제1유형_문제풀이

✓ 데이터 다루기 유형

1. 데이터 타입(object, int, float, bool 등)
2. 기초통계량(평균,중앙값,사분위수,IQR, 표준편차 등)
3. 데이터 인덱싱, 필터링, 정렬, 변경 등
4. 결측치, 이상치, 중복값 처리(제거 or 대체)
5. 데이터 Scaling(데이터 표준화(z), 데이터정규화(min-max))
6. 데이터 합치기
7. 날짜/시간 데이터, index 다루기

✓ 핵심문제 27개(1~10)

In [1]:
```
# 데이터 불러오기
import pandas as pd
import numpy as np
df = pd.read_csv("mtcars.csv")
df.head()
```

Out[1]:

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [2]:
```
# 문제 1
# mpg 변수의 제 1사분위수를 구하고 반올림하여 정수값으로 출력하시오.
df = pd.read_csv("mtcars.csv")
```


In [3]:
```
# (풀이)
Q1 = df['mpg'].quantile(0.25)
print(round(Q1))
```

```
15
```

In [4]:
```
# 문제 2
# mpg 값이 19이상 21이하인 데이터의 수를 구하시오.
df = pd.read_csv("mtcars.csv")
```

In [5]:
```
# (풀이)
cond1 = (df['mpg']>=19)
cond2 = (df['mpg']<=21)
print(len(df[cond1 & cond2]))

# print(len(df[cond1 & cond2]))
```

```
5
```

In [6]:
```
# 문제 3
# hp 변수의 IQR 값을 구하시오.
df = pd.read_csv("mtcars.csv")
```

In [7]:
```
# (풀이)
Q1 = df['hp'].quantile(0.25)
Q3 = df['hp'].quantile(0.75)
IQR = Q3-Q1
print(IQR)
```

```
83.5
```

In [8]:
```
# 문제 4
# wt 변수의 상위 10개 값의 총합을 구하여 소수점을 버리고 정수로 출력하시오.
df = pd.read_csv("mtcars.csv")
```


In [9]:
```
# (풀이)
top10 = df.sort_values('wt', ascending=False)
sum_top10 = sum( top10['wt'].head(10) )
print(int(sum_top10))  #주의:소수점 반올림이 아니라 버리는 문제
```

```
42
```


In [10]:
```
# 문제 5
```

```
In [10]: # 문제 5
# 전체 자동차에서 cyl가 6인 비율이 얼마인지 반올림하여 소수점 첫째자리까지 출력하시오.
df = pd.read_csv("mtcars.csv")
```

```
In [11]: # (풀이)
len_6 = len(df[ df['cyl'] ==6])
len_total = len(df)
print(round(len_6/len_total, 1))
```

```
0.2
```



```
In [12]: # 문제 6
# 첫번째 행부터 순서대로 10개 뽑은 후 mpg 열의 평균값을 반올림하여 정수로 출력하시오.
df = pd.read_csv("mtcars.csv")
```

```
In [13]: # (풀이)
df10 = df[:10] #  df.loc[0:9], df.head(10)
mean10 = df10['mpg'].mean()
print( round(mean10) )
```

```
20
```

```
In [14]: # 문제 7
# 첫번째 행부터 순서대로 50%까지 데이터를 뽑아 wt 변수의 중앙값을 구하시오.
df = pd.read_csv("mtcars.csv")
```

```
In [15]: # (풀이)
# 50% 데이터에 해당하는 행의 수 (정수로 구하기)
p50 = int(len(df)*0.5)
df50 = df[:p50]
print(df50['wt'].median())   # = quantile(0.50)
```

```
3.44
```

```
In [16]: # 문제 8
# 결측값이 있는 데이터의 수를 출력하시오.

# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', None,'20230127','20220203', '20220205','20230210','20230223','20230312','202
    '제품' : ['A','B', None, 'B', 'A', None,'A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, None, 5, 10, 10, 10, 15, 15, 20, None, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
```



```
In [17]: # (풀이)
# 데이터 결측치 확인
missing = df.isnull().sum()
print(missing.sum())
```

```
5
```

```
In [18]: # 문제 9
# '판매수' 컬럼의 결측값을 판매수의 중앙값으로 대체하고
#  판매수의 평균값을 반올림하여 정수로 출력하시오.

# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', None,'20230127','20220203', '20220205','20230210','20230223','20230312','202
    '제품' : ['A','B', None, 'B', 'A', None,'A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, None, 5, 10, 10, 10, 15, 15, 20, None, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
```

```
In [19]: # (풀이)
# 판매수의 중앙값
Q2 = df['판매수'].median()
df['판매수'] = df['판매수'].fillna(Q2)
mean = df['판매수'].mean()
print( round( mean ) )
```

```
15
```

```
In [20]: # 문제 10
# 판매수 컬럼에 결측치가 있는 행을 제거하고,
# 첫번째 행부터 순서대로 50%까지의 데이터를 추출하여
# 판매수 변수의 Q1(제1사분위수)값을 반올림하여 정수로 출력하시오.

# 데이터 (수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', None,'20230127','20220203', '20220205','20230210','20230223','20230312','202
    '제품' : ['A','B', None, 'B', 'A', None,'A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, None, 5, 10, 10, 10, 15, 15, 20, None, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
```



```
In [21]: # (풀이)
# 결측치 삭제(행기준)
df = df['판매수'].dropna()
# 첫번째 행부터 순서대로 50%까지의 데이터 추출
```

```
# print(len(df))
per50 = int(len(df)*0.5)
df = df[:per50]
# print(df)

# 값 구하기
print( round(df.quantile(0.25))  )
```

5

✓ 핵심문제 27개(11~20)

In [22]:
```
df = pd.read_csv("mtcars.csv")
df.head()
```

Out[22]:

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [23]:
```
# 문제 11
# cyl가 4인 자동차와 6인 자동차 그룹의 mpg 평균값 차이를 절대값으로 반올림하여 정수로 출력하시오.
df = pd.read_csv("mtcars.csv")
```

In [24]:
```
# (풀이)
cond1 = (df['cyl']==4)
cond2 = (df['cyl']==6)
cyl4 = df[cond1]['mpg'].mean()
cyl6 = df[cond2]['mpg'].mean()
print(round(abs(cyl4-cyl6)) )
```

7

In [25]:
```
# 문제 12
# hp 변수에 대해 데이터표준화(Z-score)를 진행하고 이상치의 수를 구하시오.
# (단, 이상치는 Z값이 1.5를 초과하거나 -1.5 미만인 값이다)
df = pd.read_csv("mtcars.csv")
```


In [26]:
```
# (풀이)
# Z = (X-평균)/표준편차
std = df['hp'].std()
mean_hp = df['hp'].mean()
df['zscore'] = (df['hp']-mean_hp) / std # 새로운 변수 생성
# print(df)
cond1 = (df['zscore']>1.5)
cond2 = (df['zscore']<-1.5)
print( len(df[cond1]) + len(df[cond2]) )
```

2

In [27]:
```
# 문제 13
# mpg 컬럼을 최소최대 Scaling을 진행한 후 0.7보다 큰 값을 가지는 레코드 수를 구하라.
df = pd.read_csv("mtcars.csv")
```

In [28]:
```
# (풀이)
# 공식 : (x-min)/(max-min)
min_mpg = df['mpg'].min()
max_mpg = df['mpg'].max()
df['mpg'] = (df['mpg']-min_mpg) / (max_mpg-min_mpg)
# print(df['mpg'])
cond1 = (df['mpg']>0.7)
print( len(df[ cond1 ]) )
```

5

In [29]:
```
# 문제 14
# wt컬럼에 대해 상자그림 기준으로 이상치의 개수를 구하시오.
df = pd.read_csv("mtcars.csv")
```

In [30]:
```
# (풀이)
# 이상치 : Q1, Q3 로 부터 1.5*IQR을 넘어가는 값
Q1 = df['wt'].quantile(0.25)
Q3 = df['wt'].quantile(0.75)
IQR = Q3-Q1

upper = Q3+1.5*IQR
lower = Q1-1.5*IQR

cond1 = (df['wt']>upper)
cond2 = (df['wt']<lower)
```

```
print( len(df[cond1]) + len(df[cond2]) )
```

3

In [31]:
```
# 문제 15
# 판매수 컬럼의 결측치를 최소값으로 대체하고
# 결측치가 있을 때와 최소값으로 대체했을 때 평균값의 차이를 절대값으로 반올림하여 정수로 출력하시오

# 데이터 생성 (수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', None,'20230127','20220203', '20220205','20230210','20230223','20230312','202
    '제품' : ['A','B', None, 'B', 'A', None,'A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, None, 5, 10, 10, 10, 15, 15, 20, None, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
df
```

Out[31]:

	날짜	제품	판매수	개당수익
0	20220103	A	3.0	300
1	20220105	B	NaN	400
2	None	None	5.0	500
3	20230127	B	10.0	600
4	20220203	A	10.0	400
5	20220205	None	10.0	500
6	20230210	A	15.0	500
7	20230223	B	15.0	600
8	20230312	A	20.0	600
9	20230422	B	NaN	700
10	20220505	A	30.0	600
11	20230511	A	40.0	600

In [32]:
```
# (풀이)
# 데이터 복사
df2 = df.copy()
# 최소값으로 결측치 대체
min_ea = df['판매수'].min()
df2['판매수'] = df2['판매수'].fillna(min_ea)

# 중간에 한 번 확인
# print(df2['판매수'])

# 결측치가 있을때, 대체했을때 평균
with_m = df['판매수'].mean()
without_m = df2['판매수'].mean()
print( round(abs(without_m-with_m)) )
```

2

In [33]:
```
# 문제 16
# vs변수가 0이 아닌 차량 중에 mpg 값이 가장 큰 차량의 hp 값을 출력하시오.
df = pd.read_csv("mtcars.csv")
```

In [34]:
```
# (풀이)
# vs변수가 0이 아닌 차량
df = df[ df['vs']!=0 ]
#print(df)

# mpg 값이 가장 큰 차량(내림차순)
df = df.sort_values('mpg', ascending=False)
# print(df)
# 답 구하기
print(df['hp'].iloc[0])
#print(df['hp'].loc[19])
```

65

In [35]:
```
# 문제 17
# gear 변수값이 3, 4인 두 그룹의 hp 표준편차값의 차이를 절대값으로
# 반올림하여 소수점 첫째자리까지 출력하시오.
df = pd.read_csv("mtcars.csv")
```

In [36]:
```
# (풀이)
# 2개 그룹으로 필터링
gear3 = df[ df['gear']==3 ]
gear4 = df[ df['gear']==4 ]

# 표준편차 구하기
std3 = gear3['hp'].std()
std4 = gear4['hp'].std()
print(round(abs(std3-std4), 1))
```

```
21.8
```

In [37]:
```
# 문제 18
# gear 변수의 값별로 그룹화하여 mpg평균값을 산출하고
# 평균값이 높은 그룹의 mpg 제3사분위수 값을 구하시오.
df = pd.read_csv("mtcars.csv")
```

In [38]:
```
# (풀이)
# gear, mpg 변수만 필터링
df = df.loc[:, ['gear', 'mpg'] ]

# gear 변수로 그룹화하여 mpg 평균값 보기
#print( df.groupby('gear').mean() ) # gear 4그룹이 제일 높음

# gear=4 인 그룹의 mpg Q3 구하기
gear4 = df[ df['gear']==4 ]
print(gear4['mpg'].quantile(0.75))
```

```
28.075
```

In [39]:
```
# 문제 19
# hp 항목의 상위 7번째 값으로 상위 7개 값을 변환한 후,
# hp가 150 이상인 데이터를 추출하여 hp의 평균값을 반올림하여 정수로 출력하시오.
df = pd.read_csv("mtcars.csv")
```

In [40]:
```
# (풀이)
# hp 열 기준으로 내림차순 정렬
df = df.sort_values('hp', ascending = False)

# 인덱스 초기화
df = df.reset_index(drop=True) # drop=True 는 기존 index삭제
# print(df)

# hp 상위 7번째 값 확인
top7 = df['hp'].loc[6]  # top7 = 205

# np.where 활용
import numpy as np
df['hp'] = np.where(df['hp']>=top7, top7, df['hp'])
# np.where(조건, 조건에 해당할 때 값, 그렇지 않을 때 값)

# hp 150이상인 데이터
cond1 = (df['hp']>=150)
df = df[cond1]
print(round( df['hp'].mean() ))
```



```
187
```

In [41]:
```
# 문제 20
# car변수에 Merc 문구가 포함된 자동차의 mpg 평균값을 반올림하여 정수로 출력하시오.
df = pd.read_csv("mtcars.csv")
```

In [42]:
```
# (풀이)
df2 = df [ df['car'].str.contains("Merc") ]  # 문자열 추출 알아두기
print(round(df2['mpg'].mean()) )

# 시험환경에서서 답구하는 방법(reset_index() 사용후)
# df = df.reset_index()
# df2 = df [ df['index'].str.contains("Merc") ]  # 문자열 추출 알아두기
# print(round(df2['mpg'].mean()) )
```

```
19
```

✓ 핵심문제 27개(21~27)

In [43]:
```
# 문제 21
# '22년 1분기 매출액을 구하시오
# (매출액 = 판매수*개당수익)

# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', '20230105','20230127','20220203', '20220205','20230210','20230223','20230312
    '제품' : ['A','B', 'A', 'B', 'A', 'B','A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, 5, 5, 10, 10, 10, 15, 15, 20, 25, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
df.head()
```

Out[43]:

	날짜	제품	판매수	개당수익
0	20220103	A	3	300
1	20220105	B	5	400
2	20230105	A	5	500
3	20230127	B	10	600
4	20220203	A	10	400

In [44]:
```
# (풀이1)
# 데이터 타입 datetime 으로 변경
df['날짜'] = pd.to_datetime(df['날짜'])

# 년,월,일 변수(열) 추가하기
df['year'] = df['날짜'].dt.year
df['month'] = df['날짜'].dt.month
df['day'] = df['날짜'].dt.day

# 매출액 변수 추가하기
df['매출액'] = df['판매수'] * df['개당수익']

#df.info()

# 22년으로 필터링
df = df[df['year']==2022]
#print(df.head())

# 1,2,3월 매출액 계산
m1 = df[df['month']==1]['매출액'].sum()
m2 = df[df['month']==2]['매출액'].sum()
m3 = df[df['month']==3]['매출액'].sum()
print(m1+m2+m3)
```

```
11900
```

In [45]:
```
# (풀이2)
# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', '20230105','20230127','20220203', '20220205','20230210','20230223','20230312
    '제품' : ['A','B', 'A', 'B', 'A', 'B','A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, 5, 5, 10, 10, 10, 15, 15, 20, 25, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })

# 데이터 타입 datetime 으로 변경(위와 동일)
df['날짜'] = pd.to_datetime(df['날짜'])

# 매출액 변수 추가하기(위와 동일)
df['매출액'] = df['판매수'] * df['개당수익']

# 날짜 자체를 필터링 :  df['변수'].between( , )
df_after = df [df['날짜'].between('2022-01-01', '2022-03-31')] # 좌우 모두 포함
# (주의) 날짜와 시간이 같이 있는 데이터에 between 함수를 쓸 경우 형식이 동일하게(날짜+시간) 필터링 해야함
# (ex : 2023-01-05 12:30:05 => between('2023-01-05 12:00:00', '2023-01-05 12:40:00') OK
#                           => between('2023-01-05','2023-01-05') NG


print(df_after['매출액'].sum())
```

```
11900
```

In [46]:
```
# (풀이3)
# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', '20230105','20230127','20220203', '20220205','20230210','20230223','20230312
    '제품' : ['A','B', 'A', 'B', 'A', 'B','A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, 5, 5, 10, 10, 10, 15, 15, 20, 25, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })

# 데이터 타입 datetime 으로 변경(위와 동일)
df['날짜'] = pd.to_datetime(df['날짜'])

# 매출액 변수 추가하기(위와 동일)
df['매출액'] = df['판매수'] * df['개당수익']

# 날짜를 인덱스로 설정후 loc 함수 사용
df = df.set_index('날짜')
df_after = df.loc[ (df.index<='2022-03-31') & (df.index>='2022-01-01')]
# print(df_after)
print(df_after['매출액'].sum())
```



```
11900
```

In [47]:
```
# 문제 22
# '22년과 '23년의 총 매출액 차이를 절대값으로 구하시오.
# (매출액 = 판매수*개당수익)
# 데이터 생성(수정금지)
```

```
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', '20230105','20230127','20220203', '20220205','20230210','20230223','20230312
    '제품' : ['A','B', 'A', 'B', 'A', 'B','A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, 5, 5, 10, 10, 10, 15, 15, 20, 25, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
```

In [48]:
```
# (풀이)
# 데이터 타입 datetime 으로 변경
df['날짜'] = pd.to_datetime(df['날짜'])

# 년 변수(열) 추가하기
df['year'] = df['날짜'].dt.year

# 매출액 변수 추가하기
df['매출액'] = df['판매수']*df['개당수익']
#print(df)
# 22년 매출액 구하기
df22 = df[ df['year']==2022 ]
df22_sum = df22['매출액'].sum()

# 23년 매출액 구하기
df23 = df[ df['year']==2023 ]
df23_sum = df23['매출액'].sum()

print(abs(df22_sum-df23_sum))
```

48600

In [49]:
```
# 문제 23
# '23년 총 매출액이 큰 제품의 23년 판매수를 구하시오.
# (매출액 = 판매수*개당수익)

# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', '20230105','20230127','20220203', '20220205','20230210','20230223','20230312
    '제품' : ['A','B', 'A', 'B', 'A', 'B','A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, 5, 5, 10, 10, 10, 15, 15, 20, 25, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
df.head()
```

Out[49]:

	날짜	제품	판매수	개당수익
0	20220103	A	3	300
1	20220105	B	5	400
2	20230105	A	5	500
3	20230127	B	10	600
4	20220203	A	10	400

In [50]:
```
# (풀이)
# 데이터 타입 datetime 으로 변경
df['날짜'] = pd.to_datetime(df['날짜'])

# 년 변수(열) 추가하기
df['year'] = df['날짜'].dt.year

# 매출액 변수 추가하기
df['매출액'] = df['판매수'] * df['개당수익']
# print(df)
# 2023년으로 필터링
df = df[ df['year']== 2023 ]
# df.head()

# 23년 A 매출액과 B 매출액 별도로 구하기
# A매출액
df_a = df[ df['제품']=='A' ]
A_sales = df_a['매출액'].sum()
#print(A_sales)
# B매출액
df_b = df[ df['제품']=='B' ]
B_sales = df_b['매출액'].sum()
#print(B_sales)

# A제품의 23년 판매수
A_sum = df_a['판매수'].sum()
print(A_sum)
```

80

In [51]:
```
# 문제 24
# 매출액이 4천원 초과, 1만원 미만인 데이터 수를 출력하시오.
# (매출액 = 판매수*개당수익)

# 데이터 생성(수정금지)
df = pd.DataFrame( {
    '날짜': ['20220103','20220105', '20230105','20230127','20220203', '20220205','20230210','20230223','20230312
```

```
    '제품' : ['A','B', 'A', 'B', 'A', 'B','A', 'B', 'A', 'B', 'A', 'A'],
    '판매수': [3, 5, 5, 10, 10, 10, 15, 15, 20, 25, 30, 40],
    '개당수익': [300, 400, 500, 600, 400, 500, 500, 600, 600, 700, 600, 600] })
```

In [52]:
```
# (풀이)
df['매출액'] = df['판매수']*df['개당수익']
cond1 = (df['매출액']>4000)
cond2 = (df['매출액']<10000)
print(len(df[ cond1&cond2 ]))
```


4

In [53]:
```
# 문제 25
# 23년 9월 24일 16:00~22:00 사이에 전체 제품의 판매수를 구하시오.

# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] })
time = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df['time']=time
df = df[ ['time','물품','판매수','개당수익']]
df
```

Out[53]:

	time	물품	판매수	개당수익
0	2023-09-24 12:25:00	A	5	500
1	2023-09-24 16:48:25	B	10	600
2	2023-09-24 21:11:50	A	15	500
3	2023-09-25 01:35:15	B	15	600
4	2023-09-25 05:58:40	A	20	600
5	2023-09-25 10:22:05	B	25	700
6	2023-09-25 14:45:30	A	40	600

In [54]:
```
# (풀이1)
# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] })
time = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df['time']=time
df = df[ ['time','물품','판매수','개당수익']]

#-----------------------------------------------------
# 컬럼과 인덱스에 둘다 time 변수를 놓고 풀이
# 데이터 타입 datetime 으로 변경 (항상 df.info()로 데이터 타입 확인할 것)
df['time'] = pd.to_datetime(df['time'])

# index 새로 지정
df = df.set_index('time', drop=False) # 디폴트는 drop=True
# print(df.head())

# 9월 24일 필터링 // df['변수'].between( , )
df_after = df [df['time'].between('2023-09-24 00:00:00', '2023-09-24 23:59:59')] # 형식 동일하게 날짜+시간

# 시간 필터링 16:00 ~ 22:00 (주의:시간이 index에 위치해야 함)
df = df_after.between_time(start_time='16:00',end_time='22:00') # 포함 기준

# ★이해하셨다면 날짜와 시간을 한번에 필터링 하세요★
# df = df[ df['time'].between('2023-09-24 16:00:00', '2023-09-24 22:00:00') ] # 형식 동일하게 날짜+시간


# 전체 판매수 구하기
print(df['판매수'].sum())
```


25

In [55]:
```
# (풀이2)
# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] })
time = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df['time']=time
df = df[ ['time','물품','판매수','개당수익']]

#-----------------------------------------------------
# 데이터 타입 datetime 으로 변경
df['time'] = pd.to_datetime(df['time'])
# index 새로 지정
```

```
df = df.set_index('time')

# loc 함수로 필터링
df = df.loc[ (df.index >='2023-09-24 16:00:00') & (df.index <='2023-09-24 22:00:00') ]

# 전체 판매수 구하기
print(df['판매수'].sum())
```

```
25
```


In [56]:

```
# 문제 26
# 9월 25일 00:00~12:00 까지의 B물품의 매출액 총합을 구하시오.
# (매출액 = 판매수*개당수익)

# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] } )
df['time'] = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df = df[ ['time','물품','판매수','개당수익'] ]
df = df.set_index('time', drop=True)
df
```

Out[56]:

time	물품	판매수	개당수익
2023-09-24 12:25:00	A	5	500
2023-09-24 16:48:25	B	10	600
2023-09-24 21:11:50	A	15	500
2023-09-25 01:35:15	B	15	600
2023-09-25 05:58:40	A	20	600
2023-09-25 10:22:05	B	25	700
2023-09-25 14:45:30	A	40	600

In [57]:

```
# (풀이1)
# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] }, index=time)
time = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df['time']=time
df = df[ ['time','물품','판매수','개당수익'] ]
df = df.set_index('time', drop=True)

#-----------------------------------------------------
# 매출액 변수 추가
df['매출액'] = df['판매수']*df['개당수익']

# time 변수 추가하기, index 초기화 (인덱스를 컬럼으로)
df = df.reset_index() # 디폴트 drop=False
# print(df.head())

# 데이터 타입 datetime 으로 변경
df['time'] = pd.to_datetime(df['time'])

# 9월 25일 00:00:00 ~ 12:00:00 필터링 // df['변수'].between( , )
df = df [df['time'].between('2023-09-25 00:00:00', '2023-09-25 12:00:00')] # 형식 동일하게 날짜+시간
df

# B 물품의 매출액 총합
df = df[ df['물품']=='B']
print(df['매출액'].sum())
```

```
26500
```

In [58]:

```
# (풀이2)
# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] }, index=time)
time = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df['time']=time
df = df[ ['time','물품','판매수','개당수익'] ]
df = df.set_index('time', drop=True)

#-----------------------------------------------------
# 데이터 타입 datetime 으로 변경(df.info() 로 항상확인)
# df['time'] = pd.to_datetime(df['time'])

# 매출액 변수 추가
df['매출액'] = df['판매수']*df['개당수익']
```

```
# loc 함수로 필터링
df = df.loc[ (df.index >='2023-09-25 00:00:00') & (df.index <='2023-09-25 12:00:00') ]

# (참고)
# ★ 만약 날짜와 상관없이 특정 시간대를 필터링 해야할 때
# df.between_time(start_time='00:00:00',end_time='12:00:00')
# (★주의:시간이 index에 위치해야 함)


# B 물품의 매출액 총합
df = df[ df['물품']=='B' ]
print(df['매출액'].sum())
```

```
26500
```

In [59]:
```
# 문제 27
# 9월 24일 12:00~24:00 까지의 A물품의 매출액 총합을 구하시오.
# (매출액 = 판매수*개당수익)

# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] } )
df['time'] = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df = df[ ['time','물품','판매수','개당수익'] ]
df = df.set_index('time', drop=True)
df
```

Out[59]:

time	물품	판매수	개당수익
2023-09-24 12:25:00	A	5	500
2023-09-24 16:48:25	B	10	600
2023-09-24 21:11:50	A	15	500
2023-09-25 01:35:15	B	15	600
2023-09-25 05:58:40	A	20	600
2023-09-25 10:22:05	B	25	700
2023-09-25 14:45:30	A	40	600

In [60]:
```
# (풀이1)
# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] } )
df['time'] = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df = df[ ['time','물품','판매수','개당수익'] ]
df = df.set_index('time', drop=True)

#-------------------------------------------------

# 매출액 변수 추가
df['매출액'] = df['판매수']*df['개당수익']

# time 변수 추가하기, index 초기화 (인덱스를 컬럼으로)
df = df.reset_index()

# 데이터 타입 datetime 으로 변경
df['time'] = pd.to_datetime(df['time'])

# 9월 24일 12:00 ~ 24:00 필터링 // df['변수'].between( , )
df = df [df['time'].between('2023-09-24 12:00:00', '2023-09-24 23:59:59')] # 형식이 동일하게 날짜+시간
# print(df)
# 24:00 에러 발생하기 때문에 그 전 시간으로 카운팅 or 23-09-25 00:00:00 으로 설정

# A 물품의 매출액 총합
df = df[ df['물품']=='A']
print(df['매출액'].sum())
```

```
10000
```

In [61]:
```
# (풀이2)
# 시간 데이터 만들기(수정금지)
df = pd.DataFrame( {
    '물품' : ['A', 'B', 'A', 'B', 'A', 'B', 'A'],
    '판매수': [5, 10, 15, 15, 20, 25, 40],
    '개당수익': [500, 600, 500, 600, 600, 700, 600] } )
df['time'] = pd.date_range('2023-09-24 12:25:00','2023-09-25 14:45:30', periods= 7)
df = df[ ['time','물품','판매수','개당수익'] ]
df = df.set_index('time', drop=True)
#-------------------------------------------------
```

```python
# 매출액 변수 추가
df['매출액'] = df['판매수']*df['개당수익']

# loc 함수로 필터링
df = df.loc[ (df.index >='2023-09-24 12:00:00') & (df.index <='2023-09-24 23:59:59') ]
# 주의 24:00:00 은 입력할 수 없음

# (참고)
# ★ 만약 날짜와 상관없이 특정 시간대를 필터링 해야할 때
# df.between_time(start_time='00:00:00',end_time='00:00:00')
# (★주의:시간이 index에 위치해야 함)

# A 물품의 매출액 총합
df = df[ df['물품']=='A']
print(df['매출액'].sum())
```

```
10000
```