In [2]:

```
import pandas as pd
df = pd.read_csv('./data/scores.csv')
df.head(3)
```

Out[2]:

	name	kor	eng	math
0	Aiden	100.0	90.0	95.0
1	Charles	90.0	80.0	75.0
2	Danial	95.0	100.0	100.0

판다스 카테고리형 데이터 다루기

카테고리형 데이터

- 특정 값들로만 이루어지는 자료형

가령,

- 성별 : 남자 / 여자
- 혈액형 : **A / B / O / AB**
- 학점 : **A / B / C / D / F** 로 이루어짐

카테고리형으로 변환하기 : `컬럼.astype('category')`
카테고리 이름 바꾸기 : `컬럼.cat.categories = [카테고리 리스트]`
카테고리 추가하기 : `컬럼.cat.set_categories([카테고리 리스트])`

1. 결측치 확인 / 처리하기

- 결측치는 **0**으로 대치

In [3]:

```
# 결측치 확인하기( info )
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   name    30 non-null     object 
 1   kor     27 non-null     float64
 2   eng     28 non-null     float64
 3   math    29 non-null     float64
dtypes: float64(3), object(1)
memory usage: 1.1+ KB
```

In [4]:

```
# 결측치가 있는 행 삭제하기
df.dropna(inplace = True)
```

In [5]:

```
# 결측치 확인하기
df.isnull().sum()
```

Out[5]:

```
name    0
kor     0
eng     0
math    0
dtype: int64
```

1.2. 평균 점수 컬럼 추가하기

- 등급을 매기기 위한 평균 점수 컬럼 추가

In [7]:

```
round((df.kor + df.eng + df.math) / 3 , 1)
```

Out[7]:

```
0       95.0
1       81.7
2       98.3
3      100.0
5       93.3
6       70.0
7       75.0
8       70.0
9       96.7
10      85.0
11      70.0
12      98.3
13      85.0
14      96.7
15     100.0
17      68.3
18      71.7
20      96.7
21      83.3
22      68.3
23     100.0
25      86.7
26      93.3
28      80.0
29      88.3
dtype: float64
```

In [8]:

```
df['average'] = round((df.kor + df.eng + df.math) / 3, 1)
df
```

Out[8]:

	name	kor	eng	math	average
0	Aiden	100.0	90.0	95.0	95.0
1	Charles	90.0	80.0	75.0	81.7
2	Danial	95.0	100.0	100.0	98.3
3	Evan	100.0	100.0	100.0	100.0
5	Ian	90.0	100.0	90.0	93.3
6	James	70.0	75.0	65.0	70.0
7	Julian	80.0	90.0	55.0	75.0
8	Justin	50.0	60.0	100.0	70.0
9	Kevin	100.0	100.0	90.0	96.7

	name	kor	eng	math	average
10	Leo	90.0	95.0	70.0	85.0
11	Oliver	70.0	75.0	65.0	70.0
12	Peter	100.0	95.0	100.0	98.3
13	Amy	90.0	75.0	90.0	85.0
14	Chloe	95.0	100.0	95.0	96.7
15	Danna	100.0	100.0	100.0	100.0
17	Emma	70.0	65.0	70.0	68.3
18	Jennifer	80.0	55.0	80.0	71.7
20	Linda	100.0	90.0	100.0	96.7
21	Olivia	90.0	70.0	90.0	83.3
22	Rose	70.0	65.0	70.0	68.3
23	Sofia	100.0	100.0	100.0	100.0
25	Vanessa	95.0	70.0	95.0	86.7
26	Viviana	100.0	80.0	100.0	93.3
28	Winnie	70.0	100.0	70.0	80.0
29	Zuly	80.0	90.0	95.0	88.3

1.3. 평균 점수에 따른 등급 컬럼 추가

In [9]:

```
def getGrade(x):
    if x >= 90:
        return 1
    if x >= 80:
        return 2
    if x >= 70:
        return 3
    if x >= 60:
        return 4
    return 5

df['grade'] = df['average'].apply(getGrade)
df
```

Out[9]:

	name	kor	eng	math	average	grade
0	Aiden	100.0	90.0	95.0	95.0	1
1	Charles	90.0	80.0	75.0	81.7	2
2	Danial	95.0	100.0	100.0	98.3	1
3	Evan	100.0	100.0	100.0	100.0	1
5	Ian	90.0	100.0	90.0	93.3	1
6	James	70.0	75.0	65.0	70.0	3
7	Julian	80.0	90.0	55.0	75.0	3
8	Justin	50.0	60.0	100.0	70.0	3
9	Kevin	100.0	100.0	90.0	96.7	1
10	Leo	90.0	95.0	70.0	85.0	2
11	Oliver	70.0	75.0	65.0	70.0	3
12	Peter	100.0	95.0	100.0	98.3	1
13	Amy	90.0	75.0	90.0	85.0	2
14	Chloe	95.0	100.0	95.0	96.7	1

	name	kor	eng	math	average	grade
15	Danna	100.0	100.0	100.0	100.0	1
17	Emma	70.0	65.0	70.0	68.3	4
18	Jennifer	80.0	55.0	80.0	71.7	3
20	Linda	100.0	90.0	100.0	96.7	1
21	Olivia	90.0	70.0	90.0	83.3	2
22	Rose	70.0	65.0	70.0	68.3	4
23	Sofia	100.0	100.0	100.0	100.0	1
25	Vanessa	95.0	70.0	95.0	86.7	2
26	Viviana	100.0	80.0	100.0	93.3	1
28	Winnie	70.0	100.0	70.0	80.0	2
29	Zuly	80.0	90.0	95.0	88.3	2

2. 카테고리형으로 변환하기

- 등급 컬럼(**grade**)를 카테고리 자료형으로 변환하기

In [10]:

```
# 자료형 확인하기
df.dtypes
```

Out[10]:

```
name         object
kor         float64
eng         float64
math        float64
average     float64
grade         int64
dtype: object
```

In [15]:

```
# 자료형 변환하기
df['grade'] = df['grade'].astype('category')
df.dtypes
```

Out[15]:

```
name          object
kor          float64
eng          float64
math         float64
average      float64
grade       category
dtype: object
```

In [16]:

```
df['grade'].dtype
```

Out[16]:

```
CategoricalDtype(categories=[1, 2, 3, 4], ordered=False)
```

3. 카테고리 이름 바꾸기

`컬럼.cat.categories = [카테고리 리스트]`

In [19]:

```
df['grade'].cat.categories = ['A', 'B', 'C', 'D']
```

```
# df['grade'].cat.categories = ['A', 'B', 'C', 'D', 'F']
df
```

Out[19]:

	name	kor	eng	math	average	grade
0	Aiden	100.0	90.0	95.0	95.0	A
1	Charles	90.0	80.0	75.0	81.7	B
2	Danial	95.0	100.0	100.0	98.3	A
3	Evan	100.0	100.0	100.0	100.0	A
5	Ian	90.0	100.0	90.0	93.3	A
6	James	70.0	75.0	65.0	70.0	C
7	Julian	80.0	90.0	55.0	75.0	C
8	Justin	50.0	60.0	100.0	70.0	C
9	Kevin	100.0	100.0	90.0	96.7	A
10	Leo	90.0	95.0	70.0	85.0	B
11	Oliver	70.0	75.0	65.0	70.0	C
12	Peter	100.0	95.0	100.0	98.3	A
13	Amy	90.0	75.0	90.0	85.0	B
14	Chloe	95.0	100.0	95.0	96.7	A
15	Danna	100.0	100.0	100.0	100.0	A
17	Emma	70.0	65.0	70.0	68.3	D
18	Jennifer	80.0	55.0	80.0	71.7	C
20	Linda	100.0	90.0	100.0	96.7	A
21	Olivia	90.0	70.0	90.0	83.3	B
22	Rose	70.0	65.0	70.0	68.3	D
23	Sofia	100.0	100.0	100.0	100.0	A
25	Vanessa	95.0	70.0	95.0	86.7	B
26	Viviana	100.0	80.0	100.0	93.3	A
28	Winnie	70.0	100.0	70.0	80.0	B
29	Zuly	80.0	90.0	95.0	88.3	B

실제 데이터는 **4**종류밖에 없어서 우리가 원하는 **5**종류의 데이터를 넣어 주면 에러가 발생한다.
따라서 일단 존재하는 **4**종류만 넣고, 나중에 따로 **F**학점을 넣어 주어야 한다.

4. 누락된 카테고리 추가

`컬럼.cat.set_categories([카테고리 리스트])`

In [24]:

```
df['grade'].cat.set_categories(['A', 'B', 'C', 'D', 'F'])
df['grade'].dtype
```

Out[24]:

```
CategoricalDtype(categories=['A', 'B', 'C', 'D'], ordered=False)
```

5. 데이터 용량 확인하기

- **titanic** 데이터에서 카테고리형으로 관리할 수 있는 자료형을 카테고리형으로 변환하여 데이터 용량 비교하기

5.1 데이터 준비하고 확인하기

In [25]:

```
df_titanic = pd.read_csv('./data/titanic.csv')
df_titanic.head(1)
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S

In [26]:

```
df_titanic['Survived'].unique() # 0은 사망, 1은 생존
```

Out[26]:

```
array([0, 1], dtype=int64)
```

In [27]:

```
df_titanic['Pclass'].unique()
```

Out[27]:

```
array([3, 1, 2], dtype=int64)
```

In [28]:

```
df_titanic['Name'].unique() # category 불가
```

Out[28]:

```
array(['Braund, Mr. Owen Harris',
       'Cumings, Mrs. John Bradley (Florence Briggs Thayer)',
       'Heikkinen, Miss. Laina', ..., 'Saether, Mr. Simon Sivertsen',
       'Ware, Mr. Frederick', 'Peter, Master. Michael J'], dtype=object)
```

In [35]:

```
df_titanic['Embarked'].unique()
```

Out[35]:

```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [29]:

```
df_titanic['Sex'].unique()
```

Out[29]:

```
array(['male', 'female'], dtype=object)
```

In [32]:

```
df_titanic['SibSp'].unique()
```

Out[32]:

```
array([1, 0, 3, 4, 2, 5, 8], dtype=int64)
```

In [33]:

```
df_titanic['Parch'].unique()
```

Out[33]:

```
array([0, 1, 2, 5, 3, 4, 6, 9], dtype=int64)
```

In [37]:

```
# 데이터 타입
df_titanic.dtypes
```

Out[37]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [38]:

```
# 용량 확인하기
df_titanic.info() # 122.8KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   int64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
dtypes: float64(2), int64(5), object(5)
memory usage: 122.8+ KB
```

5.2 카테고리형으로 바꾸기

`컬럼.astype('category')`

In [39]:

```
# 카테고리형으로 바꾸기( Survived, Pclass, Sex, Embarked )
df_titanic['Survived'] = df_titanic['Survived'].astype('category')
df_titanic['Pclass'] = df_titanic['Pclass'].astype('category')
df_titanic['Sex'] = df_titanic['Sex'].astype('category')
df_titanic['Embarked'] = df_titanic['Embarked'].astype('category')
```

In [40]:

```
# 데이터 타입
df_titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   category
 2   Pclass       1309 non-null   category
```

```
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   category
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   category
dtypes: category(4), float64(2), int64(3), object(3)
memory usage: 87.6+ KB
```

memory usage : 87.6KB -> 절약되었다.