국가별 음주 데이터 분석하기

step1. 탐색: 데이터의 기초 정보 살펴보기

In [1]:

```
#라이브러리 임포트
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
file_path = '../data/drinks.csv' # 불러올 파일을 상대 경로를 지정, ../는 부모폴더
drinks = pd.read_csv(file_path) # read_csv() 함수로 데이터를 데이터 프레임 형태로 불러옴
drinks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   country                       193 non-null    object 
 1   beer_servings                 193 non-null    int64  
 2   spirit_servings               193 non-null    int64  
 3   wine_servings                 193 non-null    int64  
 4   total_litres_of_pure_alcohol  193 non-null    float64
 5   continent                     170 non-null    object 
dtypes: float64(1), int64(3), object(2)
memory usage: 9.2+ KB
```

In [3]:

```
#파일을 위에서 10개만 보여줌
drinks.head(10)
```

Out[3]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7

In [4]:

```
# 피처의 통계형 수치적 정보 (갯수, 평균, 표준편차, 최소값, 최대값)
drinks.describe()
```

Out[4]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
count	193.000000	193.000000	193.000000	193.000000
mean	106.160622	80.994819	49.450777	4.717098
std	101.143103	88.284312	79.697598	3.773298
min	0.000000	0.000000	0.000000	0.000000
25%	20.000000	4.000000	1.000000	1.300000
50%	76.000000	56.000000	8.000000	4.200000
75%	188.000000	128.000000	59.000000	7.200000
max	376.000000	438.000000	370.000000	14.400000

Step2. 인사이트의 발견: 탐색과 시각화하기

피처 간의 상관관계

beer_servings, wine_servings 두 피처 간의 상관 계수 계산

1) 피어슨 상관계수(pearson correlation coefficient,PCC): 두 변수 X,Y 간의 선형 상관 관계를 계량화한 수치

corr() 함수로 피처 간의 상관 계수를 매트릭스(행렬, 숫자, 기호 등을 가로, 세로로 나열)형태로 출력할 수 있다

2) 스피어먼 상관계수: 두 변수 순위 사이의 통계적 의존성을 측정하는 비모수적인 척도

피처간의 상관 관계를 통계적으로 탐색하는 방법

- 단순 상관 분석방법: 피처가 2개 일때 상관 계수를 계산하는 방법
- 다중 상관 분석 방법: 피처가 여러 개일 때 상호간의 연관성을 분석하는 방법

In [5]:

```
#단순 상관 분석방법
#beer_servings', 'wine_servings 두 개의 상관관계
corr = drinks[['beer_servings', 'wine_servings']].corr(method = 'pearson')
corr
```

Out[5]:

	beer_servings	wine_servings
beer_servings	1.000000	0.527172
wine_servings	0.527172	1.000000

여러 피처의 상관관계 분석하기 피처 간의 상관 계수 행렬을 구한다

In [35]:

```
# 다중 상관 분석방법
corr1=drinks.corr(method='pearson') #피어슨
#corr1=[['beer_servings', spirit_servings', 'wine_servings', 'total_litres_of_pure_alcc
#drinks.corr(method='spearman') #스피어만
corr1
```

Out[35]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pu
beer_servings	1.000000	0.458819	0.527172	
spirit_servings	0.458819	1.000000	0.194797	
wine_servings	0.527172	0.194797	1.000000	
total_litres_of_pure_alcohol	0.835839	0.654968	0.667598	

[seaborn 시각화 라이브러리 활용]

- 히트맵(heatmap), 페어플롯(pairplot)기법 사용하기

- 히트맵: 히트와 지도를 뜻하는 맵을 결합시킨 단어로 색상으로 표현할 수 있는 다양한 정보를 일정한 이미지 위에 열분포 형태의 그래픽으로 출력

In [7]:

```
!pip install seaborn #seaborn 설치
```

ERROR: Invalid requirement: '#seaborn'

In [8]:

```
#seaborn 임포트
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:

```
#set() 폰트 사이즈 지정
sns.set(font_scale = 1) #label 폰트의 크기
# 그래프 출력을 위한 cols의 이름 지정
cols_view = ['beer', 'spirit', 'wine', 'alcohol']
# seaborn의 hearmap(),의 파라미터1번째는 어떤 값을 가져오는지 그 다음은 형태
hm = sns.heatmap(corr1.values, #히트맵에 출력하고자 하는 값(다중상관 관계 프레임값)
        cbar = True, # 히트맵 오른쪽 바 표시
        annot = True, #네모 위의 값 표시(상관계수 출력 여부)
        square = True, #사각형의 형태(True 는 정사각형 False는 직사각형 )
        fmt = '.2f', #소숫점 자릿수, 두 자리까지 표시
        annot_kws = {'size':15}, #사각형 안 글자 크기(상관계수 폰트 크기)
        yticklabels = cols_view, # cols_view가 y을 레이블명
        xticklabels = cols_view, # cols_view가 x을 레이블명
         linewidths= 1.5, # 상관 변수 사이에 흰색 선
         cmap="YlGnBu", # 컬러맵
        cbar_kws={"orientation": "horizontal"}) #cbar를 가로로 표시

plt.tight_layout()
plt.show()
```

-pairplot 그래프

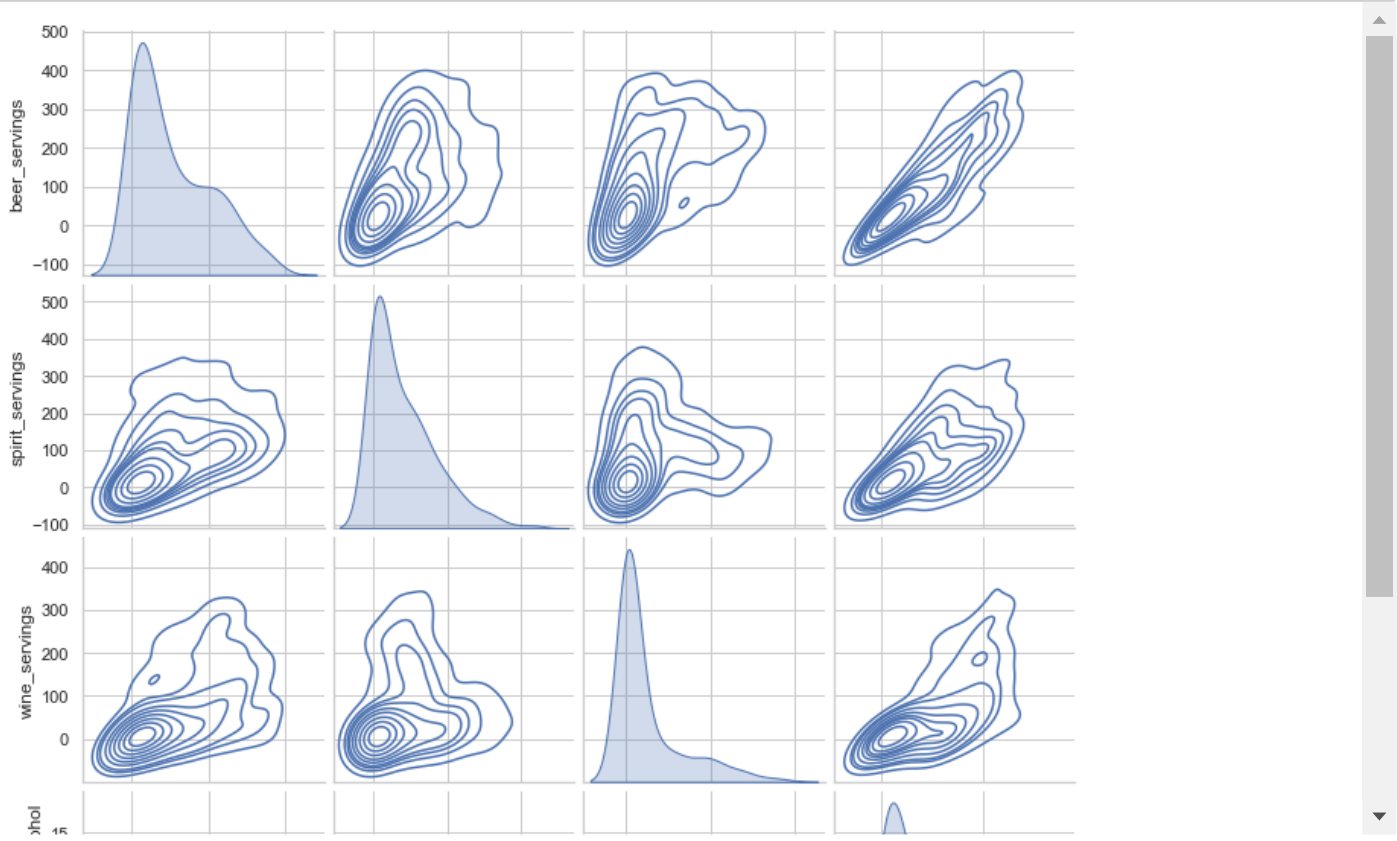
페어플롯은 데이터 프레임을 파라미터로 넣어줌

In [10]:

```
#피처 간의 산점도 그래프 출력
#set() 함수는 seaborn 플롯의 테마 및 구성을 제어하는 데 사용

#페어플롯은 데이터 프레임을 파라미터로 넣어줌
#style: darkgrid, whitegrid, dark, white,ticks : 이미지의 전반적인 모양을 스타일링, 5가지 
#context: paper, notebook, talk, poster : 어떤 상황에서 보여줄 것인가에 따라 4종류의 텍스

sns.set(style = 'whitegrid', context = 'notebook')
sns.pairplot(drinks[['beer_servings', 'spirit_servings', 'wine_servings', 'total_litres_o
plt.show()
```



인사이트 도출 1. 대륙별 평균 wine_servings 탐색

In [11]:

```
#대륙별 평균을 구한 후 wine_servings 인사이트 도출
wineAve = drinks.groupby('continent').mean()['wine_servings'] #방법1
wineAve
#wineAve = drinks.groupby('continent')['wine_servings'].mean() #방법2
```

Out[11]:

```
continent
AF    16.264151
AS     9.068182
EU   142.222222
OC    35.625000
SA    62.416667
Name: wine_servings, dtype: float64
```

인사이트 도출2. 전체 평균보다 적은 알코올을 섭취하는 대륙

In [12]:

```
#대륙 알콜 평균(totalmean) 과  대륙별 알콜 전체 섭취량(continent_mean) 비교
totalmean = drinks.total_litres_of_pure_alcohol.mean()

continent_mean = drinks.groupby('continent')['total_litres_of_pure_alcohol'].mean(
#continent_mean = drinks.groupby('continent').mean()['total_litres_of_pure_alcoho

#index.tolist()  인덱스만 추출해서 리스트로 만드는 함수
# 대륙 알콜 평균이 전체 평균보다 작은 값의 인덱스만 뽑아서 리스트로 만들어서 continent_und
continent_under_mean = continent_mean[continent_mean < totalmean].index.tolist

#values.tolist() 값을 추출해서 리스트로 만드는 함수
continent_under_mean = continent_mean[continent_mean < totalmean].values.tolis
continent_under_mean
```

Out[12]:

```
[3.0075471698113208, 2.1704545454545454, 3.38125]
```

Step3. 탐색적 분석

In [13]:

```
drinks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   country                       193 non-null    object 
 1   beer_servings                 193 non-null    int64  
 2   spirit_servings               193 non-null    int64  
 3   wine_servings                 193 non-null    int64  
 4   total_litres_of_pure_alcohol  193 non-null    float64
 5   continent                     170 non-null    object 
dtypes: float64(1), int64(3), object(2)
memory usage: 9.2+ KB
```

결측데이터 처리: 기타 대륙으로 통합 -> 'OT' (Others)

fillna() 함수는 피처의 결측값을 특정 값으로 채워주는 함수

In [14]:

```
drinks['continent'] = drinks['continent'].fillna('OT')
```

In [15]:

```
drinks.head(10)
```

Out[15]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7

In [16]:

```
drinks['continent'].value_counts()
```

Out[16]:

```
AF   53
EU   45
AS   44
OT   23
OC   16
SA   12
Name: continent, dtype: int64
```

In [17]:

```
drinks['continent'].value_counts().index
```

Out[17]:

```
Index(['AF', 'EU', 'AS', 'OT', 'OC', 'SA'], dtype='object')
```

전체 대륙 중에서 OT 가 차지하는 비율을 파이차트로 확인

In [18]:

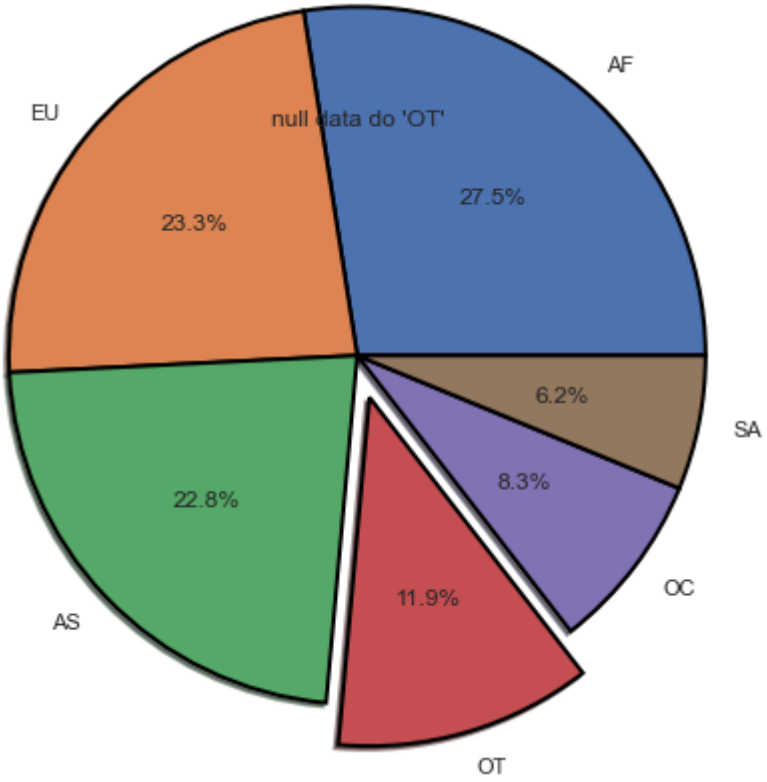
```
#그래프를 표시하기 위한 인덱스와 값을 추출해서 리스트로 생성

#drinks의 continent행의 밸류값을 카운트한다-> 인덱스만 뽑아서 -> 리스트로 만들기
labels = drinks['continent'].value_counts().index.tolist()
#drinks의 continent행의 밸류값을 카운트한다-> 밸류값만 뽑아서 -> 리스트로 만들기
fracs1 = drinks['continent'].value_counts().values.tolist()

explode = (0, 0, 0, 0.25, 0, 0) # 파이차이에 원하는 파이가 분리될 수 있도록
```

In [19]:

```
#radius 반지름 값 ,
plt.pie(fracs1, labels = labels, autopct = '%.1f%%', shadow = True, explode = explode,
plt.title('null data do \'OT\'')
plt.show()
```



그룹 단위의 데이터 분석 : 대륙별 분석

apply, agg 함수를 이용한 대륙별 분석

대륙별 spirit_servings 의 평균, 최소, 최대, 합계를 계산

In [20]:

```
drinks.groupby('continent').spirit_servings.describe()
```

Out[20]:

continent	count	mean	std	min	25%	50%	75%	max
AF	53.0	16.339623	28.102794	0.0	1.00	3.0	19.00	152.0
AS	44.0	60.840909	84.362160	0.0	1.00	16.0	98.00	326.0
EU	45.0	132.555556	77.589115	0.0	81.00	122.0	173.00	373.0
OC	16.0	58.437500	70.504817	0.0	18.00	37.0	65.25	254.0
OT	23.0	165.739130	94.993884	68.0	101.00	137.0	190.50	438.0
SA	12.0	114.750000	77.077440	25.0	65.75	108.5	148.75	302.0

In [21]:

```
drinks.groupby('continent').spirit_servings.max()
```

Out[21]:

```
continent
AF    152
AS    326
EU    373
OC    254
OT    438
SA    302
Name: spirit_servings, dtype: int64
```

agg() 함수는 apply() 함수와 거의 동일하게 함수 파라미터를 받음

agg() 함수 파라미터를 병렬로 설정하여 그룹에 대한 여러가지 연산 결과를 동시에 얻을 수 있는 함수

agg() 함수를 이용해 mean, max, min, sum 함수 파라미터를 간단히 탐색

In [22]:

```
result = drinks.groupby('continent').spirit_servings.agg(['mean', 'max', 'min', 'sum'])
result
```

Out[22]:

continent	mean	max	min	sum
AF	16.339623	152	0	866
AS	60.840909	326	0	2677
EU	132.555556	373	0	5965
OC	58.437500	254	0	935
OT	165.739130	438	68	3812
SA	114.750000	302	25	1377

2. 전체 평균보다 많은 알코올을 섭취하는 대륙

- 2-1: 전체 알코올 섭취 평균
- 2-2: 대륙별 알코올 섭취 평균
- 2-3: 비교해서 대륙별 알코올 섭취 평균이 전체 알코올 섭취 평균보다 낮은 대륙을 구하기

In [23]:

```
# 2-1: 전체 알코올 섭취 평균
total_mean = drinks.total_litres_of_pure_alcohol.mean()
total_mean
```

Out[23]:

```
4.717098445595855
```

In [24]:

```
# 2-2: 대륙별 알코올 섭취 평균
#drinks 데이터 프레임 -> 대륙을 groupby -> total_litres_of_pure_alcohol 전체 알코올 섭취
continent_mean = drinks.groupby('continent').total_litres_of_pure_alcohol.mean()
#drinks.groupby('continent')[total_litres_of_pure_alcohol]mean() 위 코드와 같은 의미
continent_mean
```

Out[24]:

```
continent
AF    3.007547
AS    2.170455
EU    8.617778
OC    3.381250
OT    5.995652
SA    6.308333
Name: total_litres_of_pure_alcohol, dtype: float64
```

In [25]:

```
# 2-3: 비교해서 대륙별 알코올 섭취 평균이 전체 알코올 섭취 평균보다 낮은 대륙을 구하기
# 데이터프레임[조건]
continent_mean[continent_mean < total_mean]
```

Out[25]:

```
continent
AF    3.007547
AS    2.170455
OC    3.381250
Name: total_litres_of_pure_alcohol, dtype: float64
```

3. 평균 beer_servings이 가장 높은 대륙

idxmin, idxmax 는 전체 인덱스 중 최소값, 최대값을 반환

- 3-1 : 대륙별 평균 beer_servings 계산
- 3-2 : 결과 중 값이 가장 높은 인덱스만 추출

In [26]:

```
drinks.groupby('continent').beer_servings.mean()
```

Out[26]:

```
continent
AF    61.471698
AS    37.045455
EU   193.777778
OC    89.687500
OT   145.434783
SA   175.083333
Name: beer_servings, dtype: float64
```

In [27]:

```
drinks.groupby('continent').beer_servings.mean().idxmin()
```

Out[27]:

```
'AS'
```

In [28]:

```
drinks.groupby('continent').beer_servings.mean().idxmax()
```

Out[28]:

```
'EU'
```

한 그래프에 여러개의 막대를 표현@@

대륙별 spirit_servings 평균, 최소, 최대 , 합계를 시각화합니다

In [29]:

```
n_groups = len(result.index)
n_groups
```

Out[29]:

```
6
```

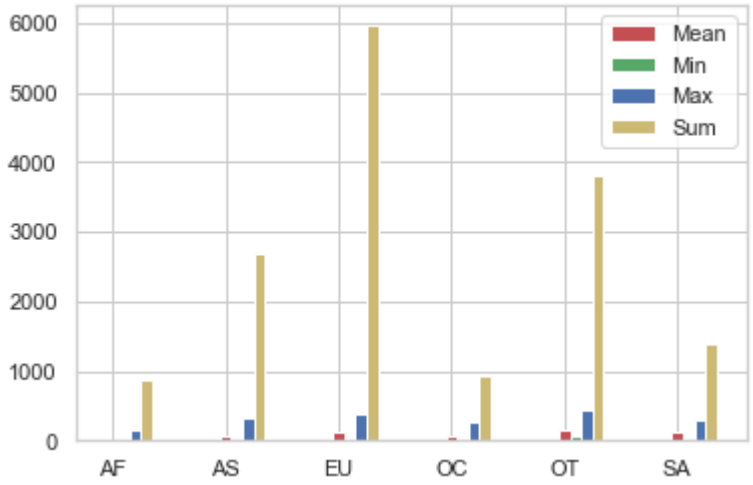
In [34]:

```
means = result['mean'].tolist()
mins = result['min'].tolist()
maxs = result['max'].tolist()
sums = result['sum'].tolist()

index = np.arange(n_groups) # n_groups 의 크기 6만큼 배열 생성하여 index에 저장
bar_width = 0.1

#평균값에 대한 바 생성
rects1 = plt.bar(index, means, bar_width, color='r', label = 'Mean')
rects2 = plt.bar(index + bar_width, mins, bar_width, color='g', label = 'Min')
rects3 = plt.bar(index + bar_width *2, maxs, bar_width, color='b', label = 'Max')
rects4 = plt.bar(index + bar_width *3,  sums, bar_width, color ='y', label = 'Sum')

plt.xticks(index, result.index.tolist())
plt.legend() #범례
plt.show()
```



In [ ]: