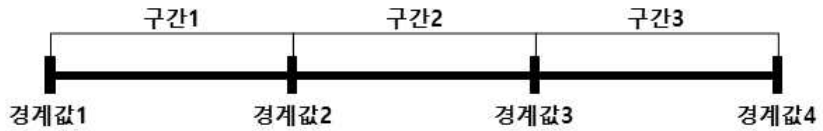
04. 데이터프레임의 응용

1. 범주형 데이터 처리

○ 구간 분할

- 데이터 분석 방법에 따라서 연속 데이터를 그대로 사용하기 보다는 일정한 구간으로 나눠서 분석하는 것이 효율적인 경우가 존재함.
- 즉, 연속적인 값을 일정한 수준이나 정도를 나타내는 이산적인 값으로 나타내어 구간별 차이를 드러내는 것임.
- 이러한 연속 변수를 일정한 구간으로 나누고, 각 구간을 범주형 변수로 변환하는 과정을 구간분할(binning) 이라고 함.
- 연속형 데이터를 여러 구간으로 범주형 데이터로 변환할 수 있는 명령어인 cut() 함수를 사용하여 구간분할을 할 수 있음.
- cut() 함수를 사용하는 방법은 다음과 같음.

```
pandas.cut(데이터배열, bins=경계값 리스트, labels=구간 이름,
include_lowest=False/True)
```



- x 옵션에 구간분할을 하고자 하는 연속형 변수를 입력하고, bins 옵션에 구간의 경계값을 입력함. 그림과 같이 만약 나누고자 하는 구간이 3개라면 경계값은 4개의 값이 존재하므로, 4개의 경계값을 입력해야 함. 여기서 구간의 첫 경계값을 포함하고자 한다면 include_lowest 옵션에 True를 입력함.
- 또한, 만약 구간의 이름을 입력하고자 한다면 labels 옵션에 구간의 이름을 입력하면 됨.
- Seaborn 모듈의 내부데이터인 mpg 데이터셋을 사용하여 연속형 변수인 horsepower 변수를 구간분할을 통하여 저출력(46~105), 보통출력(105~165), 고출력(165~230)의 값을 갖는 범주형 변수로 변환해보도록 하겠음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: print(df)
      mpg  cylinders  ...  origin                       name
0    18.0          8  ...     usa  chevrolet chevelle malibu
1    15.0          8  ...     usa          buick skylark 320
2    18.0          8  ...     usa         plymouth satellite
3    16.0          8  ...     usa              amc rebel sst
4    17.0          8  ...     usa                ford torino
..    ...        ...  ...     ...                        ...
393  27.0          4  ...     usa            ford mustang gl
394  44.0          4  ...  europe                   vw pickup
395  32.0          4  ...     usa               dodge rampage
396  28.0          4  ...     usa                 ford ranger
397  31.0          4  ...     usa                  chevy s-10

[398 rows x 9 columns]
```

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')
   ...: print(df['horsepower'])
0      130.0
1      165.0
2      150.0
3      150.0
4      140.0
       ...
393     86.0
394     52.0
395     84.0
396     79.0
397     82.0
Name: horsepower, Length: 398, dtype: float64
```

```
In [3]: import pandas as pd

In [4]: df['hp_bin'] = pd.cut(df['horsepower'],
   ...:                       bins=[46,105,165,230],
   ...:                       labels=['저출력','보통출력','고출력'],
   ...:                       include_lowest=True)

In [5]: print(df['hp_bin'])
0      보통출력
1      보통출력
2      보통출력
3      보통출력
4      보통출력
       ...
393     저출력
394     저출력
395     저출력
396     저출력
397     저출력
Name: hp_bin, Length: 398, dtype: category
Categories (3, object): ['저출력' < '보통출력' < '고출력']
```

- 먼저 mpg 데이터셋의 horsepower 변수를 구간분할을 실시하므로 데이터배열에 df['horsepower']를 입력하고, 구간의 경계값인 46, 105, 165, 230을 bins 옵션에 리스트의 형태로 입력함. 그리고 구간의 이름이 저출력, 보통출력, 고출력이므로 labels 옵션에 이를 리스트의 형태로 입력합니다. 마지막으로 구간의 첫 경계값을 포함하기 위해 include_lowest 옵션에 True를 입력함.
- 실행 결과를 살펴보면, 연속형이었던 값이 저출력, 보통출력, 고출력으로 변환된 것을 확인할 수 있음.
- 만약 분할하고자 하는 구간의 개수만 알고 있고, 경계값을 알수 없을 때, 구간의 경계값을 구하는 방법 중에 numpy 모듈의 histogram() 함수를 활용할 수 있음.
- histogram() 함수는 구간에 속하는 데이터의 개수와 경계값 리스트를 차례로 반환해주는 함수입니다. 단, histogram() 함수는 누락 데이터가 존재하지 않을때만 사용할 수 있음.
- histogram() 함수를 사용하는 방법은 다음과 같음.

numpy.histogram(데이터배열, bins=구간의 수)

- Seaborn 모듈의 내부데이터인 mpg 데이터셋을 사용하여 연속형 변수인 horsepower 변수를 3개의 구간으로 분할하기 위해 histogram() 함수를 이용하여 구간의 경계값을 확인해보도록 하겠음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')
   ...: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    object
 8   name          398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

- mpg 데이터셋의 horsepower 변수를 info() 함수를 통해 살펴보면 누락 데이터가 존재하는 것을 알 수 있음. histogram() 함수는 누락 데이터가 존재할 때 사용할 수 없으므로 누락 데이터를 제거할 필요가 있음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')
   ...: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    object
 8   name          398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

```
In [3]: df.dropna(subset=['horsepower'], axis=0, inplace=True)

In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64
 5   acceleration  392 non-null    float64
 6   model_year    392 non-null    int64
 7   origin        392 non-null    object
 8   name          392 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 30.6+ KB
```

- 지난 차시에서 살펴보았던 dropna() 함수를 이용하여 누락데이터를 제거할 수 있음.

```
In [5]: import numpy as np

In [6]: np.histogram(df['horsepower'], bins=3)
Out[6]:
(array([257, 103,  32], dtype=int64),
 array([ 46.        , 107.33333333, 168.66666667, 230.        ]))

In [7]: count, bin_dividers = np.histogram(df['horsepower'], bins=3)

In [8]: print(bin_dividers)
[ 46.         107.33333333 168.66666667 230.        ]
```

- histogram() 함수의 데이터배열에 df[‘horsepower’]를 입력하고, 나누고자 하는 구간의 수인 3을 bins 옵션에 입력함. 그 결과 출력된 배열을 살펴보면 먼저 처음에 구간에 해당하는 데이터의 수를 확인할 수 있고, 다음으로 경계값 리스트를 확인할 수 있습니다. 이 결과들을 각각 count와 bin_dividers 객체에 할당하여 활용할 수 있음.

```
In [9]: import pandas as pd

In [10]: df['hp_bin'] = pd.cut(df['horsepower'],
    ...:                       bins=bin_dividers,
    ...:                       labels=['저출력', '보통출력', '고출력'],
    ...:                       include_lowest=True)

In [11]: print(df['hp_bin'])
0      보통출력
1      보통출력
2      보통출력
3      보통출력
4      보통출력
       ...
393     저출력
394     저출력
395     저출력
396     저출력
397     저출력
Name: hp_bin, Length: 392, dtype: category
Categories (3, object): ['저출력' < '보통출력' < '고출력']
```

- 할당된 경계값 리스트인 bin_dividers 객체를 cut 함수의 bins 옵션에 입력고, 구간의 이름을 저출력, 보통출력, 고출력으로 labels 옵션에 이를 리스트의 형태로 입력함. 마지막으로 구간의 첫 경계값을 포함하기 위해 include_lowest 옵션에 True를 입력함.
- 실행 결과를 살펴보면, 연속형이었던 값이 저출력, 보통출력, 고출력으로 변환된 것을 확인할 수 있음.

○ 더미 변수

- 범주형 변수 데이터는 회귀분석 등과 같은 머신러닝 알고리즘에 바로 사용할 수 없어 이를 컴퓨터가 인식 가능한 입력값으로 변환해야 함. 이때 우리는 0과 1로 표현되는 더미변수(dummy variable)를 사용함.
- 더미변수에서 0과 1은 크고 작음을 나타내는 것이 아니라 어떤 특성이

있는지 없는지 여부를 표시함. 즉, 해당 특성이 존재하면 1로 표현하고, 그렇지 않으면 0으로 구분함.

- 더미변수로 변환하는 방법에는 원핫인코딩 방법과 축소랭크 방법이 존재함.

ID
A
B
C
A

One-Hot Encoding

A	B	C
1	0	0
0	1	0
0	0	1
1	0	0

Reduced-Rank

A	B	C
1	0	0
1	1	0
1	0	1
1	0	0

Ref.

- 원핫인코딩방법은 범주값을 모두 변수로 생성한 후, 범주값에 해당되는 특징에 1, 그렇지 않은 경우에는 0을 부여하는 방법임.
- 축소랭크 방법은 특정 범주값을 기준값(reference)로 설정하고, 기준값을 제외하고 해당하는 값에 맞으면 1, 그렇지 않으면 0을 부여하는 방법임. 회귀분석에서는 이를 Reference coding이라고도 함.
- 범주형 변수 데이터를 더미변수로 변환할 수 있는 명령어인 get_dummies() 함수를 사용하여 더미변수화 할 수 있음.
- get_dummies() 함수를 사용하는 방법은 다음과 같음.

pandas.get_dummies(데이터배열, drop_first=False/True)

- 데이터배열에 더미변수로 변환하고자 하는 범주형 변수를 입력하고, drop_first 옵션에 False와 True를 통해 원핫인코딩방법과 축소랭크 방법을 설정함. 즉, drop_first 옵션에 False(기본값)를 입력하면 원핫인코딩방법으로 더미변수를 생성하고, True를 입력하면 축소랭크 방법으로 더미변수를 생성함.
- 앞서 생성한 mpg 데이터셋의 범주형 변수로 변환한 horsepower 데이터를 이용하여 더미 변수로 변환해보도록 하겠음.

```
In [11]: print(df['hp_bin'])
0      보통출력
1      보통출력
2      보통출력
3      보통출력
4      보통출력
       ...
393     저출력
394     저출력
395     저출력
396     저출력
397     저출력
Name: hp_bin, Length: 392, dtype: category
Categories (3, object): ['저출력' < '보통출력' < '고출력']
```

```
In [12]: import pandas as pd

In [13]: horsepower_dummies = pd.get_dummies(df['hp_bin'])

In [14]: print(horsepower_dummies)
     저출력  보통출력  고출력
0      0     1    0
1      0     1    0
2      0     1    0
3      0     1    0
4      0     1    0
..   ...   ...  ...
393    1     0    0
394    1     0    0
395    1     0    0
396    1     0    0
397    1     0    0

[392 rows x 3 columns]
```

- 먼저 mpg 데이터셋의 범주형 변수로 변환시킨 hp_bin 변수를 더미변수로 변환시키므로 데이터배열에 df['hp_bin']을 입력하고, 원핫인코딩 방법으로 더미변수를 생성하기 위해 drop_first 옵션에 False를 입력하거나 생략함.
- 실행 결과를 살펴보면, 하나의 범주형 변수였던 hp_bin 변수가 저출력, 보통출력, 고출력의 3개의 변수로 생성되고 범주값에 해당하면 1, 그렇지 않으면 0으로 각각 입력되어 있는 것을 확인할 수 있음.

```
In [11]: print(df['hp_bin'])
0      보통출력
1      보통출력
2      보통출력
3      보통출력
4      보통출력
       ...
393     저출력
394     저출력
395     저출력
396     저출력
397     저출력
Name: hp_bin, Length: 392, dtype: category
Categories (3, object): ['저출력' < '보통출력' < '고출력']
```

```
In [15]: import pandas as pd

In [16]: horsepower_dummies1 = pd.get_dummies(df['hp_bin'],drop_first=True)

In [17]: print(horsepower_dummies1)
     보통출력  고출력
0       1    0
1       1    0
2       1    0
3       1    0
4       1    0
..    ...  ...
393     0    0
394     0    0
395     0    0
396     0    0
397     0    0

[392 rows x 2 columns]
```

- 그리고 축소랭크 방법으로 더미변수를 생성하기 위해 drop_first 옵션에 True를 입력함.
- 실행 결과를 살펴보면, 하나의 범주형 변수였던 hp_bin 변수가 보통출력, 고출력의 2개의 변수로 생성되고 범주값에 해당하면 1, 그렇지 않으면 0으로 각각 입력되어 있는 것을 확인할 수 있음. 여기서 저출력은 기준값으로 설정된 것을 확인할 수 있음.

2. 열 재구성

○ 열 순서 바꾸기

- 데이터프레임의 변수의 순서를 분석자가 사용하기 편하게 변경할 수 있음.
- 열 이름을 원하는 순서대로 정리해서 리스트를 만들고 데이터프레임에서 열을 다시 선택하는 방식으로 열 순서를 바꿀 수 있음.
- 열 순서를 바꾸는 방법은 다음과 같음.

객체[재구성한 열 이름의 리스트]

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: df.columns
Out[3]:
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'name'],
      dtype='object')

In [4]: list(df.columns)
Out[4]:
['mpg',
 'cylinders',
 'displacement',
 'horsepower',
 'weight',
 'acceleration',
 'model_year',
 'origin',
 'name']
```

- 데이터프레임의 변수명을 알기 위해서 columns 명령어를 사용하여 확인할 수 있음. 여기에 list() 함수를 사용하여 데이터프레임의 변수명을 리스트 형태로 변환할 수 있음.
- 확인된 데이터프레임에 존재하는 변수명을 토대로 사용자가 원하는 순서대로 열 순서를 설정할 수 있음.
- Seaborn 모듈의 내부데이터인 mpg 데이터셋을 사용하여 mpg, name, horsepower, cylinders, model_year순서로 데이터셋을 새롭게 구성하고자 함.

```
In [5]: df1 = df[['mpg', 'name', 'horsepower', 'cylinders', 'model_year']]

In [6]: print(df1)
      mpg                       name  horsepower  cylinders  model_year
0    18.0  chevrolet chevelle malibu       130.0          8          70
1    15.0          buick skylark 320       165.0          8          70
2    18.0         plymouth satellite       150.0          8          70
3    16.0              amc rebel sst       150.0          8          70
4    17.0                ford torino       140.0          8          70
..    ...                        ...         ...        ...         ...
393  27.0            ford mustang gl        86.0          4          82
394  44.0                    vw pickup       52.0          4          82
395  32.0               dodge rampage        84.0          4          82
396  28.0                 ford ranger        79.0          4          82
397  31.0                  chevy s-10        82.0          4          82

[398 rows x 5 columns]
```

- mpg 데이터셋에서 새롭게 구성하고자 하는 변수명 mpg, name, horsepower, cylinders, model_year를 순서대로 리스트로 구성하여 재구성 열 이름 리스트에 대입하면 원하는 순서대로 데이터셋이 새롭게 구성된 것을 확인할 수 있음.
- 그리고 sorted() 함수를 통해 변수 이름을 알파벳 순서로 오름차순으로 정렬할 수 있음.

sorted(리스트, reverse=False/True)

- 만약 reverse 옵션에 True를 입력하면 내림차순으로 정렬할 수 있음.
- 이를 열 순서를 바꾸는 방법을 활용하면 데이터프레임을 변수명의 오름차순 혹은 내림차순으로 정렬할 수 있음.
- Seaborn 모듈의 내부데이터인 mpg 데이터셋을 사용하여 변수 배열 순서를 알파벳 내림차순으로 데이터셋을 새롭게 구성하고자 함.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: list(df.columns)
Out[3]:
['mpg',
 'cylinders',
 'displacement',
 'horsepower',
 'weight',
 'acceleration',
 'model_year',
 'origin',
 'name']

In [4]: list1 = sorted(list(df.columns))

In [5]: df[list1]
Out[5]:
     acceleration  cylinders  ...  origin  weight
0            12.0          8  ...     usa    3504
1            11.5          8  ...     usa    3693
2            11.0          8  ...     usa    3436
3            12.0          8  ...     usa    3433
4            10.5          8  ...     usa    3449
..            ...        ...  ...     ...     ...
393          15.6          4  ...     usa    2790
394          24.6          4  ...  europe    2130
395          11.6          4  ...     usa    2295
396          18.6          4  ...     usa    2625
397          19.4          4  ...     usa    2720

[398 rows x 9 columns]
```

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: list(df.columns)
Out[3]:
['mpg',
 'cylinders',
 'displacement',
 'horsepower',
 'weight',
 'acceleration',
 'model_year',
 'origin',
 'name']

In [4]: list2 = sorted(list(df.columns), reverse=True)

In [5]: df[list2]
Out[5]:
     weight  origin  ... cylinders  acceleration
0      3504     usa  ...         8          12.0
1      3693     usa  ...         8          11.5
2      3436     usa  ...         8          11.0
3      3433     usa  ...         8          12.0
4      3449     usa  ...         8          10.5
..      ...     ...  ...       ...           ...
393    2790     usa  ...         4          15.6
394    2130  europe  ...         4          24.6
395    2295     usa  ...         4          11.6
396    2625     usa  ...         4          18.6
397    2720     usa  ...         4          19.4

[398 rows x 9 columns]
```

- mpg 데이터셋의 변수명을 확인하기 위해 columns 명령어를 이용하여 변수명 리스트를 출력하고, 이를 sorted() 함수에 적용함. 여기서 내림차순으로 정렬하기 위해 reverse 옵션에 True를 입력함.
- 이렇게 정렬된 변수명을 재구성 열 이름 리스트에 대입하면 변수 명이 내림차순으로 데이터셋이 새롭게 구성된 것을 확인할 수 있음.
- 만약 변수명을 오름차순으로 정렬하고자 한다면 reverse 옵션을 입력하지 않으면 됨.

○ 열 분리

- 하나의 열이 여러 가지 정보를 담고 있을 때 각 정보를 서로 분리해서 사용해야 하는 경우가 존재함.
- 예를 들면, 어떠한 변수에 연월일 정보가 모두 들어있을 때, 이를 연, 월, 일을 구분하여 3개의 열을 만드는 것이 필요할 수 있음.

	A	B	C	D	E
1	연월일	시가	고가	저가	
2	2021-08-01	10850	10900	10000	
3	2021-08-02	10550	10900	9990	
4	2021-08-03	10900	10950	10150	
5	2021-08-04	10800	11050	10500	
6	2021-08-05	10900	11000	10700	
7	2021-08-06	11400	11450	11000	
8	2021-08-07	11250	11450	10750	
9	2021-08-08	11350	11750	11200	
10	2021-08-09	11200	11600	10900	
11	2021-08-10	11850	11950	11300	
12	2021-08-11	13400	13400	12000	
13	2021-08-12	13600	13600	12900	
14	2021-08-13	13200	13700	13150	
15					

```
In [1]: import pandas as pd

In [2]: df = pd.read_excel('C:/Users/LeeKJ/Desktop/주가데이터

In [3]: print(df)
         연월일     시가     고가     저가
0  2021-08-01  10850  10900  10000
1  2021-08-02  10550  10900   9990
2  2021-08-03  10900  10950  10150
3  2021-08-04  10800  11050  10500
4  2021-08-05  10900  11000  10700
5  2021-08-06  11400  11450  11000
6  2021-08-07  11250  11450  10750
7  2021-08-08  11350  11750  11200
8  2021-08-09  11200  11600  10900
9  2021-08-10  11850  11950  11300
10 2021-08-11  13400  13400  12000
11 2021-08-12  13600  13600  12900
12 2021-08-13  13200  13700  13150
```

- 연월일 변수의 자료를 살펴보면 연월일이 - 기호를 사용하여 구분되어 있는 것을 확인할 수 있음. 이러한 정보를 바탕으로 연월일 변수를 연,

월, 일 변수로 구분해보도록 하겠음.

- 이를 위해 아래와 같이 총 3단계의 작업이 필요함.

1) 구분할변수.astype('str')
2) 문자형구분할변수.str.split(구분기호)
3) 구분된 시리즈자료.str.get(순서번호)

- 먼저 구분하고자 하는 변수의 데이터를 문자형으로 변경하기 위해 astype() 명령어를 사용합니다. 만약 자료가 미 문자형 데이터인 경우는 이를 생략할 수 있음.
- 그럼 후 구분하고자 하는 기호를 str.split() 명령어를 이용하여 자료를 구분함.
- 구분된 시리즈자료를 str.get() 명령어를 이용하여 순서번호를 지정하여 자료를 가져옴. 여기서 순서번호는 0부터 시작함.
- 열 분리하는 방법을 다음과 같이 한 번에 입력할 수 있음.

구분할변수.astype('str').str.split(구분기호).str.get(순서번호)

- 주어진 엑셀파일 주가데이터.xlsx에 존재하는 연월일 변수의 자료를 구분하여 연, 월, 일 변수를 새롭게 구성하고자 함.

```
In [1]: import pandas as pd

In [2]: df = pd.read_excel('C:/Users/LeeKJ/Desktop/주가데이터.xlsx')

In [3]: print(df)
         연월일    시가    고가    저가
0  2021-08-01  10850  10900  10000
1  2021-08-02  10550  10900   9990
2  2021-08-03  10900  10950  10150
3  2021-08-04  10800  11050  10500
4  2021-08-05  10900  11000  10700
5  2021-08-06  11400  11450  11000
6  2021-08-07  11250  11450  10750
7  2021-08-08  11350  11750  11200
8  2021-08-09  11200  11600  10900
9  2021-08-10  11850  11950  11300
10 2021-08-11  13400  13400  12000
11 2021-08-12  13600  13600  12900
12 2021-08-13  13200  13700  13150
```

```
In [4]: df['연월일'].astype('str')
Out[4]:
0     2021-08-01
1     2021-08-02
2     2021-08-03
3     2021-08-04
4     2021-08-05
5     2021-08-06
6     2021-08-07
7     2021-08-08
8     2021-08-09
9     2021-08-10
10    2021-08-11
11    2021-08-12
12    2021-08-13
Name: 연월일, dtype: object
```

```
In [5]: df['연월일'].astype('str').str.split('-')
Out[5]:
0     [2021, 08, 01]
1     [2021, 08, 02]
2     [2021, 08, 03]
3     [2021, 08, 04]
4     [2021, 08, 05]
5     [2021, 08, 06]
6     [2021, 08, 07]
7     [2021, 08, 08]
8     [2021, 08, 09]
9     [2021, 08, 10]
10    [2021, 08, 11]
11    [2021, 08, 12]
12    [2021, 08, 13]
Name: 연월일, dtype: object
```

- 연월일 변수의 자료는 날짜형 자료이기 때문에 자료를 문자형 자료로 변경하고, - 기호로 구분되어 있는 연월일을 구분한 후, 차례로 자료를 가져와서 연, 월, 일 변수를 구성해야 함.

```
In [6]: df['연'] = df['연월일'].astype('str').str.split('-').str.get(0)

In [7]: df['월'] = df['연월일'].astype('str').str.split('-').str.get(1)

In [8]: df['일'] = df['연월일'].astype('str').str.split('-').str.get(2)

In [9]: print(df)
         연월일    시가    고가    저가    연  월  일
0  2021-08-01  10850  10900  10000  2021  08  01
1  2021-08-02  10550  10900   9990  2021  08  02
2  2021-08-03  10900  10950  10150  2021  08  03
3  2021-08-04  10800  11050  10500  2021  08  04
4  2021-08-05  10900  11000  10700  2021  08  05
5  2021-08-06  11400  11450  11000  2021  08  06
6  2021-08-07  11250  11450  10750  2021  08  07
7  2021-08-08  11350  11750  11200  2021  08  08
8  2021-08-09  11200  11600  10900  2021  08  09
9  2021-08-10  11850  11950  11300  2021  08  10
10 2021-08-11  13400  13400  12000  2021  08  11
11 2021-08-12  13600  13600  12900  2021  08  12
12 2021-08-13  13200  13700  13150  2021  08  13
```

- 순서번호 0을 입력하여 연 자료를 추출 후 일 변수 생성, 순서번호 1을 입력하여 월 자료를 추출 후 월 변수 생성, 마지막으로 순서번호 2를 입력하여 일 자료를 추출 후 일 변수를 생성할 수 있음.
- 데이터프레임을 출력하여 연, 월, 일 변수가 데이터프레임의 마지막에 새롭게 생성되어 위치하고 있는 것을 확인할 수 있음.

3. 필터링

○ 불린 인덱싱

- 시리즈 객체에 조건식 (>, <, ==, != 등)을 적용하면 각 원소에 대해 참/거짓을 판별하여 불린(boolean) 값으로 구성된 시리즈를 반환함. 이 때, 참에 해당하는 데이터 값을 따로 선택할 수 있는데 이를 불린 인덱싱(boolean indexing)이라 함.

- 데이터프레임의 불린 인덱싱 방법은 다음과 같음.

객체.loc[불린시리즈, :]

- 즉, 행과 열의 위치를 지정하는 loc 명령어의 행 위치에 불린시리즈를 입력하여 참에 해당하는 데이터의 행 위치를 선택하고, 열 위치에 콜롬(:)을 입력하여 모든 변수를 선택하여 조건에 맞는 데이터프레임을 새롭게 생성할 수 있음.

- Seaborn 모듈의 내부데이터인 mpg 데이터셋을 사용하여 mpg가 20 보다 크고, 30보다 작은 자료로 데이터셋을 새롭게 구성하고자 함.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: print(df['mpg'])
0      18.0
1      15.0
2      18.0
3      16.0
4      17.0
       ...
393    27.0
394    44.0
395    32.0
396    28.0
397    31.0
Name: mpg, Length: 398, dtype: float64
```


```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('mpg')

In [3]: print(df['mpg'])
0      18.0
1      15.0
2      18.0
3      16.0
4      17.0
       ...
393    27.0
394    44.0
395    32.0
396    28.0
397    31.0
Name: mpg, Length: 398, dtype: float64
```


- mpg가 20보다 크고 30보다 작은 조건식을 이용하여 불린값으로 구성된 시리즈를 생성하고, 이를 데이터프레임의 행 위치에 대입하여 조건에 만족하는 행만 남게 되는 데이터프레임을 확인할 수 있음.
- 수정된 데이터프레임을 출력하여 모든 자료가 mpg가 20보다 크고 30보다 작은 것을 확인할 수 있음.

4. 데이터프레임의 응용 실습

※ seaborn 모듈의 penguins 데이터에서, body_mass_g 변수를 구간분할을 통하여 저체중(2700~3800), 보통체중(3800~5000), 고체중(5000~6300)의 값을 갖는 범주형 변수로 변환해보세요.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: print(df['body_mass_g'])
0      3750.0
1      3800.0
2      3250.0
3         NaN
4      3450.0
        ...
339       NaN
340    4850.0
341    5750.0
342    5200.0
343    5400.0
Name: body_mass_g, Length: 344, dtype: float64
```

```
In [4]: import pandas as pd

In [5]: df['bm_bin'] = pd.cut(df['body_mass_g'],
   ...:                       bins=[2700, 3800, 5000, 6300],
   ...:                       labels=['저체중', '보통체중', '고체중'],
   ...:                       include_lowest=True)

In [6]: print(df['bm_bin'])
0       저체중
1       저체중
2       저체중
3       NaN
4       저체중
       ...
339     NaN
340    보통체중
341     고체중
342     고체중
343     고체중
Name: bm_bin, Length: 344, dtype: category
Categories (3, object): ['저체중' < '보통체중' < '고체중']
```

- 먼저 penguins 데이터셋의 body_mass_g 변수를 구간분할을 실시하므로 데이터배열에 df['body_mass_g']를 입력하고, 구간의 경계값인 2700, 3800, 5000, 6300을 bins 옵션에 리스트의 형태로 입력합니다. 그리고 구간의 이름이 저체중, 보통체중, 고체중이므로 labels 옵션에 이를 리스트의 형태로 입력합니다. 마지막으로 구간의 첫 경계값을 포함하기 위해 include_lowest 옵션에 True를 입력합니다.
- 실행 결과를 살펴보면, 연속형이었던 값이 저체중, 보통체중, 고체중으로 변환된 것을 확인할 수 있습니다.

※ penguins 데이터에서 body_mass_g 변수를 3개의 구간으로 분할해 보세요. 단, 구간의 경계값은 주어져 있지 않습니다.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')
   ...: print(df)
    species     island  bill_length_mm  ...  flipper_length_mm  body_mass_g     sex
0    Adelie  Torgersen            39.1  ...              181.0       3750.0    Male
1    Adelie  Torgersen            39.5  ...              186.0       3800.0  Female
2    Adelie  Torgersen            40.3  ...              195.0       3250.0  Female
3    Adelie  Torgersen             NaN  ...                NaN          NaN     NaN
4    Adelie  Torgersen            36.7  ...              193.0       3450.0  Female
..      ...        ...             ...  ...                ...          ...     ...
339  Gentoo     Biscoe             NaN  ...                NaN          NaN     NaN
340  Gentoo     Biscoe            46.8  ...              215.0       4850.0  Female
341  Gentoo     Biscoe            50.4  ...              222.0       5750.0    Male
342  Gentoo     Biscoe            45.2  ...              212.0       5200.0  Female
343  Gentoo     Biscoe            49.9  ...              213.0       5400.0    Male

[344 rows x 7 columns]
```

```
In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

```
In [4]: df.dropna(subset=['body_mass_g'], axis=0, inplace=True)

In [5]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 342 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            342 non-null    object
 1   island             342 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 21.4+ KB
```

- penguins 데이터셋의 body_mass_g 변수를 info() 함수를 통해 살펴보면 누락 데이터가 2개(=344-342) 존재하는 것을 알 수 있습니다. histogram() 함수는 누락 데이터가 존재할 때 사용할 수 없으므로 누락 데이터를 제거할 필요가 있습니다. dropna() 함수를 이용하여 누락데이터를 제거할 수 있습니다.

```
In [6]: import numpy as np

In [7]: np.histogram(df['body_mass_g'], bins=3)
Out[7]: (array([144, 140,  58], dtype=int64), array([2700., 3900., 5100., 6300.]))

In [8]: count, bin_dividers = np.histogram(df['body_mass_g'], bins=3)

In [9]: print(bin_dividers)
[2700. 3900. 5100. 6300.]
```

- histogram() 함수의 데이터배열에 df['body_mass_g']를 입력하고, 나누고자 하는 구간의 수인 3을 bins 옵션에 입력합니다. 그 결과 출력된 배열을 살펴보면 먼저 처음에 구간에 해당하는 데이터의 수를 확인할 수 있고, 다음으로 경계값 리스트인 2700, 3900, 5100, 6300을 확인할 수 있습니다.

```
In [10]: import pandas as pd
    ...: df['bm_bin'] = pd.cut(df['body_mass_g'],
    ...:                       bins=bin_dividers,
    ...:                       labels=['저체중', '보통체중', '고체중'],
    ...:                       include_lowest=True)

In [11]: print(df['bm_bin'])
0       저체중
1       저체중
2       저체중
4       저체중
5       저체중
       ...
338    보통체중
340    보통체중
341     고체중
342     고체중
343     고체중
Name: bm_bin, Length: 342, dtype: category
Categories (3, object): ['저체중' < '보통체중' < '고체중']
```

- 할당된 경계값 리스트인 bin_dividers 객체를 cut 함수의 bins 옵션에 입력고, 구간의 이름을 저체중, 보통체중, 고체중으로 labels 옵션에 이를 리스트의 형태로 입력합니다. 마지막으로 구간의 첫 경계값을 포함하

기 위해 include_lowest 옵션에 True를 입력합니다. 실행 결과를 살펴보면, 연속형이었던 값이 저체중, 보통체중, 고체중으로 변환된 것을 확인할 수 있습니다.

※ penguins 데이터에서 범주형 변수로 변환한 bm_bin 변수를 이용하여 더미 변수로 새롭게 구성해 보세요.

```
In [12]: print(df['bm_bin'])
0       저체중
1       저체중
2       저체중
4       저체중
5       저체중
       ...
338    보통체중
340    보통체중
341     고체중
342     고체중
343     고체중
Name: bm_bin, Length: 342, dtype: category
Categories (3, object): ['저체중' < '보통체중' < '고체중']
```

```
In [13]: import pandas as pd

In [14]: bm_dummies = pd.get_dummies(df['bm_bin'])

In [15]: print(bm_dummies)
     저체중  보통체중  고체중
0      1     0    0
1      1     0    0
2      1     0    0
4      1     0    0
5      1     0    0
..   ...   ...  ...
338    0     1    0
340    0     1    0
341    0     0    1
342    0     0    1
343    0     0    1

[342 rows x 3 columns]
```

```
In [16]: import pandas as pd

In [17]: bm_dummies1 = pd.get_dummies(df['bm_bin'],drop_first=True)

In [18]: print(bm_dummies1)
     보통체중  고체중
0       0    0
1       0    0
2       0    0
4       0    0
5       0    0
..    ...  ...
338     1    0
340     1    0
341     0    1
342     0    1
343     0    1

[342 rows x 2 columns]
```

- 먼저 penguins 데이터셋의 범주형 변수로 변환시킨 bm_bin 변수를 원핫인코딩 방법으로 더미변수로 변환시키므로 데이터배열에 df['bm_bin']을 입력하고, 원핫인코딩 방법으로 더미변수를 생성하기 위해 drop_first 옵션에 False를 입력하거나 생략합니다. 실행 결과를 살펴보면, 하나의 범주형 변수였던 bm_bin 변수가 저체중, 보통체중, 고체중의 3개의 변수로 생성되고 범주값에 해당하면 1, 그렇지 않으면 0으로 각각 입력되어 있는 것을 확인할 수 있습니다.
- 그리고 축소랭크 방법으로 더미변수를 생성하기 위해 drop_first 옵션에 True를 입력합니다. 실행 결과를 살펴보면, 하나의 범주형 변수였던 bm_bin 변수가 보통체중, 고체중의 2개의 변수로 생성되고 범주값에 해당하면 1, 그렇지 않으면 0으로 각각 입력되어 있는 것을 확인할 수

있습니다. 여기서 저체중은 기준값으로 설정된 것을 확인할 수 있습니다.

※ penguins 데이터에서 'species', 'island', 'body_mass_g', 'sex' 변수만을 추출하여 새롭게 데이터셋을 구성해 보세요.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: df.columns
Out[3]:
Index(['species', 'island', 'bill_length_mm', 'bill_depth_mm',
       'flipper_length_mm', 'body_mass_g', 'sex'],
      dtype='object')

In [4]: list(df.columns)
Out[4]:
['species',
 'island',
 'bill_length_mm',
 'bill_depth_mm',
 'flipper_length_mm',
 'body_mass_g',
 'sex']
```

```
In [5]: df1 = df[['species', 'island', 'body_mass_g', 'sex']]

In [6]: print(df1)
    species     island  body_mass_g     sex
0    Adelie  Torgersen       3750.0    Male
1    Adelie  Torgersen       3800.0  Female
2    Adelie  Torgersen       3250.0  Female
3    Adelie  Torgersen          NaN     NaN
4    Adelie  Torgersen       3450.0  Female
..      ...        ...          ...     ...
339  Gentoo     Biscoe          NaN     NaN
340  Gentoo     Biscoe       4850.0  Female
341  Gentoo     Biscoe       5750.0    Male
342  Gentoo     Biscoe       5200.0  Female
343  Gentoo     Biscoe       5400.0    Male

[344 rows x 4 columns]
```

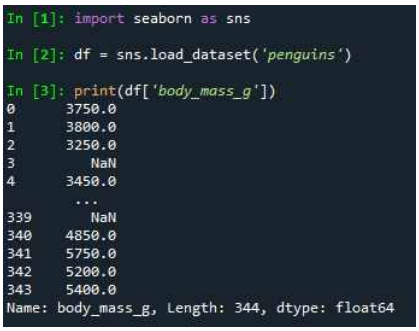
- penguins 데이터셋에서 새롭게 구성하고자 하는 변수명 'species', 'island', 'body_mass_g', 'sex'를 순서대로 리스트로 구성하여 재구성 열 이름 리스트에 대입하면 원하는 순서대로 데이터셋이 새롭게 구성된 것을 확인할 수 있습니다.

※ penguins 데이터에서 변수명을 내림차순으로 정렬하여 새롭게 데이터셋을 구성해 보세요.

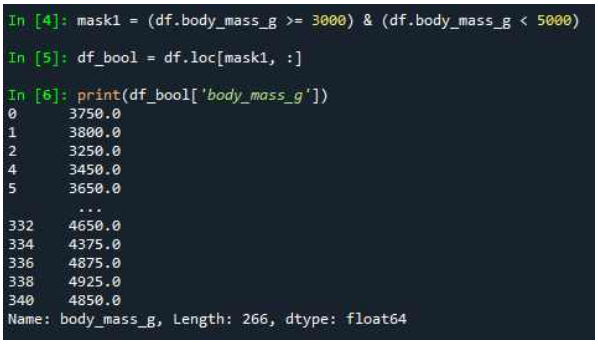
```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: list(df.columns)
Out[3]:
['species',
 'island',
 'bill_length_mm',
 'bill_depth_mm',
 'flipper_length_mm',
 'body_mass_g',
 'sex']

In [4]: list2 = sorted(list(df.columns), reverse=True)

In [5]: df[list2]
Out[5]:
    species     sex     island  ...  body_mass_g  bill_length_mm  bill_depth_mm
0    Adelie    Male  Torgersen  ...       3750.0            39.1           18.7
1    Adelie  Female  Torgersen  ...       3800.0            39.5           17.4
2    Adelie  Female  Torgersen  ...       3250.0            40.3           18.0
3    Adelie     NaN  Torgersen  ...          NaN             NaN            NaN
4    Adelie  Female  Torgersen  ...       3450.0            36.7           19.3
..      ...     ...        ...  ...          ...             ...            ...
339  Gentoo     NaN     Biscoe  ...          NaN             NaN            NaN
340  Gentoo  Female     Biscoe  ...       4850.0            46.8           14.3
341  Gentoo    Male     Biscoe  ...       5750.0            50.4           15.7
342  Gentoo  Female     Biscoe  ...       5200.0            45.2           14.8
343  Gentoo    Male     Biscoe  ...       5400.0            49.9           16.1

[344 rows x 7 columns]
```

- penguins 데이터셋의 변수명을 확인하기 위해 columns 명령어를 이용하여 변수명 리스트를 출력하고, 이를 sorted() 함수에 적용합니다. 여기서 내림차순으로 정렬하기 위해 reverse 옵션에 True를 입력합니다. 이렇게 정렬된 변수명을 재구성 열 이름 리스트에 대입하면 변수명이 내림차순으로 데이터셋이 새롭게 구성된 것을 확인할 수 있습니다.

※ penguins 데이터에서 body_mass_g 값이 3000이상이고 5000 미만인 자료만을 추출하여 새롭게 데이터셋을 구성해 보세요.



```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: print(df['body_mass_g'])
0      3750.0
1      3800.0
2      3250.0
3         NaN
4      3450.0
        ...
339       NaN
340    4850.0
341    5750.0
342    5200.0
343    5400.0
Name: body_mass_g, Length: 344, dtype: float64
```



```
In [4]: mask1 = (df.body_mass_g >= 3000) & (df.body_mass_g < 5000)

In [5]: df_bool = df.loc[mask1, :]

In [6]: print(df_bool['body_mass_g'])
0      3750.0
1      3800.0
2      3250.0
4      3450.0
5      3650.0
        ...
332    4650.0
334    4375.0
336    4875.0
338    4925.0
340    4850.0
Name: body_mass_g, Length: 266, dtype: float64
```

- body_mass_g가 3000이상이고 5000미만 조건식을 이용하여 불린값으로 구성된 시리즈를 생성하고, 이를 데이터프레임의 행 위치에 대입하여 조건에 만족하는 행만 남게되는 데이터프레임을 확인할 수 있습니다. 수정된 데이터프레임을 출력하여 모든 자료가 body_mass_g가 3000이상이고 5000미만임을 확인할 수 있습니다.

※ seaborn 모듈의 penguins 데이터에서, planets 데이터에서 method 변수의 데이터가 여러 단어루 구성되어 있습니다. 이를 첫 번째 단어의 값만으로 데이터를 구성해 보세요.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('planets')

In [3]: print(df['method'])
0       Radial Velocity
1       Radial Velocity
2       Radial Velocity
3       Radial Velocity
4       Radial Velocity
              ...
1030          Transit
1031          Transit
1032          Transit
1033          Transit
1034          Transit
Name: method, Length: 1035, dtype: object
```

```
In [4]: sp = df['method'].str.split(' ')

In [5]: print(sp)
0       [Radial, Velocity]
1       [Radial, Velocity]
2       [Radial, Velocity]
3       [Radial, Velocity]
4       [Radial, Velocity]
               ...
1030          [Transit]
1031          [Transit]
1032          [Transit]
1033          [Transit]
1034          [Transit]
Name: method, Length: 1035, dtype: object
```

```
In [6]: df['method1'] = sp.str.get(0)

In [7]: print(df['method1'])
0        Radial
1        Radial
2        Radial
3        Radial
4        Radial
          ...
1030    Transit
1031    Transit
1032    Transit
1033    Transit
1034    Transit
Name: method1, Length: 1035, dtype: object
```

- method 변수의 자료는 이미 문자형 자료이기 때문에 astype() 함수를 이용해서 문자형 변수로 변경할 필요는 없습니다. 그리고 데이터는 띄워쓰기로 구분되어 있기 때문에 str.split() 함수에 구분 문자로 띄워쓰기(' ')를 입력하여 단어들을 구분합니다. 여기에 첫 번째 단어를 데이터로 사용하므로 str.get() 함수를 이용하여 첫 번째 순서의 값 0을 입력하여 데이터를 추출할 수 있습니다.
- 데이터프레임을 출력하여 method1 변수가 새롭게 생성되어 그 자료값은 첫 번째 단어로만 구성된 것을 확인할 수 있습니다.