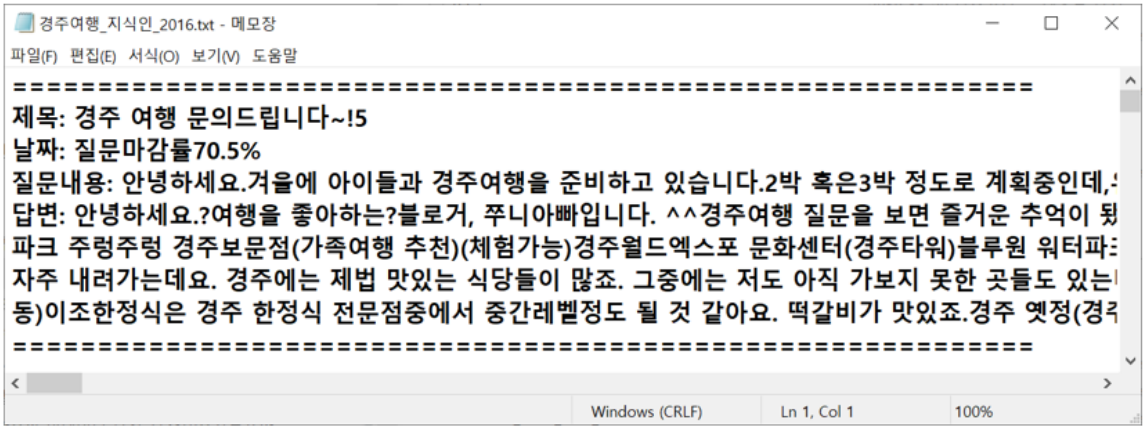
Chap 10. Pandas 모듈을 활용한 정형 데이터 관리하기

이번장에서는 파이썬에서 정형 데이터를 관리할 때 많이 사용되는 Pandas 모듈에 대해서 의미와 사용 방법을 학습하겠습니다.

1. 일반적인 데이터 유형

비정형 데이터



정형 데이터

	A	B	C	D	E
1	행정구역명	위치	총인구수	남자 인구수	여자 인구수
2	창원시	창원시청	1075168	546530	528638
3	진주시	진주시청	340241	168480	171761
4	통영시	통영시청	139439	70639	68800
5	사천시	사천시청	116485	58392	58093
6	김해시	김해시청	527240	266361	260879
7	밀양시	밀양시청	107765	52759	55006
8	거제시	거제시청	248287	130708	117579
9	양산시	양산시청	292376	147189	145187
10	의령군	의령군청	29209	14081	15128

Pandas 사용

pip install pandas

2. Pandas에서 사용되는 주요 데이터 유형과 활용

Series 유형

	행정구역명
1	
2	창원시
3	진주시
4	통영시
5	사천시
6	김해시
7	밀양시
8	거제시
9	양산시
10	의령군
11	함안군
12	창녕군
13	고성군
14	남해군
15	하동군

Data Frame 유형

	A	B	C	D	E
1	행정구역명	위치	총인구수	남자 인구수	여자 인구수
2	창원시	창원시청	1075168	546530	528638
3	진주시	진주시청	340241	168480	171761
4	통영시	통영시청	139439	70639	68800
5	사천시	사천시청	116485	58392	58093
6	김해시	김해시청	527240	266361	260879
7	밀양시	밀양시청	107765	52759	55006
8	거제시	거제시청	248287	130708	117579
9	양산시	양산시청	292376	147189	145187
10	의령군	의령군청	29209	14081	15128

1) Series 유형

(1) 생성하기

```
#pandas ex_1
import pandas as pd

#1.기본문법
member = pd.Series(['홍길동','전우치','강감찬','스티브잡스'])
member
```

```
0      홍길동
1      전우치
2      강감찬
3    스티브잡스
dtype: object
```

```
#Pandas ex_2
import pandas as pd

member=pd.Series(['홍길동','전우치','강감찬','스티브잡스'],
                 index = ['1번','2번','3번','4번'])
member
```

```
1번       홍길동
2번       전우치
3번       강감찬
4번     스티브잡스
dtype: object
```

인덱스를 바꿀 수 있음

```
#pandas ex_3
#딕셔너리 형으로 생성하기

sal_1 = {'홍길동':100 , '일지매' : 130 , '전우치' : 120 }
sal_2 = pd.Series(sal_1)
sal_2
```

```
일지매    130
전우치    120
홍길동    100
dtype: int64
```

(2) 조회하기

```
#Pandas ex_4
import pandas as pd

member=pd.Series(['홍길동','전우치','강감찬','스티브잡스'],
                 index = ['1번','2번','3번','4번'])
member
```

```
1번       홍길동
2번       전우치
3번       강감찬
4번    스티브잡스
dtype: object
```

```
member['1번']
```

```
'홍길동'
```

```
member[ ['1번','3번'] ]
```

```
1번    홍길동
3번    강감찬
dtype: object
```

(3) 데이터 연산하기

```
#pandas ex_5
sal_1 = {'일지매' : 130 , '전우치' : 120, '홍길동':100}
sal_2 = pd.Series(sal_1)
sal_2
```

```
일지매    130
전우치    120
홍길동    100
dtype: int64
```

```
sal_3 = {'홍길동' : 10 , '전우치' : 12, '강감찬':20}
sal_4 = pd.Series(sal_3)
sal_4
```

```
홍길동    10
전우치    12
강감찬    20
dtype: int64
```

```
print(sal_2 + sal_4)
```

```
강감찬      NaN
일지매      NaN
전우치    132.0
홍길동    110.0
dtype: float64
```

2) Data Frame 유형

(1) 생성하기

```
# 2) Data Frame 유형
# Pandas ex_6
# 딕셔너리와 리스트를 활용하여 Data Frame 생성하기

member3 = {'번호' : ['1번','2번','3번'] ,
           '이름' : ['홍길동','전우치','강감찬'],
           '생년' : [1975 , 1980 , 1992] }
member4 = pd.DataFrame(member3)
member4
```

	번호	이름	생년
0	1번	홍길동	1975
1	2번	전우치	1980
2	3번	강감찬	1992

```python
#Pandas ex_7
# 딕셔너리와 리스트를 활용하여 Data Frame 생성하기-컬럼이름순서 지정하기

member3 = {'번호' : ['1번','2번','3번'] ,
           '이름' : ['홍길동','전우치','강감찬'],
           '생년' : [1975 , 1980 , 1992] }
member5 = pd.DataFrame(member3 , columns=['번호','생년','이름'])
member5
```

	번호	생년	이름
0	1번	1975	홍길동
1	2번	1980	전우치
2	3번	1992	강감찬

```python
#pandas ex_7_2
no = ['1번','2번','3번']
name = ['홍길동','전우치','강감찬']
birth = [1975,1982,1980]

member6 = pd.DataFrame()
member6['번호'] = no
member6['이름'] = name
member6['생년'] = birth
member6
```

	번호	이름	생년
0	1번	홍길동	1975
1	2번	전우치	1982
2	3번	강감찬	1980

(2) 정렬하기

```python
# 정렬하기
member6.sort_values(['생년'] , ascending=[False])
```

	번호	이름	생년
1	2번	전우치	1982
2	3번	강감찬	1980
0	1번	홍길동	1975

(3) pandas 에서 xls형식과 csv 형식 불러오기 – pip install xlrd 후 작업하세요

```
1 import pandas as pd
2
3 data1 = pd.read_excel('c:₩₩temp₩₩부품구입대장2.xls')
4 data1
```

	신청날짜	사업장	품목	수량	단가	금액
0	2019-08-23	석수	ANGLE	21	32000	672000
1	2019-09-09	석수	백관	44	56000	2464000
2	2019-06-18	석수	절연판	40	45000	1800000
3	2019-03-13	석수	P/Z O-RING	11	38000	418000
4	2019-05-11	석수	알코올	38	8000	304000
5	2019-06-28	석수	실리콘	30	150000	4500000
6	2019-02-16	석수	소모품	4	2200	8800
7	2019-10-22	석수	MOTOR	14	30000	420000
8	2019-03-18	석수	PAPER BRSH	13	40000	520000
9	2019-07-17	석수	불화카리	23	83000	1909000
10	2019-07-23	석수	마크펜	33	3000	99000
11	2019-03-20	석수	NUT	20	15000	300000
12	2019-01-27	석수	윤활유	30	2000000	60000000
13	2019-07-11	석수	소모품	50	2200	110000
14	2019-05-16	석수	BITE	14	86000	1204000

```
# csv 파일 불러오기 - 인코딩 지정하기
import pandas as pd

data1 = pd.read_csv('c:₩₩temp₩₩사원별판매현황_홍길동.csv' ,
                    names = ['이름','요일','실적'] , encoding='cp949')
data1
```

	이름	요일	실적
0	이름	요일	실적
1	홍길동	월요일	100
2	홍길동	화요일	70
3	홍길동	수요일	80
4	홍길동	목요일	85
5	홍길동	금요일	65
6	홍길동	토요일	95
7	홍길동	일요일	120

(4) 특정 컬럼 조회하기

```
#data frame 에서 원하는 데이터 조회하기
member5
```

	번호	생년	이름
0	1번	1975	홍길동
1	2번	1980	전우치
2	3번	1992	강감찬

```
member5['생년'] # 2개이상의 컬럼일 경우 member5[ ['이름','생년'] ]
```

```
0    1975
1    1980
2    1992
Name: 생년, dtype: int64
```

(5) 원하는 조건으로 행 조회하기

```
1 member5.loc[0]
```

```
번호      1번
생년    1975
이름     홍길동
Name: 0, dtype: object
```

```
1 member5.loc[ member5['번호'] >= '2']
```

	번호	생년	이름
1	2번	1980	전우치
2	3번	1992	강감찬

```
1 member6 = {'번호' : [1,2,3,4,5] ,
2            '이름' : ['홍길동','전우치','강감찬','일지매','을지문덕'],
3            '매출' : [100,200,300,250,150]}
4 member7 = pd.DataFrame(member6 , columns=['번호','이름','매출'])
5 member7
```

	번호	이름	매출
0	1	홍길동	100
1	2	전우치	200
2	3	강감찬	300
3	4	일지매	250
4	5	을지문덕	150

```
1 # 매출이 100 이상 ~ 200 이하인 행만 출력하기
2 member7.loc[ (member7['매출'] >= 100) & (member7['매출'] <= 200)]
```

	번호	이름	매출
0	1	홍길동	100
1	2	전우치	200
4	5	을지문덕	150

```
# 매출이 100 이상 ~ 200 이하인 행의 이름컬럼과 매출 컬럼만 출력하기
member7.loc[ (member7['매출'] >= 100) & (member7['매출'] <= 200),['이름','매출'] ]
```

	이름	매출
0	홍길동	100
1	전우치	200
4	을지문덕	150

```
# 매출이 150 이하 이거나 또는 300 이상인 행의 이름컬럼과 매출 컬럼만 출력하기
member7.loc[ (member7['매출'] <= 150) | (member7['매출'] >= 300),['이름','매출'] ]
```

	이름	매출
0	홍길동	100
2	강감찬	300
4	을지문덕	150

(6) 새로운 행과 열 추가하기

```
# 새로운 열 추가하기
member8 = pd.DataFrame( member5 ,columns=['번호','이름','생년','지역'])
member8
```

	번호	이름	생년	지역
0	1번	홍길동	1975	NaN
1	2번	전우치	1980	NaN
2	3번	강감찬	1992	NaN

```
member8['지역'] = ['서울','대전','강원']
member8
```

	번호	이름	생년	지역
0	1번	홍길동	1975	서울
1	2번	전우치	1980	대전
2	3번	강감찬	1992	강원

```
# 새로운 행 추가하기
member8.loc[3] = ['4번','서진수',1975,'경기']
member8
```

	번호	이름	생년	지역
0	1번	홍길동	1975	서울
1	2번	전우치	1980	대전
2	3번	강감찬	1992	강원
3	4번	서진수	1975	경기

(7) 행과 열 삭제하기

```
# 행 삭제하기
member8
```

	번호	이름	생년	지역
0	1번	홍길동	1975	서울
1	2번	전우치	1980	대전
2	3번	강감찬	1992	강원
3	4번	서진수	1975	경기

```
member8.drop( [0] )
```

	번호	이름	생년	지역
1	2번	전우치	1980	대전
2	3번	강감찬	1992	강원
3	4번	서진수	1975	경기

위 방법으로 데이터 프레임에서 1번행이 삭제되었습니다.

```
# 특정 조건의 행 삭제하기
member8.drop( member8[ member8.생년 >= 1980].index )
```

	번호	이름	생년	지역
0	1번	홍길동	1975	서울
3	4번	서진수	1975	경기

```
# 2개 이상의 특정 조건을 지정하여 특정 행 삭제하기
member8.drop( member8[ (member8.생년 >= 1980) | (member8.지역=='서울')].index )
```

	번호	이름	생년	지역
3	4번	서진수	1975	경기

만약 위 데이터프레임에서 특정 컬럼을 삭제하려면 – 예를 들어 지역컬럼을 삭제하려면 아래와 같이 진행하면 됩니다.

member8.drop('지역' , axis=1)

그런데 위 명령은 현재 화면에 출력되는 내용에만 반영되고 원본의 데이터프레임에서 실제 해당 컬럼을 삭제하진 않습니다. 혹시 삭제된 상태로 저장을 해야 할 경우라면 아래와 같이 다른 변수에 위 컬럼을 삭제한 데이터 프레임을 넣고 저장해야 합니다.

member9 = member8.drop('지역' , axis = 1)

(8) merge( ) & concat( ) – DataFrame 합치기

실무를 하다 보면 여러 개의 데이터 프레임을 하나로 합쳐야 하는 경우가 종종 생깁니다.

SQL에서 Join 기능과 유사한 방식인데 아래의 예제로 살펴보겠습니다.

```
# merge 실습용 데이터 생성 1
df_1 = {'학번' : ['1001','1002','1003'] ,
        '이름' : ['홍길동','일지매','전우치'],
        '키' : [180 , 175 , 168] }
stu_1 = pd.DataFrame(df_1)
stu_1
```

	학번	이름	키
0	1001	홍길동	180
1	1002	일지매	175
2	1003	전우치	168

```
# merge 실습용 데이터 생성 1
df_2 = {'학번' : ['1001','1002','1004'] ,
        '이름' : ['홍길동','일지매','강감찬'],
        '몸무게' : [80 , 75 , 68] }
stu_2 = pd.DataFrame(df_2)
stu_2
```

	학번	이름	몸무게
0	1001	홍길동	80
1	1002	일지매	75
2	1004	강감찬	68

아래 예제는 두 데이터 프레임에 공통적으로 있는 데이터만 출력합니다

```
# 기본값으로 merge - inner join 방식
pd.merge(stu_1 , stu_2 , left_on='이름',right_on='이름')
```

	학번_x	이름	키	학번_y	몸무게
0	1001	홍길동	180	1001	80
1	1002	일지매	175	1002	75

```
# Merge 예제 - merge 할 기준을 지정하기
pd.merge(stu_1 , stu_2 , how='left',left_on='이름',right_on='이름')
```

	학번_x	이름	키	학번_y	몸무게
0	1001	홍길동	180	1001	80.0
1	1002	일지매	175	1002	75.0
2	1003	전우치	168	NaN	NaN

```
pd.merge(stu_1 , stu_2 , how='right',left_on='이름',right_on='이름')
```

	학번_x	이름	키	학번_y	몸무게
0	1001	홍길동	180.0	1001	80
1	1002	일지매	175.0	1002	75
2	NaN	강감찬	NaN	1004	68

아래 예제는 두 데이터 프레임 중에 한쪽에만 있을 경우에도 모두 출력을 시킵니다.

```
# merge 예제 - merge 할 기준을 지정하기
pd.merge(stu_1 , stu_2 , how='outer',left_on='이름',right_on='이름')
```

	학번_x	이름	키	학번_y	몸무게
0	1001	홍길동	180.0	1001	80.0
1	1002	일지매	175.0	1002	75.0
2	1003	전우치	168.0	NaN	NaN
3	NaN	강감찬	NaN	1004	68.0

아래와 같이 concat( ) 함수를 활용하여 데이터프레임 자체를 좌/우 또는 상/하로 합치는 방법도 많이 사용합니다.

```
pd.concat([stu_1, stu_2], axis=0)
```

	학번	이름	키	몸무게
0	1001	홍길동	180.0	NaN
1	1002	일지매	175.0	NaN
2	1003	전우치	168.0	NaN
0	1001	홍길동	NaN	80.0
1	1002	일지매	NaN	75.0
2	1004	강감찬	NaN	68.0

```
pd.concat([stu_1, stu_2], axis=1)
```

	학번	이름	키	학번	이름	몸무게
0	1001	홍길동	180	1001	홍길동	80
1	1002	일지매	175	1002	일지매	75
2	1003	전우치	168	1004	강감찬	68

3) NaN , NULL 값 관리하기

데이터 분석을 할 때 자주 만나는 데이터가 NA 형과 NULL 형입니다.

NaN형은 데이터가 있으나 사용할 수 없는 유형을 의미하고 NULL 형은 데이터가 아예 없는 비어 있는 경우를 말하는데 이런 데이터를 어떻게 처리하는가에 따라 결과가 달라지기 때문에 NaN형과 NULL 형을 잘 관리해야 합니다.

```
# NaN 형과 NULL 형 관리하기
from numpy import nan as NA
import pandas as pd

member9 = {'번호' : [1,2,3,4,5,6,7] ,
           '이름' : ['홍길동','전우치',NA,'일지매','을지문덕',NA,'김유신'],
           '매출' : [100,200,300,NA,150,NA ,250]}
member10 = pd.DataFrame(member9 , columns=['번호','이름','매출'])
member10
```

	번호	이름	매출
0	1	홍길동	100.0
1	2	전우치	200.0
2	3	NaN	300.0
3	4	일지매	NaN
4	5	을지문덕	150.0
5	6	NaN	NaN
6	7	김유신	250.0

아래와 같이 NaN 값과 NULL 값의 개수를 구할 수 있습니다.

```
import numpy as np

print(member10.count())
print('NULL개수:',np.count_nonzero(member10['매출'].isnull()))
print('NaN개수:',np.count_nonzero(member10['이름'].isna()))
```

```
번호    7
이름    5
매출    5
dtype: int64
NULL개수: 2
NaN개수: 2
```

아래와 같이 NaN 값을 특정 값으로 변경할 수 있습니다.

```
#NaN 값을 특정 값으로 변경하기
print(member10.fillna(0) )
print()
print(member10['이름'].fillna('이름없음'))
```

```
   번호     이름      매출
0   1    홍길동   100.0
1   2    전우치   200.0
2   3      0   300.0
3   4    일지매     0.0
4   5   을지문덕   150.0
5   6      0     0.0
6   7    김유신   250.0

0      홍길동
1      전우치
2     이름없음
3      일지매
4     을지문덕
5     이름없음
6      김유신
Name: 이름, dtype: object
```

아래와 같이 NaN 값을 포함하는 행을 모두 삭제할 수도 있습니다.

```
# NaN 값 삭제하기
drop_member10 = member10.dropna()
drop_member10
```

	번호	이름	매출
0	1	홍길동	100.0
1	2	전우치	200.0
4	5	을지문덕	150.0
6	7	김유신	250.0

4) groupby 로 그룹핑하기

```python
# groupby함수 활용하기
import pandas as pd

member11 = {'번호' : [1,2,3,4,5] ,
            '팀명' : ['영업1팀','영업2팀','영업3팀','영업2팀','영업1팀'],
            '이름' : ['홍길동','전우치','일지매','을지문덕','홍길동'],
            '매출' : [100,200,300,150,250]}
member12 = pd.DataFrame(member11 , columns=['번호','팀명','이름','매출'])
member12

# 이름별로 매출액의 합계를 구하고 싶을 경우
group_12 =  member12['매출'].groupby(member12['팀명'])
print('팀별매출합계:', group_12.sum() )
print()

group_12.sum =  member12['매출'].groupby([member12['팀명'],member12['이름']]).sum()
print('팀별 이름별 매출합계:', group_12.sum )
print()

# 팀명별로 매출액의 평균을 구하고 싶을 경우
print('팀별 매출평균:', group_12.mean() )
print( )

print('팀별 매출건수:', group_12.count() )
```

```
팀별매출합계: 팀명
영업1팀    350
영업2팀    350
영업3팀    300
Name: 매출, dtype: int64

팀별 이름별 매출합계: 팀명    이름
영업1팀  홍길동     350
영업2팀  을지문덕    150
       전우치     200
영업3팀  일지매     300
Name: 매출, dtype: int64

팀별 매출평균: 팀명
영업1팀    175
영업2팀    175
영업3팀    300
Name: 매출, dtype: int64

팀별 매출건수: 팀명
영업1팀    2
영업2팀    2
영업3팀    1
Name: 매출, dtype: int64
```

pandas 모듈에 아주 많은 기능들이 있지만 많이 사용되어 꼭 알아야 하는 기능 위주로 살펴 보았습니다. 많이 연습해서 꼭 실력으로 만드세요.

[파이썬 능력자 너도 될 수 있어~! - 서진수 저 -]

[연습문제]

1. 아래 그림과 같은 유형으로 표를 생성하세요

	번호	이름	생일
0	1번	홍길동	1975
1	2번	전우치	
2	3번	강감찬	1982

2. 주어진 '부품구입대장.xlsx' 파일을 불러와서 변수에 할당한 후 아래 그림과 같이 날짜, 품목, 수량, 금액 컬럼으로 이루어진 데이터 프레임을 생성하세요.

	날짜	품목	수량	금액
0	2019-08-23	ANGLE	21	672000
1	2019-09-09	백관	44	2464000
2	2019-06-18	절연판	40	1800000
3	2019-03-13	P/Z O-RING	11	418000
4	2019-05-11	알코올	38	304000
5	2019-06-28	실리콘	30	4500000
6	2019-02-16	소모품	4	8800
7	2019-10-22	MOTOR	14	420000
8	2019-03-18	PAPER BRSH	13	520000
9	2019-07-17	불화카리	23	1909000
10	2019-07-23	마크펜	33	99000
11	2019-03-20	NUT	20	300000
12	2019-01-27	윤활유	30	60000000
13	2019-07-11	소모품	50	110000
14	2019-05-16	BITE	14	1204000

3. 위 문제 2에서 생성한 데이터프레임에서 금액이 1000000 원이 넘는 항목만 조회하여 아래와 같이 날짜, 품목, 수량, 금액 컬럼을 출력하되 금액이 많은 것부터 먼저 출력되게 하세요.

	날짜	품목	수량	금액
12	2019-01-27	윤활유	30	60000000
5	2019-06-28	실리콘	30	4500000
1	2019-09-09	백관	44	2464000
9	2019-07-17	불화카리	23	1909000
2	2019-06-18	절연판	40	1800000
14	2019-05-16	BITE	14	1204000

4. 위 연습문제 3에서 동일한 조건으로 조회하여 아래 그림과 같이 품목과 금액 컬럼만 출력하세요.

	품목	금액
12	윤활유	60000000
5	실리콘	4500000
1	백관	2464000
9	불화카리	1909000
2	절연판	1800000
14	BITE	1204000

5. 위 문제 2 에서 품명이 '백관' 일 경우 코드 컬럼에 'A1' , '절연판' 일 경우 'B1' , '실리콘'일 경우 'C1' 값을 입력하여 아래 그림과 같이 출력되도록 코드를 작성하세요.

	날짜	품목	수량	금액	코드
0	2019-08-23	ANGLE	21	672000	
1	2019-09-09	백관	44	2464000	A1
2	2019-06-18	절연판	40	1800000	B1
3	2019-03-13	P/Z O-RING	11	418000	
4	2019-05-11	알코올	38	304000	
5	2019-06-28	실리콘	30	4500000	C1
6	2019-02-16	소모품	4	8800	
7	2019-10-22	MOTOR	14	420000	
8	2019-03-18	PAPER BRSH	13	520000	
9	2019-07-17	불화카리	23	1909000	
10	2019-07-23	마크펜	33	99000	
11	2019-03-20	NUT	20	300000	
12	2019-01-27	윤활유	30	60000000	
13	2019-07-11	소모품	50	110000	
14	2019-05-16	BITE	14	1204000	

6. 주어진 "학생_만족도_조사_실습.xlsx" 파일을 불러와서 data1 변수에 저장하세요.
(아래 화면 참조)

	연번	학년	소속	분반	1. 튜터링 활동을 위한 자료준비 및 선행학습은 적절하였다.	2. 튜티의 학습 수준 및 특성을 파악하였다.	3. 활동 중 튜티와의 상호작용을 유도하였다.	4. 튜티의 학습 수준 및 특성을 고려하여 활동하였다.	5. 튜터링 활동 시간은 적절하였다.	6. 튜터링 활동 장소는 적절하였다.	...	11. 책임감이 증진되었다.	12. 협동심이 증진되었다.	13. 의사소통능력이 증진되었다.	14. 티칭스킬이 증진되었다.	15. 학교생활 적응능력이 증진되었다.	16. 효원튜터링 프로그램에 대해 만족한다.	팀활동좋았던점	팀활동 개선점	운영상 좋았던점	운영상 개선점
0	1.0	4.0	7	NaN	5	5	5	5	5	5.0	...	5.000000	5.000000	5.000000	5.0	5.0	만족	의사소통 능력, 리더쉽 능력이 승진되었다.	NaN	보고서를 통해 피드백을 받을 수 있어 좋았다	NaN
1	2.0	4.0	4	NaN	3	4	4	4	4	4.0	...	4.000000	4.000000	4.000000	4.0	4.0	만족	다른과에서 복수전공을 하고오신 튜티를 만나 좋았다	더 열심히 임했으면 더 좋은 결과를 얻을 수 있었을 것 같다	주간 보고서를 제풀해야해서 주당 1회 활동이 의무적으로 이루어 진 것 같다	보고서 피드백에 대한 튜티의 보고서 제풀이 필요한 것 같다
2	3.0	4.0	5	NaN	3	4	4	5	4	4.0	...	4.000000	3.000000	4.000000	5.0	4.0	만족	무난하게 잘 완료	좀 더 다양한 학습활동을 했었으면	전공 과목을 대상으로 한다는 점에서 도움됨	오티때 추천할만한 학습법을 알려주면 좋을 것 같다
3	4.0	3.0	1	NaN	5	5	5	5	5	5.0	...	5.000000	5.000000	5.000000	5.0	5.0	만족	질의응답이 활발	시간배분을 잘해야 했다	설명을 잘해주셔서 좋았다	NaN
4	5.0	3.0	12	NaN	4	4	5	4	3	4.0	...	4.000000	3.000000	4.000000	5.0	5.0	만족	NaN	자투함을 개선 할 신선한 티칭방식이 필요	자율적인 튜터링 활동이여서 좋았다	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
73	NaN	NaN	11	NaN	0	0	3	21	34	NaN	...	0.051724	0.362069	0.586207	NaN	NaN	NaN	NaN	NaN	NaN	NaN
74	NaN	NaN	12	NaN	0	0	9	24	25	NaN	...	0.155172	0.413793	0.431034	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75	NaN	NaN	13	NaN	0	0	3	21	34	NaN	...	0.051724	0.362069	0.586207	NaN	NaN	NaN	NaN	NaN	NaN	NaN
76	NaN	NaN	14	NaN	0	0	3	25	30	NaN	...	0.051724	0.431034	0.517241	NaN	NaN	NaN	NaN	NaN	NaN	NaN
77	NaN	NaN	15	NaN	0	3	7	24	24	NaN	...	0.120690	0.413793	0.413793	NaN	NaN	NaN	NaN	NaN	NaN	NaN

78 rows × 24 columns

7. 위 6번의 data1에서 아래와 같이 특정 컬럼(팀활동좋았던점, 운영상 좋았던점)만 조회하여 data2 변수에 저장하세요. (아래 화면 참조)

	팀활동좋았던점	운영상 좋았던점
0	의사소통 능력, 리더쉽 능력이 승진되었다.	보고서를 통해 피드백을 받을 수 있어 좋았다
1	다른과에서 복수전공을 하고오신 튜티를 만나 좋았다	주간 보고서를 제풀해야해서 주당 1회 활동이 의무적으로 이루어 진 것 같다
2	무난하게 잘 완료	전공 과목을 대상으로 한다는 점에서 도움됨
3	질의응답이 활발	설명을 잘해주셔서 좋았다
4	NaN	자율적인 튜터링 활동이어서 좋았다
5	동기와 협동심이 증진되었으며 책임감이 향상	바로바로 피드백이 가능하여 효율적이였다
6	서로에게 도움이 되었던것 같다	프로그램이 있었기 때문에 꾸준히 할 수 있었던것 같다
7	NaN	꾸준한 피드백
8	피드백 적극적 반영	보고서피드백
9	NaN	NaN

8. 아래 그림과 같이 주어진 '과학자들.csv' 파일을 불러와서 Born 컬럼의 값에서 Died 컬럼의 값을 빼서 과학자들의 생존날짜를 구하세요.
(단 아래의 Born 컬럼과 Died컬럼의 데이터 유형은 object 형(문자형) 이라서 바로 뺄 수 없고 날짜 유형으로 변경하여 작업하세요)

```
# pandas 연습문제 8
import pandas as pd
data1 = pd.read_csv("c:₩₩temp₩₩과학자들.csv")
data1[:10]
```

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist
5	John Snow	1813-03-15	1858-06-16	45	Physician
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

[작업 후 결과]

	Name	Born	Died	Age	Occupation	태어난날짜	사망날짜	생존기간(일)
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23	28422 days

9. 아래와 같이 6개의 파일을 사용해서 주어진 문제를 해결하시오.

1) 주어진 파일중에서 "거래내역_1.csv" 파일과 "거래내역_2.csv" 파일에 있는 내용을 합쳐서 거래내역_all 변수에 저장하시오. 두 파일을 합치면 총 6786건의 데이터가 나와야 합니다.
(아래 그림을 참고하세요)

거래내역_1.csv

	transaction_id	price	payment_date	customer_id
0	T0000000113	210000	2019-02-01 01:36:57	PL563502
1	T0000000114	50000	2019-02-01 01:37:23	HD678019
2	T0000000115	120000	2019-02-01 02:34:19	HD298120
3	T0000000116	210000	2019-02-01 02:47:23	IK452215
4	T0000000117	170000	2019-02-01 04:33:46	PL542865
...	...	...	...	...
4995	T0000005108	210000	2019-06-15 02:42:41	HD315748
4996	T0000005109	150000	2019-06-15 03:36:16	HI215420
4997	T0000005110	50000	2019-06-15 03:44:06	IK880102
4998	T0000005111	210000	2019-06-15 04:14:06	IK074758
4999	T0000005112	50000	2019-06-15 04:42:38	HD444151

5000 rows × 4 columns

거래내역_2.csv

	transaction_id	price	payment_date	customer_id
0	T0000005113	295000	2019-06-15 07:20:27	TS169261
1	T0000005114	50000	2019-06-15 07:35:47	HI599892
2	T0000005115	85000	2019-06-15 07:56:36	HI421757
3	T0000005116	50000	2019-06-15 08:40:55	OA386378
4	T0000005117	120000	2019-06-15 08:44:23	TS506913
...	...	...	...	...
1781	T0000006894	180000	2019-07-31 21:20:44	HI400734
1782	T0000006895	85000	2019-07-31 21:52:48	AS339451
1783	T0000006896	100000	2019-07-31 23:35:25	OA027325
1784	T0000006897	85000	2019-07-31 23:39:35	TS624738
1785	T0000006898	85000	2019-07-31 23:41:38	AS834214

1786 rows × 4 columns

[합친 결과]

	transaction_id	price	payment_date	customer_id
0	T0000000113	210000	2019-02-01 01:36:57	PL563502
1	T0000000114	50000	2019-02-01 01:37:23	HD678019
2	T0000000115	120000	2019-02-01 02:34:19	HD298120
3	T0000000116	210000	2019-02-01 02:47:23	IK452215
4	T0000000117	170000	2019-02-01 04:33:46	PL542865
...	...	...	...	...
1781	T0000006894	180000	2019-07-31 21:20:44	HI400734
1782	T0000006895	85000	2019-07-31 21:52:48	AS339451
1783	T0000006896	100000	2019-07-31 23:35:25	OA027325
1784	T0000006897	85000	2019-07-31 23:39:35	TS624738
1785	T0000006898	85000	2019-07-31 23:41:38	AS834214

6786 rows × 4 columns

2)주어진 상세거래내역_1.csv 파일과 상세거래내역_2.csv 파일을 하나의 변수로 합친 후 위 1번에서 생성한 거래내역_all 변수와 조인하여 transaction_id, payment_date , customer_id 값을 상세거래내역 컬럼에 추가하여 아래 예시 화면처럼 출력하세요. 조인에 사용할 키는 transaction_id 를 사용하고 상세거래내역의 모든 데이터들이 출력되도록 조인방식을 사용하세요.

	detail_id	transaction_id	item_id	quantity	payment_date	customer_id
0	0	T0000000113	S005	1	2019-02-01 01:36:57	PL563502
1	1	T0000000114	S001	1	2019-02-01 01:37:23	HD678019
2	2	T0000000115	S003	1	2019-02-01 02:34:19	HD298120
3	3	T0000000116	S005	1	2019-02-01 02:47:23	IK452215
4	4	T0000000117	S002	2	2019-02-01 04:33:46	PL542865
...	...	...	...	...	...	...
7139	7139	T0000006894	S004	1	2019-07-31 21:20:44	HI400734
7140	7140	T0000006895	S002	1	2019-07-31 21:52:48	AS339451
7141	7141	T0000006896	S001	2	2019-07-31 23:35:25	OA027325
7142	7142	T0000006897	S002	1	2019-07-31 23:39:35	TS624738
7143	7143	T0000006898	S002	1	2019-07-31 23:41:38	AS834214

7144 rows × 6 columns

3) 앞의 2번 문제에서 생성된 최종 결과와 고객정보_마스터.csv 파일과 상품내역_마스터.csv 파일을 조인하여 아래의 예시화면과 같이 데이터를 조인하세요.

	customer_name_y	registration_date_y	email_y ₩	gender_y	age_y	birth_y	pref_y	item_name_y	item_price_y
0	신사임당	2019-01-07 14:34	imoto_yoshimasa@example.com	M	30	1989-07-15	대전광역시	PC-E	210000
1	이순신	2019-01-27 18:00	mifune_rokurou@example.com	M	73	1945-11-29	서울특별시	PC-A	50000
2	권율	2019-01-11 8:16	yamane_kogan@example.com	M	42	1977-05-17	광주광역시	PC-C	120000
3	이성계	2019-01-10 5:07	ikeda_natsumi@example.com	F	47	1972-03-17	인천광역시	PC-E	210000
4	한경희	2019-01-25 6:46	kurita_kenichi@example.com	M	74	1944-12-17	광주광역시	PC-B	85000
...	...	...	...	...	...	...	...	...	...
7139	박성윤	2019-01-04 13:24	shishido_akira@example.com	M	64	1955-01-13	대구광역시	PC-D	180000
7140	백무성	2019-02-11 19:34	aihara_miki@example.com	F	74	1945-02-03	대구광역시	PC-B	85000
7141	이준석	2019-04-17 9:23	matsuda_saki@example.com	F	40	1979-05-25	서울특별시	PC-A	50000
7142	박가빈	2019-02-20 18:15	shinndou_masatoshi@example.com	M	56	1963-02-21	인천광역시	PC-B	85000
7143	이승태	2019-04-07 3:20	tahara_yuuko@example.com	F	74	1944-12-18	대전광역시	PC-B	85000

[7144 rows x 24 columns]