결측값(missing values)

결측값(missing value)은 데이터셋에서 특정 위치에 값이 없는 경우를 의미합니다. 결측값은 여러 가지 이유로 발생할 수 있습니다. 예를 들어, 설문조사 응답자가 질문에 답하지 않은 경우, 센서가 일시적으로 작동하지 않아 데이터를 수집하지 못한 경우, 데이터 입력 오류 등 여러 상황에서 결측값이 생길 수 있습니다.

In [2]:
```python
import pandas as pd
```

In [3]:
```python
df = pd.read_csv("titanic-dataset.csv")
df.head(5)
```

Out[3]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [3]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

결측값 확인하기

결측값이 어디에 있는지 확인하는 것이 첫 단계입니다. Pandas는 isnull() 또는 isna() 메서드를 사용하여 결측값을 확인할 수 있습니다.

In [4]:
```python
#isnull()
#False는 결측값

df.isnull()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

In [5]:
```python
#isna()

df.isna()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

결측값 개수 확인하기

각 열에 결측값이 몇 개인지 확인하려면 sum() 메서드를 사용합니다.

In [6]:
```python
# 각 열의 결측값 개수 확인

df.isnull().sum()
```

Out[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

결측값 제거하기

결측값을 제거하는 방법은 dropna() 메서드를 사용합니다.

In [ ]:
```python
# 결측값이 있는 행 제거
df_dropped_rows = df.dropna()
print(df_dropped_rows)
```

In [ ]:
```python
# 결측값이 있는 열 제거
df_dropped_columns = df.dropna(axis=1)
print(df_dropped_columns)
```

In [ ]:
```python
# 각 열의 평균값으로 결측값 채우기
df_filled_mean = df.fillna(df.mean())
print(df_filled_mean)
```

In [ ]:
```python
# 각 열의 중앙값으로 결측값 채우기
df_filled_median = df.fillna(df.median())
print(df_filled_median)
```

In [ ]:
```python
# 각 열에 대해 다른 값으로 결측값 채우기
values = {'Name': 'Unknown', 'Age': df['Age'].mean(), 'Score': 0}
df_filled_custom = df.fillna(value=values)
print(df_filled_custom)
```

In [ ]:

In [7]:
```python
#titanic_dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [8]:
```python
df.head(3)
```

Out[8]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [9]:
```python
# 딕셔너리로 데이터 채우기
values = {'Age':0.0,'Cabin':'NaN','Embarked':'None'}
df = df.fillna(value = values)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

데이터 필터링하기

데이터프레임에서는 특정한 조건을 만족하는 데이터만 선택하여 조회할 수 있습니다.

In [14]:
```python
# 샘플 데이터 생성
data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'age': [24, 27, 22, 32, 29],
    'city': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix']
}

df = pd.DataFrame(data)
print(df)
```

```
      name  age         city
0    Alice   24     New York
1      Bob   27  Los Angeles
2  Charlie   22      Chicago
3    David   32      Houston
4      Eve   29      Phoenix
```

In [15]:
```python
# 나이가 25 이상인 행을 필터링
filtered_df = df[df['age'] >= 25]
print(filtered_df)
```

```
    name  age         city
1    Bob   27  Los Angeles
3  David   32      Houston
4    Eve   29      Phoenix
```

여러 조건으로 행 필터링 (논리 연산)

In [ ]:
```python
df[(조건식 1) & (조건식 2)] # And(그리고)
df[(조건식 1) | (조건식 2)] # Or (아니면)
```

In [17]:
```python
# 나이가 25 이상이고, 도시가 'New York'인 행을 필터링
filtered_df = df[(df['age'] >= 25) & (df['city'] == 'New York')]
print(filtered_df)
```

```
Empty DataFrame
Columns: [name, age, city]
Index: []
```

특정 열의 값으로 필터링

In [18]:
```python
filtered_df = df[df['city'] == 'Chicago']
print(filtered_df)
```

```
      name  age     city
2  Charlie   22  Chicago
```

query 메서드를 사용한 필터링

In [19]:
```python
filtered_df = df.query('age >= 25 and city == "New York"')
print(filtered_df)
```

```
Empty DataFrame
Columns: [name, age, city]
Index: []
```

isin 메서드를 사용한 필터링

In [20]:
```python
filtered_df = df[df['city'].isin(['New York', 'Chicago'])]
print(filtered_df)
```

```
      name  age      city
0    Alice   24  New York
2  Charlie   22   Chicago
```

between 메서드를 사용한 필터링

In [21]:
```python
filtered_df = df[df['age'].between(25, 30)]
print(filtered_df)
```

```
  name  age         city
1  Bob   27  Los Angeles
4  Eve   29      Phoenix
```

In [ ]:

In [10]:
```python
df.head(5)
```

Out[10]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [13]:
```python
# 특정 열의 값으로 필터링
df[df['Sex'] == 'female'].head(6)
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S

In [14]:
```python
# 성별은 male, Age는 25보다 큰거
df[(df['Sex']=='male')& (df['Age']>25)]
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
...	...	...	...	...	...	...	...	...	...	...	...	...
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

274 rows × 12 columns

In [15]:
```python
# query 메세드
df.query('Sex == "male" and Age > 25')
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
...	...	...	...	...	...	...	...	...	...	...	...	...
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

274 rows × 12 columns

In [16]:
```python
df['Embarked'].value_counts()
```

Out[16]:
```
Embarked
S       644
C       168
Q        77
None      2
Name: count, dtype: int64
```

In [19]:
```python
# isin 메서드
df[df['Embarked'].isin(['S','C'])]
```

Out[19]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	0.0	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

812 rows × 12 columns

In [20]:
```python
#between 메세드
df[df['Age'].between(25,30)]
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
34	35	0	1	Meyer, Mr. Edgar Joseph	male	28.0	1	0	PC 17604	82.1708	NaN	C
41	42	0	2	Turpin, Mrs. William John Robert (Dorothy Ann ...	female	27.0	1	0	11668	21.0000	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

131 rows × 12 columns

데이터 합치 concat & merge

concat와 merge는 파이썬의 pandas 라이브러리에서 데이터프레임을 결합하는 데 사용되는 두 가지 주요 함수입니다. 이 두 함수는 데이터프레임을 결합하는 방식이 다르기 때문에 사용 용도에 따라 적절한 함수를 선택해야 합니다.

concat

concat 함수는 주로 데이터프레임이나 시리즈를 축(axis)을 기준으로 연결할 때 사용됩니다. 데이터의 모양(shape)이 동일하거나 데이터가 동일한 인덱스나 컬럼을 공유할 때 유용합니다.

- 축: axis=0 (기본값)일 때는 행을 따라 연결하고, axis=1일 때는 열을 따라 연결합니다.

In [22]:
```python
# 예제 데이터프레임 생성
df3 = pd.DataFrame({'A': ['A0', 'A1', 'A2'], 'B': ['B0', 'B1', 'B2']})
df4 = pd.DataFrame({'A': ['A3', 'A4', 'A5'], 'B': ['B3', 'B4', 'B5']})

# 행을 기준으로 데이터프레임 연결
concatenated = pd.concat([df3, df4], axis=0)
print(concatenated)

# 열을 기준으로 데이터프레임 연결
concatenated_col = pd.concat([df3, df4], axis=1)
print(concatenated_col)
```

```
    A   B
0  A0  B0
1  A1  B1
2  A2  B2
0  A3  B3
1  A4  B4
2  A5  B5
    A   B   A   B
0  A0  B0  A3  B3
1  A1  B1  A4  B4
2  A2  B2  A5  B5
```

In [21]:
```python
df.head(3)
```

Out[21]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [24]:
```python
#데이터 나누기
dm2=df.iloc[:,0:8]
dm3 = df.iloc[:,9:12]
```

In [26]:
```python
#데이터 열을 합치기
dm4 = pd.concat([dm2,dm3],axis = 1)

dm4.head()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	NaN	S

merge

merge 함수는 데이터베이스의 JOIN 연산과 유사하게 두 데이터프레임을 특정 키(열)를 기준으로 결합합니다. 관계형 데이터베이스에서 테이블을 결합하는 것과 같은 방식으로 동작합니다.

- on: 결합할 키 열을 명시합니다.
- how: 결합 방식 ('left', 'right', 'outer', 'inner' 중 선택).

In [3]:
```python
# 예시 데이터프레임
df1 = pd.DataFrame({'key': ['K0', 'K1', 'K2'], 'A': ['A0', 'A1', 'A2'], 'B': ['B0', 'B1', 'B2']})
df2 = pd.DataFrame({'key': ['K0', 'K1', 'K2'], 'C': ['C0', 'C1', 'C2'], 'D': ['D0', 'D1', 'D2']})

# 키 열 'key'를 기준으로 결합
result = pd.merge(df1, df2, on='key')
print(result)

# outer join 예시
result = pd.merge(df1, df2, on='key', how='outer')
print(result)
```

```
  key   A   B   C   D
0  K0  A0  B0  C0  D0
1  K1  A1  B1  C1  D1
2  K2  A2  B2  C2  D2
  key   A   B   C   D
0  K0  A0  B0  C0  D0
1  K1  A1  B1  C1  D1
2  K2  A2  B2  C2  D2
```

In [ ]:

In [27]:
```python
df.head(3)
```

Out[27]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [28]:
```python
# 데이터 나누기
df2 = df.iloc[:,0:7]
df3 = df[['PassengerId','Ticket','Fare','Cabin','Embarked']]
```

In [29]:
```python
df2.head(3)
```

Out[29]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0

In [30]:
```python
df3.head(3)
```

Out[30]:

	PassengerId	Ticket	Fare	Cabin	Embarked
0	1	A/5 21171	7.2500	NaN	S
1	2	PC 17599	71.2833	C85	C
2	3	STON/O2. 3101282	7.9250	NaN	S

In [35]:
```python
#데이터 열 합치기
df5 = pd.merge(df2,df3,on='PassengerId')

df5.head(3)
```

Out[35]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	STON/O2. 3101282	7.9250	NaN	S

groupby, crosstab, pivot table

데이터 분석에서 groupby, crosstab, 그리고 pivot_table은 데이터를 요약하고 변환하는 데 매우 유용한 도구들입니다.

GroupBy

groupby 메서드는 특정 기준에 따라 데이터를 그룹으로 나누는 데 사용됩니다. 이는 집계, 변환, 필터링 등 다양한 작업에 활용될 수 있습니다.

- 사용 시기: 데이터를 서브셋으로 나누어 작업을 수행하고 싶을 때.
- 주요 작업: sum(), mean(), count(), max(), min(), apply().

In [3]:
```python
# 샘플 데이터프레임
df = pd.DataFrame({
    'A': ['foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'foo'],
    'B': ['one', 'one', 'two', 'two', 'one', 'two', 'one', 'two'],
    'C': range(8)
})

# A 열을 기준으로 그룹화하고 C 열을 합계
grouped = df.groupby('A')['C'].sum()
print(grouped)
```

```
A
bar     9
foo    19
Name: C, dtype: int64
```

In [ ]:

In [36]:
```python
df.head(3)
```

Out[36]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [37]:
```python
# # groupby(['Sex', 'Pclass'])로 데이터를 'Sex'와 'Pclass' 컬럼을 기준으로 그룹화
df.groupby(['Sex','Pclass'])['Survived'].mean()
```

Out[37]:
```
Sex     Pclass
female  1         0.968085
        2         0.921053
        3         0.500000
male    1         0.368852
        2         0.157407
        3         0.135447
Name: Survived, dtype: float64
```

In [38]:
```python
# reset_index()를 사용하여 그룹화된 결과를 데이터프레임 형태로 변환
df.groupby(['Sex','Pclass'])['Survived'].mean().reset_index()
```

Out[38]:

	Sex	Pclass	Survived
0	female	1	0.968085
1	female	2	0.921053
2	female	3	0.500000
3	male	1	0.368852
4	male	2	0.157407
5	male	3	0.135447

Crosstab (교차표)

crosstab 함수는 두 개 이상의 요인 간의 교차표를 계산합니다. 이는 범주형 변수 간의 관계를 이해하는 데 유용합니다.

- 사용 시기: 빈도표(분할표)를 만들고 싶을 때.
- 주요 작업: 발생 횟수 계산, 카운트 정규화.

In [4]:
```python
# 샘플 데이터프레임
df = pd.DataFrame({
    'A': ['foo', 'foo', 'foo', 'bar', 'bar', 'foo', 'foo', 'bar'],
    'B': ['one', 'one', 'two', 'one', 'two', 'two', 'two', 'one'],
    'C': range(8)
})

# A와 B 열 간의 교차표
crosstab_result = pd.crosstab(df['A'], df['B'])
print(crosstab_result)
```

```
B    one  two
A            
bar    2    1
foo    2    3
```

In [ ]:

In [39]:
```python
df.head(3)
```

Out[39]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [40]:
```python
#pd.crosstab() 함수는 두 개 이상의 요인(factor)으로 교차표를 생성
pd.crosstab(df['Cabin'],df['Sex'])
```

Out[40]:

Sex	female	male
Cabin		
A10	0	1
A14	0	1
A16	1	0
A19	0	1
A20	0	1
...	...	...
F38	0	1
F4	1	1
G6	4	0
NaN	217	470
T	0	1

148 rows × 2 columns

Pivot Table (피벗 테이블)

pivot_table 함수는 groupby의 일반화된 형태로, 다차원 테이블을 만들고 복잡한 집계를 수행할 수 있습니다.

- 사용 시기: 다중 레벨 인덱스를 사용하여 데이터를 요약하고, 집계 함수를 적용하고 싶을 때.
- 주요 작업: 데이터 요약, 다중 레벨 집계.

In [5]:
```python
# 샘플 데이터프레임
df = pd.DataFrame({
    'A': ['foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'foo'],
    'B': ['one', 'one', 'two', 'two', 'one', 'two', 'one', 'two'],
    'C': range(8)
})

# C 열의 값을 사용한 피벗 테이블
pivot_table_result = df.pivot_table(values='C', index='A', columns='B', aggfunc='sum')
print(pivot_table_result)
```

```
B    one  two
A            
bar    1    8
foo   10    9
```

In [ ]:

In [43]:
```python
import numpy as np
```

In [41]:
```python
df.head(3)
```

Out[41]:

	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	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [44]:
```python
# df.pivot_table() 함수는 피벗 테이블 생성
# 데이터프레임 df를 입력으로 사용함
# index=['Sex', 'Pclass']는 피벗 테이블의 인덱스로 사용할 열 지정
# 여기서는 성별(Sex)과 객실 등급(Pclass)별로 데이터 그룹화함
# aggfunc={'Age': np.mean, 'Survived': np.sum}는 집계 함수를 지정함
# Age 컬럼은 평균(np.mean)을 계산하고, Survived 컬럼은 합계(np.sum)를 계산함

# 이 코드는 성별과 객실 등급별로 Age의 평균과 Survived의 합계를 계산하여
# 새로운 피벗 테이블을 생성함

pivot_table_result = df.pivot_table(df, index = ['Sex','Pclass'],aggfunc = {'Age':np.mean,'Survived':np.sum})
pivot_table_result
```

```
/var/folders/8f/xt_fjqzx5n19d61dsmb06v400000gn/T/ipykernel_932/378468514.py:1: FutureWarning: The provided callable <function mean at 0x1081b9c60> is currently using SeriesGroupBy.mean. In a future version of pandas, the provided callable will be used directly. To keep current behavior pass the string "mean" instead.
  pivot_table_result = df.pivot_table(df, index = ['Sex','Pclass'],aggfunc = {'Age':np.mean,'Survived':np.sum})
/var/folders/8f/xt_fjqzx5n19d61dsmb06v400000gn/T/ipykernel_932/378468514.py:1: FutureWarning: The provided callable <function sum at 0x1080dc430> is currently using SeriesGroupBy.sum. In a future version of pandas, the provided callable will be used directly. To keep current behavior pass the string "sum" instead.
  pivot_table_result = df.pivot_table(df, index = ['Sex','Pclass'],aggfunc = {'Age':np.mean,'Survived':np.sum})
```

Out[44]:

		Age	Survived
Sex	**Pclass**		
female	**1**	31.297872	91
	2	27.967105	70
	3	15.406250	72
male	**1**	34.175574	45
	2	28.178981	17
	3	19.326859	47

In [ ]: