

1유형 - 컬러 구분된 Python 코드

```
import pandas as pd
```

```
import numpy as np
```

```
file_path =  
'C:/Python/PyProject/24년프로젝트/12.빅분기실기/빅분기/data/유형모음/bigdata_1type_master_wit  
h_nan.csv'  
  
df = pd.read_csv(file_path)  
# print(df)  
# print(df.info())
```

'''

1. 결측치 처리 및 통계요약

```
# 소득(income)과 저축(savings)의 결측치를 각각 평균과 중앙값으로 대체한 후 대체한 소득의 평균과 저축의 중앙값을  
구하라.
```

'''

```
income = df['income']  
  
income_mean = income.mean()  
# print(income_mean)  
  
income.fillna(income_mean, inplace=True)  
# print(income)
```

```
savings = df['savings']  
  
savings_mid = savings.median()  
  
savings.fillna(savings_mid, inplace=True)
```

```
# print(df[['income','savings']])
```

'''

2. 연령대파생변수 생성

나이를 기준으로 다음과 같이 연령대를 구분하여 새로운 열(age_gorup)을 생성하시오

- 10대, 20대, 30대, 40대, 50대, 60대이상 으로 그룹하고 각 연령별 평균소득을 구하시오.

'''

```
# print(df[df['age']<20])
```

```
# 경계정하기
```

```
bins = [10,20,30,40,50,60,float('inf')]
```

```
labels = ['10대','20대','30대','40대','50대','60대이상']
```

```
df[ 'age_group' ] = pd.cut(df['age'],bins=bins, labels = labels, right = False)
```

```
print(df)
```

```
group_income = df.groupby('age_group')['income'].mean()
```

```
print(round(group_income,2))
```

이런방법도 있다. 숫자가 아닌 카테고리로 그룹화하고 뭔가 연산을 할때

```
# import pandas as pd
```

```
# data = {
```

```
#     '카테고리': ['과일', '채소', '과일', '채소', '과일'],
```

```

#     '판매량']: [100, 150, 200, 130, 160]
#
# }

# df = pd.DataFrame(data)

# # 카테고리별 판매량의 합계를 새로운 열로 추가
# df['총판매량'] = df.groupby('카테고리')['판매량'].transform('sum')

```

```
# print(df)
```

기본 통계량

sum: 그룹의 합계

mean: 그룹의 평균

median: 그룹의 중앙값

min: 그룹의 최솟값

max: 그룹의 최댓값

std: 그룹의 표준편차

var: 그룹의 분산

count: 그룹의 개수

'''

3. 지역-성별별 대출총합

region과 gender별로 총 대출금(loan_amount)을 합산하고, 성별 간 대출 차이가 가장 큰 지역을 구 하시오.

'''

```
rg_group = df.groupby(['region','gender'])['loan_amount'].sum().unstack()
```

#다중 인덱스 그룹이 되면 열로 변환해주는 unstack을 반드시 해줘야 그룹간 연산이 가능하다.

```
print(rg_group)
```

```
rg_group['diff'] = rg_group['남자'] - rg_group['여자']
```

```
max_diff_region = rg_group['diff'].idxmax()
```

```
max_diff = rg_group['diff'].max()
```

```
print(f'성별 간 대출 차이 가장 큰 지역 : {max_diff_region}, 대출 차이 : {max_diff} ')
```

```
## 피벗 테이블로 할 수도 있다.(여러 상황에서 더 요긴하게 쓰일 수 있는 방법)
```

```
rg_group_new = df.pivot_table(index = 'region',columns = 'gender',values = 'loan_amount',aggfunc = 'sum')
```

```
print(rg_group_new)
```

```
rg_group_new['diff'] = abs(rg_group_new['남자'] - rg_group_new['여자'])
```

```
max_index = rg_group_new['diff'].idxmax()
```

```
print(max_index)
```

```
max_diff = rg_group_new.loc[max_index, 'diff']
```

```
print(max_diff)
```

'''

서브 퀴즈 #1

대전과 광주의 남녀 대출액 차이 계산하기

'''

```
# # '대전'과 '광주' 행 가져오기
```

```
# daejun_row = rg_group.loc['대전']
# gwangju_row = rg_group.loc['광주']

# diff = daejun_row - gwangju_row
# print(diff)
```

```
data = rg_group_new.loc['광주'] - rg_group_new.loc['대전']
print(data)
```

!!!

서브 퀴즈 #2

남녀 대출액 차이가 가장 큰 지역과 가장 낮은 지역간의 남자 대출액 비율, 여자 대출액 비율을 구하라.

!!!

```
max_region = rg_group_new['diff'].idxmax()
```

```
min_region = rg_group_new['diff'].idxmin()
```

```
ratio = rg_group_new.loc[max_region] / rg_group_new.loc[min_region]
```

```
print(ratio)
```

```
# rg_group_new = rg_group_new.sort_values('diff', ascending = False)
# print(rg_group_new)
```

*****

'''

4. 신용 점수 결측치 처리 및 평균

credit_score 결측치는 직업(job)별 평균값으로 채우고, 전체 평균 신용점수를 구하시오.

'''

```
job_mean = df.groupby('job')['credit_score'].mean()  
print(job_mean)
```

그룹별 결측치 확인

```
# na_before = df.groupby('job')['credit_score'].apply(lambda  
x:x.isna().sum())  
  
# print(na_before)  
  
na_before = df['credit_score'].isna().groupby(df['job']).sum()  
print(na_before)
```

job 이름별로 mapping하는 것

```
df['credit_score'] = df['credit_score'].fillna(df['job'].map(job_mean))  
# print(df['credit_score'].info())
```

결측치가 그룹별로 잘 채워졌는지 확인

```
# na_after = df.groupby('job')['credit_score'].apply(lambda  
x:x.isna().sum())  
  
na_after = df['credit_score'].isna().groupby(df['job']).sum()  
print(na_after)
```

```
# print(df['credit_score'].info())  
# credit_mean_job = df.groupby(by='job')['credit_score'].mean()
```

```
# df['credit_score'] =  
df['credit_score'].fillna(df['job'].map(credit_mean_job))  
  
# total_mean_credit = df['credit_score'].mean()  
  
# print(total_mean_credit)
```

'''

5. 저축률 파생 및 상위분석

저축률(savings_rate) = savings/income으로 파생변수를 만들고, 이 값이 가장 높은 상위 5명을 추출하시오

'''

```
df[ 'savings_rate'] = df['savings'] / df['income']  
df = df.sort_values('savings_rate', ascending=False)  
print(df[['ID', 'savings_rate']].head(5))
```

```
df[ 'savings_rate'] = df['savings'] / df['income']
```

```
top_5_savings_rate = df.sort_values(by='savings_rate', ascending=False)  
print(top_5_savings_rate[['ID', 'savings_rate']].head(5)[ 'ID'])
```

'''

6. 소비대비 대출 비율 분석

expenditure 대비 loan_amount 비율(loan_ratio)을 계산하고, 이 비율이 가장 높은 사람의 ID와 지역을 구하시오

'''

```
df[ 'loan_ratio'] = df['expenditure'] / df['loan_amount']  
  
# loan_ratio가 가장 높은 사람 찾기  
  
highest_loan_ratio = df.loc[df['loan_ratio'].idxmax()]
```

```
# 결과 출력
```

```
print(f'loan_ratio가 가장 높은 사람 : {highest_loan_ratio["ID"]}, 지역 : {highest_loan_ratio["region"]}')
```

```
## 설명
```

```
# index = df['loan_ratio'].idxmax()는 loan_ratio가 max인 인덱스를 반환한다.
```

```
# 그 index의 행 정보를 데이터프레임으로 가져오려면, df.loc[index]가 되어야한다.
```

```
### index 추출하기.
```

```
# idx로 찾을 수 있는 값들은 주로 데이터프레임의 특정 열에서 최대값, 최소값, 또는 특정 조건을 만족하는 인덱스를 찾는 데 사용됩니다. 다음은 일반적으로 사용되는 몇 가지 메서드입니다:
```

```
# idxmax(): 주어진 열에서 최대값의 인덱스를 반환합니다.
```

```
# max_index = df['column_name'].idxmax()
```

```
# idxmin(): 주어진 열에서 최소값의 인덱스를 반환합니다.
```

```
# min_index = df['column_name'].idxmin()
```

```
# 조건을 만족하는 인덱스 찾기: 특정 조건을 만족하는 행의 인덱스를 찾을 수 있습니다.
```

```
# condition_index = df[df['column_name'] > value].index
```

```
# first_valid_index(): 첫 번째 유효한 값의 인덱스를 반환합니다.
```

```
# first_valid_index = df['column_name'].first_valid_index()
```

```
# last_valid_index(): 마지막 유효한 값의 인덱스를 반환합니다.
```

```
# last_valid_index = df['column_name'].last_valid_index()
```

'''

7. 직업별 평균 신용점수와 연령 비교

job별 평균 신용점수를 계산하고, 신용점수가 가장 높은 직업군과 그들의 평균연령을 구하시오.

'''

```
job_credit = df.groupby('job')[ 'credit_score'].sum().reset_index()
high_job = job_credit.sort_values('credit_score',ascending=False).iloc[0]['job']
mean_age_high_job = df[df['job']==high_job]['age'].mean()
print(mean_age_high_job)
```

```
job_credit = df.groupby('job')[ 'credit_score'].mean()
```

```
# print(job_credit)
```

```
max_credit_job = job_credit.idxmax()
```

```
max_credit_filter_mean_age = df[df['job']==max_credit_job]['age'].mean()
print(max_credit_filter_mean_age)
```

'''

8. 연도별 대출금 총합

date를 기준으로 연도별로 그룹화하여 총 대출금(loan_amount)합계를 계산하시오.

'''

```
print(df.columns)
```

```
# 'date' 열을 datetime 형식으로 변환
```

```
df[ 'date' ] = pd.to_datetime(df['date'])
df[ 'yr' ] = df['date'].dt.year # year, month, day
```

```
yr_group_loan = df.groupby('yr')[['loan_amount']].max()
print(yr_group_loan)
```

'''

9. 상환기간 이상치 탐지

loan_term에서 사분위수를 기준으로 이상치를 판단하고, 이상치로 판단된 대출기간을 가진 레코드 수를 구하시오.

'''

```
Q1 = df['loan_term'].quantile(0.25)
```

```
Q3 = df['loan_term'].quantile(0.75)
```

```
IQR = Q3-Q1
```

```
upper = Q3 + 1.5*IQR
```

```
lower = Q1 - 1.5*IQR
```

```
df_upper_out = df[df['loan_term'] > upper]
```

```
df_lower_out = df[df['loan_term'] < lower]
```

```
outlier = len(df_upper_out) + len(df_lower_out)
```

```
print(outlier)
```

이상치 탐지

```
outliers = df[(df['loan_term'] < lower) | (df['loan_term'] > upper)]
```

```
# 이상치로 판단된 대출기간을 가진 레코드 수  
num_outliers = outliers.shape[0]  
  
# 결과 출력  
print(f"이상치로 판단된 대출기간을 가진 레코드 수: {num_outliers}")  
  
print(df.shape)
```

'''

10. 신용점수 상위자 분석

신용점수 상위 10%의 사람들 중 loan_type이 '신용대출'인 사람의 수를 구하시오

'''

```
# print(df.columns)  
# print(df['loan_type'])
```

```
line = df['credit_score'].quantile(0.9)  
credit_top10 = df[(df['credit_score'] >= line) & (df['loan_type'] == '신용대출')]  
print(len(credit_top10))
```

'''

11. 직업별 대출 평균과 상관관계

직업별 평균 대출금(loan_amount)와 평균 소비(expenditure)간의 상관계수를 구하시오.

'''

```
# 직업별 평균 대출금과 평균 소비를 계산하고 데이터프레임으로 변환
```

```
correlation_df = df.groupby('job').agg(  
    loan_amount=('loan_amount', 'mean'),  
    expenditure=('expenditure', 'mean'))  
.reset_index()
```

```
print(correlation_df)
```

```
# 상관계수 계산
```

```
correlation = correlation_df[['loan_amount', 'expenditure']].corr()  
print(correlation) # loan_amount와 expenditure 간의 상관계수
```

```
corr = correlation.corr().iloc[0, 1] # correlation 행렬에서의 행, 열 위치에 있는 값만 빼오기.
```

```
print("직업별 평균 대출금과 평균 소비 간의 상관계수:" , corr)
```

```
correlation_df = df.groupby('job').agg(loan_amount = ('loan_amount','mean'), expenditure = ('expenditure','mean')).reset_index()  
print(correlation_df)
```

```
# agg방식
```

```
job_df = df.groupby('job').agg(loan_amount = ('loan_amount', 'mean'))  
.reset_index()  
print(job_df)
```

```
#### 서브퀘스트 #####
```

```
#일반 그룹
```

```
job_df2 = df.groupby(['region', 'job'])[['loan_amount', 'expenditure']].mean().reset_index()
print(job_df2)
```

'''

12. 교육수준별 대출 위험도

교육수준(education_level)별로 late_payment_count의 평균을 구하고, 가장 위험도가 높은 교육수준을 구하라.

'''

```
edu_level = df.groupby('education_level')['late_payment_count'].mean()
print(edu_level.idxmax())
```

'''

13. 날짜 파생 및 계절 분석

date에서 월을 추출하여 season파생변수 생성하고 계절별 평균 대출금을 구하라.

'''

```
df['date'] = pd.to_datetime(df['date'])
df['M'] = df['date'].dt.month
```

```
df['season'] = "
```

for문으로 돌리기

```
for idx in df.index:
```

```
    i = df.loc[idx, 'M']
```

```
    if i in [3,4,5]:
```

```
        df.loc[idx,'season'] = '봄'
```

```
elif i in [6,7,8]:  
    df.loc[idx,'season'] = '여름'  
  
elif i in [9,10,11]:  
    df.loc[idx,'season'] = '가을'  
  
elif i in [12,1,2]:  
    df.loc[idx,'season'] = '겨울'
```

```
print(df[['M', 'season']])
```

```
## 함수를 만들고 함수를 apply 하기
```

```
def get_season(month):  
  
    if month in [3,4,5]:  
        return '봄'  
  
    elif month in [6,7,8]:  
        return '여름'  
  
    elif month in [9,10,11]:  
        return '가을'  
  
    else:  
        return '겨울'
```

```
df['season'] = df['M'].apply(get_season)
```

```
print(df[['M', 'season']])
```

```
mean_loan_season = df.groupby('season')['loan_amount'].mean()  
print(mean_loan_season)
```

'''

14. 성별 소비 총합 비교

성별(gender)별로 총 expenditure를 계산하고, 더 많이 소비한 성별과 그 총합을 구하라.

'''

```
gen_exp = df.groupby('gender')['expenditure'].sum()  
print(gen_exp)
```

```
max_gender = gen_exp.idxmax() # 소비 가장 큰 성별
```

```
max_value = gen_exp.max() # 그 소비액
```

```
print(f'가장 많이 쓴 성별: {max_gender}, 총 소비액: {max_value}')
```

'''

15. 신용불량 가능성 판단

credit_score가 500미만이면서 late_payment_count가 5 이상인 사람을 신용불량 가능성 있음으로 간주하자.

해당 인원수는?

'''

```
df_low_credit = df[(df['credit_score'] < 500) & (df['late_payment_count']  
>= 5)]
```

```
print(len(df_low_credit))
```

'''

16. 직업성별 조합별 평균대출

job,gender 조합별 평균 대출금(loan_amount)을 계산하고, 가장 평균이 높은 조합을 구하라.

'''

```
job_gender_combi = df.pivot_table(index = ['job'], columns = ['gender'],  
values = ['loan_amount'], aggfunc = 'mean')
```

```
print(job_gender_combi)
```

```
print(job_gender_combi.idxmax()) ## 시리즈 아닌 데이터프레임 형식이라 gender별로 각 최대조합을  
찾는다.
```

stack을 사용해서 시리즈타입으로 바꿔주고 하면 전체에서 최대조합을 찾는다.

```
max_combination = job_gender_combi.stack().idxmax()
```

```
max_value = job_gender_combi.stack().max()
```

groupby로 하기.

```
job_gender_avg = df.groupby(['job', 'gender'])['loan_amount'].mean()
```

```
max_combination = job_gender_avg.idxmax()
```

```
max_value = job_gender_avg.max()
```

```
print(job_gender_combi)
```

```
print(f"가장 평균 대출금이 높은 조합: {max_combination}, 평균 대출금: {max_value}")
```

'''

17. 저축률 상위 10% 연령분석

savings/income 상위 10%의 사람들의 평균나이를 구하시오.

'''

```
df['saving_ratio'] = df['savings'] / df['income']
```

```
thresholds = df['saving_ratio'].quantile(0.9)
```

```
top10 = df[df['saving_ratio'] > thresholds]
print(top10['age'].mean())
```

'''

18. 결측보간후 분석

savings 결측치를 선형보간법으로 채운 후 그 평균을 다시 계산하라.

'''

```
file_path =
'C:/Python/PyProject/24년프로젝트/12.빅분기실기/빅분기/data/유형모음/bigdata_1type_master_wit
h_nan.csv'

df = pd.read_csv(file_path)
print(df['savings'].info())

# 1. savings 결측치 선형 보간
df['savings'] = df['savings'].interpolate(method='linear')

# 2. 보간 후 savings 평균 계산
new_average = df['savings'].mean()
print(f"결측치 보간 후 savings의 평균: {new_average:.2f}")

# #####참고
# # fillna: 앞의 값으로 채움
# df['savings'].fillna(method='ffill')
# # fillna: 뒤의 값으로 채움
# df['savings'].fillna(method='bfill')
```

'''

19. 소비저축비율파생 및 그룹화

expenditure/savings 값을 새로운 파생열로 생성하고, 이 값을 기준으로 상위 3개 직업군을 구하라.

'''

```
df[ 'exp_sav' ] = df['expenditure'] / df['savings']
df_sorted = df.sort_values(by ='exp_sav',ascending = False)
print(df_sorted['job'].iloc[:3])
```

상위 3개 직업군 (중복 제거 방식)

```
top_3_jobs = df_sorted['job'].drop_duplicates().head(3)
print(top_3_jobs)
```

'''

20. 연도별 소득증가분석

연도별 income의 평균을 구하고, 가장 소득이 상승한 연도 구간을 찾아 상승률도 함께 구하라.

'''

```
df[ 'date' ] = pd.to_datetime(df['date'])
df[ 'year' ] = df['date'].dt.year
yr_income = df.groupby('year')[ 'income' ].mean().reset_index()
print(yr_income)
```

```
yr_income['rising_gap'] =
yr_income['rising_rate'] =
```

상승 차이 및 상승률 계산

```
yr_income['rising_gap'] = yr_income['income'].diff()
```

```
yr_income['rising_rate'] = yr_income['income'].pct_change()
```

```
idx = yr_income['rising_gap'].idxmax()
```

```
print(f"가장 소득이 상승한 연도구간 : {yr_income['year'].loc[idx-1]} → {yr_income['year'].loc[idx]}")
```

```
print(f"해당 상승률 : {yr_income['rising_rate'].loc[idx]*100}")
```

```
# year
```

```
# 2010    4825.878909
```

```
# 2011    4699.845882
```

```
# 2012    4716.698000
```

```
# 2013    4731.821081
```

```
# 2014    5028.956341
```

```
# 2015    4820.120455
```

```
# 2016    4640.424909
```

```
# 2017    4857.310208
```

```
# 2018    4796.060682
```

```
# 2019    4830.498235
```

9회실기 1유형-1

지역코드별 성별에 따라 정리 된 개인 대출 데이터이다.

지역코드별 성별에 따른 총 대출액을 구하고, 성별간 총 대출액 차이가 큰 지역의 지역코드 값을 구하라

(단, 총 대출액은 K대출과 W대출을 더한 값이다.)

'''

```
df91 = pd.read_csv('C:/Python/PyProject/빅분기/기출 및 필수모의/9회/loan(9회1-1).csv')
print(df91)
```

```
df91['loan_sum'] = df91['K은행대출'] + df91['W은행대출']
```

```
gender_sum = df91.pivot_table(index='지역코드', columns='성별', values
='loan_sum', aggfunc='sum')

print(gender_sum)

gender_sum['diff'] = abs(gender_sum['남자'] - gender_sum['여자'])

print(gender_sum['diff'].idxmax())
```

'''

9회실기 1유형-2

Arrest.csv 데이터는 연도위르 범죄유형별, 범죄건수, 검거건수 데이터다.

검거율 = 검거건수/범죄건수로 새로운 열 추가하고 연도, 범죄유형별로 검거율 구한 후

```
# 각 연도별 최대 검거율 (검거율 = 검거건수 / 발생건수) 을 가진 범죄유형을 찾아,
# 해당 연도 및 유형의 검거건수들의 총합을 구하라.
```

데이터는 ym, crime, cases, arrest 로 되어있다.,

'''

```
df92 = pd.read_csv('C:/Python/PyProject/빅분기/기출 및 필수모의/9회/arrest(9회1-2).csv')
print(df92)
```

```
df92['검거율'] = df92['arrest'] / df92['cases']
```

```
print(df92['ym'].unique())
```

```
# 월을 두 자리로 보정한 뒤 다시 문자열로 결합
```

```
df92['ym'] = df92['ym'].astype(str).str.split('.').apply(lambda x: f'{x[0]}.{x[1].zfill(2)}')
```

```
df92['ym'] = pd.to_datetime(df92['ym'], format='%Y.%m')
```

```
df92['year'] = df92['ym'].dt.year
```

```
print(df92)
```

```
new_df92 = df92.groupby(['year','crime'])['검거율'].mean()
```

```
print(new_df92)
```

```
print(new_df92.idxmax()) # 2009, 강간
```

```
filter_df92 = df92[(df92['year']==2009) & (df92['crime'] == '강간')]
```

```
print(filter_df92['arrest'].sum())
```

'''

9회실기기 1유형3

Years.csv는 (ID, dept, year, amount, grade, satisfy) = (사번, 부서, 근속연수, 연봉, 성과등급, 교육만족도) 이다.
다음결과를 출력하라

- 1) 교육만족도의 결측값을 전체 교육만족도의 평균값으로 대체하시오
- 2) 근속연수의 결측값을 (부서, 성과등급)별 근속연수의 평균값으로 대체하시오. 단, 근속연수의 평균값을 정수(소수점이하버림)으로 대체하시오.
- 3) $a = \text{연봉}/\text{근속연수}$ 를 새로운 열로 추가하고, a의 값이 3번째로 큰 사람의 근속연수를 출력하시오.
단, 근속연수는 정수(소수점 이하 버림)으로 출력하시오.
- 4) $b = \text{연봉}/\text{교육만족도}$ 를 새로운 열로 추가하고, b의 값이 2번째로 큰 사람의 교육만족도를 출력하시오.
- 5) 3)+4)의 값을 출력하시오.

'''

```
df93 = pd.read_csv('C:/Python/PyProject/빅분기/기출 및 필수모의/9회/Years_expanded(9회1-3).csv')
```

```
df93['year'] = abs(df93['year'])  
df93.loc[df93['year'] == 0, 'year'] = np.nan
```

#1

```
satisfy_mean = df93['satisfy'].dropna().mean()  
df93['satisfy'] = df93['satisfy'].fillna(satisfy_mean)
```

#2

```
df93['group_mean'] = df93.groupby(['dept','grade'])['year'].transform('mean')  
df93['group_mean'] = df93['group_mean'].fillna(df93['group_mean'].dropna().mean())  
df93['group_mean'] = df93['group_mean'].astype(int)  
df93['year'] = df93['year'].fillna(df93['group_mean'])  
print(df93)
```

#3

```
df93['a'] = df93['amount'] / df93['year']

rank3_year = int(df93.sort_values(by='a', ascending=False).iloc[2]['year'])

print(rank3_year)
```

#4

```
df93['b'] = df93['amount'] / df93['satisfy']

rank2_satisfy = df93.sort_values(by='b').iloc[1]['satisfy']

print(rank2_satisfy)
```