Manupulating data using pandas

Introduction to Pandas

In this section of the course we will learn how to use pandas for data analysis. Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language You can think of pandas as an extremely powerful version of Excel, with a lot more features. In this section of the course, you should go through the notebooks in this order:

- Introduction to Pandas
- Series
- DataFrames
- Missing Data
- GroupBy
- Merging,Joining,and Concatenating
- Operations
- Data Input and Output

First you must have pandas library which can be installed using this command

In [91]:

```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\anaconda3\lib\site-packages (0.2
4.2)
Requirement already satisfied: pytz>=2011k in c:\anaconda3\lib\site-packages
(from pandas) (2019.1)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\anaconda3\lib\si
te-packages (from pandas) (2.8.0)
Requirement already satisfied: numpy>=1.12.0 in c:\anaconda3\lib\site-packag
es (from pandas) (1.16.4)
Requirement already satisfied: six>=1.5 in c:\anaconda3\lib\site-packages (f
rom python-dateutil>=2.5.0->pandas) (1.12.0)
Could not fetch URL https://pypi.org/simple/pip/: (https://pypi.org/simple/p
ip/:) There was a problem confirming the ssl certificate: HTTPSConnectionPoo
l(host='pypi.org', port=443): Max retries exceeded with url: /simple/pip/ (C
aused by SSLError(SSLCertVerificationError(1, '[SSL: CERTIFICATE_VERIFY_FAIL
ED] certificate verify failed: unable to get local issuer certificate (_ssl.
c:1056)'))) - skipping
```

Import Pandas library using this command

In [92]:

```
import pandas as pd
```

Series

The first main data type we will learn about for pandas is the Series data type.

A Series is very similar to a NumPy array (it is built on top of the NumPy array object).

What differentiates the NumPy array from a Series?

1) is that a Series can have axis labels, meaning it can be indexed by a label, instead of just a number location.

2) It also doesn't need to hold numeric data, it can hold any arbitrary Python Object.

Let's explore this concept through some examples:

In [93]:

```
import numpy as np
import pandas as pd
```

You can convert a list,numpy array, or dictionary to a Series:

In [94]:

```
labels = ['a','b','c']
my_list = [10,20,30]
arr = np.array([10,20,30])
d = {'a':10,'b':20,'c':30}
```

Using Lists

In [95]:

```
pd.Series(data=my_list)
```

Out[95]:

```
0    10
1    20
2    30
dtype: int64
```

In [96]:

```
pd.Series(data=my_list,index=labels)
```

Out[96]:

```
a    10
b    20
c    30
dtype: int64
```

NumPy Arrays

In [97]:

```
pd.Series(arr)
```

Out[97]:

```
0    10
1    20
2    30
dtype: int32
```

In [98]:

```
pd.Series(arr,labels)
```

Out[98]:

```
a    10
b    20
c    30
dtype: int32
```

Dictionary

In [99]:

```
pd.Series(d)
```

Out[99]:

```
a    10
b    20
c    30
dtype: int64
```

Using an Index

The key to using a Series is understanding its index. Pandas makes use of these index names or numbers by allowing for fast look ups of information

In [100]:

```
ser1 = pd.Series([1,2,3,4],index = ['USA', 'Germany','USSR', 'Japan'])
```

In [101]:

```
ser2 = pd.Series([6,7,8,9],index = ['USA', 'Germany','Italy', 'Japan'])
```

In [102]:

```
ser1
```

Out[102]:

```
USA        1
Germany    2
USSR       3
Japan      4
dtype: int64
```

In [103]:

```
ser2
```

Out[103]:

```
USA        6
Germany    7
Italy      8
Japan      9
dtype: int64
```

In [104]:

```
ser1[0]
```

Out[104]:

```
1
```

In [105]:

```
ser2[3]
```

Out[105]:

```
9
```

Operations are then also done based off of index:

In [106]:

```
ser1 + ser2
```

Out[106]:

```
Germany     9.0
Italy       NaN
Japan      13.0
USA         7.0
USSR        NaN
dtype: float64
```

DataFrames

One basic structure that you get with pandas is a data frame. A data frame is a two dimensional grid, rather similar to a relational database table except in memory.

DataFrames are the workhorse of pandas and are directly inspired by the R programming language.

We can think of a DataFrame as a bunch of Series objects put together to share the same index

In [107]:

```
from numpy.random import randn
np.random.seed(101)
```

In [108]:

```
df = pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split())
```

In [109]:

```
df
```

Out[109]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

Selection and Indexing

How to grab data from a DataFrame

In [110]:

```
df['W']
```

Out[110]:

```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

In [111]:

```
# Pass a list of column names
df[['W','Z']]
```

Out[111]:

	W	Z
A	2.706850	0.503826
B	0.651118	0.605965
C	-2.018168	-0.589001
D	0.188695	0.955057
E	0.190794	0.683509

DataFrame Columns are just Series

In [112]:

```
type(df['W'])
```

Out[112]:

```
pandas.core.series.Series
```

Creating a new column:

In [113]:

```
df['new'] = df['W'] + df['Y']
```

In [114]:

```
df
```

Out[114]:

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

Removing Columns

In [115]:

```
df.drop('new',axis=1)

# axis = 1 is referring to column
```

Out[115]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

but the 'new' column not permenantly deleted from memory

In [116]:

```
df
```

Out[116]:

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

Not inplace unless specified!

In [117]:

```
df.drop('new',axis=1,inplace=True)
```

In [118]:

```
df
```

Out[118]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

Removing Rows

In [119]:

```
df.drop('E',axis=0)
```

Out[119]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057

Selecting Rows

In [120]:

```
df.loc['A']
```

Out[120]:

```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

Or you can select based off of position instead of label

In [121]:

```
df.iloc[2]
```

Out[121]:

```
W   -2.018168
X    0.740122
Y    0.528813
Z   -0.589001
Name: C, dtype: float64
```

*_Selecting subset of rows and columns_ *

In [122]:

```
df.loc['B','Y']
```

Out[122]:

```
-0.8480769834036315
```

In [123]:

```
df.loc[['A','B'],['W','Y']]
```

Out[123]:

	W	Y
A	2.706850	0.907969
B	0.651118	-0.848077

Example 1

In [124]:

```
##{key : list[]:value[]}

data = {'state': ['Jakarta', 'Jakarta', 'Jakarta', 'Selangor', 'Selangor','Kelantan','Kelar
       'year': [2000, 2001, 2002, 2001, 2002, 2001, 2002],
       'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 1.2, 3.2]}
frame = pd.DataFrame(data)
frame
```

Out[124]:

	state	year	pop
0	Jakarta	2000	1.5
1	Jakarta	2001	1.7
2	Jakarta	2002	3.6
3	Selangor	2001	2.4
4	Selangor	2002	2.9
5	Kelantan	2001	1.2
6	Kelantan	2002	3.2

In [125]:

```
frame = pd.DataFrame(data, columns=['year', 'state', 'pop'])
frame
```

Out[125]:

	year	state	pop
0	2000	Jakarta	1.5
1	2001	Jakarta	1.7
2	2002	Jakarta	3.6
3	2001	Selangor	2.4
4	2002	Selangor	2.9
5	2001	Kelantan	1.2
6	2002	Kelantan	3.2

Adding new column

In [126]:

```
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                     index = ['one', 'two', 'three', 'four', 'five', 'six','seven'])
frame2
```

Out[126]:

	year	state	pop	debt
one	2000	Jakarta	1.5	NaN
two	2001	Jakarta	1.7	NaN
three	2002	Jakarta	3.6	NaN
four	2001	Selangor	2.4	NaN
five	2002	Selangor	2.9	NaN
six	2001	Kelantan	1.2	NaN
seven	2002	Kelantan	3.2	NaN

Selecting column

In [127]:

```
frame2.columns
```

Out[127]:

```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

In [128]:

```
frame2['state']
```

Out[128]:

```
one        Jakarta
two        Jakarta
three      Jakarta
four      Selangor
five      Selangor
six       Kelantan
seven     Kelantan
Name: state, dtype: object
```

In [129]:

```
frame2.year
```

Out[129]:

```
one      2000
two      2001
three    2002
four     2001
five     2002
six      2001
seven    2002
Name: year, dtype: int64
```

In [130]:

```
frame2.loc['two']
```

Out[130]:

```
year        2001
state    Jakarta
pop          1.7
debt         NaN
Name: two, dtype: object
```

loc is location will list all elements under loc [two]. loc will call base on assignee name

In [131]:

```
frame2.loc['two','state']
```

Out[131]:

```
'Jakarta'
```

In [132]:

```
## iloc  base on index

frame2.iloc[1,1]
```

Out[132]:

```
'Jakarta'
```

In [133]:

```
frame2.loc['two':,:'state']
```

Out[133]:

	year	state
two	2001	Jakarta
three	2002	Jakarta
four	2001	Selangor
five	2002	Selangor
six	2001	Kelantan
seven	2002	Kelantan

In [134]:

```
frame2
```

Out[134]:

	year	state	pop	debt
one	2000	Jakarta	1.5	NaN
two	2001	Jakarta	1.7	NaN
three	2002	Jakarta	3.6	NaN
four	2001	Selangor	2.4	NaN
five	2002	Selangor	2.9	NaN
six	2001	Kelantan	1.2	NaN
seven	2002	Kelantan	3.2	NaN

The debt value is NaN. We can assign value for 'debt'

In [135]:

```
frame2['debt'] = 16.5
frame2
```

Out[135]:

	year	state	pop	debt
one	2000	Jakarta	1.5	16.5
two	2001	Jakarta	1.7	16.5
three	2002	Jakarta	3.6	16.5
four	2001	Selangor	2.4	16.5
five	2002	Selangor	2.9	16.5
six	2001	Kelantan	1.2	16.5
seven	2002	Kelantan	3.2	16.5

In [136]:

```
frame2['debt'] = np.arange(7.)
frame2
```

Out[136]:

	year	state	pop	debt
one	2000	Jakarta	1.5	0.0
two	2001	Jakarta	1.7	1.0
three	2002	Jakarta	3.6	2.0
four	2001	Selangor	2.4	3.0
five	2002	Selangor	2.9	4.0
six	2001	Kelantan	1.2	5.0
seven	2002	Kelantan	3.2	6.0

In [137]:

```
frame2['debt'] = [10,20,15,13,11,67,87]
frame2
```

Out[137]:

	year	state	pop	debt
one	2000	Jakarta	1.5	10
two	2001	Jakarta	1.7	20
three	2002	Jakarta	3.6	15
four	2001	Selangor	2.4	13
five	2002	Selangor	2.9	11
six	2001	Kelantan	1.2	67
seven	2002	Kelantan	3.2	87

In [138]:

```
'Jakarta' in frame2.columns
```

Out[138]:

```
False
```

Use Case Exercise

In [139]:

```
import numpy as np
import pandas as pd
```

DATA 1

In [140]:

```
df = pd.read_csv('data2.csv')
```

Checking Top 10 and bottom 10 data

In [141]:

```
df.head()
```

Out[141]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	China	1398.72	9596.96	12234.78	Asia	NaN
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947
2	US	329.74	9833.52	19485.39	N.America	1776-07-04
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07

In [142]:

```
df.tail()
```

Out[142]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
15	Argentina	44.94	2780.40	637.49	S.America	1816-07-09
16	Algeria	43.38	2381.74	167.56	Africa	5/7/1962
17	Canada	37.59	9984.67	1647.12	N.America	1867-07-01
18	Australia	25.47	7692.02	1408.68	Oceania	NaN
19	Kazakhstan	18.53	2724.90	159.41	Asia	16/12/1991

In [143]:

```
df.dtypes
```

Out[143]:

```
COUNTRY        object
POPULATION    float64
AREA          float64
GDP           float64
CONTINENTS     object
IND_DAY        object
dtype: object
```

In [144]:

```
df.shape
```

Out[144]:

```
(20, 6)
```

In [145]:

```
pd.isnull(df)
```

Out[145]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	False	False	False	False	False	True
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	True	False
9	False	False	False	False	False	False

In [146]:

```
newdf1=df[['COUNTRY','POPULATION','CONTINENTS','IND_DAY']]
newdf1
```

Out[146]:

	COUNTRY	POPULATION	CONTINENTS	IND_DAY
0	China	1398.72	Asia	NaN
1	India	1351.16	Asia	15/8/1947
2	US	329.74	N.America	1776-07-04
3	Indonesia	268.07	Asia	17/8/1945
4	Brazil	210.32	S.America	1822-09-07
5	Pakistan	205.71	Asia	14/8/1947
6	Nigeria	200.96	Africa	1/10/1960
7	Bangladesh	167.09	Asia	26/3/1971
8	Russia	146.79	NaN	12/6/1992
9	Mexico	126.58	N.America	1810-09-16

In [147]:

```
newdf1['CONTINENTS'].fillna('transcontinental')
```

Out[147]:

```
0                 Asia
1                 Asia
2            N.America
3                 Asia
4            S.America
5                 Asia
6               Africa
7                 Asia
8     transcontinental
9            N.America
10                Asia
11              Europe
12              Europe
13              Europe
14              Europe
15           S.America
16              Africa
17           N.America
```

In [148]:

```
newdf1

#no changes in Rusia
```

Out[148]:

	COUNTRY	POPULATION	CONTINENTS	IND_DAY
0	China	1398.72	Asia	NaN
1	India	1351.16	Asia	15/8/1947
2	US	329.74	N.America	1776-07-04
3	Indonesia	268.07	Asia	17/8/1945
4	Brazil	210.32	S.America	1822-09-07
5	Pakistan	205.71	Asia	14/8/1947
6	Nigeria	200.96	Africa	1/10/1960
7	Bangladesh	167.09	Asia	26/3/1971
8	Russia	146.79	NaN	12/6/1992
9	Mexico	126.58	N.America	1810-09-16

In [149]:

```
newdf1['CONTINENTS'].fillna('Transcontinental', inplace=True)
newdf1
```

```
C:\Anaconda3\lib\site-packages\pandas\core\generic.py:6130: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  self._update_inplace(new_data)
```

Out[149]:

	COUNTRY	POPULATION	CONTINENTS	IND_DAY
0	China	1398.72	Asia	NaN
1	India	1351.16	Asia	15/8/1947
2	US	329.74	N.America	1776-07-04
3	Indonesia	268.07	Asia	17/8/1945
4	Brazil	210.03	S.America	1822-09-07

In [150]:

```
newdf1.dtypes
```

Out[150]:

```
COUNTRY        object
POPULATION    float64
CONTINENTS     object
IND_DAY        object
dtype: object
```

How to change date format

In [151]:

```
newdf1['IND_DAY']=pd.to_datetime(newdf1['IND_DAY'])
newdf1
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
```

Out[151]:

	COUNTRY	POPULATION	CONTINENTS	IND_DAY
0	China	1398.72	Asia	NaT
1	India	1351.16	Asia	1947-08-15
2	US	329.74	N.America	1776-07-04
3	Indonesia	268.07	Asia	1945-08-17

In [152]:

```
newdf1.dtypes
```

Out[152]:

```
COUNTRY                object
POPULATION            float64
CONTINENTS             object
IND_DAY        datetime64[ns]
dtype: object
```

How to fill in missing date

In [153]:

```
newdf1['IND_DAY'].fillna(pd.Timestamp("20210423"))
```

Out[153]:

```
0    2021-04-23
1    1947-08-15
2    1776-07-04
3    1945-08-17
4    1822-09-07
5    1947-08-14
6    1960-01-10
7    1971-03-26
8    1992-12-06
9    1810-09-16
10   2021-04-23
11   2021-04-23
12   1789-07-14
13   2021-04-23
14   2021-04-23
15   1816-07-09
16   1962-05-07
17   1867-07-01
```

In [154]:

```
newdf1['IND_DAY'].astype(str).replace({'NaT': "No date"})
```

Out[154]:

```
0        No date
1     1947-08-15
2     1776-07-04
3     1945-08-17
4     1822-09-07
5     1947-08-14
6     1960-01-10
7     1971-03-26
8     1992-12-06
9     1810-09-16
10       No date
11       No date
12    1789-07-14
13       No date
14       No date
15    1816-07-09
16    1962-05-07
17    1867-07-01
```

In [155]:

```
newdf1['IND_DAY'].fillna(value = 'No date')
```

Out[155]:

```
0                 No date
1     1947-08-15 00:00:00
2     1776-07-04 00:00:00
3     1945-08-17 00:00:00
4     1822-09-07 00:00:00
5     1947-08-14 00:00:00
6     1960-01-10 00:00:00
7     1971-03-26 00:00:00
8     1992-12-06 00:00:00
9     1810-09-16 00:00:00
10                No date
11                No date
12    1789-07-14 00:00:00
13                No date
14                No date
15    1816-07-09 00:00:00
16    1962-05-07 00:00:00
17    1867-07-01 00:00:00
```

In [156]:

```
newdf1['IND_DAY'].fillna(value = 'No date', inplace = True)
newdf1
```

Out[156]:

	COUNTRY	POPULATION	CONTINENTS	IND_DAY
0	China	1398.72	Asia	No date
1	India	1351.16	Asia	1947-08-15 00:00:00
2	US	329.74	N.America	1776-07-04 00:00:00
3	Indonesia	268.07	Asia	1945-08-17 00:00:00
4	Brazil	210.32	S.America	1822-09-07 00:00:00
5	Pakistan	205.71	Asia	1947-08-14 00:00:00
6	Nigeria	200.96	Africa	1960-01-10 00:00:00
7	Bangladesh	167.09	Asia	1971-03-26 00:00:00
8	Russia	146.79	Transcontinental	1992-12-06 00:00:00
9	Mexico	126.58	N.America	1810-09-16 00:00:00

References:

[1] https://stackoverflow.com/questions/42818262/pandas-dataframe-replace-nat-with-none (https://stackoverflow.com/questions/42818262/pandas-dataframe-replace-nat-with-none)
[2] https://pandas.pydata.org/pandas-docs/stable/user_guide/timedeltas.html (https://pandas.pydata.org/pandas-docs/stable/user_guide/timedeltas.html)
[3] https://stackoverflow.com/questions/32327314/how-to-rearrange-a-date-in-python (https://stackoverflow.com/questions/32327314/how-to-rearrange-a-date-in-python)

Add new data set call 'data3'

In [157]:

```
df1 = pd.read_csv('data3.csv')
df1
```

```
---------------------------------------------------------------------------
-
UnicodeDecodeError                        Traceback (most recent call las
t)
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_token
s()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_with_
dtype()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._string_conver
t()

pandas/_libs/parsers.pyx in pandas._libs.parsers._string_box_utf8()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xf4 in position 1: in
valid continuation byte

During handling of the above exception, another exception occurred:
```

In [158]:

```
df1 = pd.read_csv('data3.csv',encoding='latin-1')
df1
```

Out[158]:

	country	cases	deaths	region
0	United States	32,669,121	584,226	North America
1	India	16,257,309	186,928	Asia
2	Brazil	14,172,139	383,757	South America
3	France	5,408,606	102,164	Europe
4	Russia	4,736,121	107,103	Europe
5	Turkey	4,501,382	37,329	Asia
6	United Kingdom	4,398,431	127,345	Europe
7	Italy	3,920,945	118,357	Europe
8	Spain	3,456,886	77,496	Europe
9	Germany	3,238,054	81,693	Europe

```
In [159]:
```

```
df1.dtypes
```

```
Out[159]:

country    object
cases      object
deaths     object
region     object
dtype: object
```

```
In [160]:
```

```
df1.shape
```

```
Out[160]:

(220, 4)
```

Change Data Type

change cases and death to float

```
In [161]:
```

```
df1['deaths'] = df1['deaths'].str.replace(',','')
df1['deaths'] = df1.deaths.astype(float)
df1['cases'] = df1['cases'].str.replace(',','')
df1['cases'] = df1.cases.astype(float)
```

```
In [162]:
```

```
df1.dtypes
```

```
Out[162]:

country     object
cases      float64
deaths     float64
region      object
dtype: object
```

```
In [163]:
```

```
df.head(2)
```

```
Out[163]:
```

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	China	1398.72	9596.96	12234.78	Asia	NaN
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947

In [164]:

```
df1.head(2)
```

Out[164]:

	country	cases	deaths	region
0	United States	32669121.0	584226.0	North America
1	India	16257309.0	186928.0	Asia

Setting header to data case

Different way writing the header. Let's change it.

In [165]:

```
# .capitalize to change first letter as capital letter.
df1.columns=df1.columns.str.capitalize()
df1.head(2)
```

Out[165]:

	Country	Cases	Deaths	Region
0	United States	32669121.0	584226.0	North America
1	India	16257309.0	186928.0	Asia

In [166]:

```
df1.columns=df1.columns.str.upper()

#.upper() to change header to uppercase

df1.head(2)
```

Out[166]:

	COUNTRY	CASES	DEATHS	REGION
0	United States	32669121.0	584226.0	North America
1	India	16257309.0	186928.0	Asia

In [167]:

```
df1.shape
```

Out[167]:

```
(220, 4)
```

In [168]:

```
df.shape
```

Out[168]:

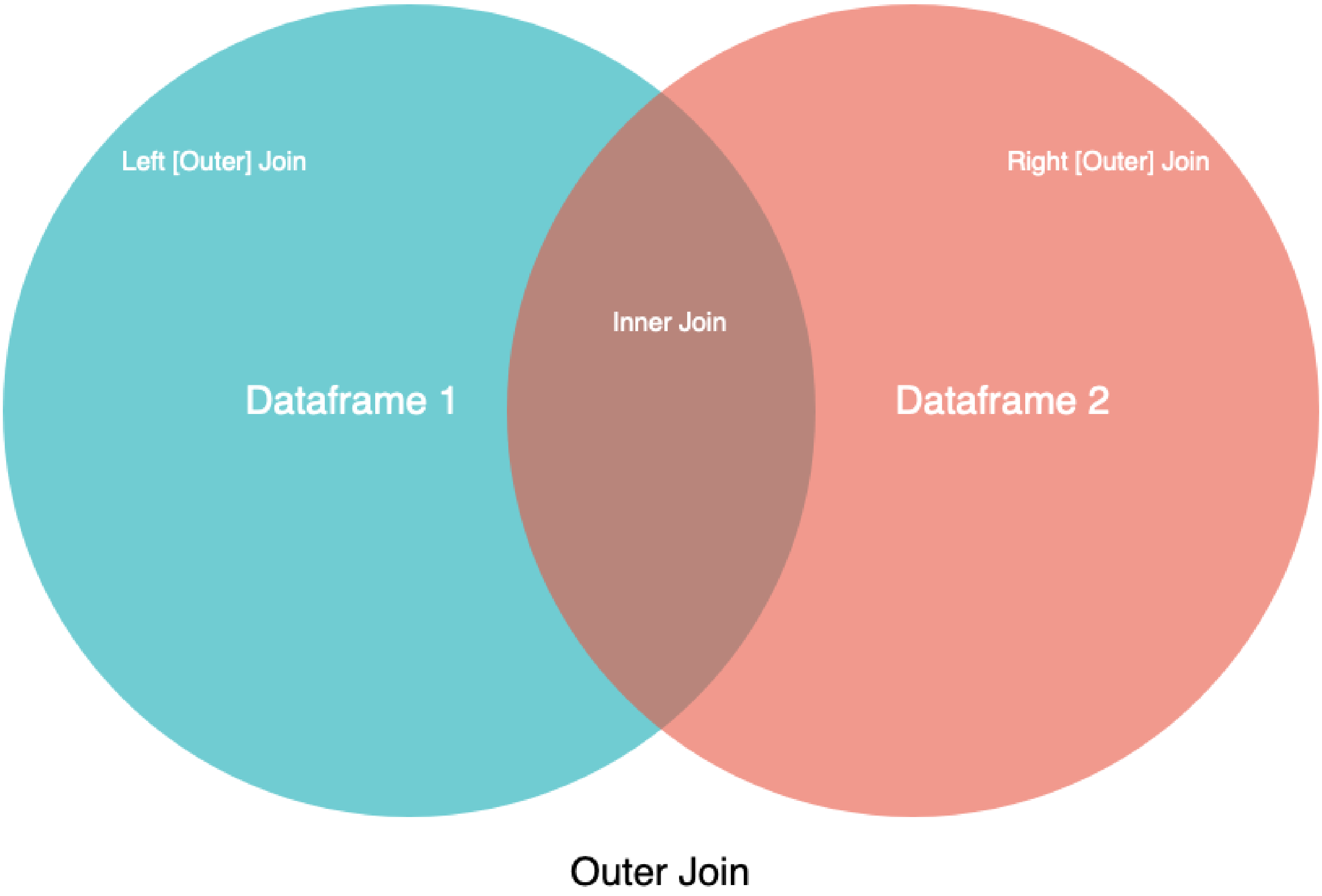
```
(20, 6)
```

Merge

Pandas provides a single function, merge(), as the entry point for all standard database join operations between DataFrame or named Series objects.

MERGE combining data on common columns or indices.

You can achieve both many-to-one and many-to-many joins with merge()



When gluing together multiple DataFrames, you have a choice of how to handle the other axes (other than the one being concatenated). This can be done in the following two ways

Take the union of them all, join='outer'. This is the default option as it results in zero information loss.

Take the intersection, join='inner'.

In [169]:

```
merge_df=pd.merge(df, df1)
merge_df
```

Out[169]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS	R
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0	
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0	
2	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0	
3	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	14172139.0	383757.0	South A
4	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0	
5	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0	
6	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0	
7	Russia	146.79	17098.25	1530.75	NaN	12/6/1992	4736121.0	107103.0	
8	Mexico	126.58	1964.38	1158.23	N.America	1810-09-	2319519.0	214095.0	North A

By default, how = inner, which will merge only match data.

In [170]:

```
merge_df.shape
```

Out[170]:

```
(18, 9)
```

In [171]:

```
merge_df.CONTINENTS=merge_df.CONTINENTS.replace(['N.America','S.America'],['North America',
merge_df
```

Out[171]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEAT
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	463
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	18692
2	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	4417
3	Brazil	210.32	8515.77	2055.51	South America	1822-09-07	14172139.0	38375
4	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	1684
5	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	206
6	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	1078
7	Russia	146.79	17098.25	1530.75	NaN	12/6/1992	4736121.0	10710
8	Mexico	126.58	1964.38	1158.23	North America	1810-09-16	2319519.0	21409
9	Japan	126.22	377.97	4872.42	Asia	NaN	547137.0	977
10	Germany	83.02	357.11	3693.20	Europe	NaN	3238054.0	8169
11	France	67.02	640.68	2582.49	Europe	1789-07-14	5408606.0	10216
12	Italy	60.36	301.34	1943.84	Europe	NaN	3920945.0	11835
13	Argentina	44.94	2780.40	637.49	South America	1816-07-09	2796768.0	6062
14	Algeria	43.38	2381.74	167.56	Africa	5/7/1962	120363.0	318
15	Canada	37.59	9984.67	1647.12	North America	1867-07-01	1155834.0	2382
16	Australia	25.47	7692.02	1408.68	Oceania	NaN	29626.0	91
17	Kazakhstan	18.53	2724.90	159.41	Asia	16/12/1991	300733.0	351

In [172]:

```
test1merge_df=pd.merge(df, df1, how='inner')
test1merge_df
```

Out[172]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS	R
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0	
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0	
2	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0	
3	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	14172139.0	383757.0	South A
4	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0	
5	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0	
6	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0	
7	Russia	146.79	17098.25	1530.75	NaN	12/6/1992	4736121.0	107103.0	
8	Mexico	126.58	1964.38	1158.23	N.America	1810-09-	2319519.0	214095.0	North A

In [173]:

```
test1merge_df.shape
```

Out[173]:

```
(18, 9)
```

In [174]:

```
merge_df=pd.merge(df, df1, how='outer')
merge_df
# will merge all data
```

Out[174]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS	
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0	
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0	
2	US	329.74	9833.52	19485.39	N.America	1776-07-04	NaN	NaN	
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0	
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	14172139.0	383757.0	South
5	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0	
6	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0	
7	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0	
8	Russia	146.79	17098.25	1530.75	NaN	12/6/1992	4736121.0	107103.0	

In [175]:

```
merge_df.shape
```

Out[175]:

```
(222, 9)
```

In [176]:

```
# Let's try how='left' or 'right'

test1=pd.merge(df, df1, how='left')
test1
```

Out[176]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS	R
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0	
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0	
2	US	329.74	9833.52	19485.39	N.America	1776-07-04	NaN	NaN	
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0	
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	14172139.0	383757.0	South A
5	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0	
6	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0	
7	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0	
8	Russia	146.79	17098.25	1530.75	NaN	12/6/1992	[illegible]	[illegible]	

how='left', will merge base on left file, in this example is df

In [177]:

```
test2=pd.merge(df, df1, how='right')
test2
```

Out[177]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS	
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0	
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0	
2	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0	
3	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	14172139.0	383757.0	South
4	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0	
5	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0	
6	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0	
7	Russia	146.79	17098.25	1530.75	NaN	12/6/1992	4736121.0	107103.0	
8	Mexico	126.58	1964.38	1158.23	N.America	1810-09-	2319519.0	214095.0	North

how='right', will merge base on right file, in this example is df1

Concatenating

With concatenation, your datasets are just stitched together along an axis — either the row axis or column axis.

In [178]:

```
concat_df=pd.concat([df, df1], axis=1)
concat_df
```

Out[178]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	COUNTRY	CASES	DEAT
0	China	1398.72	9596.96	12234.78	Asia	NaN	United States	32669121.0	58422
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	India	16257309.0	18692
2	US	329.74	9833.52	19485.39	N.America	1776-07-04	Brazil	14172139.0	38375
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	France	5408606.0	10216
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	Russia	4736121.0	10710
5	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	Turkey	4501382.0	3732
6	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	United Kingdom	4398431.0	12734

Let's try append the data

Let's call add new dataset call data4

In [179]:

```
df2=pd.read_csv('data4.csv')
df2
```

Out[179]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	Egypt	93	640.68	375.77	Asia	1867-07-01
1	Germany	81	242.50	245.63	Europe	1789-07-14
2	Iran	80	301.34	143.00	Europe	NaN
3	Turkey	79	NaN	250.00	NaN	NaN

In [180]:

```
test3 = df.append(df2, ignore_index=True, sort=False)
test3
```

Out[180]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	China	1398.72	9596.96	12234.78	Asia	NaN
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947
2	US	329.74	9833.52	19485.39	N.America	1776-07-04
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07
5	Pakistan	205.71	881.91	302.14	Asia	14/8/1947
6	Nigeria	200.96	923.77	375.77	Africa	1/10/1960
7	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971
8	Russia	146.79	17098.25	1530.75	NaN	12/6/1992
9	Mexico	126.58	1964.38	1158.23	N.America	1810-09-16

data from df 1 and data 4 are combine at row level

LET'S MOVE TO GROUPBY

In [181]:

```
df.head(2)
```

Out[181]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	China	1398.72	9596.96	12234.78	Asia	NaN
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947

In [182]:

```
df1.head(2)
```

Out[182]:

	COUNTRY	CASES	DEATHS	REGION
0	United States	32669121.0	584226.0	North America
1	India	16257309.0	186928.0	Asia

In [183]:

```
df2.head()
```

Out[183]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY
0	Egypt	93	640.68	375.77	Asia	1867-07-01
1	Germany	81	242.50	245.63	Europe	1789-07-14
2	Iran	80	301.34	143.00	Europe	NaN
3	Turkey	79	NaN	250.00	NaN	NaN

In [184]:

```
merge_df.head()
```

Out[184]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0
2	US	329.74	9833.52	19485.39	N.America	1776-07-04	NaN	NaN
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07	14172139.0	383757.0

In [185]:

```
merge_df.shape
```

Out[185]:

```
(222, 9)
```

In [186]:

```
merge_df.dtypes
```

Out[186]:

```
COUNTRY         object
POPULATION     float64
AREA           float64
GDP            float64
CONTINENTS      object
IND_DAY         object
CASES          float64
DEATHS         float64
REGION          object
dtype: object
```

CONTINENT and REGION actually referring to the same thing. Let's try to fix it

Checking the elements

In [187]:

```
reg=merge_df.groupby('REGION').sum()
reg
```

Out[187]:

	POPULATION	AREA	GDP	CASES	DEATHS
REGION					
Africa	244.34	3305.51	543.33	4513248.0	119538.0
Asia	3535.50	18927.50	21405.59	35616438.0	482532.0
Australia/Oceania	25.47	7692.02	1408.68	61971.0	1184.0
Europe	357.19	18397.38	9750.28	43483441.0	989869.0
North America	164.17	11949.05	2805.35	37760260.0	852502.0
South America	255.26	11296.17	2693.00	23897427.0	639607.0

In [188]:

```
con=merge_df.groupby('CONTINENTS').sum()
con
```

Out[188]:

	POPULATION	AREA	GDP	CASES	DEATHS
CONTINENTS					
Africa	244.34	3305.51	543.33	284951.0	5242.0
Asia	3535.50	18927.50	21405.59	20342739.0	276648.0
Europe	276.84	1541.63	10850.76	12567605.0	302214.0
N.America	493.91	21782.57	22290.74	3475353.0	237917.0
Oceania	25.47	7692.02	1408.68	29626.0	910.0
S.America	255.26	11296.17	2693.00	16968907.0	444377.0

In [189]:

```
merge_df.CONTINENTS=merge_df.CONTINENTS.replace(['N.America','S.America'],['North America',
merge_df
```

Out[189]:

	COUNTRY	POPULATION	AREA	GDP	CONTINENTS	IND_DAY	CASES	DEATHS	
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0	
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0	
2	US	329.74	9833.52	19485.39	North America	1776-07-04	NaN	NaN	
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0	
4	Brazil	210.32	8515.77	2055.51	South America	1822-09-07	14172139.0	383757.0	South
5	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0	
6	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0	
7	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0	

In [190]:

```
reg=merge_df.groupby('REGION').sum()
reg
```

Out[190]:

	POPULATION	AREA	GDP	CASES	DEATHS
REGION					
Africa	244.34	3305.51	543.33	4513248.0	119538.0
Asia	3535.50	18927.50	21405.59	35616438.0	482532.0
Australia/Oceania	25.47	7692.02	1408.68	61971.0	1184.0
Europe	357.19	18397.38	9750.28	43483441.0	989869.0
North America	164.17	11949.05	2805.35	37760260.0	852502.0
South America	255.26	11296.17	2693.00	23897427.0	639607.0

In [191]:

```
con=merge_df.groupby('CONTINENTS').sum()
con
```

Out[191]:

	POPULATION	AREA	GDP	CASES	DEATHS
CONTINENTS					
Africa	244.34	3305.51	543.33	284951.0	5242.0
Asia	3535.50	18927.50	21405.59	20342739.0	276648.0
Europe	276.84	1541.63	10850.76	12567605.0	302214.0
North America	493.91	21782.57	22290.74	3475353.0	237917.0
Oceania	25.47	7692.02	1408.68	29626.0	910.0
South America	255.26	11296.17	2693.00	16968907.0	444377.0

In [192]:

```
df = df.rename(columns={'CONTINENTS': 'REGION'})
df.head()
```

Out[192]:

	COUNTRY	POPULATION	AREA	GDP	REGION	IND_DAY
0	China	1398.72	9596.96	12234.78	Asia	NaN
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947
2	US	329.74	9833.52	19485.39	N.America	1776-07-04
3	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945
4	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07

In [193]:

```
df1.head()
```

Out[193]:

	COUNTRY	CASES	DEATHS	REGION
0	United States	32669121.0	584226.0	North America
1	India	16257309.0	186928.0	Asia
2	Brazil	14172139.0	383757.0	South America
3	France	5408606.0	102164.0	Europe
4	Russia	4736121.0	107103.0	Europe

In [194]:

```
df101=pd.merge(df, df1)
df101
```

Out[194]:

	COUNTRY	POPULATION	AREA	GDP	REGION	IND_DAY	CASES	DEATHS
0	China	1398.72	9596.96	12234.78	Asia	NaN	90566.0	4636.0
1	India	1351.16	3287.26	2575.67	Asia	15/8/1947	16257309.0	186928.0
2	Indonesia	268.07	1910.93	1015.54	Asia	17/8/1945	1626812.0	44172.0
3	Pakistan	205.71	881.91	302.14	Asia	14/8/1947	784108.0	16842.0
4	Nigeria	200.96	923.77	375.77	Africa	1/10/1960	164588.0	2061.0
5	Bangladesh	167.09	147.57	245.63	Asia	26/3/1971	736074.0	10781.0
6	Japan	126.22	377.97	4872.42	Asia	NaN	547137.0	9777.0
7	Germany	83.02	357.11	3693.20	Europe	NaN	3238054.0	81693.0
8	France	67.02	640.68	2582.49	Europe	1789-07-14	5408606.0	102164.0
9	Italy	60.36	301.34	1943.84	Europe	NaN	3920945.0	118357.0
10	Algeria	43.38	2381.74	167.56	Africa	5/7/1962	120363.0	3181.0
11	Kazakhstan	18.53	2724.90	159.41	Asia	16/12/1991	300733.0	3512.0

In [195]:

```
reg=df101.groupby('REGION').sum()
reg
```

Out[195]:

```
         POPULATION     AREA       GDP       CASES    DEATHS
REGION
Africa       244.34   3305.51    543.33    284951.0    5242.0
  Asia      3535.50  18927.50  21405.59  20342739.0  276648.0
Europe       210.40   1299.13   8219.53  12567605.0  302214.0
```

In [ ]:

```

```

In [ ]:

```

```