1 파이썬 데이터분석 특화 데이터 형태: Pandas

- **목적:** 데이터 과학자를 위해 테이블형태로 데이터를 다룰 수 있게 해주는 가장 많이 사용되는 패키지(Python용 엑셀)

  - Wes McKinney가 투자운용 회사인 AQR에 재직중이던 2008년 초에 개발됨
  - NumPy를 기반으로 업데이트 개발
  - Python을 사용한 데이터 분석 및 관리의 표준으로 없어서는 안될 도구
  - Python 기반 데이터 전문가, Kaggle 도전 또는 데이터 프로세스를 자동화가 필요한 사람에게 필수적
  - 일반인이 데이터분석을 접하기 쉽게 만들어준 결정적인 라이브러리
  - Pandas만으로도 충분히 데이터 분석이 가능할 정도로 고수준의 함수들을 내장

    - Series는 1차원의 배열같은 구조의 데이터를 저장하기 위한 Python 데이터 형태
    - DataFrame는 Table형식의 2D 데이터를 저장하기위한 Python 데이터 형태
    - DataFrame은 복수의 Series가 합쳐진 것으로 각 Series는 동일한 자료형을 가짐
    - 데이터에는 여러 행과 열이있을 수 있으며, 각 행은 데이터 샘플이고 각 열은 샘플(행)을 설명하는 변수
    - 일반적으로 Excel 데이터 세트와 유사하나 DataFrames는 누락 된 값을 피하고 행이나 열 사이에 간격과 빈 값이 없음
    - 다양한 형태의 데이터를 받아 다양한 통계, 시각화 함수를 제공

1.1 설치 및 호출

In [1]:
```
# 주피터 노트북에서 Pandas 설치
# Anaconda 설치하면 기 설치되어 있어서 진행할 필요 없음
!pip install pandas
```
executed in 2.89s, finished 15:04:14 2021-11-19

```
Requirement already satisfied: pandas in c:₩users₩kk₩anaconda3₩lib₩site-packages (1.3.2)
```

```
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:₩Users₩KK₩anaconda3₩python.exe -m pip install --upgrade pip' command.
```

```
Requirement already satisfied: numpy>=1.17.3 in c:₩users₩kk₩anaconda3₩lib₩site-packages (from pandas) (1.19.5)
Requirement already satisfied: python-dateutil>=2.7.3 in c:₩users₩kk₩anaconda3₩lib₩site-packages (from pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:₩users₩kk₩appdata₩roaming₩python₩python38₩site-packages (from pandas) (2019.3)
Requirement already satisfied: six>=1.5 in c:₩users₩kk₩anaconda3₩lib₩site-packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

In [2]:
```
# Pandas 사용을 위해 패키지 불러오기
# import 패키지명 as 닉네임
# 관례적으로 pd라는 약자를 많이 사용
import pandas as pd
```
executed in 466ms, finished 15:04:14 2021-11-19

1.2 Series & DataFrame

- 엑셀에 익숙한 사용자를 위해 제작 된 테이블형태의 데이터 구조
- 엑셀과 같은 표 형태의 데이터는 **데이터프레임(DataFrame)**, 데이터프레임의 한 열을 **시리즈(Series)** 라고 칭함

In [3]:
```
# Series 생성하기
import pandas as pd
ds = pd.Series([1,2,3,4,5])
ds
```
executed in 14ms, finished 15:04:14 2021-11-19

Out[3]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [4]:
```
# Series의 값을 배열로 변환하기
ds.values
```
executed in 14ms, finished 15:04:14 2021-11-19

Out[4]:
```
array([1, 2, 3, 4, 5], dtype=int64)
```

In [5]:
```
# Series의 행의 위치를 반환하기
ds.index
```
executed in 13ms, finished 15:04:14 2021-11-19

Out[5]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [6]:
```
# Series 생성하되 index를 내가 원하는데로 변환하기
ds = pd.Series([1,2,3,4,5], index=['a','b','c','d','e'])
ds
```

executed in 13ms, finished 15:04:14 2021-11-19

Out[6]:
```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

In [7]:
```
# Series에서 c인덱스의 행값 출력
ds['c']
```
executed in 13ms, finished 15:04:14 2021-11-19

Out[7]:
```
3
```

In [8]:
```
# Series에서도 3보다 큰지 논리연산 가능
ds>3
```
executed in 13ms, finished 15:04:14 2021-11-19

Out[8]:
```
a    False
b    False
c    False
d     True
e     True
dtype: bool
```

In [9]:
```
# Series에서도 3보다 큰지 논리값 기준 True인 값을 출력
ds[ds>3]
```
executed in 13ms, finished 15:04:14 2021-11-19

Out[9]:
```
d    4
e    5
dtype: int64
```

In [10]:
```
# 개별값의 곱하기 2 출력
ds*2
```
executed in 14ms, finished 15:04:14 2021-11-19

Out[10]:
```
a     2
b     4
c     6
d     8
e    10
dtype: int64
```

In [11]:
```
# 개별 값이 비어있는(Null)인지 여부 논리값 출력
ds.isnull()
```
executed in 12ms, finished 15:04:14 2021-11-19

Out[11]:
```
a    False
b    False
c    False
d    False
e    False
dtype: bool
```

In [12]:
```
# 개별 값이 비어있지않은지(Full) 여부 논리값 출력
ds.notnull()
```
executed in 14ms, finished 15:04:14 2021-11-19

Out[12]:
```
a    True
b    True
c    True
d    True
e    True
dtype: bool
```

In [13]:
```
# 개별 값이 몇개 비어있는(Null)인지 출력
ds.isnull().sum()
```
executed in 13ms, finished 15:04:15 2021-11-19

Out[13]:
```
0
```

In [14]:
```
# 딕셔너리 데이터 생성
df = {'column1':[1,2,3,4,5],
      'another_column':['this', 'column', 'has', 'strings', 'inside!'],
      'float_column':[0.1, 0.5, 33, 48, 42.5555],
      'binary_column':[True, False, True, True, False]}
df
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[14]:
```
{'column1': [1, 2, 3, 4, 5],
 'another_column': ['this', 'column', 'has', 'strings', 'inside!'],
 'float_column': [0.1, 0.5, 33, 48, 42.5555],
 'binary_column': [True, False, True, True, False]}
```

In [15]:
```
# 딕셔너리를 DataFrame 형태로 변환하기
# DataFrame의 약자로 형식적으로 df 변수명을 사용
df = pd.DataFrame(df)
df
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[15]:

	column1	another_column	float_column	binary_column
0	1	this	0.1000	True
1	2	column	0.5000	False

2	3	has	33.0000	True
3	4	strings	48.0000	True
4	5	inside!	42.5555	False

In [16]:
```
# DataFrame의 새로운 열을 만들고 100으로 값을 채우기
df['column_test'] = 100
df
```
executed in 12ms, finished 15:04:15 2021-11-19

Out[16]:

	column1	another_column	float_column	binary_column	column_test
0	1	this	0.1000	True	100
1	2	column	0.5000	False	100
2	3	has	33.0000	True	100
3	4	strings	48.0000	True	100
4	5	inside!	42.5555	False	100

In [17]:
```
# DataFrame의 새로운 열을 만들고 numpy.arange 사용해서 0~4로 값을 채우기
import numpy as np
df['seq_test'] = np.arange(5)
df
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[17]:

	column1	another_column	float_column	binary_column	column_test	seq_test
0	1	this	0.1000	True	100	0
1	2	column	0.5000	False	100	1
2	3	has	33.0000	True	100	2
3	4	strings	48.0000	True	100	3
4	5	inside!	42.5555	False	100	4

In [18]:
```
# DataFrame의 특정 열 삭제하기
del df['column_test']
df
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[18]:

	column1	another_column	float_column	binary_column	seq_test
0	1	this	0.1000	True	0
1	2	column	0.5000	False	1
2	3	has	33.0000	True	2
3	4	strings	48.0000	True	3
4	5	inside!	42.5555	False	4

In [19]:
```
# 전치행렬: 대각선 기준 회전하기
df.T
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[19]:

	0	1	2	3	4
column1	1	2	3	4	5
another_column	this	column	has	strings	inside!
float_column	0.1	0.5	33.0	48.0	42.5555
binary_column	True	False	True	True	False
seq_test	0	1	2	3	4

In [20]:
```
# DataFrame의 열 이름 출력하기
df.columns
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[20]:
```
Index(['column1', 'another_column', 'float_column', 'binary_column',
       'seq_test'],
      dtype='object')
```

In [21]:
```
# DataFrame의 행 이름 출력하기
df.index
```
executed in 14ms, finished 15:04:15 2021-11-19

Out[21]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [22]:
```
# DataFrame의 행과 열이름을 제외한 값들만 배열로 출력하기
df.values
```
executed in 12ms, finished 15:04:15 2021-11-19

Out[22]:
```
array([[1, 'this', 0.1, True, 0],
       [2, 'column', 0.5, False, 1],
       [3, 'has', 33.0, True, 2],
       [4, 'strings', 48.0, True, 3],
       [5, 'inside!', 42.5555, False, 4]], dtype=object)
```

1.3 데이터 불러오기(Data Loading)

> **"pandas는 다양한 데이터 파일 형태를 지원(불러올수 있음)하며, 주로 파일 확장자가 csv, xlsx, sql 등의 파일 로딩"**

```
read_csv()
read_excel()
read_sql()
```

1) CSV는 무엇인가?

- CSV(쉼표 구분 값)파일은 숫자와 텍스트 데이터를 저장하는 일반적인 파일 형식
- Python을 사용하여 CSV 파일을 불러오고 핸들링하고 출력하는 기능은 모든 데이터 과학자 또는 비즈니스 분석가에게 핵심 기술

2) 데이터 위치확인

```
# 현재 Jupyter Notebook 작업 폴더위치 출력
# 내장함수 사용
!dir
# 외장함수 사용
import os
os.getcwd()
```

In [23]:
```
# 내장함수 사용
!dir
```
executed in 60ms, finished 15:04:15 2021-11-19

```
 D 드라이브의 볼륨: Data
 볼륨 일련 번호: FA02-E3DF

 D:₩DataScience₩Lecture₩[DataScience] 디렉터리

2021-11-19  오후 03:03    <DIR>          .
2021-11-19  오후 03:03    <DIR>          ..
2021-11-18  오후 02:10    <DIR>          .ipynb_checkpoints
2021-11-07  오후 06:13    <DIR>          Data
2021-11-08  오후 06:30    <DIR>          Document
2021-11-07  오후 06:13    <DIR>          Image
2021-11-07  오후 06:13         5,244,523 Lecture0_DataAnalysis_RealDataBusiness_KK.pptx
2021-11-07  오후 06:13        15,759,460 Lecture0_DataScience_DigitalTransformation_KK.pptx
2021-11-07  오후 06:13         4,971,823 Lecture0_GlobalBusinessTrend_DataScienceAI_KK.pptx
2021-11-07  오후 06:13         8,748,484 Lecture0_RealDataBusiness_SamsungDigitalEconomy_KK.pptx
2021-11-07  오후 06:13            29,384 Lecture0_데이터싸이언스문화.docx
2021-11-07  오후 06:13            17,449 Lecture0_하드웨어_소프트웨어_데이터분석.docx
2021-11-15  오후 05:31            34,333 Lecture1-1_Basic_Programming_KK.ipynb
2021-11-07  오후 06:13            13,218 Lecture1-2_Basic_RealDataAnalysis_KK.ipynb
2021-11-07  오후 06:13            25,588 Lecture1-3_Basic_MathStatistics_KK.ipynb
```

In [24]:
```
# 외장함수 사용
import os
os.getcwd()
```
executed in 13ms, finished 15:04:15 2021-11-19

Out[24]:
```
'D:₩₩DataScience₩₩Lecture₩₩[DataScience]'
```

3) 데이터 불러오기

- **절대경로: "컴퓨터"** 기준 로딩할 데이터의 폴더위치로 어디서 분석 하든 바뀌지 않을 경로
  - 데이터폴더와 작업공간을 분리시킬 수 있음
  - 데이터위치가 변경되면 코드를 변경해야 함
  - 작업공간이 어디에 있든 실행됨
- **상대경로: "작업파일"** 기준 로딩할 데이터의 폴더위치로 분석파일 위치에 따라 바뀔수 있는 경로
  - 데이터폴더와 작업공간을 함께 위치시킴
  - 데이터위치가 변경되어도 코드를 변경할 필요가 없음
  - 데이터를 포함하지 않으면 실행되지 않음

In [25]:
```
# 절대경로 방식으로 수동 파일경로 입력
# 절대경로는 PC내에서 실제 존재하는 위치
# 강의내 설명경로가 아닌 본인 PC상 파일의 위치경로로 반영해야 함
import pandas as pd

df_abs = pd.read_csv(r'D:₩DataScience₩Lecture₩[DataScience]₩Data₩FoodAgricultureOrganization₩Food_Agriculture_Organization_UN_Full.c
df_abs
```
executed in 166ms, finished 15:04:15 2021-11-19

Out[25]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
					Maize and			1000									

4	AF	2	Afghanistan	2514	products	5521	Feed	tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

In [26]:
```
# 상대경로 방식으로 수동 파일경로 입력
# 상대경로는 현 Jupyter Notebook에서부터 PC내에서 실제 존재하는 위치까지의 거리
# 강의내 설명경로와 동일해도 진행 가능
import pandas as pd

df_rel = pd.read_csv(r'.₩Data₩FoodAgricultureOrganization₩Food_Agriculture_Organization_UN_Full.csv')
df_rel
```

executed in 163ms, finished 15:04:15 2021-11-19

Out[26]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

In [27]:
```
# 절대경로 폴더 분리 관리
location_abs = r'D:₩DataScience₩Lecture₩[DataScience]₩Data₩FoodAgricultureOrganization'
location_abs
```

executed in 14ms, finished 15:04:15 2021-11-19

Out[27]: 'D:₩₩DataScience₩₩Lecture₩₩[DataScience]₩₩Data₩₩FoodAgricultureOrganization'

In [28]:
```
# 상대경로 폴더 분리 관리
location_rel = r'.₩Data₩FoodAgricultureOrganization'
location_rel
```

executed in 14ms, finished 15:04:15 2021-11-19

Out[28]: '.₩₩Data₩₩FoodAgricultureOrganization'

In [29]:
```
# 현재경로와 상대경로를 사용하여 절대경로 폴더 생성
os.getcwd() + location_rel[1:]
```

executed in 13ms, finished 15:04:15 2021-11-19

Out[29]: 'D:₩₩DataScience₩₩Lecture₩₩[DataScience]₩₩Data₩₩FoodAgricultureOrganization'

In [30]:
```
# 비교해보면 같음
os.getcwd() + location_rel[1:] == location_abs
```

executed in 14ms, finished 15:04:15 2021-11-19

Out[30]: True

In [31]:
```
# 절대경로와 파일이름 결합
# 파이썬이 폴더를 인식하는 구분자로 함께 결합
file_name = 'Food_Agriculture_Organization_UN_Full.csv'
location_abs + '₩₩' + file_name
```

executed in 14ms, finished 15:04:15 2021-11-19

Out[31]: 'D:₩₩DataScience₩₩Lecture₩₩[DataScience]₩₩Data₩₩FoodAgricultureOrganization₩₩Food_Agriculture_Organization_UN_Full.csv'

In [32]:
```
# 구분자는 PC환경과 OS환경에따라 다르기 때문에 사람마다 다를 수 있음
# 구분자를 자동으로 인식하는 방법
os.path.join(location_abs, file_name)
```

executed in 14ms, finished 15:04:15 2021-11-19

Out[32]:
```
'D:₩₩DataScience₩₩Lecture₩₩[DataScience]₩₩Data₩₩FoodAgricultureOrganization₩₩Food_Agriculture_Organization_UN_Full.csv'
```

In [33]:
```
# 비교해보면 같음
location_abs + '₩₩' + file_name == os.path.join(location_abs, file_name)
```

executed in 14ms, finished 15:04:15 2021-11-19

Out[33]:
```
True
```

In [34]:
```
# 절대경로 사용한 데이터 로딩
# pandas 패키지의 read_csv() 함수를 사용하여 파일을 불러들여
# 데이터프레임을 만들고 df_abs 이름의 변수로 저장
import pandas as pd

location_abs = r'D:₩DataScience₩Lecture₩[DataScience]₩Data₩FoodAgricultureOrganization'
file_name = 'Food_Agriculture_Organization_UN_Full.csv'
df_abs = pd.read_csv(os.path.join(location_abs, file_name))
df_abs
```

executed in 148ms, finished 15:04:15 2021-11-19

Out[34]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

In [35]:
```
# 상대경로 사용한 데이터 로딩
import pandas as pd

location_rel = r'.₩Data₩FoodAgricultureOrganization'
file_name = 'Food_Agriculture_Organization_UN_Full.csv'
df_rel = pd.read_csv(os.path.join(location_rel, file_name))
df_rel
```

executed in 209ms, finished 15:04:16 2021-11-19

Out[35]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

In [36]:
```
# 두개의 DataFrame이 동일한지 여부 확인
df_abs.equals(df_rel)
```
executed in 28ms, finished 15:04:16 2021-11-19

Out[36]: True

- Pandas는 DataFrame을 기본적으로 20개의 열과 60개의 행만 표시(나머지 중간부분은 자름)
- DataFrame 출력 제한을 변경하는 옵션 존재
  (https://pandas.pydata.org/pandas-docs/stable/options.html)

> - **pd.options.display.width:** 표시되는 디스플레이의 너비로 둘 이상의 행에 걸쳐 행을 줄 바꿈하는 경우
> - **pd.options.display.max_rows:** 표시되는 최대 행 수
> - **pd.options.display.max_columns:** 표시되는 최대 열 수

In [37]:
```
# 화면에 표시되는 출력 결과물을 갯수를 설정 가능
pd.options.display.max_rows = 10
pd.options.display.max_columns = 20
```
executed in 14ms, finished 15:04:16 2021-11-19

In [38]:
```
df_rel
```
executed in 57ms, finished 15:04:16 2021-11-19

Out[38]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

1.4 데이터의 특성확인(Descriptive Statistics)

> - 데이터를 불러온 후, 가장 처음하는 작업으로 데이터 구조, 형태, 특성을 빠르게 확인하는게 목적
> - **DataFrame.describe()** 함수는 적용되는 모든 변수 또는 그룹의 통계를 빠르게 표시하는 유용한 요약 도구

Function	Description
count	Number of non-null observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Unbiased standard deviation
var	Unbiased variance
sem	Unbiased standard error of the mean
skew	Unbiased skewness (3rd moment)
kurt	Unbiased kurtosis (4th moment)
quantile	Sample quantile (value at %)
cumsum	Cumulative sum
cumprod	Cumulative product

cummax	Cumulative maximum
cummin	Cumulative minimum

In [39]:
```
# 데이터의 행과 열의 수 확인
df_rel.shape
```
executed in 13ms, finished 15:04:16 2021-11-19

Out[39]: (21477, 63)

In [40]:
```
# numpy 자체 함수로도 추정 가능
np.shape(df_rel)
```
executed in 13ms, finished 15:04:16 2021-11-19

Out[40]: (21477, 63)

In [41]:
```
# 데이터의 차원 확인
df_rel.ndim
```
executed in 13ms, finished 15:04:16 2021-11-19

Out[41]: 2

In [42]:
```
# 첫 5개의 샘플 추출
df_rel.head(5)
```
executed in 41ms, finished 15:04:16 2021-11-19

Out[42]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	315.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	178.0

5 rows × 63 columns

In [43]:
```
# 첫 10개의 샘플 추출
df_rel.head(10)
```
executed in 29ms, finished 15:04:16 2021-11-19

Out[43]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	315.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	178.0
5	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	...	231.0	67.0	82.0	67.0	69.0	71.0	82.0
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0	14.0
7	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	...	2.0	1.0	1.0	0.0	0.0	0.0	0.0
8	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	...	276.0	294.0	294.0	260.0	242.0	250.0	192.0
9	AF	2	Afghanistan	2536	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	...	50.0	29.0	61.0	65.0	54.0	114.0	83.0

10 rows × 63 columns

In [44]:
```
# 마지막 5개의 샘플 확인
# 데이터가 잘 가져왔는지 확인 할 때 사용
df_rel.tail(5)
```
executed in 30ms, finished 15:04:16 2021-11-19

Out[44]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	41
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	1
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	2

21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 63 columns

In [45]:
```
# 인덱스로 데이터 확인
# 해당 인덱스번호에 해당하는 데이터 추출(복수 추출 가능)
df_rel.take([10])
```
executed in 29ms, finished 15:04:16 2021-11-19

Out[45]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y201
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.

1 rows × 63 columns

In [46]:
```
# 특정 인덱스 행 추출
df_rel.take([10, 20, 25])
```
executed in 28ms, finished 15:04:16 2021-11-19

Out[46]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
20	AF	2	Afghanistan	2571	Soyabean Oil	5142	Food	1000 tonnes	33.94	67.71	...	6.0	35.0	18.0	21.0	11.0	6.0	15.0	
25	AF	2	Afghanistan	2577	Palm Oil	5142	Food	1000 tonnes	33.94	67.71	...	71.0	69.0	56.0	51.0	36.0	53.0	59.0	

3 rows × 63 columns

In [47]:
```
# 인덱스 값 확인
df_rel.index
```
executed in 13ms, finished 15:04:16 2021-11-19

Out[47]:
```
RangeIndex(start=0, stop=21477, step=1)
```

In [48]:
```
# 컬럼명 확인
df_rel.columns
```
executed in 14ms, finished 15:04:16 2021-11-19

Out[48]:
```
Index(['Area Abbreviation', 'Area Code', 'Area', 'Item Code', 'Item',
       'Element Code', 'Element', 'Unit', 'latitude', 'longitude', 'Y1961',
       'Y1962', 'Y1963', 'Y1964', 'Y1965', 'Y1966', 'Y1967', 'Y1968', 'Y1969',
       'Y1970', 'Y1971', 'Y1972', 'Y1973', 'Y1974', 'Y1975', 'Y1976', 'Y1977',
       'Y1978', 'Y1979', 'Y1980', 'Y1981', 'Y1982', 'Y1983', 'Y1984', 'Y1985',
       'Y1986', 'Y1987', 'Y1988', 'Y1989', 'Y1990', 'Y1991', 'Y1992', 'Y1993',
       'Y1994', 'Y1995', 'Y1996', 'Y1997', 'Y1998', 'Y1999', 'Y2000', 'Y2001',
       'Y2002', 'Y2003', 'Y2004', 'Y2005', 'Y2006', 'Y2007', 'Y2008', 'Y2009',
       'Y2010', 'Y2011', 'Y2012', 'Y2013'],
      dtype='object')
```

In [49]:
```
# 데이터 값의 형태 확인
# object == str
df_rel.dtypes
```
executed in 14ms, finished 15:04:16 2021-11-19

Out[49]:
```
Area Abbreviation     object
Area Code              int64
Area                  object
Item Code              int64
Item                  object
                       ...
Y2009                float64
Y2010                float64
Y2011                float64
Y2012                  int64
Y2013                  int64
Length: 63, dtype: object
```

In [50]:
```
# 데이터의 전반적인 정보를 확인
df_rel.info()
```
executed in 59ms, finished 15:04:16 2021-11-19

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21477 entries, 0 to 21476
Data columns (total 63 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Area Abbreviation  21356 non-null  object
 1   Area Code          21477 non-null  int64
 2   Area               21477 non-null  object
 3   Item Code          21477 non-null  int64
 4   Item               21477 non-null  object
 5   Element Code       21477 non-null  int64
 6   Element            21477 non-null  object
 7   Unit               21477 non-null  object
 8   latitude           21477 non-null  float64
```

```
 8   latitude          21477 non-null  float64
 9   longitude         21477 non-null  float64
 10  Y1961             17938 non-null  float64
 11  Y1962             17938 non-null  float64
 12  Y1963             17938 non-null  float64
 13  Y1964             17938 non-null  float64
 14  Y1965             17938 non-null  float64
 15  Y1966             17938 non-null  float64
 16  Y1967             17938 non-null  float64
 17  Y1968             17938 non-null  float64
 18  Y1969             17938 non-null  float64
 19  Y1970             17938 non-null  float64
 20  Y1971             17938 non-null  float64
 21  Y1972             17938 non-null  float64
 22  Y1973             17938 non-null  float64
 23  Y1974             17938 non-null  float64
 24  Y1975             17938 non-null  float64
 25  Y1976             17938 non-null  float64
 26  Y1977             17938 non-null  float64
 27  Y1978             17938 non-null  float64
 28  Y1979             17938 non-null  float64
 29  Y1980             17938 non-null  float64
 30  Y1981             17938 non-null  float64
 31  Y1982             17938 non-null  float64
 32  Y1983             17938 non-null  float64
 33  Y1984             17938 non-null  float64
 34  Y1985             17938 non-null  float64
 35  Y1986             17938 non-null  float64
 36  Y1987             17938 non-null  float64
 37  Y1988             17938 non-null  float64
 38  Y1989             17938 non-null  float64
 39  Y1990             18062 non-null  float64
 40  Y1991             18062 non-null  float64
 41  Y1992             20490 non-null  float64
 42  Y1993             20865 non-null  float64
 43  Y1994             20865 non-null  float64
 44  Y1995             20865 non-null  float64
 45  Y1996             20865 non-null  float64
 46  Y1997             20865 non-null  float64
 47  Y1998             20865 non-null  float64
 48  Y1999             20865 non-null  float64
 49  Y2000             21128 non-null  float64
 50  Y2001             21128 non-null  float64
 51  Y2002             21128 non-null  float64
 52  Y2003             21128 non-null  float64
 53  Y2004             21128 non-null  float64
 54  Y2005             21128 non-null  float64
 55  Y2006             21373 non-null  float64
 56  Y2007             21373 non-null  float64
 57  Y2008             21373 non-null  float64
 58  Y2009             21373 non-null  float64
 59  Y2010             21373 non-null  float64
 60  Y2011             21373 non-null  float64
 61  Y2012             21477 non-null  int64
 62  Y2013             21477 non-null  int64
dtypes: float64(53), int64(5), object(5)
memory usage: 10.3+ MB
```

In [51]:
```
# 특정 컬럼 값의 type 변경
df_rel['Item Code'] = df_rel['Item Code'].astype(str)
df_rel.dtypes
```

executed in 27ms, finished 15:04:16 2021-11-19

Out[51]:
```
Area Abbreviation     object
Area Code              int64
Area                  object
Item Code             object
Item                  object
                       ...
Y2009                float64
Y2010                float64
Y2011                float64
Y2012                  int64
Y2013                  int64
Length: 63, dtype: object
```

In [52]:
```
# 가로로 값의 합
df_rel.sum(axis=1)
```

executed in 194ms, finished 15:04:16 2021-11-19

```
<ipython-input-52-e58f99a79ee6>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.sum(axis=1)
```

Out[52]:
```
0        138171.65
1         20527.65
2         10814.65
3         13774.65
4         15075.65
             ...
21472     23570.13
21473      6319.13
21474      6330.13
21475      5333.13
21476      5333.13
Length: 21477, dtype: float64
```

In [53]:
```
# 세로로 값의 합
df_rel.sum(axis=0)
```

executed in 344ms, finished 15:04:17 2021-11-19

```
<ipython-input-53-63367971c0ac>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.sum(axis=0)
```

Out[53]:
```
Area Code                                                    2694277
Area            AfghanistanAfghanistanAfghanistanAfghanistanAf...
Item Code       2511280525132513251425142514251725202531253625372 5...
Item            Wheat and productsRice (Milled Equivalent)Barl...
Element Code                                                111931405
                                      ...
Y2009                                                    11211891.0
Y2010                                                    11445072.0
Y2011                                                    11827802.0
Y2012                                                      12039345
Y2013                                                      12361248
Length: 62, dtype: object
```

In [54]:
```
# 세로로 값의 최소값
df_rel.min()
```

executed in 29ms, finished 15:04:17 2021-11-19

```
<ipython-input-54-cfffd0d129e1>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.min()
```

Out[54]:
```
Area Code                        1
Area                   Afghanistan
Item Code                     2511
Item            Alcoholic Beverages
Element Code                  5142
                        ...
Y2009                          0.0
Y2010                          0.0
Y2011                          0.0
Y2012                         -169
Y2013                         -246
Length: 62, dtype: object
```

In [55]:
```
# 세로로 값의 평균
df_rel.mean()
```

executed in 344ms, finished 15:04:17 2021-11-19

```
<ipython-input-55-d1abdc4f184f>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.mean()
```

Out[55]:
```
Area Code        125.449411
Item Code               inf
Element Code    5211.687154
latitude          20.450613
longitude         15.794445
                     ...
Y2009            524.581996
Y2010            535.492069
Y2011            553.399242
Y2012            560.569214
Y2013            575.557480
Length: 58, dtype: float64
```

In [56]:
```
# 세로로 값의 중앙값
df_rel.median()
```

executed in 30ms, finished 15:04:17 2021-11-19

```
<ipython-input-56-c7233c251614>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.median()
```

Out[56]:
```
Area Code        120.00
Item Code       2640.00
Element Code    5142.00
latitude          20.59
longitude         19.15
                  ...
Y2009              7.00
Y2010              7.00
Y2011              8.00
Y2012              8.00
Y2013              8.00
Length: 58, dtype: float64
```

In [57]:
```
# 세로로 값의 분산
df_rel.var()
```

executed in 27ms, finished 15:04:17 2021-11-19

```
<ipython-input-57-10cdc6ca670b>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.var()
```

Out[57]:
```
Area Code       5.309767e+03
Element Code    2.155614e+04
latitude        6.065550e+02
longitude       4.357598e+03
Y1961           3.474960e+06
                    ...
Y2009           3.075744e+07
Y2010           3.273086e+07
Y2011           3.461053e+07
Y2012           3.657771e+07
Y2013           3.866824e+07
Length: 57, dtype: float64
```

In [58]:
```
# 세로로 값의 표준편차
df_rel.std()
```

executed in 29ms, finished 15:04:17 2021-11-19

```
<ipython-input-58-55f95a84086f>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_rel.std()
```

Out[58]:
```
Area Code         72.868149
Element Code     146.820079
latitude          24.628336
longitude         66.012104
Y1961           1864.124336
                    ...
Y2009           5545.939303
Y2010           5721.089425
Y2011           5883.071604
Y2012           6047.950804
Y2013           6218.379479
Length: 57, dtype: float64
```

In [59]:
```
# DataFrame에서 특정 컬럼을 선택, 즉 Seires 선택
# DataFrame[컬럼명]
df_rel['Area Code']
```

executed in 14ms, finished 15:04:17 2021-11-19

Out[59]:
```
0          2
1          2
2          2
3          2
4          2
         ...
21472    181
21473    181
21474    181
21475    181
21476    181
Name: Area Code, Length: 21477, dtype: int64
```

In [60]:
```
# 컬럼명 선택시 괄호를 하나를 사용하면 1차원(Series) 형태, 두개를 사용하면 2차원(DF) 형태로 출력
df_rel[['Area Code']]
```

executed in 14ms, finished 15:04:17 2021-11-19

Out[60]:

	Area Code
0	2
1	2
2	2
3	2
4	2
...	...
21472	181
21473	181
21474	181
21475	181
21476	181

21477 rows × 1 columns

In [61]:
```
# 특정 컬럼의 중복되지 않은 독립된 값 출력
df_rel['Area Code'].unique()
```

executed in 13ms, finished 15:04:17 2021-11-19

Out[61]:
```
array([  2,   3,   4,   7,   8,   9,   1,  10,  11,  52,  12,  16,  14,
        57, 255,  23,  53,  17,  19,  80,  20,  21,  26,  27, 233,  35,
       115,  32,  33,  37,  39,  40,  96, 128,  41, 214,  44,  46,  48,
       107,  98,  49,  50, 167, 116,  54,  72,  55,  56,  58,  59,  60,
        63, 238,  66,  67,  68,  70,  74,  75,  73,  79,  81,  84,  86,
        89,  90, 175,  91,  93,  95,  97,  99, 100, 101, 102, 103, 104,
       105, 106, 109, 110, 112, 108, 114,  83, 118, 113, 120, 119, 121,
       122, 123, 126, 256, 129, 130, 131, 132, 133, 134, 136, 137, 138,
       141, 273, 143, 144,  28, 147, 149, 150, 153, 156, 157, 158, 159,
       162, 221, 165, 166, 169, 170, 171, 173, 174, 117, 146, 183, 185,
       184, 188, 189, 191, 244, 193, 194, 195, 272, 197, 199, 198,  25,
       202, 203,  38, 276, 207, 209, 210, 211, 208, 216, 154, 176, 217,
       220, 222, 223, 213, 226, 230, 225, 229, 215, 231, 234, 235, 155,
       236, 237, 249, 251, 181], dtype=int64)
```

In [62]:
```
# 특정 컬럼의 값들이 대상 값에 포함되었는지 여부 확인
df_rel['Area Code'].isin([203])
```

executed in 14ms, finished 15:04:17 2021-11-19

Out[62]:
```
0        False
1        False
2        False
3        False
4        False
         ...
21472    False
21473    False
21474    False
21475    False
```

```
21476    False
Name: Area Code, Length: 21477, dtype: bool
```

In [63]:
```
# 특정 컬럼의 값들이 대상 값에 포함되었다면 몇개인지 확인
df_rel['Area Code'].isin([203]).sum()
```
executed in 14ms, finished 15:04:17 2021-11-19

Out[63]:
```
150
```

In [64]:
```
# 특정 컬럼의 값들이 대상 값에 포함되었다면 몇개인지 확인
df_rel['Area Code'].isin([106]).sum()
```
executed in 14ms, finished 15:04:17 2021-11-19

Out[64]:
```
148
```

In [65]:
```
# 특정 컬럼의 값들이 무엇이 있고 몇개씩인지 한꺼번에 출력 (빈도확인)
df_rel['Area Code'].value_counts()
```
executed in 14ms, finished 15:04:17 2021-11-19

Out[65]:
```
203    150
106    148
79     147
41     146
110    143
      ...
193     91
213     90
176     86
2       83
122     75
Name: Area Code, Length: 174, dtype: int64
```

In [66]:
```
# 선택한 Series의 기초통계 출력
df_rel['Y2007'].describe()
```
executed in 14ms, finished 15:04:17 2021-11-19

Out[66]:
```
count     21373.000000
mean        508.482104
std        5298.939807
min           0.000000
25%           0.000000
50%           7.000000
75%          80.000000
max      402975.000000
Name: Y2007, dtype: float64
```

In [67]:
```
# 선택한 Series의 기초통계 출력
df_rel['Area'].describe()
```
executed in 22ms, finished 15:04:17 2021-11-19

Out[67]:
```
count     21477
unique      174
top       Spain
freq        150
Name: Area, dtype: object
```

In [68]:
```
# 전체 DataFrame의 기초통계 출력
# 기본적으로 숫자값들에 대한 통계만 포함
df_rel.describe()
```
executed in 147ms, finished 15:04:17 2021-11-19

Out[68]:

	Area Code	Element Code	latitude	longitude	Y1961	Y1962	Y1963	Y1964	Y1965	Y1966	..
count	21477.000000	21477.000000	21477.000000	21477.000000	17938.000000	17938.000000	17938.000000	17938.000000	17938.000000	17938.000000	..
mean	125.449411	5211.687154	20.450613	15.794445	195.262069	200.782250	205.464600	209.925577	217.556751	225.988962	..
std	72.868149	146.820079	24.628336	66.012104	1864.124336	1884.265591	1861.174739	1862.000116	2014.934333	2100.228354	..
min	1.000000	5142.000000	-40.900000	-172.100000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	..
25%	63.000000	5142.000000	6.430000	-11.780000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	..
50%	120.000000	5142.000000	20.590000	19.150000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	..
75%	188.000000	5142.000000	41.150000	46.870000	21.000000	22.000000	23.000000	24.000000	25.000000	26.000000	..
max	276.000000	5521.000000	64.960000	179.410000	112227.000000	109130.000000	106356.000000	104234.000000	119378.000000	118495.000000	..

8 rows × 57 columns

In [69]:
```
# 문자도 포함한 DataFrame 기초통계 출력
# 문자 컬럼에선 다른 통계정보가 출력
df_rel.describe(include='all')
```
executed in 164ms, finished 15:04:17 2021-11-19

Out[69]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	
count	21356	21477.000000	21477	21477	21477	21477.000000	21477	21477	21477.000000	21477.000000	...	21128.000000	21128.000000	2
unique	173	NaN	174	117	115	NaN	2	1	NaN	NaN	...	NaN	NaN	
top	ES	NaN	Spain	2905	Milk - Excluding Butter	NaN	Food	1000 tonnes	NaN	NaN	...	NaN	NaN	
freq	150	NaN	150	347	558	NaN	17528	21477	NaN	NaN	...	NaN	NaN	
mean	NaN	125.449411	NaN	NaN	NaN	5211.687154	NaN	NaN	20.450613	15.794445	...	486.690742	493.153256	

...	...	...	...	...	...	...	...	...	...	...	...	...	...
min	NaN	1.000000	NaN	NaN	NaN	5142.000000	NaN	NaN	-40.900000	-172.100000	...	0.000000	0.000000
25%	NaN	63.000000	NaN	NaN	NaN	5142.000000	NaN	NaN	6.430000	-11.780000	...	0.000000	0.000000
50%	NaN	120.000000	NaN	NaN	NaN	5142.000000	NaN	NaN	20.590000	19.150000	...	6.000000	6.000000
75%	NaN	188.000000	NaN	NaN	NaN	5142.000000	NaN	NaN	41.150000	46.870000	...	75.000000	77.000000
max	NaN	276.000000	NaN	NaN	NaN	5521.000000	NaN	NaN	64.960000	179.410000	...	360767.000000	373694.000000

11 rows × 63 columns

In [70]:
```
# 전치행렬: 대각선 기준 대칭으로 회전
df_rel.describe(include='all').T
```
executed in 183ms, finished 15:04:18 2021-11-19

Out[70]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Area Abbreviation	21356	173	ES	150	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Area Code	21477.0	NaN	NaN	NaN	125.449411	72.868149	1.0	63.0	120.0	188.0	276.0
Area	21477	174	Spain	150	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Item Code	21477	117	2905	347	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Item	21477	115	Milk - Excluding Butter	558	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
Y2009	21373.0	NaN	NaN	NaN	524.581996	5545.939303	0.0	0.0	7.0	83.0	434724.0
Y2010	21373.0	NaN	NaN	NaN	535.492069	5721.089425	0.0	0.0	7.0	83.0	451838.0
Y2011	21373.0	NaN	NaN	NaN	553.399242	5883.071604	0.0	0.0	8.0	86.0	462696.0
Y2012	21477.0	NaN	NaN	NaN	560.569214	6047.950804	-169.0	0.0	8.0	88.0	479028.0
Y2013	21477.0	NaN	NaN	NaN	575.55748	6218.379479	-246.0	0.0	8.0	90.0	489299.0

63 rows × 11 columns

1.5 값 선택과 다루기(Selection and Indexing)

1) Series

> - NumPy 배열과 유사한 원리로 작동되는데 Series는 정수가 아니어도 Indexing이 가능

In [71]:
```
# 데이터 복사 저장
df_series = df_rel['Area Abbreviation'].copy()
df_series
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[71]:
```
0        AF
1        AF
2        AF
3        AF
4        AF
        ..
21472    ZW
21473    ZW
21474    ZW
21475    ZW
21476    ZW
Name: Area Abbreviation, Length: 21477, dtype: object
```

In [72]:
```
# 1번과 3번 인덱스에 해당되는 샘플 출력
df_series[[1,3]]
```
executed in 13ms, finished 15:04:18 2021-11-19

Out[72]:
```
1    AF
3    AF
Name: Area Abbreviation, dtype: object
```

In [73]:
```
# 인덱스의 비교 연산 가능
df_series.index < 10
```
executed in 13ms, finished 15:04:18 2021-11-19

Out[73]:
```
array([ True,  True,  True, ..., False, False, False])
```

In [74]:
```
# 인덱스가 10보다 작은 Series만 추출
df_series[df_series.index < 10]
```
executed in 12ms, finished 15:04:18 2021-11-19

Out[74]:
```
0    AF
1    AF
2    AF
3    AF
4    AF
5    AF
6    AF
7    AF
8    AF
9    AF
Name: Area Abbreviation, dtype: object
```

In [75]:
```
# 10이상 50만의 인덱스 값만 추출
```

```
df_series[1:5]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[75]:
```
1    AF
2    AF
3    AF
4    AF
Name: Area Abbreviation, dtype: object
```

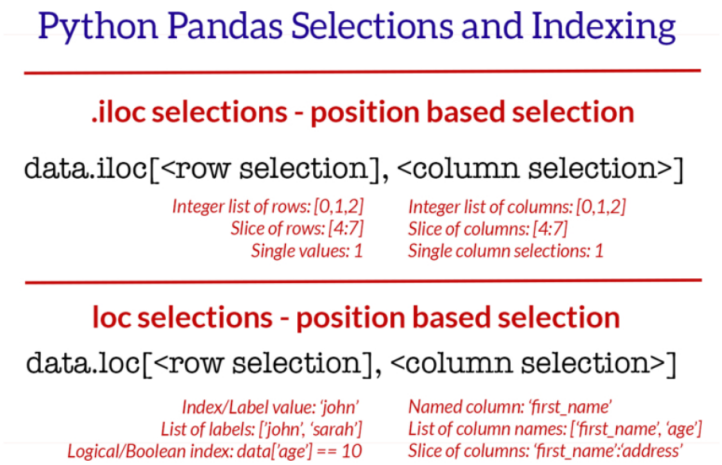
In [76]:
```
# 1이상 3미만의 인덱스 값 변환
df_series[1:3] = 'Test'
df_series
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[76]:
```
0          AF
1        Test
2        Test
3          AF
4          AF
         ...
21472      ZW
21473      ZW
21474      ZW
21475      ZW
21476      ZW
Name: Area Abbreviation, Length: 21477, dtype: object
```

2) DataFrame

- 값을 선택하고 Indexing하는 두 가지 주요 옵션이 있으며 특정 목록이나 단일 값을 선택하여 출력가능

  - **iloc:** 이름과 상관없이 값의 위치를 기준으로 계산
    - 하나의 행이 선택되면 Series를 반환하고 여러 행이 선택되면 DataFrame을 반환
    - 여러 열 또는 여러 행을 선택할 때 선택항목([1:5])에서 선택한 행/열은 1번째 숫자 이상에서 5번째 숫자 미만을 추출
  - **loc:** 값의 위치와 상관없이 실제 이름(인덱스/컬럼명)을 기준으로 계산
    - Labal / Index / Bool / Logical 기반 값을 전달하여 Series나 DataFrame의 값을 반환



In [77]:
```
# 인덱스넘버로 데이터에 접근하는 .iloc[색인]
df_rel.iloc[0]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[77]:
```
Area Abbreviation                    AF
Area Code                             2
Area                          Afghanistan
Item Code                          2511
Item                 Wheat and products
                                 ...
Y2009                           4538.0
Y2010                           4605.0
Y2011                           4711.0
Y2012                             4810
Y2013                             4895
Name: 0, Length: 63, dtype: object
```

In [78]:
```
# 여러개의 인덱스 샘플 출력
df_rel.iloc[0:5]
```
executed in 27ms, finished 15:04:18 2021-11-19

Out[78]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.0

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	315.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	178.0

5 rows × 63 columns

In [79]:
```
# 특정 간격으로 샘플 출력
df_rel.iloc[:10:2]
```
executed in 29ms, finished 15:04:18 2021-11-19

Out[79]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	315.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	178.0
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0	14.0
8	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	...	276.0	294.0	294.0	260.0	242.0	250.0	192.0

5 rows × 63 columns

In [80]:
```
# 첫번째 0, 10, 20 인덱스 샘플 접근 iloc사용
# 두개 이상의 인덱스가 포함되면 괄호를 두개씩 사용
# 하나만 쓰면 인덱스를의미하는지 컬럼명을 의미하는지 모호
df_rel.iloc[[0, 10, 20]]
```
executed in 29ms, finished 15:04:18 2021-11-19

Out[80]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.0
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	AF	2	Afghanistan	2571	Soyabean Oil	5142	Food	1000 tonnes	33.94	67.71	...	6.0	35.0	18.0	21.0	11.0	6.0	15.0

3 rows × 63 columns

In [81]:
```
# 마지막 인덱스 샘플 출력
df_rel.iloc[[-1]]
```
executed in 29ms, finished 15:04:18 2021-11-19

Out[81]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	

1 rows × 63 columns

In [82]:
```
# 특정 컬럼의 위치를 선택하여 출력
# : 기호는 시작부터 끝까지를 의미
# 하나의 컬럼만 선택하면 Series로 출력
df_rel.iloc[:,0]
```
executed in 13ms, finished 15:04:18 2021-11-19

Out[82]:
```
0        AF
1        AF
2        AF
3        AF
4        AF
         ..
21472    ZW
21473    ZW
21474    ZW
21475    ZW
21476    ZW
Name: Area Abbreviation, Length: 21477, dtype: object
```

In [83]:
```
# 하나의 컬럼만 선택하더라도 괄호를 2개를 쓰면 DataFrame로 출력
df_rel.iloc[:,[1]]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[83]:

	Area Code
0	2
1	2
2	2
3	2
4	2
...	...
21472	181
21473	181
21474	181
21475	181
21476	181

21477 rows × 1 columns

In [84]:
```
# 마지막 컬럼위치만 선택하여 출력
df_rel.iloc[:,[-1]]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[84]:

	Y2013
0	4895
1	422
2	360
3	89
4	200
...	...
21472	451
21473	15
21474	40
21475	0
21476	0

21477 rows × 1 columns

In [85]:
```
# 0번째 이상 2번째 미만 컬럼 추출
df_rel.iloc[:,0:2]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[85]:

	Area Abbreviation	Area Code
0	AF	2
1	AF	2
2	AF	2
3	AF	2
4	AF	2
...	...	...
21472	ZW	181
21473	ZW	181
21474	ZW	181
21475	ZW	181
21476	ZW	181

21477 rows × 2 columns

In [86]:
```
# 특정 인덱스와 특정 컬럼의 위치만 추출
# 0, 3, 6, 24번째 행
# 0, 5, 6번째 열
df_rel.iloc[[0,3,6,24],[0,5,6]]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[86]:

	Area Abbreviation	Element Code	Element
0	AF	5142	Food
3	AF	5142	Food
6	AF	5142	Food
24	AF	5142	Food

In [87]:
```
# 0번째 이상 5번째 미만 행
# 5번째 이상 8번째 미만 열
df_rel.iloc[0:5, 5:8]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[87]:

	Element Code	Element	Unit
0	5142	Food	1000 tonnes

1	5142	Food	1000 tonnes
2	5521	Feed	1000 tonnes
3	5142	Food	1000 tonnes
4	5521	Feed	1000 tonnes

In [88]:
```
# 0번째 위치의 샘플이 아니라 인덱스의 이름이 '0'인 샘플 출력
df_rel.loc[0]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[88]:
```
Area Abbreviation                    AF
Area Code                             2
Area                          Afghanistan
Item Code                            2511
Item                 Wheat and products
                                   ...
Y2009                             4538.0
Y2010                             4605.0
Y2011                             4711.0
Y2012                               4810
Y2013                               4895
Name: 0, Length: 63, dtype: object
```

In [89]:
```
# 1번째 위치의 샘플이 아니라 인덱스의 이름이 '1'인 샘플 출력
# 괄호를 2개썼기 때문에 DataFrame 형식 출력
df_rel.loc[[1]]
```
executed in 29ms, finished 15:04:18 2021-11-19

Out[89]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0	

1 rows × 63 columns

In [90]:
```
# 인덱스 이름이 1과 3인 샘플 출력
df_rel.loc[[1,3]]
```
executed in 29ms, finished 15:04:18 2021-11-19

Out[90]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0	
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0	

2 rows × 63 columns

In [91]:
```
# 컬럼명이 'Item', 'Y2013'인 열
# 컬럼명만 사용한 데이터 출력시 loc기호를 사용하면 오류
# 인덱스인지 컬럼명인지 헷갈릴 수 있기 때문에 loc사용할 때는 인덱스만 또는 인덱스+컬럼명만 가능 (컬럼명만 불가)
# df_rel.loc['Item','Y2013']
```
executed in 15ms, finished 15:04:18 2021-11-19

In [92]:
```
df_rel[['Item','Y2013']]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[92]:

	Item	Y2013
0	Wheat and products	4895
1	Rice (Milled Equivalent)	422
2	Barley and products	360
3	Barley and products	89
4	Maize and products	200
...	...	...
21472	Milk - Excluding Butter	451
21473	Fish, Seafood	15
21474	Fish, Seafood	40
21475	Aquatic Products, Other	0
21476	Miscellaneous	0

21477 rows × 2 columns

In [93]:
```
# 인덱스 이름이 1과 3인 행
# 컬럼명이 'Item', 'Y2013'인 열
df_rel.loc[[1,3],['Item','Y2013']]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[93]:

	Item	Y2013
1	Rice (Milled Equivalent)	422
3	Barley and products	89

In [94]:
```
# 인덱스 이름이 1과 3인 행
# 컬럼명이 'Item' 부터 'Y2013'까지인 열
df_rel.loc[[1,3],'Item':'Y2013']
```
executed in 27ms, finished 15:04:18 2021-11-19

Out[94]:

	Item	Element Code	Element	Unit	latitude	longitude	Y1961	Y1962	Y1963	Y1964	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y2011	Y2(
1	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	183.0	182.0	220.0	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0	476.0	
3	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	237.0	237.0	238.0	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0	72.0	

2 rows × 59 columns

In [95]:
```
# 인덱스 이름이 1부터 10까지인 행
# 컬럼명이 'Item' 부터 'Y2013'까지인 열
df_rel.loc[1:10,'Item':'Y2013']
```
executed in 30ms, finished 15:04:18 2021-11-19

Out[95]:

	Item	Element Code	Element	Unit	latitude	longitude	Y1961	Y1962	Y1963	Y1964	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y2011	Y2
1	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	183.0	182.0	220.0	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0	476.0	
2	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76.0	76.0	76.0	76.0	...	58.0	236.0	262.0	263.0	230.0	379.0	315.0	203.0	
3	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	237.0	237.0	238.0	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0	72.0	
4	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210.0	210.0	214.0	216.0	...	120.0	208.0	233.0	249.0	247.0	195.0	178.0	191.0	
5	Maize and products	5142	Food	1000 tonnes	33.94	67.71	403.0	403.0	410.0	415.0	...	231.0	67.0	82.0	67.0	69.0	71.0	82.0	73.0	
6	Millet and products	5142	Food	1000 tonnes	33.94	67.71	17.0	18.0	19.0	20.0	...	15.0	21.0	11.0	19.0	21.0	18.0	14.0	14.0	
7	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	0.0	0.0	0.0	0.0	...	2.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
8	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	111.0	97.0	103.0	110.0	...	276.0	294.0	294.0	260.0	242.0	250.0	192.0	169.0	
9	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	45.0	45.0	45.0	45.0	...	50.0	29.0	61.0	65.0	54.0	114.0	83.0	83.0	
10	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

10 rows × 59 columns

In [96]:
```
# 컬럼명 확인함수 columns에 인덱싱 및 슬라이싱 적용 가능
df_rel.columns[:5]
```
executed in 13ms, finished 15:04:18 2021-11-19

Out[96]:
```
Index(['Area Abbreviation', 'Area Code', 'Area', 'Item Code', 'Item'], dtype='object')
```

In [97]:
```
# 열(컬럼) 필터링 쉽게 적용
df_rel[df_rel.columns[:5]]
```
executed in 13ms, finished 15:04:18 2021-11-19

Out[97]:

	Area Abbreviation	Area Code	Area	Item Code	Item
0	AF	2	Afghanistan	2511	Wheat and products
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)
2	AF	2	Afghanistan	2513	Barley and products
3	AF	2	Afghanistan	2513	Barley and products
4	AF	2	Afghanistan	2514	Maize and products
...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter
21473	ZW	181	Zimbabwe	2960	Fish, Seafood
21474	ZW	181	Zimbabwe	2960	Fish, Seafood
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other
21476	ZW	181	Zimbabwe	2928	Miscellaneous

21477 rows × 5 columns

In [98]:
```
# 반복문으로 컬럼명을 순환하면서 Series에 접근하고,
# 각 컬럼의 고유값의 갯수를 출력
for col in df_rel.columns:
    print(col, df_rel[col].nunique())
```
executed in 47ms, finished 15:04:18 2021-11-19

```
Area Abbreviation 173
Area Code 174
Area 174
Item Code 117
```

```
Item 115
Element Code 2
Element 2
Unit 1
latitude 173
longitude 174
Y1961 1197
Y1962 1215
Y1963 1209
Y1964 1236
Y1965 1259
Y1966 1263
Y1967 1283
Y1968 1300
Y1969 1309
Y1970 1322
Y1971 1351
Y1972 1360
Y1973 1374
Y1974 1388
Y1975 1405
Y1976 1410
Y1977 1411
Y1978 1463
Y1979 1473
Y1980 1477
Y1981 1469
Y1982 1508
Y1983 1528
Y1984 1540
Y1985 1538
Y1986 1563
Y1987 1592
Y1988 1613
Y1989 1622
Y1990 1621
Y1991 1632
Y1992 1747
Y1993 1785
Y1994 1796
Y1995 1796
Y1996 1807
Y1997 1810
Y1998 1844
Y1999 1859
Y2000 1892
Y2001 1881
Y2002 1909
Y2003 1935
Y2004 1944
Y2005 1963
Y2006 1987
Y2007 1994
Y2008 2028
Y2009 2029
Y2010 2046
Y2011 2081
Y2012 2084
Y2013 2107
```

In [99]:

```
# 'Item' Seires의 값이 'Sugar beet'인지 여부 출력
df_rel['Item'] == 'Sugar beet'
```

executed in 14ms, finished 15:04:18 2021-11-19

Out[99]:

```
0        False
1        False
2        False
3        False
4        False
         ...
21472    False
21473    False
21474    False
21475    False
21476    False
Name: Item, Length: 21477, dtype: bool
```

In [100]:

```
# 조건문을 통해 샘플 인덱싱
# 'Item' Seires의 값이 'Sugar beet'인지 여부를 조건으로 사용하여 True인 인덱스 이름의 샘플 출력
df_rel.loc[df_rel['Item'] == 'Sugar beet']
```

executed in 28ms, finished 15:04:18 2021-11-19

Out[100]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
103	AL	3	Albania	2537	Sugar beet	5521	Feed	1000 tonnes	41.15	20.17	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
699	AM	1	Armenia	2537	Sugar beet	5521	Feed	1000 tonnes	40.07	45.04	...	1.0	2.0	1.0	2.0	1.0	3.0	1.0	
832	AU	10	Australia	2537	Sugar beet	5521	Feed	1000 tonnes	-25.27	133.78	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1099	AZ	52	Azerbaijan	2537	Sugar beet	5521	Feed	1000 tonnes	40.14	47.58	...	1.0	1.0	4.0	3.0	4.0	4.0	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
20020	AE	225	United Arab	2537	Sugar	5521	Feed	1000	23.42	53.85		0.0	0.0	0.0	1.0	1.0	0.0	0.0	

											...							
			Emirates		beet			tonnes			...							
20676	UZ	235	Uzbekistan	2537	Sugar beet	5521	Feed	1000 tonnes	41.38	64.59	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20900	VE	236	Venezuela (Bolivarian Republic of)	2537	Sugar beet	5142	Food	1000 tonnes	6.42	-66.59	...	17.0	20.0	23.0	21.0	21.0	21.0	30.0
21135	YE	249	Yemen	2537	Sugar beet	5521	Feed	1000 tonnes	15.55	48.52	...	0.0	0.0	1.0	13.0	0.0	0.0	0.0
21374	ZW	181	Zimbabwe	2537	Sugar beet	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

66 rows × 63 columns

In [101]:
```
# loc를 사용하지 않아도 각 인덱스 이름마다 True/Fasle가 분명하기 때문에 출력 가능
df_rel[df_rel['Item'] == 'Sugar beet']
```

executed in 44ms, finished 15:04:18 2021-11-19

Out[101]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
103	AL	3	Albania	2537	Sugar beet	5521	Feed	1000 tonnes	41.15	20.17	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
699	AM	1	Armenia	2537	Sugar beet	5521	Feed	1000 tonnes	40.07	45.04	...	1.0	2.0	1.0	2.0	1.0	3.0	1.0	
832	AU	10	Australia	2537	Sugar beet	5521	Feed	1000 tonnes	-25.27	133.78	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1099	AZ	52	Azerbaijan	2537	Sugar beet	5521	Feed	1000 tonnes	40.14	47.58	...	1.0	1.0	4.0	3.0	4.0	4.0	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
20020	AE	225	United Arab Emirates	2537	Sugar beet	5521	Feed	1000 tonnes	23.42	53.85	...	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
20676	UZ	235	Uzbekistan	2537	Sugar beet	5521	Feed	1000 tonnes	41.38	64.59	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
20900	VE	236	Venezuela (Bolivarian Republic of)	2537	Sugar beet	5142	Food	1000 tonnes	6.42	-66.59	...	17.0	20.0	23.0	21.0	21.0	21.0	30.0	
21135	YE	249	Yemen	2537	Sugar beet	5521	Feed	1000 tonnes	15.55	48.52	...	0.0	0.0	1.0	13.0	0.0	0.0	0.0	
21374	ZW	181	Zimbabwe	2537	Sugar beet	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

66 rows × 63 columns

In [102]:
```
# 'Item' Seires의 값이 'Sugar beet'인지 여부를 조건으로 사용하여 True인 인덱스 이름의 샘플 출력
# 'Area' 컬럼명만 선택하여 Series 출력
df_rel.loc[df_rel['Item'] == 'Sugar beet', 'Area']
```

executed in 14ms, finished 15:04:18 2021-11-19

Out[102]:
```
10                                 Afghanistan
103                                    Albania
699                                    Armenia
832                                  Australia
1099                                Azerbaijan
                                ...
20020                      United Arab Emirates
20676                                Uzbekistan
20900    Venezuela (Bolivarian Republic of)
21135                                     Yemen
21374                                  Zimbabwe
Name: Area, Length: 66, dtype: object
```

In [103]:
```
# 'Item' Seires의 값이 'Sugar beet'인지 여부를 조건으로 사용하여 True인 인덱스 이름의 샘플 출력
# 'Area' 컬럼명만 괄호 2개로 선택하여 DataFrame 출력
df_rel.loc[df_rel['Item'] == 'Sugar beet', ['Area']]
```

executed in 15ms, finished 15:04:18 2021-11-19

Out[103]:

	Area
10	Afghanistan
103	Albania
699	Armenia
832	Australia
1099	Azerbaijan
...	...
20020	United Arab Emirates
20676	Uzbekistan
20900	Venezuela (Bolivarian Republic of)
21135	Yemen
21374	Zimbabwe

66 rows × 1 columns

In [104]:
```
# 'Item' Seires의 값이 'Sugar beet'인지 여부를 조건으로 사용하여 True인 인덱스 이름의 샘플 출력
# 'Area', 'Item', 'latitude' 컬럼명 출력
df_rel.loc[df_rel['Item'] == 'Sugar beet', ['Area', 'Item', 'latitude']]
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[104]:

	Area	Item	latitude
10	Afghanistan	Sugar beet	33.94
103	Albania	Sugar beet	41.15
699	Armenia	Sugar beet	40.07
832	Australia	Sugar beet	-25.27
1099	Azerbaijan	Sugar beet	40.14
...	...	...	...
20020	United Arab Emirates	Sugar beet	23.42
20676	Uzbekistan	Sugar beet	41.38
20900	Venezuela (Bolivarian Republic of)	Sugar beet	6.42
21135	Yemen	Sugar beet	15.55
21374	Zimbabwe	Sugar beet	-19.02

66 rows × 3 columns

In [105]:
```
# 'Item' Seires의 값이 'Sugar beet'인지 여부를 조건으로 사용하여 True인 인덱스 이름의 샘플 출력
# 'Area'부터 'latitude'컬럼까지 출력
df_rel.loc[df_rel['Item'] == 'Sugar beet', 'Area':'latitude']
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[105]:

	Area	Item Code	Item	Element Code	Element	Unit	latitude
10	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94
103	Albania	2537	Sugar beet	5521	Feed	1000 tonnes	41.15
699	Armenia	2537	Sugar beet	5521	Feed	1000 tonnes	40.07
832	Australia	2537	Sugar beet	5521	Feed	1000 tonnes	-25.27
1099	Azerbaijan	2537	Sugar beet	5521	Feed	1000 tonnes	40.14
...	...	...	...	...	...	...	...
20020	United Arab Emirates	2537	Sugar beet	5521	Feed	1000 tonnes	23.42
20676	Uzbekistan	2537	Sugar beet	5521	Feed	1000 tonnes	41.38
20900	Venezuela (Bolivarian Republic of)	2537	Sugar beet	5142	Food	1000 tonnes	6.42
21135	Yemen	2537	Sugar beet	5521	Feed	1000 tonnes	15.55
21374	Zimbabwe	2537	Sugar beet	5142	Food	1000 tonnes	-19.02

66 rows × 7 columns

In [106]:
```
# 'Item' Seires의 값이 'Sugar beet'인지 여부를 조건으로 사용하여 True인 인덱스 이름의 샘플 출력
# 'Area' 컬럼명만 괄호 2개로 선택하여 DataFrame 출력
# 컬럼의 unique 값과 갯수를 출력
df_rel.loc[df_rel['Item'] == 'Sugar beet', ['Area']].value_counts()
```
executed in 13ms, finished 15:04:18 2021-11-19

Out[106]:
```
Area
Germany              2
Afghanistan          1
Russian Federation   1
Mali                 1
Malta                1
                    ..
Italy                1
Jordan               1
Kazakhstan           1
Kenya                1
Zimbabwe             1
Length: 65, dtype: int64
```

In [107]:
```
# 컬럼의 unique 값과 갯수를 출력하되 빈도수에 따라 내림차순으로 정렬
df_rel.loc[df_rel['Item'] == 'Sugar beet', ['Area']].value_counts(ascending=True)
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[107]:
```
Area
Afghanistan    1
Lithuania      1
Mali           1
Malta          1
Mexico         1
              ..
Jordan         1
Kazakhstan     1
Kuwait         1
Zimbabwe       1
Germany        2
Length: 65, dtype: int64
```

In [108]:
```
# array 형식으로 변환
df_rel.loc[df_rel['Item'] == 'Sugar beet', ['Area']].values
```
executed in 14ms, finished 15:04:18 2021-11-19

Out[108]:
```
array([['Afghanistan'],
       ['Albania'],
```

```
['Armenia'],
['Australia'],
['Azerbaijan'],
['Belarus'],
['Botswana'],
['Brunei Darussalam'],
['Bulgaria'],
['Canada'],
['China, Hong Kong SAR'],
['China, Macao SAR'],
['China, mainland'],
['Colombia'],
["C?e d'Ivoire"],
['Croatia'],
['Cyprus'],
['Czechia'],
['Ecuador'],
['Egypt'],
['El Salvador'],
['Estonia'],
['Germany'],
['Germany'],
['Greece'],
['Honduras'],
['Iceland'],
['Iraq'],
['Italy'],
['Jordan'],
['Kazakhstan'],
['Kenya'],
['Kuwait'],
['Latvia'],
['Lebanon'],
['Lithuania'],
['Mali'],
['Malta'],
['Mexico'],
['Montenegro'],
['Nepal'],
['New Zealand'],
['Niger'],
['Norway'],
['Oman'],
['Philippines'],
['Republic of Korea'],
['Republic of Moldova'],
['Romania'],
['Russian Federation'],
['Saudi Arabia'],
['Serbia'],
['Sierra Leone'],
['Slovenia'],
['South Africa'],
['Sudan'],
['Swaziland'],
['Sweden'],
['The former Yugoslav Republic of Macedonia'],
['Turkmenistan'],
['Ukraine'],
['United Arab Emirates'],
['Uzbekistan'],
['Venezuela (Bolivarian Republic of)'],
['Yemen'],
['Zimbabwe']], dtype=object)
```


- **행과 열의 삭제:** drop 기능 사용

> - 열 또는 여러 열을 삭제하려면 열 이름을 사용하고 "axis"를 1로 지정
> - drop 함수는 열이 제거 된 새 DataFrame을 반환하며 원래 DataFrame 값이 수정되려면 drop 함수 내 **"inplace"** 변수를 True로 설정
> - "axis = 0"을 지정하여"drop"기능을 사용하면 행이 제거
> - drop 함수는 숫자 인덱싱이 아닌 "Label" 기반으로 행을 제거하며, 숫자위치나 Index를 기준으로 행을 삭제하려면 iloc을 사용하여 가능

In [109]:
```
# 'Area' 컬럼 삭제
df_rel.drop("Area", axis=1)
```

executed in 44ms, finished 15:04:18 2021-11-19

Out[109]:

	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	Y1961	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y20
0	AF	2	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	1928.0	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605
1	AF	2	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	...	419.0	445.0	546.0	455.0	490.0	415.0	442
2	AF	2	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76.0	...	58.0	236.0	262.0	263.0	230.0	379.0	315
3	AF	2	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	...	185.0	43.0	44.0	48.0	62.0	55.0	60
4	AF	2	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210.0	...	120.0	208.0	233.0	249.0	247.0	195.0	178
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21472	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	230.0	...	373.0	357.0	359.0	356.0	341.0	385.0	418

21473	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	27.0	...	5.0	4.0	9.0	6.0	9.0	5.0	15
21474	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	6.0	...	18.0	14.0	17.0	14.0	15.0	18.0	29
21475	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0
21476	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0

21477 rows × 62 columns

In [110]:
```
# 'Area' 컬럼 삭제 다른 방법
df_rel.drop(columns='Area')
```
executed in 44ms, finished 15:04:18 2021-11-19

Out[110]:

	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	Y1961	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y20
0	AF	2	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	1928.0	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605
1	AF	2	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	...	419.0	445.0	546.0	455.0	490.0	415.0	442
2	AF	2	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76.0	...	58.0	236.0	262.0	263.0	230.0	379.0	315
3	AF	2	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	...	185.0	43.0	44.0	48.0	62.0	55.0	60
4	AF	2	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210.0	...	120.0	208.0	233.0	249.0	247.0	195.0	178
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21472	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	230.0	...	373.0	357.0	359.0	356.0	341.0	385.0	418
21473	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	27.0	...	5.0	4.0	9.0	6.0	9.0	5.0	15
21474	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	6.0	...	18.0	14.0	17.0	14.0	15.0	18.0	29
21475	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0
21476	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0

21477 rows × 62 columns

In [111]:
```
# 컬럼명 확인
df_rel.columns
```
executed in 14ms, finished 15:04:19 2021-11-19

Out[111]:
```
Index(['Area Abbreviation', 'Area Code', 'Area', 'Item Code', 'Item',
       'Element Code', 'Element', 'Unit', 'latitude', 'longitude', 'Y1961',
       'Y1962', 'Y1963', 'Y1964', 'Y1965', 'Y1966', 'Y1967', 'Y1968', 'Y1969',
       'Y1970', 'Y1971', 'Y1972', 'Y1973', 'Y1974', 'Y1975', 'Y1976', 'Y1977',
       'Y1978', 'Y1979', 'Y1980', 'Y1981', 'Y1982', 'Y1983', 'Y1984', 'Y1985',
       'Y1986', 'Y1987', 'Y1988', 'Y1989', 'Y1990', 'Y1991', 'Y1992', 'Y1993',
       'Y1994', 'Y1995', 'Y1996', 'Y1997', 'Y1998', 'Y1999', 'Y2000', 'Y2001',
       'Y2002', 'Y2003', 'Y2004', 'Y2005', 'Y2006', 'Y2007', 'Y2008', 'Y2009',
       'Y2010', 'Y2011', 'Y2012', 'Y2013'],
      dtype='object')
```

In [112]:
```
# 실제 DataFrame에 삭제되지 않은 이유는 inplace=True라는 파라미터를 사용하지 않았기 때문
# 파이썬에선 무분별한 삭제를 막기 위해 별도 파라미터 입력을 사용해서 확실하게 삭제
df_test = df_rel.copy()
df_test.drop("Area", axis=1, inplace=True)
# is same as
# df_test = df_test.drop('Area', axis=1)
```
executed in 14ms, finished 15:04:19 2021-11-19

In [113]:
```
df_test.columns
```
executed in 13ms, finished 15:04:19 2021-11-19

Out[113]:
```
Index(['Area Abbreviation', 'Area Code', 'Item Code', 'Item', 'Element Code',
       'Element', 'Unit', 'latitude', 'longitude', 'Y1961', 'Y1962', 'Y1963',
       'Y1964', 'Y1965', 'Y1966', 'Y1967', 'Y1968', 'Y1969', 'Y1970', 'Y1971',
       'Y1972', 'Y1973', 'Y1974', 'Y1975', 'Y1976', 'Y1977', 'Y1978', 'Y1979',
       'Y1980', 'Y1981', 'Y1982', 'Y1983', 'Y1984', 'Y1985', 'Y1986', 'Y1987',
       'Y1988', 'Y1989', 'Y1990', 'Y1991', 'Y1992', 'Y1993', 'Y1994', 'Y1995',
       'Y1996', 'Y1997', 'Y1998', 'Y1999', 'Y2000', 'Y2001', 'Y2002', 'Y2003',
       'Y2004', 'Y2005', 'Y2006', 'Y2007', 'Y2008', 'Y2009', 'Y2010', 'Y2011',
       'Y2012', 'Y2013'],
      dtype='object')
```

In [114]:
```
# 여러개의 컬럼명 "Y2011", "Y2012", "Y2013" 삭제
df_rel.drop(["Y2011", "Y2012", "Y2013"], axis=1)
```
executed in 45ms, finished 15:04:19 2021-11-19

Out[114]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2001	Y2002	Y2003	Y2004	Y2005	Y2006
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	2668.0	2776.0	3095.0	3249.0	3486.0	3704.0

Typesetting math: 100%

1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	411.0	448.0	460.0	419.0	445.0	546.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	29.0	70.0	48.0	58.0	236.0	262.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	83.0	122.0	144.0	185.0	43.0	44.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	48.0	89.0	63.0	120.0	208.0	233.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	439.0	360.0	386.0	373.0	357.0	359.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	1.0	0.0	5.0	4.0	9.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	16.0	14.0	18.0	14.0	17.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 60 columns

In [115]:
```
#  0,1,5번째의 인덱스 삭제
df_rel.drop([0,1,5], axis=0)
```
executed in 44ms, finished 15:04:19 2021-11-19

Out[115]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	3
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	1
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0	
7	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	...	2.0	1.0	1.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	4
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	

21474 rows × 63 columns

In [116]:
```
# 0~4번째 인덱스 샘플을 삭제하는 것은 5번째 이상의 인덱스 샘플만 선택하는 것과 같음
df_rel.iloc[5:,]
```
executed in 44ms, finished 15:04:19 2021-11-19

Out[116]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y
5	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	...	231.0	67.0	82.0	67.0	69.0	71.0	
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0	
7	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	...	2.0	1.0	1.0	0.0	0.0	0.0	
8	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	...	276.0	294.0	294.0	260.0	242.0	250.0	1
9	AF	2	Afghanistan	2536	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	...	50.0	29.0	61.0	65.0	54.0	114.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	4
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	

21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21472 rows × 63 columns

1.6 행과 열의 이름바꾸기(Renaming)

- **행 이름 변경**

> - **DataFrame.index = "값"** 형태로 수정 가능("값"의 길이는 index의 길이와 같아야 함)
> - **DataFrame.set_index(column_name)** 함수로 특정 column을 index로 반영 가능
> - **DataFrame.reset_index()** 함수로 초기 index로 변경 가능

- **열 이름 변경**

> - **DataFrame.columns = "값"** 형태로 수정 가능("값"의 길이는 column의 길이와 같아야 함)
> - **DataFrame.rename(columns={'Old':'New'})** 함수로 쉽게 수행 가능
> - **{'old_column_name': 'new_column_name',...}** 형식으로 이전이름과 새이름을 dictionary 형태로 mapping하여 변경

In [117]:
```
df_rel.index
```
executed in 14ms, finished 15:04:19 2021-11-19

Out[117]:
```
RangeIndex(start=0, stop=21477, step=1)
```

In [118]:
```
# 인덱스 값을 임의의 숫자범위로 바꾸기
# 임의의 숫자범위는 기존 인덱스의 갯수와 같아야 함
df_rel.index = range(100,21577)
df_rel
```
executed in 44ms, finished 15:04:19 2021-11-19

Out[118]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
100	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
101	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
102	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
103	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
104	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21572	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21573	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21574	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21575	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21576	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

In [119]:
```
# 인덱스를 특정 컬럼명으로 바꿈
df_rel.set_index("Area")
```
executed in 44ms, finished 15:04:19 2021-11-19

Out[119]:

Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	Y1961	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
Afghanistan	AF	2	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	1928.0	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
Afghanistan	AF	2	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	...	419.0	445.0	546.0	455.0	490.0	415.0
Afghanistan	AF	2	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76.0	...	58.0	236.0	262.0	263.0	230.0	379.0
Afghanistan	AF	2	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	...	185.0	43.0	44.0	48.0	62.0	55.0
Afghanistan	AF	2	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210.0	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	230.0	...	373.0	357.0	359.0	356.0	341.0	385.0
Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	27.0	...	5.0	4.0	9.0	6.0	9.0	5.0
Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	6.0	...	18.0	14.0	17.0	14.0	15.0	18.0

Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	6.0	...	18.0	14.0	17.0	14.0	15.0	18.0
Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 62 columns

In [120]:
```
# 인덱스를 특정 컬럼명으로 바꿈
# 인덱스 이름 중에서 'Afghanistan'으로 샘플 삭제
df_rel.set_index("Area").drop("Afghanistan", axis=0)
```

executed in 58ms, finished 15:04:19 2021-11-19

Out[120]:

Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	Y1961	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2
Albania	AL	3	2511	Wheat and products	5521	Feed	1000 tonnes	41.15	20.17	10.0	...	28.0	28.0	30.0	28.0	28.0	30.0	
Albania	AL	3	2511	Wheat and products	5142	Food	1000 tonnes	41.15	20.17	166.0	...	449.0	468.0	422.0	425.0	435.0	415.0	4
Albania	AL	3	2805	Rice (Milled Equivalent)	5521	Feed	1000 tonnes	41.15	20.17	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
Albania	AL	3	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	41.15	20.17	2.0	...	23.0	24.0	30.0	27.0	20.0	23.0	
Albania	AL	3	2513	Barley and products	5521	Feed	1000 tonnes	41.15	20.17	2.0	...	9.0	4.0	9.0	2.0	3.0	4.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	230.0	...	373.0	357.0	359.0	356.0	341.0	385.0	4
Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	27.0	...	5.0	4.0	9.0	6.0	9.0	5.0	
Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	6.0	...	18.0	14.0	17.0	14.0	15.0	18.0	
Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

21394 rows × 62 columns

In [121]:
```
# inplace=True 파라미터를 사용하지 않아서 실제 인덱스는 변경되지 않음
# 데이터 삭제 뿐만 아니라 인덱스를 바꾸는 것도 무분별한 변환을 막기 위해 파라미터 입력 추가
# inplace=True 파라미터 사용하여 Area 컬럼명을 인덱스로 설정
df_rel.set_index("Area", inplace=True)
df_rel
```

executed in 45ms, finished 15:04:19 2021-11-19

Out[121]:

Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	Y1961	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
Afghanistan	AF	2	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	1928.0	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
Afghanistan	AF	2	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	...	419.0	445.0	546.0	455.0	490.0	415.0
Afghanistan	AF	2	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76.0	...	58.0	236.0	262.0	263.0	230.0	379.0
Afghanistan	AF	2	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	...	185.0	43.0	44.0	48.0	62.0	55.0
Afghanistan	AF	2	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210.0	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	230.0	...	373.0	357.0	359.0	356.0	341.0	385.0
Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	27.0	...	5.0	4.0	9.0	6.0	9.0	5.0
Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	6.0	...	18.0	14.0	17.0	14.0	15.0	18.0
Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 62 columns

In [122]:
```
# 기존 인덱스로 복원하기
# 복원의 경우도 무분별한 복원을 막으려 inplace=True 파라미터를 써야 최종 변환
# 새 인덱스가 첫번째 컬럼명으로 반영
df_rel.reset_index(inplace=True)
df_rel
```

executed in 44ms, finished 15:04:19 2021-11-19

Out[122]:

	Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	Afghanistan	AF	2	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	Afghanistan	AF	2	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	Afghanistan	AF	2	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	Afghanistan	AF	2	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	Afghanistan	AF	2	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21472	Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21477 rows × 63 columns

In [123]:
```
# 여러가지의 기능을 연속적으로 적용 가능
# 파이썬의 경우 .(점) 기호를 붙여 연속적인 연산이 가능
# 1) 'Area' 컬럼을 인덱스로 적용
# 2) 인덱스명이 'Afghanistan' 샘플 삭제
# 3) 기존 인덱스로 복원
df_rel = df_rel.set_index('Area').drop('Afghanistan', axis=0).reset_index()
df_rel
```

executed in 104ms, finished 15:04:19 2021-11-19

Out[123]:

	Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2(
0	Albania	AL	3	2511	Wheat and products	5521	Feed	1000 tonnes	41.15	20.17	...	28.0	28.0	30.0	28.0	28.0	30.0	2
1	Albania	AL	3	2511	Wheat and products	5142	Food	1000 tonnes	41.15	20.17	...	449.0	468.0	422.0	425.0	435.0	415.0	43
2	Albania	AL	3	2805	Rice (Milled Equivalent)	5521	Feed	1000 tonnes	41.15	20.17	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	Albania	AL	3	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	41.15	20.17	...	23.0	24.0	30.0	27.0	20.0	23.0	2
4	Albania	AL	3	2513	Barley and products	5521	Feed	1000 tonnes	41.15	20.17	...	9.0	4.0	9.0	2.0	3.0	4.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21389	Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	41
21390	Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	1
21391	Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	2
21392	Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	
21393	Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	

21394 rows × 63 columns

In [124]:
```
df_test = df_rel.copy()
df_test.columns
```

executed in 29ms, finished 15:04:19 2021-11-19

Out[124]:
```
Index(['Area', 'Area Abbreviation', 'Area Code', 'Item Code', 'Item',
       'Element Code', 'Element', 'Unit', 'latitude', 'longitude', 'Y1961',
       'Y1962', 'Y1963', 'Y1964', 'Y1965', 'Y1966', 'Y1967', 'Y1968', 'Y1969',
       'Y1970', 'Y1971', 'Y1972', 'Y1973', 'Y1974', 'Y1975', 'Y1976', 'Y1977',
       'Y1978', 'Y1979', 'Y1980', 'Y1981', 'Y1982', 'Y1983', 'Y1984', 'Y1985',
       'Y1986', 'Y1987', 'Y1988', 'Y1989', 'Y1990', 'Y1991', 'Y1992', 'Y1993',
       'Y1994', 'Y1995', 'Y1996', 'Y1997', 'Y1998', 'Y1999', 'Y2000', 'Y2001',
       'Y2002', 'Y2003', 'Y2004', 'Y2005', 'Y2006', 'Y2007', 'Y2008', 'Y2009',
       'Y2010', 'Y2011', 'Y2012', 'Y2013'],
      dtype='object')
```

In [125]:
```
# df_test 데이터에서 컬럼명을 Test0 ~ test63까지로 바꾸기
df_test.columns = ['Test'+str(i) for i in range(len(df_rel.columns))]
df_test
```

executed in 44ms, finished 15:04:19 2021-11-19

Out[125]:

	Test0	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	...	Test53	Test54	Test55	Test56	Test57	Test58	Test59	Test60	Te
0	Albania	AL	3	2511	Wheat and products	5521	Feed	1000 tonnes	41.15	20.17	...	28.0	28.0	30.0	28.0	28.0	30.0	26.0	25.0	
1	Albania	AL	3	2511	Wheat and products	5142	Food	1000 tonnes	41.15	20.17	...	449.0	468.0	422.0	425.0	435.0	415.0	432.0	439.0	
2	Albania	AL	3	2805	Rice (Milled Equivalent)	5521	Feed	1000 tonnes	41.15	20.17	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	Albania	AL	3	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	41.15	20.17	...	23.0	24.0	30.0	27.0	20.0	23.0	24.0	21.0	
4	Albania	AL	3	2513	Barley and products	5521	Feed	1000 tonnes	41.15	20.17	...	9.0	4.0	9.0	2.0	3.0	4.0	7.0	8.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21389	Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	418.0	457.0	
21390	Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	15.0	15.0	
21391	Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	29.0	40.0	
21392	Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
21393	Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

21394 rows × 63 columns

In [126]:
```
# 딕셔너리 형태로 반영하여 이전 컬럼명을 신규 컬럼명으로 전환
# Area 컬럼명을 New_Area로 변환
df_rel.rename(columns={'Area':'New_Area'})
```

executed in 59ms, finished 15:04:19 2021-11-19

Out[126]:

	New_Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2
0	Albania	AL	3	2511	Wheat and products	5521	Feed	1000 tonnes	41.15	20.17	...	28.0	28.0	30.0	28.0	28.0	30.0	2
1	Albania	AL	3	2511	Wheat and products	5142	Food	1000 tonnes	41.15	20.17	...	449.0	468.0	422.0	425.0	435.0	415.0	43
2	Albania	AL	3	2805	Rice (Milled Equivalent)	5521	Feed	1000 tonnes	41.15	20.17	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	Albania	AL	3	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	41.15	20.17	...	23.0	24.0	30.0	27.0	20.0	23.0	2
4	Albania	AL	3	2513	Barley and products	5521	Feed	1000 tonnes	41.15	20.17	...	9.0	4.0	9.0	2.0	3.0	4.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21389	Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	41
21390	Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	1
21391	Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	2
21392	Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	
21393	Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	

21394 rows × 63 columns

In [127]:
```
# 딕셔너리 형태로 반영하여 이전 컬럼명을 신규 컬럼명으로 전환
# Area 컬럼명을 New_Area로 변환
# Y2013 컬럼명을 Year_2013로 변환
df_rel.rename(columns={'Area':'New_Area',
                       'Y2013':'Year_2013'})
```

executed in 90ms, finished 15:04:19 2021-11-19

Out[127]:

	New_Area	Area Abbreviation	Area Code	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2
0	Albania	AL	3	2511	Wheat and products	5521	Feed	1000 tonnes	41.15	20.17	...	28.0	28.0	30.0	28.0	28.0	30.0	2
1	Albania	AL	3	2511	Wheat and products	5142	Food	1000 tonnes	41.15	20.17	...	449.0	468.0	422.0	425.0	435.0	415.0	43
2	Albania	AL	3	2805	Rice (Milled Equivalent)	5521	Feed	1000 tonnes	41.15	20.17	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	Albania	AL	3	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	41.15	20.17	...	23.0	24.0	30.0	27.0	20.0	23.0	2
4	Albania	AL	3	2513	Barley and products	5521	Feed	1000 tonnes	41.15	20.17	...	9.0	4.0	9.0	2.0	3.0	4.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21389	Zimbabwe	ZW	181	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	41

21390	Zimbabwe	ZW	181	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21391	Zimbabwe	ZW	181	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21392	Zimbabwe	ZW	181	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21393	Zimbabwe	ZW	181	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

21394 rows × 63 columns

In [128]:
```
# lambda 함수를 사용하여 직접 컬럼명을 지정
# 컬럼명을 모두 대문자로 반영
# 공백은 _기호를 변환
# inplace=True 파라미터로 실제값도 변경
df_rel.rename(columns=lambda x: x.upper().replace(' ', '_'), inplace=True)
```

executed in 11ms, finished 15:04:19 2021-11-19

In [129]:
```
# 확인하기
df_rel.columns
```

executed in 25ms, finished 15:04:19 2021-11-19

Out[129]:
```
Index(['AREA', 'AREA_ABBREVIATION', 'AREA_CODE', 'ITEM_CODE', 'ITEM',
       'ELEMENT_CODE', 'ELEMENT', 'UNIT', 'LATITUDE', 'LONGITUDE', 'Y1961',
       'Y1962', 'Y1963', 'Y1964', 'Y1965', 'Y1966', 'Y1967', 'Y1968', 'Y1969',
       'Y1970', 'Y1971', 'Y1972', 'Y1973', 'Y1974', 'Y1975', 'Y1976', 'Y1977',
       'Y1978', 'Y1979', 'Y1980', 'Y1981', 'Y1982', 'Y1983', 'Y1984', 'Y1985',
       'Y1986', 'Y1987', 'Y1988', 'Y1989', 'Y1990', 'Y1991', 'Y1992', 'Y1993',
       'Y1994', 'Y1995', 'Y1996', 'Y1997', 'Y1998', 'Y1999', 'Y2000', 'Y2001',
       'Y2002', 'Y2003', 'Y2004', 'Y2005', 'Y2006', 'Y2007', 'Y2008', 'Y2009',
       'Y2010', 'Y2011', 'Y2012', 'Y2013'],
      dtype='object')
```

1.7 결과 출력(Exporting and Saving)

- **목적:** 데이터 전처리 또는 정리가 끝난 결과를 "파일"로 저장해 두는 것이 필요

> - to_csv는 DataFrame을 CSV 파일에 기록
> - to_excel은 DataFrame 정보를 Excel 파일에 기록

In [130]:
```
# 데이터프레임을 csv 파일로 저장
# index=False 파라미터는 인덱스 이름은 저장하지 않는다는 의미
df_rel.to_csv("Tutorial_Pandas_Output.csv", index=False)
```

executed in 630ms, finished 15:04:20 2021-11-19

In [131]:
```
# 데이터프레임을 excel 파일로 저장
# excel 파일이 csv 파일과 달리 sheet 이름을 적용하므로 sheet_name 파라미터로 변경가능
# index=False 파라미터는 인덱스 이름은 저장하지 않는다는 의미
df_rel.to_excel("Tutorial_Pandas_Output.xlsx", sheet_name="Sheet 1", index=False)
```

executed in 15.7s, finished 15:04:36 2021-11-19

1.8 데이터프레임들의 병합(Merge and Join DataFrames)

- 실제 현업에서는 각 부서의 목적에 맞게 또는 보안상 여러개의 데이터프레임으로 분산되어 저장 보유
- 데이터가 분산될 경우 데이터의 크기가 작아지고 다루기가 쉬워지며 필요에 따라 다른 데이터를 병합함
- 데이터 프레임 병합 및 결합은 데이터 분석가 지망생이 마스터해야 하는 핵심 프로세스

> - 두 데이터의 병합은 "공통속성/공통열" 기반 하나의 데이터로 가져오고 각각의 행 정렬 프로세스

1.8.1 SQL 방식

- pandasql 패키지를 사용하여 파이썬에서 sql 언어의 로직으로 데이터 처리 가능

In [132]:
```
# pandasql 설치
!pip install pandasql
```

executed in 2.56s, finished 15:04:38 2021-11-19

```
Requirement already satisfied: pandasql in c:₩users₩kk₩anaconda3₩lib₩site-packages (0.7.3)
Requirement already satisfied: numpy in c:₩users₩kk₩anaconda3₩lib₩site-packages (from pandasql) (1.19.5)
Requirement already satisfied: pandas in c:₩users₩kk₩anaconda3₩lib₩site-packages (from pandasql) (1.3.2)
Requirement already satisfied: sqlalchemy in c:₩users₩kk₩anaconda3₩lib₩site-packages (from pandasql) (1.4.7)
Requirement already satisfied: python-dateutil>=2.7.3 in c:₩users₩kk₩anaconda3₩lib₩site-packages (from pandas->pandasql) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:₩users₩kk₩appdata₩roaming₩python₩python38₩site-packages (from pandas->pandasql) (201
9.3)
Requirement already satisfied: six>=1.5 in c:₩users₩kk₩anaconda3₩lib₩site-packages (from python-dateutil>=2.7.3->pandas->pandasql)
(1.15.0)
Requirement already satisfied: greenlet!=0.4.17 in c:₩users₩kk₩anaconda3₩lib₩site-packages (from sqlalchemy->pandasql) (1.0.0)
```

```
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:₩Users₩KK₩anaconda3₩python.exe -m pip install --upgrade pip' command.
```

In [133]:
```
# pandasql 패키지를 psql 이름으로 불러오기
import pandasql as psql
```

executed in 150ms, finished 15:04:38 2021-11-19

In [134]:
```
# 데이터 불러오기
import pandas as pd

user_usage = pd.read_csv('https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/master/user_usage.csv')
user_device = pd.read_csv('https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/master/user_device.csv')
display(user_usage.head(), user_device.head())
```

executed in 2.67s, finished 15:04:41 2021-11-19

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	21.97	4.82	1557.33	22787
1	1710.08	136.88	7267.55	22788
2	1710.08	136.88	7267.55	22789
3	94.46	35.17	519.12	22790
4	71.59	79.26	1557.33	22792

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1

```
select 컬럼명
from 테이블명
where 조건식
group by 컬럼명
having 조건식
order by 컬럼명
```

- **select:** 필수 구성으로 선택할 컬럼을 지칭하며 *기호는 모든 열 의미
- **from:** 필수 구성으로 컬럼을 가지고 올 데이터 지칭
- **where:** 선택 구성으로 특정 조건을 만족하는 데이터 추출
- **group by:** 선택 구성으로 컬럼들의 우선순위를 부여하여 그룹핑
- **having:** 선택 구성으로 group by 연산시 조건을 만족하는 경우 부여
- **order by:** 선택 구성으로 오름차순/내림차순 부여

In [135]:
```
# 모든 열 선택 출력
psql.sqldf("select * from user_device")
```

executed in 313ms, finished 15:04:41 2021-11-19

Out[135]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1
...	...	...	...	...	...	...
267	23049	29725	android	6.0	SM-G900F	1
268	23050	29726	ios	10.2	iPhone7,2	3
269	23051	29726	ios	10.2	iPhone7,2	3
270	23052	29727	ios	10.1	iPhone8,4	3
271	23053	20257	android	5.1	Vodafone Smart ultra 6	1

272 rows × 6 columns

In [136]:
```
# 특정 열만 선택 출력
psql.sqldf("select user_id, use_id, platform, device, use_type_id from user_device")
```

executed in 39ms, finished 15:04:41 2021-11-19

Out[136]:

	user_id	use_id	platform	device	use_type_id
0	26980	22782	ios	iPhone7,2	2
1	29628	22783	android	Nexus 5	3
2	28473	22784	android	SM-G903F	1
3	15200	22785	ios	iPhone7,2	3
4	28239	22786	android	ONE E1003	1
...	...	...	...	...	...
267	29725	23049	android	SM-G900F	1
268	29726	23050	ios	iPhone7,2	3
269	29726	23051	ios	iPhone7,2	3
270	29727	23052	ios	iPhone8,4	3
271	20257	23053	android	Vodafone Smart ultra 6	1

272 rows × 5 columns

In [137]:
```
# 특정 조건을 만족하는 열 선택 출력
psql.sqldf("select * from user_device ₩
            where platform_version >= 10")
```
executed in 43ms, finished 15:04:41 2021-11-19

Out[137]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22785	15200	ios	10.2	iPhone7,2	3
2	22791	28775	ios	10.2	iPhone6,2	3
3	22796	29641	ios	10.2	iPhone7,2	3
4	22797	29642	ios	10.1	iPhone5,2	3
...	...	...	...	...	...	...
59	23045	29720	ios	10.2	iPhone7,1	2
60	23047	29720	ios	10.2	iPhone7,1	2
61	23050	29726	ios	10.2	iPhone7,2	3
62	23051	29726	ios	10.2	iPhone7,2	3
63	23052	29727	ios	10.1	iPhone8,4	3

64 rows × 6 columns

In [138]:
```
# 특정 조건을 만족하는 열 선택 출력
psql.sqldf("select * from user_device ₩
            where platform_version >= 10 and use_type_id = 3")
```
executed in 44ms, finished 15:04:41 2021-11-19

Out[138]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22785	15200	ios	10.2	iPhone7,2	3
1	22791	28775	ios	10.2	iPhone6,2	3
2	22796	29641	ios	10.2	iPhone7,2	3
3	22797	29642	ios	10.1	iPhone5,2	3
4	22798	29642	ios	10.1	iPhone5,2	3
...	...	...	...	...	...	...
32	23035	29718	ios	10.2	iPhone7,2	3
33	23037	29720	ios	10.2	iPhone7,1	3
34	23050	29726	ios	10.2	iPhone7,2	3
35	23051	29726	ios	10.2	iPhone7,2	3
36	23052	29727	ios	10.1	iPhone8,4	3

37 rows × 6 columns

In [139]:
```
# 특정 조건을 만족하는 열의 연산 출력
psql.sqldf("select user_id, count(*) from user_device ₩
            where platform_version >= 10 and use_type_id = 3")
```
executed in 29ms, finished 15:04:41 2021-11-19

Out[139]:

	user_id	count(*)
0	15200	37

In [140]:
```
# 특정 조건을 만족하는 열의 연산 출력
psql.sqldf("select user_id, avg(platform_version) from user_device ₩
            where platform_version >= 10 and use_type_id = 3")
```
executed in 44ms, finished 15:04:42 2021-11-19

Out[140]:

	user_id	avg(platform_version)
0	15200	10.151351

In [141]:
```
# 특정 조건을 만족하는 열의 연산하고 그룹핑하여 퍼진정도 확인
psql.sqldf("select user_id, avg(platform_version) from user_device ₩
            where platform_version >= 10 and use_type_id = 3 ₩
            group by user_id")
```
executed in 43ms, finished 15:04:42 2021-11-19

Out[141]:

	user_id	avg(platform_version)
0	8981	10.0
1	15200	10.2
2	19912	10.2
3	21186	10.1
4	28431	10.2
...	...	...
20	29715	10.1
21	29718	10.2
22	29720	10.2

23	29726	10.2
24	29727	10.1

25 rows × 2 columns

In [142]:
```
# 특정 조건을 만족하는 열의 연산하고 그룹핑하여 퍼진정도 확인
psql.sqldf("select user_id, avg(platform_version), count(use_id), device from user_device ₩
            where platform_version >= 10 and use_type_id = 3 ₩
            group by user_id")
```

executed in 28ms, finished 15:04:42 2021-11-19

Out[142]:

	user_id	avg(platform_version)	count(use_id)	device
0	8981	10.0	1	iPhone7,2
1	15200	10.2	1	iPhone7,2
2	19912	10.2	1	iPhone8,1
3	21186	10.1	1	iPhone5,2
4	28431	10.2	1	iPhone7,2
...	...	...	...	...
20	29715	10.1	1	iPhone6,2
21	29718	10.2	1	iPhone7,2
22	29720	10.2	1	iPhone7,1
23	29726	10.2	2	iPhone7,2
24	29727	10.1	1	iPhone8,4

25 rows × 4 columns

In [143]:
```
# 특정 조건을 만족하는 열의 연산하고 특정 조건을 만족하는 그룹핑 파악
psql.sqldf("select user_id, avg(platform_version), count(use_id), device from user_device ₩
            where platform_version >= 10 and use_type_id = 3 ₩
            group by user_id ₩
            having avg(platform_version) >= 10.2")
```

executed in 44ms, finished 15:04:42 2021-11-19

Out[143]:

	user_id	avg(platform_version)	count(use_id)	device
0	15200	10.2	1	iPhone7,2
1	19912	10.2	1	iPhone8,1
2	28431	10.2	1	iPhone7,2
3	28775	10.2	1	iPhone6,2
4	29445	10.2	2	iPhone6,2
...	...	...	...	...
10	29676	10.2	1	iPhone5,4
11	29699	10.2	1	iPhone6,2
12	29718	10.2	1	iPhone7,2
13	29720	10.2	1	iPhone7,1
14	29726	10.2	2	iPhone7,2

15 rows × 4 columns

In [144]:
```
# 특정 조건을 만족하는 열의 연산하고 특정 조건을 만족하는 그룹핑을 정렬하여 파악
psql.sqldf("select user_id, avg(platform_version), count(use_id), device from user_device ₩
            where platform_version >= 10 and use_type_id = 3 ₩
            group by user_id ₩
            having avg(platform_version) >= 10.2 ₩
            order by count(use_id) desc")
```

executed in 29ms, finished 15:04:42 2021-11-19

Out[144]:

	user_id	avg(platform_version)	count(use_id)	device
0	29641	10.2	4	iPhone7,2
1	29726	10.2	2	iPhone7,2
2	29657	10.2	2	iPhone8,4
3	29445	10.2	2	iPhone6,2
4	29720	10.2	1	iPhone7,1
...	...	...	...	...
10	29648	10.2	1	iPhone7,2
11	28775	10.2	1	iPhone6,2
12	28431	10.2	1	iPhone7,2
13	19912	10.2	1	iPhone8,1
14	15200	10.2	1	iPhone7,2

15 rows × 4 columns

In [145]:
```
# 특정 조건을 만족하는 열의 연산하고 특정 조건을 만족하는 그룹핑을 정렬하여 파악
psql.sqldf("select user_id, avg(platform_version), count(use_id), device from user_device ₩
            where platform_version >= 10 and use_type_id = 3 ₩
            group by user_id ₩
            having avg(platform_version) >= 10.2 ₩
```

```
            having avg(platform_version) >= 10.2 ₩
            order by count(use_id) asc")
```

executed in 44ms, finished 15:04:42 2021-11-19

Out[145]:

	user_id	avg(platform_version)	count(use_id)	device
0	15200	10.2	1	iPhone7,2
1	19912	10.2	1	iPhone8,1
2	28431	10.2	1	iPhone7,2
3	28775	10.2	1	iPhone6,2
4	29648	10.2	1	iPhone7,2
...	...	...	...	...
10	29720	10.2	1	iPhone7,1
11	29445	10.2	2	iPhone6,2
12	29657	10.2	2	iPhone8,4
13	29726	10.2	2	iPhone7,2
14	29641	10.2	4	iPhone7,2

15 rows × 4 columns

In [146]:

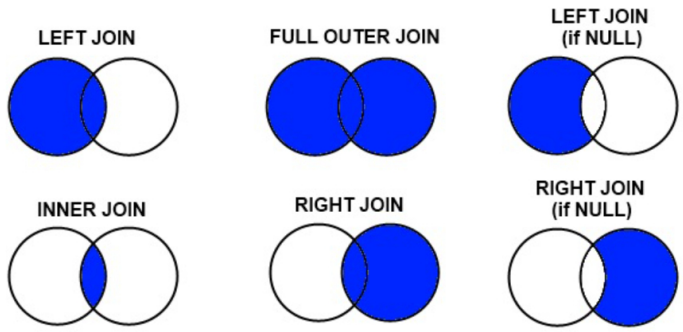
```
# 변수로 저장하여 출력 가능
user_device_filter = psql.sqldf("select user_id, avg(platform_version), count(use_id), device from user_device ₩
                                 where platform_version >= 10 and use_type_id = 3 ₩
                                 group by user_id ₩
                                 having avg(platform_version) >= 10.2 ₩
                                 order by count(use_id) asc")
user_device_filter
```

executed in 29ms, finished 15:04:42 2021-11-19

Out[146]:

	user_id	avg(platform_version)	count(use_id)	device
0	15200	10.2	1	iPhone7,2
1	19912	10.2	1	iPhone8,1
2	28431	10.2	1	iPhone7,2
3	28775	10.2	1	iPhone6,2
4	29648	10.2	1	iPhone7,2
...	...	...	...	...
10	29720	10.2	1	iPhone7,1
11	29445	10.2	2	iPhone6,2
12	29657	10.2	2	iPhone8,4
13	29726	10.2	2	iPhone7,2
14	29641	10.2	4	iPhone7,2

15 rows × 4 columns




```
select (테이블)A.컬럼명, (테이블)B.컬럼명
from 기준테이블 A
[something] join 조인테이블 B
on (테이블)A.기준키 = (테이블)B.기준키
```

- **조인(join):** 두개 이상의 테이블이나 데이터베이스를 연결하여 데이터 검색 방법

> - **[inner] join:** 교집합으로 A테이블과 B테이블이 모두 가진 데이터를 결합하여 출력
> - **[left outer] join:** A테이블의 모든 데이터와 B테이블의 중복 데이터만 결합하여 출력
> - **[right outer] join:** B테이블의 모든 데이터와 A테이블의 중복 데이터만 결합하여 출력
> - **[full outer] join:** 합집합으로 A테이블과 B테이블을 모두 결합하여 출력
> - **[left outer] join (if null):** A테이블의 모든 데이터 중와 B테이블의 중복 데이터만 결합하는데 B테이블의 key를 가진 행은 삭제하여 출력
> - **[right outer] join (if null):** B테이블의 모든 데이터 중와 A테이블의 중복 데이터만 결합하는데 A테이블의 key를 가진 행은 삭제하여 출력
> - **참고:** 결합과정에서 값이 없는 경우는 NaN으로 채워짐

In [147]:

```
# 로딩된 데이터 정리
display(user_device, user_device_filter, user_usage)
```

executed in 29ms, finished 15:04:42 2021-11-19

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1
...	...	...	...	...	...	...
267	23049	29725	android	6.0	SM-G900F	1
268	23050	29726	ios	10.2	iPhone7,2	3
269	23051	29726	ios	10.2	iPhone7,2	3
270	23052	29727	ios	10.1	iPhone8,4	3
271	23053	20257	android	5.1	Vodafone Smart ultra 6	1

272 rows × 6 columns

	user_id	avg(platform_version)	count(use_id)	device
0	15200	10.2	1	iPhone7,2
1	19912	10.2	1	iPhone8,1
2	28431	10.2	1	iPhone7,2
3	28775	10.2	1	iPhone6,2
4	29648	10.2	1	iPhone7,2
...	...	...	...	...
10	29720	10.2	1	iPhone7,1
11	29445	10.2	2	iPhone6,2
12	29657	10.2	2	iPhone8,4
13	29726	10.2	2	iPhone7,2
14	29641	10.2	4	iPhone7,2

15 rows × 4 columns

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	21.97	4.82	1557.33	22787
1	1710.08	136.88	7267.55	22788
2	1710.08	136.88	7267.55	22789
3	94.46	35.17	519.12	22790
4	71.59	79.26	1557.33	22792
...	...	...	...	...
235	260.66	68.44	896.96	25008
236	97.12	36.50	2815.00	25040
237	355.93	12.37	6828.09	25046
238	632.06	120.46	1453.16	25058
239	488.70	906.92	3089.85	25220

240 rows × 4 columns

In [148]:
```
# user_device, use_device_filter 결합하여 정리
psql.sqldf("select A.* from user_device A ₩
            inner join user_device_filter B ₩
            on A.user_id = B.user_id")
```

executed in 44ms, finished 15:04:42 2021-11-19

Out[148]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22785	15200	ios	10.2	iPhone7,2	3
1	22791	28775	ios	10.2	iPhone6,2	3
2	22796	29641	ios	10.2	iPhone7,2	3
3	22809	29648	ios	10.2	iPhone7,2	3
4	22810	29649	ios	10.2	iPhone8,1	3
...	...	...	...	...	...	...
19	23038	29720	ios	10.2	iPhone7,1	2
20	23045	29720	ios	10.2	iPhone7,1	2
21	23047	29720	ios	10.2	iPhone7,1	2
22	23050	29726	ios	10.2	iPhone7,2	3
23	23051	29726	ios	10.2	iPhone7,2	3

24 rows × 6 columns

In [149]:
```
# 새로운 이름으로 저장
user_device_new = psql.sqldf("select A.* from user_device A ₩
            inner join user_device_filter B ₩
```

```
            on A.user_id = B.user_id")
user_device_new
```

executed in 43ms, finished 15:04:42 2021-11-19

Out[149]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22785	15200	ios	10.2	iPhone7,2	3
1	22791	28775	ios	10.2	iPhone6,2	3
2	22796	29641	ios	10.2	iPhone7,2	3
3	22809	29648	ios	10.2	iPhone7,2	3
4	22810	29649	ios	10.2	iPhone8,1	3
...	...	...	...	...	...	...
19	23038	29720	ios	10.2	iPhone7,1	2
20	23045	29720	ios	10.2	iPhone7,1	2
21	23047	29720	ios	10.2	iPhone7,1	2
22	23050	29726	ios	10.2	iPhone7,2	3
23	23051	29726	ios	10.2	iPhone7,2	3

24 rows × 6 columns

In [150]:

```
# user_device_new와 user_usage 데이터 inner join 결합
# 상위 플랫폼을 쓰는 고객의 사용 기록 없음
psql.sqldf("select A.*, B.* from user_device_new A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id")
```

executed in 42ms, finished 15:04:42 2021-11-19

Out[150]:

use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id

In [151]:

```
# user_device와 user_usage 데이터 inner join 결합
psql.sqldf("select A.*, B.* from user_device A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id")
```

executed in 58ms, finished 15:04:42 2021-11-19

Out[151]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22787	12921	android	4.3	GT-I9505	1	21.97	4.82	1557.33	22787
1	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
2	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22789
3	22790	29592	android	5.1	D2303	1	94.46	35.17	519.12	22790
4	22792	28217	android	5.1	SM-G361F	1	71.59	79.26	1557.33	22792
...	...	...	...	...	...	...	...	...	...	...
154	23043	28953	android	6.0	SM-G900F	1	198.59	90.49	5191.12	23043
155	23044	28953	android	6.0	SM-G900F	1	198.59	90.49	3114.67	23044
156	23046	29454	android	6.0	Moto G (4)	1	106.65	82.13	5191.12	23046
157	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12	23049
158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12	23053

159 rows × 10 columns

In [152]:

```
# user_device와 user_usage 데이터 inner join 결합 및 연산 등 지원
psql.sqldf("select A.*, B.* from user_device A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id ₩
            where outgoing_mins_per_month >= 500")
```

executed in 58ms, finished 15:04:42 2021-11-19

Out[152]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
1	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22789
2	22804	29645	android	6.0	SM-G935F	1	554.41	150.06	3114.67	22804
3	22813	23415	android	4.4	HTC Desire 510	1	797.06	7.67	519.12	22813
4	22814	23415	android	4.4	HTC Desire 510	1	797.06	7.67	15573.33	22814
...	...	...	...	...	...	...	...	...	...	...
6	22816	23415	android	4.4	HTC Desire 510	1	797.06	7.67	15573.33	22816
7	22817	23415	android	4.4	HTC Desire 510	1	797.06	7.67	15573.33	22817
8	22829	29653	ios	10.1	iPhone7,2	2	681.44	47.35	1271.39	22829
9	22858	29544	android	7.0	ONEPLUS A3003	1	1221.85	69.20	6229.33	22858
10	23026	22763	android	6.0	ONE A2003	1	532.98	44.36	2076.45	23026

11 rows × 10 columns

In [153]:

```
# user_device와 user_usage 데이터 inner join 결합 및 연산 등 지원
psql.sqldf("select A.*, B.* from user_device A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id ₩
```

```
            where outgoing_mins_per_month >= 500 ₩
            group by user_id")
```

executed in 44ms, finished 15:04:42 2021-11-19

Out[153]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	23026	22763	android	6.0	ONE A2003	1	532.98	44.36	2076.45	23026
1	22813	23415	android	4.4	HTC Desire 510	1	797.06	7.67	519.12	22813
2	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
3	22858	29544	android	7.0	ONEPLUS A3003	1	1221.85	69.20	6229.33	22858
4	22804	29645	android	6.0	SM-G935F	1	554.41	150.06	3114.67	22804
5	22829	29653	ios	10.1	iPhone7,2	2	681.44	47.35	1271.39	22829

In [154]:

```
# user_device와 user_usage 데이터 inner join 결합 및 연산 등 지원
psql.sqldf("select A.*, B.* from user_device A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id ₩
            where outgoing_mins_per_month >= 500 ₩
            group by user_id ₩
            having platform_version >= 6.0 ₩
            order by platform_version desc")
```

executed in 43ms, finished 15:04:42 2021-11-19

Out[154]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22829	29653	ios	10.1	iPhone7,2	2	681.44	47.35	1271.39	22829
1	22858	29544	android	7.0	ONEPLUS A3003	1	1221.85	69.20	6229.33	22858
2	22804	29645	android	6.0	SM-G935F	1	554.41	150.06	3114.67	22804
3	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
4	23026	22763	android	6.0	ONE A2003	1	532.98	44.36	2076.45	23026

In [155]:

```
psql.sqldf("select A.*, B.* from user_device A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id")
```

executed in 61ms, finished 15:04:42 2021-11-19

Out[155]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22787	12921	android	4.3	GT-I9505	1	21.97	4.82	1557.33	22787
1	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
2	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22789
3	22790	29592	android	5.1	D2303	1	94.46	35.17	519.12	22790
4	22792	28217	android	5.1	SM-G361F	1	71.59	79.26	1557.33	22792
...	...	...	...	...	...	...	...	...	...	...
154	23043	28953	android	6.0	SM-G900F	1	198.59	90.49	5191.12	23043
155	23044	28953	android	6.0	SM-G900F	1	198.59	90.49	3114.67	23044
156	23046	29454	android	6.0	Moto G (4)	1	106.65	82.13	5191.12	23046
157	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12	23049
158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12	23053

159 rows × 10 columns

In [156]:

```
# left join 결과 확인
psql.sqldf("select A.*, B.* from user_device A ₩
            left join user_usage B ₩
            on A.use_id = B.use_id")
```

executed in 42ms, finished 15:04:42 2021-11-19

Out[156]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22782	26980	ios	10.2	iPhone7,2	2	NaN	NaN	NaN	NaN
1	22783	29628	android	6.0	Nexus 5	3	NaN	NaN	NaN	NaN
2	22784	28473	android	5.1	SM-G903F	1	NaN	NaN	NaN	NaN
3	22785	15200	ios	10.2	iPhone7,2	3	NaN	NaN	NaN	NaN
4	22786	28239	android	6.0	ONE E1003	1	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
267	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12	23049.0
268	23050	29726	ios	10.2	iPhone7,2	3	NaN	NaN	NaN	NaN
269	23051	29726	ios	10.2	iPhone7,2	3	NaN	NaN	NaN	NaN
270	23052	29727	ios	10.1	iPhone8,4	3	NaN	NaN	NaN	NaN
271	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12	23053.0

272 rows × 10 columns

In [157]:

```
# right join 결과 확인
# right join의 오류 가능성으로 left join의 기능 유도
# psql.sqldf("select A.*, B.* from user_device A ₩
#             right join user_usage B ₩
```

```
#              on A.use_id = B.use_id")
```

executed in 13ms, finished 15:04:42 2021-11-19

In [158]:
```
# right join 결과 확인
psql.sqldf("select A.*, B.* from user_usage B ₩
            left join user_device A ₩
            on A.use_id = B.use_id")
```

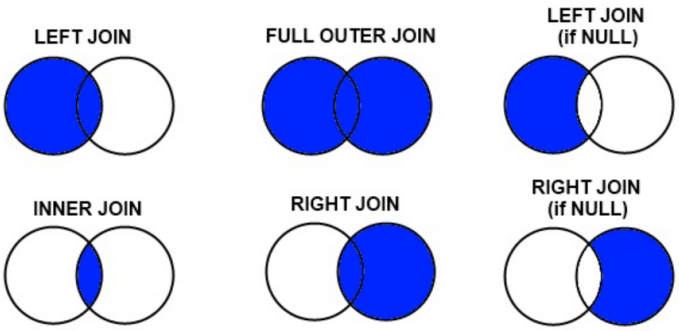
executed in 44ms, finished 15:04:42 2021-11-19

Out[158]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22787.0	12921.0	android	4.3	GT-I9505	1.0	21.97	4.82	1557.33	22787
1	22788.0	28714.0	android	6.0	SM-G930F	1.0	1710.08	136.88	7267.55	22788
2	22789.0	28714.0	android	6.0	SM-G930F	1.0	1710.08	136.88	7267.55	22789
3	22790.0	29592.0	android	5.1	D2303	1.0	94.46	35.17	519.12	22790
4	22792.0	28217.0	android	5.1	SM-G361F	1.0	71.59	79.26	1557.33	22792
...	...	...	...	...	...	...	...	...	...	...
235	NaN	NaN	None	NaN	None	NaN	260.66	68.44	896.96	25008
236	NaN	NaN	None	NaN	None	NaN	97.12	36.50	2815.00	25040
237	NaN	NaN	None	NaN	None	NaN	355.93	12.37	6828.09	25046
238	NaN	NaN	None	NaN	None	NaN	632.06	120.46	1453.16	25058
239	NaN	NaN	None	NaN	None	NaN	488.70	906.92	3089.85	25220

240 rows × 10 columns

1.8.2 PANDAS 방식




```
pd.merge(left=기준 데이터프레임,
         right=병합할 데이터프레임,
         how=병합할 방식,    # {'left', 'right', 'outer', 'inner', 'cross'}, default 'inner'
         [on=key값],    # 두 데이터프레임의 key 값이 같을 경우
         [left_on=left_key값]    # 기존 데이터프레임 key값 설정
         [right_on=right_key값])    # 병합할 데이터프레임 key값 설정
```

In [159]:
```
# 예시 데이터 생성
df1 = pd.DataFrame({
    '이름': ['원영', '사쿠라', '유리', '예나', '유진', '나코', '은비', '혜원', '히토미', '채원', '민주', '째욘'],
    '국어': [100, 70, 70, 70, 60, 90, 90, 70, 70, 80, 100, 100],
    '영어': [100, 90, 80, 50, 70, 100, 70, 90, 100, 100, 80, 100]
    }, columns=['이름', '국어', '영어'])

df2 = pd.DataFrame({
    '일어': [80, 100, 100, 90, 70, 50, 100],
    '수학': [90, 70, 100, 80, 70, 80, 90],
    'name': ['원영', '사쿠라', '나코', '히토미', '예나', '은비', '째욘'],
    }, columns=['일어', '수학', 'name'])

display(df1, df2)
```

executed in 29ms, finished 15:04:42 2021-11-19

	이름	국어	영어
0	원영	100	100
1	사쿠라	70	90
2	유리	70	80
3	예나	70	50
4	유진	60	70
...	...	...	...
7	혜원	70	90
8	히토미	70	100
9	채원	80	100
10	민주	100	80
11	째욘	100	100

12 rows × 3 columns

	일어	수학	name
0	80	90	원영
1	100	70	사쿠라
2	100	100	나코
3	90	80	히토미
4	70	70	예나
5	50	80	은비
6	100	90	째욘

In [160]:
```
# key 값이 다른 경우 outer 결합 예시
# 서로 값이 없는 경우는 NaN으로 채워짐
pd.merge(df1, df2, left_on='이름', right_on='name', how='outer')
```
executed in 12ms, finished 15:04:42 2021-11-19

Out[160]:

	이름	국어	영어	일어	수학	name
0	원영	100	100	80.0	90.0	원영
1	사쿠라	70	90	100.0	70.0	사쿠라
2	유리	70	80	NaN	NaN	NaN
3	예나	70	50	70.0	70.0	예나
4	유진	60	70	NaN	NaN	NaN
...	...	...	...	...	...	...
7	혜원	70	90	NaN	NaN	NaN
8	히토미	70	100	90.0	80.0	히토미
9	채원	80	100	NaN	NaN	NaN
10	민주	100	80	NaN	NaN	NaN
11	째욘	100	100	100.0	90.0	째욘

12 rows × 6 columns

In [161]:
```
# 데이터 불러오기
import pandas as pd

user_usage = pd.read_csv('https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/master/user_usage.csv')
user_device = pd.read_csv('https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/master/user_device.csv')
display(user_usage.head(), user_device.head())
```
executed in 166ms, finished 15:04:43 2021-11-19

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	21.97	4.82	1557.33	22787
1	1710.08	136.88	7267.55	22788
2	1710.08	136.88	7267.55	22789
3	94.46	35.17	519.12	22790
4	71.59	79.26	1557.33	22792

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1

In [162]:
```
psql.sqldf("select A.*, B.* from user_device A ₩
            inner join user_usage B ₩
            on A.use_id = B.use_id")
```
executed in 74ms, finished 15:04:43 2021-11-19

Out[162]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22787	12921	android	4.3	GT-I9505	1	21.97	4.82	1557.33	22787
1	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
2	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22789
3	22790	29592	android	5.1	D2303	1	94.46	35.17	519.12	22790
4	22792	28217	android	5.1	SM-G361F	1	71.59	79.26	1557.33	22792
...	...	...	...	...	...	...	...	...	...	...
154	23043	28953	android	6.0	SM-G900F	1	198.59	90.49	5191.12	23043
155	23044	28953	android	6.0	SM-G900F	1	198.59	90.49	3114.67	23044
156	23046	29454	android	6.0	Moto G (4)	1	106.65	82.13	5191.12	23046
157	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12	23049

158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12	23053

159 rows × 10 columns

In [163]:
```
# pandas 사용 inner join 결합
pd.merge(user_device, user_usage, how='inner', on='use_id')
```
executed in 28ms, finished 15:04:43 2021-11-19

Out[163]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb
0	22787	12921	android	4.3	GT-I9505	1	21.97	4.82	1557.33
1	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55
2	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55
3	22790	29592	android	5.1	D2303	1	94.46	35.17	519.12
4	22792	28217	android	5.1	SM-G361F	1	71.59	79.26	1557.33
...	...	...	...	...	...	...	...	...	...
154	23043	28953	android	6.0	SM-G900F	1	198.59	90.49	5191.12
155	23044	28953	android	6.0	SM-G900F	1	198.59	90.49	3114.67
156	23046	29454	android	6.0	Moto G (4)	1	106.65	82.13	5191.12
157	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12
158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12

159 rows × 9 columns

In [164]:
```
psql.sqldf("select A.*, B.* from user_device A ₩
            left join user_usage B ₩
            on A.use_id = B.use_id")
```
executed in 74ms, finished 15:04:43 2021-11-19

Out[164]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22782	26980	ios	10.2	iPhone7,2	2	NaN	NaN	NaN	NaN
1	22783	29628	android	6.0	Nexus 5	3	NaN	NaN	NaN	NaN
2	22784	28473	android	5.1	SM-G903F	1	NaN	NaN	NaN	NaN
3	22785	15200	ios	10.2	iPhone7,2	3	NaN	NaN	NaN	NaN
4	22786	28239	android	6.0	ONE E1003	1	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
267	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12	23049.0
268	23050	29726	ios	10.2	iPhone7,2	3	NaN	NaN	NaN	NaN
269	23051	29726	ios	10.2	iPhone7,2	3	NaN	NaN	NaN	NaN
270	23052	29727	ios	10.1	iPhone8,4	3	NaN	NaN	NaN	NaN
271	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12	23053.0

272 rows × 10 columns

In [165]:
```
# pandas 사용 left join 결합
pd.merge(user_device, user_usage, how='left', on='use_id')
```
executed in 29ms, finished 15:04:43 2021-11-19

Out[165]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb
0	22782	26980	ios	10.2	iPhone7,2	2	NaN	NaN	NaN
1	22783	29628	android	6.0	Nexus 5	3	NaN	NaN	NaN
2	22784	28473	android	5.1	SM-G903F	1	NaN	NaN	NaN
3	22785	15200	ios	10.2	iPhone7,2	3	NaN	NaN	NaN
4	22786	28239	android	6.0	ONE E1003	1	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
267	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12
268	23050	29726	ios	10.2	iPhone7,2	3	NaN	NaN	NaN
269	23051	29726	ios	10.2	iPhone7,2	3	NaN	NaN	NaN
270	23052	29727	ios	10.1	iPhone8,4	3	NaN	NaN	NaN
271	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12

272 rows × 9 columns

In [166]:
```
psql.sqldf("select A.*, B.* from user_usage B ₩
            left join user_device A ₩
            on A.use_id = B.use_id")
```
executed in 59ms, finished 15:04:43 2021-11-19

Out[166]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22787.0	12921.0	android	4.3	GT-I9505	1.0	21.97	4.82	1557.33	22787
1	22788.0	28714.0	android	6.0	SM-G930F	1.0	1710.08	136.88	7267.55	22788
2	22789.0	28714.0	android	6.0	SM-G930F	1.0	1710.08	136.88	7267.55	22789
3	22790.0	29592.0	android	5.1	D2303	1.0	94.46	35.17	519.12	22790

4	22792.0	28217.0	android	5.1	SM-G361F	1.0	71.59	79.26	1557.33	22792
...	...	...	...	...	...	...	...	...	...	...
235	NaN	NaN	None	NaN	None	NaN	260.66	68.44	896.96	25008
236	NaN	NaN	None	NaN	None	NaN	97.12	36.50	2815.00	25040
237	NaN	NaN	None	NaN	None	NaN	355.93	12.37	6828.09	25046
238	NaN	NaN	None	NaN	None	NaN	632.06	120.46	1453.16	25058
239	NaN	NaN	None	NaN	None	NaN	488.70	906.92	3089.85	25220

240 rows × 10 columns

In [167]:
```
# pandas 사용 right join 결합
pd.merge(user_device, user_usage, how='right', on='use_id')
```
executed in 29ms, finished 15:04:43 2021-11-19

Out[167]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb
0	22787	12921.0	android	4.3	GT-I9505	1.0	21.97	4.82	1557.33
1	22788	28714.0	android	6.0	SM-G930F	1.0	1710.08	136.88	7267.55
2	22789	28714.0	android	6.0	SM-G930F	1.0	1710.08	136.88	7267.55
3	22790	29592.0	android	5.1	D2303	1.0	94.46	35.17	519.12
4	22792	28217.0	android	5.1	SM-G361F	1.0	71.59	79.26	1557.33
...	...	...	...	...	...	...	...	...	...
235	25008	NaN	NaN	NaN	NaN	NaN	260.66	68.44	896.96
236	25040	NaN	NaN	NaN	NaN	NaN	97.12	36.50	2815.00
237	25046	NaN	NaN	NaN	NaN	NaN	355.93	12.37	6828.09
238	25058	NaN	NaN	NaN	NaN	NaN	632.06	120.46	1453.16
239	25220	NaN	NaN	NaN	NaN	NaN	488.70	906.92	3089.85

240 rows × 9 columns

In [168]:
```
# 변수로 저장하여 파이썬 함수 사용가능
result = pd.merge(user_device, user_usage, how='inner', on='use_id')
result.shape
```
executed in 11ms, finished 15:04:43 2021-11-19

Out[168]: (159, 9)

In [169]:
```
# pandas 기능 사용 가능
result.describe(include='all')
```
executed in 42ms, finished 15:04:43 2021-11-19

Out[169]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb
count	159.000000	159.000000	159	159.000000	159	159.000000	159.000000	159.000000	159.000000
unique	NaN	NaN	2	NaN	55	NaN	NaN	NaN	NaN
top	NaN	NaN	android	NaN	SM-G900F	NaN	NaN	NaN	NaN
freq	NaN	NaN	157	NaN	30	NaN	NaN	NaN	NaN
mean	22922.327044	25960.918239	NaN	5.554717	NaN	1.012579	203.331509	87.978742	4180.378616
...	...	...	...	...	...	...	...	...	...
min	22787.000000	2873.000000	NaN	4.100000	NaN	1.000000	0.500000	0.250000	0.000000
25%	22861.500000	24683.500000	NaN	5.000000	NaN	1.000000	70.070000	22.855000	1557.330000
50%	22931.000000	29366.000000	NaN	6.000000	NaN	1.000000	137.060000	62.850000	2076.450000
75%	22986.500000	29673.000000	NaN	6.000000	NaN	1.000000	241.035000	119.675000	5191.120000
max	23053.000000	29725.000000	NaN	10.100000	NaN	2.000000	1710.080000	540.600000	31146.670000

11 rows × 9 columns

1.9 Questions

아마존 구매내역 데이터로 주어진 Task를 작성/출력 하며, 모든 답안의 길이는 한줄부터 여러줄이 가능함

- **데이터 정보**
  - **파일명:**
    1. ecommerce_userinfo.csv
    2. ecommerce_userpurchase.csv
  - **파일위치:**
    1. r'.\Data\Ecommerce\ecommerce_userinfo.csv'
    2. r'.\Data\Ecommerce\ecommerce_userpurchase.csv'

In [170]:
```
# 데이터 로딩
# 1) 파일위치의 2개의 csv 파일을 불러들임
```

```
# 2) 각 DataFrame의 이름은 1. user_info, 2. user_purchase로 변수명 지정
```
executed in 14ms, finished 15:04:43 2021-11-19

In [171]:
```
# user_info와 user_purchase의 첫 10개의 샘플 출력
```
executed in 14ms, finished 15:04:43 2021-11-19

In [172]:
```
# user_info와 user_purchase의 행과 열의 크기 출력
```
executed in 14ms, finished 15:04:43 2021-11-19

In [173]:
```
# user_info와 user_purchase의 정보를 출력하여 데이터의 타입(Dtype)을 확인
```
executed in 14ms, finished 15:04:43 2021-11-19

In [174]:
```
# user_info와 user_purchase의 컬럼명 출력
```
executed in 13ms, finished 15:04:43 2021-11-19

In [175]:
```
# user_info 데이터에서 1000, 3000, 5000, 7000번째 인덱스 행 삭제 후 저장
```
executed in 11ms, finished 15:04:43 2021-11-19

In [176]:
```
# user_purchase 데이터에서 Company, Browser Info 열 삭제 후 저장
```
executed in 13ms, finished 15:04:43 2021-11-19

In [177]:
```
# user_purchase 데이터에서 컬럼명 변경 후 저장
# Language를 대문자 LANGUAGE로 변경
```
executed in 14ms, finished 15:04:43 2021-11-19

In [178]:
```
# user_info와 user_purchase의 기초통계 출력
# 문자값이 있다면 문자값도 통계정보를 출력
```
executed in 13ms, finished 15:04:43 2021-11-19

In [179]:
```
# user_info와 user_purchase의 각 컬럼별 고유값(nunique)의 갯수를 출력
# 컬럼명과 고유값을 함께 출력 ex) IP Address 100
```
executed in 14ms, finished 15:04:43 2021-11-19

In [180]:
```
# user_info와 user_purchase에서,
# (반복문과 조건문 사용하여) 고유값(nunique) 갯수와 데이터프레임의 행의 수가 일치하는 컬럼명 출력
```
executed in 14ms, finished 15:04:43 2021-11-19

In [181]:
```
# 위에서 찾은 컬럼명들 중 데이터 병합시 key값으로 사용할 컬럼명을 아래 주석에 작성
# user_info :
# user_purchase :
```
executed in 12ms, finished 15:04:43 2021-11-19

In [182]:
```
# user_info와 user_purchase를 inner join으로 병합
# 1) 병합 방법은 pandas를 사용할 것
# 2) 병합된 데이터는 ecommerce 변수명으로 저장할 것
```
executed in 13ms, finished 15:04:43 2021-11-19

In [183]:
```
# 병합 데이터의 기초통계 출력
# 문자값이 있다면 문자값도 통계정보를 출력
# 최종 출력은 대칭으로 회전하여 출력
```
executed in 14ms, finished 15:04:43 2021-11-19

In [184]:
```
# ecommerce 데이터에서 가장 많은 거래에 사용된 이메일 주소는 무엇인가?
```
executed in 13ms, finished 15:04:43 2021-11-19

In [185]:
```
# ecommerce 데이터에서 구매가격의 최소값, 평균, 중앙값, 최대값은 얼마인가?
```
executed in 15ms, finished 15:04:43 2021-11-19

In [186]:
```
# 영어(en)를 사용하여 거래를 한 IP 주소의 갯수는 얼마인가?
```
executed in 12ms, finished 15:04:43 2021-11-19

In [187]:
```
# 직업이 "Lawyer"인 사람이 거래를 한 IP 주소의 갯수는 얼마인가?
```
executed in 13ms, finished 15:04:43 2021-11-19

In [188]:
```
# 오전과 오후의 거래량은 어떻게 되는가?
```
executed in 14ms, finished 15:04:43 2021-11-19

In [189]:
```
# 거래량 top 5의 직업은 무엇인가?
```
executed in 14ms, finished 15:04:43 2021-11-19

In [190]:

```
# 카드번호가 4917375496558677로 결제한 고객이 취소를 요청한다 고객의 이메일은 무엇인가?
```

executed in 13ms, finished 15:04:43 2021-11-19

In [191]:

```
# 카드종류가(CC Provider) American Express를 사용하여 99달러 이상 주문한 거래내역을 출력하라
```

executed in 14ms, finished 15:04:43 2021-11-19

In [ ]: