# 단변량분석_숫자형변수


# 1.환경준비

## (1) 라이브러리 불러오기

In [44]:
```
import numpy as np
import pandas as pd

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

## (2) 데이터 로딩

### 1) titanic

- url : 'https://raw.githubusercontent.com/DA4BAM/dataset/master/titanic_simple.csv'

**[titanic_simple 데이터 셋 정보]**

- PassengerId : 승객번호
- Survived : 생존여부(1:생존, 0:사망)
- Pclass : 객실등급(1:1등급, 2:2등급, 3:3등급)
- Name : 승객이름
- Sex : 성별(male, female)
- Age : 나이
- Fare : 운임($)
- Embarked : 승선지역(Southampton, Cherbourg, Queenstown)

In [45]:
```
path = 'https://raw.githubusercontent.com/DA4BAM/dataset/master/titanic_simple.csv'
titanic = pd.read_csv(path)
```

```
titanic.head()
```

Out[45]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 7.2500 | Southampton |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 71.2833 | Cherbourg |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 7.9250 | Southampton |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 53.1000 | Southampton |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 8.0500 | Southampton |

## 2) New York Air Quality

- url : https://raw.githubusercontent.com/DA4BAM/dataset/master/air2.csv

**[airquality 데이터 셋 정보]**

- Ozone: 오존 농도
- Solar.R: 태양복사량
- Wind: 풍속
- Temp: 기온
- Date : 연,월,일

In [46]:
```
path = 'https://raw.githubusercontent.com/DA4BAM/dataset/master/air2.csv'
air = pd.read_csv(path)
air.head()
```

Out[46]:

| | Ozone | Solar.R | Wind | Temp | Date |
|---|---|---|---|---|---|
| **0** | 41 | 190.0 | 7.4 | 67 | 1973-05-01 |
| **1** | 36 | 118.0 | 8.0 | 72 | 1973-05-02 |
| **2** | 12 | 149.0 | 12.6 | 74 | 1973-05-03 |
| **3** | 18 | 313.0 | 11.5 | 62 | 1973-05-04 |
| **4** | 19 | NaN | 14.3 | 56 | 1973-05-05 |

# 2.숫자형 변수

## (1) 수치화 : 대푯값

### 1) 평균(산술평균)

In [47]:
```
# 넘파이 함수 이용하기- 넘파이 어레이로 변화시켜서 계산. 시리즈, 리스트 가능
np.mean(titanic['Fare'])
```

Out[47]:
```
32.204207968574636
```

In [48]:
```
# 판다스의 mean 메서드 이용하기- 시리즈에 딸려 있는 메소드
titanic['Fare'].mean()
```

Out[48]:
```
32.204207968574636
```

## 2) 중앙값(중위수, median)

자료의 순서상 가운데 위치한 값

In [49]:
```
# 넘파이 함수 이용하기
np.median(titanic['Fare'])
```

Out[49]:
```
14.4542
```

In [50]:
```
# 판다스의 median 메서드 이용하기
titanic['Fare'].median()
```

Out[50]:
```
14.4542
```

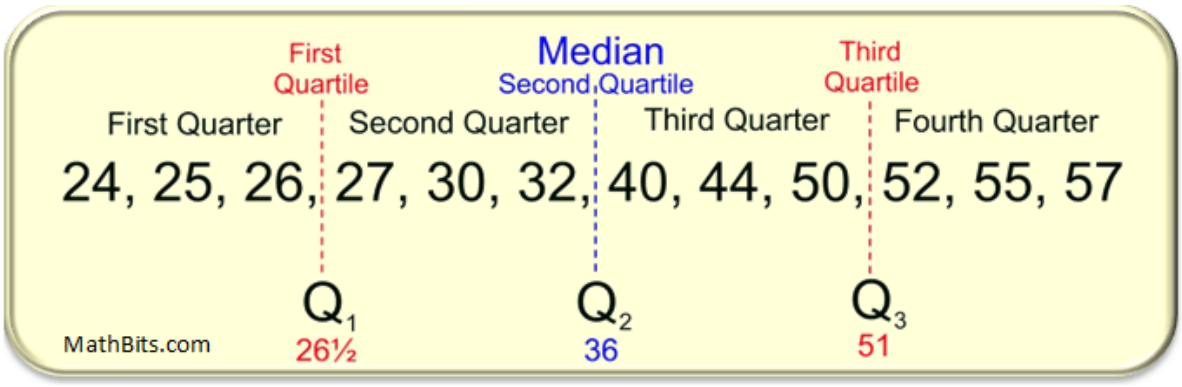
## 3) 최빈값

- 자료 중에서 가장 빈번하게 나타나는 값(빈도가 높은 값)
- 이산형 데이터(셀 수 있는 데이터)

In [51]:
```
# 판다스 메서드 이용하기
titanic['Pclass'].mode()
```

Out[51]:
```
0    3
Name: Pclass, dtype: int64
```

## 4) 4분위수




In [52]:
```
titanic[['Fare']].describe().T
```

Out[52]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Fare** | 891.0 | 32.204208 | 49.693429 | 0.0 | 7.9104 | 14.4542 | 31.0 | 512.3292 |

-연습문제-

연습문제를 풀어 봅시다.

In [53]:
```
# titanic의 Age에 대해서 다음을 각각 확인해 봅시다.

# .describe())
# 갯수, 평균, 표준편차, 최소값, 25%, 50%, 75%, 최댓값
titanic['Age'].describe()
```

Out[53]:
```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

In [54]:
```
# air의 Ozone에 대해서 다음을 각각 확인해 봅시다.
# 갯수, 평균, 표준편차, 최소값, 25%, 50%, 75%, 최댓값
air['Ozone'].describe()
```

Out[54]:
```
count    153.000000
mean      42.052288
std       30.156127
min        1.000000
25%       20.000000
50%       34.000000
75%       59.000000
max      168.000000
Name: Ozone, dtype: float64
```

# (2) 수치화 : 기초통계량

## 1) 시리즈.describe()

In [55]:
```
titanic['Fare'].describe()
```

Out[55]:
```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

## 2) 데이터프레임.describe()

In [56]:
```
titanic.head()
```

Out[56]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 7.2500 | Southampton |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 71.2833 | Cherbourg |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 7.9250 | Southampton |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 53.1000 | Southampton |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 8.0500 | Southampton |

In [57]:
```
# 데이터프레임의 숫자타입 변수들 기초통계량 조회
titanic.describe()
```

Out[57]:

| | PassengerId | Survived | Pclass | Age | Fare |
|---|---|---|---|---|---|
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 32.204208 |
| **std** | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 49.693429 |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 |
| **25%** | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 7.910400 |
| **50%** | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 14.454200 |
| **75%** | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 31.000000 |
| **max** | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 512.329200 |

In [58]:
```
# 데이터프레임의 전체 변수들 기초통계량 조회
titanic.describe(include='all').T
```

Out[58]:

| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% |
|---|---|---|---|---|---|---|---|---|---|---|
| **PassengerId** | 891.0 | NaN | NaN | NaN | 446.0 | 257.353842 | 1.0 | 223.5 | 446.0 | 668.5 |
| **Survived** | 891.0 | NaN | NaN | NaN | 0.383838 | 0.486592 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Pclass** | 891.0 | NaN | NaN | NaN | 2.308642 | 0.836071 | 1.0 | 2.0 | 3.0 | 3.0 |
| **Name** | 891 | 891 | Braund, Mr. Owen Harris | 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| **Sex** | 891 | 2 | male | 577 | NaN | NaN | NaN | NaN | NaN | NaN |
| **Age** | 714.0 | NaN | NaN | NaN | 29.699118 | 14.526497 | 0.42 | 20.125 | 28.0 | 38.0 |
| **Fare** | 891.0 | NaN | NaN | NaN | 32.204208 | 49.693429 | 0.0 | 7.9104 | 14.4542 | 31.0 |
| **Embarked** | 889 | 3 | Southampton | 644 | NaN | NaN | NaN | NaN | NaN | NaN |


air 데이터프레임에 대해서 기초 통계량을 구하고, 내용을 파악해 봅시다.

In [59]:
```
air.describe().T
```

Out[59]:

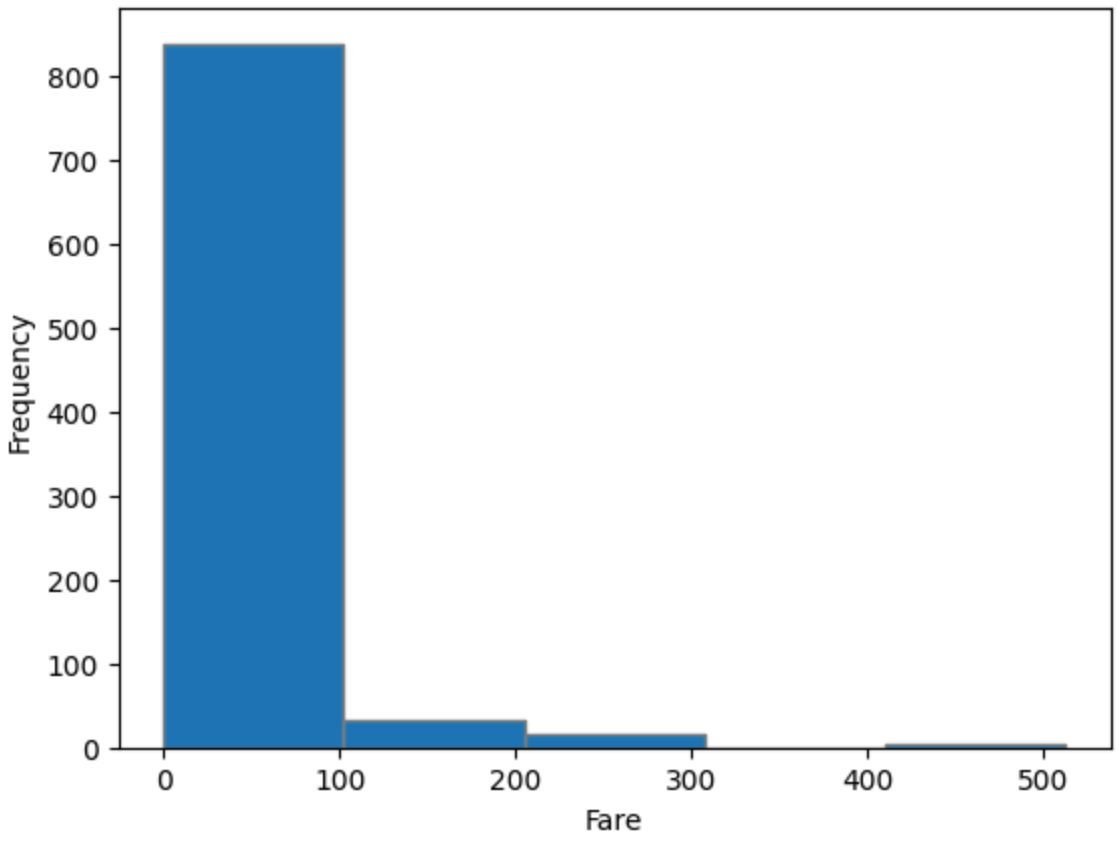
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Ozone** | 153.0 | 42.052288 | 30.156127 | 1.0 | 20.00 | 34.0 | 59.00 | 168.0 |
| **Solar.R** | 146.0 | 185.931507 | 90.058422 | 7.0 | 115.75 | 205.0 | 258.75 | 334.0 |
| **Wind** | 153.0 | 9.957516 | 3.523001 | 1.7 | 7.40 | 9.7 | 11.50 | 20.7 |
| **Temp** | 153.0 | 77.882353 | 9.465270 | 56.0 | 72.00 | 79.0 | 85.00 | 97.0 |

# (3) 시각화

## 1) 히스토그램

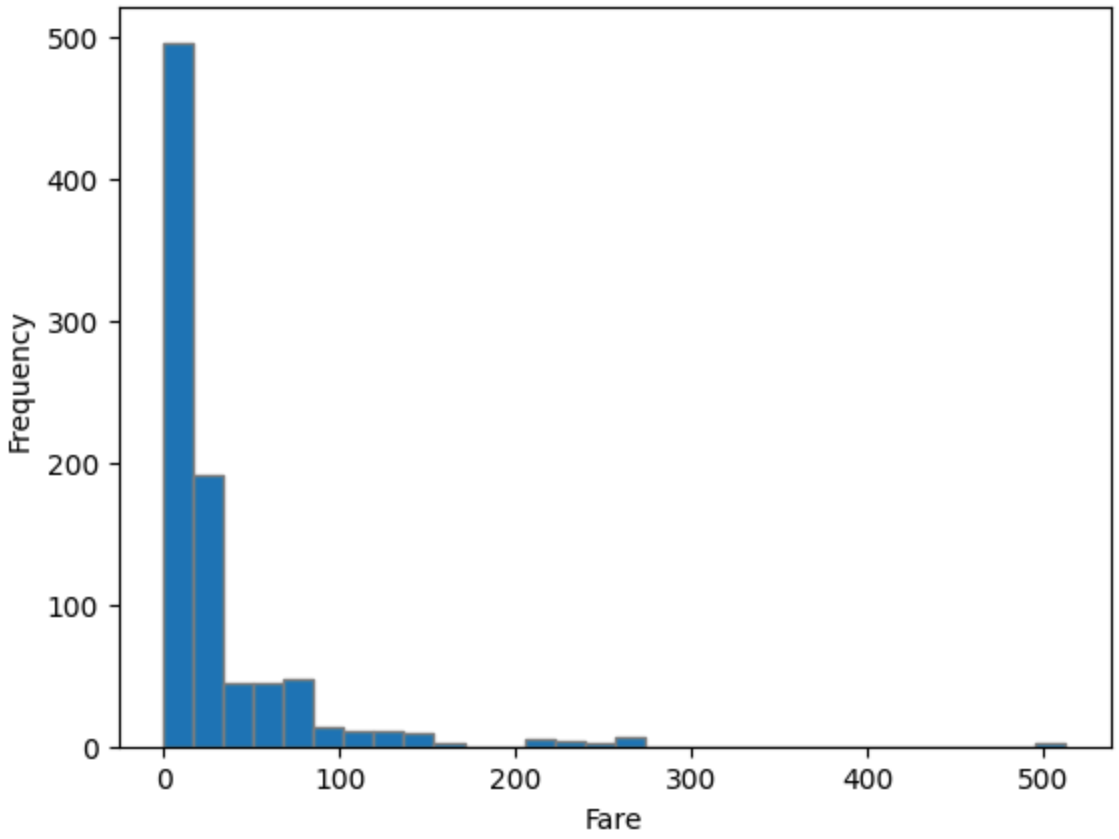
- 히스토그램 기본

In [60]:
```
# 히스토그램 : 도수분포표를 그래도 그래프로 만든것
# (숫자형 데이터를 정리하는데 가장 기본적인 그래프)
plt.hist(titanic.Fare, bins = 5, edgecolor = 'gray')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.show()
```

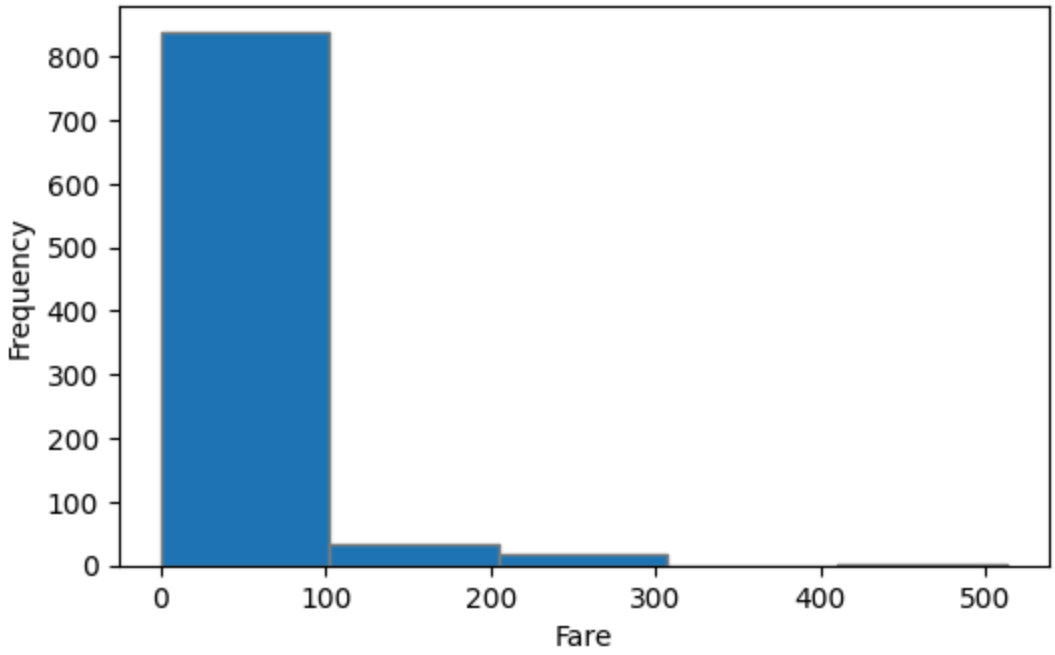


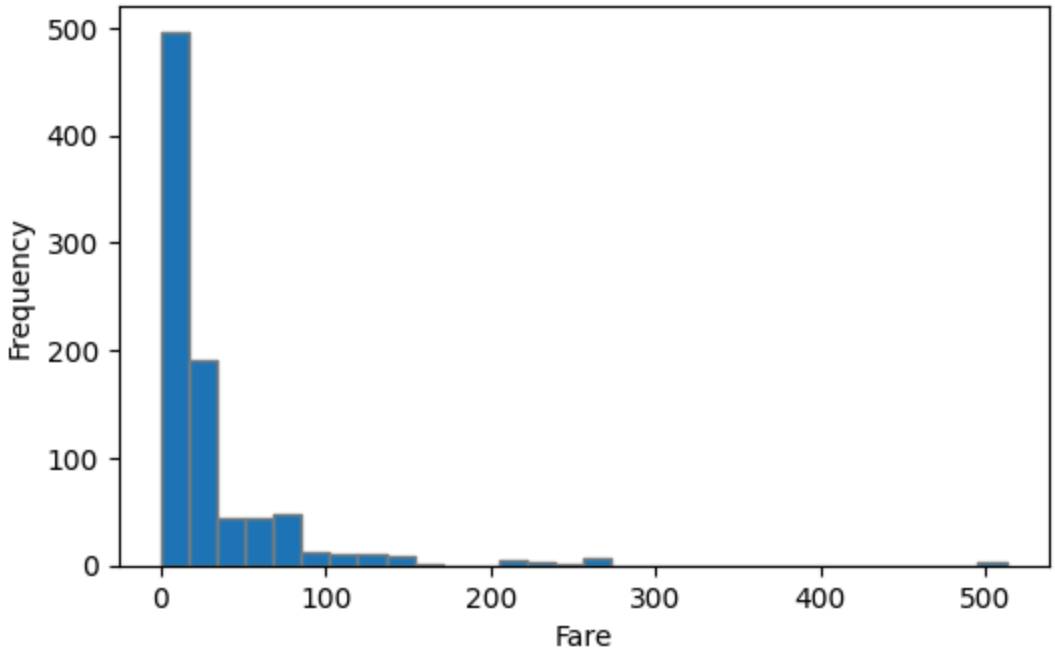
- 구간의 갯수를 조정해 봅시다.

In [61]:
```
plt.hist(titanic.Fare, bins = 30, edgecolor = 'gray')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.show()
```



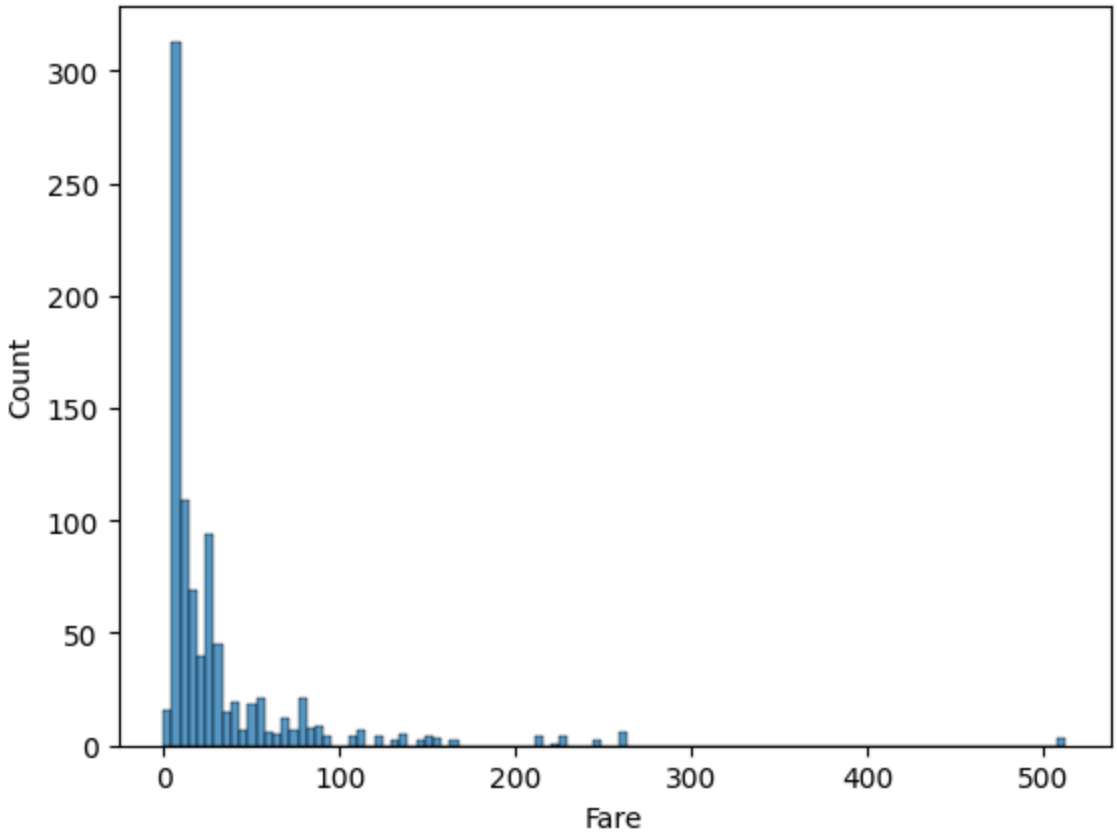

In [62]:
```
plt.figure(figsize=(6,8))
plt.subplot(2,1,1)
plt.hist(titanic.Fare, bins = 5, edgecolor = 'gray')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.subplot(2,1,2)
plt.hist(titanic.Fare, bins = 30, edgecolor = 'gray')
plt.xlabel('Fare')
plt.ylabel('Frequency')

plt.show()
```



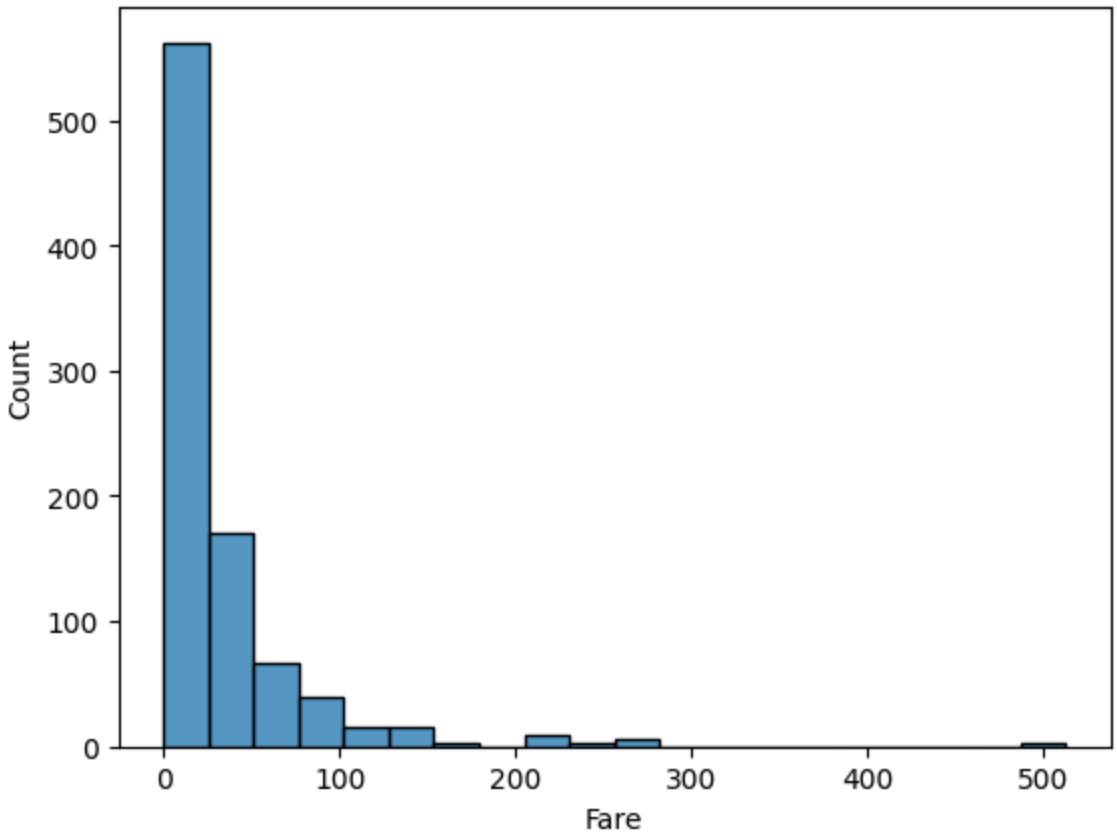




- sns.histplot

In [63]:
```
sns.histplot(x= 'Fare', data = titanic)
plt.show()
```



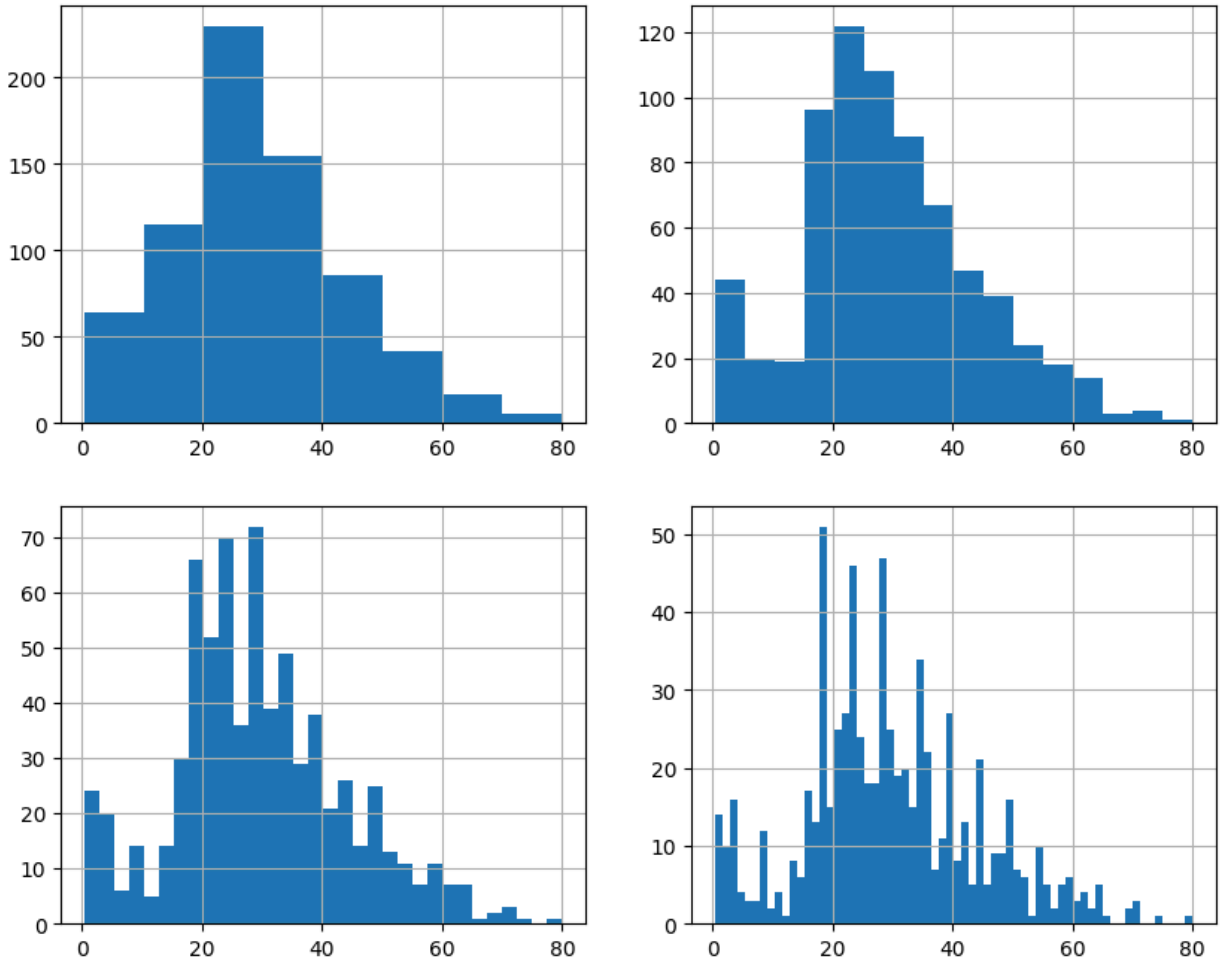

In [64]:
```
sns.histplot(x= 'Fare', data = titanic, bins = 20)
plt.show()
```



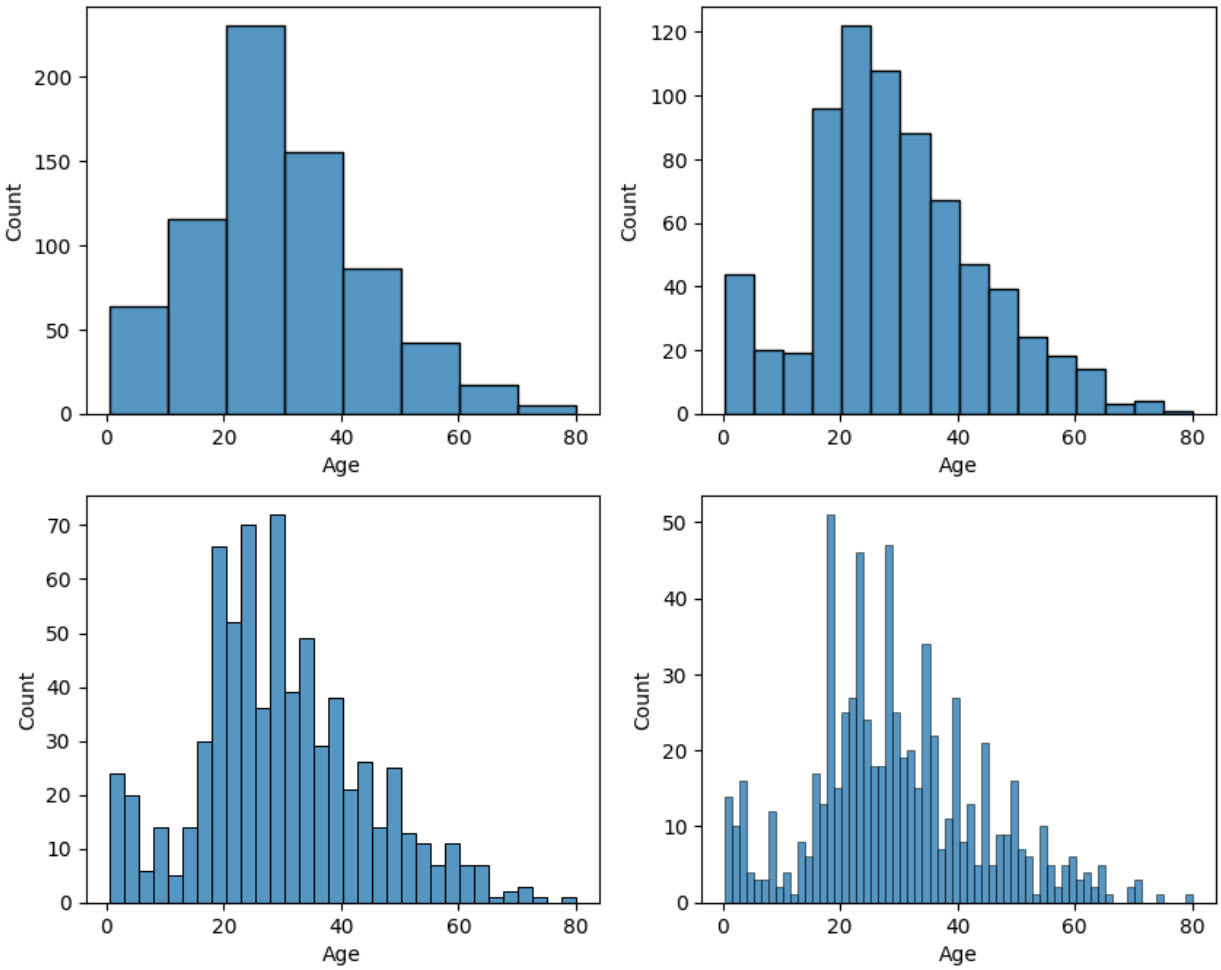

[문제1] titanic의 Age에 칼럼에 대해 히스토그램을 그려 봅시다.

- bins = 8, 16, 32, 64

In [65]:
```
bin = [8, 16, 32, 64]
plt.figure(figsize=(10, 8))

# 2행 2열 1번째
plt.subplot(2,2,1)
plt.hist(titanic['Age'], bins=bin[0])
plt.grid()

plt.subplot(2,2,2)
plt.hist(titanic['Age'], bins=bin[1])
plt.grid()

plt.subplot(2,2,3)
plt.hist(titanic['Age'], bins=bin[2])
plt.grid()

plt.subplot(2,2,4)
plt.hist(titanic['Age'], bins=bin[3])
plt.grid()

# 자동 완성
plt.tight_layout

plt.show()
```




In [66]:
```
bin = [8, 16, 32, 64]
plt.figure(figsize=(10, 8))

for i, t in enumerate(bin):
    plt.subplot(2, 2, i+1)
    sns.histplot(x = 'Age', data=titanic, bins=t)
plt.show()
```

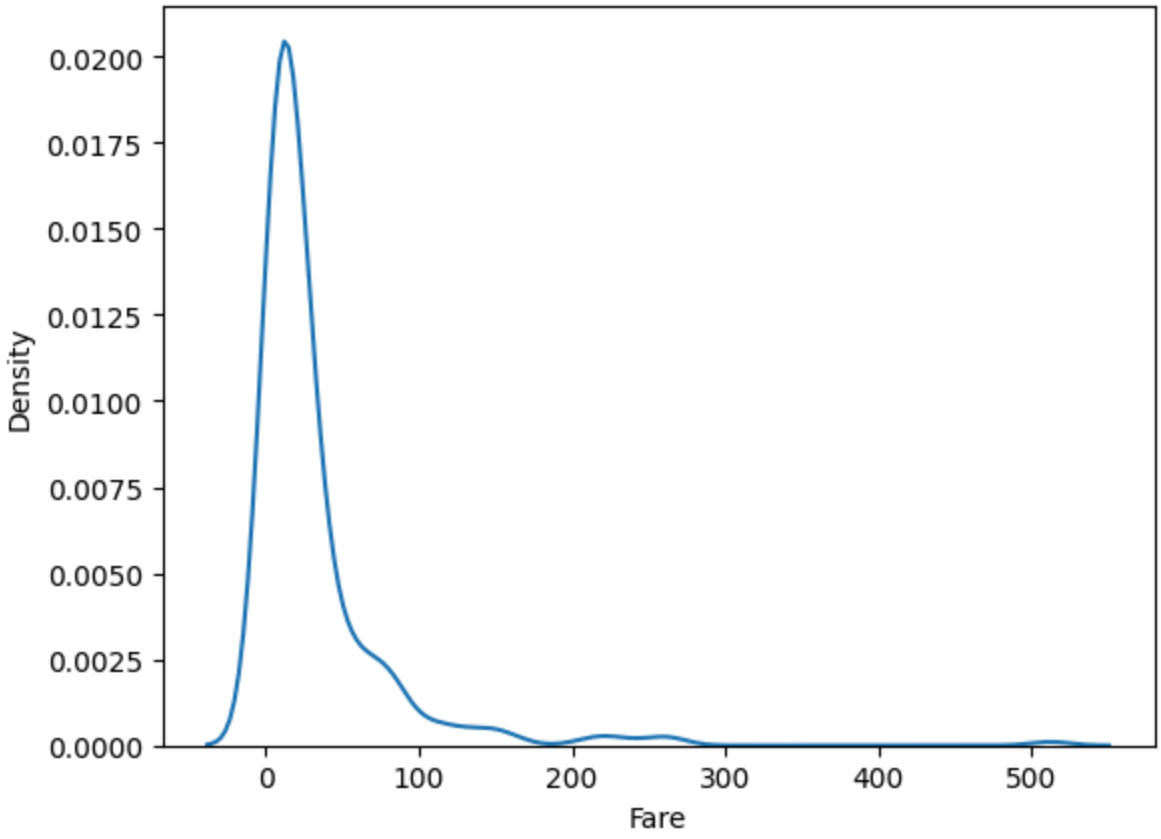



## 2) 밀도함수 그래프(kde plot)

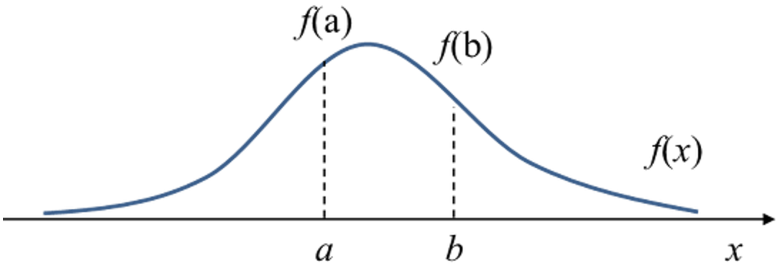
- 히스토그램의 단점
  - 구간(bin)의 너비를 어떻게 잡는지에 따라 전혀 다른 모양이 될 수 있음
- 밀도함수 그래프
  - 막대의 너비를 가정하지 않고 모든 점에서 데이터의 밀도를 추정하는 커널 밀도 추정(Kernel Density Estimation)방식을 사용하여 이러한 단점을 해결.
  - 밀도함수 그래프 아래 면적은 1

- 밀도함수 그래프 그리기

In [67]:
```
sns.kdeplot(titanic['Fare'])
# sns.kdeplot(x='Fare', data = titanic)

plt.show()
```




- 밀도 추정
  - 측정된(관측된) 데이터로부터 전체 데이터 분포의 특성을 추정
  - 예를 들어... OO역 사거리 일일 교통량을 측정한다고 해 봅시다.
    - 어제는 1200대 차량이 통과했고, 오늘은 1420대, 내일은, 모레는...
    - 이렇게 3개월간 매일 측정했다고 할 때,
    - 우리는 약 90일치의 데이터를 가지고 일일 교통량 분포를 히스토그램으로 그려볼 수 있습니다.
    - 그리고 나서 특정한 날의 교통량이 얼마나 될지 확률로 나타냄.

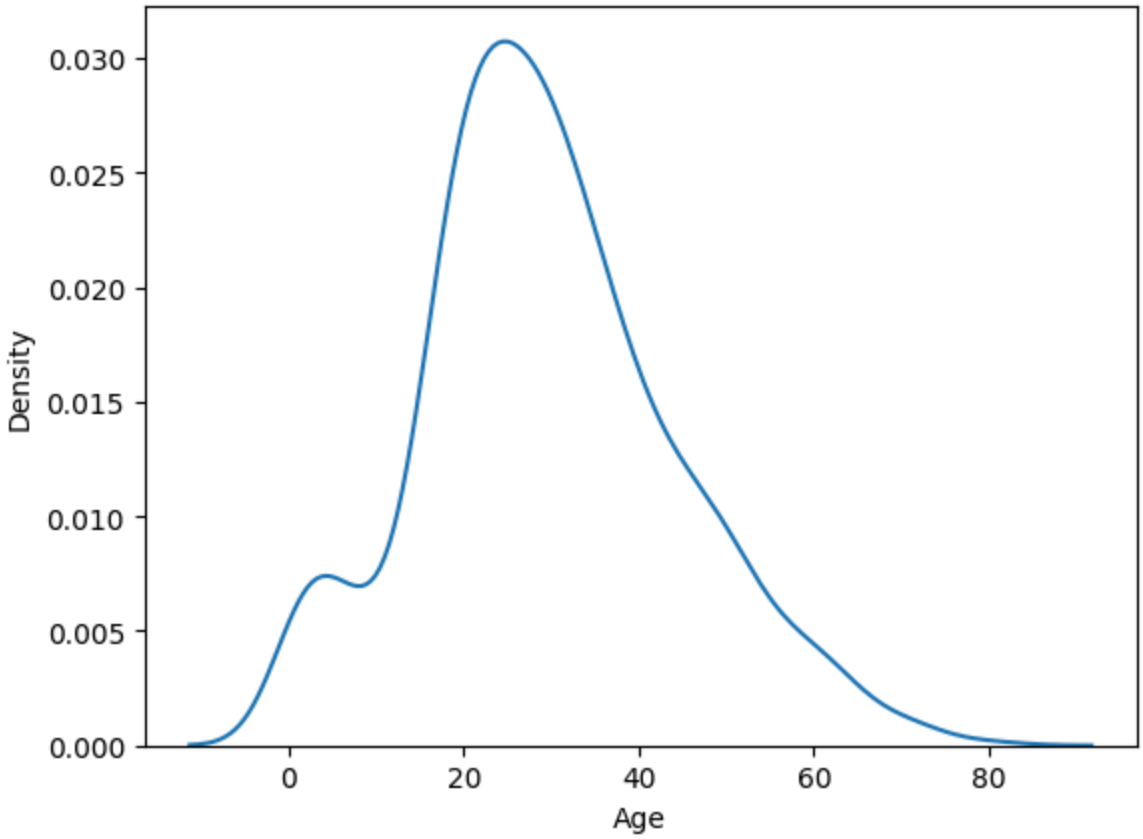



$$P(a \leq x \leq b) = \int_a^b f(x)dx$$


[문1] titanic Age에 대해서 밀도함수 그래프를 그려봅시다.

히스토그램과 어떤 차이가 있나요?

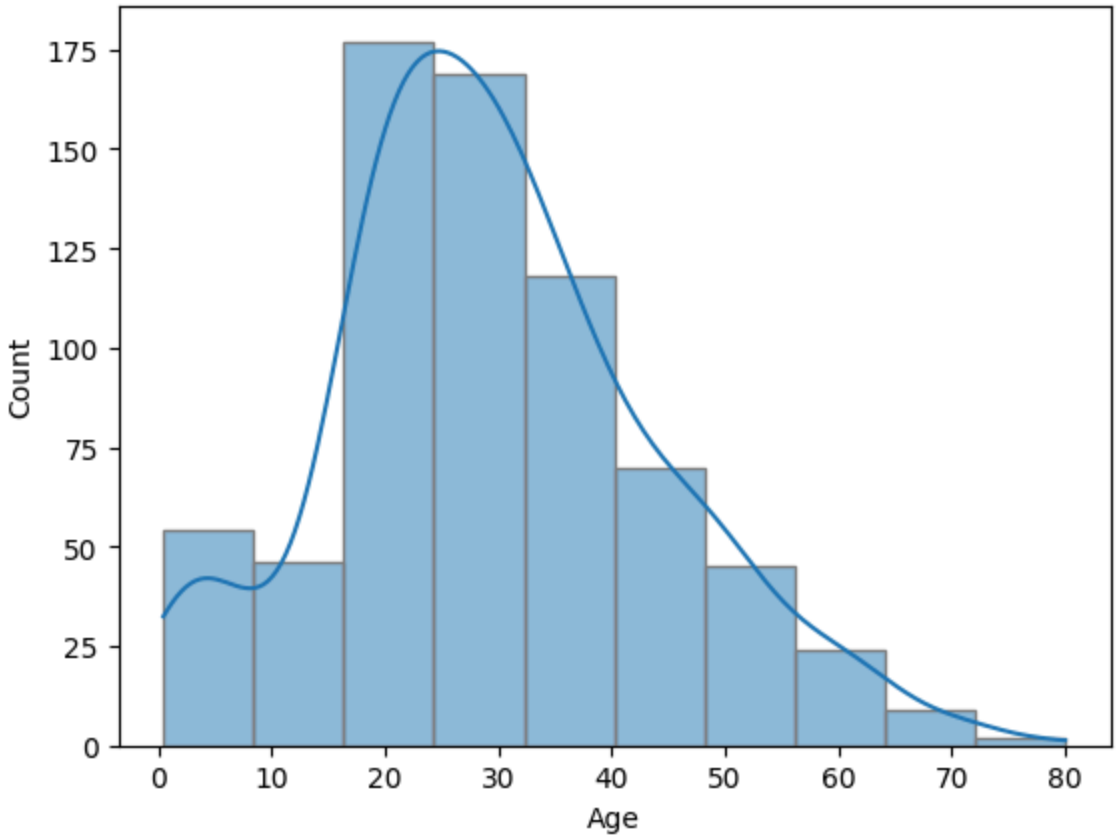
In [68]:
```
sns.kdeplot(titanic['Age'])

plt.show()
```




[문2] titanic Age에 대해서 히스토그램을 그려봅시다.

- 단, 이번에는 sns.histplot 을 이용합니다.
- 옵션으로 kde = True 를 지정해 봅시다.

In [69]:
```
# kde = True 밀도 추정 선 그래프에 추가
sns.histplot(titanic['Age'], bins=10, edgecolor = 'gray', kde=True)
plt.show()
```



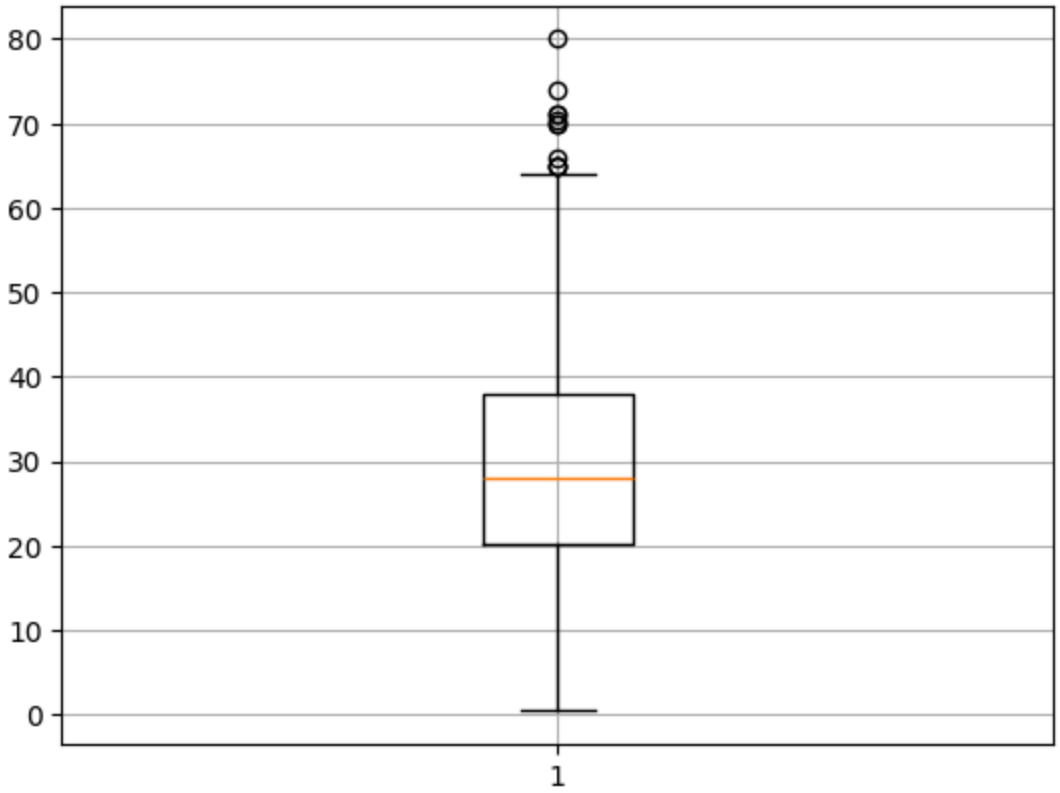

## 3) boxplot

**주의사항** : 값에 **NaN**이 있으면 그래프가 그려지지 않습니다.

- boxplot 기본

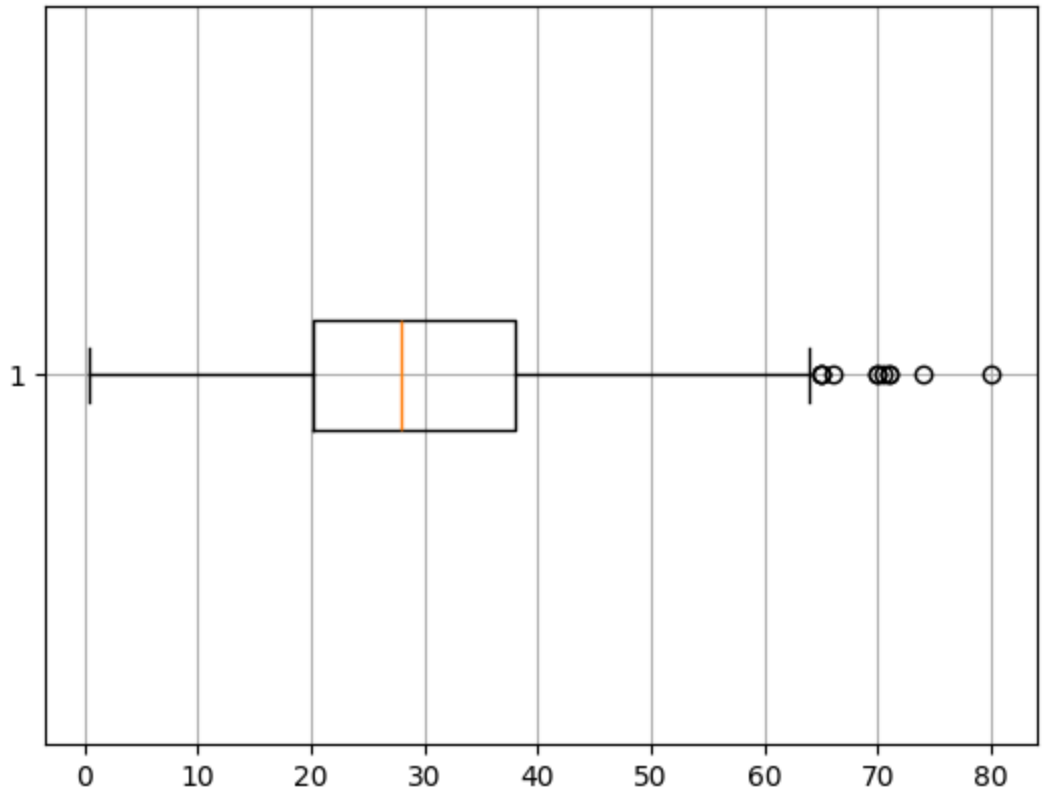
In [70]:
```
# titanic['Age']에는 NaN이 있습니다. 이를 제외한 데이터
temp = titanic.loc[titanic['Age'].notnull()]
```

In [71]:
```
plt.boxplot(temp['Age'])
plt.grid()
plt.show()
```



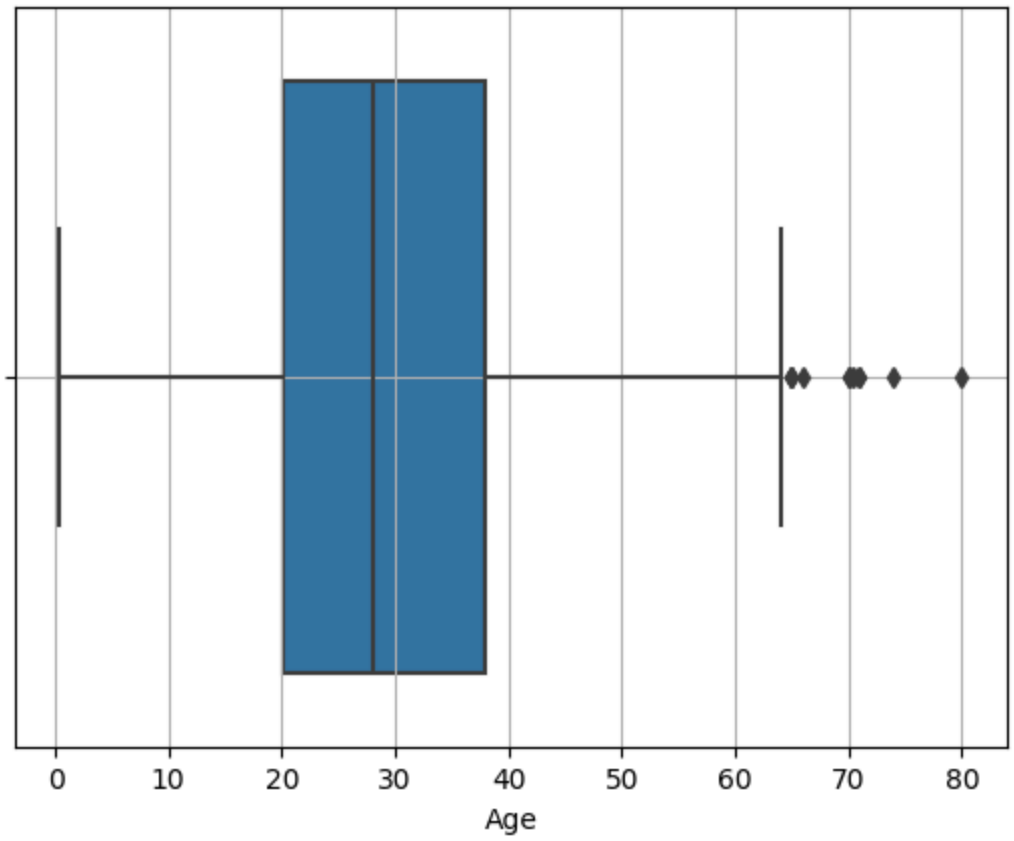

- 옆으로 그리기

In [72]:
```
plt.boxplot(temp['Age'], vert = False)
plt.grid()
plt.show()
```

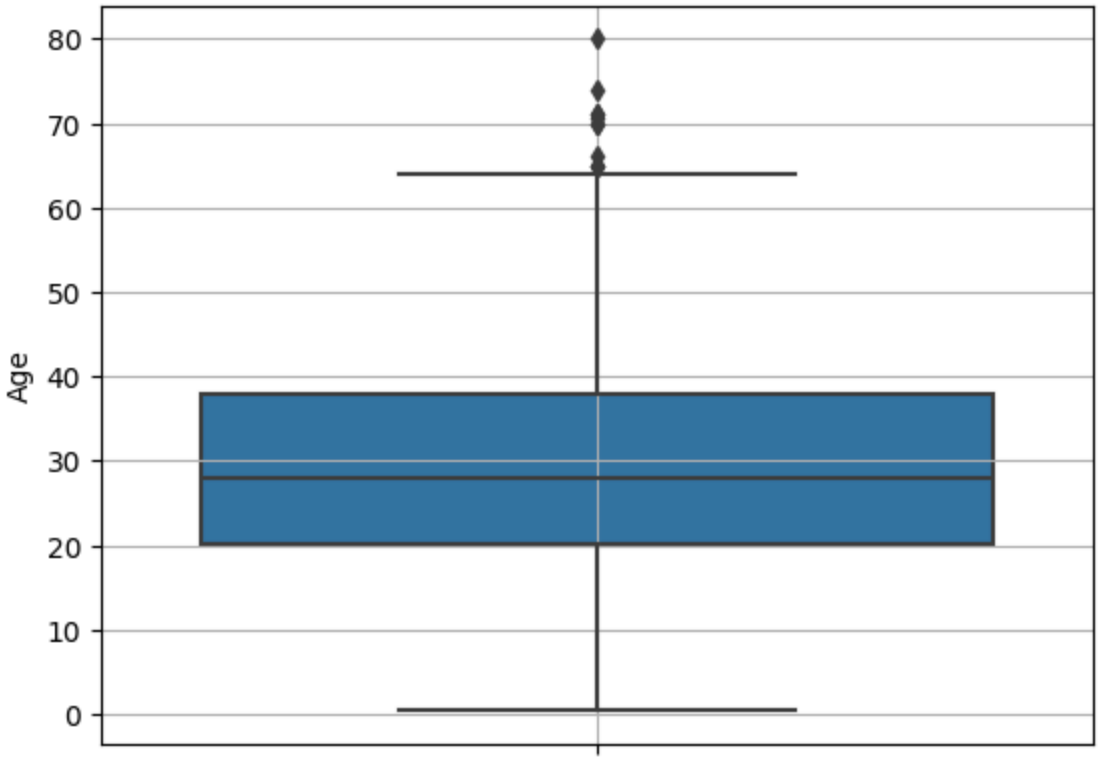


- sns.boxplot
  - seaborn 패키지 함수들은 NaN을 알아서 빼줍니다.

In [73]:
```
sns.boxplot(x = titanic['Age'])
#sns.boxplot(x = 'Age', data = titanic)
plt.grid()
plt.show()
```




In [74]:
```
sns.boxplot(y = titanic['Age'])
plt.grid()
plt.show()
```





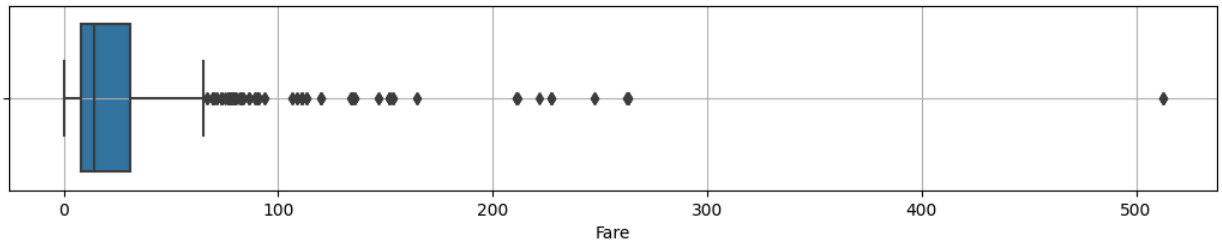
[문1] titanic의 Fare에 대해서 각각 박스 플롯을 그리고 해석해 봅시다.

In [75]:
```
# describe()
titanic[['Fare']].describe().T
```

Out[75]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Fare** | 891.0 | 32.204208 | 49.693429 | 0.0 | 7.9104 | 14.4542 | 31.0 | 512.3292 |

In [76]:
```
plt.figure(figsize=(13, 2))
sns.boxplot(x='Fare', data = titanic)
plt.grid()
plt.show()
```




# 3.복습문제

## (1) 환경준비

- 라이브러리 불러오기

In [77]:
```
import numpy as np
import pandas as pd

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

- 보스톤 집값 데이터를 이용하여 다음의 복습문제를 풀어 봅시다.

**변수설명**

> - medv : 1978 보스턴 주택 가격, 506개 타운의 주택 가격 중앙값 (단위 1,000 달러) <== Target
> - crim 범죄율
> - zn 25,000 평방피트를 초과 거주지역 비율
> - indus 비소매상업지역 면적 비율
> - chas 찰스강변 위치(범주 : 강변1, 아니면 0)
> - nox 일산화질소 농도
> - rm 주택당 방 수
> - age 1940년 이전에 건축된 주택의 비율
> - dis 직업센터의 거리
> - rad 방사형 고속도로까지의 거리
> - tax 재산세율
> - ptratio 학생/교사 비율
> - lstat 인구 중 하위 계층 비율

In [78]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [79]:
```
# 보스톤 집값 데이터
boston = pd.read_csv('https://raw.githubusercontent.com/DA4BAM/dataset/master/boston.csv')
boston.head()
```

Out[79]:

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | lstat | medv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 4.98 | 24.0 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 9.14 | 21.6 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 4.03 | 34.7 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 2.94 | 33.4 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 5.33 | 36.2 |

# (2) 단변량 분석

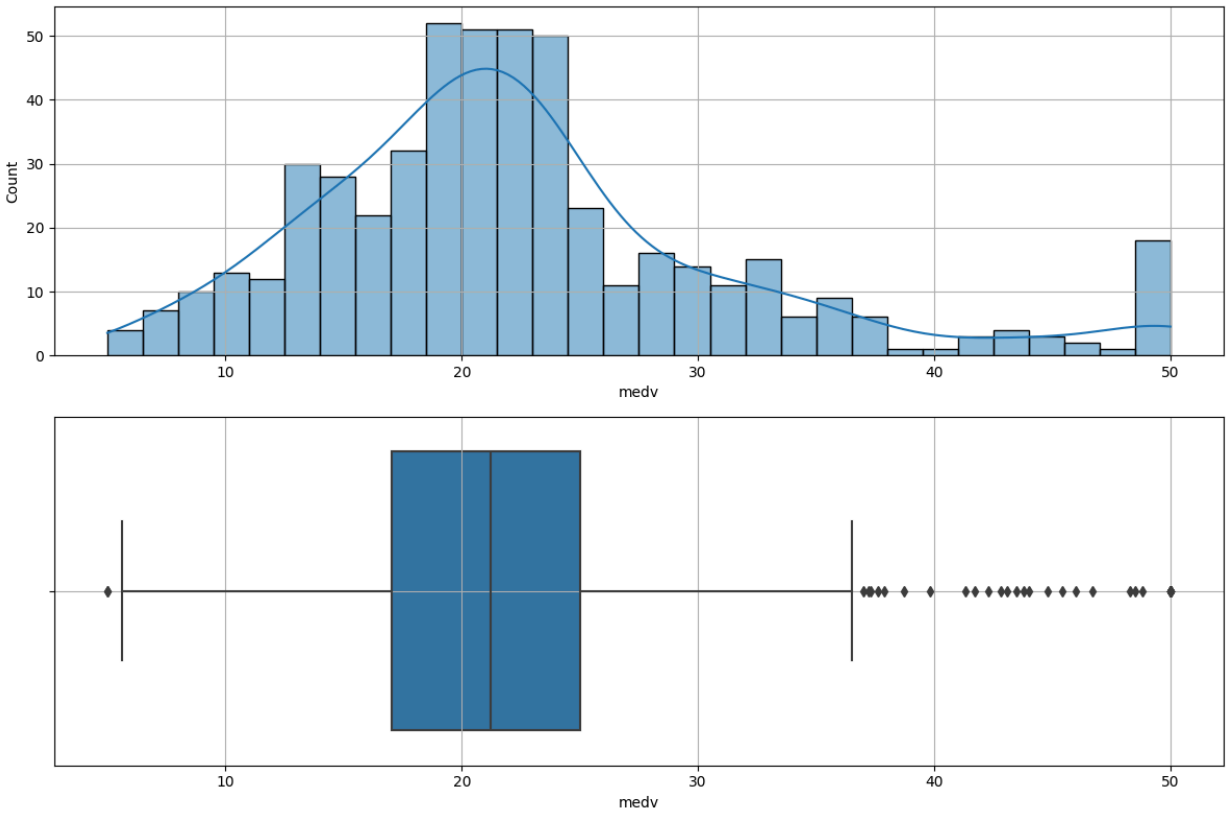
- 단변량 분석 코드를 함수로 만들기(중요!)
  - 복잡하고 반복적인 코드를 함수로 만들어서 사용해 봅시다.

In [86]:
```
def eda_1_n(data, var, bins = 30) :
    # 기초 통계량
    display(data[[var]].describe().T)

    # 시각화
    plt.figure(figsize = (12,8))
    plt.subplot(2,1,1)
    sns.histplot(data[var], bins = bins, kde = True)
    plt.grid()

    plt.subplot(2,1,2)
    sns.boxplot(x = data[var])
    plt.grid()
    plt.tight_layout()
    plt.show()
```

- medv(집값)

In [87]:
```
eda_1_n(boston, 'medv')
```

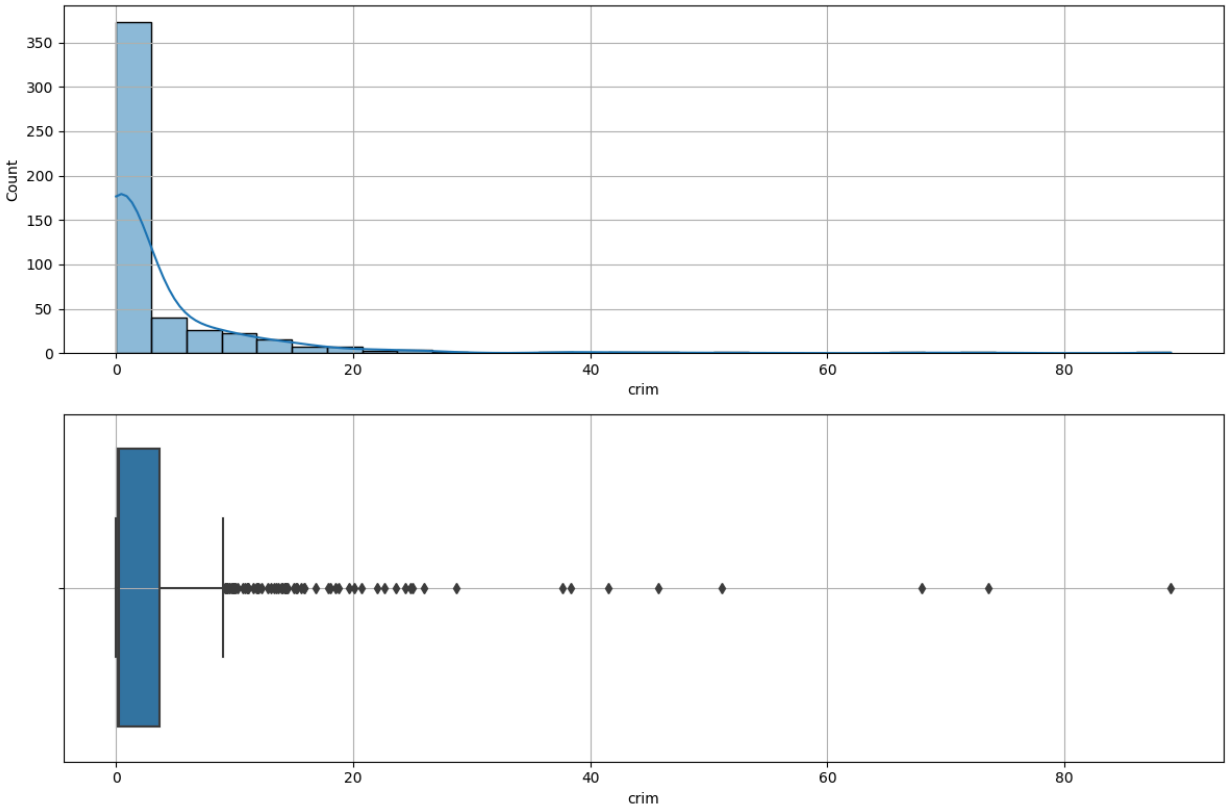
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **medv** | 506.0 | 22.532806 | 9.197104 | 5.0 | 17.025 | 21.2 | 25.0 | 50.0 |




- crim(범죄율)

In [88]:
```
eda_1_n(boston, 'crim') # 범죄율  # 분자/ 분모, 기간 등 분석하기 위해 정확하게 알아야 한다.
```

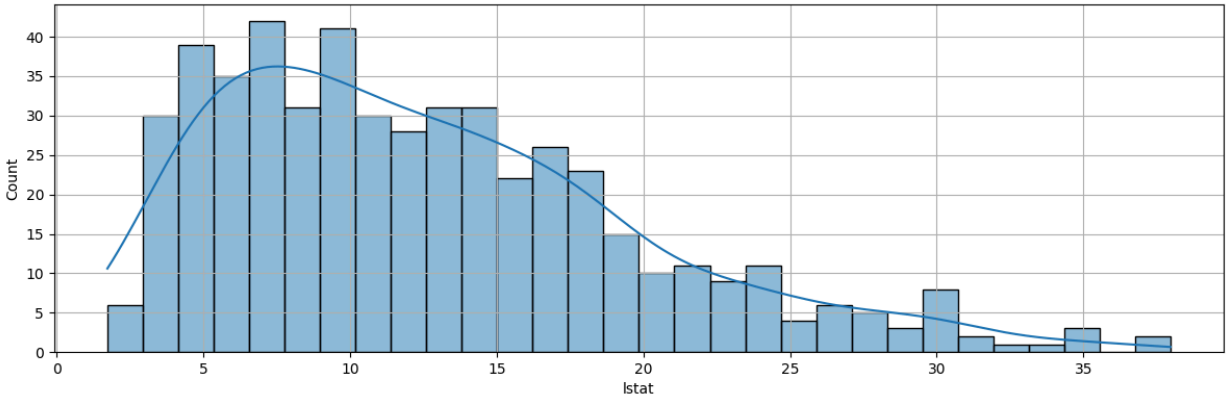
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **crim** | 506.0 | 3.613524 | 8.601545 | 0.00632 | 0.082045 | 0.25651 | 3.677083 | 88.9762 |

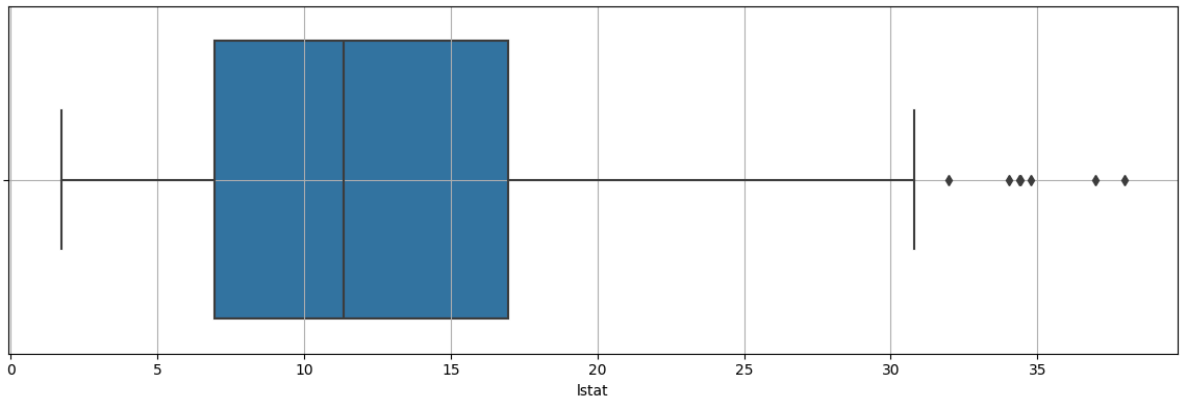



- lstat(하위계층 비율)

In [89]:
```
eda_1_n(boston, 'lstat')
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **lstat** | 506.0 | 12.653063 | 7.141062 | 1.73 | 6.95 | 11.36 | 16.955 | 37.97 |

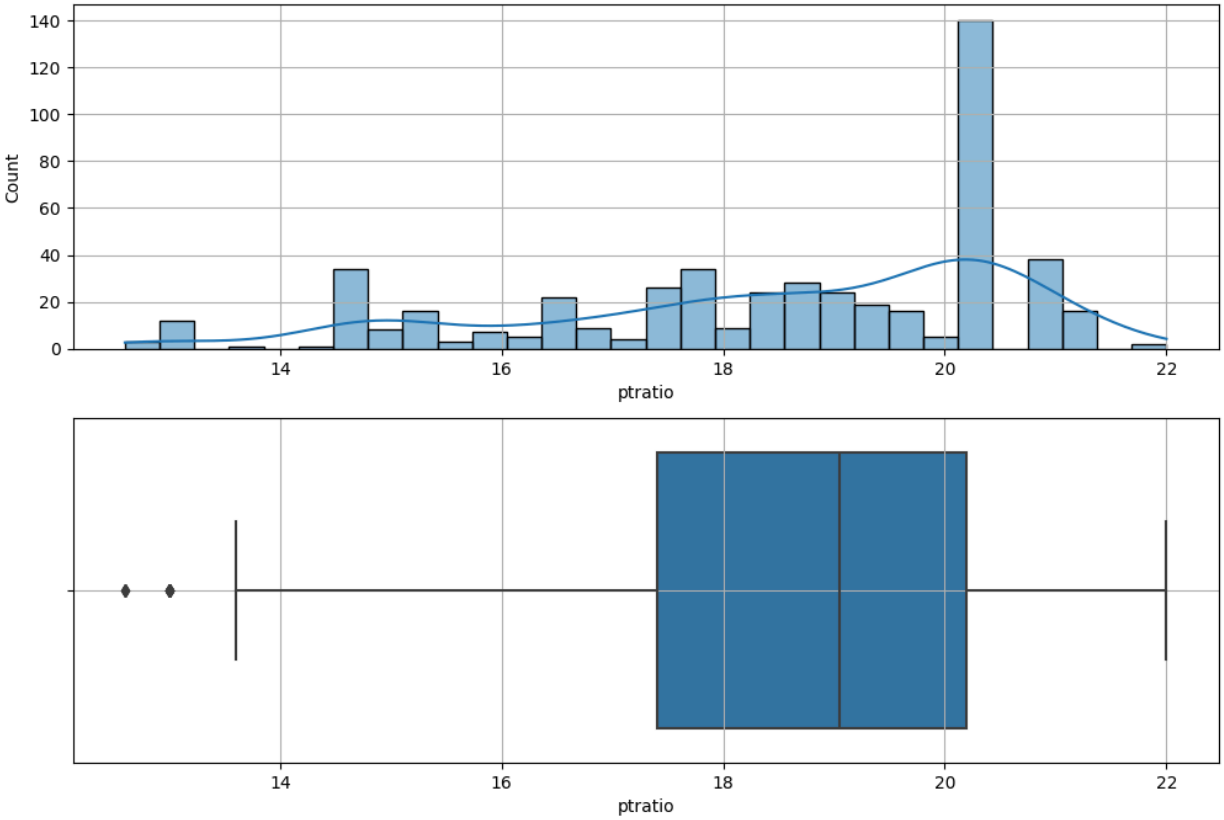



- ptratio(교사1명당 학생수)

In [84]:
```
eda_1_n(boston, 'ptratio') # 학생수 / 교사1명
```

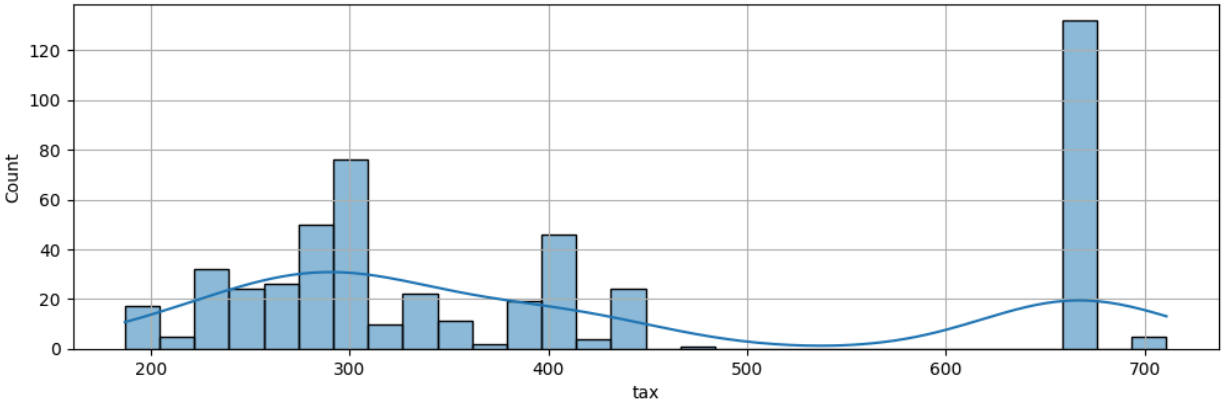
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **ptratio** | 506.0 | 18.455534 | 2.164946 | 12.6 | 17.4 | 19.05 | 20.2 | 22.0 |

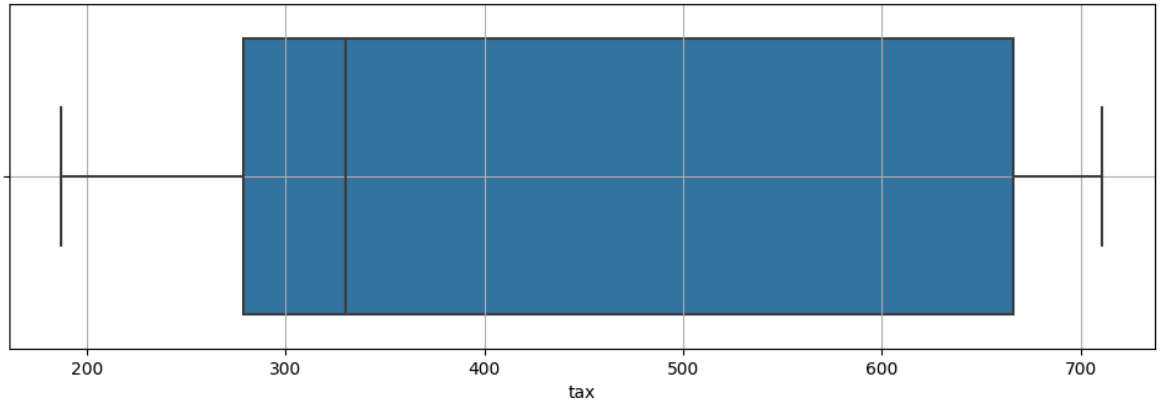



- tax(재산세)

In [85]:
```
eda_1_n(boston, 'tax')
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **tax** | 506.0 | 408.237154 | 168.537116 | 187.0 | 279.0 | 330.0 | 666.0 | 711.0 |


Count
tax
0
20
40
60
80
100
120
200
300
400
500
600
700


tax
200
300
400
500
600
700

In [ ]: