06_02_Pandas

1. 합계와 평균에서의 NaN

```
import numpy as np
import pandas as pd

data = [[2, np.nan],
        [7, -3],
        [np.nan, np.nan],
        [1, -2]]
df = pd.DataFrame(data,
                  columns = ["one", "two"],
                  index = ["a", "b", "c", "d"])
display(df)
```

	one	two
a	2.0	NaN
b	7.0	-3.0
c	NaN	NaN
d	1.0	-2.0

- 열 방향 합계 구하기

```
display(df.sum(axis=0))
```

```
one    10.0
two    -5.0
dtype: float64
```

- 행 방향 합계 구하기

```
display(df.sum(axis=1))
```

```
a    2.0
b    4.0
c    0.0    # NaN은 0으로 간주되니 주의
d   -1.0
dtype: float64
```

- 특정 열의 합계 구하기

```
display(df["one"].sum())
```

```
10.0
```

- 특정 행의 합계 구하기

```
display(df.loc["b"].sum())
```

```
4.0
```

- 평균 구하기
  - **평균을 구할때는 합계와는 달리 NaN을 배제**

```
display(df["one"].mean())
```

```
3.3333333333333335
```

2. 결측값 처리

- 평균으로 대체하기

```
df["one"].fillna(value=df["one"].mean(), inplace = False)
```

```
a    2.000000
b    7.000000
c    3.333333   # 결측값이 평균값으로 대체 됨
d    1.000000
Name: one, dtype: float64
```

- 최소값으로 대체하기

```
df["two"].fillna(value=df["two"].min(), inplace = False)
```

```
a   -3.0    # 결측값이 최소값으로 대체 됨
b   -3.0
c   -3.0    # 결측값이 최소값으로 대체 됨
d   -2.0
Name: two, dtype: float64
```

3. 정렬하기

```
# random값을 도출해서 DataFrame을 생성
np.random.seed(0)

## 0~9까지의 정수형 난수를 생성(6, 4) 형태로 생성
df = pd.DataFrame(np.random.randint(0,10,(6,4)))
df.columns = ["A", "B", "C", "D"]
df.index = pd.date_range("20190101", periods = 6)

display(df)
```

	A	B	C	D
2019-01-01	5	0	3	3
2019-01-02	7	9	3	5
2019-01-03	2	4	7	6
2019-01-04	8	8	1	6
2019-01-05	7	7	8	1
2019-01-06	5	9	8	9

- 뒤섞기

```
# 순열 랜덤 치환
random_date = np.random.permutation(df.index)

# 원본은 고정되어 있고 바뀐 결과 DataFrame 리턴
df2 = df.reindex(index = random_date, columns=["B", "A", "D", "C"])

display(df2)
```

	B	A	D	C
2019-01-06	9	5	9	8
2019-01-01	0	5	3	3
2019-01-05	7	7	1	8
2019-01-02	9	7	5	3
2019-01-04	8	8	6	1
2019-01-03	4	2	6	7

- Column 기반 Index 정렬

```
df2.sort_index(axis=1, ascending = True)    # False는 내림차순
```

	A	B	C	D
2019-01-06	5	9	8	9
2019-01-01	5	0	3	3
2019-01-05	7	7	8	1
2019-01-02	7	9	3	5
2019-01-04	8	8	1	6
2019-01-03	2	4	7	6

- Value 기반 정렬

```
df2.sort_values(by = ["B", "A"])    # 기본적으로 오름차순
df2.sort_values(by = ["B", "A"], ascending = False) # 내림차순
```

	B	A	D	C
2019-01-01	0	5	3	3
2019-01-03	4	2	6	7
2019-01-05	7	7	1	8
2019-01-04	8	8	6	1
2019-01-06	9	5	9	8
2019-01-02	9	7	5	3

4. 기타 데이터 다루기

- 새로운 Column을 추가

```
df["E"] = ["AA", "BB", "CC", "CC", "AA", "CC"]
```

	A	B	C	D	E
2019-01-01	5	0	3	3	AA
2019-01-02	7	9	3	5	BB
2019-01-03	2	4	7	6	CC
2019-01-04	8	8	1	6	CC
2019-01-05	7	7	8	1	AA
2019-01-06	5	9	8	9	CC

- 배열로 중복을 제외한 값 출력

```
df["E"].unique()
```

```
# ndarray 타입
array(['AA', 'BB', 'CC'], dtype=object)
```

- 각 Value 값들의 개수를 세는 함수

```
df["E"].value_counts()
```

```
# Series 타입
CC    3
AA    2
BB    1
Name: E, dtype: int64
```

- Boolean Mask를 만들기 위한 함수

```
df["E"].isin(["AA"])
```

```
# Series 타입
2019-01-01     True
2019-01-02    False
2019-01-03    False
2019-01-04    False
2019-01-05     True
2019-01-06    False
Freq: D, Name: E, dtype: bool
```

5. DataFrame Merge

```
import numpy as np
import pandas as pd

data1 = {"학번" : [1, 2, 3, 4],
         "이름" : ["홍길동", "김길동", "이순신", "강감찬"],
         "학년" : [2 ,4, 1, 3]}
data2 = {"학번" : [1, 2, 4, 5],
         "학과" : ["컴퓨터", "경영", "철학", "기계"],
         "학점" : [3.4, 1.9, 4.5, 2.7]}
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)
```

- 키의 이름이 같을 경우

```
# pd.merge(테이블1, 테이블2, on=기준, how= "inner" or "outer"("left",
"right"))
```

```
pd.merge(df1, df2, on = "학번", how ="inner")
```

	학번	이름	학년	학과	학점
0	1	홍길동	2	컴퓨터	3.4
1	2	김길동	4	경영	1.9
2	4	강감찬	3	철학	4.5

- 키의 이름이 다를 경우

```
# merge한 DataFrame의 Column명은 뒤의 DataFrame의 Column명
pd.merge(df1, df2, left_on = "학번", right_on = "학생학번", how="inner")
```

	학번	이름	학년	학생학번	학과	학점
0	1	홍길동	2	1	컴퓨터	3.4
1	2	김길동	4	2	경영	1.9
2	4	강감찬	3	4	철학	4.5

- 인덱스를 기준으로 merge할 경우

```
data1 = {"이름" : ["홍길동", "김길동", "이순신", "강감찬"],
         "학년" : [2 ,4, 1, 3]}
data2 = {"학과" : ["컴퓨터", "경영", "철학", "기계"],
         "학점" : [3.4, 1.9, 4.5, 2.7]}
df1 = pd.DataFrame(data1, index=[1, 2, 3, 4])
df2 = pd.DataFrame(data2, index=[1, 2, 4, 5])
```

```
pd.merge(df1, df2, left_index = True, right_index = True, how = "inner")
```

	이름	학년	학과	학점
1	홍길동	2	컴퓨터	3.4
2	김길동	4	경영	1.9
4	강감찬	3	철학	4.5

6. 연습

- 실습 데이터 다운로드
- Data Loading

```
import numpy as np
import pandas as pd

df = pd.read_csv("./data/LoanData/LoanStats3d.csv", sep = ",", skiprows = 1)
```

- 데이터 수 확인

```
display(df.shape)
```

- 필요한 Column만 추출해서 새로운 DataFrame 생성

```
# loan_amnt : 대출 금액
# loan_status : 대출 상태
# grade : 대출 등급
# int_rate : 이자율
# term : 대출기간
df2 = df[["loan_amnt", "loan_status", "grade", "int_rate", "term"]]
```

- 새로운 DataFrame 확인

```
df2.head()
```

- 각 Column의 값 확인

```
display(df2["loan_status"].unique())
display(df2["grade"].unique())
display(df2["term"].unique())
```

- 결측값 제거

```
display(df2.shape)
df2.dropna(how = "any", inplace = True)
```

- 대출 기간별 대출 총액 구하기

```
print("대출기간 36 개월 대출 총액 : {:,}".format(df2.loc[df2["term"] == " 36 months", "loan_amnt"].sum()))
print("대출기간 60 개월 대출 총액 : {:,}".format(df2.loc[df2["term"] == " 60 months", "loan_amnt"].sum()))

terms = df2["term"].unique()
term_result = {}

for t in terms:
    term_sum = df2.loc[df2["term"] == t, "loan_amnt"].sum()
    term_result[t] = term_sum

for key in term_result:
    print("{}".format(key), end = " : ")
    print("{}".format("{:,}".format(term_result[key])))
```

-
```
## 대출 중 불량한 상태의 대출에 대해서
## 불량한 상태의 대출( Charged Off, Late (31-120 days), Late (16-30 days), In Grace Period )
## 각 Grade의 대출 건수를 구하라

bad_loan_status = df2["loan_status"].unique()[[1, 3, 4, 5]]

# 불량 대출에 대한 mask 생성
bad_mask = df2["loan_status"].isin(bad_loan_status)
print(bad_mask.sum())

df3 = df2.loc[bad_mask, "grade"]
df3.value_counts().sort_index()
```