



# [LAB-06] 10. 데이터 분포, 집계 예제 (Wage Dataset)



## #01. 준비작업



### 1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



### 2. 그래프 초기화

```
my_dpi = 200
fpath = "./NotoSansKR-Regular.ttf"
fm.fontManager.addfont(fpath)
fprop = fm.FontProperties(fname=fpath)
fname = fprop.get_name()
    출
plt.rcParams['font.family'] = fname      # 이미지 선명도(100~300)
plt.rcParams['font.size']   = 6            # 한글을 지원하는 폰트 파일의 경로
                                        # 폰트의 글꼴을 시스템에 등록함
                                        # 폰트의 속성을 읽어옴
                                        # 읽어온 속성에서 폰트의 이름만 추
                                        # 그라프에 한글 폰트 적용
plt.rcParams['axes.unicode_minus'] = False # 기본 폰트 크기
                                            # 그라프에 마이너스 깨짐 방지
```



### [3] 데이터 가져오기

```
origin = load_data('wage')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/wage.xlsx
[94m[desc] [0m Wage 데이터 셋은 경제 및 노동 시장에 관련된 정보를 담고 있는 데이터셋
(출처: ADsP 기출문제)
```

field	description
-----	-----
year	년도

age	나이
maritl	결혼여부
race	근로자의 인종
education	교육수준
region	지역
jobclass	직군
health	건강상태
health_ins	건강보험 가입 여부
logwage	임금(로그값)
wage	임금

	year	age	maritl	race	education	region	jobclass	health	health_ins
0	2006	18	1. Never Married	1. White	1. < HS Grad	1. Middle Atlantic	1. Industrial	1. <=Good	1. N
1	2004	24	1. Never Married	1. White	1. College Grad	1. Middle Atlantic	1. Information	1. >=Very Good	1. N
2	2003	45	1. Married	1. White	1. Some College	1. Middle Atlantic	1. Industrial	1. <=Good	1. Y
3	2003	43	1. Married	1. Asian	1. College Grad	1. Middle Atlantic	1. Information	1. >=Very Good	1. Y
4	2005	50	1. Divorced	1. White	1. HS Grad	1. Middle Atlantic	1. Information	1. <=Good	1. Y
...	...	...	...	...	...	...	...	...	...
2995	2008	44	1. Married	1. White	1. Some College	1. Middle Atlantic	1. Industrial	1. >=Very Good	1. Y
2996	2007	30	1. Married	1. White	1. HS Grad	1. Middle Atlantic	1. Industrial	1. >=Very Good	1. N
2997	2005	27	1. Married	1. Black	1. < HS Grad	1. Middle Atlantic	1. Industrial	1. <=Good	1. N
2998	2005	27	1. Never Married	1. White	1. Some College	1. Middle Atlantic	1. Industrial	1. >=Very Good	1. Y
2999	2009	55	1. Separated	1. White	1. HS Grad	1. Middle Atlantic	1. Industrial	1. <=Good	1. Y

3000 rows × 11 columns



## #02. 데이터 확인

데이터 전처리 전에 데이터의 크기, 타입 등을 확인한 후 필요하다면 적절한 타입 변환이 필요함



### 1. 데이터 타입 확인

```
origin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   year        3000 non-null    int64  
 1   age          3000 non-null    int64  
 2   maritl       3000 non-null    object  
 3   race         3000 non-null    object  
 4   education    3000 non-null    object  
 5   region       3000 non-null    object  
 6   jobclass     3000 non-null    object  
 7   health       3000 non-null    object  
 8   health_ins  3000 non-null    object  
 9   logwage      3000 non-null    float64 
 10  wage         3000 non-null    float64 
dtypes: float64(2), int64(2), object(7)
memory usage: 257.9+ KB
```



### 2. 결측치 유무 확인

```
origin.isna().sum()
```

```
year      0
age       0
maritl    0
race      0
education 0
region    0
jobclass  0
health    0
health_ins 0
logwage   0
```

```
wage          0  
dtype: int64
```

## #03. 데이터 전처리

### 1. 명목형 타입 변환

```
df1 = origin.astype({'year': 'category', 'maritl': 'category',  
                    'race': 'category', 'education': 'category', 'region':  
                    'category', 'jobclass': 'category', 'health': 'category',  
                    'health_ins': 'category'})  
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3000 entries, 0 to 2999  
Data columns (total 11 columns):  
 #   Column      Non-Null Count  Dtype     
 ---  --          --          --  
 0   year        3000 non-null   category  
 1   age          3000 non-null   int64  
 2   maritl       3000 non-null   category  
 3   race         3000 non-null   category  
 4   education    3000 non-null   category  
 5   region       3000 non-null   category  
 6   jobclass     3000 non-null   category  
 7   health        3000 non-null   category  
 8   health_ins   3000 non-null   category  
 9   logwage      3000 non-null   float64  
 10  wage          3000 non-null   float64  
dtypes: category(8), float64(2), int64(1)  
memory usage: 95.3 KB
```

## #04. 수치형 변수에 대한 데이터 분포 확인

### 1. 기술통계

전처리 과정에서 명목형으로 변경된 변수는 요약통계량에서 자동으로 제외된다.

```
df1.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	3000.0	42.414667	11.542406	18.000000	33.750000	42.000000	51.000000	80.000000
logwage	3000.0	4.653905	0.351753	3.000000	4.447158	4.653213	4.857332	5.763128
wage	3000.0	111.703608	41.728595	20.085537	85.383940	104.921507	128.680488	318.342430

## 기술통계를 통해 확인되는 객관적 사실

### 1. 공통 특성

- age, logwage, wage 세 변수 모두 관측치 수는 동일하게 3000개이다.
- 세 변수 모두 최소값과 최대값 사이의 범위가 넓어 분산이 존재한다.

### 2. age 변수

- 평균은 42.41세, 중앙값은 42세이다.
- 25% 지점: 33.75세, 75% 지점: 51세이다.
- 최소값은 18세, 최대값은 80세이다.
- 평균(42.41)과 중앙값(42)이 거의 동일하여 분포가 비교적 대칭적임을 알 수 있다.

### 3. logwage 변수

- 평균은 4.6539, 중앙값은 4.6532이다.
- 25% 지점: 약 4.447, 75% 지점: 약 4.857이다.
- 최소값은 3, 최대값은 5.7631이다.
- 평균과 중앙값이 유사해 대칭성에 가까운 분포 형태임을 확인할 수 있다.

### 4. wage 변수

- 평균 임금은 약 111.70, 중앙값은 약 104.92이다.
- 25% 지점: 약 85.38, 75% 지점: 약 128.68이다.
- 최소값은 20.09, 최대값은 318.34이다.
- 평균이 중앙값보다 크기 때문에 오른쪽 꼬리가 존재하는 분포임을 알 수 있다.

### 5. 세 변수 간 비교

- age는 18~80세 사이로 성인 노동시장 전반을 포괄한다.
- wage는 평균이 중앙값보다 더 커 로그 변환 전보다 치우침이 더 심함을 보여준다.
- 표준편차는 wage(41.73) > age(11.54) > logwage(0.35) 순으로 크다.



## 2. 상자그림

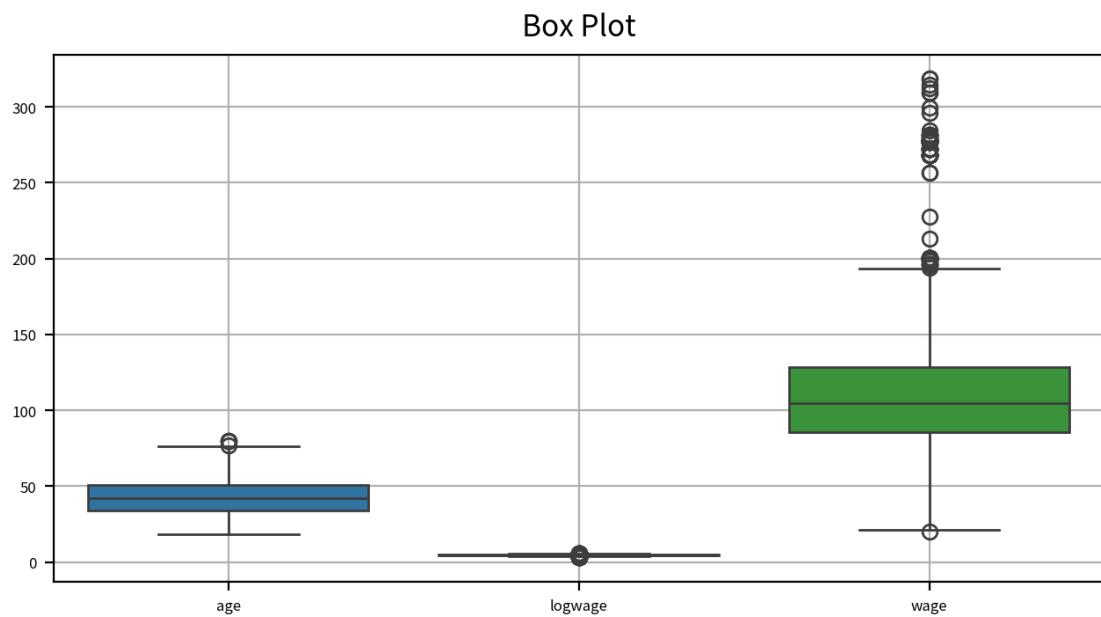
상자그림을 위한 `boxplot` 함수의 `data` 파라미터에 데이터프레임을 통째로 지정할 경우 자동으로 명목형 변수는 제외하고 시각화 한다.

```
# 1) 그래프 초기화
width_px  = 1280                         # 그래프 가로 크기
height_px = 720                            # 그래프 세로 크기
rows      = 1                               # 그래프 행 수
cols      = 1                               # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax  = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) BoxPlot 그리기
sb.boxplot(data=df1)

# 3) 그래프 꾸미기
ax.set_title("Box Plot", fontsize=12, pad=8)
ax.grid(True)                             # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                        # 여백 제거
plt.show()                                # 그래프 화면 출력
plt.close()                               # 그래프 작업 종료
```



png

데이터 구간 차이가 심할 경우 하나의 그래프에 모든 컬럼을 시각화 하는 것은 해석하기 어려운 결과물을 생성하게 된다.

| 각각 개별적으로 그리는 것이 좋다.



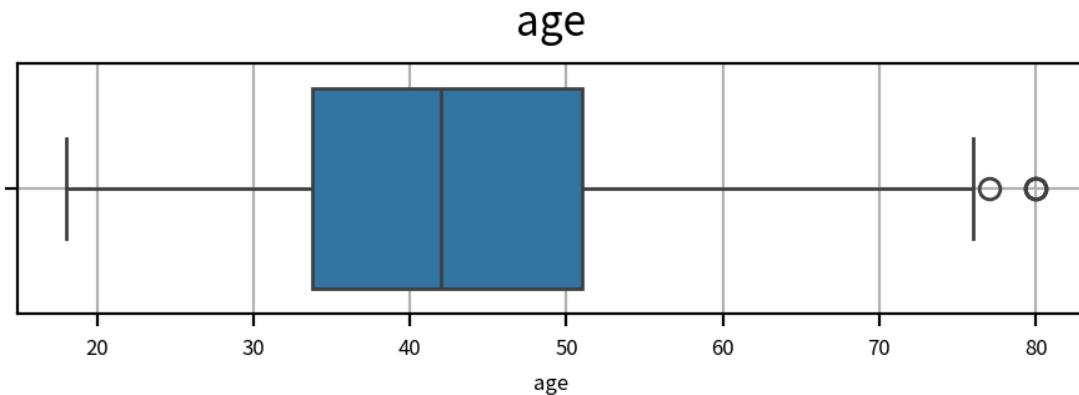
### 3. 상자그림 개별 표시

```
# 1) 그래프 초기화
width_px  = 900                      # 그래프 가로 크기
height_px = 350                       # 그래프 세로 크기
rows      = 1                          # 그래프 행 수
cols      = 1                          # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) BoxPlot 그리기
sb.boxplot(data=df1, x='age')

# 3) 그래프 꾸미기
ax.set_title("age", fontsize=12, pad=8)
ax.grid(True)                         # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                   # 여백 제거
plt.show()                           # 그래프 화면 출력
plt.close()                          # 그래프 작업 종료
```



png

```
# 1) 그래프 초기화
width_px  = 900                      # 그래프 가로 크기
height_px = 350                       # 그래프 세로 크기
rows      = 1                          # 그래프 행 수
cols      = 1                          # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

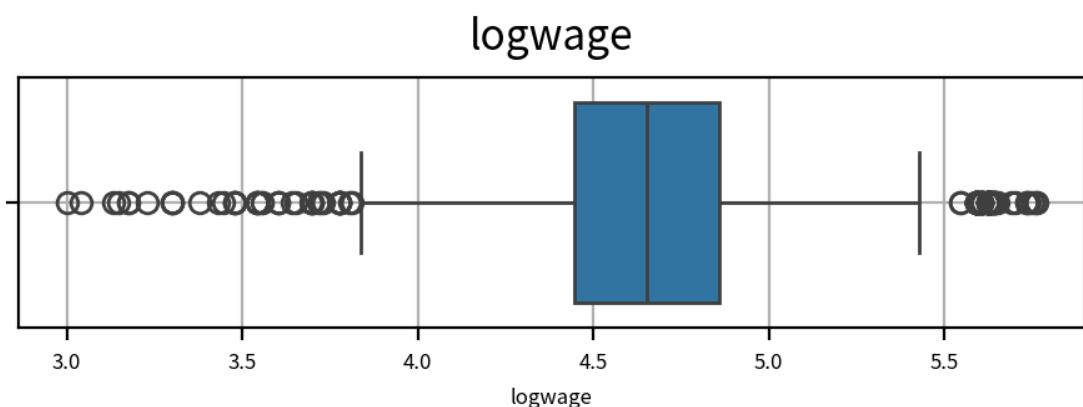
```

# 2) BoxPlot 그리기
sb.boxplot(data=df1, x='logwage')

# 3) 그래프 꾸미기
ax.set_title("logwage", fontsize=12, pad=8)
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png

```

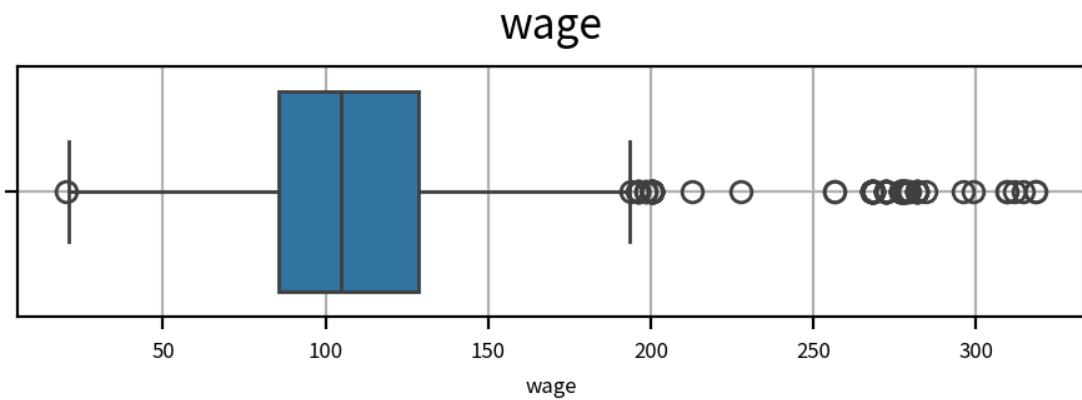
# 1) 그래프 초기화
width_px = 900 # 그래프 가로 크기
height_px = 350 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) BoxPlot 그리기
sb.boxplot(data=df1, x='wage')

# 3) 그래프 꾸미기
ax.set_title("wage", fontsize=12, pad=8)
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png

## 반복문 활용하기

동일한 설정에서 컬럼 이름만 변경하는 경우 컬럼이름을 리스트로 담아두고 반복문으로 처리할 수 있다.

```

fields = ['age', 'logwage', 'wage']

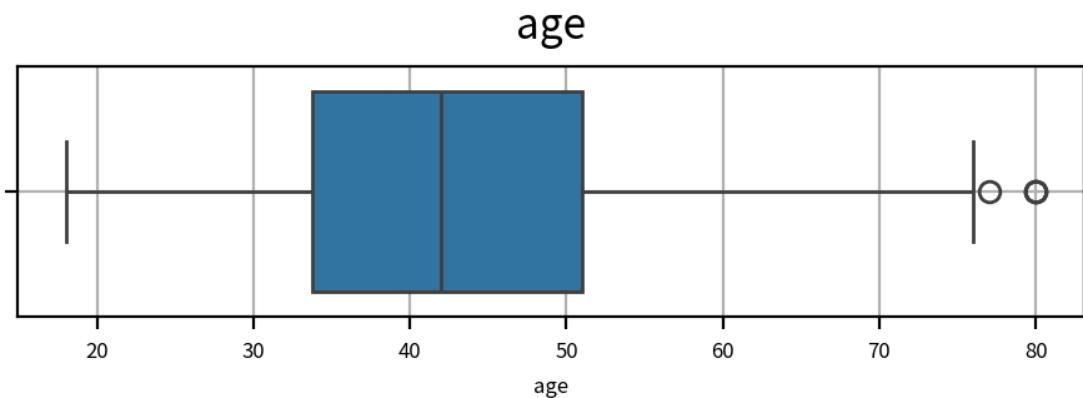
for f in fields:
    # 1) 그래프 초기화
    width_px  = 900                         # 그래프 가로 크기
    height_px = 350                          # 그래프 세로 크기
    rows = 1                                  # 그래프 행 수
    cols = 1                                  # 그래프 열 수
    figsize = (width_px / my_dpi, height_px / my_dpi)
    fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

    # 2) BoxPlot 그리기
    sb.boxplot(data=df1, x=f)

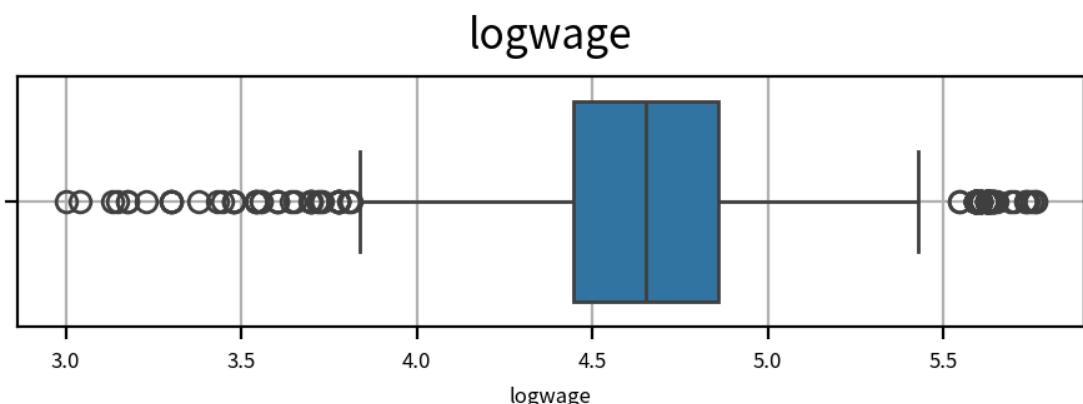
    # 3) 그래프 꾸미기
    ax.set_title(f, fontsize=12, pad=8)
    ax.grid(True)                            # 배경 격자 표시/숨김

    # 4) 출력
    plt.tight_layout()                      # 여백 제거
    plt.show()                             # 그래프 화면 출력
    plt.close()                            # 그래프 작업 종료

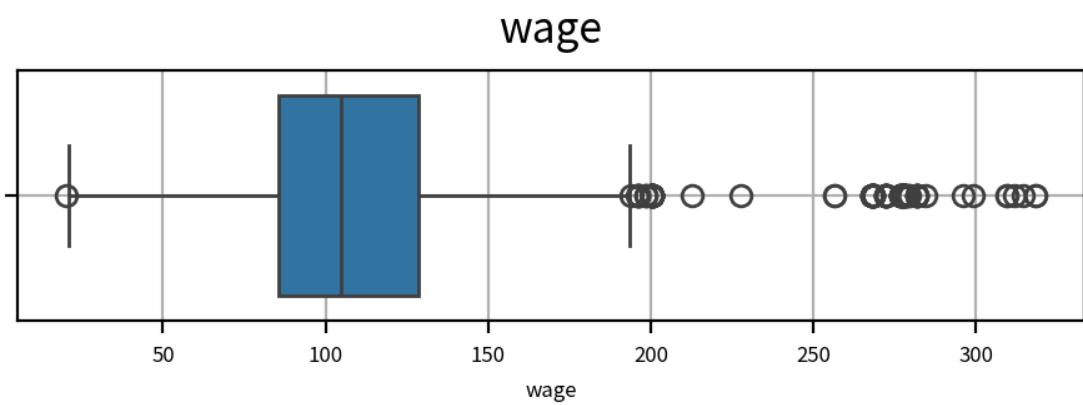
```



png



png



png

### 알 수 있는 사실 [요약]

- 조사 인원은 3000명이다.
- 조사 인원의 연령은 18세~80세 까지 이고, 평균 연령은 42.4세이다.
- 조사 인원의 임금은 20~318이고, 평균 임금은 111.7, 로그 변환 값은 4.65이다.
- 연령에 대한 표준 편자는 11.50이고, 임금의 표준 편자는 41.730이다.

- 임금에 대한 표준편차가 크다.



## #05. 명목형 변수의 데이터 분포 확인

명목형의 분포는 집단별 데이터 수를 집계하는 것으로 표현된다.

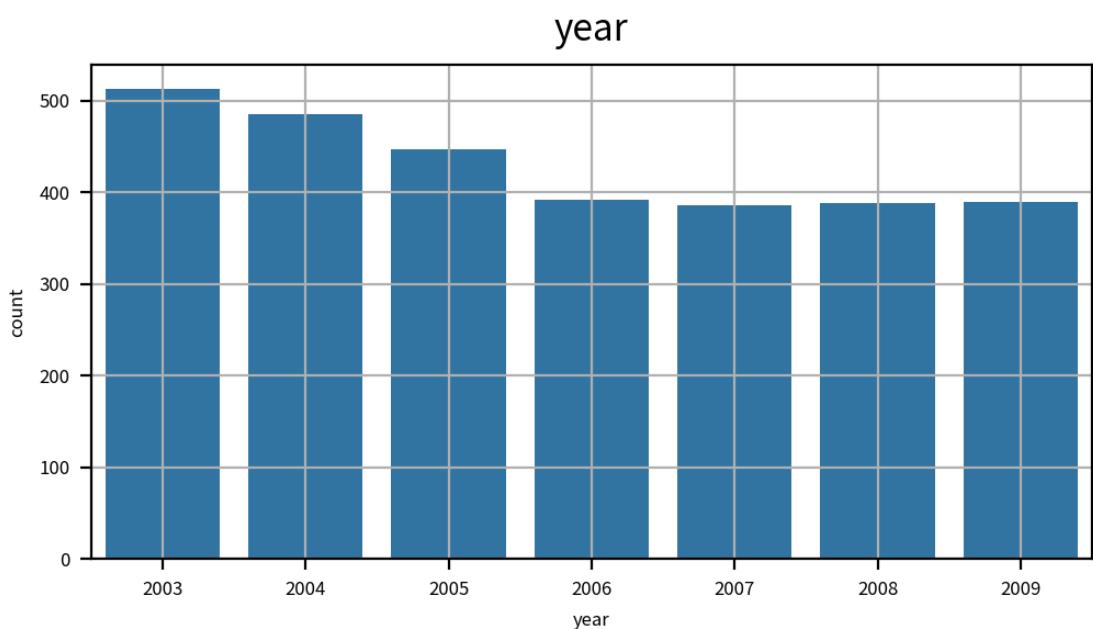
```
fields = ['year', 'maritl', 'race', 'education', 'region',
          'jobclass', 'health', 'health_ins']

for f in fields:
    # 1) 그래프 초기화
    width_px  = 1024                         # 그래프 가로 크기
    height_px = 600                           # 그래프 세로 크기
    rows = 1                                    # 그래프 행 수
    cols = 1                                    # 그래프 열 수
    figsize = (width_px / my_dpi, height_px / my_dpi)
    fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

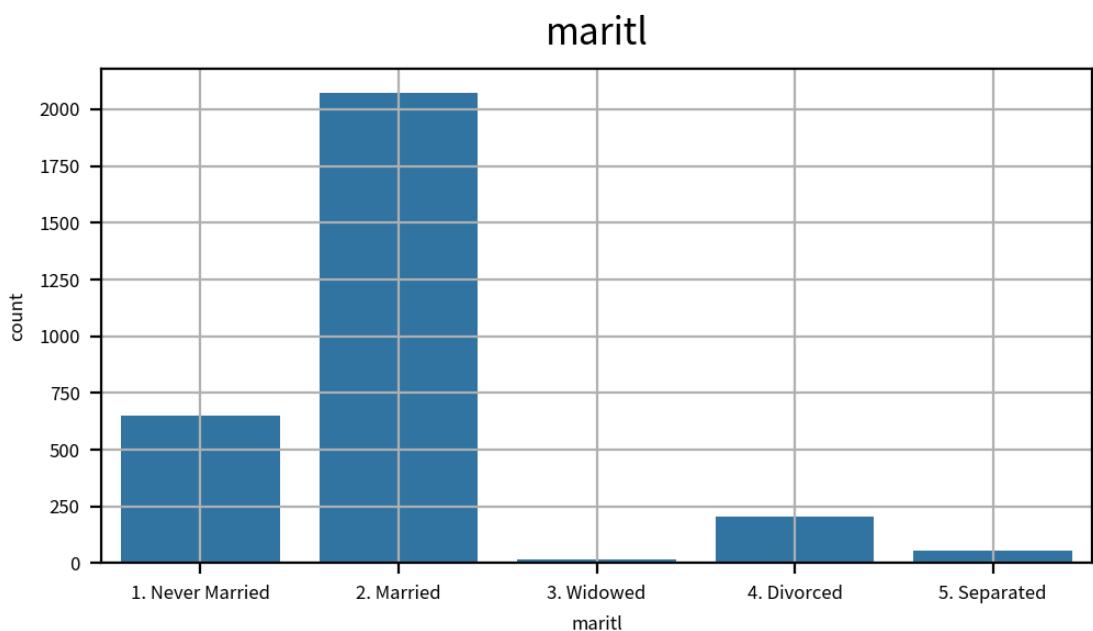
    # 2) CountPlot 그리기
    sb.countplot(data=df1, x=f)

    # 3) 그래프 꾸미기
    ax.set_title(f, fontsize=12, pad=8)
    ax.grid(True)                            # 배경 격자 표시/숨김

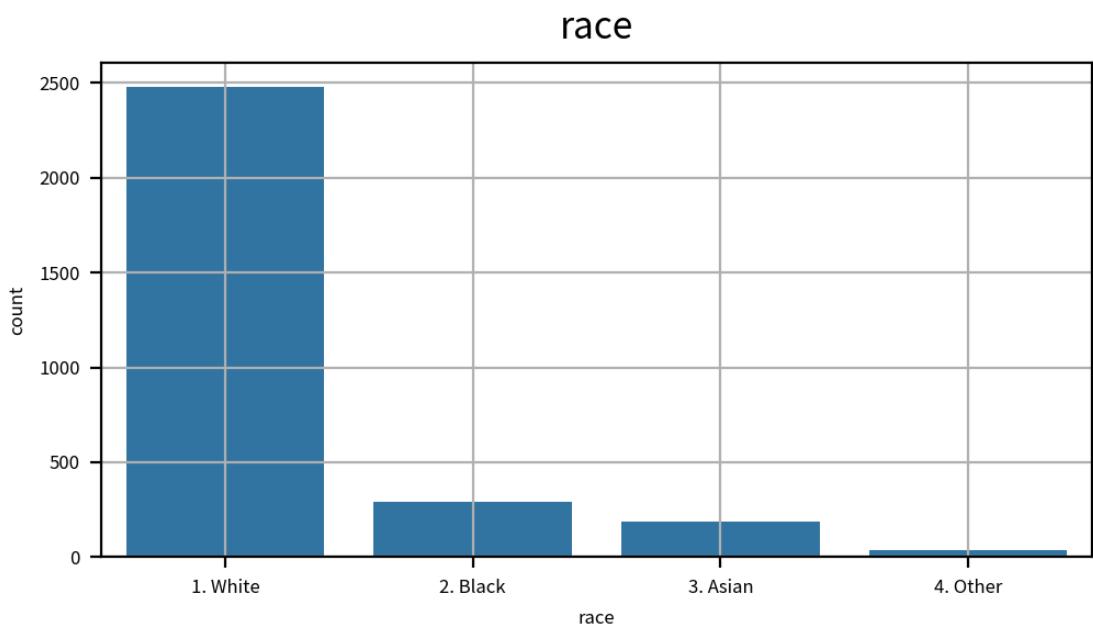
    # 4) 출력
    plt.tight_layout()                      # 여백 제거
    plt.show()                             # 그래프 화면 출력
    plt.close()                            # 그래프 작업 종료
```



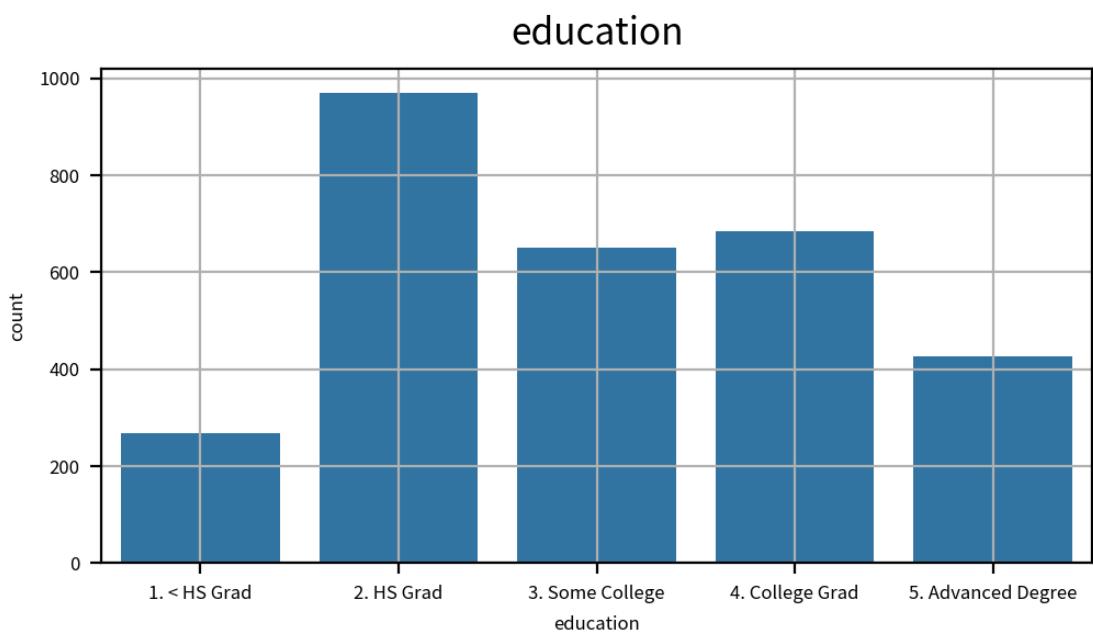
png



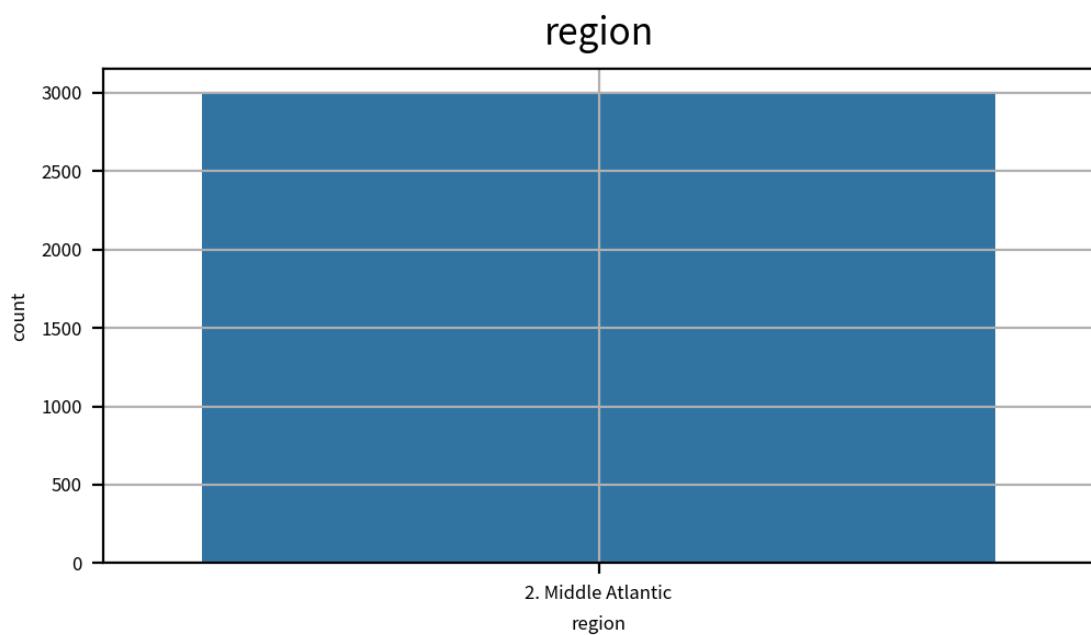
png



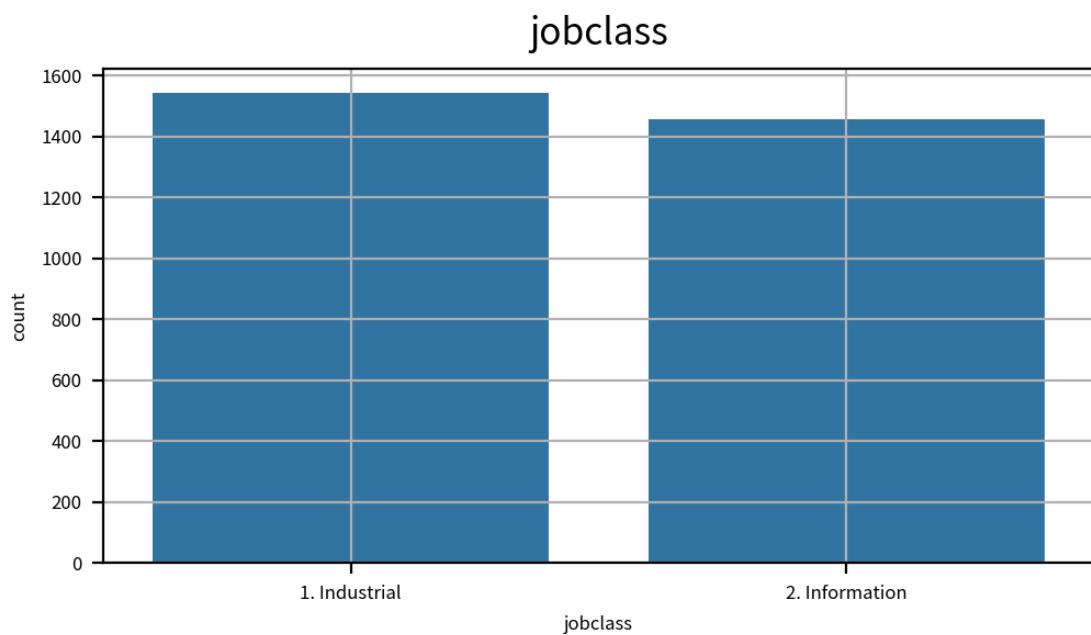
png



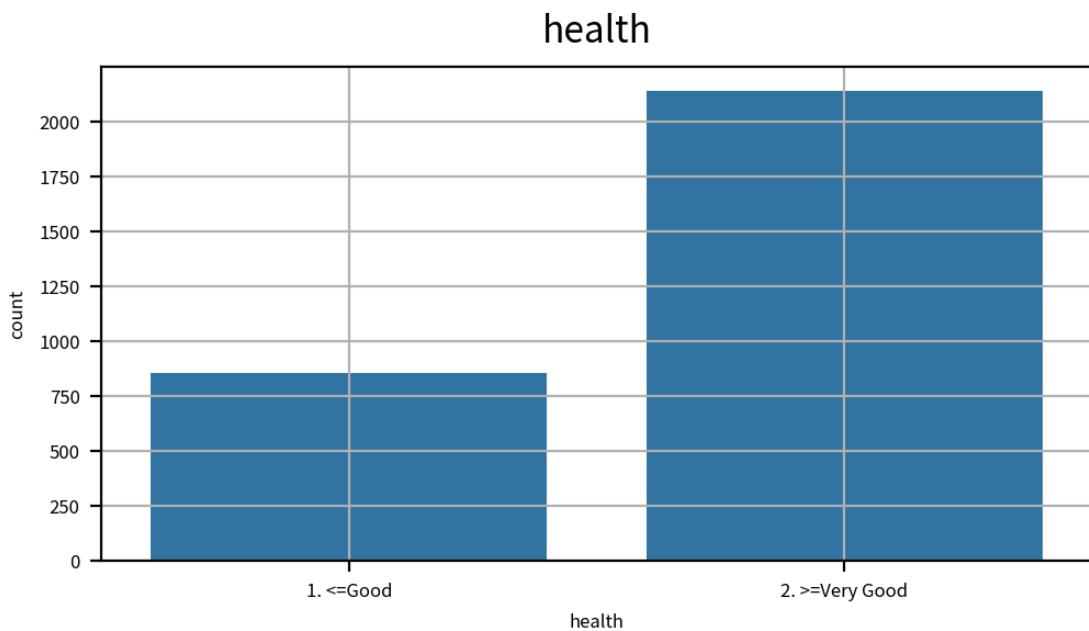
png



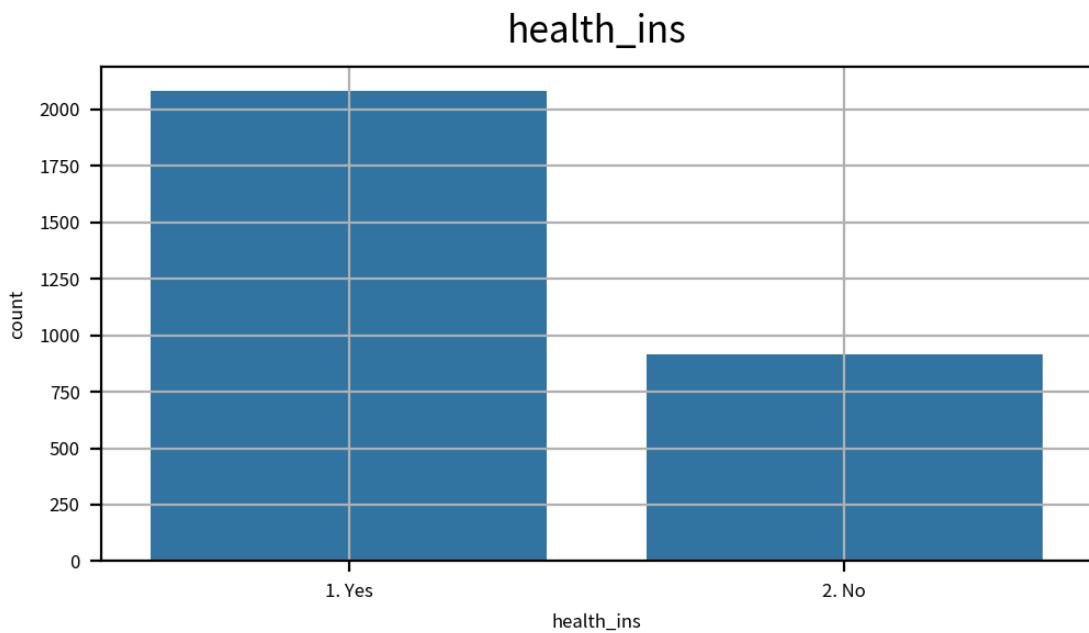
png



png



png



png



## 알 수 있는 사실

1. 기혼자가 그렇지 않은 사람보다 많다.
2. 백인이 다른 인종보다 더 많다.
3. 고등학교 졸업(HS Grad)가 가장 많고 그 다음으로 많은 경우는 대학교 졸업(College Grad), 대학교 중퇴(Some Colleage), 대학원 졸업(Advanced Degree) 순이다. 일부 고등학교 미만의 학력도 보인다.
4. 조사 지역은 중앙 대서양 연안으로만 한정되어 있다.

5. 직군은 생산직과 사무직이 비슷한 비율로 분포되어 있지만 생산직이 약간 더 많은 비율을 차지한다.
6. 건강상태는 대부분의 조사 대상이 좋은 상태를 보이고 있다.
7. 건강 보험은 대부분의 조사 대상이 가입되어 있다.



## #06. 결혼 여부에 따른 임금 수준 비교

결혼 여부에 따라 임금 수준을 히스토그램으로 비교하기



### 값의 종류 확인

```
married = sorted(list(df1['marital'].unique()))
married
```

```
['1. Never Married', '2. Married', '3. Widowed', '4. Divorced', '5.
Separated']
```

```
bins_count = 13 # 13개 구간으로 나누기
```

```
for m in married:
    mdf = df1.query("marital == @m")

    # 1) 그래프 초기화
    width_px = 1280                      # 그래프 가로 크기
    height_px = 480                        # 그래프 세로 크기
    rows = 1                               # 그래프 행수
    cols = 1                               # 그래프 열수
    figsize = (width_px / my_dpi, height_px / my_dpi)
    fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

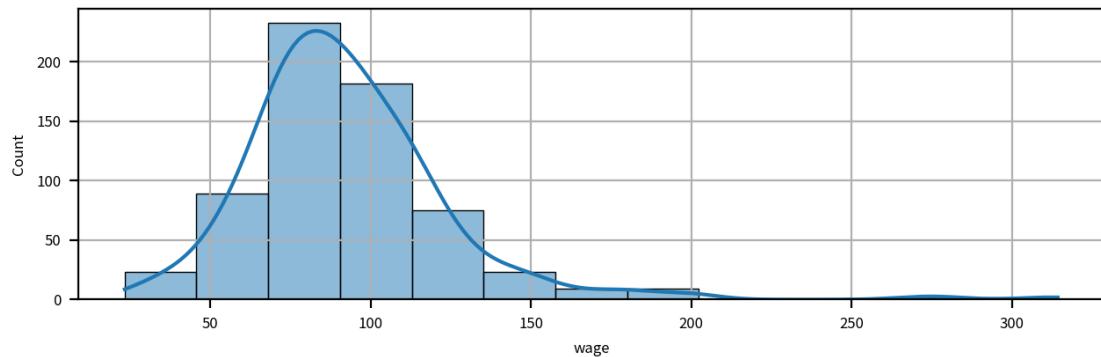
    # 2) KDE Plot 그리기
    sb.histplot(data=mdf, x="wage", bins=bins_count,
                edgecolor="#000000", linewidth=0.5, kde=True)

    # 3) 그래프 꾸미기
    ax.set_title("marital=%s" % m, fontsize=12, pad=8)
    ax.grid(True)                          # 배경 격자 표시/숨김

    # 4) 출력
    plt.tight_layout()                   # 여백 제거
```

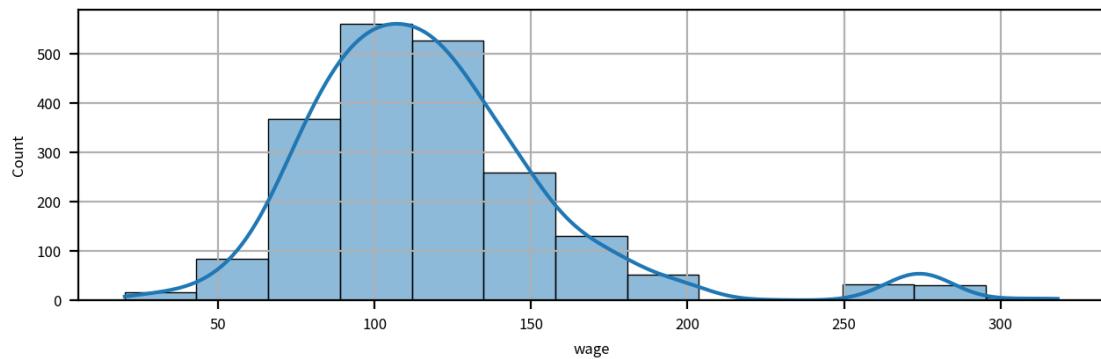
```
plt.show()          # 그래프 화면 출력  
plt.close()        # 그래프 작업 종료
```

maritl=1. Never Married



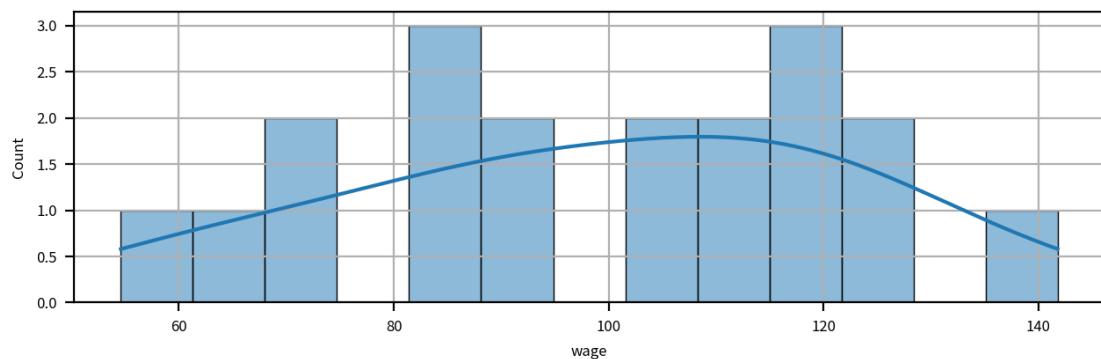
png

maritl=2. Married

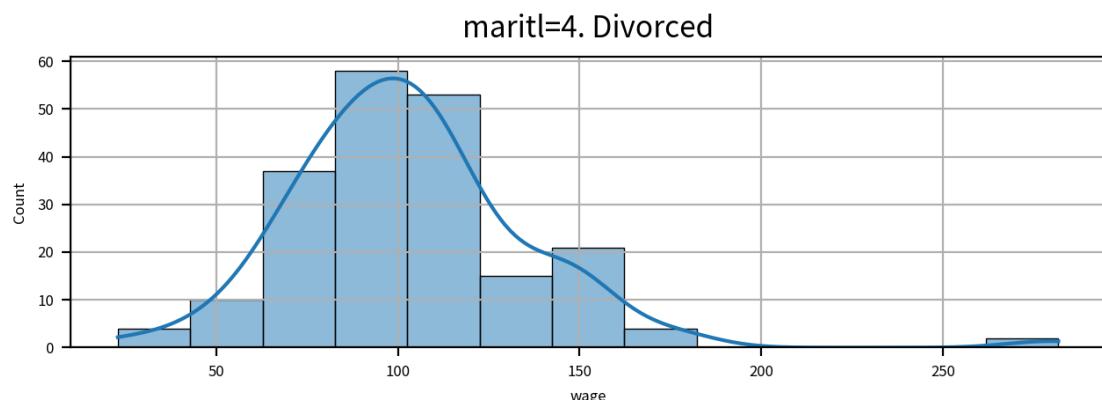


png

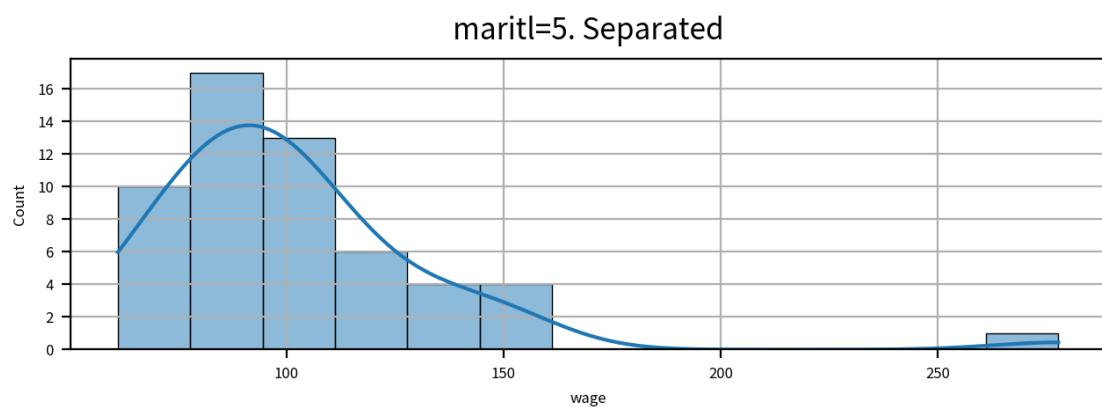
maritl=3. Widowed



png



png



png



## 알 수 있는 사실

### 1. 1. Never Married / 2. Married

- 미혼인 경우보다는 기혼인 경우의 소득 수준이 더 높으며 미혼인 고액소득자보다 기혼인 고액 소득자가 더 많다.

### 2. 3. Widowed

- 미망인의 경우도 미혼인 경우보다는 소득 수준이 높은 것으로 나타났다.

### 3. 4. Divorced / 5. Separated

- 이혼 혹은 별거의 경우 미혼인 경우보다도 소득수준이 낮은 것으로 나타났다.



## #07. 교육 수준에 따른 임금 수준 비교

### 6번 응용

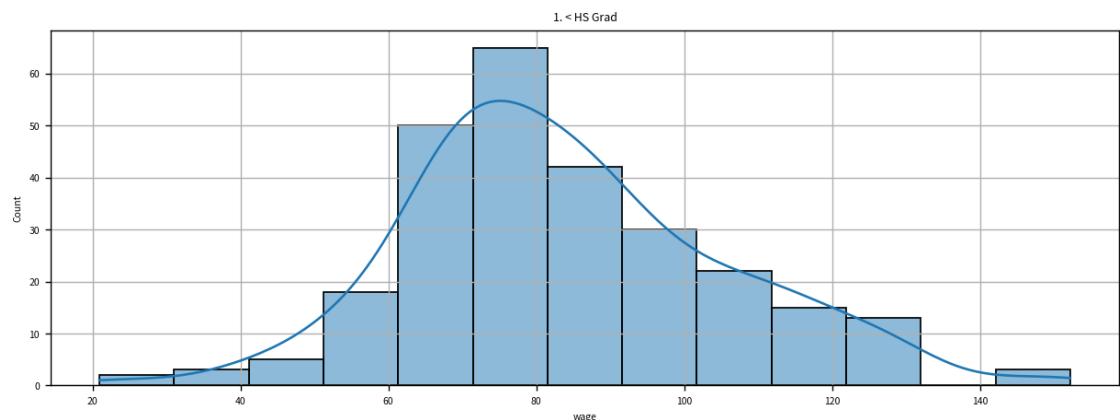
```
education = sorted(list(df1['education'].unique()))
education
```

```
[ '1. < HS Grad',
  '2. HS Grad',
  '3. Some College',
  '4. College Grad',
  '5. Advanced Degree']
```

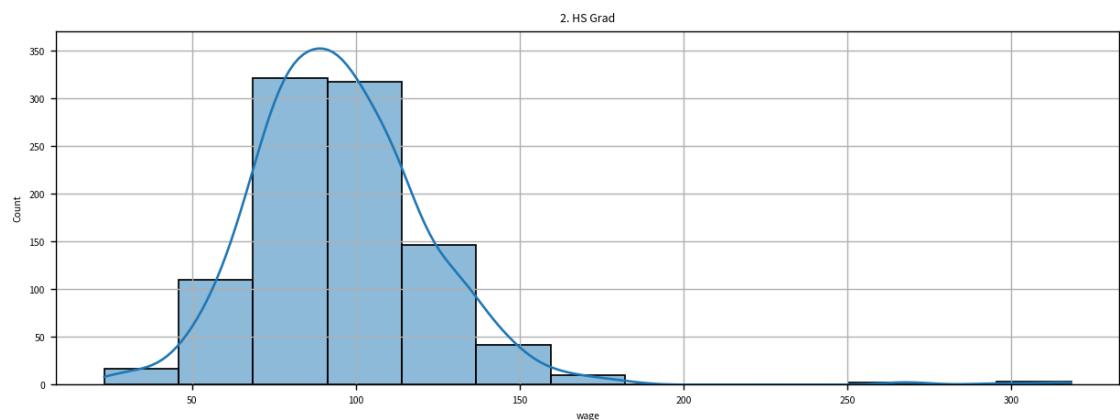
```
bins_count = 13 # 13개 구간으로 나누기

for e in education:
    mdf = df1.query("education == @e")

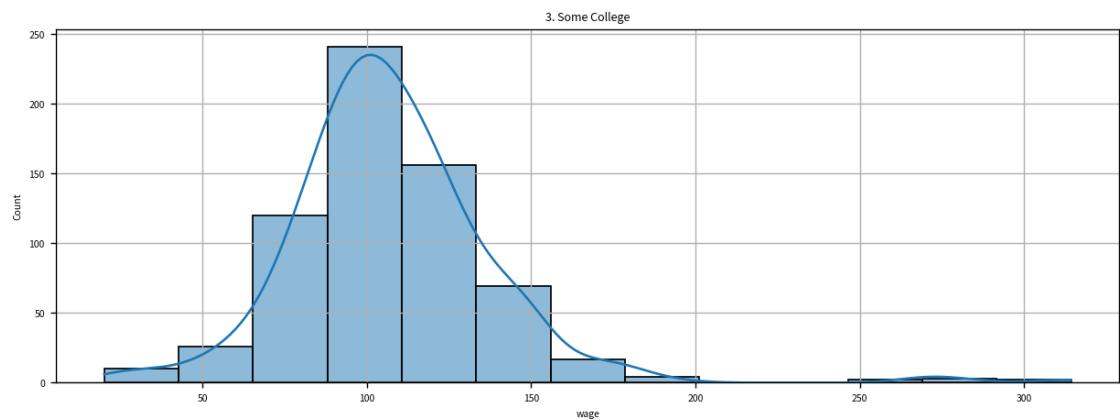
    plt.figure(figsize=(12, 4), dpi=125)
    sb.histplot(data=mdf, x='wage', bins=bins_count, kde=True)
    plt.title(e)
    plt.grid()
    plt.show()
    plt.close()
```



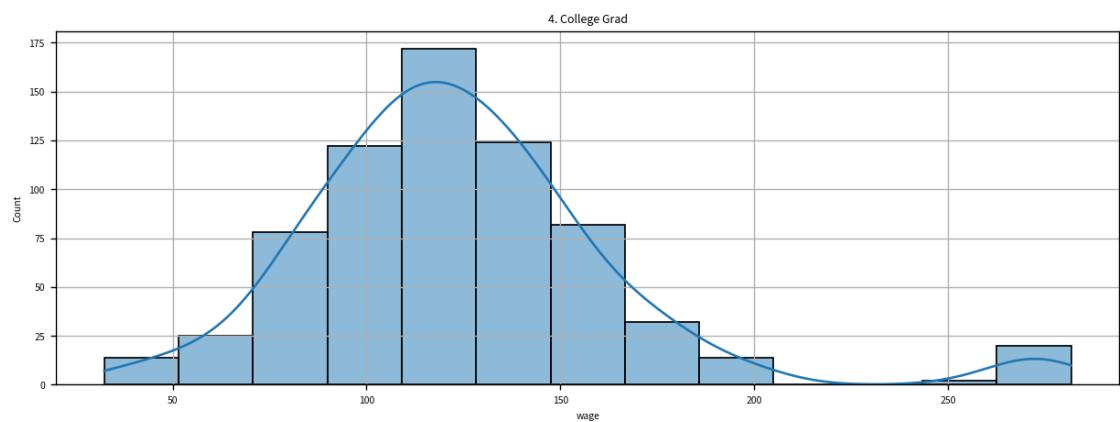
png



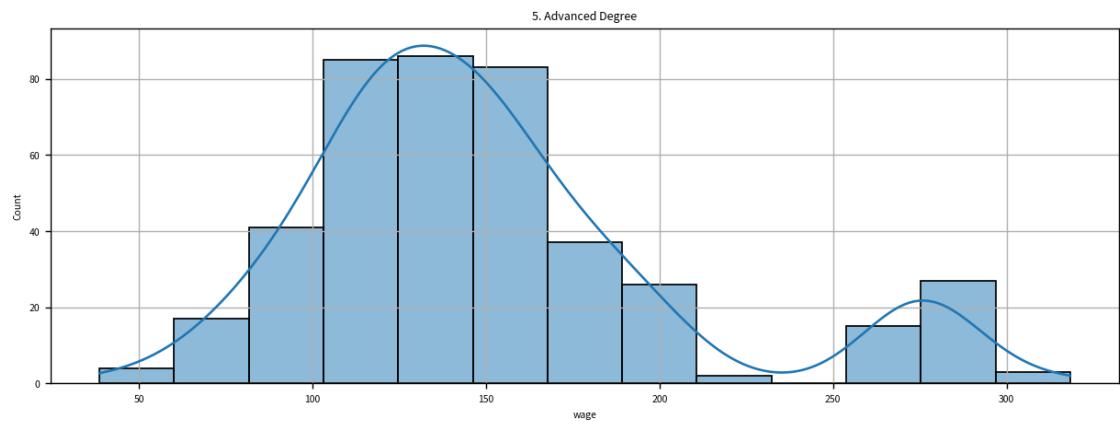
png



png



png



png



## 알 수 있는 사실

교육 수준이 높을 수록 소득수준도 높다.

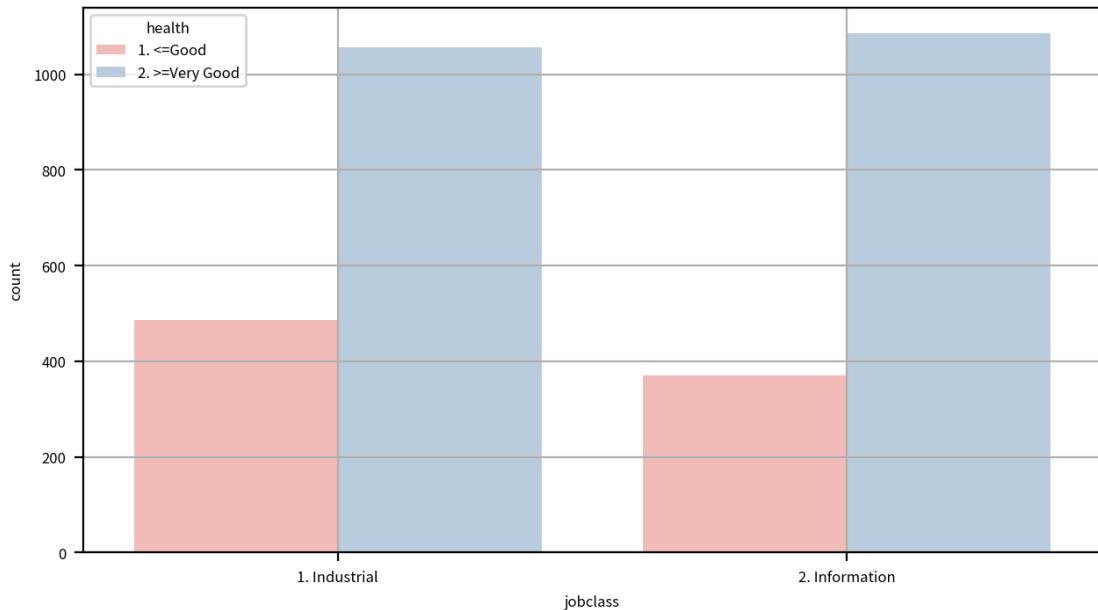
## #08. 직군별 건강상태 확인

```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax  = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) CountPlot 그리기
sb.countplot(data=df1, x="jobclass", hue="health", palette="Pastel1")

# 3) 그래프 꾸미기
ax.grid(True)                          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                     # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```



png

알 수 있는 사실

대부분 근로자의 건강상태가 좋은 편이지만 평균 이하의 건강상태를 보이는 근로자는 생산직군에 더 많이 분포되어 있다.



# 데이터 시각화 연습문제

---



## 공통 요구사항

---

모든 문제에는 시각화 결과에서 알아낼 수 있는 객관적 사실을 서술해야 합니다.



### 문제 1

교육 수준 별로 나이와 임금의 관계를 탐색하기 위한 시각화를 구현하고, 시각화 결과에서 알 수 있는 사실을 설명하세요.



### 문제 2

인종(race)별로 직업군(jobclass)에 따른 임금(wage) 분포를 비교할 수 있는 시각화를 구현하고, 시각화 결과에서 알 수 있는 사실을 설명하세요.



### 문제 3

연도(year)에 따른 임금(wage)의 변화 추이를 교육 수준(education)별로 나누어 시각화하고, 시각화 결과에서 알 수 있는 사실을 설명하세요.