[3-4. 데이터 합치고, 변형하기(merge, concat)]

1. merge 함수

pandas의 merge 함수는 두 데이터프레임을 병합하는 기능을 제공하며, SQL의 JOIN 연산과 유사하게 작동합니다.

1-1. 기본 사용법

In [ ]:
```
pandas.merge(
    left, right,
    how='inner', on=None, left_on=None, right_on=None,
    left_index=False, right_index=False, sort=True
)
```

- left : 병합할 첫 번째 DataFrame
- right : 병합할 두 번째 DataFrame
- how : ['left', 'right', 'outer', 'inner']. 병합 방식을 지정하며, 기본값은 inner 입니다.
  - 내부 조인(inner join) : 양쪽 DataFrame 모두에 존재하는 키(열)에 대해서만 병합
  - 외부 조인(outer join) : 양쪽 DataFrame 중 하나라도 키(열)가 존재하면 병합하고, 누락된 데이터는 'NaN'으로 표시
  - 왼쪽 조인(left join) : 왼쪽 DataFrame의 키(열)를 기준으로 병합하며, 오른쪽 DataFrame에서 매칭되지 않는 경우 'NaN'으로 표시
  - 오른쪽 조인(right join) : 오른쪽 DataFrame의 키(열)를 기준으로 병합하며, 왼쪽 DataFrame에서 매칭되지 않는 경우 'NaN'으로 표시
- on : 병합할 때 사용할 열 이름. 이 매개변수가 주어지면, 이 이름을 가진 열이 양쪽 DataFrame에서 병합의 기준이 됩니다. 두 DataFrame에 모두 존재하는 열 이름이어야 합니다.
- left_on : 첫 번째 DataFrame에서 병합 키로 사용할 열 이름
- right_on : 두 번째 DataFrame에서 병합 키로 사용할 열 이름
- left_index : 첫 번째 DataFrame의 인덱스를 병합 키로 사용할지 여부를 지정(기본값은 False)
- right_index : 첫 번째 DataFrame의 인덱스를 병합 키로 사용할지 여부를 지정(기본값은 False)
- sort : 병합 결과를 병합 키에 따라 정렬할지 여부를 지정(기본값은 False)

1-2. 기본 사용 예시

In [ ]:
```python
import pandas as pd
from pandas import DataFrame, Series

df1 = DataFrame({
    'employee': ['Bob', 'Jake', 'Lisa', 'Sue'],
    'group': ['Accounting', 'Engineering', 'Engineering', 'HR']
})
df2 = DataFrame({
    'employee': ['Lisa', 'Bob', 'Jake', 'Sue'],
    'hire_date': [2004, 2008, 2012, 2014]
})

print("df1 : \n", df1)
print("\ndf2 : \n", df2)
```

```
df1 :
   employee        group
0       Bob   Accounting
1      Jake  Engineering
2      Lisa  Engineering
3       Sue           HR

df2 :
   employee  hire_date
0      Lisa       2004
1       Bob       2008
2      Jake       2012
3       Sue       2014
```

In [ ]:
```python
# employee를 기준으로 두 DataFrame을 내부 조인(inner join)
merged_df = pd.merge(df1, df2, on='employee')

print(merged_df)
```

```
  employee        group  hire_date
0      Bob   Accounting       2008
1     Jake  Engineering       2012
2     Lisa  Engineering       2004
3      Sue           HR       2014
```

In [ ]:
```python
# 이 경우 how 매개변수를 어떻게 지정하여도 결과는 동일함
merged_df = pd.merge(df1, df2, on='employee', how='inner')

print(merged_df)
```

```
  employee        group  hire_date
0      Bob   Accounting       2008
1     Jake  Engineering       2012
2     Lisa  Engineering       2004
3      Sue           HR       2014
```

1-3. 서로 다른 키 값으로 merge하는 경우

In [ ]:
```
# 첫 번째 DataFrame 생성
df1 = DataFrame({
    'employee_id': ['1', '2', '3', '4'],
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age' : [30, 37, 26, 45]
})

# 두 번째 DataFrame 생성
df2 = DataFrame({
    'id': ['1', '2', '4', '5'],
    'salary': [70000, 60000, 80000, 90000],
    'bonus': [10000, 20000, 10000, 20000]
})

# 서로 다른 키(employee_id, id)로 두 DataFrame 병합
merged_df = pd.merge(df1, df2, left_on='employee_id', right_on='id')

print(merged_df)
```

```
  employee_id   name  age id  salary  bonus
0           1  Alice   30  1   70000  10000
1           2    Bob   37  2   60000  20000
2           4  David   45  4   80000  10000
```

이 경우 employee_id와 id 값이 일치하는 행만을 포함하며, 두 키 모두 결과에 포함됩니다.

필요한 경우 drop 메서드를 사용하여 중복되는 하나의 키(id 또는 employee_id)를 삭제할 수 있습니다.

In [ ]:
```
merged_df = merged_df.drop('id', axis=1)
print(merged_df)
```

```
  employee_id   name  age  salary  bonus
0           1  Alice   30   70000  10000
1           2    Bob   37   60000  20000
2           4  David   45   80000  10000
```

1-4. 인덱스를 기준으로 merge하는 경우

기존 df1의 employee_id와 df2의 id를 인덱스로 변경

In [ ]:
```
df1.set_index('employee_id', inplace=True)
print(df1)
```

```
                name  age
employee_id
1              Alice   30
2                Bob   37
3            Charlie   26
4              David   45
```

In [ ]:
```
df2.set_index('id', inplace=True)
print(df2)
```

```
    salary  bonus
id
1    70000  10000
2    60000  20000
4    80000  10000
5    90000  20000
```

인덱스를 기준으로 병합

In [ ]:
```
merged_df = pd.merge(
    df1, df2,
    left_index=True, right_index=True,
    how='left'
)

print(merged_df)
```

```
                name  age   salary    bonus
employee_id
1              Alice   30  70000.0  10000.0
2                Bob   37  60000.0  20000.0
3            Charlie   26      NaN      NaN
4              David   45  80000.0  10000.0
```

2. join 메서드

join 메서드는 한 DataFrame을 다른 DataFrame의 인덱스에 따라 병합할 때 사용됩니다. 인덱스를 기준으로 merge 작업을 할 때 편리합니다.

join 메서드는 인덱스를 기준으로 병합을 수행하며, 'on' 매개변수를 사용해 특정 컬럼을 병합 기준으로 지정할 수도 있습니다.

2-1. 기본 사용법

In [ ]:
```
DataFrame.join(other, on=None, how='left', sort=False)
```

- other : 현재 DataFrame에 병합할 다른 DataFrame, Series, 또는 DataFrame의 리스트
- on : 다른 DataFrame의 컬럼을 현재 DataFrame의 인덱스와 병합하기 위해 사용할 컬럼 이름
- how : 병합 방식('left', 'right', 'outer', 'inner' 중 하나) 기본값은 'left'
- sort : 결과를 병합 키에 따라 정렬할지 여부를 결정하는 블리언 값. 기본값은 False

2-2. 사용 예시

In [ ]:
```
joined_df = df1.join(df2, how='left')
print(joined_df)
```

```
                name  age   salary    bonus
employee_id
1              Alice   30  70000.0  10000.0
2                Bob   37  60000.0  20000.0
3            Charlie   26      NaN      NaN
4              David   45  80000.0  10000.0
```

3. concat 함수

concat 함수는 pandas에서 여러 객체(주로 DataFrame이나 Series)를 축(axis)을 따라 연결하는 데 사용되는 함수입니다. 이 함수는 리스트나 딕셔너리 형태의 pandas 객체들을 받아서 단일 객체로 결합합니다. concat 함수는 인덱스를 기준으로 데이터를 연결하며, 필요에 따라 인덱스를 무시하고 데이터를 순차적으로 연결할 수도 있습니다.

In [ ]:
```
pandas.concat(objs, axis=0, join='outer', ignore_index=False)
```

- objs : 연결하려는 pandas 객체의 리스트 또는 딕셔너리.
- axis : 연결 방향을 결정합니다. 0은 행 방향(세로), 1은 열 방향(가로) 입니다.
- join : ['outer', 'inner'], 기본값은 'outer' 입니다.
- ignore_index : True로 설정하면 연결 후의 인덱스를 [0, ..., n-1]로 새로 설정합니다. 기본값은 False 입니다.

In [ ]:
```
import pandas as pd

df1 = pd.DataFrame({'A': ['A1', 'A2'], 'B': ['B1', 'B2']})
df2 = pd.DataFrame({'A': ['A3', 'A4'], 'B': ['B3', 'B4']})

result = pd.concat([df1, df2])
print(result)
```

```
    A   B
0  A1  B1
1  A2  B2
0  A3  B3
1  A4  B4
```

In [ ]:
```
# 열 방향(가로)으로 DataFrame 연결하기
result = pd.concat([df1, df2], axis=1)
print(result)
```

```
    A   B   A   B
0  A1  B1  A3  B3
1  A2  B2  A4  B4
```

In [ ]:
```
# 인덱스 무시하고 연결하기
result = pd.concat([df1, df2], ignore_index=True)
print(result)
```

```
    A   B
0  A1  B1
1  A2  B2
2  A3  B3
3  A4  B4
```