
Ch08 그래프 만들기

```
#.scatterplot() #.scatter() #.barplot() #.countplot() #.hist() #.pie() #.plot() #.lineplot() #.boxplot() #.corr() #.heatmap()  
#.rcParams #.reindex() #.to_datetime() #.rename() #.grid() #.figure() #.xlabel() #.ylabel() #.legend()  
#.quantile() #.round() #.pivot() #.zeros_like() #.triu_indices_from() #.add_node() #.add_edge() #.add_nodes_from()  
#.add_edges_from() #.DiGraph() #.subplots() #.spring_layout() #.draw() #.title() #..degree #.values()  
#열변수명추출 #.normalize
```

08-1 파이썬으로 만들 수 있는 그래프 살펴보기

- > 막대, 선, 파이, 상자, 산점도, 히스토그램, 등고선 그래프 등
- > 워드 클라우드, 지도 그래프, 네트워크 그래프, 모션 차트, 인터랙티브 그래프 등

Library를 통해 그릴 수 있는 그래프들

1. **Matplotlib**: Matplotlib은 Python의 가장 인기 있는 그래픽 라이브러리 중 하나로, 선 그래프, 막대 그래프, 히스토그램, 산점도, 등고선 그래프 등 다양한 종류의 그래프를 그릴 수 있습니다.
2. **Seaborn**: Seaborn은 Matplotlib을 기반으로 한 통계적 데이터 시각화 라이브러리로, 예쁘고 간편한 API를 제공하여 복잡한 시각화를 빠르게 생성할 수 있습니다. 주로 히트맵, 박스 플롯, 카운트 플롯 등에 사용됩니다.
3. **Plotly**: Plotly는 상호작용 그래프를 생성하는데 사용되는 라이브러리로, 웹 기반 대화식 그래프를 만들 수 있습니다. 특히 대시보드 및 웹 애플리케이션 개발에 유용합니다.
4. **Bokeh**: Bokeh는 대화식 시각화를 위한 강력한 라이브러리로, 웹 기반 대시보드 및 애플리케이션을 만들기에 적합합니다.
5. **Pandas Plotting**: Pandas 자체에서도 데이터프레임에 포함된 데이터를 기반으로 그래프를 그릴 수 있는 기능을 제공합니다. DataFrame.plot() 메서드를 사용하여 다양한 종류의 그래프를 그릴 수 있습니다.
6. **Altair**: Altair는 선언적 문법을 사용하여 인터랙티브한 그래프를 만들기 위한 라이브러리입니다. 데이터를 시각화하기 위한 간단하고 직관적인 접근 방식을 제공합니다.
7. **ggplot**: ggplot은 R 언어에서 영감을 받아 만든 Python 패키지로, Grammar of Graphics 원칙에 따라 그래프를 생성합니다.

8. Holoviews: Holoviews는 복잡한 시각화 작업을 간단하게 만들어주는 라이브러리로, Matplotlib, Bokeh, Plotly 등 다양한 백엔드와 함께 사용할 수 있습니다.

9. NetworkX: NetworkX는 그래프 이론 및 네트워크 분석을 위한 라이브러리로, 그래프 구조를 시각화하는데 사용됩니다.

10. D3.js: D3.js는 JavaScript 기반의 데이터 시각화 라이브러리이지만, Python과 연동하여 사용할 수도 있으며, 웹 기반 대화식 시각화를 만드는데 매우 강력합니다.

※ 본 장에서는 Seaborn 라이브러리를 활용한 그래프 그리기에 중점으로 다룸

> Matplotlib는 가장 오래된 파이썬 시각화 라이브러리 중 하나여서 사용자가 그림의 많은 측면을 수동으로 설정해야 하는 면이 많음.

> Seaborn은 Matplotlib의 상위 계층에 구축된 통계 데이터 시각화 라이브러리로서 Matplotlib에 비해 간단한 인터페이스를 제공하며, 통계적 그래픽스를 위한 더 높은 수준의 추상화를 제공함

> Pandas Plot은 Pandas 데이터프레임 객체에 내장된 시각화 기능을 제공하고, Matplotlib을 사용하는 데 필요한 작업 처리 위주로 구성되어 그래프 유형이 일부 제한됨

In []:

08-2 산점도 - 변수 간 관계 표현하기

산점도 만들기

> 데이터의 두 변수 간의 관계를 시각화하기 위해 사용되는 간단하면서도 매우 유용한 그래프

In [2]:

```
## mpg 데이터 불러오기
import pandas as pd

mpg = pd.read_csv('mpg.csv')
mpg
```

Out[2]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

234 rows × 11 columns

seaborn.scatterplot()으로 그리기

- > **data** : 데이터 프레임 또는 배열 데이터 (필수)
- > **x** 및 **y** : x와 y 축에 사용될 데이터 열의 이름이나 위치 (필수)
- > **hue**: 데이터 포인트의 색상으로 구분할 수 있는 열의 이름
- > **style**: 데이터 포인트의 스타일을 지정할 열의 이름
- > **size**: 데이터 포인트의 크기를 지정할 열의 이름
- > **palette**: 색상 팔레트를 지정하여 데이터 포인트의 색상을 제어합니다.
- > **hue_order** 및 **hue_norm**: hue에 지정된 열의 순서 및 정규화
- > **size_order** 및 **size_norm**: size에 지정된 열의 순서 및 정규화
- > **sizes**: 데이터 포인트의 크기 범위

> **markers**: 스타일을 지정하는 데 사용할 마커 종류

> **legend**: 범례 표시 여부

> **ax**: 그림을 그릴 **Matplotlib** 축을 지정

[연습] **displ**과 **hwy** 간의 산점도 그래프 그리기

x축은 **displ**, y축은 **hwy**를 나타냄

In [3]: `## scatterplot()로 산점도 그래프 그리기`

```
import seaborn as sns
```

```
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

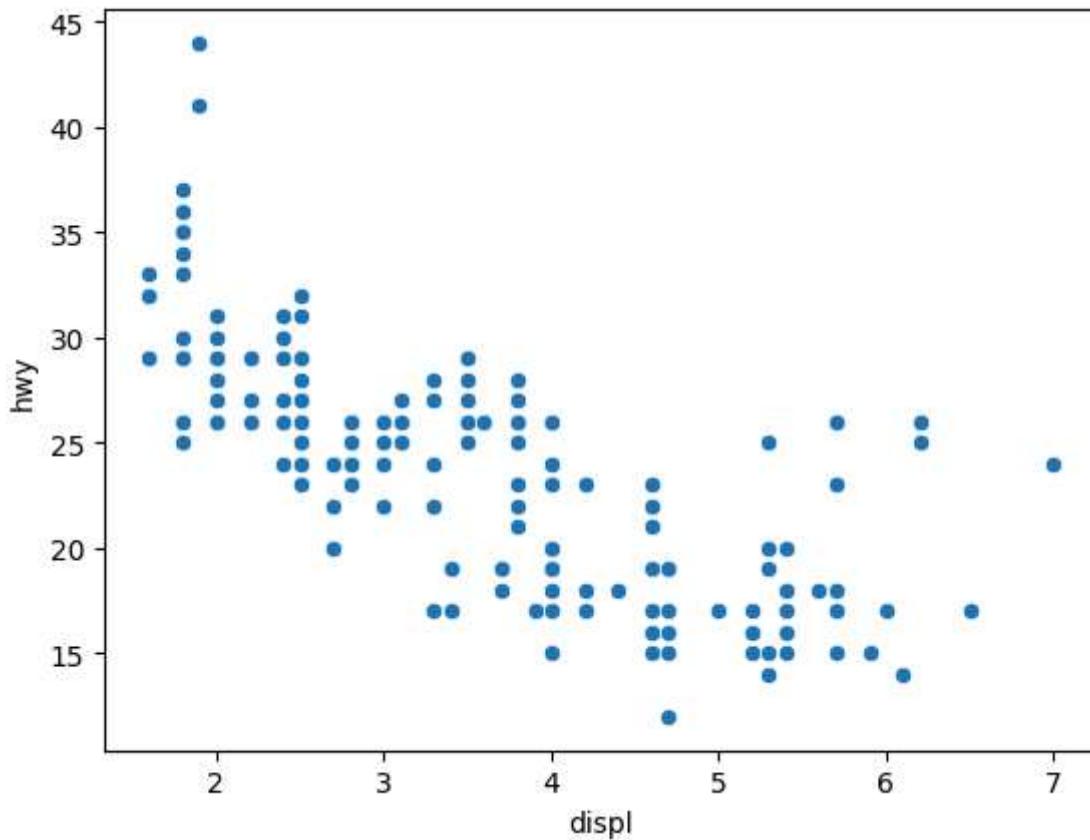
```
    if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
    if pd.api.types.is_categorical_dtype(vector):
```

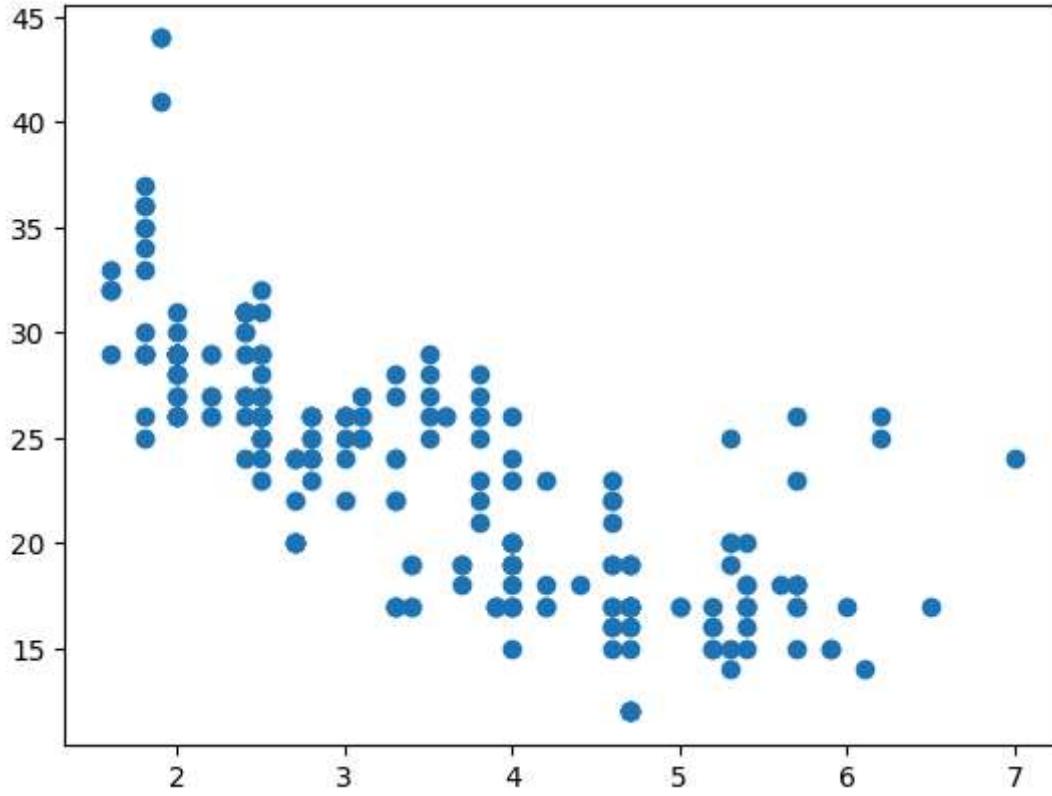
```
<Axes: xlabel='displ', ylabel='hwy'>
```

Out[3]:



```
In [10]: ## matplotlib 라이브러리로 그리기
import matplotlib.pyplot as plt

plt.scatter(mpg['displ'], mpg['hwy'])
plt.show()
```



'displ' 열 값 분포 확인

```
In [5]: ## 'displ' 열 값 분포 확인
import seaborn as sns

mpg['displ'].sort_values().value_counts(sort=False) # 'displ' 열 값을 추출해서 정렬한 다음 개수 구하기(최종 결과 정령 금지)
```

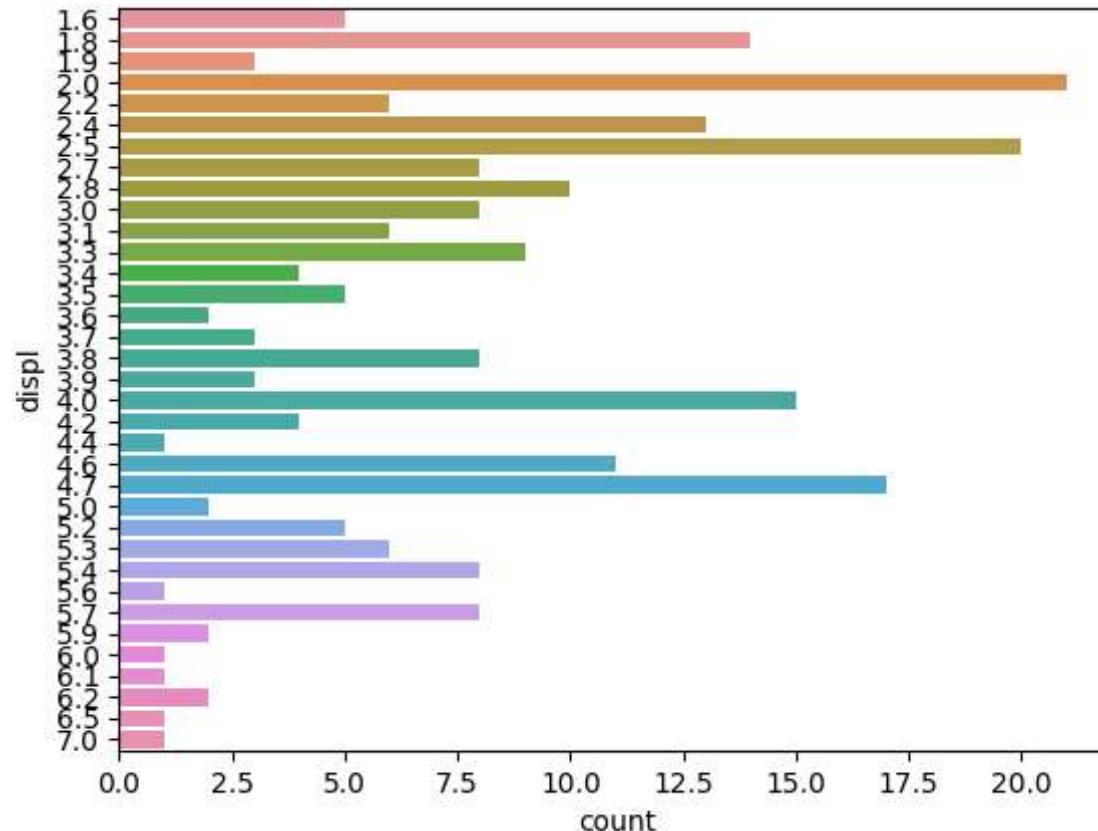
```
Out[5]: displ
1.6      5
1.8     14
1.9      3
2.0     21
2.2      6
2.4     13
2.5     20
2.7      8
2.8     10
3.0      8
3.1      6
3.3      9
3.4      4
3.5      5
3.6      2
3.7      3
3.8      8
3.9      3
4.0     15
4.2      4
4.4      1
4.6     11
4.7     17
5.0      2
5.2      5
5.3      6
5.4      8
5.6      1
5.7      8
5.9      2
6.0      1
6.1      1
6.2      2
6.5      1
7.0      1
Name: count, dtype: int64
```

```
In [11]: ## 빈도 막대 그래프로 분포 확인하기
import seaborn as sns

sns.countplot(data = mpg, y = 'displ')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[11]: <Axes: xlabel='count', ylabel='displ'>



축 범위 설정하기

In [12]: `## x축 범위 3~6으로 제한, y축 범위 10~30으로 제한
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy')`

```
.set(xlim = [3, 6], ylim = [10, 30])
```

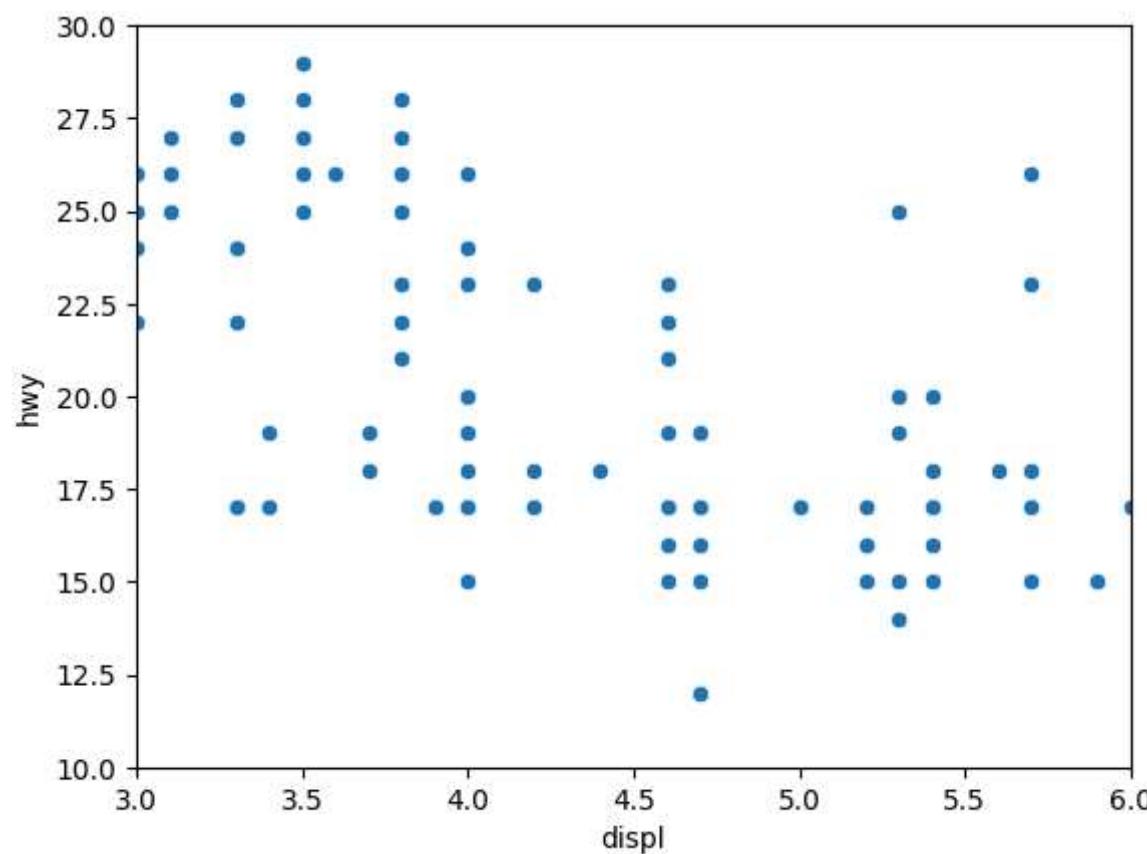
```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
    if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
    if pd.api.types.is_categorical_dtype(vector):
```

```
Out[12]: [(3.0, 6.0), (10.0, 30.0)]
```



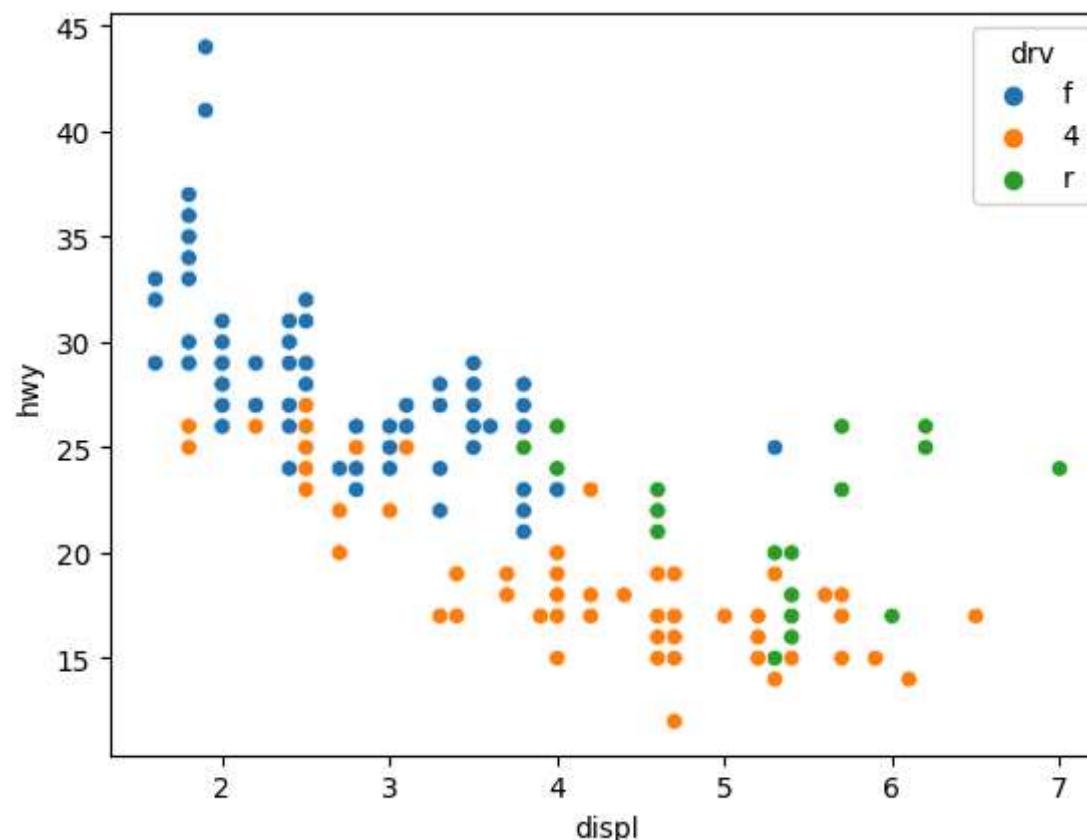
종류별로 표식 색깔 바꾸기

```
In [4]: ## drv별로 표식 색깔 다르게 표현
```

```
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy', hue = 'drv') #색조(hue)는 'drv'로 구분
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
```

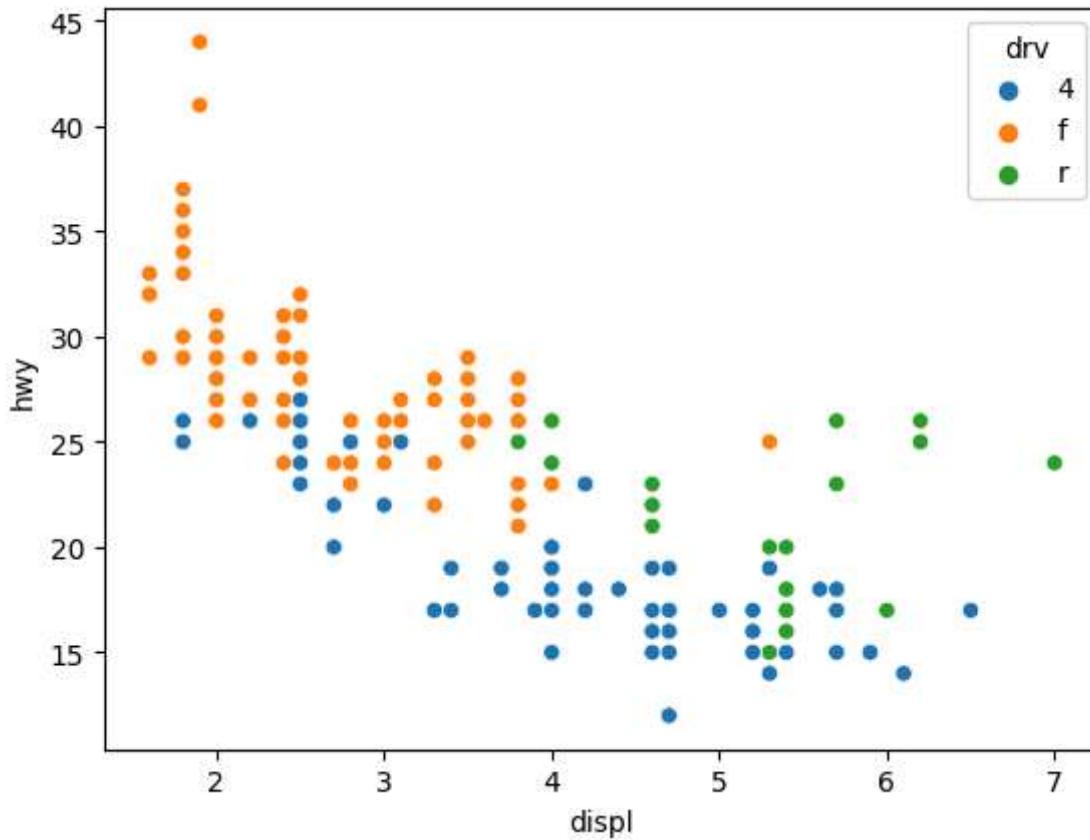
Out[4]: <Axes: xlabel='displ', ylabel='hwy'>



```
In [5]: ## drv별로 표식 색깔 다르게 표현
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy', hue = 'drv', hue_order=['4', 'f', 'r']) #색조(hue)는 'drv'로 구분
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
... if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
... if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
... if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
... if pd.api.types.is_categorical_dtype(vector):
<Axes: xlabel='displ', ylabel='hwy'>
```

Out[5]:



색상 palette 사용

예> sns.scatterplot(data = mpg, x = 'displ', y = 'hwy', hue = 'drv', palette='husl')

> **deep**: 깊은 색조를 가진 팔레트

> **muted**: 어조가 낮은 색조를 가진 팔레트

> **pastel**: 연한 색조를 가진 팔레트

> **bright**: 밝은 색조를 가진 팔레트

> **dark**: 어두운 색조를 가진 팔레트

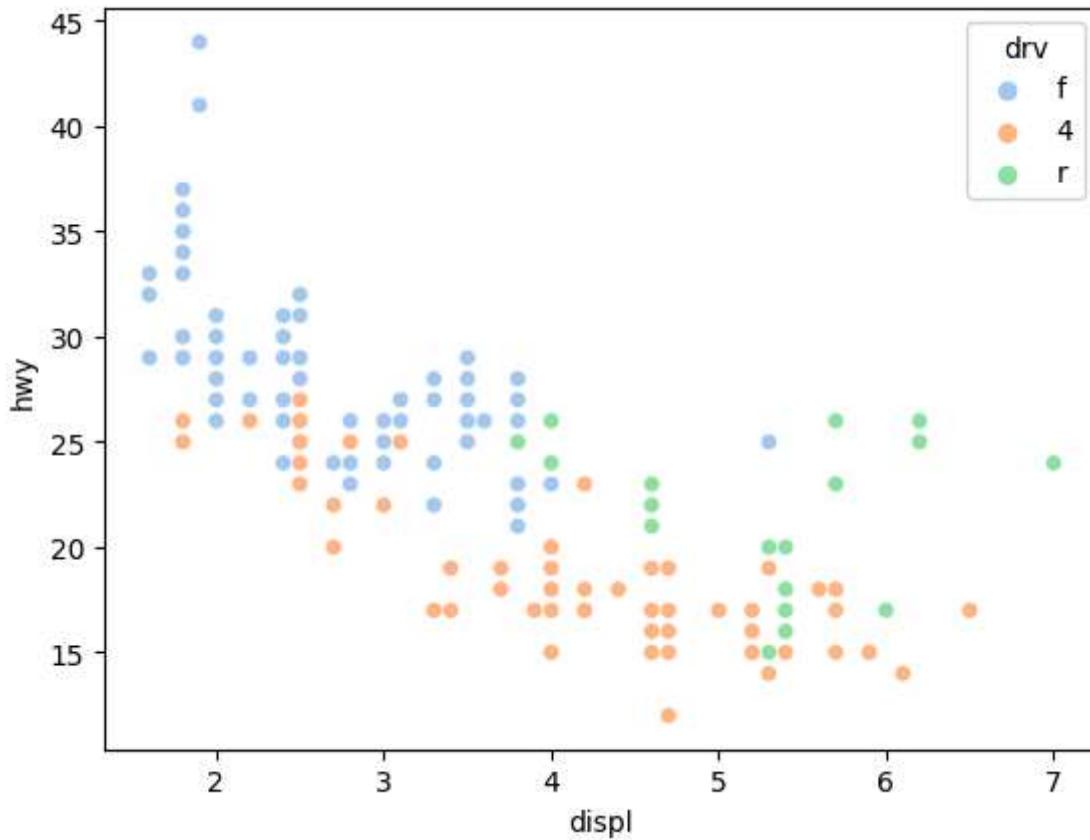
> **colorblind**: 색맹에도 구별이 쉬운 팔레트

> **husl**: HUSL 색상 공간에서 색상을 선택하는 팔레트

> **cube helix**: Cubehelix 색상 맵에서 파생된 팔레트

```
In [17]: ## drv별로 표식 색깔 다르게 표현
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy', hue = 'drv', palette='pastel') #색조(hue)는 'drv'로 구분
```

C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn__oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn__oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn__oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn__oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn__oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn__oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
Out[17]: <Axes: xlabel='displ', ylabel='hwy'>



(알아 두면 좋아요) 그래프 활용하기

그래프 이미지 복사하기

> Shift + (마우스 우측 클릭)

그래프 설정 바꾸기

`matplotlib.pyplot.rcParams.update()` 사용

> `runtime configuration Parameter`를 런타임(실행 중)에 조정하기 위해 사용

```
In [97]: ## 그래프 설정 바꾸기: 개별
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.dpi' : '150'}) # 해상도, 기본값 72
```

```
plt.rcParams.update({'figure.figsize' : [8, 6]})      # 그림 크기, 기본값 [6, 4]
plt.rcParams.update({'font.size' : '15'})            # 글자 크기, 기본값 10
plt.rcParams.update({'font.family' : 'Malgun Gothic'}) # 폰트, 기본값 sans-serif
```

```
In [98]: ## 그래프 설정 바꾸기: 한번에
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.dpi' : '150',
                     'figure.figsize' : [8, 6],           # 해상도, 기본값 72
                     'font.size' : '15',                # 그림 크기, 기본값 [6, 4]
                     'font.family' : 'Malgun Gothic'}) # 글자 크기, 기본값 10
                                         # 폰트, 기본값 sans-serif
```

설정 Default 값으로 되돌리기

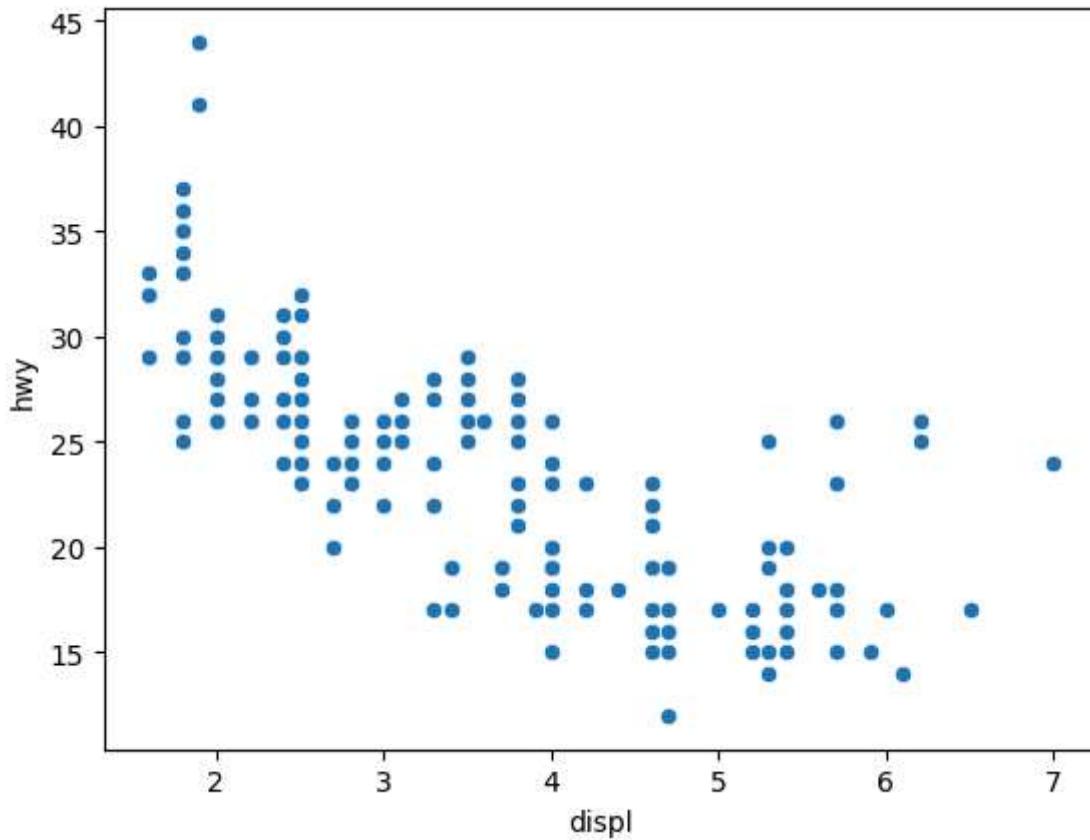
```
In [9]: # 모든 설정 되돌리기
plt.rcParams.update(plt.rcParamsDefault)
```

설명 메시지 숨기기

> 코드 맨 뒤에 ';'를 추가하면 '<Axes: xlabel='displ', ylabel='hwy'>' 메시지를 숨김

```
In [13]: ## 설명 메시지 숨기기
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy'); #맨 뒤에 ';' 추가
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
```



In []:

08-3 막대 그래프 - 집단 간 차이 표현하기

평균 막대 그래프 만들기

1. 집단별 평균표 만들기

> **groupby()** : 집단별로 통계값을 구해줌

>> `groupby(집단구분_변수명).agg(변수명 = (대상_변수명, 적용할_함수명))`

> **agg()** : 대상 변수에 대해 단일 통계값을 구해줌

>> `agg(변수명 = (대상_변수명, 적용할_함수명))`

>> 적용 가능 내장함수 : 'sum', 'mean', 'median', 'min', 'max', 'count', 'std', 'var'

```
In [18]: ## mpg 데이터 불러오기
import pandas as pd

mpg = pd.read_csv('mpg.csv')
mpg
```

```
Out[18]:
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

234 rows × 11 columns

구동방식('drv')별 고속도로주행연비('hwy') 평균 구하기

```
In [19]: ## 구동방식('drv')별 고속도로주행연비('hwy') 평균 구하기
# 구분 열 'drv'가 index 열로 지정됨
df_mpg = mpg.groupby('drv') \
    .agg(mean_hwy = ('hwy', 'mean'))
df_mpg
```

```
Out[19]: mean_hwy
```

drv
4 19.174757
f 28.160377
r 21.000000

```
In [20]: ## 구동방식('drv')별 고속도로주행연비('hwy') 평균 구하기
# 구분 열 'drv'를 index로 미 사용(자동 부여 index 사용)
df_mpg = mpg.groupby('drv', as_index = False) \
            .agg(mean_hwy = ('hwy', 'mean'))
df_mpg
```

```
Out[20]: drv mean_hwy
```

drv	mean_hwy
0 4 19.174757	
1 f 28.160377	
2 r 21.000000	

2. 그래프 만들기

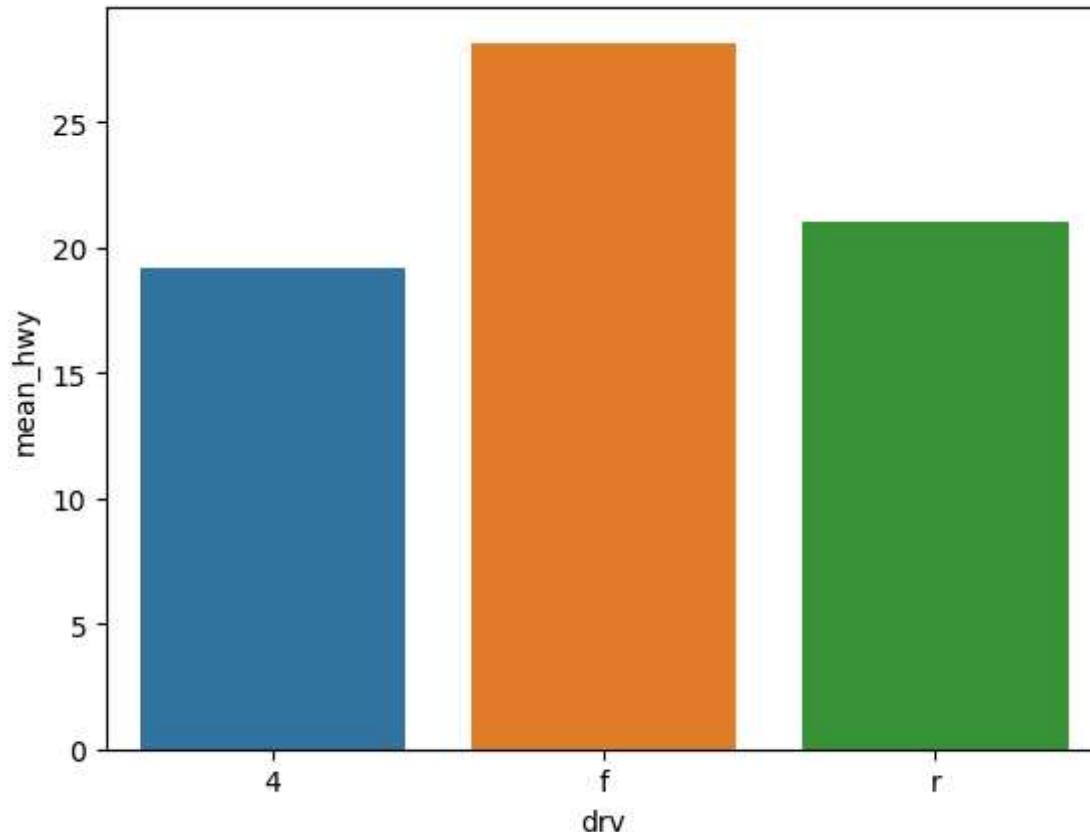
seaborn.barplot()으로 그리기

- > **data** (필수): 데이터 프레임을 지정
- > **x** (필수): x축에 표시될 데이터 열 또는 값을 지정
- > **y** (필수): y축에 표시될 데이터 열 또는 값을 지정
- > **hue**: 데이터를 분할할 기준 열을 지정
- > **order** 및 **ci**: 막대의 순서와 신뢰 구간을 설정
- > **estimator**: 막대 그래프에서 막대의 높이를 계산하는 방법을 지정(기본값은 평균)
- > **palette**: 막대의 색상을 지정할 때 사용할 색상 팔레트를 정의
- > **orient**: 막대 그래프의 방향을 '**v**' (수직) 또는 '**h**' (수평)으로 설정
- > **ax**: 그래프를 그릴 **Matplotlib** 축을 지정

```
In [21]: ## 막대 그래프 만들기  
import seaborn as sns  
  
sns.barplot(data = df_mpg, x = 'drv', y = 'mean_hwy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):
```

```
Out[21]: <Axes: xlabel='drv', ylabel='mean_hwy'>
```

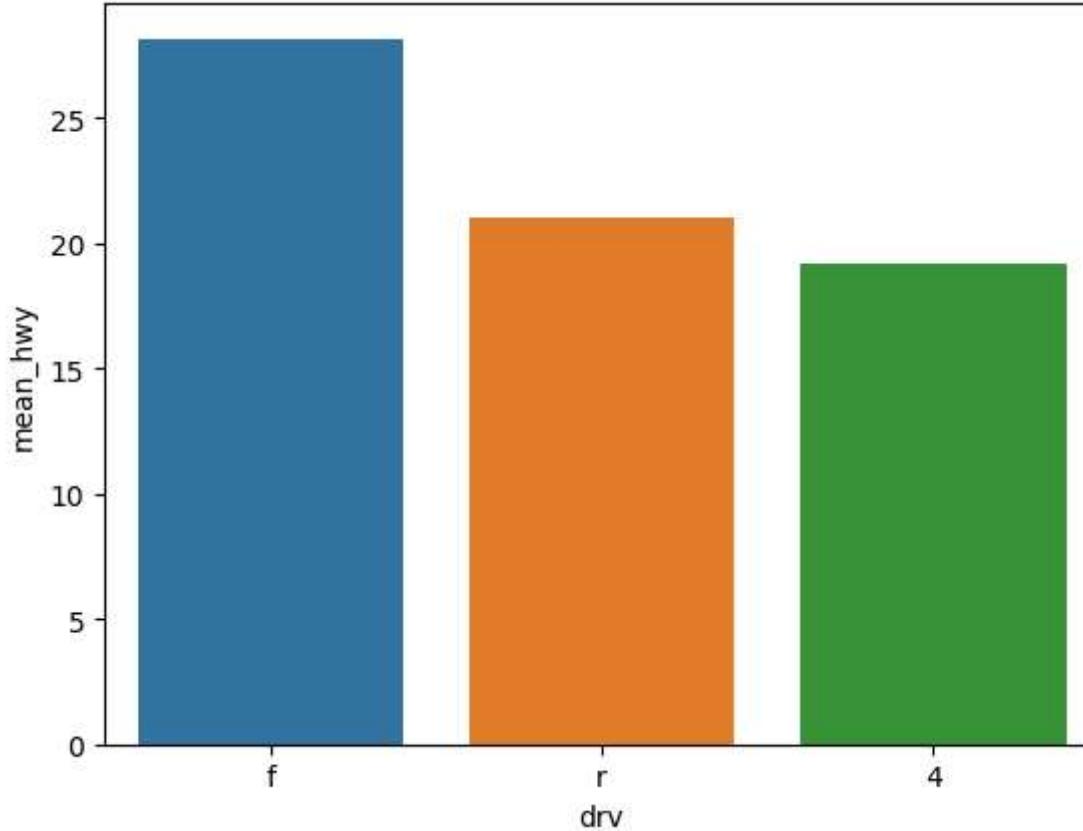


3. 크기순으로 정렬하기

```
In [25]: ## 데이터 프레임 정렬하기  
df_mpg = df_mpg.sort_values('mean_hwy', ascending = False)  
  
## 막대 그래프 만들기  
sns.barplot(data = df_mpg, x = 'drv', y = 'mean_hwy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
... if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
... if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
... if pd.api.types.is_categorical_dtype(vector):
```

```
Out[25]: <Axes: xlabel='drv', ylabel='mean_hwy'>
```



matplotlib 라이브러리로 그리기

```
In [23]: ## matplotlib 라이브러리로 그리기
import matplotlib.pyplot as plt

# 데이터 생성
categories = ['A', 'B', 'C', 'D']
values = [25, 30, 20, 35]

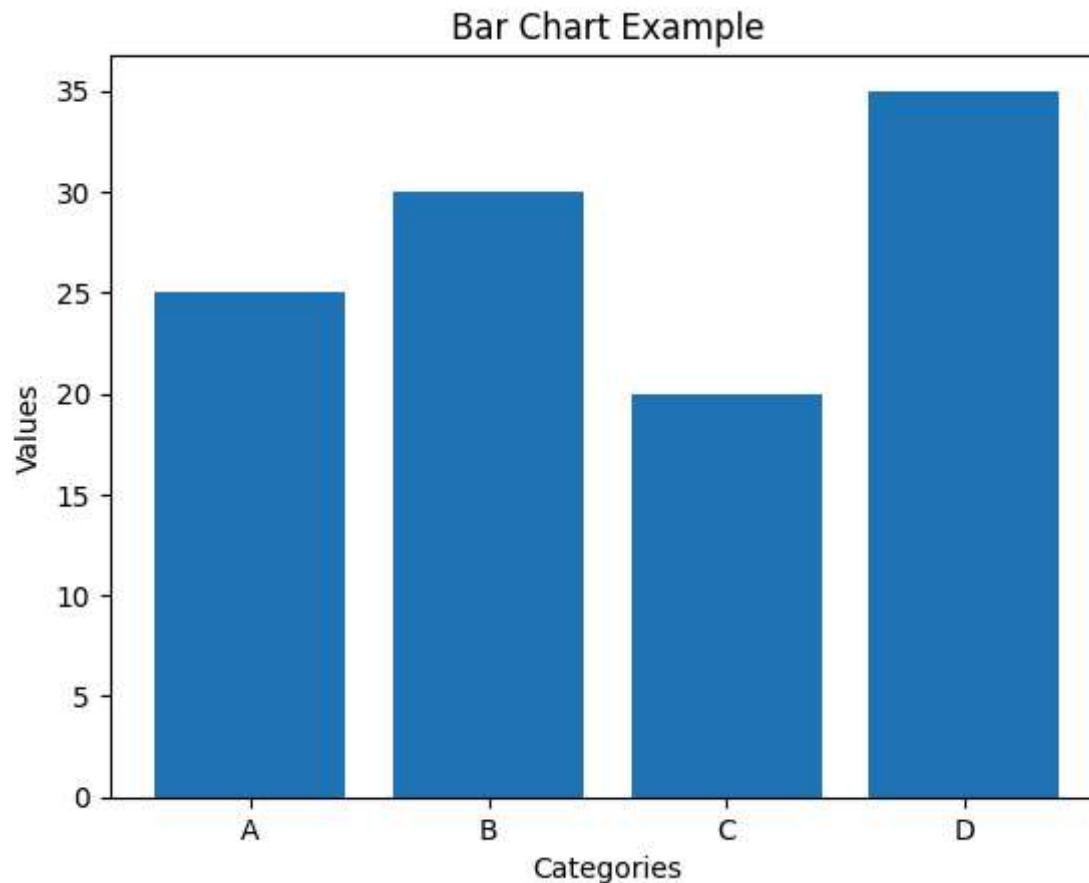
# plt.bar()를 사용하여 막대 그래프 그리기
plt.bar(categories, values)

# 그래프 제목 설정
```

```
plt.title('Bar Chart Example')

# x축과 y축 레이블 설정
plt.xlabel('Categories')
plt.ylabel('Values')

# 그래프 표시
plt.show()
```



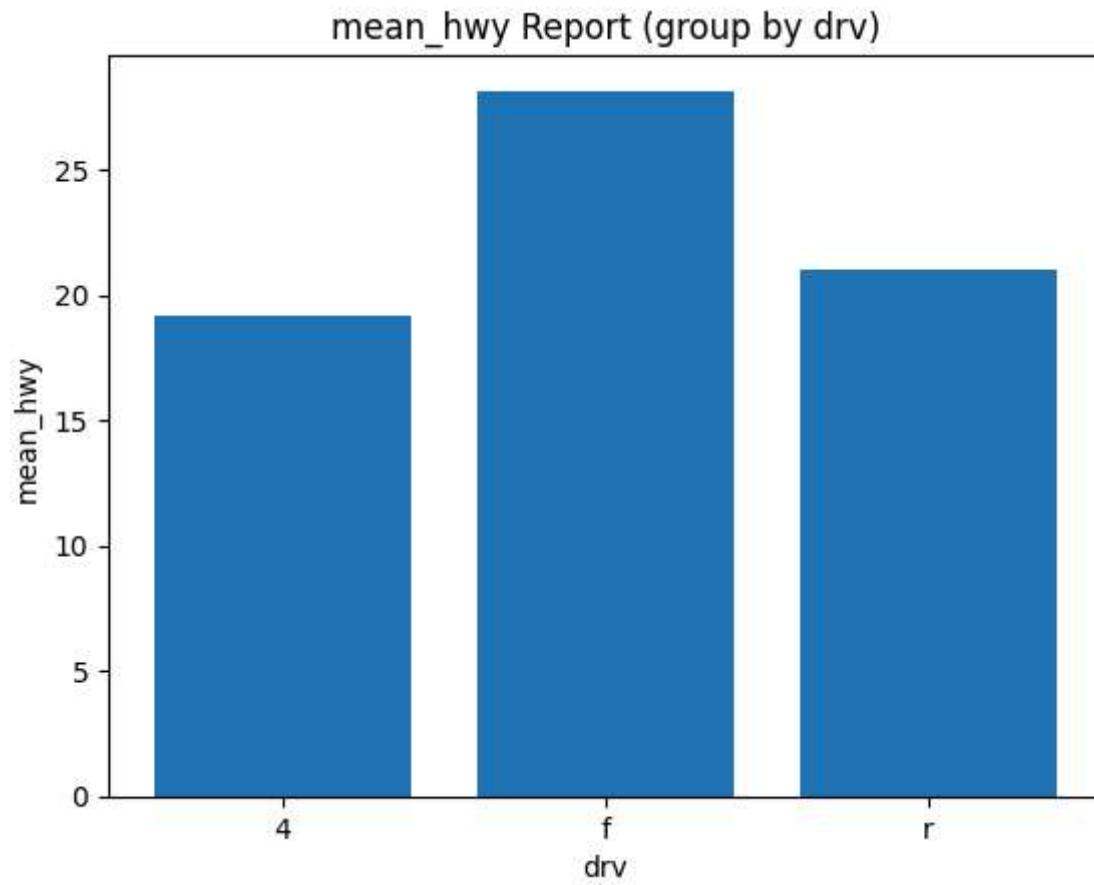
```
In [24]: ## matplotlib 라이브러리로 그리기
import matplotlib.pyplot as plt

plt.bar(df_mpg['drv'], df_mpg['mean_hwy']) #그래프 그리기

plt.title("mean_hwy Report (group by drv)") #주석 달기
```

```
plt.xlabel('drv')
plt.ylabel('mean_hwy')

plt.show() #그래프 표시
```



seaborn.barplot()으로 빈도 막대 그래프 만들기

```
>> agg(변수명 = (대상_변수명, 적용할_함수명))

>> 적용 가능 내장함수 : 'sum', 'mean', 'median', 'min', 'max', 'count', 'std', 'var'
```

1. 집단별 빈도표 만들기

```
In [27]: ## 집단별 빈도표 만들기  
df_mpg = mpg.groupby('drv', as_index = False) ¶  
        .agg(n = ('drv', 'count'))  
  
df_mpg
```

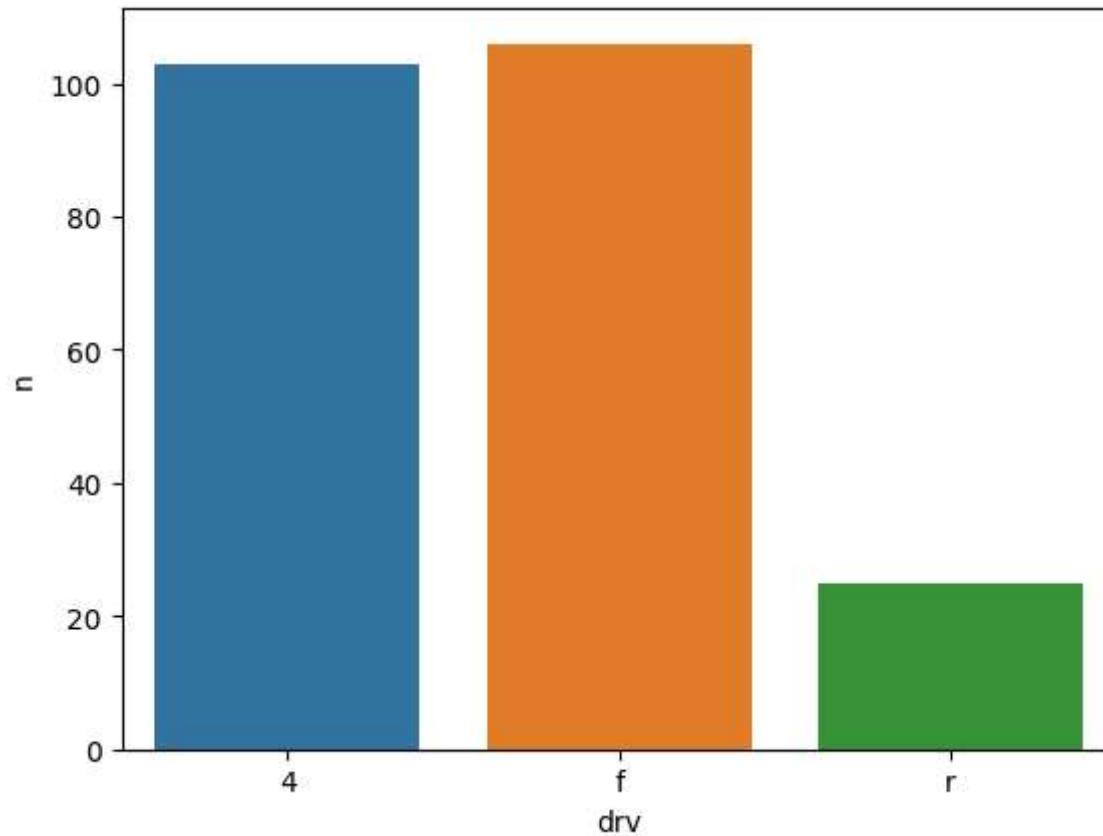
```
Out[27]:   drv    n  
0     4  103  
1     f  106  
2     r   25
```

2. 그래프 만들기

```
In [30]: ## 막대 그래프 만들기  
import seaborn as sns  
  
sns.barplot(data = df_mpg, x = 'drv', y = 'n')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
    if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
    if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
    if pd.api.types.is_categorical_dtype(vector):
```

```
Out[30]: <Axes: xlabel='drv', ylabel='n'>
```



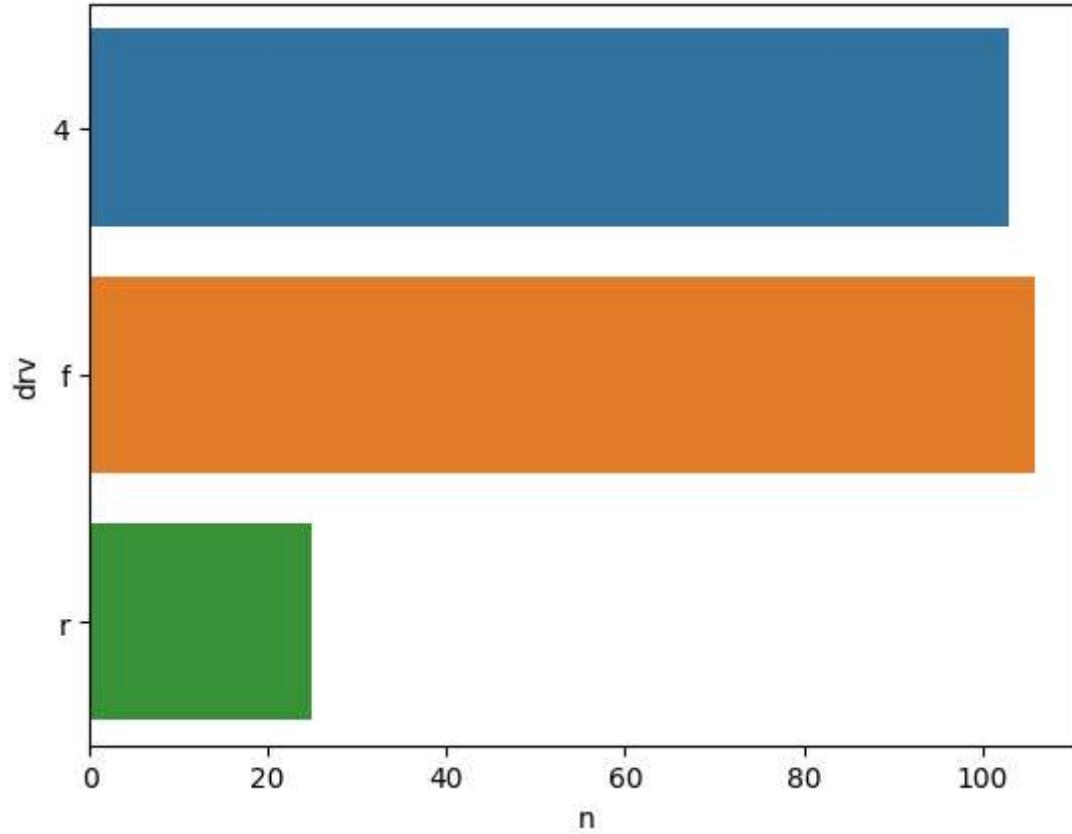
x, y 축 바꾸어 그리기

```
In [31]: ## 막대 그래프 만들기
import seaborn as sns

sns.barplot(data = df_mpg, y = 'drv', x = 'n')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
Out[31]: <Axes: xlabel='n', ylabel='drv'>
```



seaborn.countplot()으로 빈도 막대 그래프 그리기

countplot(): 데이터프레임을 기반으로 메서드가 알아서 빈도수를 구해서 작성

- > **data** (필수): 데이터 프레임을 지정
- > **x, y** (필수 중 하나): 데이터의 범주형 변수를 지정
- > **hue**: 데이터를 분할할 기준 열을 지정
- > **order**: 범주형 변수의 순서를 명시적으로 지정
- > **palette**: 그래프에 사용할 색상 팔레트를 정의
- > **orient**: 그래프의 방향을 'v' (수직) 또는 'h' (수평)로 설정

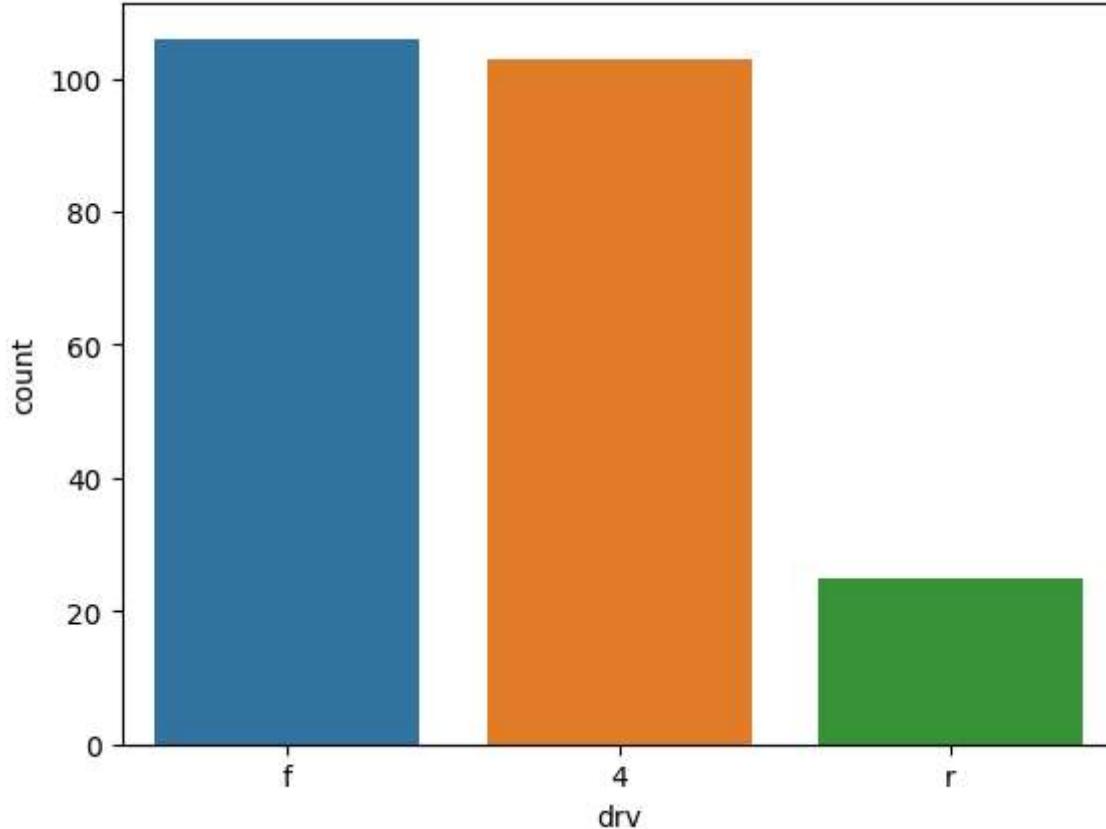
- > **ax**: 그래프를 그릴 **Matplotlib** 축을 지정
- > **dodge**: 여러 범주를 하나의 막대 안에 그리거나 구분할 때 사용
- > **saturation**: 색상 포화도를 조절하여 색상의 밝기를 조절
- > **color**: 모든 막대의 색상을 일괄적으로 지정
- > **tick_label**: x축 또는 y축에 범주 레이블을 지정
- > **ax**: 그래프를 그릴 **Matplotlib** 축을 지정
- > **linewidth**: 막대의 테두리 선의 두께를 설정
- > **edgecolor**: 막대의 테두리 선 색상을 설정
- > **label**: 범례 항목의 레이블을 지정

```
In [32]: ## 빈도 막대 그래프 만들기  
import seaborn as sns
```

```
sns.countplot(data = mpg, x = 'drv')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
... if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
... if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
... if pd.api.types.is_categorical_dtype(vector):
```

```
Out[32]: <Axes: xlabel='drv', ylabel='count'>
```



x축 순서 조정

```
In [39]: # 4, f, r 순으로 막대 정렬: 순서 강제 지정  
sns.countplot(data = mpg, x = 'drv', order = ['4', 'f', 'r'])
```

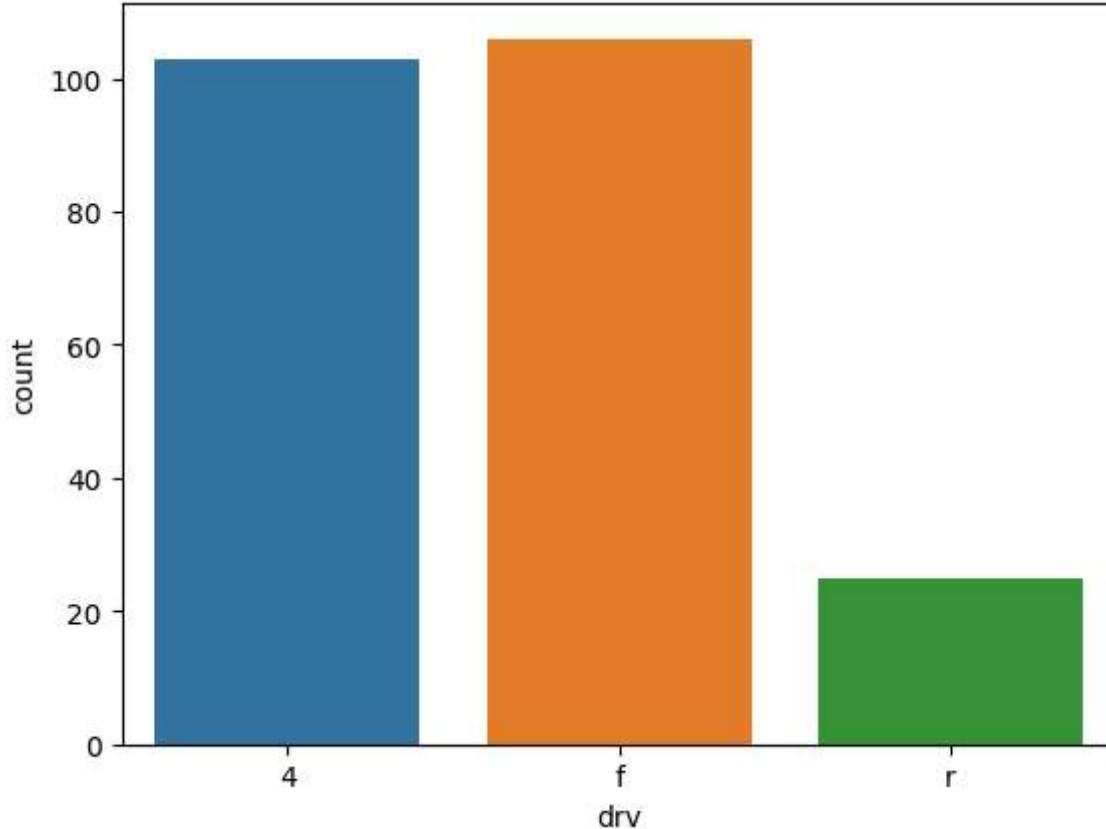
```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
... if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
... if pd.api.types.is_categorical_dtype(vector):
```

```
Out[39]: <Axes: xlabel='drv', ylabel='count'>
```



열 변수명을 추출하여 사용

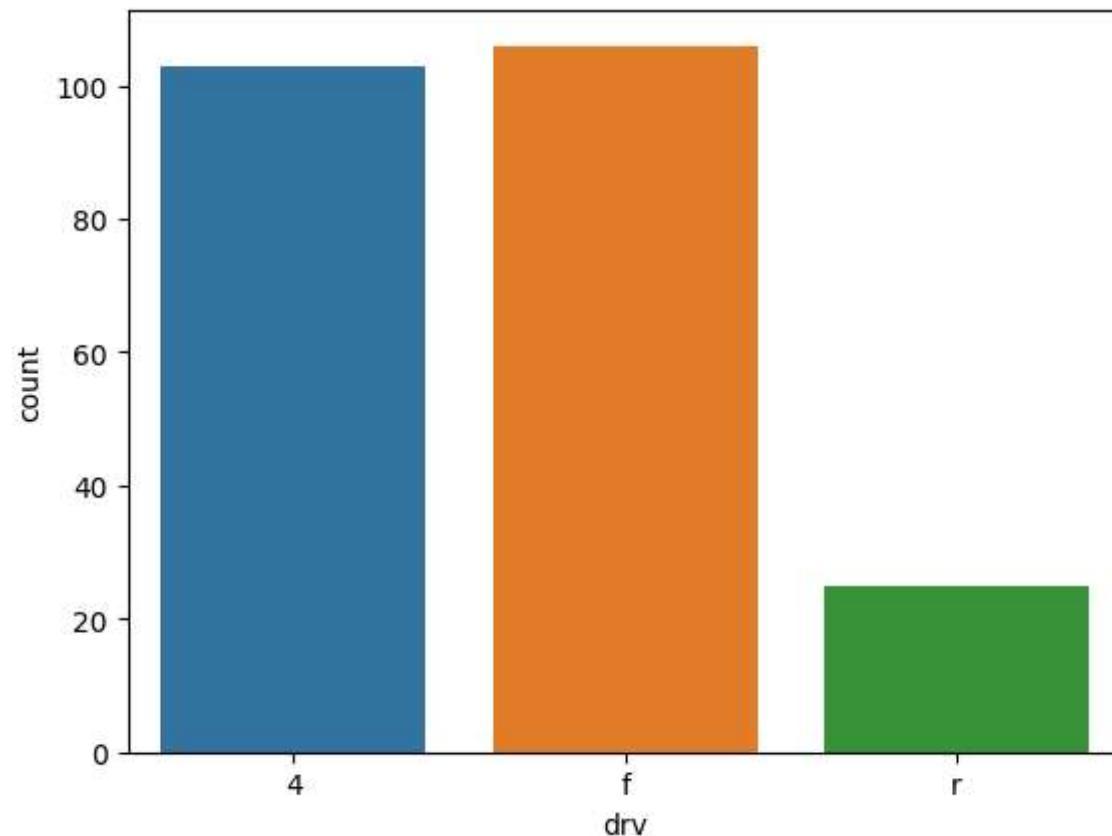
```
In [34]: ## 열 값 도메인 추출: unique()
drv_order = df_mpg['drv'].unique()
drv_order
```

```
Out[34]: array(['4', 'f', 'r'], dtype=object)
```

```
In [35]: ## 4, f, r 순으로 막대 정렬: 순서 정렬 지정
drv_order = df_mpg['drv'].unique() #열 값 도메인 추출
sns.countplot(data = mpg, x = 'drv', order = drv_order)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[35]: <Axes: xlabel='drv', ylabel='count'>



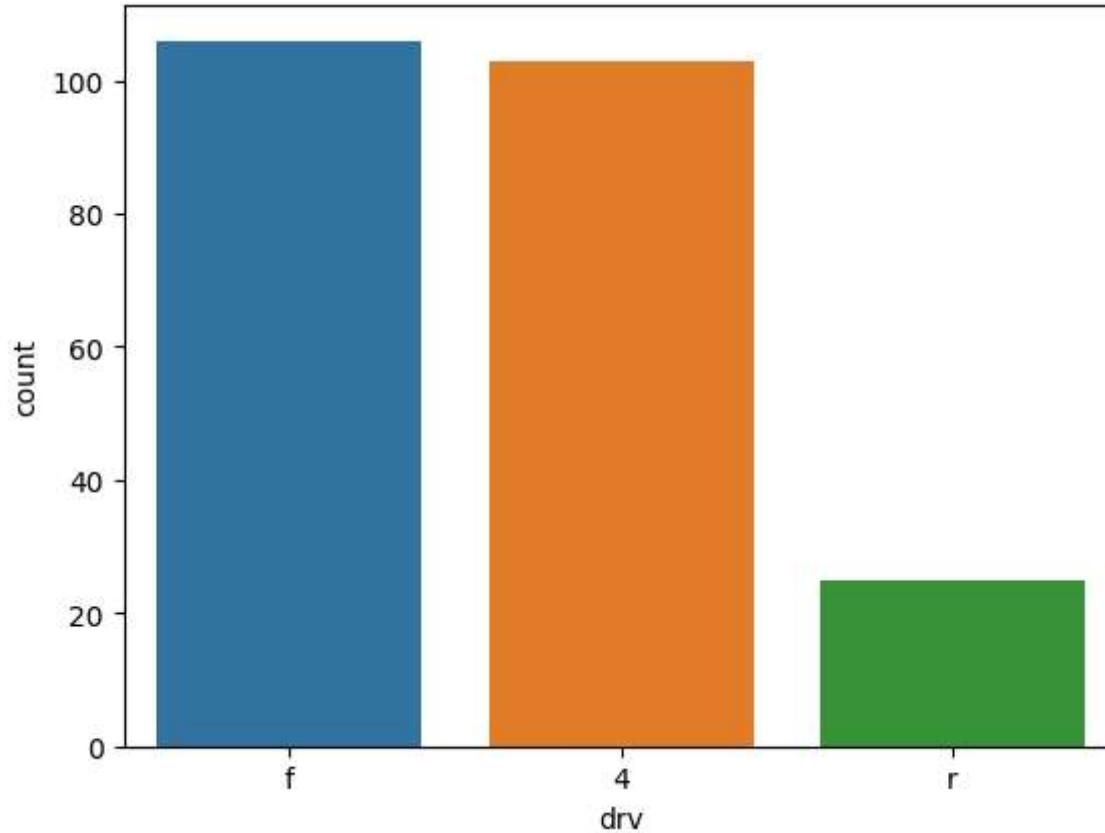
index를 추출하여 사용

```
In [36]: ## drv의 값을 빈도가 높은 순으로 열기
drv_order = mpg['drv'].value_counts().index #count가 큰 순으로 데이터가 구성됨
print(mpg['drv'].value_counts())
drv_order
```

```
drv
f... 106
4... 103
r... 25
Name: count, dtype: int64
Index(['f', '4', 'r'], dtype='object', name='drv')
Out[36]:
```

```
In [37]: ## drv 빈도 높은 순으로 막대 정렬
drv_order = mpg['drv'].value_counts().index
sns.countplot(data = mpg, x = 'drv', order = drv_order)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
... if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
... if pd.api.types.is_categorical_dtype(vector):
<Axes: xlabel='drv', ylabel='count'>
Out[37]:
```



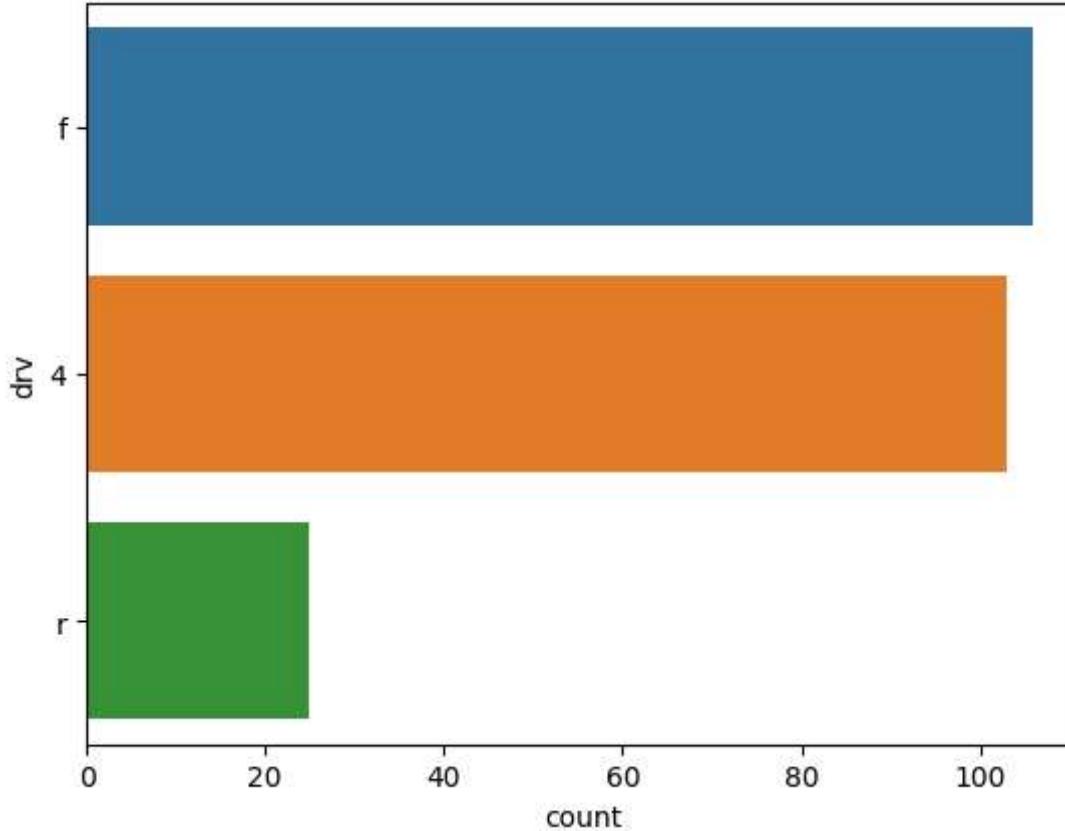
x, y 축 바꾸어 그리기

```
In [38]: ## 막대 그래프 만들기
import seaborn as sns

sns.countplot(data = mpg, y = 'drv', order = drv_order)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
Out[38]: <Axes: xlabel='count', ylabel='drv'>
```

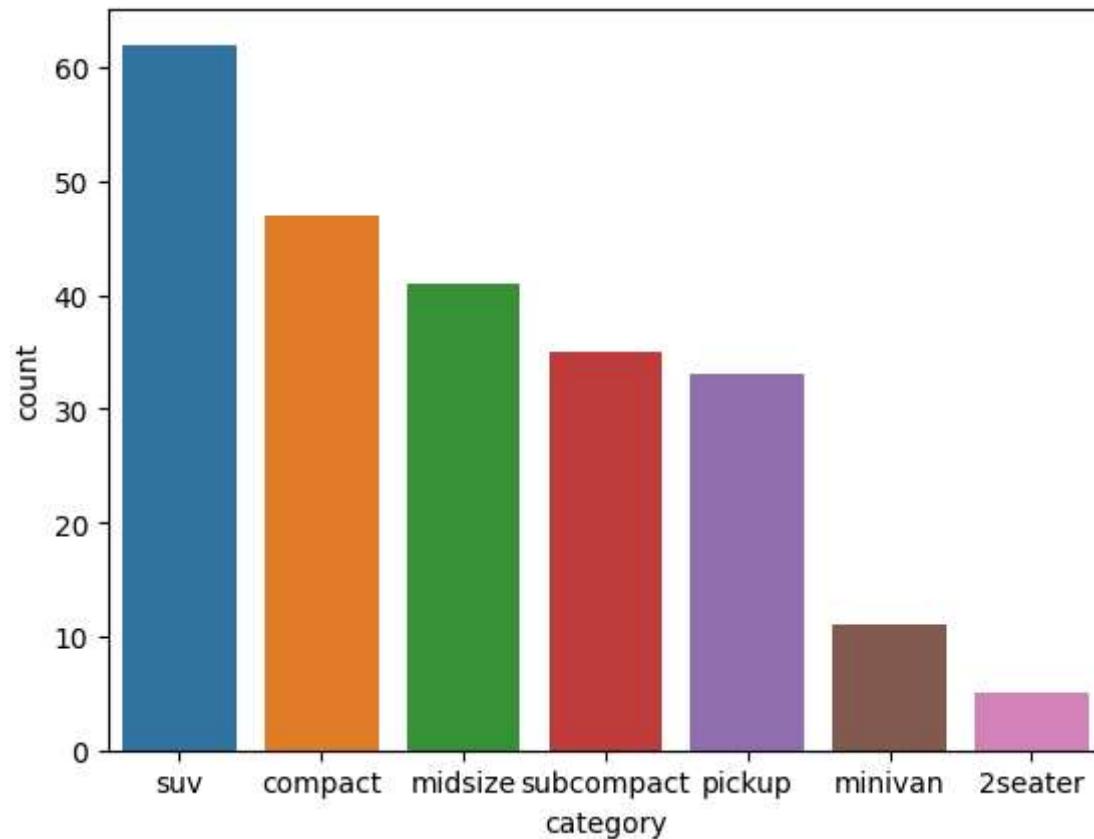


```
In [118]: ## 그래프 설정 바꾸기: 한번에
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.dpi' : '150',          # 해상도, 기본값 72
                     'figure.figsize' : [6, 4],      # 그림 크기, 기본값 [6, 4]
                     'font.size' : '10',           # 글자 크기, 기본값 10
                     'font.family' : 'Malgun Gothic'}) # 폰트, 기본값 sans-serif
```

```
In [40]: ## 'category'별 빈도수 그래프 막대 그리기
sns.countplot(data = mpg, x = 'category', order = mpg['category'].value_counts().index)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
```

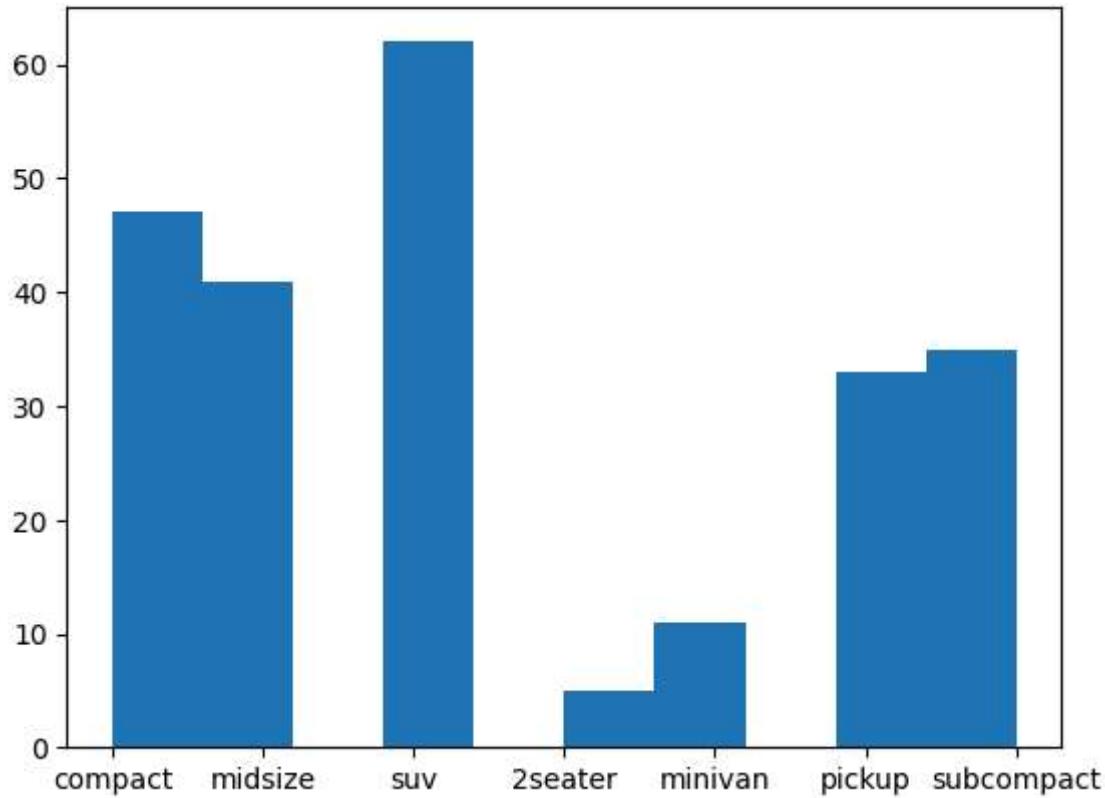
Out[40]: <Axes: xlabel='category', ylabel='count'>



matplotlib 라이브러리로 그리기

```
In [41]: ## matplotlib 라이브러리로 그래프 그리기
import matplotlib.pyplot as plt

plt.hist(mpg['category'])
plt.show()
```



~별, ~별 빈도수 막대 그래프 그리기

> hue 매개변수 사용

groupby() 후 sns.barplot() 사용

```
In [49]: ## 제조회사(manufacturer)별 구동방식(drv)별 cty 평균 구하기
df_mpg = mpg.groupby(['manufacturer', 'category'], as_index = False) \
    .agg(n = ('category', 'count')) \
df_mpg.head()
```

```
Out[49]:   manufacturer category n
```

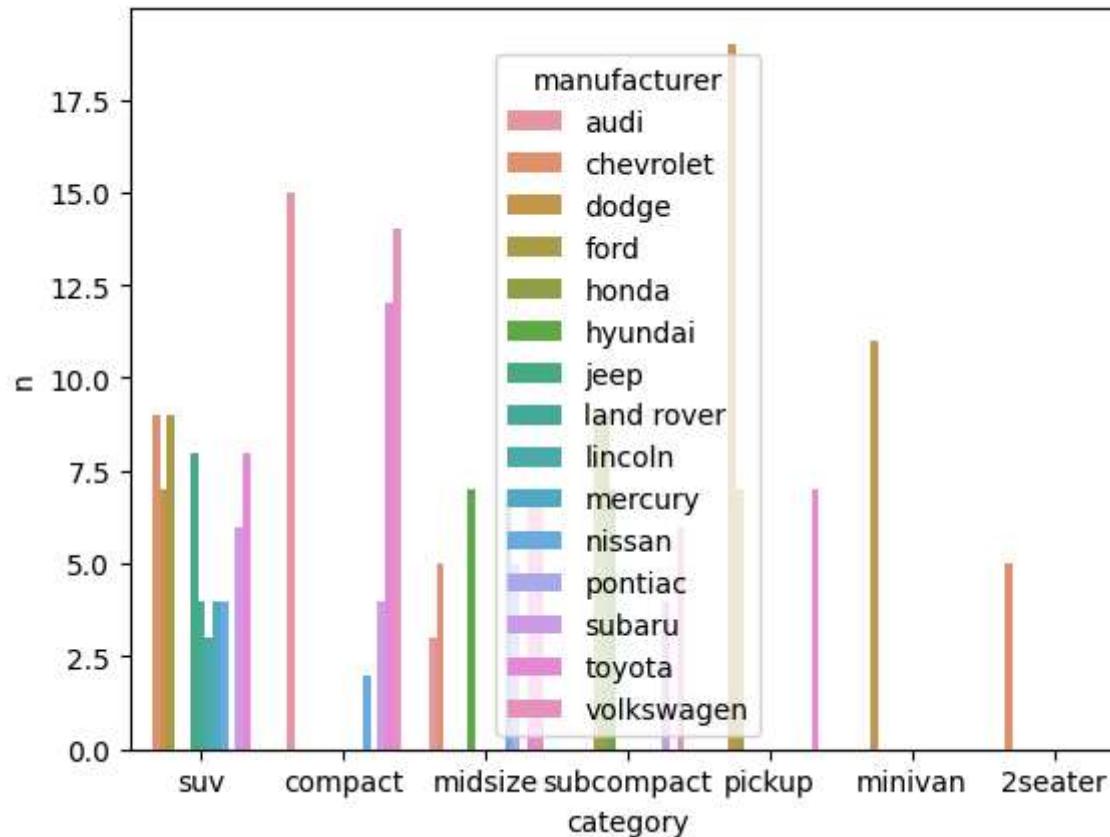
	manufacturer	category	n
0	audi	compact	15
1	audi	midsize	3
2	chevrolet	2seater	5
3	chevrolet	midsize	5
4	chevrolet	suv	9

```
In [53]: ## 막대 그래프 만들기  
import seaborn as sns
```

```
sns.barplot(data = df_mpg, x = 'category', y = 'n', hue = 'manufacturer', order = mpg['category'].value_counts().index)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be  
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):
```

```
Out[53]: <Axes: xlabel='category', ylabel='n'>
```



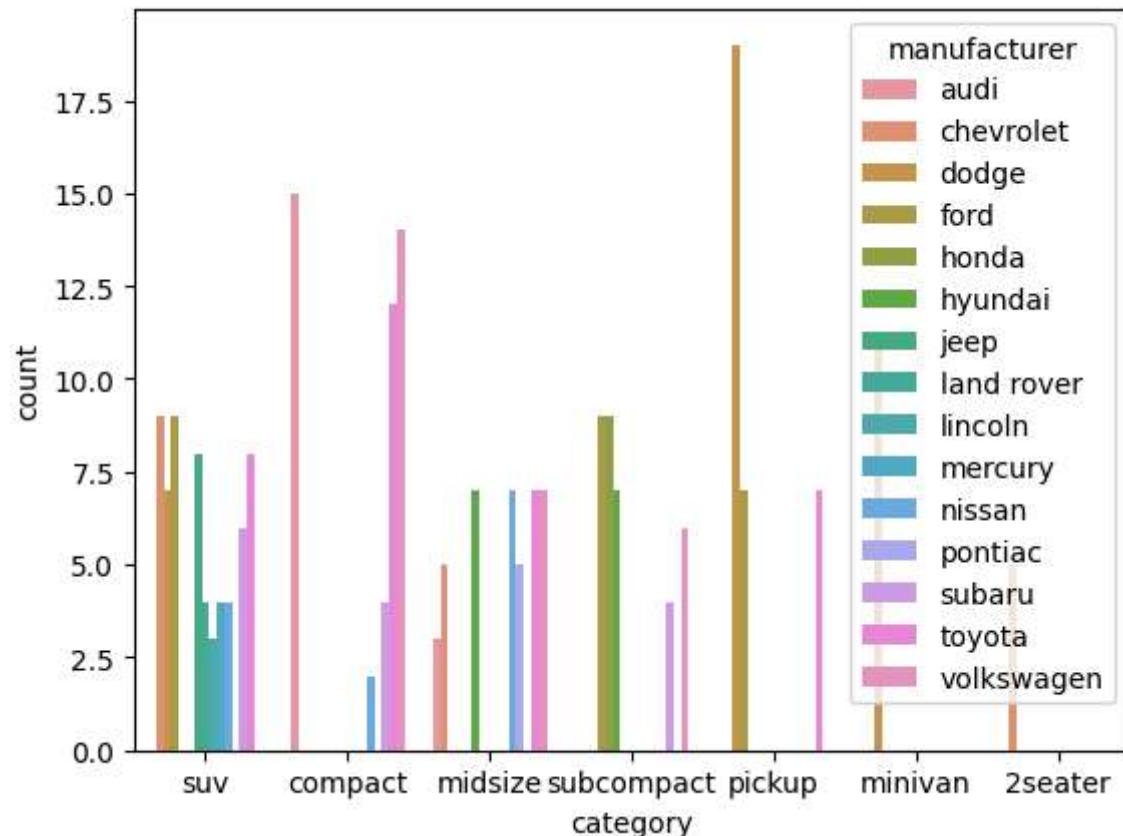
`sns.countplot()` 사용

```
In [42]: ## 'category'별 'manufacturer'별 빈도수 그래프 막대 그리기
import seaborn as sns

sns.countplot(data = mpg, x = 'category', order = mpg['category'].value_counts().index, hue = 'manufacturer')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[42]: <Axes: xlabel='category', ylabel='count'>



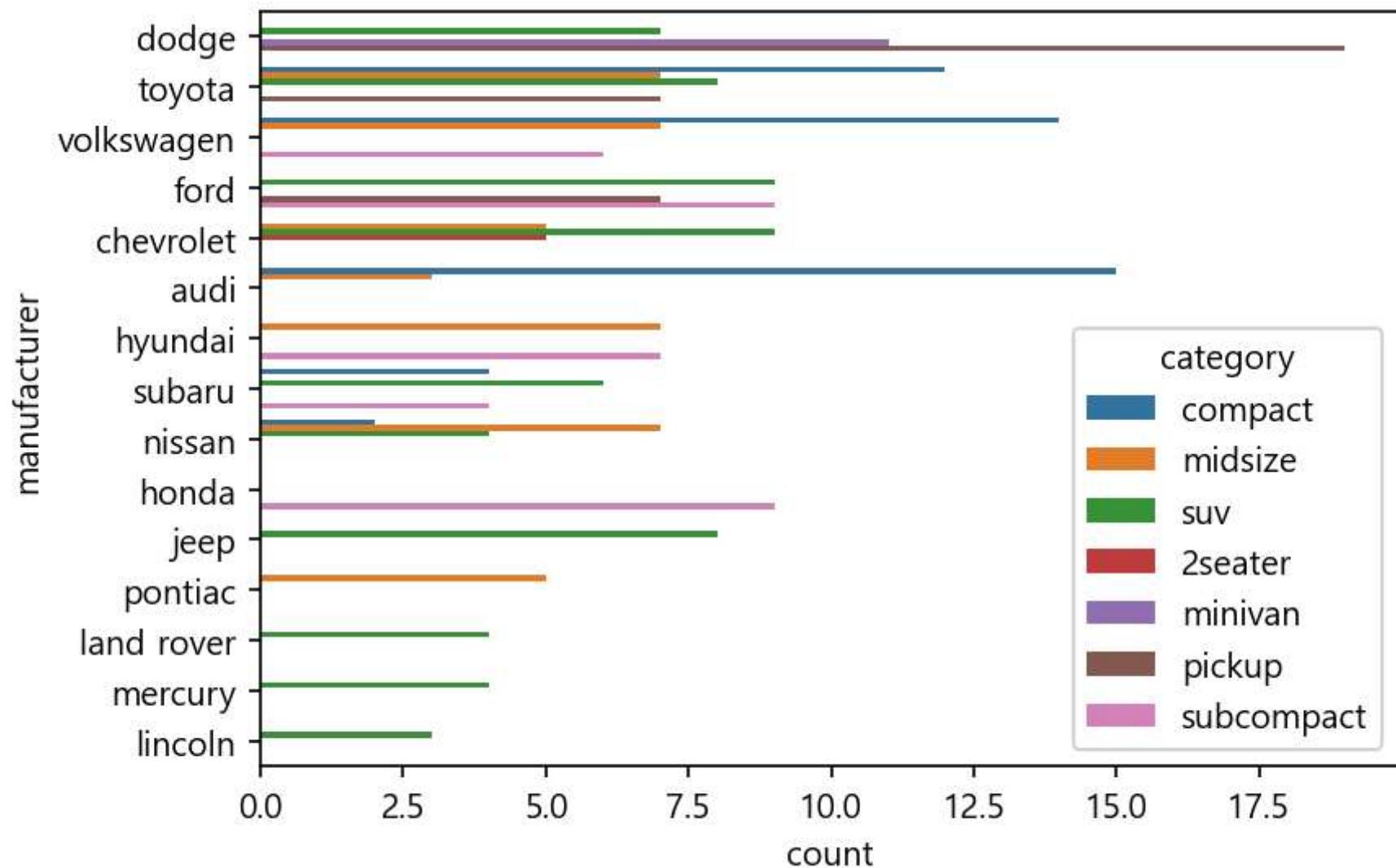
In [122...]

```
## 'manufacturer'별 'category'별 빈도수 그래프 막대 그리기
import seaborn as sns

sns.countplot(data = mpg, y = 'manufacturer', order = mpg['manufacturer'].value_counts().index, hue = 'category')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
```

Out[122]: <Axes: xlabel='count', ylabel='manufacturer'>



In []:

08-3 파이 그래프 - 집단 간 점유율 표현하기

```
In [54]: ## mpg 데이터 불러오기  
import pandas as pd  
  
mpg = pd.read_csv('mpg.csv')  
mpg
```

```
Out[54]:
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

234 rows × 11 columns

구동방식('drv')별 빈도수('count_drv') 구하기

```
In [56]: ## 구동방식('drv')별 고속도로주행연비('hwy') 평균 구하기  
# 구분 열 'drv'가 index 열로 지정됨  
df_mpg = mpg.groupby('drv', as_index = False) #  
        .agg(count_drv = ('drv', 'count'))  
df_mpg
```

Out[56]:

	drv	count_drv
0	4	103
1	f	106
2	r	25

빈도수 파이 그래프 그리기

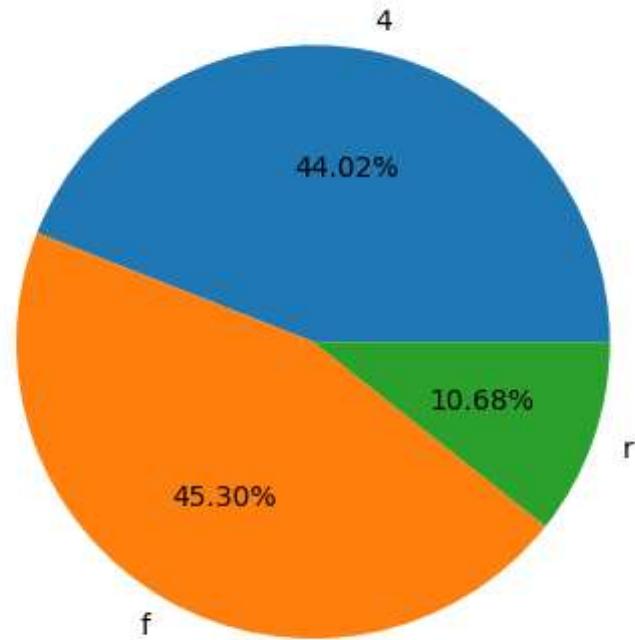
matplotlib.pie() 사용

- > **x** (필수): 파이 차트에 사용될 데이터를 지정
- > **labels**: 파이 차트에서 각 섹션의 레이블로 사용될 문자열 목록을 지정
- > **colors**: 파이 차트의 섹션 색상을 지정(색상 이름 또는 16진수 RGB 코드)
- > **autopct**: 각 섹션의 비율을 표시할 형식을 지정("%1.1f%%")
- > **startangle**: 파이 차트의 시작 각도를 지정(기본값은 0도(East)이며, 반시계 방향으로 증가)
- > **explode**: 특정 섹션을 파이 차트에서 분리하여 강조 표시(분리 안할 섹션은 0으로 설정하고, 분리할 섹션은 0 이상 반지름 비율 값으로 지정)
- > **shadow**: True로 설정하면 파이 차트에 그림자가 표시
- > **pctdistance**: **autopct**로 지정한 비율 텍스트의 위치를 조절(0에서 1 사이의 값을 사용, 텍스트가 중심에서 얼마나 떨어져 있는지를 설정)
- > **labeldistance**: **labels**로 지정한 레이블의 위치를 조절(0에서 1 사이의 값을 사용하여 레이블이 중심에서 얼마나 떨어져 있는지를 설정)
- > **radius**: 파이 차트의 크기를 조절(기본값은 1이며, 1보다 작게 설정하면 작은 원 그래프)
- > **counterclock**: False로 설정하면 시계 방향으로 그래프
- > **wedgeprops**: 섹션 웨지의 속성을 설정
- > **textprops**: 비율 텍스트의 속성을 설정
- > **frame**: True로 설정하면 파이 차트에 외곽 프레임이 표시
- > **rotatelabels**: 레이블을 회전시킬지 여부를 설정

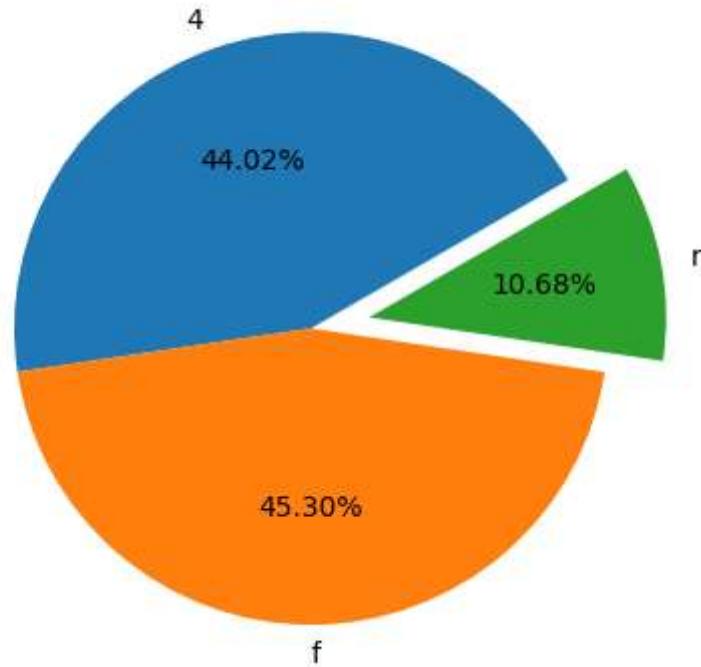
In [57]:

```
#### 파이그래프 그리기
##### > 'drv'별 'count_drv'에 대한 비율 파이그래프 그리기
import matplotlib.pyplot as plt

plt.pie(df_mpg['count_drv'], labels = df_mpg['drv'], autopct = '%.2f%%')
plt.show()
```



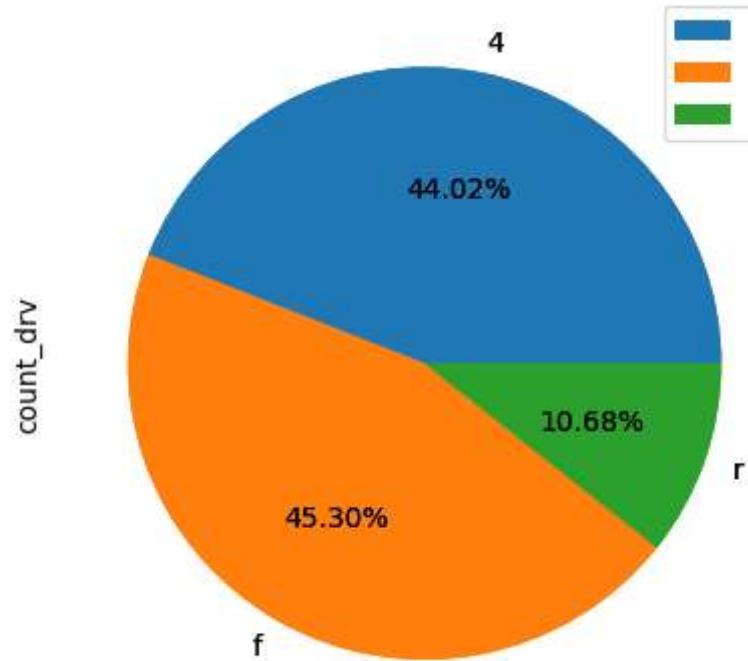
```
In [73]: ## [matplotlib로 그래프 표현] startangle 적용
expl_list = [0, 0, 0.2] #중심으로 이격 표현
plt.pie(df_mpg['count_drv'], labels = df_mpg['drv'], autopct = '%0.2f%%', explode = expl_list, startangle=30) #시작점: East 기준,
plt.show()
```



또 다른 방식 : `pandas.plot()` 활용

```
In [77]: ## 또 다른 방식 : pandas.plot() 활용
import matplotlib.pyplot as plt

df_mpg.plot(kind='pie', y='count_drv', labels = df_mpg['drv'], autopct = '%0.2f%%')
df_mpg['count_drv'].plot(kind='pie', labels = df_mpg['drv'], autopct = '%0.2f%%')
plt.show()
```



In []:

08-4 선 그래프 - 시간에 따라 달라지는 데이터 표현하기

선 그래프 그리기

In [78]:
mpg 데이터 불러오기
import pandas as pd

```
mpg = pd.read_csv('mpg.csv')  
mpg
```

Out[78]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

234 rows × 11 columns

In [79]:

```
## 'category'별 빈도수 구하기
df_mpg = mpg.groupby('category', as_index = False) \
    .agg(count_cat = ('category', 'count'))
df_mpg
```

Out[79]:

	category	count_cat
0	2seater	5
1	compact	47
2	midsize	41
3	minivan	11
4	pickup	33
5	subcompact	35
6	suv	62

seaborn.lineplot()으로 그리기

- > **data** (필수): 데이터 프레임을 지정
- > **x** (필수): x축에 사용할 데이터 열 또는 값의 이름 또는 위치를 지정
- > **y** (필수): y축에 사용할 데이터 열 또는 값의 이름 또는 위치를 지정
- > **hue**: 데이터를 분할할 기준 열을 지정
- > **style**: 선의 스타일을 지정할 열의 이름을 지정
- > **markers**: 데이터 포인트의 마커 스타일을 지정
- > **palette**: 색상 팔레트를 지정
- > **hue_order** 및 **hue_norm**: hue에 지정된 열의 순서 및 정규화를 설정
- > **estimator**: 데이터 포인트를 그룹화할 때 사용할 통계적 함수를 지정
- > **ci**: 신뢰 구간을 설정
- > **n_boot**: 부트스트랩 반복 횟수를 설정
- > **sort**: x축 값을 정렬할지 여부를 설정
- > **err_style**: 신뢰 구간을 표시하는 스타일을 설정
- > **err_kws**: err_style에 따른 신뢰 구간의 추가 매개변수를 지정
- > **legend**: 범례 표시 여부를 설정
- > **ax**: 그래프를 그릴 **Matplotlib** 축을 지정
- > **drawstyle**: 선 그래프의 스타일을 설정('default', 'steps', 'steps-pre', 'steps-mid', 'steps-post' 중에서 선택)

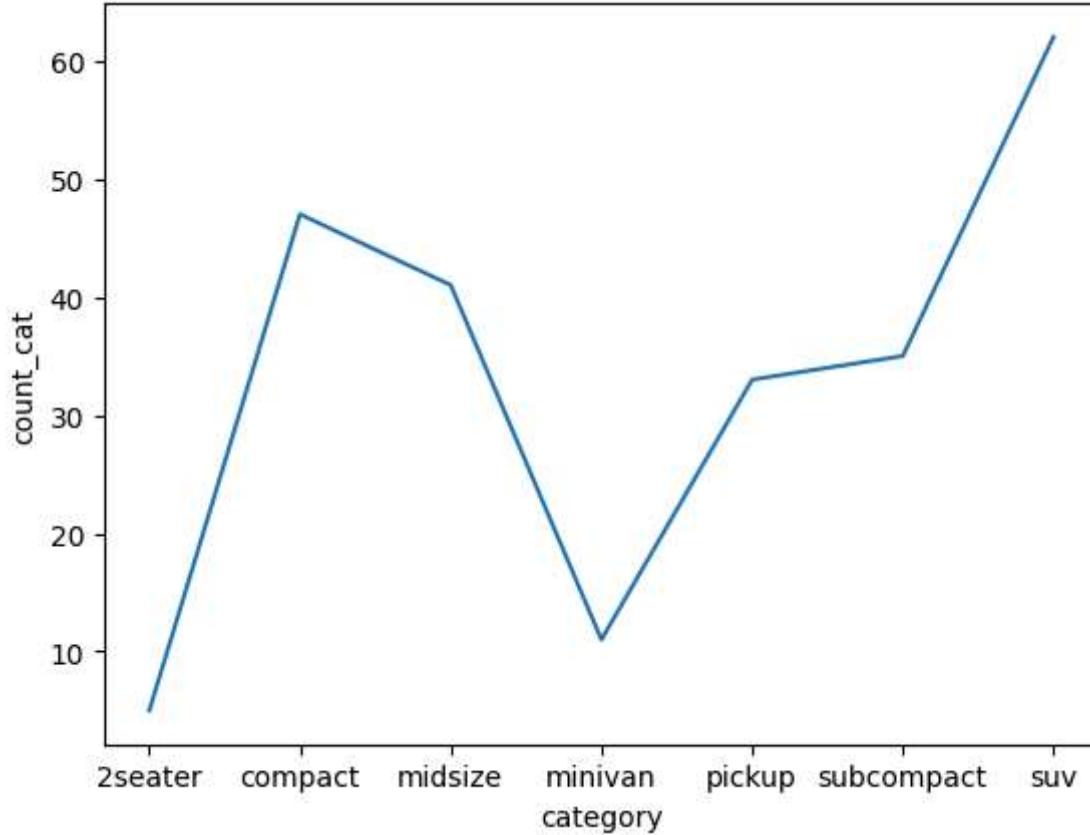
- > **dashes**: 선의 대시 스타일을 설정
- > **markerize**: 데이터 포인트 마커의 크기를 설정
- > **linewidth**: 선 그래프의 두께를 설정
- > **units**: 데이터 그룹화를 위한 열을 지정
- > **units_norm**: 데이터 그룹화 시에 정규화할 열을 지정
- > **alpha**: 선 그래프의 투명도를 설정
- > **axlabel**: x축과 y축의 레이블을 지정
- > **y_units**: y축 데이터의 물리적 단위를 지정

```
In [80]: ## 'category'의 빈도수에 대한 선 그래프 구하기
import seaborn as sns
```

```
sns.lineplot(data = df_mpg, x = 'category', y = 'count_cat')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
```

```
Out[80]: <Axes: xlabel='category', ylabel='count_cat'>
```

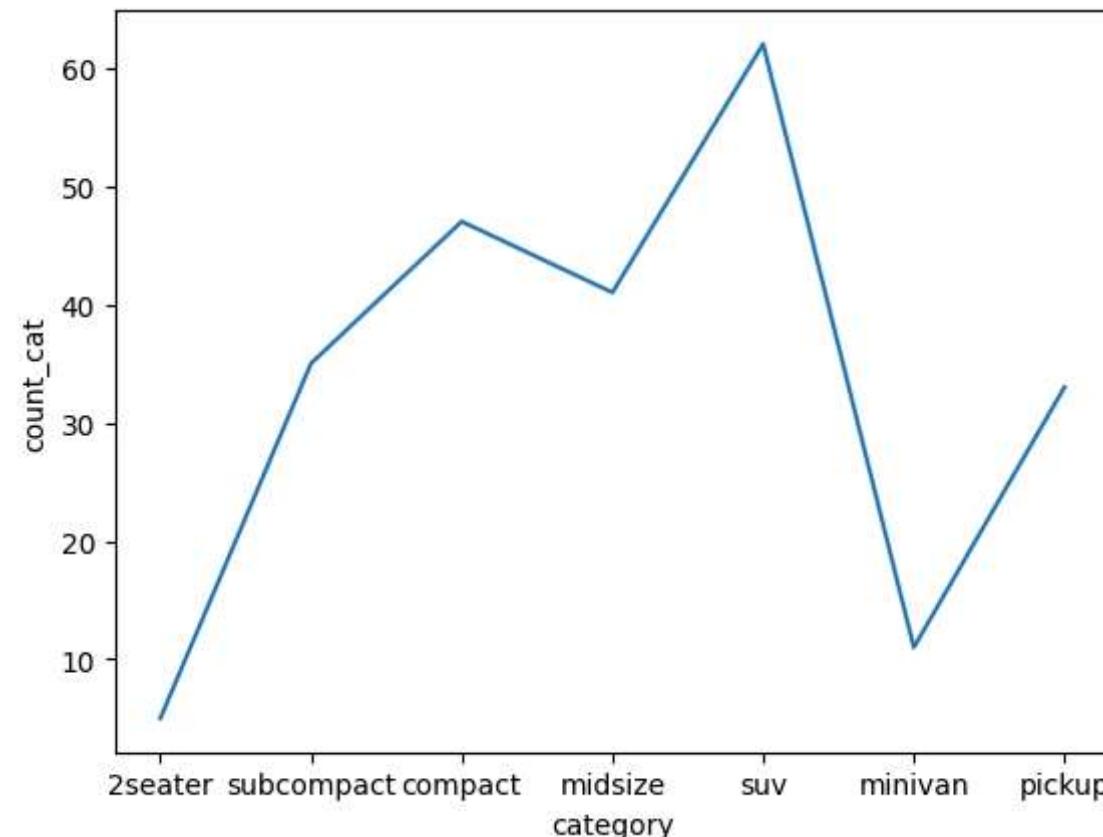


```
In [82]: ## 'category'의 빈도수에 대한 선 그래프 구하기
import seaborn as sns

cat_order = [0, 5, 1, 2, 6, 3, 4]
df_mpg = df_mpg.reindex(cat_order) ## 행 순서 index 순서로 재정렬 하기
sns.lineplot(data = df_mpg, x = 'category', y = 'count_cat')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
```

Out[82]: <Axes: xlabel='category', ylabel='count_cat'>



시계열 그래프 만들기

economics.csv 데이터 사용

> 월별 실업자 정보

```
In [83]: ## economics 데이터 불러오기
import pandas as pd

economics = pd.read_csv('economics.csv')
economics
```

```
Out[83]:
```

	date	pce	pop	psavert	uempmed	unemploy
0	1967-07-01	506.7	198712.000	12.6	4.5	2944
1	1967-08-01	509.8	198911.000	12.6	4.7	2945
2	1967-09-01	515.6	199113.000	11.9	4.6	2958
3	1967-10-01	512.2	199311.000	12.9	4.9	3143
4	1967-11-01	517.4	199498.000	12.8	4.7	3066
...
569	2014-12-01	12062.0	319746.157	7.6	12.9	8717
570	2015-01-01	12046.0	319928.646	7.7	13.2	8903
571	2015-02-01	12082.4	320074.511	7.9	12.9	8610
572	2015-03-01	12158.3	320230.786	7.4	12.0	8504
573	2015-04-01	12193.8	320402.295	7.6	11.5	8526

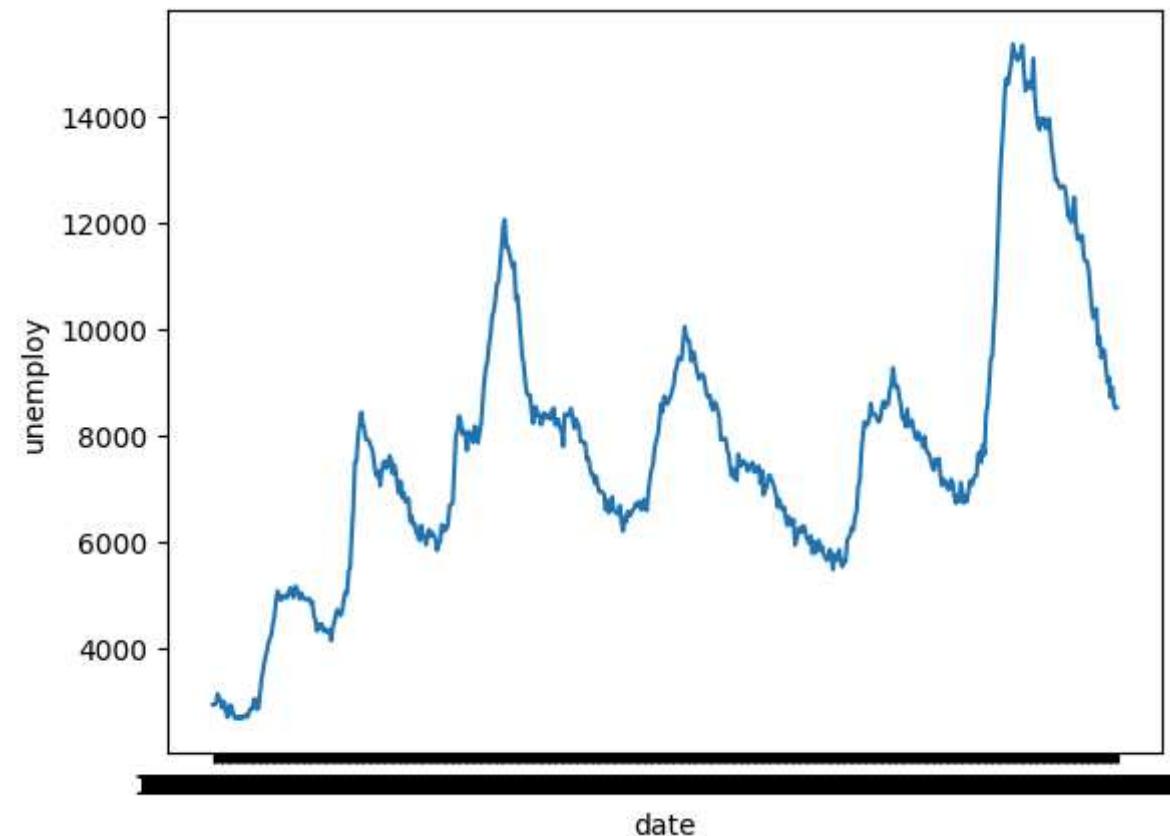
574 rows × 6 columns

```
In [84]: ## lineplot()로 시계열 그래프 만들기
import seaborn as sns

sns.lineplot(data = economics, x = 'date', y = 'unemploy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
```

Out[84]: <Axes: xlabel='date', ylabel='unemploy'>



x축에 연도 표시하기

(1) 날짜 시간 타입 변수 만들기

```
In [45]: ## 'date' 변수 타입 확인 >> Object(문자) 타입  
economics.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 574 entries, 0 to 573  
Data columns (total 6 columns):  
 # ... Column ... Non-Null Count Dtype ...  
--- ...  
 0 ... date ... 574 non-null ... object  
 1 ... pce ... 574 non-null ... float64  
 2 ... pop ... 574 non-null ... float64  
 3 ... psavert ... 574 non-null ... float64  
 4 ... uempmed ... 574 non-null ... float64  
 5 ... unemploy ... 574 non-null ... int64  
 dtypes: float64(4), int64(1), object(1)  
 memory usage: 27.0+ KB
```

```
In [90]: ## economics의 'date' 값 확인  
economics['date']
```

```
Out[90]: 0 ... 1967-07-01  
1 ... 1967-08-01  
2 ... 1967-09-01  
3 ... 1967-10-01  
4 ... 1967-11-01  
...  
569 ... 2014-12-01  
570 ... 2015-01-01  
571 ... 2015-02-01  
572 ... 2015-03-01  
573 ... 2015-04-01  
Name: date, Length: 574, dtype: object
```

```
In [137...]: ## 날짜 자료형에서 년, 월, 일 추출하기  
economics['date'].dt.year # 'date' 열의 type()은 문자열
```

```
-----  
AttributeError: Traceback (most recent call last)  
~\AppData\Local\Temp\ipykernel_21880\2169447457.py in <module>  
    1 ## 날짜 자료형에서 년, 월, 일 추출하기  
----> 2 economics['date'].dt.year # 'date' 열의 type()은 문자열  
  
~\Anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)  
    6200     ):  
    6201         return self[name]  
-> 6202         return object.__getattribute__(self, name)  
    6203  
    6204     @final  
  
~\Anaconda3\lib\site-packages\pandas\core\accessor.py in __get__(self, obj, cls)  
    222             # we're accessing the attribute of the class, i.e., Dataset.geo  
    223             return self._accessor  
--> 224         accessor_obj = self._accessor(obj)  
    225             # Replace the property with the accessor object. Inspired by:  
    226             # https://www.pydanny.com/cached-property.html  
  
~\Anaconda3\lib\site-packages\pandas\core\indexes\accessors.py in __new__(cls, data)  
    606             return PeriodProperties(data, orig)  
    607  
--> 608         raise AttributeError("Can only use .dt accessor with datetimelike values")  
  
AttributeError: Can only use .dt accessor with datetimelike values
```

In [92]: `## datetime 데이터 초기화하기
import pandas as pd`

```
dt = pd.to_datetime('2024/04/08 19:40:01')  
dt.year  
dt.month  
dt.day  
dt.hour  
dt.minute  
dt.second
```

Out[92]: 1

In [93]: `## 날짜 시간 타입 변수 만들기
economics['date2'] = pd.to_datetime(economics['date']) # 날짜 시간 타입으로 바꾸어 생성`

```
# 변수 타입 확인  
economics.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 574 entries, 0 to 573  
Data columns (total 7 columns):  
 # ... Column ... Non-Null Count Dtype ...  
--- ...  
 0 ... date ... 574 non-null ... object ...  
 1 ... pce ... 574 non-null ... float64 ...  
 2 ... pop ... 574 non-null ... float64 ...  
 3 ... psavert ... 574 non-null ... float64 ...  
 4 ... uempmed ... 574 non-null ... float64 ...  
 5 ... unemploy ... 574 non-null ... int64 ...  
 6 ... date2 ... 574 non-null ... datetime64[ns]  
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)  
memory usage: 31.5+ KB
```

```
In [94]: ## 날짜 자료형에서 년, 월, 일 추출하기  
year = economics['date2'].dt.year  
month = economics['date2'].dt.month  
day = economics['date2'].dt.day  
print(year, month, day)
```

```
0      1967
1      1967
2      1967
3      1967
4      1967
...
569    2014
570    2015
571    2015
572    2015
573    2015
Name: date2, Length: 574, dtype: int32 0      7
1      8
2      9
3      10
4     11
...
569    12
570    1
571    2
572    3
573    4
Name: date2, Length: 574, dtype: int32 0      1
1      1
2      1
3      1
4      1
...
569    1
570    1
571    1
572    1
573    1
Name: date2, Length: 574, dtype: int32
```

'year' 열변수 추가

> 'date' 컬럼의 년도 부분만 추출하여 적용

> 'year' 열의 데이터 타입은 **integer**

```
In [95]: ## 'date2'에서 년도를 추출하여 'year' 열 변수 만들기
economics['year'] = economics['date2'].dt.year
economics.head()
```

Out[95]:

	date	pce	pop	psavert	uempmed	unemploy	date2	year
0	1967-07-01	506.7	198712.0	12.6	4.5	2944	1967-07-01	1967
1	1967-08-01	509.8	198911.0	12.6	4.7	2945	1967-08-01	1967
2	1967-09-01	515.6	199113.0	11.9	4.6	2958	1967-09-01	1967
3	1967-10-01	512.2	199311.0	12.9	4.9	3143	1967-10-01	1967
4	1967-11-01	517.4	199498.0	12.8	4.7	3066	1967-11-01	1967

In [96]: `economics.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 574 entries, 0 to 573
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype  
 --- 
 0   date       574 non-null    object 
 1   pce        574 non-null    float64 
 2   pop        574 non-null    float64 
 3   psavert    574 non-null    float64 
 4   uempmed    574 non-null    float64 
 5   unemploy   574 non-null    int64  
 6   date2      574 non-null    datetime64[ns]
 7   year       574 non-null    int32  
dtypes: datetime64[ns](1), float64(4), int32(1), int64(1), object(1)
memory usage: 33.8+ KB
```

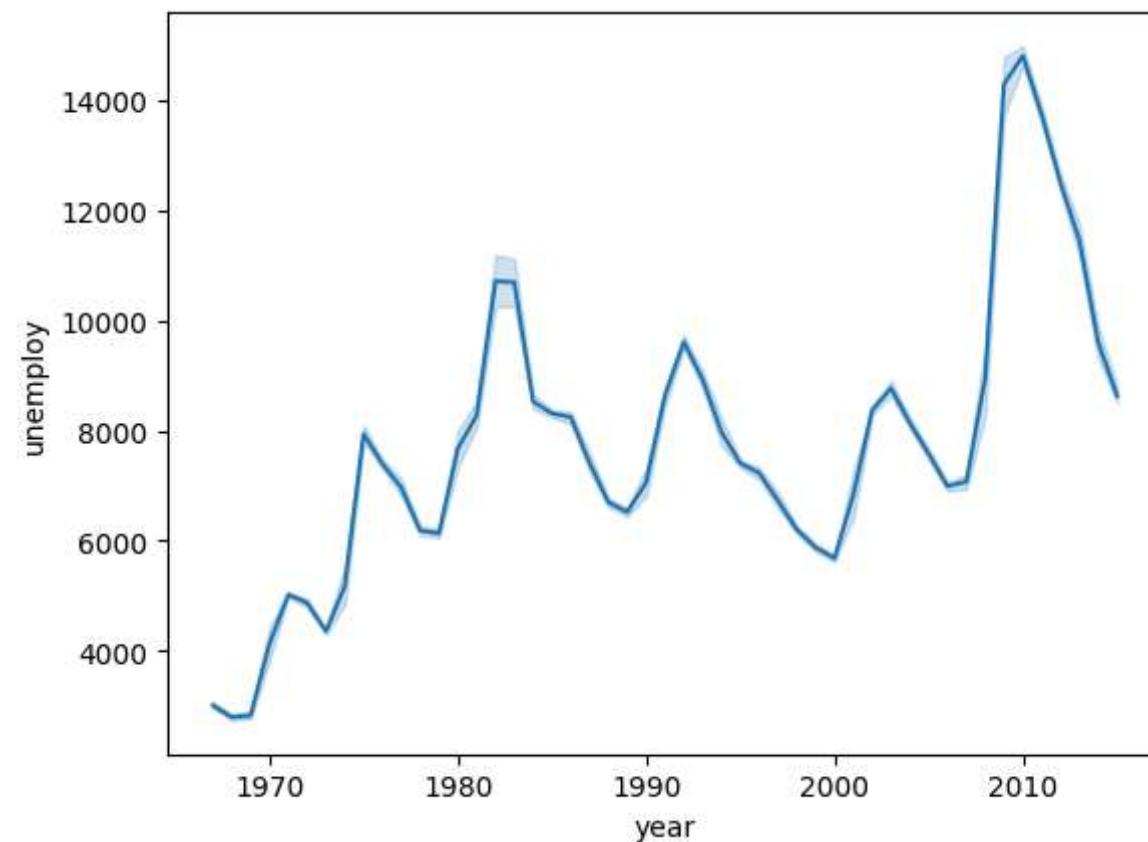
(2) x축에 연도 표시하기

In [97]: `## x축에 연도 표시`

```
sns.lineplot(data = economics, x = 'year', y = 'unemploy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
```

Out[97]: <Axes: xlabel='year', ylabel='unemploy'>



In [100...]: ## 신뢰구간 제거: confidence interval

```
sns.lineplot(data = economics, x = 'year', y = 'unemploy', ci = None)
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_23692\3599886570.py:2: FutureWarning:
```

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

```
... sns.lineplot(data = economics, x = 'year', y = 'unemploy', ci = None)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
... if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
... if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

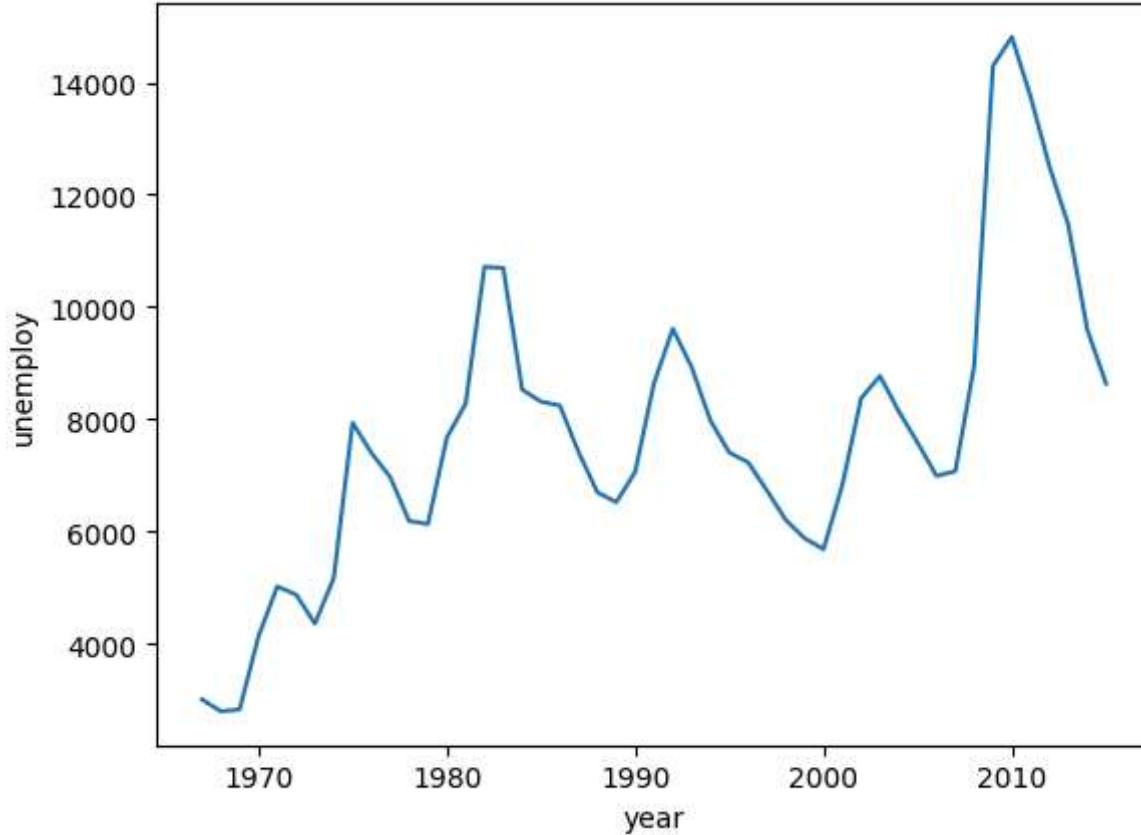
```
... with pd.option_context('mode.use_inf_as_na', True):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

```
... with pd.option_context('mode.use_inf_as_na', True):
```

```
<Axes: xlabel='year', ylabel='unemploy'>
```

Out[100]:



```
In [101]: #date 변수의 속성을 datetime으로 바꾸어 그래프 그리기  
economics['date'] = pd.to_datetime(economics['date']) #날짜 시간 타입으로 바꾸어 생성  
sns.lineplot(data = economics, x = 'date', y = 'unemploy', ci = None)
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_23692\1162804648.py:3: FutureWarning:
```

```
The 'ci' parameter is deprecated. Use `errorbar=None` for the same effect.
```

```
...sns.lineplot(data = economics, x = 'date', y = 'unemploy', ci = None)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
...if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
```

```
...if pd.api.types.is_categorical_dtype(vector):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

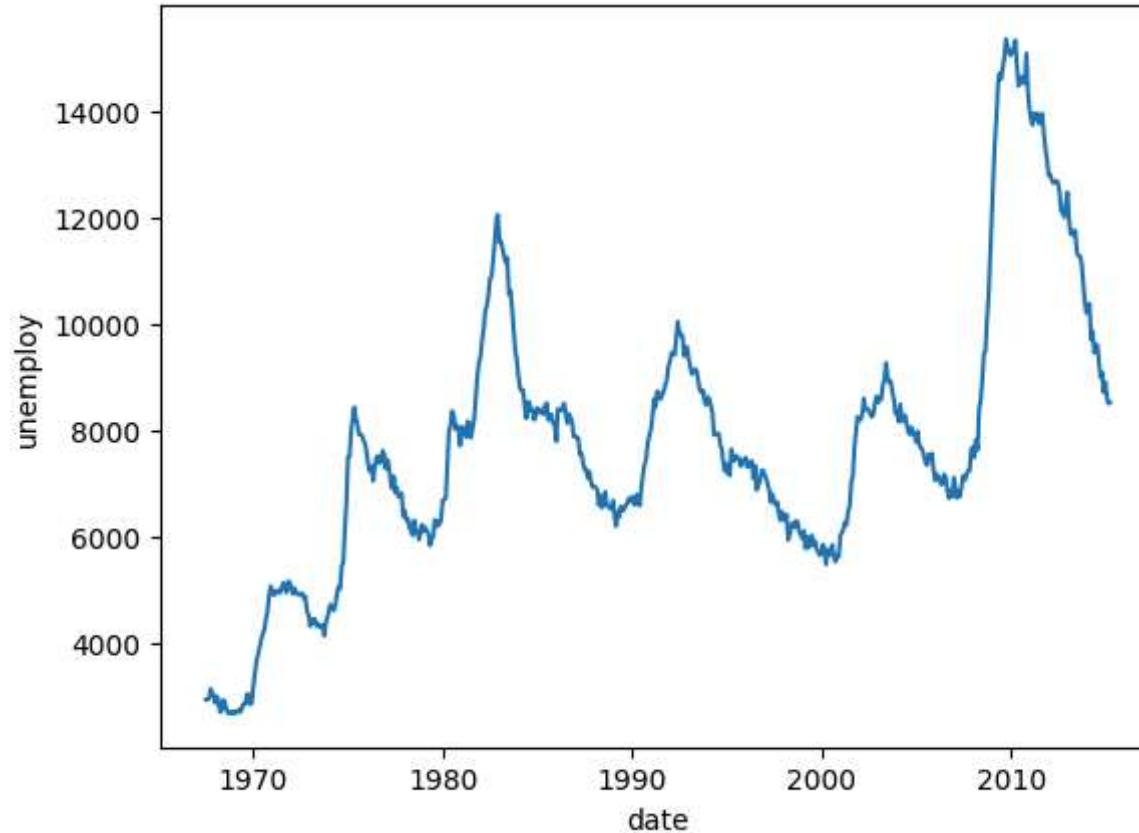
```
...with pd.option_context('mode.use_inf_as_na', True):
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

```
...with pd.option_context('mode.use_inf_as_na', True):
```

```
<Axes: xlabel='date', ylabel='unemploy'>
```

```
Out[101]:
```



```
In [104]: economics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 574 entries, 0 to 573
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   date      574 non-null    datetime64[ns]
 1   pce       574 non-null    float64 
 2   pop       574 non-null    float64 
 3   psavert   574 non-null    float64 
 4   uempmed   574 non-null    float64 
 5   unemploy  574 non-null    int64   
 6   date2     574 non-null    datetime64[ns]
 7   연도      574 non-null    int32  
dtypes: datetime64[ns](2), float64(4), int32(1), int64(1)
memory usage: 33.8 KB
```

(알아 두면 좋아요) seaborn 더 알아보기

한글 표현하기

In [102...]

```
## 맑은 고딕 폰트 설정
import matplotlib.pyplot as plt
plt.rcParams.update({'font.family': 'Malgun Gothic'})
```

In [103...]

```
## [Data Frame] 관련 함수 : 컬럼명 변경
economics = economics.rename(columns = {'year' : '연도'})          # 컬럼명 변경
economics.head()
```

Out[103]:

	date	pce	pop	psavert	uempmed	unemploy	date2	연도
0	1967-07-01	506.7	198712.0	12.6	4.5	2944	1967-07-01	1967
1	1967-08-01	509.8	198911.0	12.6	4.7	2945	1967-08-01	1967
2	1967-09-01	515.6	199113.0	11.9	4.6	2958	1967-09-01	1967
3	1967-10-01	512.2	199311.0	12.9	4.9	3143	1967-10-01	1967
4	1967-11-01	517.4	199498.0	12.8	4.7	3066	1967-11-01	1967

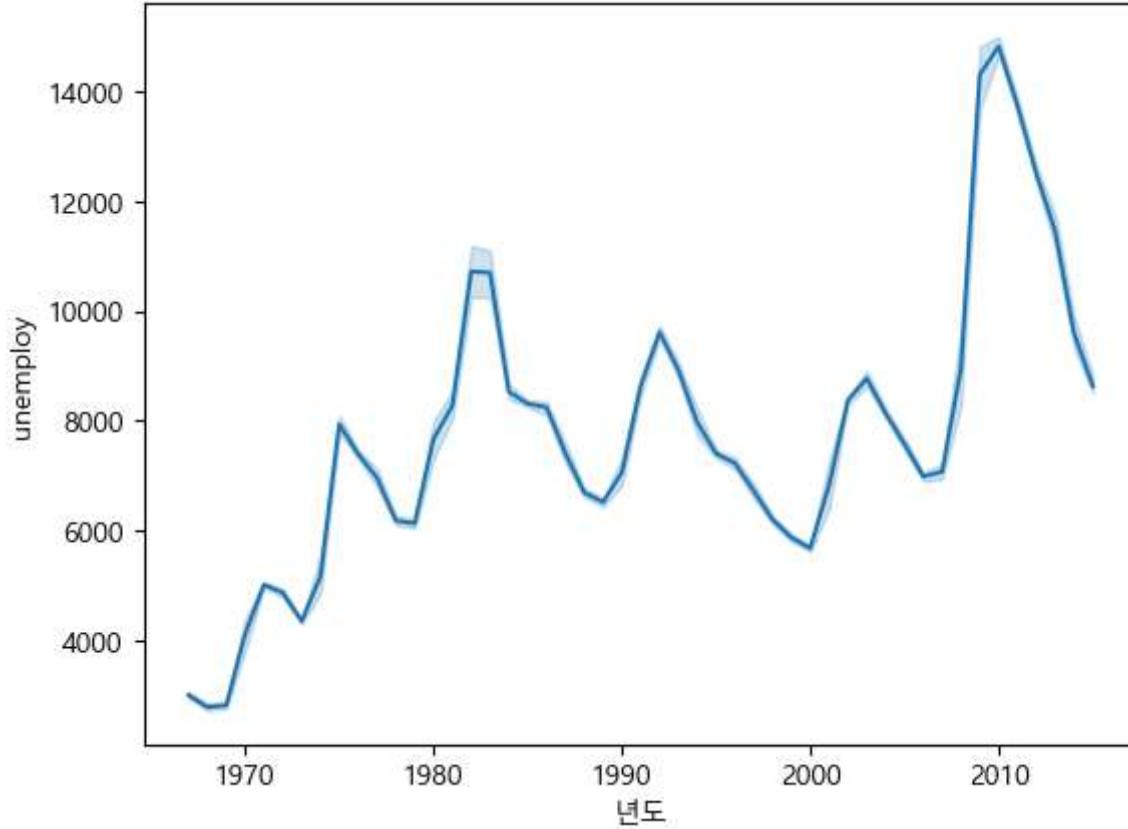
In [105]:

```
## x축에 연도 표시  
sns.lineplot(data = economics, x = '년도', y = 'unemploy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead  
.. if pd.api.types.is_categorical_dtype(vector):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.  
.. with pd.option_context('mode.use_inf_as_na', True):  
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.  
.. with pd.option_context('mode.use_inf_as_na', True):
```

Out[105]:

```
<Axes: xlabel='년도', ylabel='unemploy'>
```



matplotlib 라이브러리로 그리기

plot() 이용

- > color : b(blue), g(green), r(red), w(white), y(yellow), C(청록), k(검정)
- > linestyle : -(실선), --(파선), -(일점쇄선), :(점선)
- > marker : o(원), +(+), D(diamond), s(square), ^(삼각형), v(역삼각형), .(점)

다중 그래프 그리기

```
In [108]: ## 다중 그래프 그리기
import matplotlib.pyplot as plt
```

```
# 데이터 생성
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [1, 3, 5, 7, 9]
y3 = [3, 6, 3, 9, 5]

# 그래프 그리기
plt.figure(figsize=(8, 6)) # 그림 크기 설정 (inch, 선택사항)

# 첫 번째 꺥은선 그래프
plt.plot(x, y1, label='선 그래프 1', marker='o', linestyle='-', color='b')

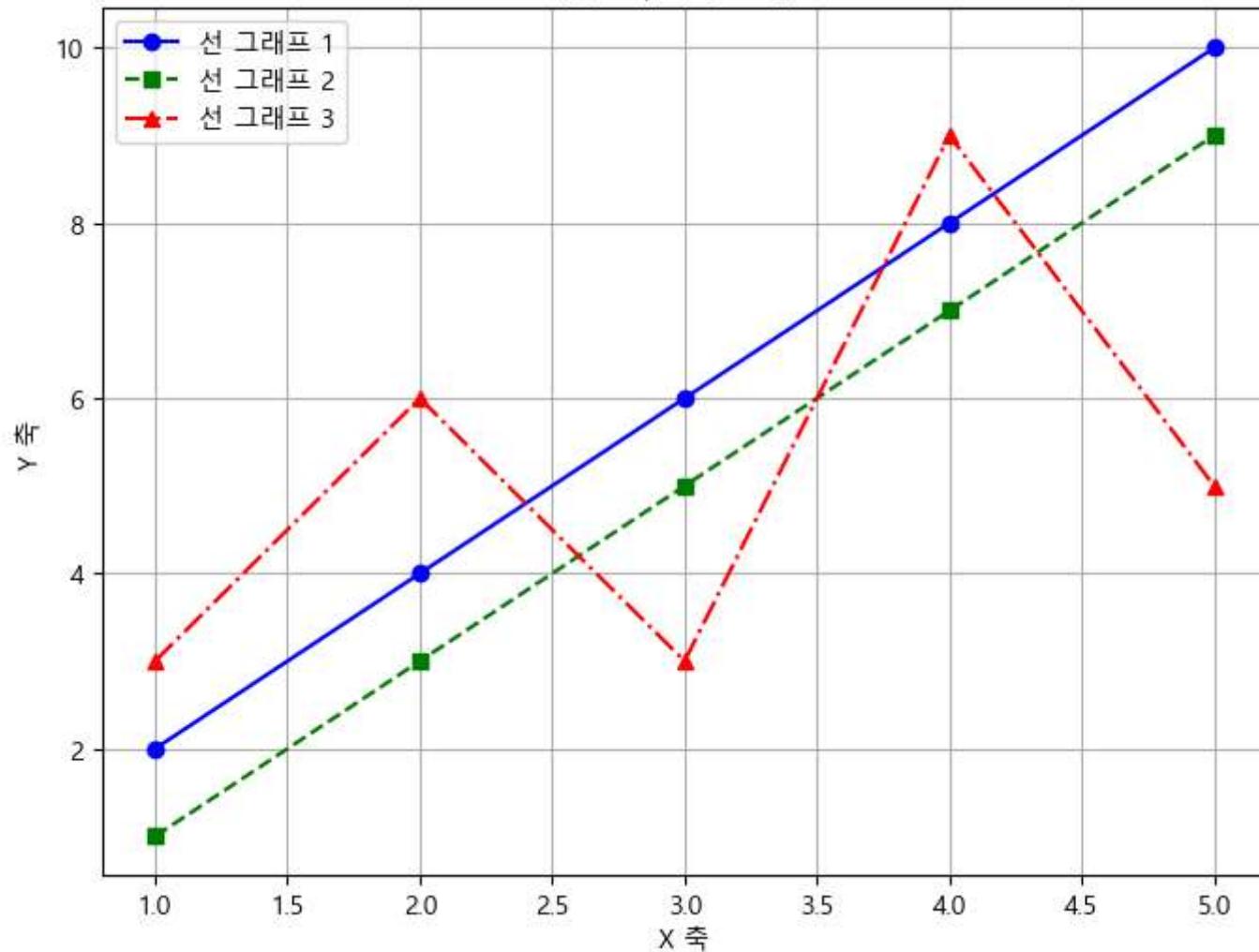
# 두 번째 꺥은선 그래프
plt.plot(x, y2, label='선 그래프 2', marker='s', linestyle='--', color='g')

# 세 번째 꺥은선 그래프
plt.plot(x, y3, label='선 그래프 3', marker='^', linestyle='-.', color='r')

# 그래프 제목 및 레이블 설정
plt.title('다중 꺥은선 그래프')
plt.xlabel('X 축')
plt.ylabel('Y 축')
plt.legend() # 범례 표시
plt.grid(True) # 그리드 표시 (선택사항)

# 그래프 표시
plt.show()
```

다중 깍은선 그래프



```
In [58]: ##다중 그래프 그리기
import matplotlib.pyplot as plt

# 그래프 그리기
plt.figure(figsize=(8, 6)) # 그림 크기 설정 (선택사항)

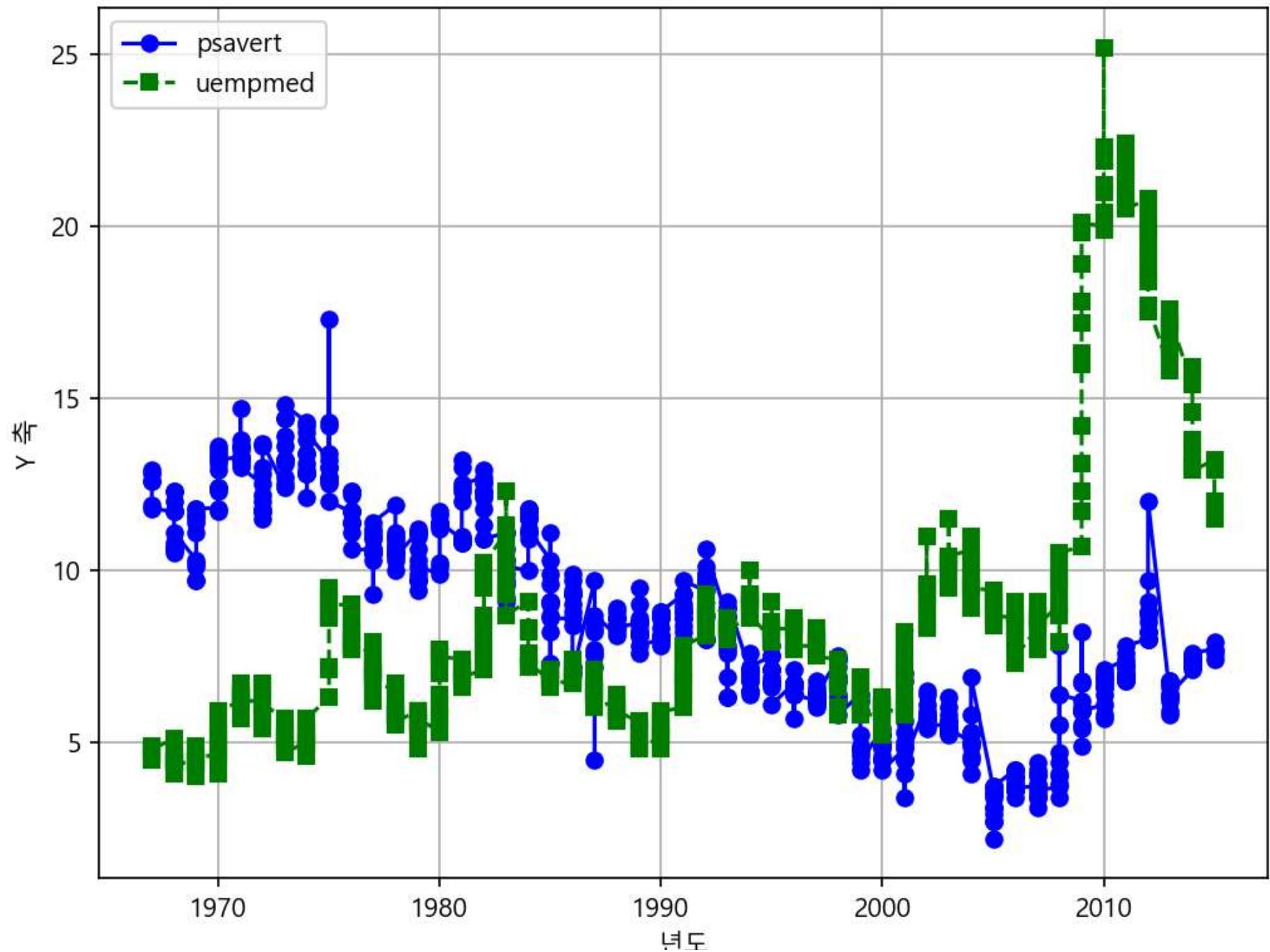
# 첫 번째 깍은선 그래프
plt.plot(economics['년도'], economics['psavert'], label='psavert', marker='o', linestyle='--', color='b')
```

```
# 두 번째 꺥은선 그래프
plt.plot(economics['년도'], economics['uempmed'], label='uempmed', marker='s', linestyle='--', color='g')

# 그래프 제목 및 레이블 설정
plt.title('다중 꺥은선 그래프')
plt.xlabel('년도')
plt.ylabel('Y 축')
plt.legend() # 범례 표시

# 그래프 표시
plt.grid(True) # 그리드 표시 (선택사항)
plt.show()
```

다중 겹은선 그래프



In []:

08-5 상자 그림 - 집단 간 분포 차이 표현하기

boxplot()으로 상자 그림 만들기

<boxplot 해석>

- > Outer 극단치
- > Maximum 극단치 경계 최대값
- > Upper Quartile 상위 25%(Q3 : 3사분위)
- > Median 중위(Q2 : 2사분위)
- > Lower Quartile 하위 25%(Q1 : 1사분위)
- > Minimum 극단치 경계 최솟값

<https://img1.daumcdn.net/thumb/R1280x0/?scode=mtistory2&fname=https%3A%2F%2Ft1.daumcdn.net%2Fcfile%2Ftistory%2F99D3C43359C2943722>

In [109]:

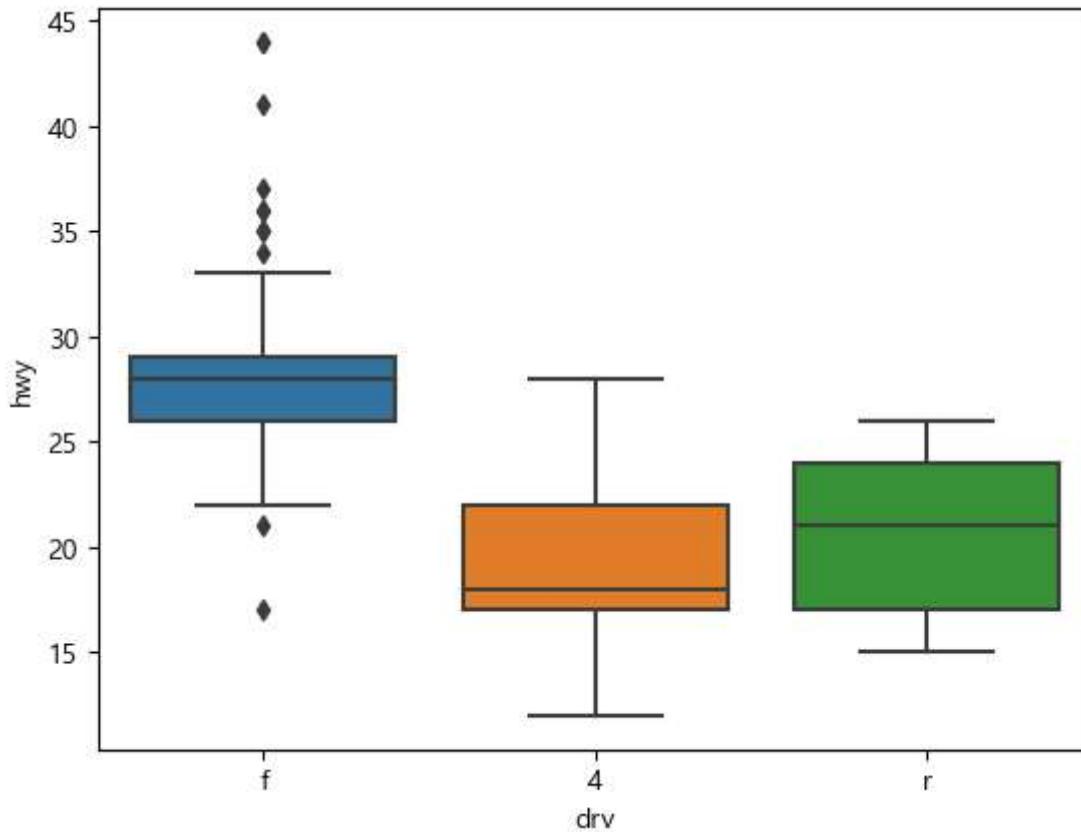
```
## boxplot()으로 상자 그림 만들기
import seaborn as sns

sns.boxplot(data = mpg, x = 'drv', y = 'hwy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
```

Out[109]:

```
<Axes: xlabel='drv', ylabel='hwy'>
```



In []:

boxplot() 구하기

(1) 1사분위수, 3사분위수 구하기

In [110]:

```
## 1사분위수 구하기  
pct25 = mpg['hwy'].quantile(.25)  
pct25
```

Out[110]: 18.0

In [111]:

```
## 3사분위수 구하기  
pct75 = mpg['hwy'].quantile(.75)  
pct75
```

```
Out[111]: 27.0
```

(2) IQR 구하기

> IQR(사분위 범위)는 1사분위수와 3사분위수의 거리

```
In [112...]: ## IQR 구하기  
iqr = pct75 - pct25  
iqr
```

```
Out[112]: 9.0
```

(3) 하한, 상한 구하기

> 하한 : $Q1 - 1.5 \times IQR$

> 상한 : $Q3 + 1.5 \times IQR$

```
In [113...]: min_line = pct25 - 1.5 * iqr # 하한  
min_line
```

```
Out[113]: 4.5
```

```
In [114...]: max_line = pct75 + 1.5 * iqr # 상한  
max_line
```

```
Out[114]: 40.5
```

```
In [115...]: ## 'hwy' 열의 Maximum과 Minimum 초기화 확인  
mpg.query('hwy < 4.5 | hwy > 40.5')
```

```
Out[115]:   manufacturer    model  displ  year cyl      trans  drv  cty  hwy fl category  
  212    volkswagen     jetta    1.9  1999    4 manual(m5)   f  33   44  d compact  
  221    volkswagen  new beetle   1.9  1999    4 manual(m5)   f  35   44  d subcompact  
  222    volkswagen  new beetle   1.9  1999    4    auto(l4)   f  29   41  d subcompact
```

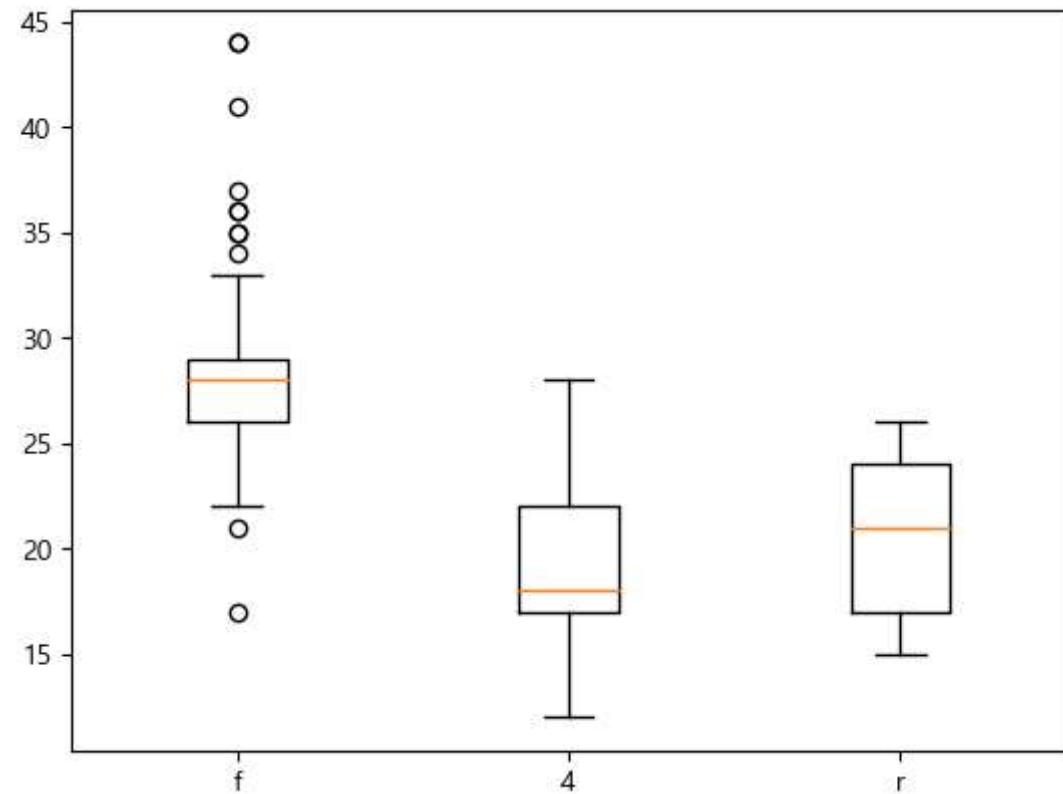
matplotlib 라이브러리로 그리기

In [116]:

```
## matplotlib 라이브러리로 그래프 그리기
import matplotlib.pyplot as plt

hwy_f = mpg[mpg['drv'] == 'f']['hwy']
hwy_4 = mpg[mpg['drv'] == '4']['hwy']
hwy_r = mpg[mpg['drv'] == 'r']['hwy']
hwy_drv = [hwy_f, hwy_4, hwy_r]

plt.boxplot(hwy_drv, labels = ['f', '4', 'r'])
plt.show()
```



In []:

08-6 점유 비율 그림 - 집단 내 점유 비율 표현하기

> .groupby()로 점유율(proportion)을 구한 다음, .pivot()한 데이터프레임 사용

pandas.plot.barh()으로 그리기

> Pandas 자체에서도 데이터프레임에 포함된 데이터를 기반으로 그래프를 그릴 수 있는 기능을 DataFrame.plot() 메서드로 제공함.

(1) 점유 비율 데이터 프레임 구성하기

> 빈도수 비율(0.0~1.0)로 normalize하여 구하기: pandas.value_counts(normalize = True)

In [119...]

```
## 'manufacturer'별 'category' 점유 비율(proportion) 구하기
df_cat = mpg.groupby('manufacturer', as_index = False)[
    ['category']].value_counts(normalize = True) # 빈도수 비율(0.0~1.0)로 구하기
df_cat.head()
```

Out[119]:

	manufacturer	category	proportion
0	audi	compact	0.833333
1	audi	midsized	0.166667
2	chevrolet	suv	0.473684
3	chevrolet	midsized	0.263158
4	chevrolet	2seater	0.263158

In [120...]

```
## 점유 비율(proportion)을 백분위로 환산하기
df_cat['proportion'] = (df_cat['proportion'] * 100).round(1)
df_cat.head()
```

Out[120]:

	manufacturer	category	proportion
0	audi	compact	83.3
1	audi	midsized	16.7
2	chevrolet	suv	47.4
3	chevrolet	midsized	26.3
4	chevrolet	2seater	26.3

(2) 피봇 데이터 프레임 만들기

> 제조사별 카테고리별 점유 비율로 피봇

pandas.pivot()

- > **index:** 새로운 데이터프레임의 행 인덱스로 사용할 열 이름이나 열 이름의 리스트
- > **columns:** 새로운 데이터프레임의 열 이름으로 사용할 열 이름이나 열 이름의 리스트
- > **values:** 새로운 데이터프레임의 데이터로 사용할 열 이름
- > **aggfunc:** 중복되는 (행, 열) 쌍에 대한 집계 함수를 지정. 기본값은 `numpy.mean`
- > **fill_value:** 결측값을 채울 값을 지정
- > **dropna:** `True`로 설정하면 결측값이 있는 행과 열을 삭제. 기본값은 `False`

In [122...]

```
### 분석을 위한 데이터 피봇:  
pivot_df = df_cat[['manufacturer', 'category', 'proportion']].pivot(index = 'manufacturer', columns = 'category', values = 'proportion')  
pivot_df
```

Out[122]:

category	2seater	compact	midsize	minivan	pickup	subcompact	suv
manufacturer							
audi	NaN	83.3	16.7	NaN	NaN	NaN	NaN
chevrolet	26.3	NaN	26.3	NaN	NaN	NaN	47.4
dodge	NaN	NaN	NaN	29.7	51.4	NaN	18.9
ford	NaN	NaN	NaN	NaN	28.0	36.0	36.0
honda	NaN	NaN	NaN	NaN	NaN	100.0	NaN
hyundai	NaN	NaN	50.0	NaN	NaN	50.0	NaN
jeep	NaN	NaN	NaN	NaN	NaN	NaN	100.0
land rover	NaN	NaN	NaN	NaN	NaN	NaN	100.0
lincoln	NaN	NaN	NaN	NaN	NaN	NaN	100.0
mercury	NaN	NaN	NaN	NaN	NaN	NaN	100.0
nissan	NaN	15.4	53.8	NaN	NaN	NaN	30.8
pontiac	NaN	NaN	100.0	NaN	NaN	NaN	NaN
subaru	NaN	28.6	NaN	NaN	NaN	28.6	42.9
toyota	NaN	35.3	20.6	NaN	20.6	NaN	23.5
volkswagen	NaN	51.9	25.9	NaN	NaN	22.2	NaN

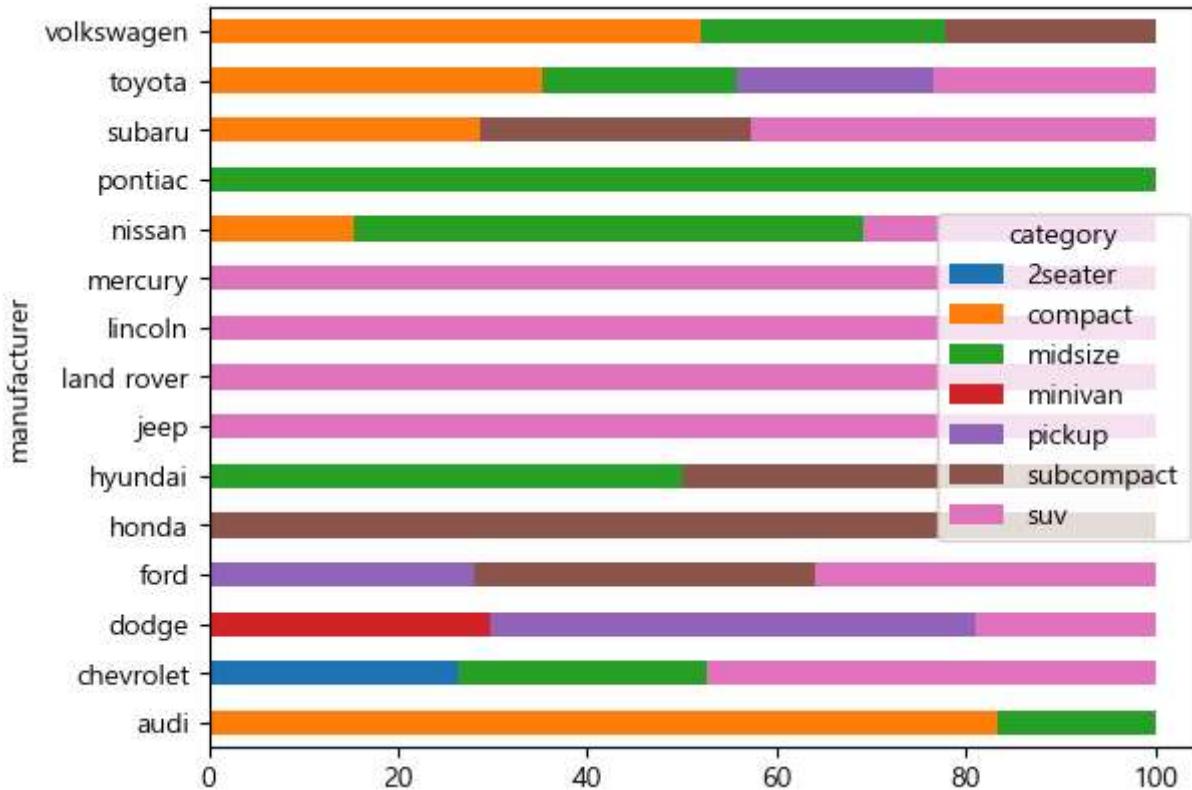
(3) 피봇 데이터 프레임으로 누적 막대 그래프 그리기

In [125...]

```
### 비율 그래프  
pivot_df.plot.barh(stacked=True) #수평, 누적 막대 그래프
```

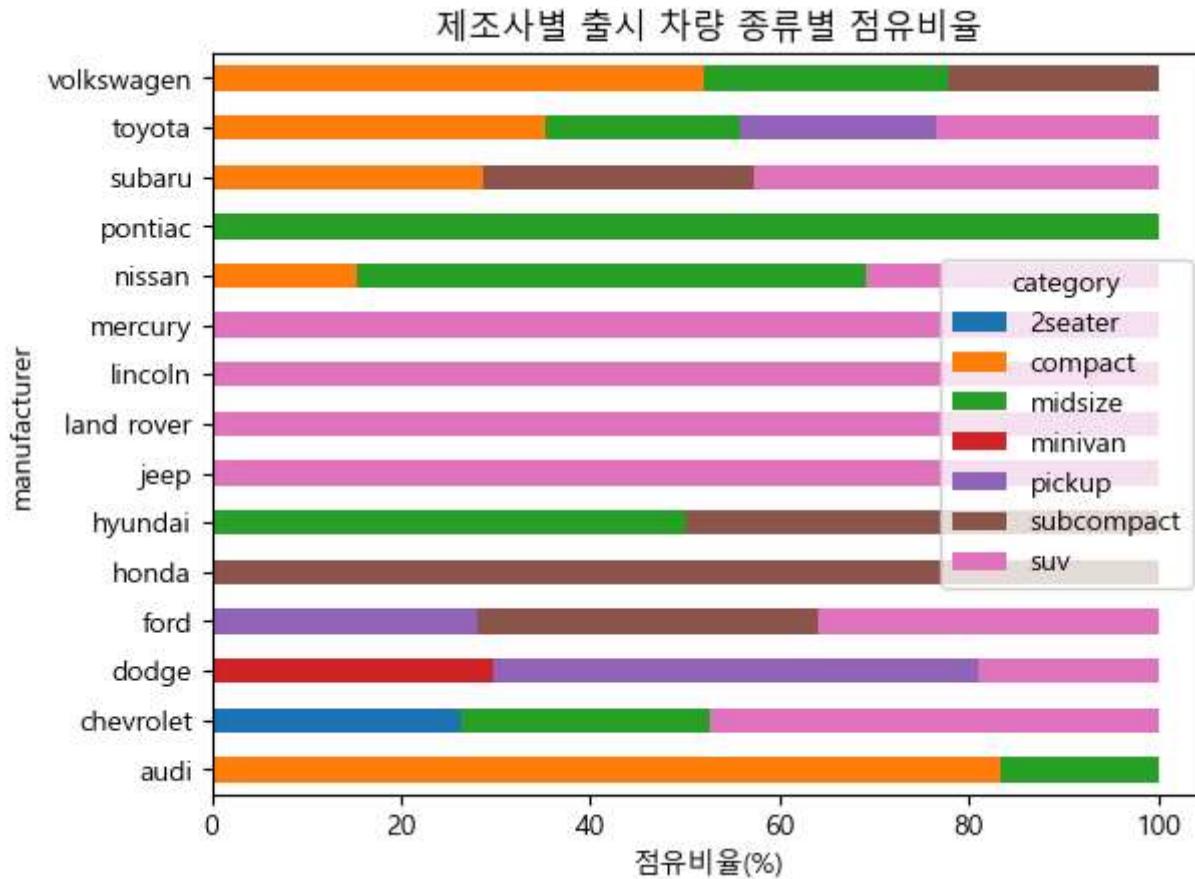
Out[125]:

```
<Axes: ylabel='manufacturer'>
```



```
In [127]: ### 비율 그래프: title, label 달기  
import matplotlib.pyplot as plt
```

```
pivot_df.plot.barh(stacked =True) #수평, 누적 막대 그래프  
plt.title("제조사별 출시 차량 종류별 점유비율")  
plt.xlabel('점유비율(%)')  
plt.show()
```



In []:

08-7 히트맵 그림 - 상관 관계 정도 표현하기

heatmap()으로 그리기

(1) 상관 랭컬용 데이터 추출하기

> 실린더 수에 따른 연비의 상관관계 확인하기

In [128...]

```
## mpg 데이터 불러오기
import pandas as pd
```

```
mpg = pd.read_csv('mpg.csv')
mpg
```

Out[128]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

234 rows × 11 columns

In [129...]

```
## 상관행렬용 데이터 추출
df_mpg = mpg[['cyl', 'cty', 'hwy']]
```

Out[129]:

	cyl	cty	hwy
0	4	18	29
1	4	21	29
2	4	20	31
3	4	21	30
4	6	16	26
...
229	4	19	28
230	4	21	29
231	6	16	26
232	6	18	26
233	6	17	26

234 rows × 3 columns

In [130...]

```
## 연비 평균 열('cty_hwy') 추가
df_mpg['total'] = (df_mpg['cty'] + df_mpg['hwy']) / 2
df_mpg
```

C:\Users\ADMIN\AppData\Local\Temp\ipykernel_23692\209784403.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_mpg['total'] = (df_mpg['cty'] + df_mpg['hwy']) / 2
```

Out[130]:

	cyl	cty	hwy	total
0	4	18	29	23.5
1	4	21	29	25.0
2	4	20	31	25.5
3	4	21	30	25.5
4	6	16	26	21.0
...
229	4	19	28	23.5
230	4	21	29	25.0
231	6	16	26	21.0
232	6	18	26	22.0
233	6	17	26	21.5

234 rows × 4 columns

(2) 상관행렬 만들기

In [131...]

```
## 상관행렬 만들기
df_corr = df_mpg.corr() # 상관행렬 만들기
df_corr = round(df_corr, 2) # 소수점 2자리까지 반올림
df_corr
```

Out[131]:

	cyl	cty	hwy	total
cyl	1.00	-0.81	-0.76	-0.79
cty	-0.81	1.00	0.96	0.98
hwy	-0.76	0.96	1.00	0.99
total	-0.79	0.98	0.99	1.00

(3) 상관 그래프 그리기

seaborn.heatmap() 이용

- > **data** (필수): 열 지도로 표현할 데이터를 포함하는 2D 배열 또는 Pandas DataFrame을 지정
- > **annot**: 각 셀에 값을 표시할지 여부를 설정(기본값은 **False**)
- > **fmt**: **annot=True**일 때 셀 내의 숫자 값의 서식을 지정
- > **cmap**: 열 지도의 색상 맵을 지정('viridis', 'coolwarm', 'Blues', 'RdBu_r' 등의 내장 색상 맵을 사용)
- > **cbar**: 색상 막대를 표시할지 여부를 설정(기본값은 **True**)
- > **cbar_kws**: 색상 막대의 추가 매개변수를 설정
- > **xticklabels** 및 **yticklabels**: 열 지도의 x축 및 y축 눈금 레이블을 표시할지 여부를 설정
- > **square**: 셀이 정사각형인지 여부를 설정
- > **linewidths** 및 **linecolor**: 셀 사이의 구분선의 두께와 색상을 설정
- > **center**: 색상 스케일의 중심을 조절
- > **vmin** 및 **vmax**: 열 지도의 색상 스케일 범위를 지정
- > **annot_kws**: **annot=True**일 때 표시된 값의 추가 매개변수를 설정
- > **mask**: 특정 값 또는 조건에 따라 특정 셀을 숨기는 데 사용
- > **ax**: 그래프를 그릴 Matplotlib 축을 지정
- > **square**: **True**로 설정하면 셀이 정사각형으로 표시

In [135...]

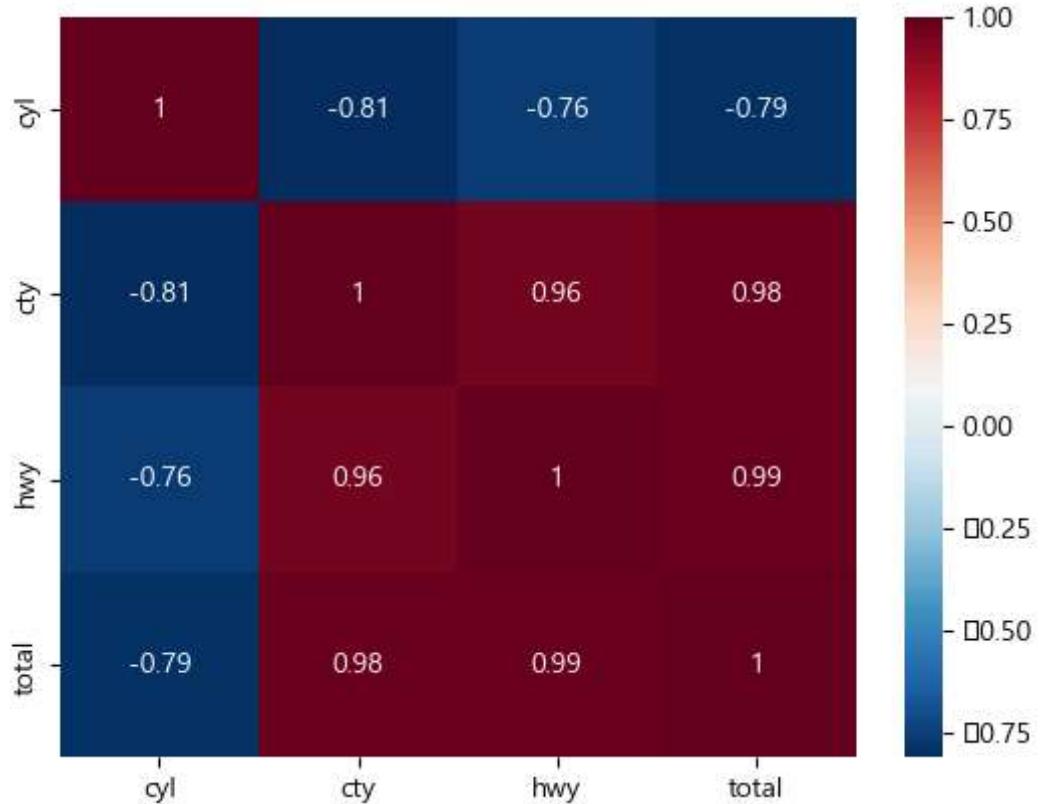
```
## 히트맵 만들기
import seaborn as sns

sns.heatmap(df_corr,
            annot = True,      # 상관계수 표시
            cmap = 'RdBu_r')  # 컬러맵 사용
```

C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\utils.py:80: UserWarning: Glyph 8722 (MINUS SIGN) missing from current font.
fig.canvas.draw()

Out[135]:

```
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 8722 (MINUS SIGN) missing from current  
font.  
func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 8722 (MINUS SIGN) missing from cu  
rrent font.  
fig.canvas.print_figure(bytes_io, **kw)
```



(4) 하단만 표시하기

(4-1) 마스크 행렬 만들기

> 마스크 행렬은 상단 제거용 행렬

In [161]:

```
## 마스크 행렬 만들기
import numpy as np

mask = np.zeros_like(df_corr) # df_corr 크기로 0이 채워진 초기화 행렬 생성
mask
```

Out[161]:

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

```
In [162...]: ## 오른쪽(위) 대각 행렬 값을 1로 바꾸기  
mask[np.triu_indices_from(mask)] = 1 #Triangle Up  
mask
```

```
Out[162]: array([[1., 1., 1., 1.],  
... [0., 1., 1., 1.],  
... [0., 0., 1., 1.],  
... [0., 0., 0., 1.]])
```

```
In [150...]: ## 왼쪽(아래) 대각 행렬 값을 1로 바꾸기  
mask[np.triu_indices_from(mask)] = 0 #Triangle Up  
mask[np.tril_indices_from(mask)] = 1 #Triangle Low  
mask
```

```
Out[150]: array([[1., 0., 0., 0.],  
... [1., 1., 0., 0.],  
... [1., 1., 1., 0.],  
... [1., 1., 1., 1.]])
```

```
In [156...]: ## 오른쪽(위) 대각 행렬 값을 1로 바꾸기  
mask[np.tril_indices_from(mask)] = 0 #Triangle Low  
mask[np.triu_indices_from(mask)] = 1 #Triangle Up  
mask
```

```
Out[156]: array([[1., 1., 1., 1.],  
... [0., 1., 1., 1.],  
... [0., 0., 1., 1.],  
... [0., 0., 0., 1.]])
```

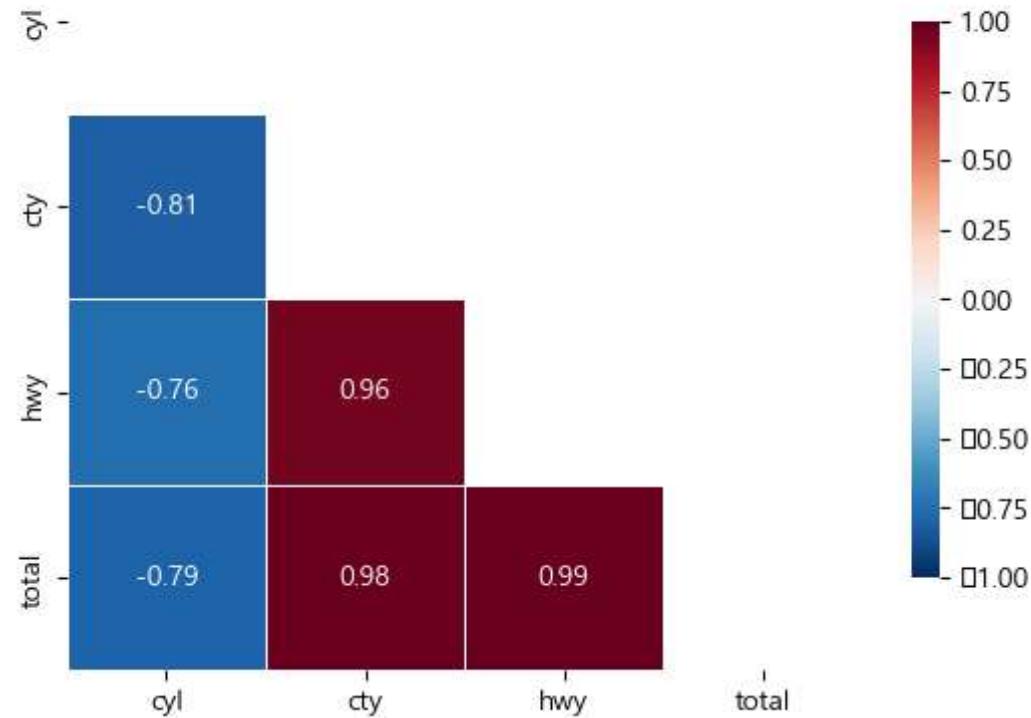
(4-2) Mask 행렬이 적용된 히트맵 그리기

```
In [163...]: ## Mask가 적용된 히트맵 만들기  
sns.heatmap(df_corr,  
            annot = True, # 상관계수 표시  
            mask = mask,  
            linewidths = 0.5, # 경계 구분선 추가  
            vmax = 1, # 진한 파란색으로 표현할 최대값  
            vmin = -1, # 진한 붉은색으로 표현할 최대값  
            cbar_kws = {'shrink': 0.75}, # 범례 크기 반으로  
            cmap = 'RdBu_r') # 컬러맵
```

C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\utils.py:80: UserWarning: Glyph 8722 (MINUS SIGN) missing from current font.
fig.canvas.draw()

```
Out[163]: <Axes: >
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 8722 (MINUS SIGN) missing from current font.  
... func(*args, **kwargs)  
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 8722 (MINUS SIGN) missing from cu  
rrent font.  
... fig.canvas.print_figure(bytes_io, **kw)
```



(4-3) 상관관계=1인 구간 제거하기

```
In [165...]
```

```
## 상관관계 = 1인 구간 제거하기  
mask_new = mask[1:, :-1] # mask의 첫 행과 마지막 열 제거  
mask_new
```

```
Out[165]: array([[0., 1., 1.],
   ... [0., 0., 1.],
   ... [0., 0., 0.]])
```

```
In [166...]: ## 상관관계 = 1인 구간 제거하기
corr_new = df_corr.iloc[1:, :-1] # 데이터에서 첫 행과 마지막 열 제거
corr_new
```

```
Out[166]:      cyl  cty  hwy
              cyl    -0.81  1.00  0.96
              cty    -0.76  0.96  1.00
              hwy    -0.79  0.98  0.99
              total  -0.79  0.98  0.99
```

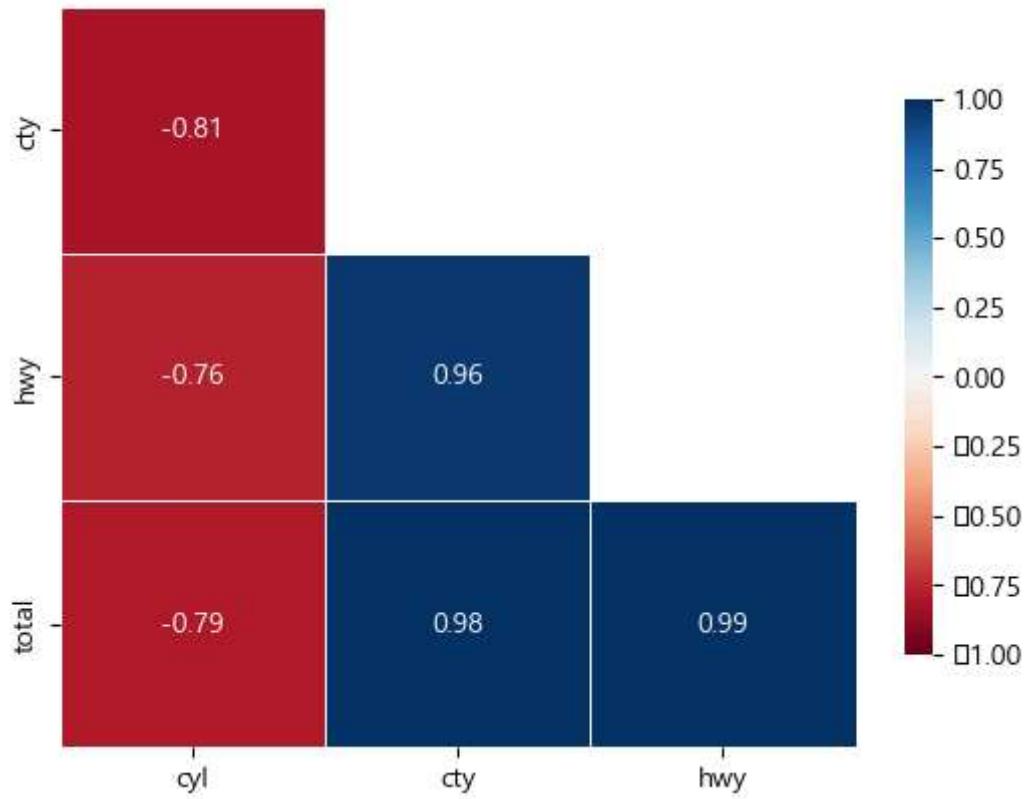
(4-4) Mask_new 행렬이 적용된 히트맵 그리기

```
In [167...]: ## Mask가 적용된 히트맵 만들기
sns.heatmap(corr_new,
             annot = True, # 상관계수 표시
             mask = mask_new,
             linewidths = 0.5, # 경계 구분선 추가
             vmax = 1, # 진한 파란색으로 표현할 최대값
             vmin = -1, # 진한 붉은색으로 표현할 최대값
             cbar_kws = {'shrink': 0.75}, # 범례 크기 반으로
             cmap = 'RdBu') # 컬러맵
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\utils.py:80: UserWarning: Glyph 8722 (MINUS SIGN) missing from current font.
  fig.canvas.draw()
```

```
Out[167]: <Axes: >
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\events.py:89: UserWarning: Glyph 8722 (MINUS SIGN) missing from current
font.
  func(*args, **kwargs)
C:\Users\ADMIN\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: UserWarning: Glyph 8722 (MINUS SIGN) missing from cu
rrent font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In []:

08-8 네트워크 그래프 그리기

> pip install networkx 필요

Node와 Edge를 추가하고 그리기

In [144...]

```
!pip install networkx
```

Requirement already satisfied: networkx in c:\users\admin\anaconda3\lib\site-packages (2.8.4)

```
WARNING: Ignoring invalid distribution -atplotlib (c:\users\wadmin\anaconda3\lib\site-packages)
```

networkx.draw()로 그리기

- > **G** (필수): 그래프 객체를 지정
- > **pos**: 노드의 위치 정보를 지정하는 딕셔너리
- > **arrows**: 화살표를 그릴지 여부를 설정(**default**는 **False**)
- > **with_labels**: 노드 레이블을 표시할지 여부를 설정(기본값은 **True**)
- > **node_size**: 노드 크기를 설정
- > **node_color**: 노드 색상을 설정
- > **node_shape**: 노드 모양을 설정
- > **font_size**: 레이블 텍스트의 글꼴 크기를 설정
- > **font_color**: 레이블 텍스트의 색상을 설정
- > **font_weight**: 레이블 텍스트의 글꼴 두께를 설정
- > **edge_color**: 엣지(간선)의 색상을 설정
- > **width**: 엣지(간선)의 두께를 설정
- > **edge_cmap**: 엣지 색상 맵을 설정
- > **edge_vmin**, **edge_vmax**: 엣지 색상 맵의 최소값과 최대값을 설정
- > **edge_width**: 엣지(간선)의 두께를 설정
- > **alpha**: 그래프 요소의 투명도를 설정
- > **style**: 노드 및 엣지(간선) 스타일을 설정
- > **labels**: 노드 레이블을 포함하는 딕셔너리를 설정
- > **legend**: 범례를 추가할지 여부를 설정
- > **legend_title**: 범례의 제목을 설정
- > **bbox**: 그래프를 그릴 영역을 설정
- > **ax**: 그래프를 그릴 **Matplotlib** 축을 지정

In [180...]

```
## 노드와 에지 그래프 그리기
import networkx as nx
import matplotlib.pyplot as plt

# 빈 그래프 생성
G = nx.Graph()

# 노드 추가
G.add_node(1)
G.add_node(2)
G.add_node(3)
G.add_node(4)

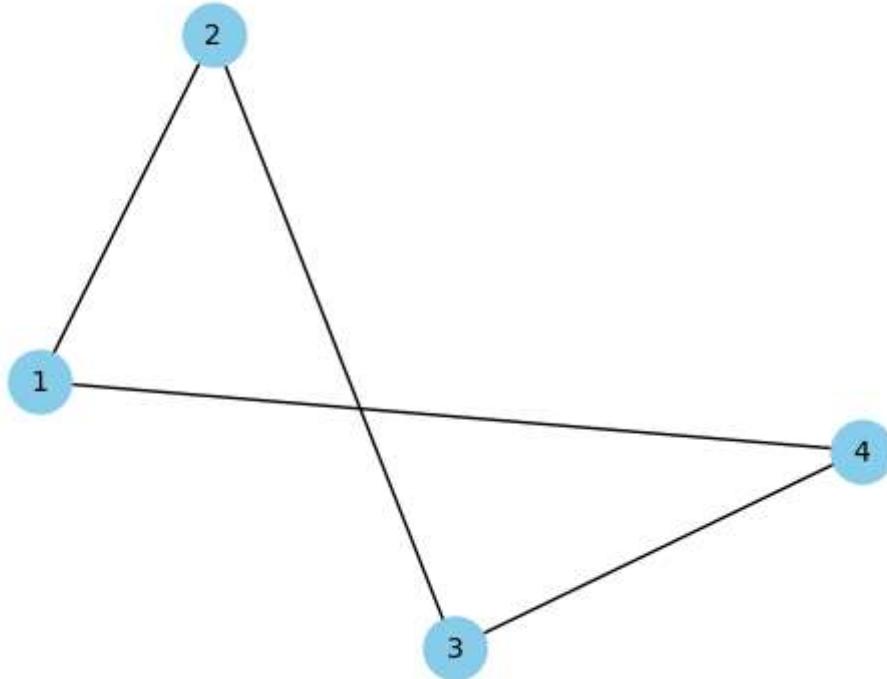
# 간선(엣지) 추가
G.add_edge(1, 2)
G.add_edge(2, 3)
G.add_edge(3, 4)
G.add_edge(4, 1)

# 하나의 서브플롯을 가진 Figure와 Axes 객체 생성
fig, ax = plt.subplots()

# 그래프 그리기
pos = nx.spring_layout(G) # 그래프 레이아웃 정의
nx.draw(G, pos, with_labels=True, node_size=500, node_color="skyblue", font_size=10, font_color="black", ax=ax)
plt.title("네트워크 그래프 예제")

plt.show()
```

네트워크 그래프 예제



In [169...]

```
## 한글 폰트 지정하기
import matplotlib.pyplot as plt

plt.rcParams.update({'font.family' : 'Malgun Gothic'}) # 폰트, 기본값 sans-serif
```

In [184...]

```
## 방향성 그래프 그리기
import networkx as nx
import matplotlib.pyplot as plt

# 네트워크 데이터 생성
nlist = ['A', 'B', 'C', 'D', 'E']
elist = [('A', 'B'), ('B', 'C'), ('C', 'D'), ('C', 'A')]
G = nx.DiGraph()
G.add_nodes_from(nlist)
G.add_edges_from(elist)

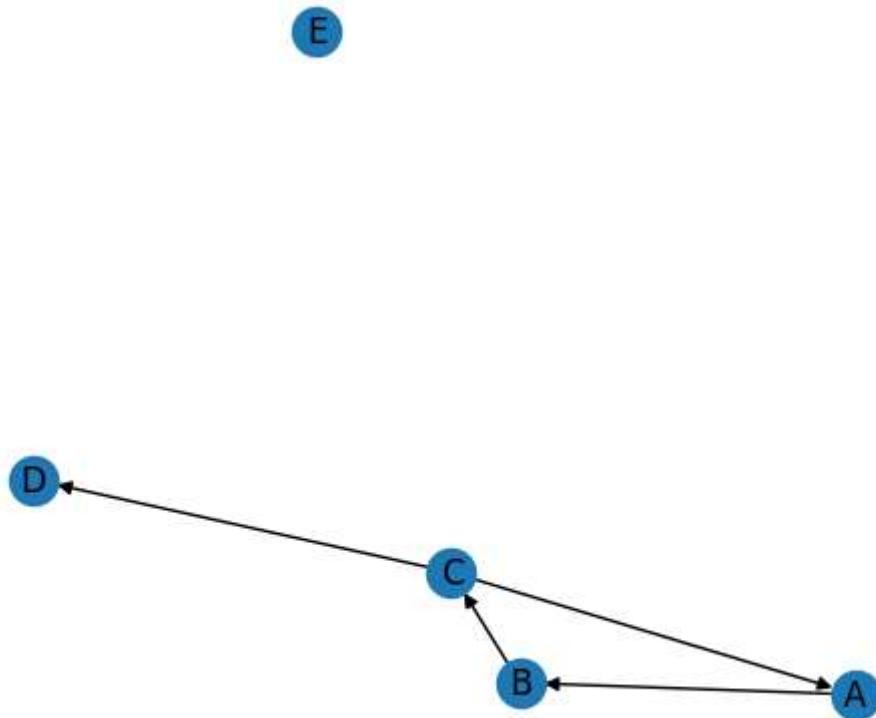
# 하나의 서브플롯을 가진 Figure와 Axes 객체 생성
```

```
fig, ax = plt.subplots()

# 네트워크 그래프 그리기
pos = nx.spring_layout(G) # 그래프 레이아웃 정의
nx.draw(G, pos, with_labels=True, ax=ax)
plt.title("네트워크(방향성) 그래프 예제")

plt.show()
```

네트워크(방향성) 그래프 예제



```
In [185...]: ## 방향성 그래프 그리기 : Node size 조절하기
import networkx as nx
import matplotlib.pyplot as plt

# 네트워크 데이터 생성
nlist = ['A', 'B', 'C', 'D', 'E']
elist = [('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'E')]
G = nx.DiGraph()
G.add_nodes_from(nlist)
```

```
G.add_edges_from(elist)

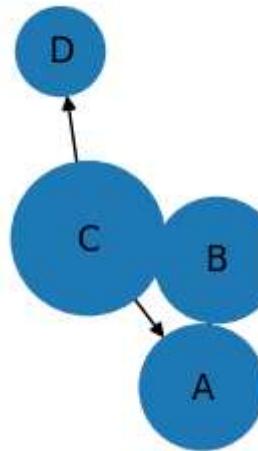
#Node size 구하기
d = dict(G.degree)
nsize = [v * 1000 for v in d.values()]

# 하나의 서브플롯을 가진 Figure와 Axes 객체 생성
fig, ax = plt.subplots()

# 네트워크 그래프 그리기
pos = nx.spring_layout(G) # 그래프 레이아웃 정의
nx.draw(G, pos, with_labels=True, node_size=nsize, ax=ax)
plt.title("네트워크(방향성) 그래프 예제")

plt.show()
```

네트워크(방향성) 그래프 예제



E

In []:

[과제] Ch08 그래프 그리기

: csv > Data Frame > Data Frame 파생변수 추가 > 그래프 그리기

1. csv > Data Frame 데이터 변환

- > Test_result2.csv 파일을 데이터 프레임으로 변환
- > index는 제외하고 변환
- > 변환된 데이터 프레임의 정보 확인

2. 그룹별 통계 데이터 프레임 만들기

- >'Grade'별 학생수 구하기

3. 'Grade'별 'Cnt_men'의 빈도수에 대한 막대 그래프 그리기

- > 'A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'F' 순으로 그리기

4. 'Grade'별 'Cnt_men'의 빈도수에 대한 선 그래프 그리기

- > 'A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'F' 순으로 그리기

5. 파이 그래프 그리기

- > 등급별 학생 수(Cnt_men)에 대한 파이그래프 그리기

6. Data Frame에 'Grade_group' 파생변수를 추가

- > 'Grade_group'는 'Grade'에 대한 등급 그룹 : ['A+', 'A'] : 'A', ['B+', 'B'] : 'B', ['C+', 'C'] : 'C', ['D+', 'D'] : 'D', ['F'] : 'F'

7. 'Grade' 그룹별 통계 프레임 만들기

- >'Grade_group'별 학생수 'Cnt_men' 구하기
- >'Grade_group'별 학생수 비율 'Rate_men' 구하기

8. 파이 그래프 그리기

- > 'Grade_group'별 학생 수(Cnt_men)에 대한 파이그래프 그리기

9. 막대 그래프 그리기

- > 'Class'별 'Grade'의 빈도 수 비교하는 막대 그래프 그리기

10. 분포 상자 그래프 그리기

> 'Class' 간의 총점 'Total'에 분포 비교 상자 그래프 그리기

```
In [51]: ##### 1. csv > Data Frame 데이터 변환
##### > Test_result2.csv 파일을 데이터 프레임으로 변환
##### > index는 제외하고 변환
import pandas as pd

df_test = pd.read_csv('Test_result2_csv.csv')
df_test
```

Out[51]:

	ID	Name	Class	Attendance	Homework	Midterm	Final	Total	Grade
0	2022001	고○○	B	14	17	22	22	75	C+
1	2022002	석○○	A	18	20	27	25	90	A
2	2022003	강○○	A	20	19	22	25	86	B+
3	2022004	민○○	A	19	19	24	26	88	B+
4	2022005	지○○	A	20	19	24	23	86	B+
...
69	2022070	유○○	B	18	19	20	23	80	B
70	2022071	김○○	B	20	19	23	19	81	B
71	2022072	이○○	B	18	17	23	24	82	B
72	2022073	은○○	A	14	18	29	26	87	B+
73	2022074	김○○	B	20	17	28	27	92	A

74 rows × 9 columns

```
In [52]: ##### 1. csv > Data Frame 데이터 변환
##### > 변환된 데이터 프레임의 정보 확인
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74 entries, 0 to 73
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   ID          74 non-null    int64  
 1   Name         74 non-null    object  
 2   Class        74 non-null    object  
 3   Attendance   74 non-null    int64  
 4   Homework     74 non-null    int64  
 5   Midterm      74 non-null    int64  
 6   Final         74 non-null    int64  
 7   Total         74 non-null    int64  
 8   Grade         74 non-null    object  
dtypes: int64(6), object(3)
memory usage: 5.3+ KB
```

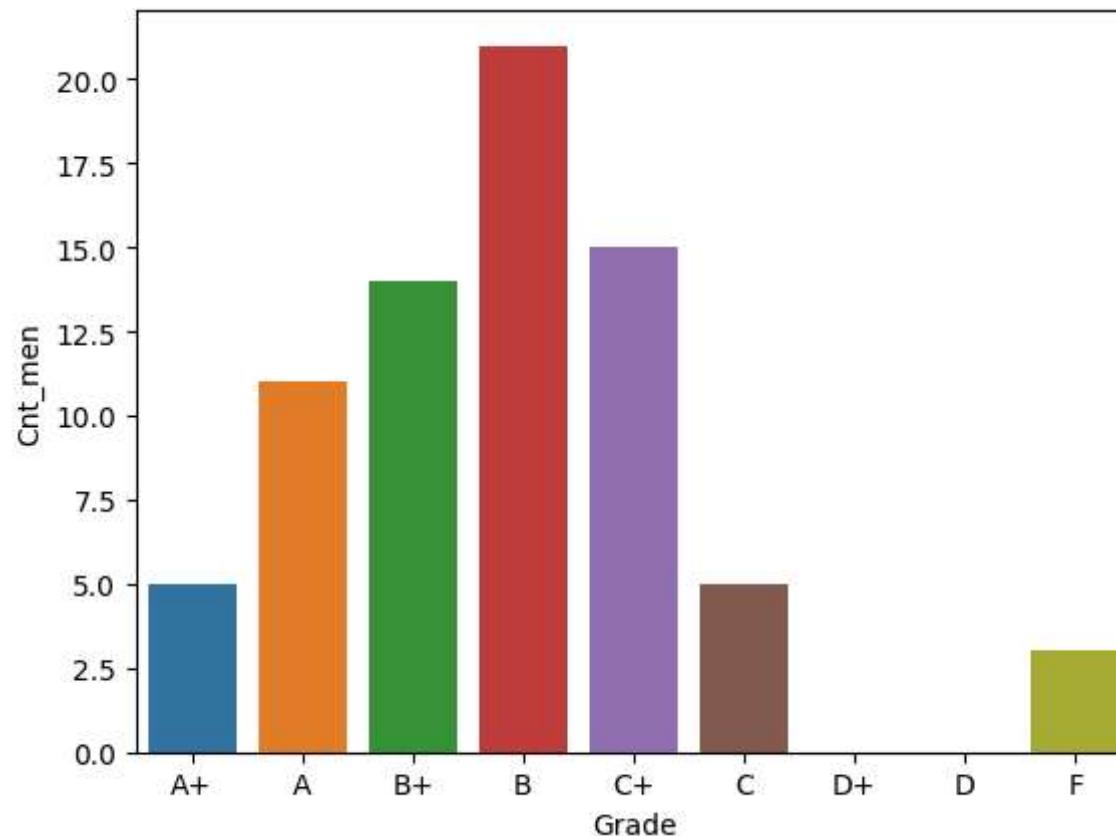
```
In [53]: ##### 2. 그룹별 통계 데이터 프레임 만들기
##### >'Grade'별 학생수 구하기
```

```
Out[53]:   Grade  Cnt_men
0       A        11
1      A+         5
2       B        21
3      B+        14
4       C         5
5      C+        15
6       F         3
```

```
In [56]: ##### 3. 'Grade'별 'Cnt_men'의 빈도수에 대한 막대 그래프 그리기
##### > 'A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'F' 순으로 그리기
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

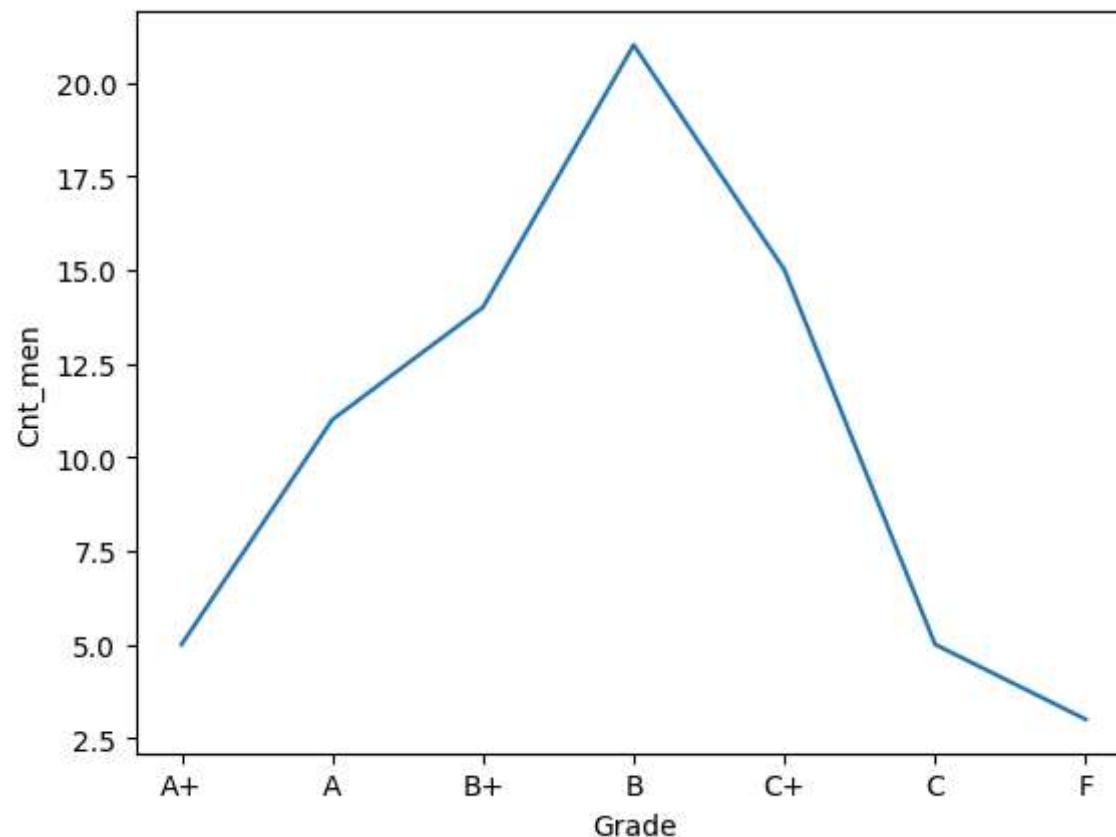
```
Out[56]: <Axes: xlabel='Grade', ylabel='Cnt_men'>
```



```
In [57]: ##### . 'Grade_group'의 빈도수에 대한 선 그래프 구하기
##### > 'A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'F' 순으로 그리기
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
.. with pd.option_context('mode.use_inf_as_na', True):
```

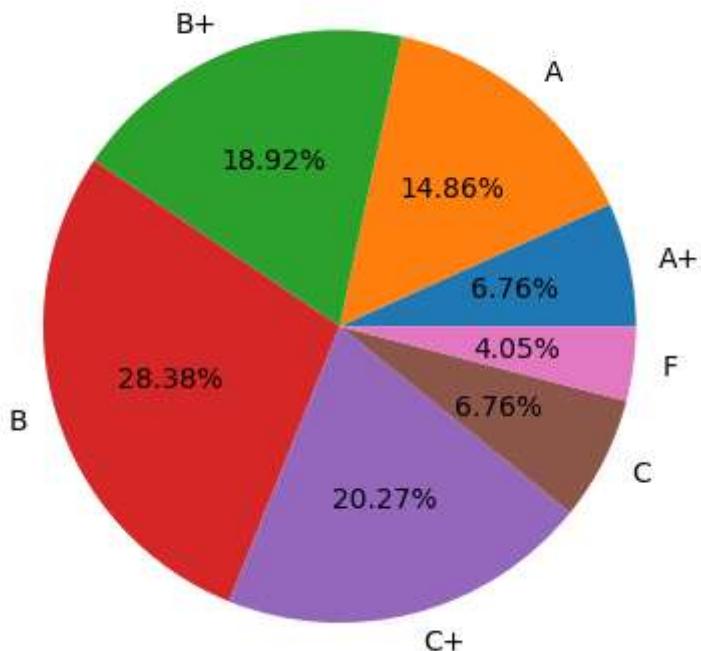
Out[57]: <Axes: xlabel='Grade', ylabel='Cnt_men'>



```
In [58]: ##### 5. 파이 그래프 그리기
```

```
##### > 등급별 학생 수(Cnt_men)에 대한 파이그래프 그리기
```

```
Out[58]: ([<matplotlib.patches.Wedge at 0x14a056f1610>,
... <matplotlib.patches.Wedge at 0x14a056f1520>,
... <matplotlib.patches.Wedge at 0x14a05701220>,
... <matplotlib.patches.Wedge at 0x14a057018b0>,
... <matplotlib.patches.Wedge at 0x14a05701f40>,
... <matplotlib.patches.Wedge at 0x14a0570f610>,
... <matplotlib.patches.Wedge at 0x14a0570fca0>],
[Text(1.0753107650415499, 0.23174718679146183, 'A+'),
Text(0.6910420111020038, 0.85558393183840636, 'A'),
Text(-0.410142132015283, 1.020677927431547, 'B+'),
Text(-1.0517833898740623, -0.32210510828769306, 'B'),
Text(0.2771759550683305, -1.0645062188319798, 'C+'),
Text(0.982218477486053, -0.4952240528134518, 'C'),
Text(1.0910904861210058, -0.1397195444317911, 'F')],
[Text(0.5865331445681181, 0.12640755643170642, '6.76%'),
Text(0.37693200605563837, 0.4668214463913073, '14.86%'),
Text(-0.22371389019015434, 0.5567334149626619, '18.92%'),
Text(-0.5737000308403976, -0.17569369542965074, '28.38%'),
Text(0.1511868845827257, -0.5806397557265344, '20.27%'),
Text(0.5357555331742107, -0.2701222106255191, '6.76%'),
Text(0.5951402651569122, -0.07621066059915878, '4.05%')])
```



```
In [59]: #### 6. Data Frame에 'Grade_group' 파생변수를 추가
##### > 'Grade_group'은 'Grade'에 대한 등급 그룹 : ['A+', 'A'] : 'A', ['B+', 'B'] : 'B',
#####      ['C+', 'C'] : 'C', ['D+', 'D'] : 'D', ['F'] : 'F'
```

Out[59]:

	Grade	Cnt_men	Grade_group
1	A+	5	A
0	A	11	A
3	B+	14	B
2	B	21	B
5	C+	15	C
4	C	5	C
6	F	3	F

```
In [60]: ##### **7. 'Grade' 그룹별 통계 프레임 만들기**
##### >'Grade_group'별 학생수 'Cnt_men' 구하기
##### >'Grade_group'별 학생수 비율 'Rate_men' 구하기
```

```
Out[60]:   Grade_group  Cnt_men  Rate_men
```

	Grade_group	Cnt_men	Rate_men
0	A	16	0.216216
1	B	35	0.472973
2	C	20	0.270270
3	F	3	0.040541

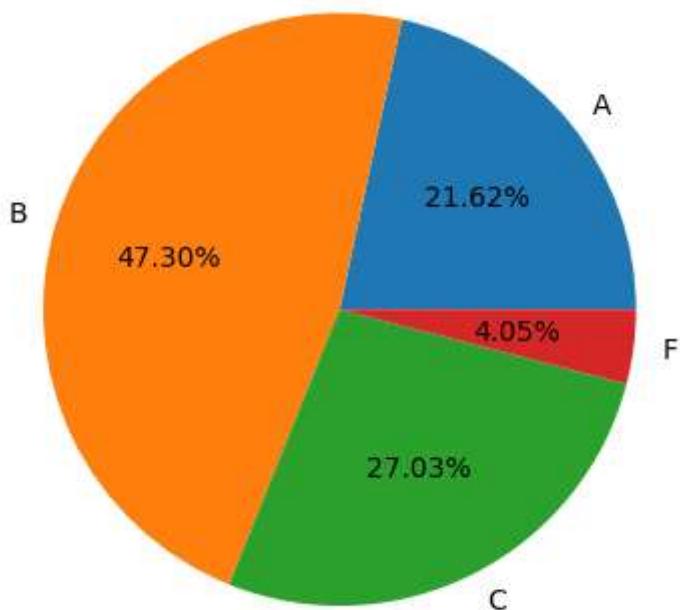
```
In [61]: ##### 8. 'Grade_group'별 파이 그래프 그리기
##### >'Grade_group'별 학생수 'Cnt_men' 구하기
##### >'Grade_group'별 학생수 비율 'Rate_men' 구하기
```

```
Out[61]:   Grade_group  Cnt_men  Rate_men
```

	Grade_group	Cnt_men	Rate_men
0	A	16	0.216216
1	B	35	0.472973
2	C	20	0.270270
3	F	3	0.040541

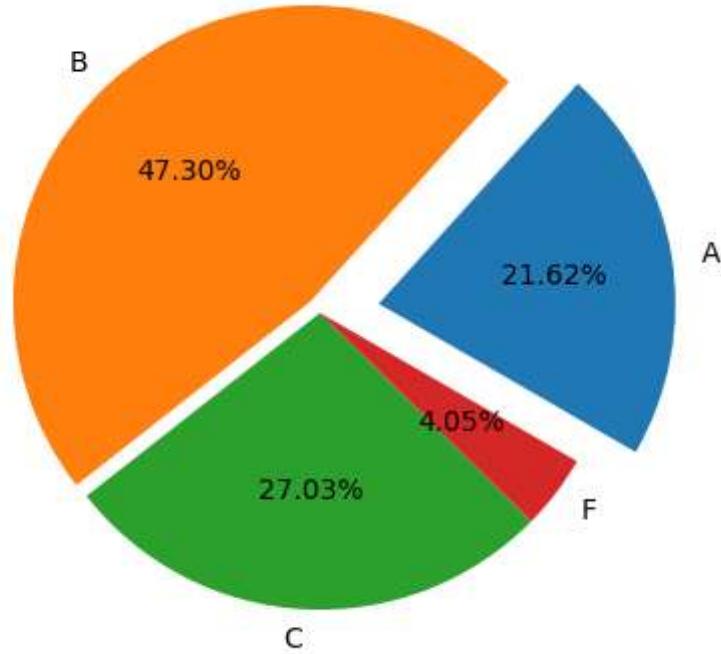
```
In [62]: ##### 8.1 'Grade_group'별 파이 그래프 그리기
##### > 'Grade_group'별 학생 수(Cnt_men)에 대한 파이그래프 그리기
```

```
Out[62]: ([<matplotlib.patches.Wedge at 0x14a055e4c40>,
... <matplotlib.patches.Wedge at 0x14a055e4b50>,
... <matplotlib.patches.Wedge at 0x14a055f3970>,
... <matplotlib.patches.Wedge at 0x14a05601040>],
[Text(0.8558393183840636, 0.6910420111020037, 'A'),
Text(-1.0517834049528894, 0.32210505905015907, 'B'),
Text(0.4952240298229679, -0.9822184890776087, 'C'),
Text(1.0910904599580111, -0.13971974874231627, 'F')],
[Text(0.4668214463913073, 0.3769320060556383, '21.62%'),
Text(-0.5737000390652124, 0.17569366857281402, '47.30%'),
Text(0.27012219808525517, -0.5357555394968774, '27.03%'),
Text(0.5951402508861878, -0.07621077204126342, '4.05%')])
```



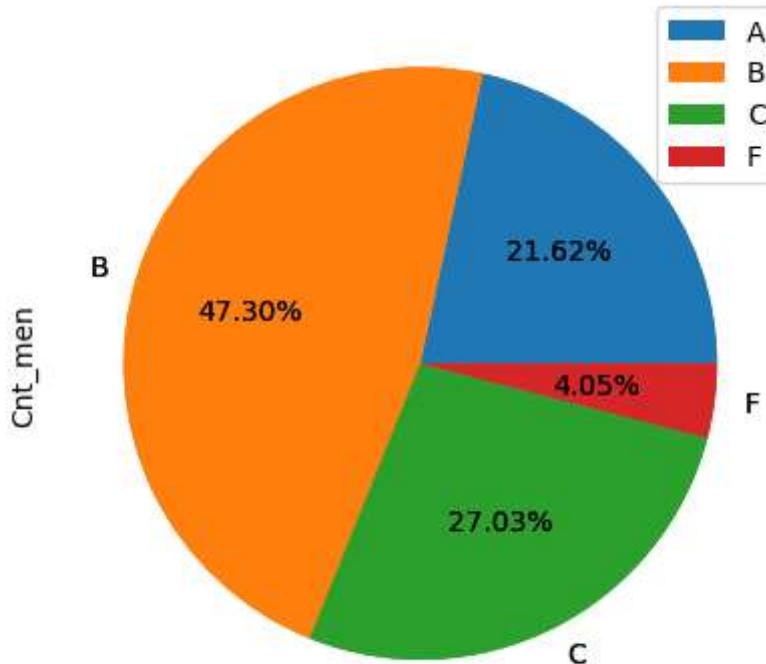
```
In [63]: ##### 8.2 'Grade_group'별 파이 그래프 그리기
##### > 'Grade_group'별 학생 수(Cnt_men)에 대한 파이그래프 그리기
```

```
Out[63]: ([<matplotlib.patches.Wedge at 0x14a056448b0>,
<matplotlib.patches.Wedge at 0x14a056216a0>,
<matplotlib.patches.Wedge at 0x14a056565e0>,
<matplotlib.patches.Wedge at 0x14a05656c70>],
[Text(1.2842813417071883, 0.20154760068724467, 'A'),
Text(-0.7839012829177014, 0.8414266329514309, 'B'),
Text(-0.062232654147610984, -1.0982381785194613, 'C'),
Text(0.8750521817793278, -0.6665460818002298, 'F')],
[Text(0.7903269795121157, 0.12402929273061208, '21.62%'),
Text(-0.44307463817087467, 0.4755889664508087, '47.30%'),
Text(-0.03394508408051508, -0.5990390064651606, '27.03%'),
Text(0.47730119006145144, -0.3635705900728526, '4.05%')])
```



```
In [64]: ##### 8.3 'Grade_group'별 학생 수(Cnt_men)에 대한 파이 그래프 그리기
```

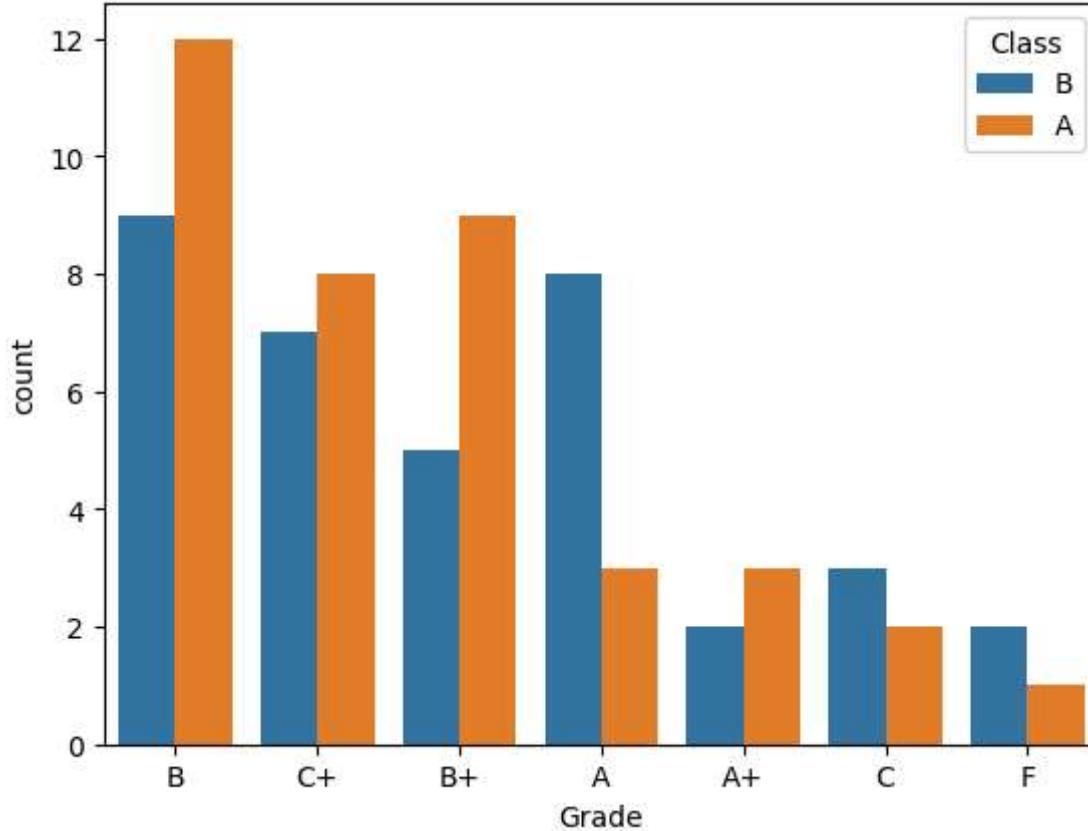
```
Out[64]: <Axes: ylabel='Cnt_men'>
```



```
In [65]: ##### 9. 'Class'별 'Grade'의 빈도 수 비교하는 막대 그래프 그리기
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\_\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
.. if pd.api.types.is_categorical_dtype(vector):
```

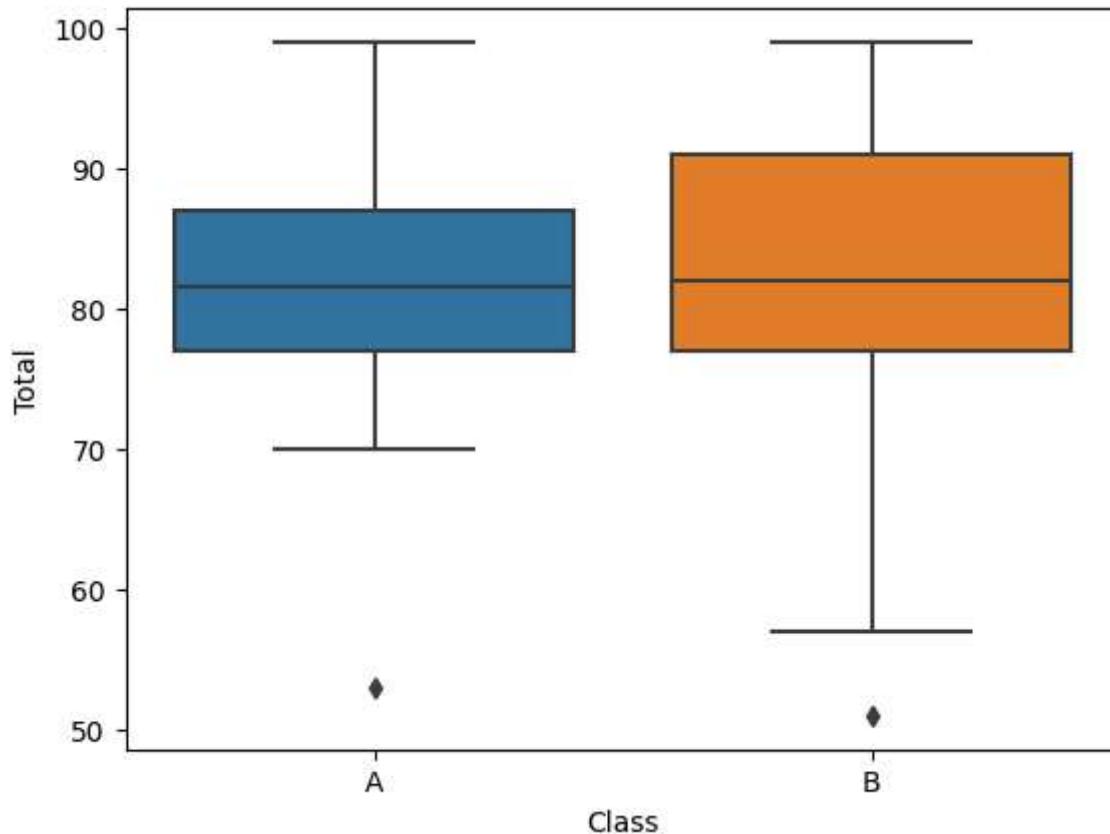
```
Out[65]: <Axes: xlabel='Grade', ylabel='count'>
```



```
In [66]: ##### 10. 'Class' 간의 총점 'Total'에 분포 비교 상자 그래프 그리기
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
C:\Users\ADMIN\anaconda3\lib\site-packages\seaborn\oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
<Axes: xlabel='Class', ylabel='Total'>
```

```
Out[66]:
```



In []:

정리하기

```
In [ ]: ### 1. 산점도  
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy')  
  
# 축 제한  
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy')  
    .set(xlim = [3, 6], ylim = [10, 30])  
  
# 종류별로 표식 색깔 바꾸기  
sns.scatterplot(data = mpg, x = 'displ', y = 'hwy', hue = 'drv')
```

```
### 2. 막대 그래프
```

```
## 평균 막대 그래프
```

```
# 1단계. 평균표 만들기
```

```
df_mpg = mpg.groupby('drv', as_index = False) #&  
    .agg(mean_hwy = ('hwy', 'mean'))
```

```
# 2단계. 그래프 만들기
```

```
sns.barplot(data = df_mpg, x = 'drv', y = 'mean_hwy')
```

```
## 빈도 막대 그래프
```

```
sns.countplot(data = mpg, x = 'drv')
```

```
### 3. 선 그래프
```

```
sns.lineplot(data = economics, x = 'date', y = 'unemploy')
```