02_데이터프레임과 시리즈

• 나만의 데이터 만들기

> Series() 메소드에 리스트를 전달하여 시리즈 생성

```
s = pd.Series(['Wes McKinney','Creator of Pandas'])
s
```

Out :

```
0          Wes McKinney
1    Creator of Pandas
dtype: object
```

> index 인자에 문자열을 리스트로 담아서 전달 가능

```
s = pd.Series(['Wes McKinney','Creator of Pandas'],
              index=['Person','Who'])
s
```

Out :

```
Person          Wes McKinney
Who        Creator of Pandas
dtype: object
```

• 나만의 데이터 만들기

> **데이터프레임 만들기**

- DataFrame() 메소드에 딕셔너리를 전달하여 데이터프레임 생성

```
scientists = pd.DataFrame({
    'Name':['Rosaline Franklin','William Gosset'],
    'Occupation':['Chemist','Statistician'],
    'Born':['1920-07-25','1876-06-13'],
    'Died':['1958-04-16','1937-10-16'],
    'Age':[37,61]})
scientists
```

Out :

	Name	Occupation	Born	Died	Age
0	Rosaline Franklin	Chemist	1920-07-25	1958-04-16	37
1	William Gosset	Statistician	1876-06-13	1937-10-16	61

• 나만의 데이터 만들기

> **데이터프레임 만들기**

- DataFrame() 메소드에 딕셔너리를 전달하여 데이터프레임 생성

```
Scientists = pd.DataFrame(
                    data={'Occupation':['Chemist','Statistician'],
                          'Born':['1920-07-25','1876-06-13'],
                          'Died':['1958-04-16','1937-10-16'],
                          'Age':[37,61]},
                    index=['Rosaline Franklin','William Gosset'],
                    columns=['Occupation','Born','Age','Died'])
scientists
```

Out :

	Occupation	Born	Age	Died
Rosaline Franklin	Chemist	1920-07-25	37	1958-04-16
William Gosset	Statistician	1876-06-13	61	1937-10-16

• 시리즈 다루기 - 기초

> 데이터프레임에서 시리즈 선택하기

```
first_row = scientists.loc['William Gosset']
print(type(first_row))
print(first_row)
```

Out :

```
<class 'pandas.core.series.Series'>
Occupation    Statistician
Born            1876-06-13
Age                     61
Died            1937-10-16
Name: William Gosset, dtype: object
```

• 시리즈 다루기 - 기초

> index 속성 사용하기

```
first_row.index
```

Out : Index(['Occupation', 'Born', 'Age', 'Died'],
dtype='object')

> values 속성 사용하기

```
first_row.values
```

Out : array(['Statistician', '1876-06-13', 61, '1937-10-16'],
dtype=object)

• 시리즈 다루기 - 기초

> index 속성 응용하기

```
first_row.index[0]
```

Out : 'Occupation'

> values 속성 응용하기

```
first_row.values[0]
```

Out : 'Statistician'

• 시리즈 다루기 – 기초

▪ 컬럼명으로 추출하기

> scientists 데이터프레임의 Age 열 추출

```
ages = scientists['Age']
ages
```

Out :

```
Rosaline Franklin    37
William Gosset       61
Name: Age, dtype: int64
```

> scientists 데이터프레임의 Born, Died 열 추출

```
dates = scientists[['Born', 'Died']]
dates
```

Out :

	Born	Died
Rosaline Franklin	1920-07-25	1958-04-16
William Gosset	1876-06-13	1937-10-16

• 시리즈 다루기 – 응용

> 시리즈, 데이터프레임 관련 메소드

시리즈 메소드	설명
isin	시리즈에 포함된 값이 있는지 확인
count	데이터 개수 반환
value_counts	데이터 별 개수 반환
min	최소값 반환
max	최대값 반환
mean	산술 평균 반환
median	중간값 반환
quantile	사분위 값(25%, 50%, 75%) 반환
describe	요약 통계량 계산
replace	특정 값을 가진 시리즈 값을 교체
sample	시리즈에서 임의의 값을 반환
sort_values	값을 정렬
to_frame	시리즈를 데이터프레임으로 변환

• 시리즈 다루기 – 응용

> 데이터로 함수 응용하기

```
scientists = pd.read_csv('data/scientists.csv')
ages = scientists['Age']
scientists
```

Out :

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist
5	John Snow	1813-03-15	1858-06-16	45	Physician
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

• 시리즈 다루기 – 응용

- 데이터 포함여부 확인

> isin() 메소드 사용

```
scientists.isin([37, 41, 45])
```

Out :

	Name	Born	Died	Age	Occupation
0	False	False	False	True	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
5	False	False	False	True	False
6	False	False	False	True	False
7	False	False	False	False	False

• 시리즈 다루기 – 응용

▪ 데이터 개수 확인하기

> count() 메소드 사용

```
scientists.count()
```

Out :

```
Name        8
Born        8
Died        8
Age         8
Occupation  8
dtype: int64
```

• 시리즈 다루기 – 응용

- 데이터 별 개수 확인하기

> value_counts() 메소드 사용

```
scientists['Occupation'].value_counts()
```

Out :

```
Chemist              2
Statistician         1
Nurse                1
Biologist            1
Physician            1
Computer Scientist   1
Mathematician        1
Name: Occupation, dtype: int64
```

• 시리즈 다루기 – 응용

- 기초 통계 메소드 사용하기

> mean(), min(), max(), std() 메소드 사용

```
print(ages.mean())
print(ages.min())
print(ages.max())
print(ages.quantile(q = 0.25))
print(ages.median())
```

Out : 59.125
37
90
44.0
58.5

• 시리즈 다루기 – 응용

- 기초 통계 메소드 사용하기

> describe() 메소드 사용

```
print(ages.describe())
```

Out :

```
count     8.000000
mean     59.125000
std      18.325918
min      37.000000
25%      44.000000
50%      58.500000
75%      68.750000
max      90.000000
Name: Age, dtype: float64
```

• 시리즈 다루기 – 응용

- 데이터 변경하기

> replace() 메소드 사용

```
scientists.replace(37, 40)
```

Out :

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	40	Chemist

> 인덱싱 사용

```
scientists.loc[0, 'Age'] = 40
scientists
```

Out :

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	40	Chemist

• 시리즈 다루기 – 응용

- 데이터에서 표본 추출

> sample(n) 메소드 사용

```
scientists.sample(3)
```

Out :

	Name	Born	Died	Age	Occupation
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist

> sample(frac) 메소드 사용

```
scientists.sample(frac=0.2)
```

Out :

	Name	Born	Died	Age	Occupation
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist

• 시리즈 다루기 – 응용

- 데이터 정렬

> sort_index() 메소드 사용

```
scientists.sort_index(ascending = True)
```

Out :

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist
5	John Snow	1813-03-15	1858-06-16	45	Physician
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

• 시리즈 다루기 – 응용

- 데이터 정렬

> sort_values() 메소드 사용

```
scientists.sort_values('Age', ascending = True)
```

Out :

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
5	John Snow	1813-03-15	1858-06-16	45	Physician
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse

• 시리즈 다루기 – 응용

- 시리즈를 데이터프레임으로 변환

> to_frame() 메소드 사용

```
ages.to_frame()
```

Out :

	Age
0	37
1	61
2	90
3	66
4	56
5	45
6	41
7	77

• 시리즈 다루기 – 응용

- 브로드캐스팅 (Broadcasting)

> 시리즈와 같이 여러 개의 값을 가진 데이터(벡터)와 단순 크기를 나타내는 데이터(스칼라) 간의 연산을 지원하는 것

```
ages + ages
```

Out :

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

```
ages + 100
```

Out :

```
0    137
1    161
2    190
3    166
4    156
5    145
6    141
7    177
Name: Age, dtype: int64
```

• 시리즈 다루기 – 응용

> 길이가 서로 다른 벡터를 연산하는 경우
  - 같은 인덱스의 값만 계산

```
ages + pd.Series([1, 100])
```

Out :

```
0     38.0
1    161.0
2      NaN
3      NaN
4      NaN
5      NaN
6      NaN
7      NaN
dtype: float64
```

0, 1 인덱스만 계산되고 나머지는 누락값(NaN)으로 처리

• 시리즈 다루기 – 응용

> sort_index([ascending = False])

```
rev_ages = ages.sort_index(ascending=False)
rev_ages
```

Out :

```
7    77
6    41
5    45
4    56
3    66
2    90
1    61
0    37
Name: Age, dtype: int64
```

• 시리즈 다루기 – 응용

> 정렬 후 연산

```
ages * 2
```

Out :

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

=

```
ages + rev_ages
```

Out :

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

• 데이터프레임 다루기

> 브로드캐스팅 (Broadcasting)
- 시리즈와 같이 모든 요소를 대상으로 연산
- 2를 곱하면 숫자는 2를 곱하고 문자열은 2배로 늘어남

```
scientists * 2
```

Out :

	Name	Born	Died	Age	Occupation
0	Rosaline FranklinRosaline Franklin	1920-07-251920-07-25	1958-04-161958-04-16	74	ChemistChemist
1	William GossetWilliam Gosset	1876-06-131876-06-13	1937-10-161937-10-16	122	StatisticianStatistician
2	Florence NightingaleFlorence Nightingale	1820-05-121820-05-12	1910-08-131910-08-13	180	NurseNurse
3	Marie CurieMarie Curie	1867-11-071867-11-07	1934-07-041934-07-04	132	ChemistChemist
4	Rachel CarsonRachel Carson	1907-05-271907-05-27	1964-04-141964-04-14	112	BiologistBiologist
5	John SnowJohn Snow	1813-03-151813-03-15	1858-06-161858-06-16	90	PhysicianPhysician
6	Alan TuringAlan Turing	1912-06-231912-06-23	1954-06-071954-06-07	82	Computer ScientistComputer Scientist
7	Johann GaussJohann Gauss	1777-04-301777-04-30	1855-02-231855-02-23	154	MathematicianMathematician

• 데이터프레임 다루기

> Boolean Array

- 데이터프레임의 Age 열의 평균보다 높은 데이터 출력

```
scientists = pd.read_csv('data/scientists.csv')
scientists[scientists['Age'] > scientists['Age'].mean()]
```

Out :

	Name	Born	Died	Age	Occupation
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

• 시리즈와 데이터프레임의 데이터 처리하기

> **열의 자료형 바꾸기**

- scientists 데이터프레임의 Age 열의 자료형 변환

```
print(scientists['Age'].dtype)
```

Out : int64

- Age 열의 int를 float 자료형으로 변경

```
scientists['Age'].astype(float)
```

Out :

```
0    37.0
1    61.0
2    90.0
3    66.0
4    56.0
5    45.0
6    41.0
7    77.0
Name: Age, dtype: float64
```

• 시리즈와 데이터프레임의 데이터 처리하기

> 열의 자료형 바꾸기

- scientists 데이터프레임의 Born과 Died 열의 자료형 확인

```
print(scientists['Born'].dtype)
print(scientists['Died'].dtype)
```

Out : object
object

- Born 열의 날짜 형식 문자열을 datetime 자료형으로 변경

```
born_datetime = pd.to_datetime(scientists['Born'], format='%Y-%m-%d')
print(born_datetime)
```

Out :

```
0    1920-07-25
1    1876-06-13
2    1820-05-12
3    1867-11-07
4    1907-05-27
5    1813-03-15
6    1912-06-23
7    1777-04-30
Name: Born, dtype: datetime64[ns]
```

• 시리즈와 데이터프레임의 데이터 처리하기

> **열의 자료형 바꾸기**

- Died 열의 날짜 형식 문자열을 datetime 자료형으로 변경

```
died_datetime = pd.to_datetime(scientists['Died'], format='%Y-%m-%d')
print(died_datetime)
```

Out :

```
0   1958-04-16
1   1937-10-16
2   1910-08-13
3   1934-07-04
4   1964-04-14
5   1858-06-16
6   1954-06-07
7   1855-02-23
Name: Died, dtype: datetime64[ns]
```

- astype으로 변경

```
print(scientists['Died'].astype('datetime64'))
```

Out :

```
0   1958-04-16
1   1937-10-16
2   1910-08-13
3   1934-07-04
4   1964-04-14
5   1858-06-16
6   1954-06-07
7   1855-02-23
Name: Died, dtype: datetime64[ns]
```

• 시리즈와 데이터프레임의 데이터 처리하기

> 열의 자료형 바꾸기

- datetime으로 바뀐 2개의 자료를 새로운 열로 추가

```
scientists['born_dt'], scientists['died_dt']=(born_datetime, died_datetime)
scientists.head()
```

Out :

	Name	Born	Died	Age	Occupation	born_dt	died_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16
1	William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14

• 시리즈와 데이터프레임의 데이터 처리하기

> datetime 시간 형식 지정자

지정자	설명	결과	지정자	설명	결과
%Y	년(4자리)	2002	%y	년(2자리)	02
%m	월	01-12	%B, %b	월(영어)	January, Jan
%d	일	01-31			
%H	시(24시간)	00-23	%I	시(12시간)	01-12
%M	분	00-59			
%S	초	00-59	%u	요일	1-7(월-일)
%w	요일	0-6(일-토)	%A, %a	요일(영어)	Sunday, Sun
%p	오전, 오후	AM, PM	%f	마이크로초	000000-999999
%z	UTC 차이	UTC+0900	%Z	기준 지역명	UTC, EST, ...
%j	올해 지난 일	001-366	%U	올해 지난 주	00-53
%c, %x	날짜와 시간				

• 시리즈와 데이터프레임의 데이터 처리하기

> **열의 자료형 바꾸기**
  - 과학자 삶의 시간 계산하기 (died_dt – born_dt)

```
scientists['age_days_dt']=scientists['died_dt']-scientists['born_dt']
scientists['age_days_dt']
```

Out :

```
0    13779 days
1    22404 days
2    32964 days
3    24345 days
4    20777 days
5    16529 days
6    15324 days
7    28422 days
Name: age_days_dt, dtype: timedelta64[ns]
```

• 시리즈와 데이터프레임의 데이터 처리하기

> **열의 자료형 바꾸기**
  - datetime에서 연도, 월, 일, 분기 데이터 추출하기(dt 접근자)

```
print(scientists['born_dt'].dt.year)
print(scientists['born_dt'].dt.month)
print(scientists['born_dt'].dt.day)
print(scientists['born_dt'].dt.quarter)
```

Out :

```
0    1920
1    1876
2    1820
3    1867
4    1907
5    1813
6    1912
7    1777
Name: born_dt, dtype: int64
0     7
1     6
2     5
3    11
4     5
5     3
6     6
7     4
Name: born_dt, dtype: int64
0    25
1    13
2    12
3     7
4    27
5    15
6    23
7    30
Name: born_dt, dtype: int64
0    3
1    2
2    2
3    4
4    2
5    1
6    2
7    2
Name: born_dt, dtype: int64
```

• 시리즈와 데이터프레임의 데이터 처리하기

> 열의 자료형 바꾸기

- to_numeric 메소드로 error 제어

```
scientists.loc[[1, 3, 5, 7], 'Age'] = 'missing'
print(scientists['Age'].dtype)
scientists
```

Out : object

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	missing	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	missing	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	missing	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	missing	Mathematician	1777-04-30	1855-02-23	28422 days

• 시리즈와 데이터프레임의 데이터 처리하기

> 열의 자료형 바꾸기

- to_numeric 메소드로 error 제어

```
scientists['Age'].astype(int)
```

Out :

```
ValueError: invalid literal for int() with base 10: 'missing'
```

옵션	설명
raise	숫자로 변환할 수 없는 값이 있으면 오류 발생 (기본값)
coerce	숫자로 변환할 수 없는 값을 누락값으로 지정
ignore	아무 작업도 하지 않음

• 시리즈와 데이터프레임의 데이터 처리하기

> 열의 자료형 바꾸기

- to_numeric 메소드로 error 제어

```
pd.to_numeric(scientists['Age'], errors = 'coerce')
```

Out :

```
0    37.0
1     NaN
2    90.0
3     NaN
4    56.0
5     NaN
6    41.0
7     NaN
Name: Age, dtype: float64
```

```
pd.to_numeric(scientists['Age'], errors = 'ignore')
```

Out :

```
0        37
1   missing
2        90
3   missing
4        56
5   missing
6        41
7   missing
Name: Age, dtype: object
```

• 시리즈와 데이터프레임의 데이터 처리하기

> **데이터프레임의 열 삭제하기**

- drop('열 이름', axis=1)

```
scientists.drop('Age', axis=1).head(2)
```

Out :

	Name	Born	Died	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	Statistician	1876-06-13	1937-10-16	22404 days

- drop('인덱스', axis=0)

```
scientists.drop(0, axis=0).head(2)
```

Out :

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
1	William Gosset	1876-06-13	1937-10-16	missing	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	32964 days

• 시리즈와 데이터프레임의 데이터 처리하기

> **데이터프레임의 열 삭제하기**
  - 여러 개 한번에 삭제하기

```
scientists.drop(['Age','Occupation'],axis=1).head(2)
```

Out :

	Name	Born	Died	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	1876-06-13	1937-10-16	22404 days

• 데이터 저장하고 불러오기

> pickle
- 바이너리 형태로 직렬화 하여 저장
- 적은 용량으로 저장 가능

```
names = scientists['Name']
names.to_pickle('scientists_names_series.pickle')
```

```
scientists.to_pickle('scientists_df.pickle')
```

scientists_df.pickle

scientists_names_series.pickle

• 데이터 저장하고 불러오기

> pickle
  - 바이너리 형태로 저장된 데이터 불러오기

```
pd.read_pickle('scientists_names_series.pickle')
```

Out :

```
0       Rosaline Franklin
1          William Gosset
2    Florence Nightingale
3            Marie Curie
4           Rachel Carson
5               John Snow
6             Alan Turing
7            Johann Gauss
Name: Name, dtype: object
```

• 데이터 저장하고 불러오기

> pickle
  - 바이너리 형태로 저장된 데이터 불러오기

```
pd.read_pickle('scientists_df.pickle')
```

Out :

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	66	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	56	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	41	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	90	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	37	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	61	Mathematician	1777-04-30	1855-02-23	28422 days

• 데이터 저장하고 불러오기

> CSV, TSV로 저장하기

```
scientists.to_csv('data/scientists_df.csv')
```

Out :

```
,Name,Born,Died,Age,Occupation,born_dt,died_dt,age_days_dt
0,Rosaline Franklin,1920-07-25,1958-04-16,66,Chemist,1920-07-25,1958-04-16,13779 days
1,William Gosset,1876-06-13,1937-10-16,56,Statistician,1876-06-13,1937-10-16,22404 days
2,Florence Nightingale,1820-05-12,1910-08-13,41,Nurse,1820-05-12,1910-08-13,32964 days
3,Marie Curie,1867-11-07,1934-07-04,77,Chemist,1867-11-07,1934-07-04,24345 days
4,Rachel Carson,1907-05-27,1964-04-14,90,Biologist,1907-05-27,1964-04-14,20777 days
5,John Snow,1813-03-15,1858-06-16,45,Physician,1813-03-15,1858-06-16,16529 days
6,Alan Turing,1912-06-23,1954-06-07,37,Computer Scientist,1912-06-23,1954-06-07,15324 days
7,Johann Gauss,1777-04-30,1855-02-23,61,Mathematician,1777-04-30,1855-02-23,28422 days
```

```
scientists.to_csv('data/scientists_df.tsv', sep='₩t')
```

Out :

```
	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	66	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	56	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	41	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	90	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	37	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	61	Mathematician	1777-04-30	1855-02-23	28422 days
```

• 데이터 저장하고 불러오기

> CSV, TSV 데이터프레임으로 불러오기

```
pd.read_csv('data/scientists_df.csv')
```

Out :

	Unnamed: 0	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	0	Rosaline Franklin	1920-07-25	1958-04-16	66	Chemist	1920-07-25	1958-04-16	13779 days
1	1	William Gosset	1876-06-13	1937-10-16	56	Statistician	1876-06-13	1937-10-16	22404 days
2	2	Florence Nightingale	1820-05-12	1910-08-13	41	Nurse	1820-05-12	1910-08-13	32964 days
3	3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
4	4	Rachel Carson	1907-05-27	1964-04-14	90	Biologist	1907-05-27	1964-04-14	20777 days
5	5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
6	6	Alan Turing	1912-06-23	1954-06-07	37	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	7	Johann Gauss	1777-04-30	1855-02-23	61	Mathematician	1777-04-30	1855-02-23	28422 days

```
pd.read_csv('data/scientists_df.tsv', sep='₩t')
```

Out :

	Unnamed: 0	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	0	Rosaline Franklin	1920-07-25	1958-04-16	66	Chemist	1920-07-25	1958-04-16	13779 days
1	1	William Gosset	1876-06-13	1937-10-16	56	Statistician	1876-06-13	1937-10-16	22404 days
2	2	Florence Nightingale	1820-05-12	1910-08-13	41	Nurse	1820-05-12	1910-08-13	32964 days
3	3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
4	4	Rachel Carson	1907-05-27	1964-04-14	90	Biologist	1907-05-27	1964-04-14	20777 days
5	5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
6	6	Alan Turing	1912-06-23	1954-06-07	37	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	7	Johann Gauss	1777-04-30	1855-02-23	61	Mathematician	1777-04-30	1855-02-23	28422 days

• 데이터 저장하고 불러오기

> 특정 컬럼을 인덱스로 지정하여 불러오기

```
pd.read_csv('data/scientists_df.csv', index_col = 0)
```

Out :

Name	Unnamed: 0	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
Rosaline Franklin	0	1920-07-25	1958-04-16	66	Chemist	1920-07-25	1958-04-16	13779 days
William Gosset	1	1876-06-13	1937-10-16	56	Statistician	1876-06-13	1937-10-16	22404 days
Florence Nightingale	2	1820-05-12	1910-08-13	41	Nurse	1820-05-12	1910-08-13	32964 days
Marie Curie	3	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
Rachel Carson	4	1907-05-27	1964-04-14	90	Biologist	1907-05-27	1964-04-14	20777 days
John Snow	5	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
Alan Turing	6	1912-06-23	1954-06-07	37	Computer Scientist	1912-06-23	1954-06-07	15324 days
Johann Gauss	7	1777-04-30	1855-02-23	61	Mathematician	1777-04-30	1855-02-23	28422 days

```
pd.read_csv('data/scientists_df.csv', index_col = 'Name')
```

Out :

Name	Unnamed: 0	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
Rosaline Franklin	0	1920-07-25	1958-04-16	66	Chemist	1920-07-25	1958-04-16	13779 days
William Gosset	1	1876-06-13	1937-10-16	56	Statistician	1876-06-13	1937-10-16	22404 days
Florence Nightingale	2	1820-05-12	1910-08-13	41	Nurse	1820-05-12	1910-08-13	32964 days
Marie Curie	3	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
Rachel Carson	4	1907-05-27	1964-04-14	90	Biologist	1907-05-27	1964-04-14	20777 days
John Snow	5	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
Alan Turing	6	1912-06-23	1954-06-07	37	Computer Scientist	1912-06-23	1954-06-07	15324 days
Johann Gauss	7	1777-04-30	1855-02-23	61	Mathematician	1777-04-30	1855-02-23	28422 days

• 데이터 저장하고 불러오기

> 인덱스를 제외하고 저장하기

```
scientists.to_csv('data/scientists_notindex.csv', index=False)
pd.read_csv('data/scientists_notindex.csv')
```

Out :

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23	28422 days

• 데이터 저장하고 불러오기

> datetime 데이터 불러오면서 지정하기

```
scientists = pd.read_csv('scientists.csv', parse_dates=['Born', 'Died'])
scientists.info()
```

Out :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Name        8 non-null      object
 1   Born        8 non-null      datetime64[ns]
 2   Died        8 non-null      datetime64[ns]
 3   Age         8 non-null      int64
 4   Occupation  8 non-null      object
dtypes: datetime64[ns](2), int64(1), object(2)
memory usage: 448.0+ bytes
```