

# Ch07 데이터 정제하기

```
#.nan() #.isna() #.dropna() #.fillna() #.groupby() #.agg() #.isnull()  
#.where() #.replace() #.boxplot() #.quantile()
```

데이터 정제란? 정상적이지 않은 데이터를 찾아 원하는 방향으로 바로잡는 것

- > 결측치(missing value)는 누락된 값을 의미, 즉 원하는 값이 아닌 값
- > Python에서는 결측치가 'NaN'으로 표시된다. NaN은 "Not a Number"의 약자
- > 이상치(anomaly)는 정상 범위에서 벗어난 이상한 값

In [ ]:

## 07-1 결측치 정제하기

### 결측치 생성하기

#### numpy.nan

- > Python에서는 NaN 값을 실수형(float)의 한 형태로 처리

In [4]:

```
## 결측치 만들기  
import pandas as pd  
  
df = pd.DataFrame({'sex' : ['M', 'F', np.nan, 'M', 'F'],  
                  'score' : [5, 4, 3, 4, np.nan]})  
df
```

```
Out[4]:
```

	sex	score
0	M	5.0
1	F	4.0
2	NaN	3.0
3	M	4.0
4	F	NaN

```
In [5]: ## 결측치는 연산에서 자동 제외  
df['score'] + 10
```

```
Out[5]: 0    15.0  
1    14.0  
2    13.0  
3    14.0  
4     NaN  
Name: score, dtype: float64
```

```
In [6]: ## NaN은 실수형  
type(df['sex'][2])
```

```
Out[6]: float
```

---

## [ChatGPT]에게 물어봐요!

[질문] 결측치 값 'NaN'이 float형인 이유는?

[ChatGPT답변]

'NaN'(Not a Number) 값은 float 형식으로 표현됩니다. 이것은 다음과 같은 이유로 선택되었습니다:

**표준화:** 'NaN'은 IEEE 부동 소수점 표준에 따른 부동 소수점 값으로 간주됩니다. 이는 데이터 타입을 표준화하고 플랫폼 간 이식성을 유지하기 위한 것입니다.

**타입 일관성:** 대부분의 수치형 데이터는 부동 소수점으로 처리됩니다. 'NaN'을 float로 표현하면 데이터 유형이 일관되게 유지됩니다.

**편의성:** 많은 수치 연산과 라이브러리는 부동 소수점을 사용합니다. 'NaN'을 float로 표현하면 결측치 처리를 위한 다양한 수치 연산이 간편해집니다.

**연산 규칙:** 'NaN'은 수치형 데이터와 관련된 연산 규칙에 따라 동작합니다. 이는 데이터의 누락 여부를 확인하고 처리하기 위한 표준화된 방법을 제공합니다.

In [ ]:

## 결측치 확인하기

> `.isna()`

> `.isna().sum()`

In [7]: `## 결측치 존재 확인하기  
pd.isna(df)`

Out[7]: `sex score`

<b>0</b>	False	False
<b>1</b>	False	False
<b>2</b>	True	False
<b>3</b>	False	False
<b>4</b>	False	True

In [8]: `## 결측치 존재 확인하기  
df.isna()`

Out[8]: `sex score`

<b>0</b>	False	False
<b>1</b>	False	False
<b>2</b>	True	False
<b>3</b>	False	False
<b>4</b>	False	True

In [9]: `## 결측치 존재 확인하기  
df[['sex', 'score']].isna()`

```
Out[9]:    sex  score
```

	sex	score
0	False	False
1	False	False
2	True	False
3	False	False
4	False	True

```
In [10]: ## 결측치 존재 확인하기  
df['sex'].isna()
```

```
Out[10]: 0 ... False  
1 ... False  
2 ... True  
3 ... False  
4 ... False  
Name: sex, dtype: bool
```

```
In [11]: ## 결측치 빈도 확인하기  
pd.isna(df).sum()
```

```
Out[11]: sex      1  
score     1  
dtype: int64
```

```
In [12]: ## 결측치 빈도 확인하기  
df[['sex', 'score']].isna().sum()
```

```
Out[12]: sex      1  
score     1  
dtype: int64
```

```
In [ ]:
```

## 결측치 행으로 데이터 프레임 만들기

```
In [13]: df['sex'].isna()
```

```
Out[13]: 0    False
         1    False
         2     True
         3    False
         4    False
Name: sex, dtype: bool
```

```
In [14]: ## 결측치 행 확인: 결측치 행으로 데이터 프레임 만들기
df[df['sex'].isna()]
```

```
Out[14]:   sex  score
           2  NaN    3.0
```

```
In [15]: ## 결측치 행 확인: 결측치 행으로 데이터 프레임 만들기
df[df['score'].isna()]
```

```
Out[15]:   sex  score
           4    F    NaN
```

```
In [ ]:
```

## 결측치 제거하기

> `.dropna()`

>> `axis`: 행(0) 단위로 결측치를 제거. 만약 `axis=1`로 설정하면 열 단위로 제거

>> `how`: 결측치 제거 방식

>>> `'any'` (기본값): 하나 이상의 결측치가 있으면 해당 행 또는 열을 제거

>>> `'all'`: 행과 열의 모든 값이 결측치인 경우에만 제거

>> `thresh`: 유효한 값의 개수가 이 값보다 적은 행이나 열은 제거하지 않음.

>> `subset`: 제거할 때 영향을 받는 특정 열(들)을 지정

>> `inplace`: 기본값은 `False`, 새로운 `DataFrame` 또는 `Series`를 반환, 만약 `True`로 설정하면 원본 데이터만 변경

```
In [16]: df
```

Out[16]:

	sex	score
0	M	5.0
1	F	4.0
2	NaN	3.0
3	M	4.0
4	F	NaN

In [17]: ## > 결측치 있는 행 제거하기

```
df.dropna(subset = ['score']) # score 결측치 제거
```

Out[17]:

	sex	score
0	M	5.0
1	F	4.0
2	NaN	3.0
3	M	4.0

In [18]: ## > 결측치가 제거된 프레임 만들기

```
df = df.dropna(subset = ['score']) # score 변수(열)에서 결측치가 제거된 데이터 프레임 만들기  
df
```

Out[18]:

	sex	score
0	M	5.0
1	F	4.0
2	NaN	3.0
3	M	4.0

모든 결측치가 제거된 데이터 추출하기

> `.dropna()` 결측치가 있는 행 제거

```
In [19]: ## #모든 결측치가 제거된 데이터 추출하기  
df_nomiss = df.dropna() # 모든 결측치가 있는 행 제거  
df_nomiss
```

```
Out[19]:   sex  score  
0    M    5.0  
1    F    4.0  
3    M    4.0
```

```
In [15]: ## #모든 결측치가 제거된 데이터 추출하기  
df_nomiss = df.dropna(axis = 1) # 모든 결측치가 있는 행 제거  
df_nomiss
```

```
Out[15]:   score  
0      5.0  
1      4.0  
2      3.0  
3      4.0
```

## 여러 변수에 결측치 없는 데이터 추출하기

```
In [18]: ## 여러 변수에 결측치 없는 데이터 추출하기  
df_nomiss = df.dropna(subset = ['score', 'sex']) # score, sex 결측치 제거  
df_nomiss
```

```
Out[18]:   sex  score  
0    M    5.0  
1    F    4.0  
3    M    4.0
```

## 결측치 제거하지 않고 분석하기

> `.mean()`, `.sum()`, `.groupby()`는 결측치를 제외하고 연산을 수행

```
In [26]: ##### **<> 결측치 제거하지 않고 분석하기**  
df[df['sex'] == 'F']['score'].mean()
```

```
Out[26]: 4.0
```

```
In [29]: ##### **<> 결측치 제거하지 않고 분석하기**  
df[df['sex'] == 'M']['score'].mean()
```

```
Out[29]: 4.5
```

```
In [104...]: ##### **<> 결측치 제거하지 않고 분석하기**  
df.groupby('sex').agg(mean_score = ('score', 'mean'))
```

```
Out[104]: mean_score
```

sex	mean_score
F	4.0
M	4.5

```
In [ ]:
```

## 결측치 대체하기

> 결측치를 원하는 값으로 일괄 대체

> `.fillna()`

```
In [20]: ## 결측치 만들기  
import pandas as pd  
import numpy as np  
df = pd.DataFrame({'sex' : ['M', 'F', np.nan, 'M', 'F'],  
                  'score' : [5, 4, 3, 4, np.nan]})  
df
```

Out[20]:

	sex	score
0	M	5.0
1	F	4.0
2	NaN	3.0
3	M	4.0
4	F	NaN

In [21]:

```
## 결측치 0으로 치환
df['score'] = df['score'].fillna(0) #결측치를 0으로 대체
df
```

Out[21]:

	sex	score
0	M	5.0
1	F	4.0
2	NaN	3.0
3	M	4.0
4	F	0.0

In [18]:

```
## 결측치 'None'으로 치환
df['sex'].fillna(value = 'None', inplace = True) # df 자체 결측치를 'None'로 대체
df
```

Out[18]:

	sex	score
0	M	5.0
1	F	4.0
2	None	3.0
3	M	4.0
4	F	0.0

```
In [22]: ## 외부 데이터 불러오기  
exam = pd.read_csv('exam_csv.csv')  
exam
```

```
# 데이터 불러오기
```

```
Out[22]:
```

	<b>id</b>	<b>nclass</b>	<b>math</b>	<b>english</b>	<b>science</b>
<b>0</b>	1	1	50	98	50
<b>1</b>	2	1	60	97	60
<b>2</b>	3	1	45	86	78
<b>3</b>	4	1	30	98	58
<b>4</b>	5	2	25	80	65
<b>5</b>	6	2	50	89	98
<b>6</b>	7	2	80	90	45
<b>7</b>	8	2	90	78	25
<b>8</b>	9	3	20	98	15
<b>9</b>	10	3	50	98	45
<b>10</b>	11	3	65	65	65
<b>11</b>	12	3	45	85	32
<b>12</b>	13	4	46	98	65
<b>13</b>	14	4	48	87	12
<b>14</b>	15	4	75	56	78
<b>15</b>	16	4	58	98	65
<b>16</b>	17	5	65	68	98
<b>17</b>	18	5	80	78	90
<b>18</b>	19	5	89	68	87
<b>19</b>	20	5	78	83	58

In [23]: `## 임의의 위치 값을 결측치로 대체하기`

```
exam.loc[[2, 7, 14], ['math']] = np.nan # 2, 7, 14행의 math에 NaN 할당  
exam
```

Out[23]:

	<b>id</b>	<b>nclass</b>	<b>math</b>	<b>english</b>	<b>science</b>
<b>0</b>	1	1	50.0	98	50
<b>1</b>	2	1	60.0	97	60
<b>2</b>	3	1	NaN	86	78
<b>3</b>	4	1	30.0	98	58
<b>4</b>	5	2	25.0	80	65
<b>5</b>	6	2	50.0	89	98
<b>6</b>	7	2	80.0	90	45
<b>7</b>	8	2	NaN	78	25
<b>8</b>	9	3	20.0	98	15
<b>9</b>	10	3	50.0	98	45
<b>10</b>	11	3	65.0	65	65
<b>11</b>	12	3	45.0	85	32
<b>12</b>	13	4	46.0	98	65
<b>13</b>	14	4	48.0	87	12
<b>14</b>	15	4	NaN	56	78
<b>15</b>	16	4	58.0	98	65
<b>16</b>	17	5	65.0	68	98
<b>17</b>	18	5	80.0	78	90
<b>18</b>	19	5	89.0	68	87
<b>19</b>	20	5	78.0	83	58

```
In [24]: ## 'math'의 NaN 값을 평균값으로 일괄 대체하기
avg = exam['math'].mean() #열 평균값 구하기
exam['math'] = exam['math'].fillna(avg) # math가 NaN이면 avg로 대체
exam
```

```
Out[24]:
```

	<b>id</b>	<b>nclass</b>	<b>math</b>	<b>english</b>	<b>science</b>
<b>0</b>	1	1	50.000000	98	50
<b>1</b>	2	1	60.000000	97	60
<b>2</b>	3	1	55.235294	86	78
<b>3</b>	4	1	30.000000	98	58
<b>4</b>	5	2	25.000000	80	65
<b>5</b>	6	2	50.000000	89	98
<b>6</b>	7	2	80.000000	90	45
<b>7</b>	8	2	55.235294	78	25
<b>8</b>	9	3	20.000000	98	15
<b>9</b>	10	3	50.000000	98	45
<b>10</b>	11	3	65.000000	65	65
<b>11</b>	12	3	45.000000	85	32
<b>12</b>	13	4	46.000000	98	65
<b>13</b>	14	4	48.000000	87	12
<b>14</b>	15	4	55.235294	56	78
<b>15</b>	16	4	58.000000	98	65
<b>16</b>	17	5	65.000000	68	98
<b>17</b>	18	5	80.000000	78	90
<b>18</b>	19	5	89.000000	68	87
<b>19</b>	20	5	78.000000	83	58

```
In [25]: ## 결측치 빈도 확인하기  
exam['math'].isna().sum()
```

```
Out[25]: 0
```

```
In [ ]:
```

## Null('NaN'인 값을 다른 값으로 대체하기: .fillna(), .isnull()

```
In [26]: ## Null('NaN'인 값을 다른 값으로 대체하기: .fillna()  
import pandas as pd  
import numpy as np  
  
data = {'column1': [1, 2, np.nan, 4, 5],  
        'column2': [np.nan, 2, 3, 4, np.nan]}  
  
df = pd.DataFrame(data)  
  
# NaN 값을 0으로 대체  
df_filled = df.fillna(0)  
df_filled
```

```
Out[26]:   column1  column2
```

	column1	column2
0	1.0	0.0
1	2.0	2.0
2	0.0	3.0
3	4.0	4.0
4	5.0	0.0

```
In [27]: ## Null('NaN'인 값을 다른 값으로 대체하기: pd.isnull로 확인 후 변경  
import pandas as pd  
  
data = {'column1': [1, 2, np.nan, 4, 5],  
        'column2': [np.nan, 2, 3, 4, np.nan]}  
  
df = pd.DataFrame(data)
```

```
# NaN 값을 0으로 대체
for index, row in df.iterrows():
    if pd.isnull(df.at[index, 'column1']):
        df.at[index, 'column1'] = 0
df
```

Out[27]:

	column1	column2
0	1.0	NaN
1	2.0	2.0
2	0.0	3.0
3	4.0	4.0
4	5.0	NaN

In [28]: ## Null('NaN'인 값을 다른 값으로 대체하기: pd.isnull로 확인 후 변경

```
# NaN 값을 0으로 대체
import numpy as np    #수치연산 지원 패키지 numpy
df['column2'] = np.where(df['column2'].isna(), 0, df['column2'])
df
```

Out[28]:

	column1	column2
0	1.0	0.0
1	2.0	2.0
2	0.0	3.0
3	4.0	4.0
4	5.0	0.0

In [ ]:

## 07-2 이상치 정제하기

> 이상치(anomaly)는 정상 범위에서 벗어난 이상한 값

```
In [28]: ##### 이상치(anomaly) 데이터 생성  
df = pd.DataFrame({'sex' : [1, 2, 1, 3, 2, 1],  
                   'score' : [5, 4, 3, 4, 2, 6]})  
df
```

Out[28]:

	sex	score
0	1	5
1	2	4
2	1	3
3	3	4
4	2	2
5	1	6

## 이상치 확인하기

```
In [46]: ##### > 이상치(anomaly)  
df['sex'].value_counts().sort_index() #빈도순 정렬(True)
```

Out[46]:

```
sex  
1    3  
2    2  
3    1  
Name: count, dtype: int64
```

## 이상치 대체하기

> Python에서는 NaN 값을 실수형의 한 형태로 처리

```
In [29]: ## sex가 3이면 NaN 부여  
import numpy as np  
df['sex'] = np.where(df['sex'] == 3, np.nan, df['sex']) # np.where(Expression, True, False) # score가 5보다 크면 NaN 부여  
df['score'] = np.where(df['score'] > 5, np.nan, df['score'])  
df
```

Out[29]:

	sex	score
0	1.0	5.0
1	2.0	4.0
2	1.0	3.0
3	NaN	4.0
4	2.0	2.0
5	1.0	NaN

## [연습] 결측치를 제외하고 통계 구하기

> 'sex'별 'score' 평균에 대한 데이터 프레임을 생성

In [30]:

```
# sex, score 결측치 제거
# sex별 분리
# score 평균 구하기
df.dropna(subset = ['sex', 'score']) # 
    .groupby('sex') #
        .agg(mean_score = ('score', 'mean'))
```

Out[30]:

sex	mean_score
1.0	4.0
2.0	3.0

In [32]:

```
# sex, score 결측치 제거
# sex별 분리
# score 평균 구하기
df.groupby('sex') #
    .agg(mean_score = ('score', 'mean'))
```

Out[32]: mean\_score

sex
1.0 4.0
2.0 3.0

---

(알아 두면 좋아요) np.where()는 문자와 NaN을 함께 반환할 수 없습니다.

> 왜냐하면, NaN은 실수형으로 관리되기 때문임.

In [31]:

```
df = pd.DataFrame({'x1' : [1, 1, 2, 2]})  
df['x2'] = np.where(df['x1'] == 1, 10, np.nan) # 조건에 맞으면 숫자 부여  
df
```

Out[31]:

	x1	x2
0	1	10.0
1	1	10.0
2	2	NaN
3	2	NaN

In [32]:

```
df = pd.DataFrame({'x1' : [1, 1, 2, 2]})  
df['x2'] = np.where(df['x1'] == 1, 'a', np.nan) # np.nan은 float형으로 취급하기 때문에 str형과 함께 반환 불가  
df
```

Out[32]:

	x1	x2
0	1	a
1	1	a
2	2	nan
3	2	nan

```
In [37]: ## 문자열 변수에 np.nan을 대입하면 'nan' 문자열 값으로 대입됨  
df['x2'].isna()
```

```
Out[37]: 0    False  
1    False  
2    False  
3    False  
Name: x2, dtype: bool
```

```
In [38]: ## 문자열 변수에 np.nan을 대입하면 'nan' 문자열 값으로 대입됨  
type(df.loc[2, 'x2'])
```

```
Out[38]: str
```

```
In [39]: # 'nan'을 NaN으로 바꾸기  
df['x2'] = df['x2'].replace('nan', np.nan)  
df
```

```
Out[39]:   x1    x2  
0    1     a  
1    1     a  
2    2    NaN  
3    2    NaN
```

```
In [40]: df['x2'].isna()
```

```
Out[40]: 0    False  
1    False  
2    True  
3    True  
Name: x2, dtype: bool
```

```
In [ ]:
```

## boxplot을 활용하여 극단적인 값을 확인하기

seaborn 라이브러리의 boxplot을 활용하여 극단적인 값을 시각화함

상자 그림을 참고로 극단치 기준을 정함

## 1. 상자 그림 살펴보기

```
In [33]: ## 'mpg.csv'의 데이터 프레임 구축
mpg = pd.read_csv('mpg.csv')
mpg
```

```
Out[33]:
```

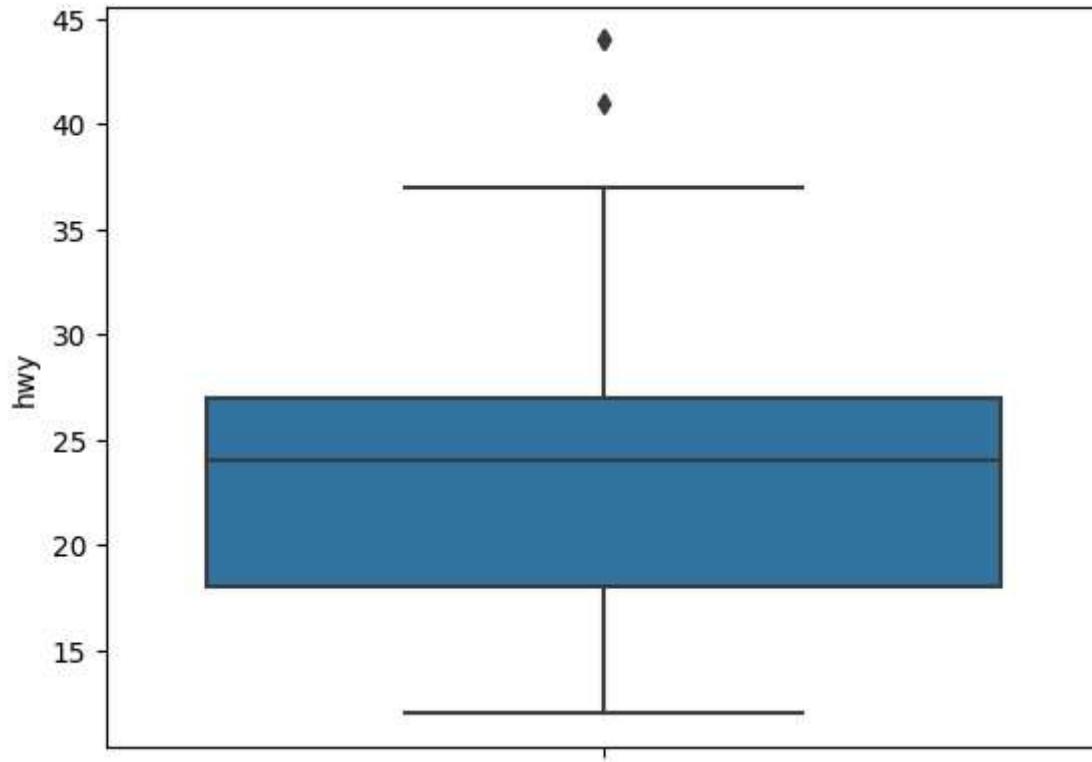
	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...	...	...	...	...	...	...	...	...	...	...	...
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 [34]: ## mpg의 boxplot 그래프 그리기
import seaborn as sns
sns.boxplot(data = mpg, y = 'hwy')
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0 is required for
this version of SciPy (detected version 1.25.2)
... warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}""
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[34]: <Axes: ylabel='hwy'>



## boxplot 해석

- > Outer 극단치
- > Maximum 극단치 경계 최대값
- > Upper Quartile 상위 25%(Q3 : 3사분위)
- > Median 중위(Q2 : 2사분위)
- > LowerR 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>

<https://search.pstatic.net/sunny/>

src=https%3A%2F%2Fi.stack.imgur.com%2FGqrkS.png&type=sc960\_832

## 2. 극단치 기준값 구하기

### (1) 1사분위수, 3사분위수 구하기

```
In [61]: ## 1사분위수 구하기  
pct25 = mpg['hwy'].quantile(.25)  
pct25
```

```
Out[61]: 18.0
```

```
In [62]: ## 3사분위수 구하기  
pct75 = mpg['hwy'].quantile(.75)  
pct75
```

```
Out[62]: 27.0
```

### (2) IQR 구하기

> IQR(사분위 범위)는 1사분위수와 3사분위수의 거리

```
In [40]: ## IQR 구하기  
iqr = pct75 - pct25  
iqr
```

```
Out[40]: 9.0
```

### (3) 하한, 상한 구하기

> 하한 Minimum : Q1 - 1.5\*IQR

> 하한 Maximum : Q3 + 1.5\*IQR

```
In [41]: pct25 - 1.5 * iqr # 하한
```

```
Out[41]: 4.5
```

```
In [42]: pct75 + 1.5 * iqr # 상한
```

Out[42]: 40.5

### 3. 극단치를 결측 처리하기

In [63]: # 4.5 ~ 40.5 벗어나면 NaN 부여

```
import numpy as np
mpg['hwy'] = np.where((mpg['hwy'] < 4.5) | (mpg['hwy'] > 40.5), np.nan, mpg['hwy'])

# 결측치 빈도 확인
mpg['hwy'].isna().sum()
```

Out[63]: 3

In [44]: ## 결측치 행 확인: 'hwy' 열의 값이 NaN인 경우만 데이터 프레임 구성

```
mpg[mpg['hwy'].isna()] # 'hwy' 열의 값이 NaN인 경우만 데이터 프레임 구성
```

Out[44]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	category
212	volkswagen	jetta	1.9	1999	4	manual(m5)	f	33	Nan	d	compact
221	volkswagen	new beetle	1.9	1999	4	manual(m5)	f	35	Nan	d	subcompact
222	volkswagen	new beetle	1.9	1999	4	auto(l4)	f	29	Nan	d	subcompact

In [45]: ## 결측치 제거하지 않고 분석하기

```
mpg.groupby('drv')  
.agg(mean_hwy = ('hwy', 'mean'))
```

Out[45]: mean\_hwy

drv	mean_hwy
4	19.174757
f	27.728155
r	21.000000

### 4. 결측치 제거하고 분석하기

> .mean(), .sum(), .groupby()는 결측치를 제외하고 연산을 수행

```
In [46]: ## 결측치 제거하고 분석하기
# hwy 결측치 제거
# drv별 분리
# hwy 평균 구하기
mpg.dropna(subset = ['hwy']) %>
  .groupby('drv') %>
  .agg(mean_hwy = ('hwy', 'mean'))
```

```
Out[46]: mean_hwy
```

```
drv
_____
4    19.174757
f    27.728155
r    21.000000
```

```
In [ ]:
```

## [실습 7-1] 결측치를 제외하고 통계 구하기

1. 'mpg.csv' 파일을 읽어서 데이터 프레임을 구성하기
2. 데이터 프레임의 'cty' 열에 대한 boxplot 그래프 그리기
3. 하한값, 상한값 구하기
4. 극단치를 결측 처리하기  
  > 결측 처리 결과 'cty' 값이 NaN인 행 검색
5. 'cty'의 결측치 제거하고 분석하기  
  > model별 cty 평균 구하기

```
In [85]: ##### 1. 'mpg.csv'의 데이터 프레임 구축
mpg = pd.read_csv('mpg.csv')
mpg
```

Out[85]:

	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 [ ]:

## [실습 7-2] mpg 데이터 가공하기

- 'mpg.csv' 파일을 읽어서 데이터 프레임을 구성하기
- mpg 데이터 프레임에서 'trans' 열의 값 빈도 분포를 확인
- 'auto' 파생변수 생성
  - > 'trans' 열 값에 'auto'가 있으면 1, 'manual'이 있으면 0으로 처리, 그 외는 NaN 처리
- 'gear' 파생변수 생성
  - > 'trans' 열 값의 ')' 앞에 값으로 판단, 숫자가 아니면 NaN 처리
- 'auto'와 'gear' 열의 결측치(NaN) 확인
  - > 결측치 빈도 확인

> 결측치 행 확인

6. 'auto'와 'gear'별 분포 수 데이터 프레임 생성

> 단, 결측치(NaN) 행은 사전에 차단

In [ ]:

## 정리하기

```
In [ ]: ## 1. 결측치 정제하기
pd.isna(df).sum() # 결측치 확인
df_nomiss = df.dropna(subset = ['score']) # 결측치 제거
df_nomiss = df.dropna(subset = ['score', 'sex']) # 여러 변수 동시에 결측치 제거

## 2. 이상치 정제하기

# 이상치 확인
df['sex'].value_counts(sort = False)

# 이상치 결측 처리
df['sex'] = np.where(df['sex'] == 3, np.nan, df['sex'])

# 상자 그림으로 극단치 기준값 찾기
pct25 = mpg['hwy'].quantile(.25) # 1사분위수
pct75 = mpg['hwy'].quantile(.75) # 3사분위수
iqr = pct75 - pct25 # IQR
pct25 - 1.5 * iqr # 하한
pct75 + 1.5 * iqr # 상한

# 극단치 결측 처리
mpg['hwy'] = np.where((mpg['hwy'] < 4.5) | (mpg['hwy'] > 40.5), np.nan, mpg['hwy'])
```