

## [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
```

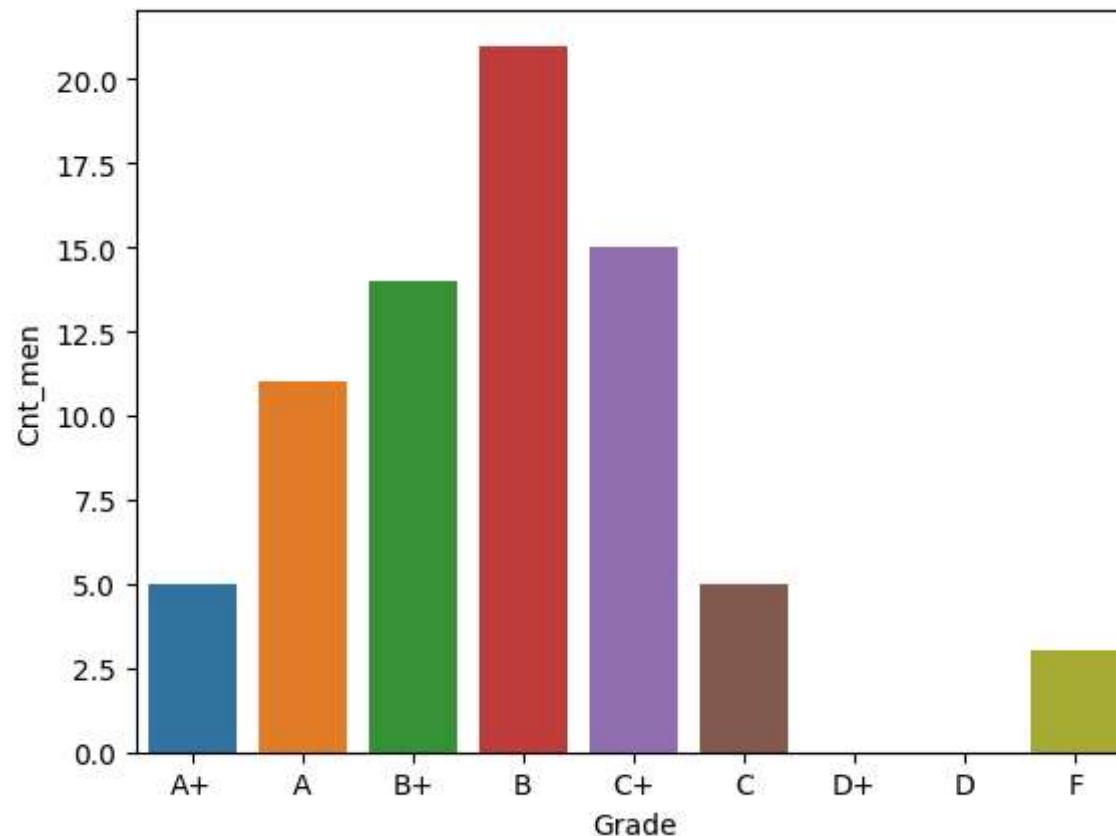
```
In [53]: ##### 2. 그룹별 통계 데이터 프레임 만들기
##### >'Grade'별 학생수 구하기
```

```
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):
```

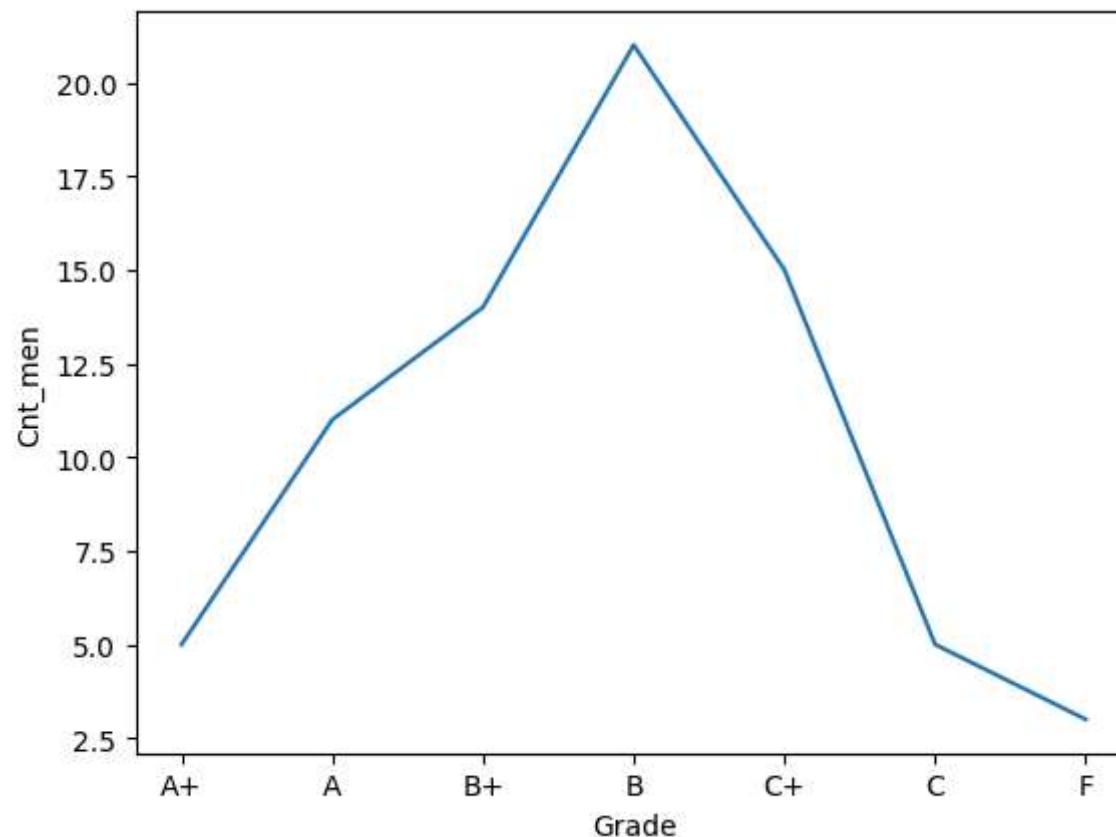
```
Out[56]: <Axes: xlabel='Grade', ylabel='Cnt_men'>
```



```
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):
```

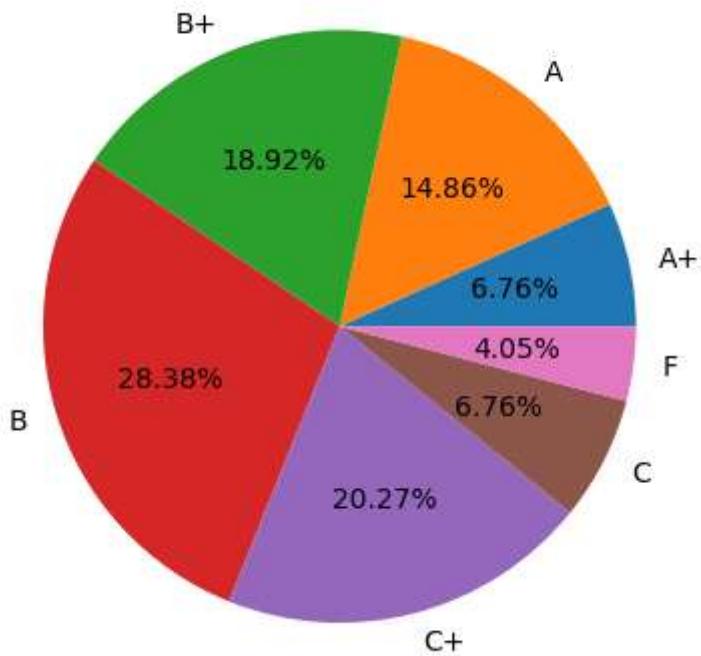
Out[57]: <Axes: xlabel='Grade', ylabel='Cnt\_men'>



```
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

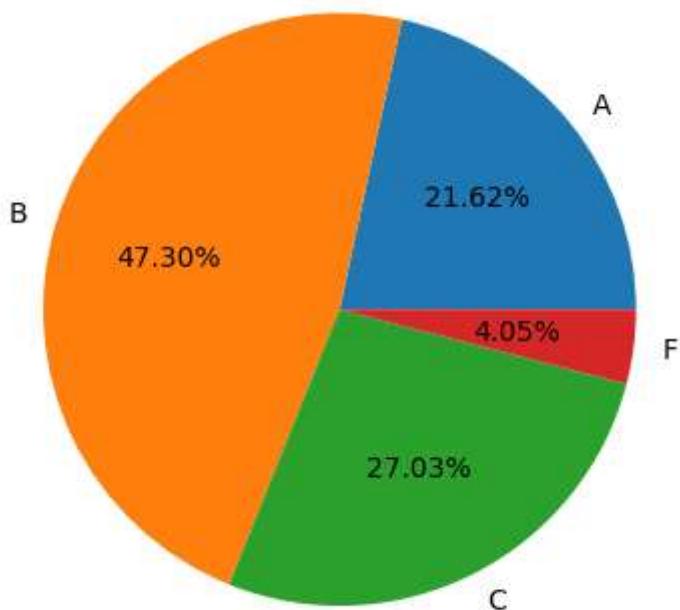
```
In [61]: ##### 8. 'Grade_group'별 파이 그래프 그리기
##### >'Grade_group'별 학생수 'Cnt_men' 구하기
##### >'Grade_group'별 학생수 비율 'Rate_men' 구하기
```

```
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

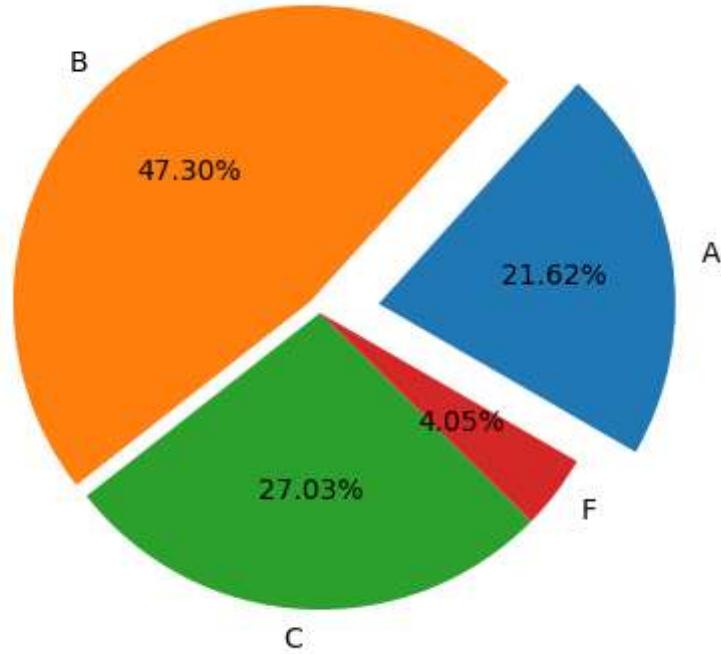
```
In [62]: ##### 8.1 'Grade_group'별 파이 그래프 그리기
##### > 'Grade_group'별 학생 수(Cnt_men)에 대한 파이그래프 그리기
```

```
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%')])
```



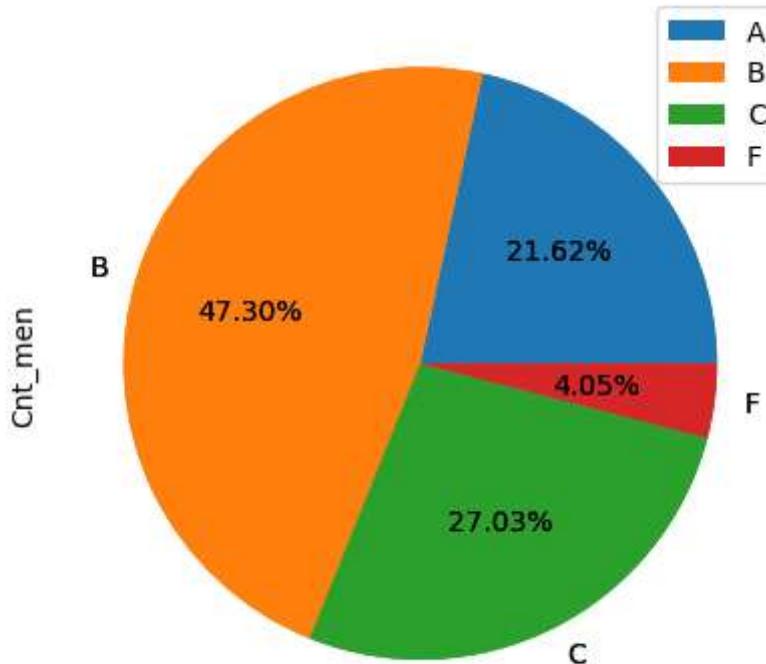
```
In [63]: ##### 8.2 'Grade_group'별 파이 그래프 그리기
##### > 'Grade_group'별 학생 수(Cnt_men)에 대한 파이그래프 그리기
```

```
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
```

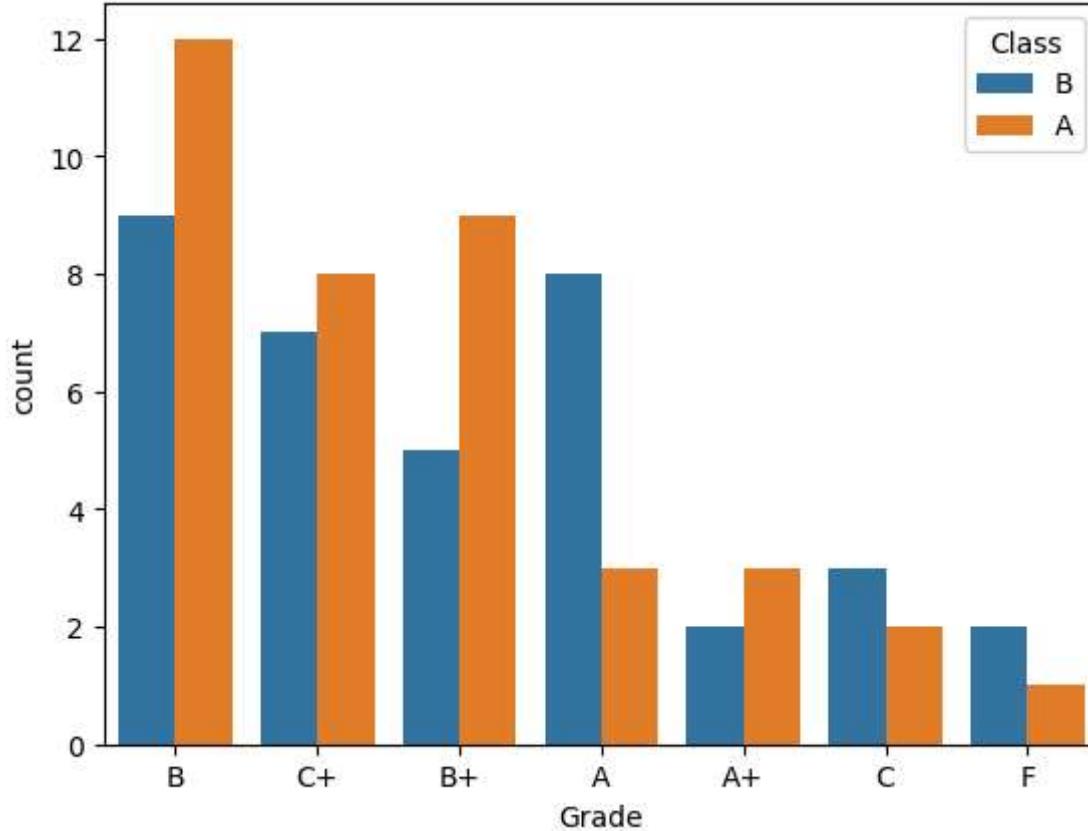
```
Out[64]: <Axes: ylabel='Cnt_men'>
```



```
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):
```

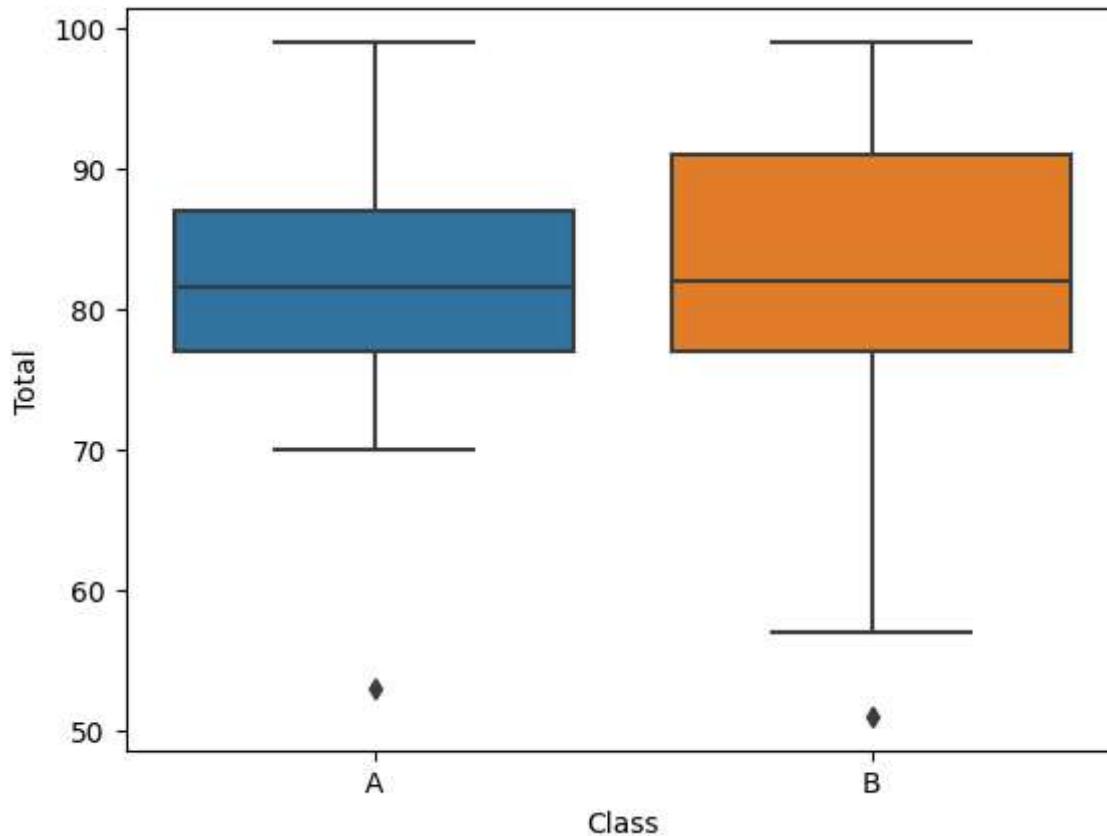
```
Out[65]: <Axes: xlabel='Grade', ylabel='count'>
```



```
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 [ ]: