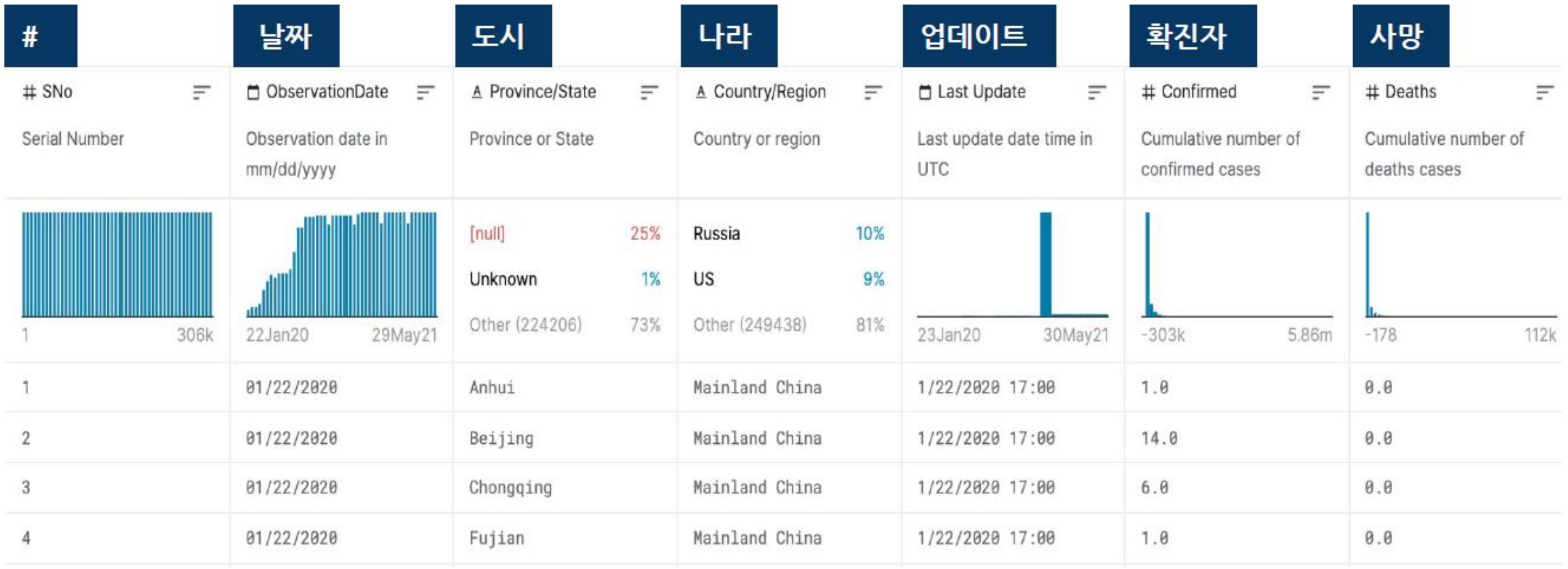
Pandas: Learn To Use DataFrame With Example

Younghoon Kim
(nongaussian@hanyang.ac.kr)

COVID-19 Dataset

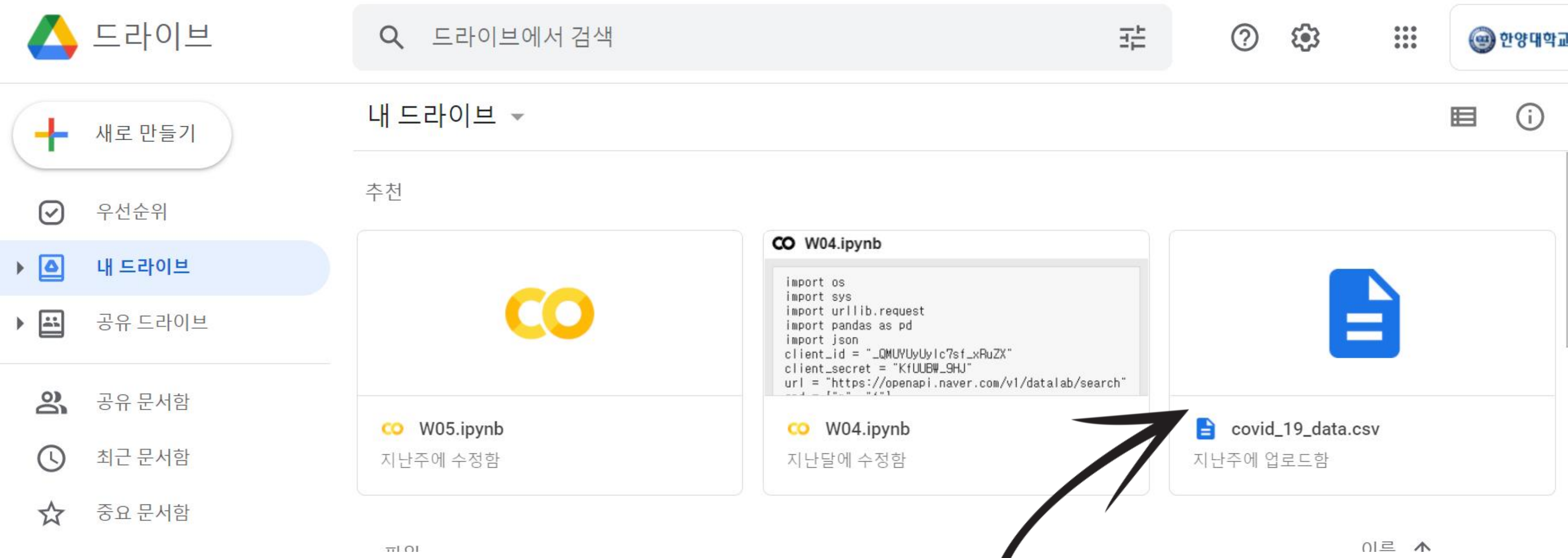
- Download (also available in LMS)
  - https://www.kaggle.com/datasets/sudalairajkumar/novel-corona-virus-2019-dataset?select=covid_19_data.csv
  - 22.54MB

Attributes



#	날짜	도시	나라	업데이트	확진자	사망
# SNo	ObservationDate	Province/State	Country/Region	Last Update	# Confirmed	# Deaths
Serial Number	Observation date in mm/dd/yyyy	Province or State	Country or region	Last update date time in UTC	Cumulative number of confirmed cases	Cumulative number of deaths cases
1 306k	22Jan20 29May21	[null] 25% Unknown 1% Other (224206) 73%	Russia 10% US 9% Other (249438) 81%	23Jan20 30May21	-303k 5.86m	-178 112k
1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0
2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0
3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0
4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0

Upload Files on Colab



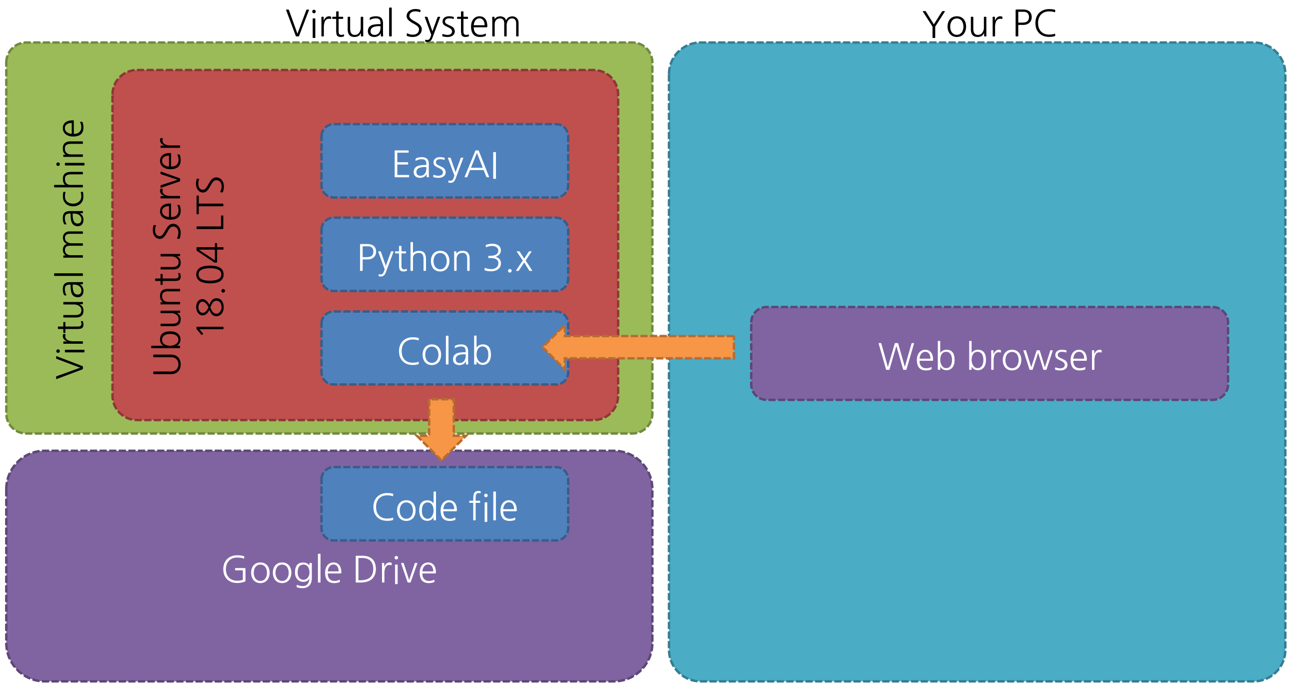
```
import os
import sys
import urllib.request
import pandas as pd
import json
client_id = "_QMUYUyUylc7sf_xRuZX"
client_secret = "KfUUBW_9HJ"
url = "https://openapi.naver.com/v1/datalab/search"
```

Drag & Drop

Read Files into Pandas DataFrame

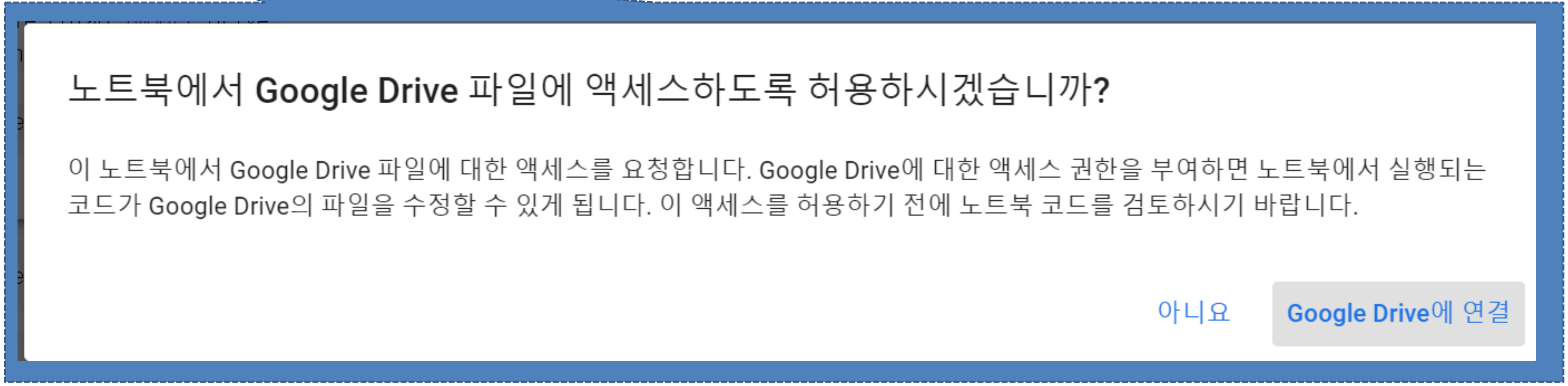
```
import pandas as pd
import os, sys
from google.colab import drive
drive.mount('/content/drive')

df = pd.read_csv('drive/MyDrive/covid_19_data.csv’)
df
```

Recall..


Virtual System
Your PC
Virtual machine
Ubuntu Server 18.04 LTS
EasyAI
Python 3.x
Colab
Web browser
Code file
Google Drive

Read Files into Pandas DataFrame

```
import pandas as pd
import os, sys
from google.colab import drive
drive.mount('/content/drive')
```

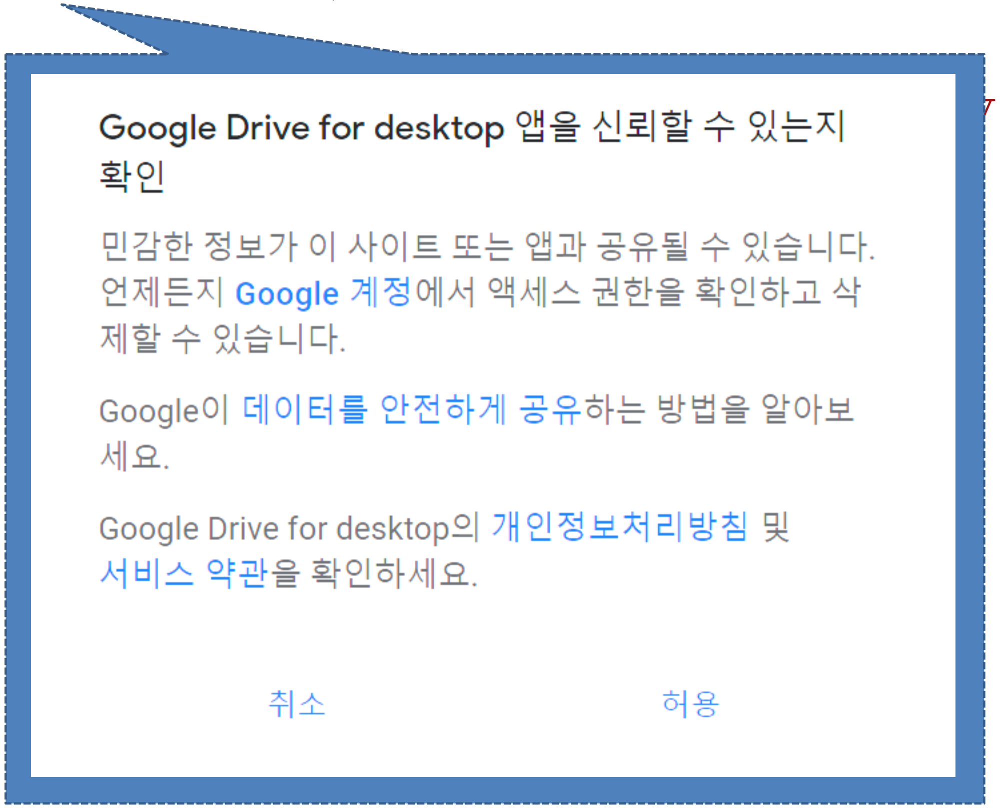


노트북에서 Google Drive 파일에 액세스하도록 허용하시겠습니까?

이 노트북에서 Google Drive 파일에 대한 액세스를 요청합니다. Google Drive에 대한 액세스 권한을 부여하면 노트북에서 실행되는 코드가 Google Drive의 파일을 수정할 수 있게 됩니다. 이 액세스를 허용하기 전에 노트북 코드를 검토하시기 바랍니다.

Read Files into Pandas DataFrame

```
import pandas as pd
import os, sys
from google.colab import drive
drive.mount('/content/drive')

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



Google Drive for desktop 앱을 신뢰할 수 있는지 확인

민감한 정보가 이 사이트 또는 앱과 공유될 수 있습니다. 언제든지 Google 계정에서 액세스 권한을 확인하고 삭제할 수 있습니다.

Google이 데이터를 안전하게 공유하는 방법을 알아보세요.

Google Drive for desktop의 개인정보처리방침 및 서비스 약관을 확인하세요.

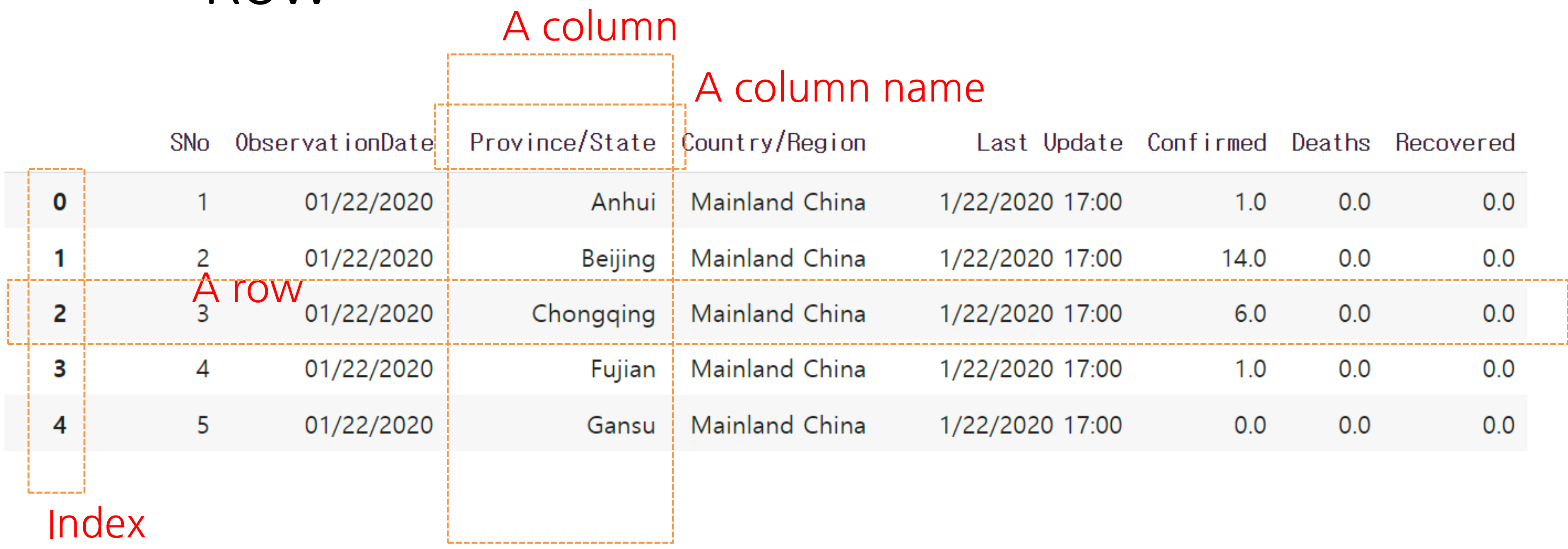
Read Files into Pandas DataFrame

```python
import pandas as pd
import os, sys
from google.colab import drive
drive.mount('/content/drive')

df = pd.read_csv('drive/MyDrive/covid_19_data.csv’)
df
```

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

DataFrame

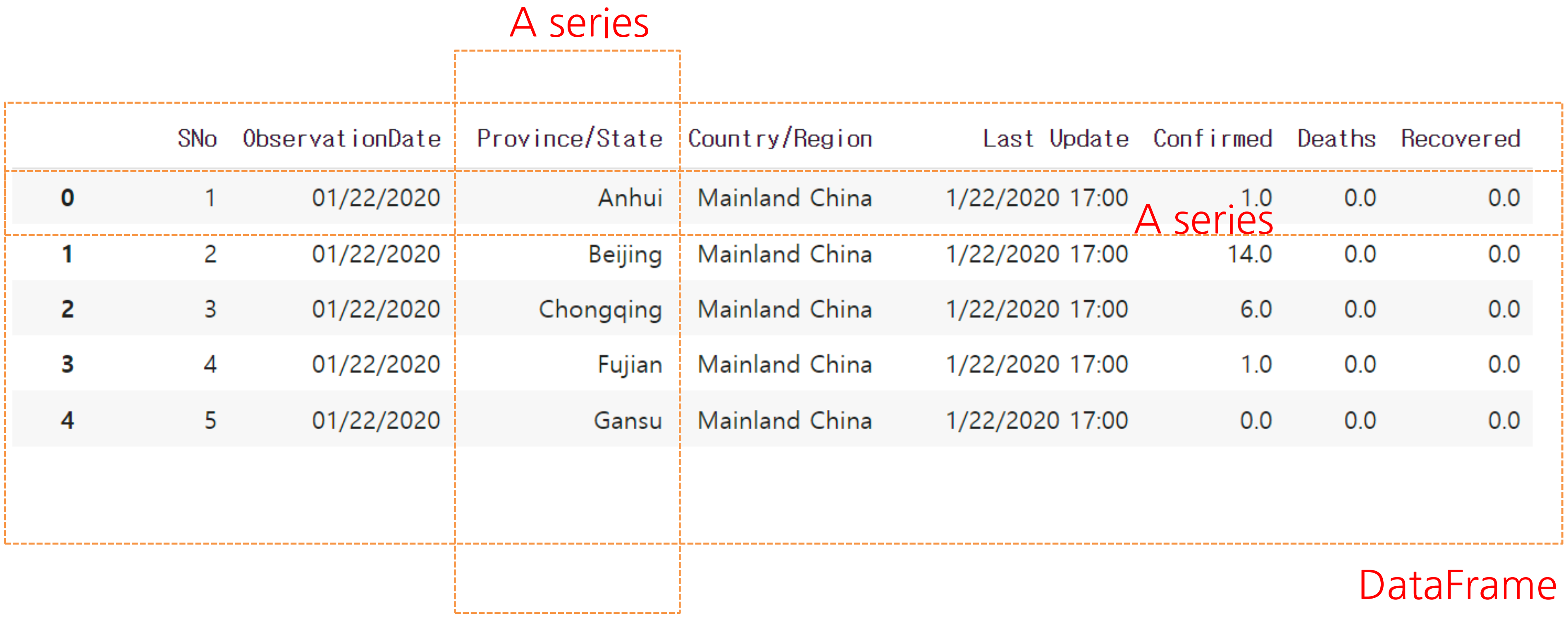
- A table
  - Column
  - Row



	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

DataFrame

- 2-dim table = DataFrame
- 1-dim vector = Series


A series
A series
DataFrame

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

Accessing DataFrame

- Index
- Range of indice
- List of True/False for selecting each row

df.loc[○ , ○]

- Column name
- Range of column names
- List of column names to select

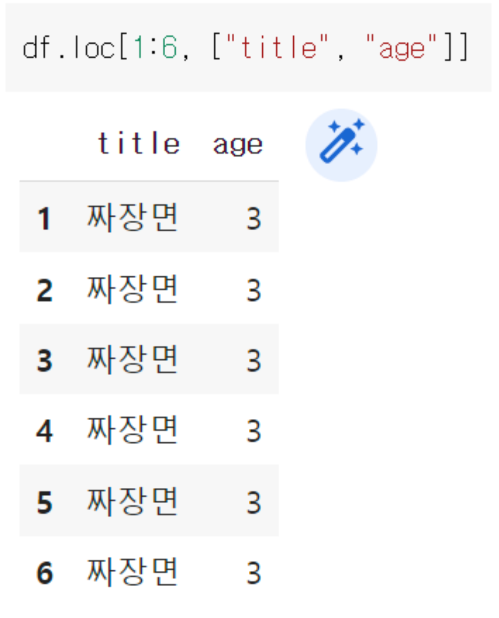
Accessing DataFrame

- Index = 0

df.loc[0 , :]

- Range of columns = all columns

Example



```
df.loc[1:6, ["title", "age"]]
```

```
   title  age
1  짜장면    3
2  짜장면    3
3  짜장면    3
4  짜장면    3
5  짜장면    3
6  짜장면    3
```

Range of indices:
From 1 to 6

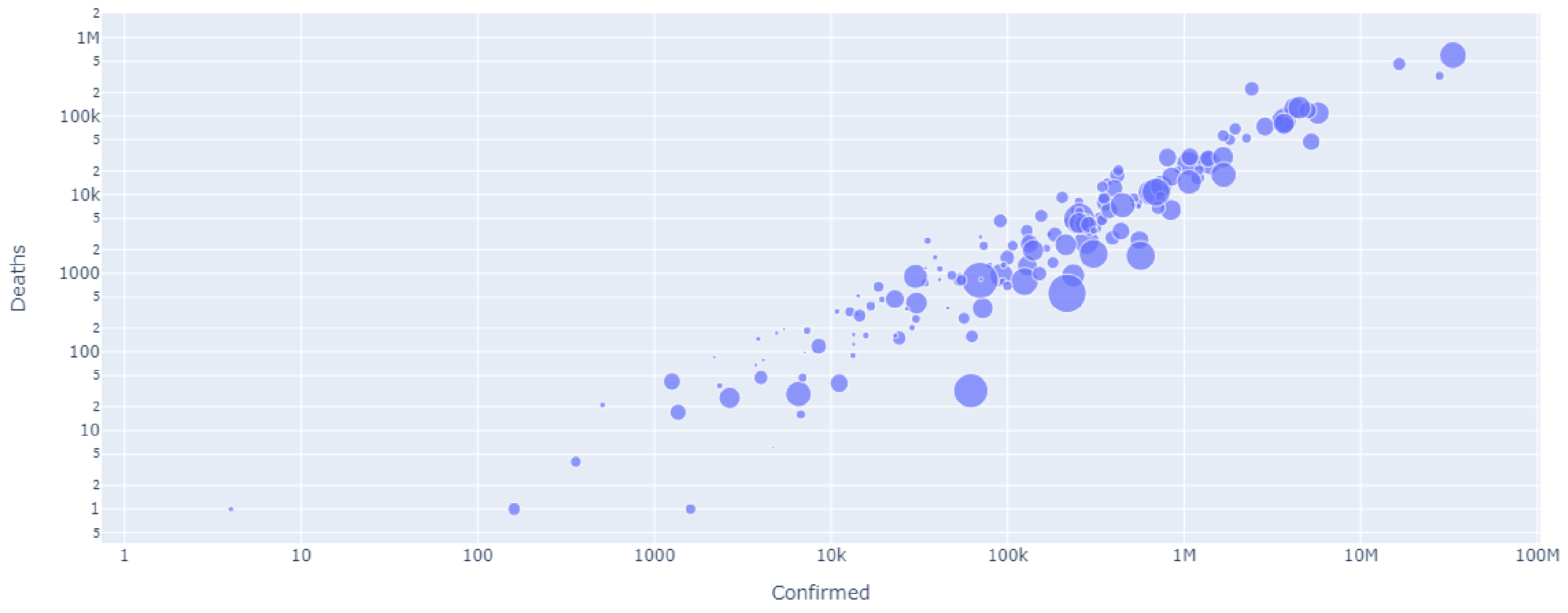
df.loc[1:6 , ["title", "age"]]

- Select two columns "title" and "age"

Practice

- Read covid_19_data.csv file
- Access the dataframe

Goal of Today’s Pandas Study



Change Column Name

Source

Target

```
df = df.rename(columns={
     'ObservationDate': 'Date',
     'Province/State': 'City',
     'Country/Region': 'Country’
})
```

Column list

```
df.columns
```

```
Index(['SNo', 'Date', 'City', 'Country', 'Last Update',
'Confirmed', 'Deaths', 'Recovered'], dtype='object')
```

Drop Columns

Columns to remove

```
df = df.drop(columns=['Last Update'])
```

```
df.columns
```

```
Index(['SNo', 'Date', 'City', 'Country',
'Confirmed', 'Deaths', 'Recovered'], dtype='object')
```

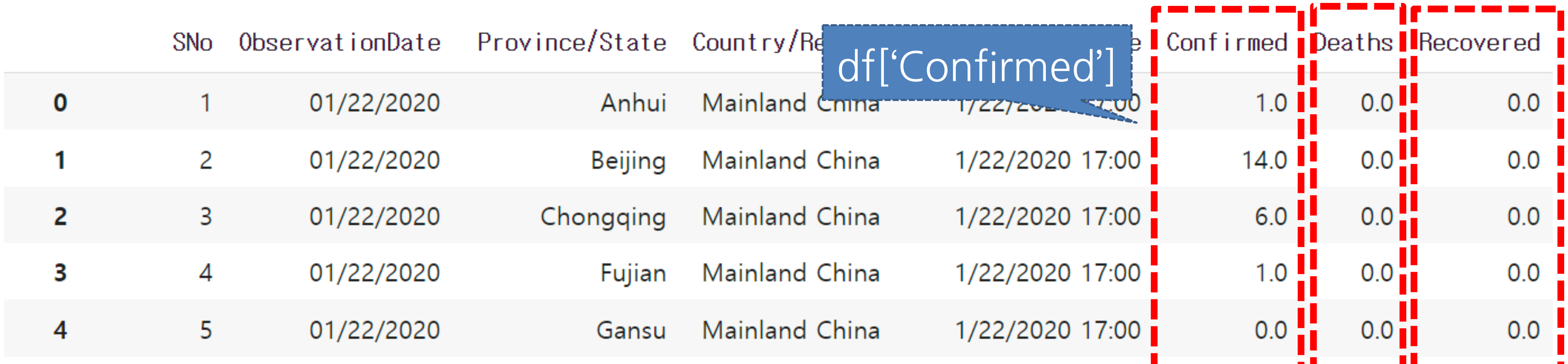
Add A New Column

- Active
  - Confirmed - Death - Recovered

df['Recovered']

df['Death']

df['Confirmed']



	SNo	ObservationDate	Province/State	Country/Re	e	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/202[illegible]:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

Add A New Column

```
df['Active'] = df['Confirmed'] - df['Deaths'] - df['Recovered']
df
```

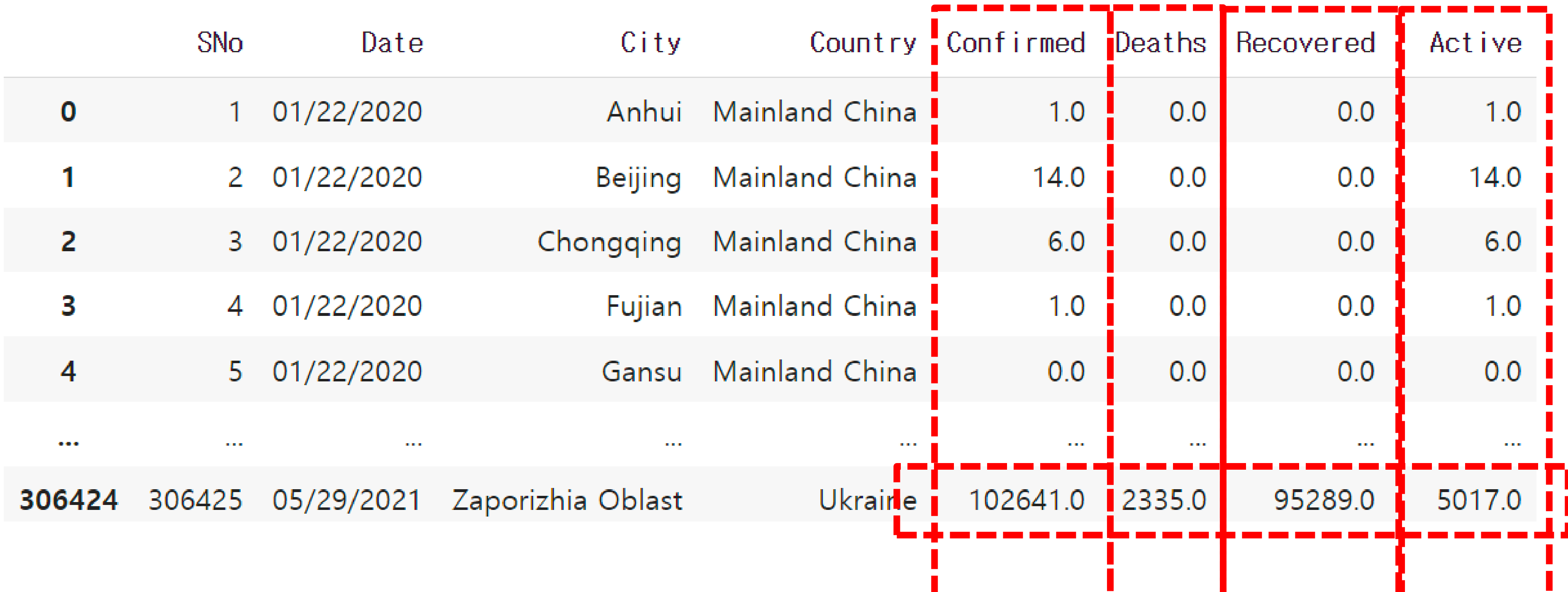
df[‘Active’]

	SNo	Date	City	Country	Confirmed	Deaths	Recovered	Active
0	1	01/22/2020	Anhui	Mainland China	1.0	0.0	0.0	1.0
1	2	01/22/2020	Beijing	Mainland China	14.0	0.0	0.0	14.0
2	3	01/22/2020	Chongqing	Mainland China	6.0	0.0	0.0	6.0
3	4	01/22/2020	Fujian	Mainland China	1.0	0.0	0.0	1.0
4	5	01/22/2020	Gansu	Mainland China	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
306424	306425	05/29/2021	Zaporizhia Oblast	Ukraine	102641.0	2335.0	95289.0	5017.0

Element-wise Operation

- Operation between series is computed by element-wise

```
df['Active'] = df['Confirmed'] - df['Deaths'] - df['Recovered']
```



	SNo	Date	City	Country	Confirmed	Deaths	Recovered	Active
0	1	01/22/2020	Anhui	Mainland China	1.0	0.0	0.0	1.0
1	2	01/22/2020	Beijing	Mainland China	14.0	0.0	0.0	14.0
2	3	01/22/2020	Chongqing	Mainland China	6.0	0.0	0.0	6.0
3	4	01/22/2020	Fujian	Mainland China	1.0	0.0	0.0	1.0
4	5	01/22/2020	Gansu	Mainland China	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
306424	306425	05/29/2021	Zaporizhia Oblast	Ukraine	102641.0	2335.0	95289.0	5017.0

Practice

- Change the long column names
- Drop columns
- Add columns

Exercise

- Retrieve the records of South Korea

Exercise

- Retrieve the records of South Korea

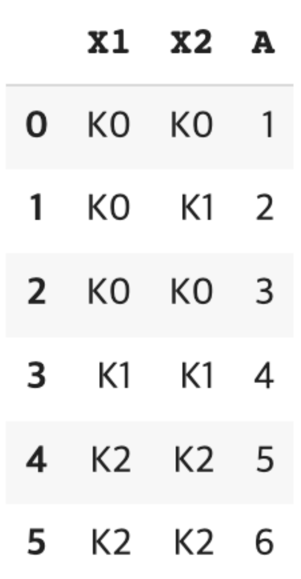
```
df_korea = df.loc[df['Country'] == 'South Korea']
df_korea
```

```
          SNo        Date  City      Country  Confirmed  Deaths  Recovered  Active
37         38  01/22/2020   NaN  South Korea        1.0     0.0        0.0     1.0
77         78  01/23/2020   NaN  South Korea        1.0     0.0        0.0     1.0
125       126  01/24/2020   NaN  South Korea        2.0     0.0        0.0     2.0
168       169  01/25/2020   NaN  South Korea        2.0     0.0        0.0     2.0
216       217  01/26/2020   NaN  South Korea        3.0     0.0        0.0     3.0
...       ...         ...   ...          ...        ...     ...        ...     ...
302749 302750  05/25/2021   NaN  South Korea   137682.0  1940.0   127582.0  8160.0
```

Grouping Data Using Gro

- Given a DataFrame as

```
import pandas as pd

df = pd.DataFrame({
   'X1': ['K0', 'K0', 'K0', 'K1', 'K2', 'K2'],
   'X2': ['K0', 'K1', 'K0', 'K1', 'K2', 'K2'],
   'A': [1, 2, 3, 4, 5, 6]}
)
```

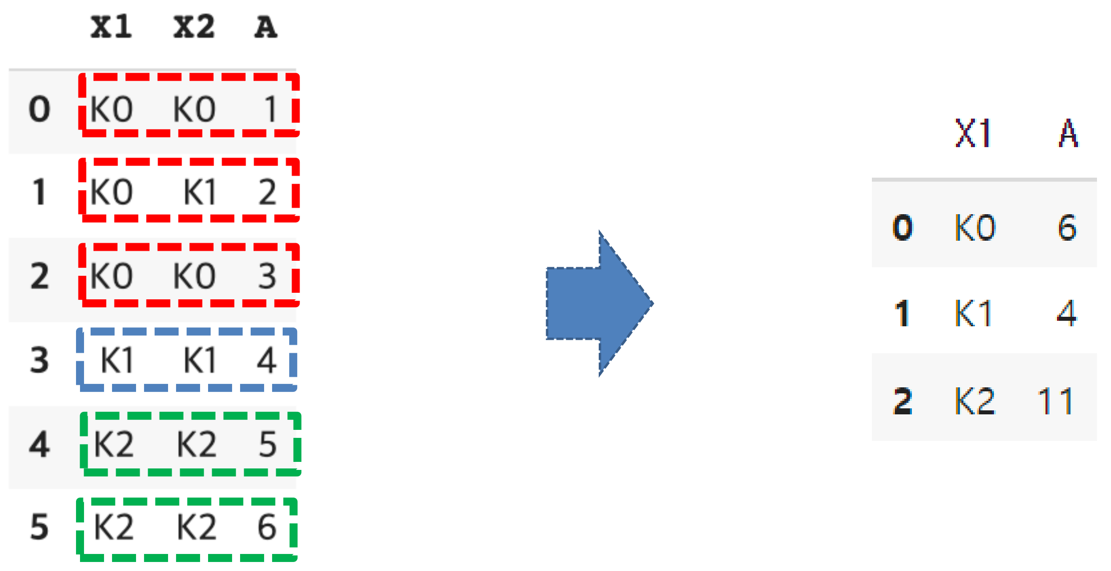


	X1	X2	A
0	K0	K0	1
1	K0	K1	2
2	K0	K0	3
3	K1	K1	4
4	K2	K2	5
5	K2	K2	6

- Answer the data frames the following two lines would return
  - `df.groupby(['X1'], as_index=False).sum()`
  - `df.groupby(['X1', 'X2'], as_index=False).sum()`

Groupby

```
df.groupby(['X1'], as_index=False).sum()
```

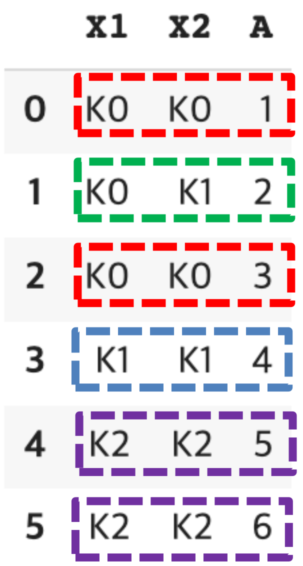


	X1	X2	A
0	K0	K0	1
1	K0	K1	2
2	K0	K0	3
3	K1	K1	4
4	K2	K2	5
5	K2	K2	6

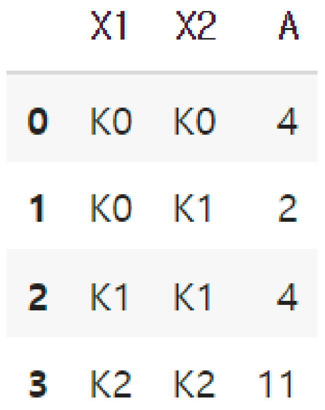
	X1	A
0	K0	6
1	K1	4
2	K2	11

Groupby

```
df.groupby(['X1', 'X2'], as_index=False).sum()
```



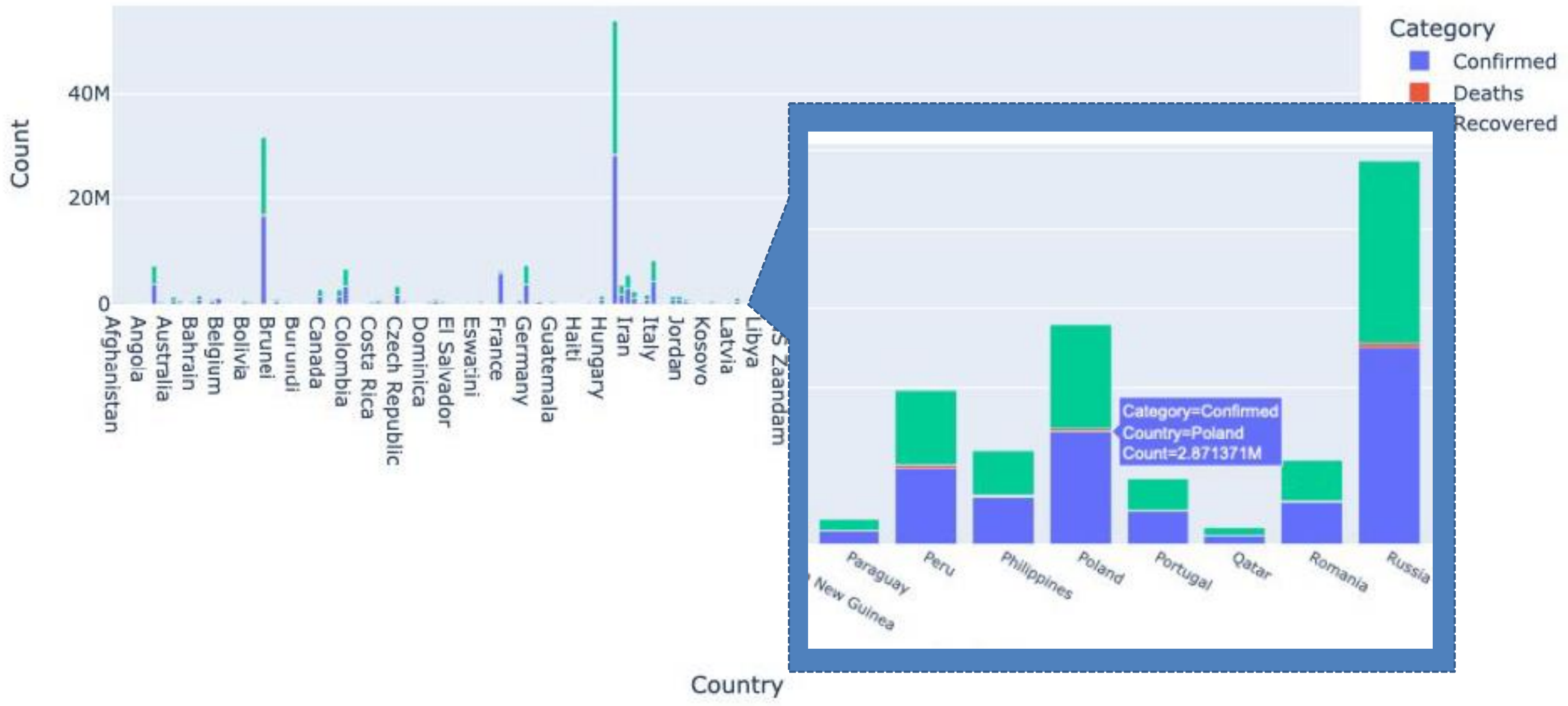
	X1	X2	A
0	K0	K0	1
1	K0	K1	2
2	K0	K0	3
3	K1	K1	4
4	K2	K2	5
5	K2	K2	6



	X1	X2	A
0	K0	K0	4
1	K0	K1	2
2	K1	K1	4
3	K2	K2	11

Stacked Histogram

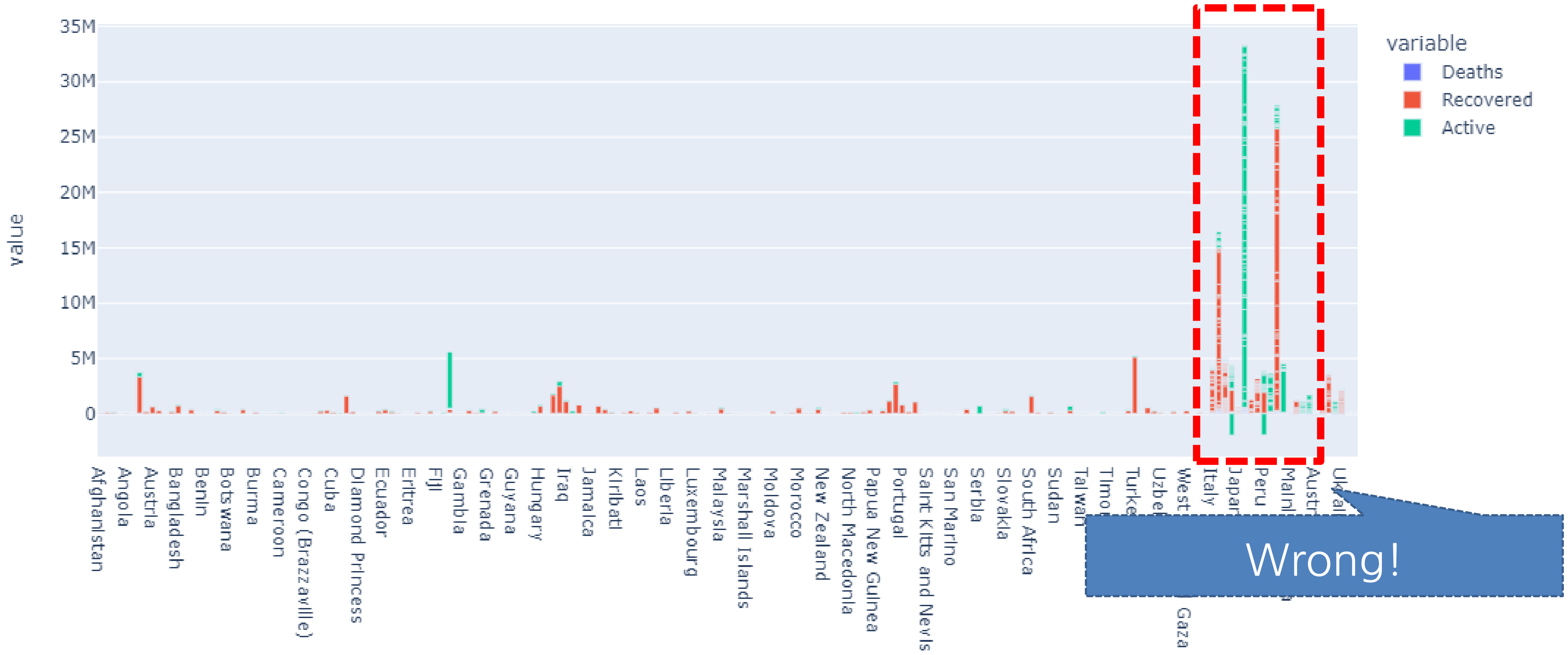
- Plot Confirmed / Death / Recovered by countries
  - Data = 05/29/2021



Stacked Histogram

```
import plotly.express as px

df_lastday = df.loc[df['Date'] == '05/29/2021']
df_lastday

fig = px.bar(df_lastday, x='Country’,
             y=['Deaths', 'Recovered', 'Active'])
fig.show()
```

Stacked Histogram



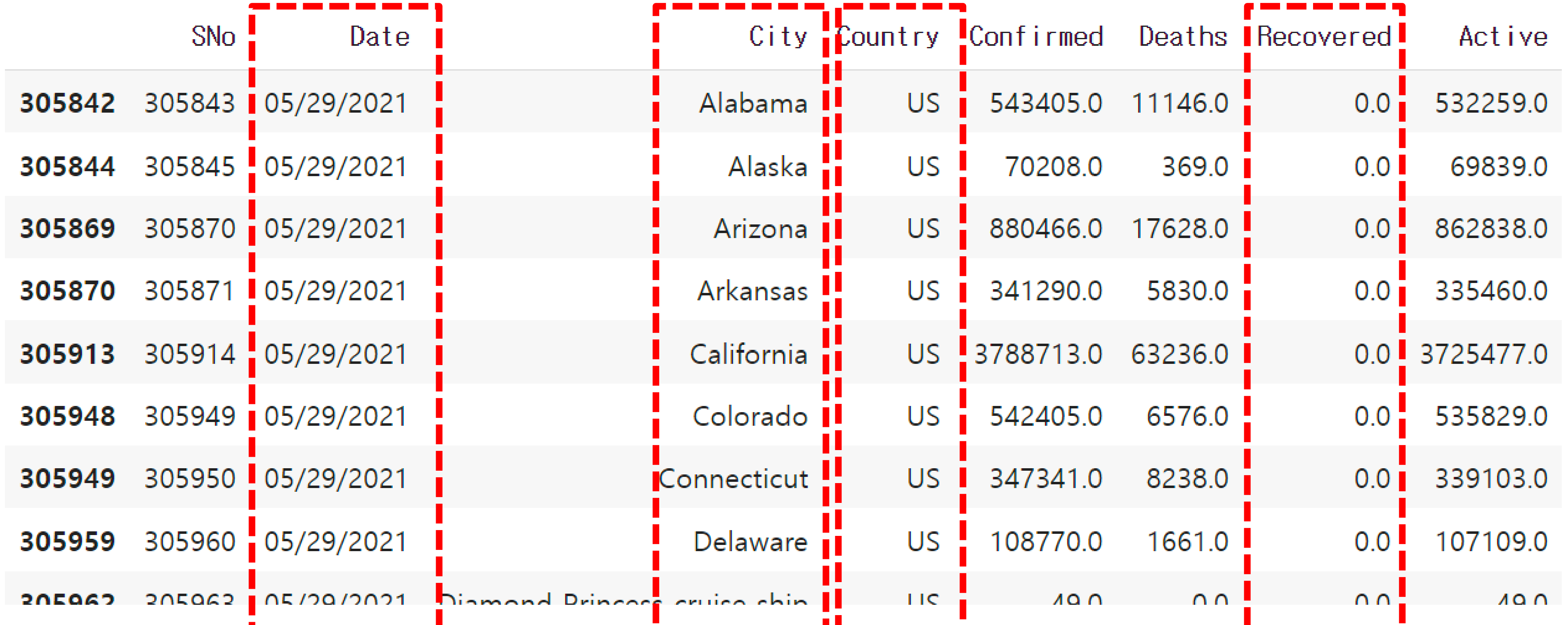
Wrong!

See Data

- Retrieve data of the last date for US

```
df.loc[(df['Date'] == '05/29/2021') & (df['Country'] == 'US')]
```

Data is by city for US



	SNo	Date	City	Country	Confirmed	Deaths	Recovered	Active
305842	305843	05/29/2021	Alabama	US	543405.0	11146.0	0.0	532259.0
305844	305845	05/29/2021	Alaska	US	70208.0	369.0	0.0	69839.0
305869	305870	05/29/2021	Arizona	US	880466.0	17628.0	0.0	862838.0
305870	305871	05/29/2021	Arkansas	US	341290.0	5830.0	0.0	335460.0
305913	305914	05/29/2021	California	US	3788713.0	63236.0	0.0	3725477.0
305948	305949	05/29/2021	Colorado	US	542405.0	6576.0	0.0	535829.0
305949	305950	05/29/2021	Connecticut	US	347341.0	8238.0	0.0	339103.0
305959	305960	05/29/2021	Delaware	US	108770.0	1661.0	0.0	107109.0
305962	305963	05/29/2021	Diamond Princess cruise ship	US	49.0	0.0	0.0	49.0

Group By Country

- Group by countries and sum the figures with the data of date`'05/29/2021 '`

Group By Country

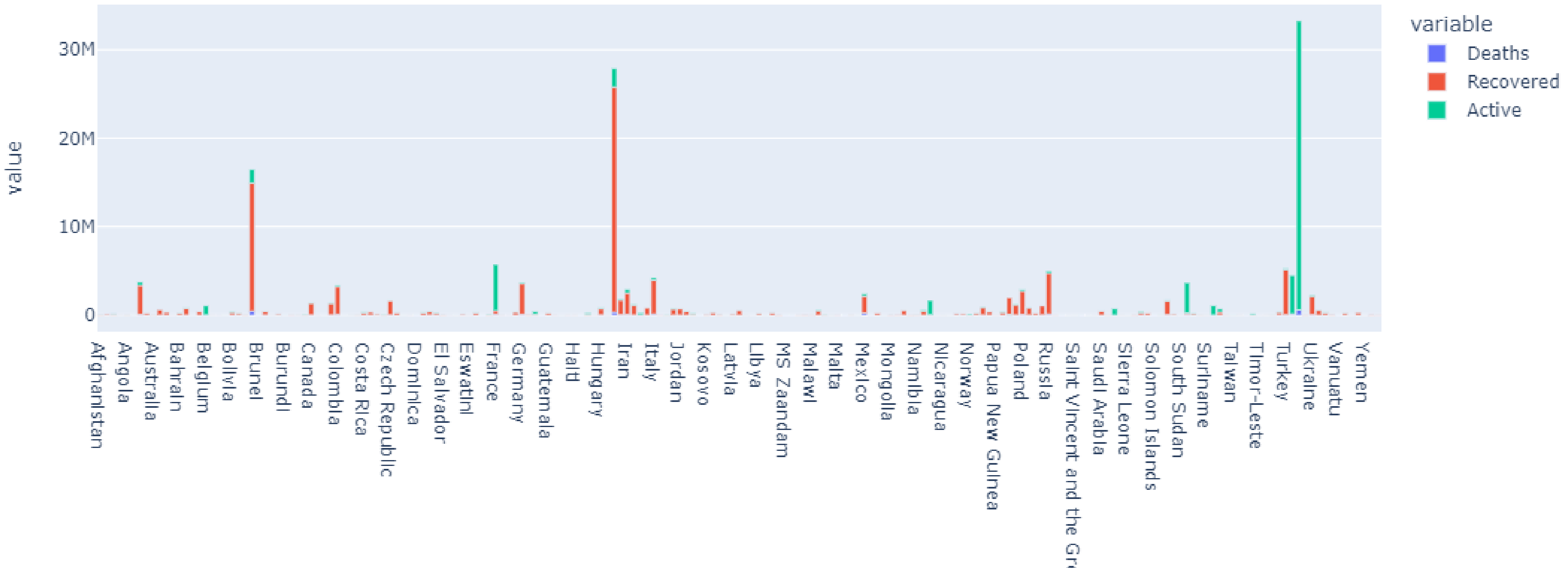
- Group by countries and sum the values with the data of date'05/29/2021 '

```
df_lastday = df.loc[df['Date'] == '05/29/2021’]\
                  .groupby(['Country'], as_index=False).sum()
df_lastday
```

	Country	SNo	Confirmed	Deaths	Recovered	Active
0	Afghanistan	305665	70111.0	2899.0	57281.0	9931.0
1	Albania	305666	132297.0	2449.0	129215.0	633.0
2	Algeria	305667	128456.0	3460.0	89419.0	35577.0
3	Andorra	305668	13693.0	127.0	13416.0	150.0
4	Angola	305669	34180.0	757.0	27646.0	5777.0

Stacked Histogram

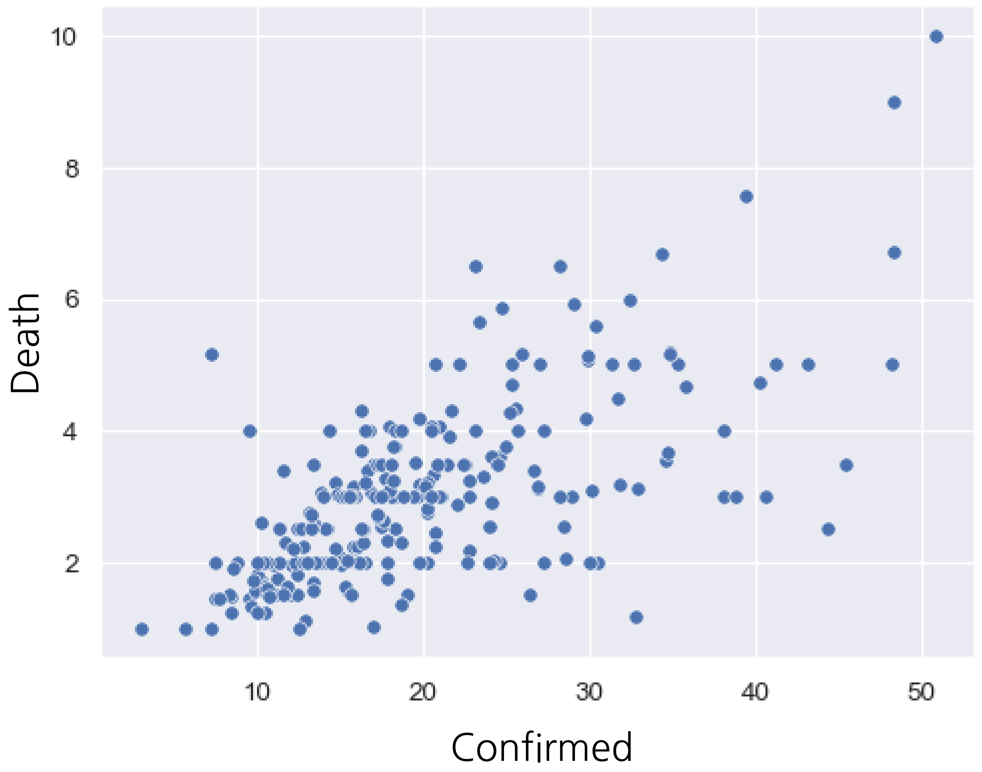
```
import plotly.express as px

fig = px.bar(df_lastday, x='Country’,
                        y=['Deaths', 'Recovered', 'Active'])
fig.show()
```



Correlation Between Confirmed and Death

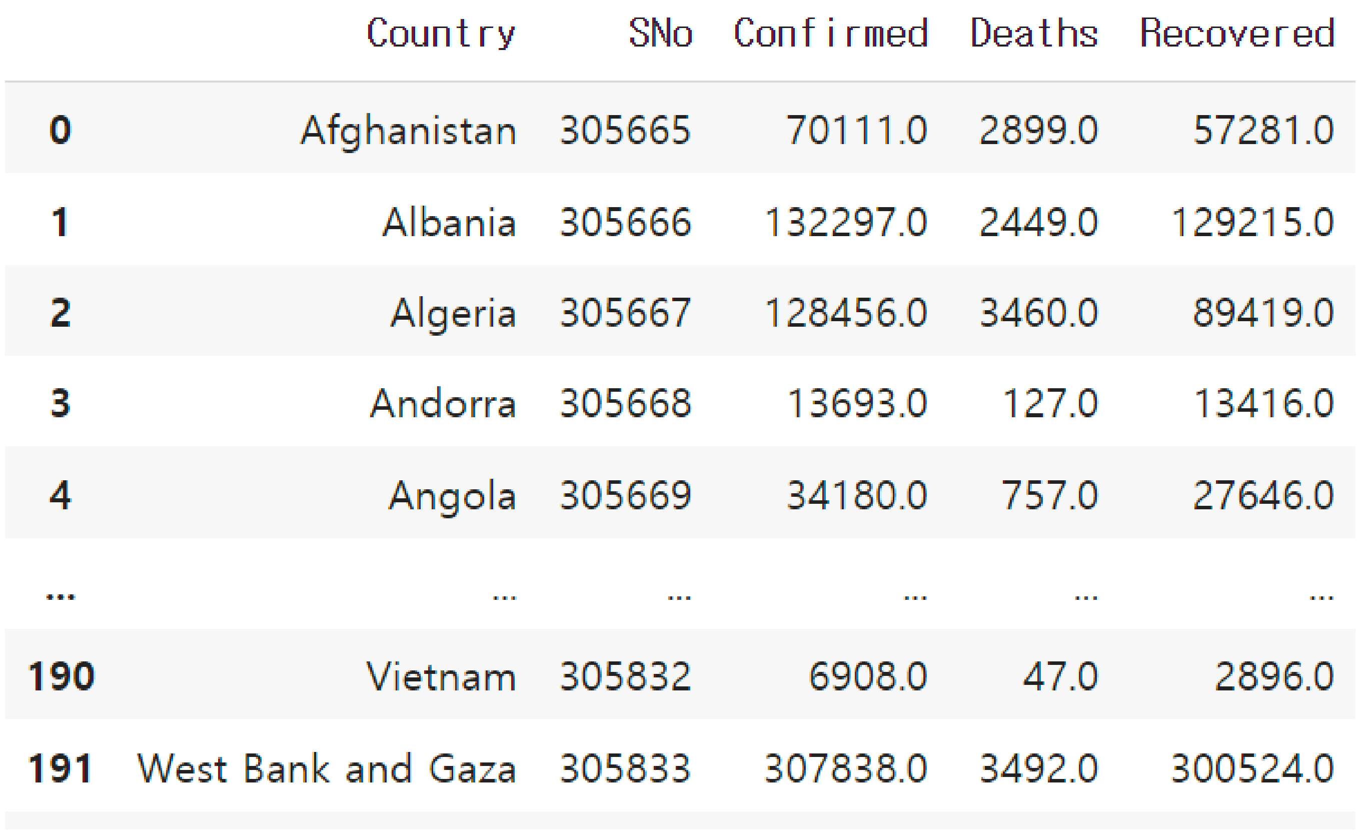
- Plot a graph to see the correlation between confirmed and death(Date: 05/29/2021)

Scatter graph



Group By Country

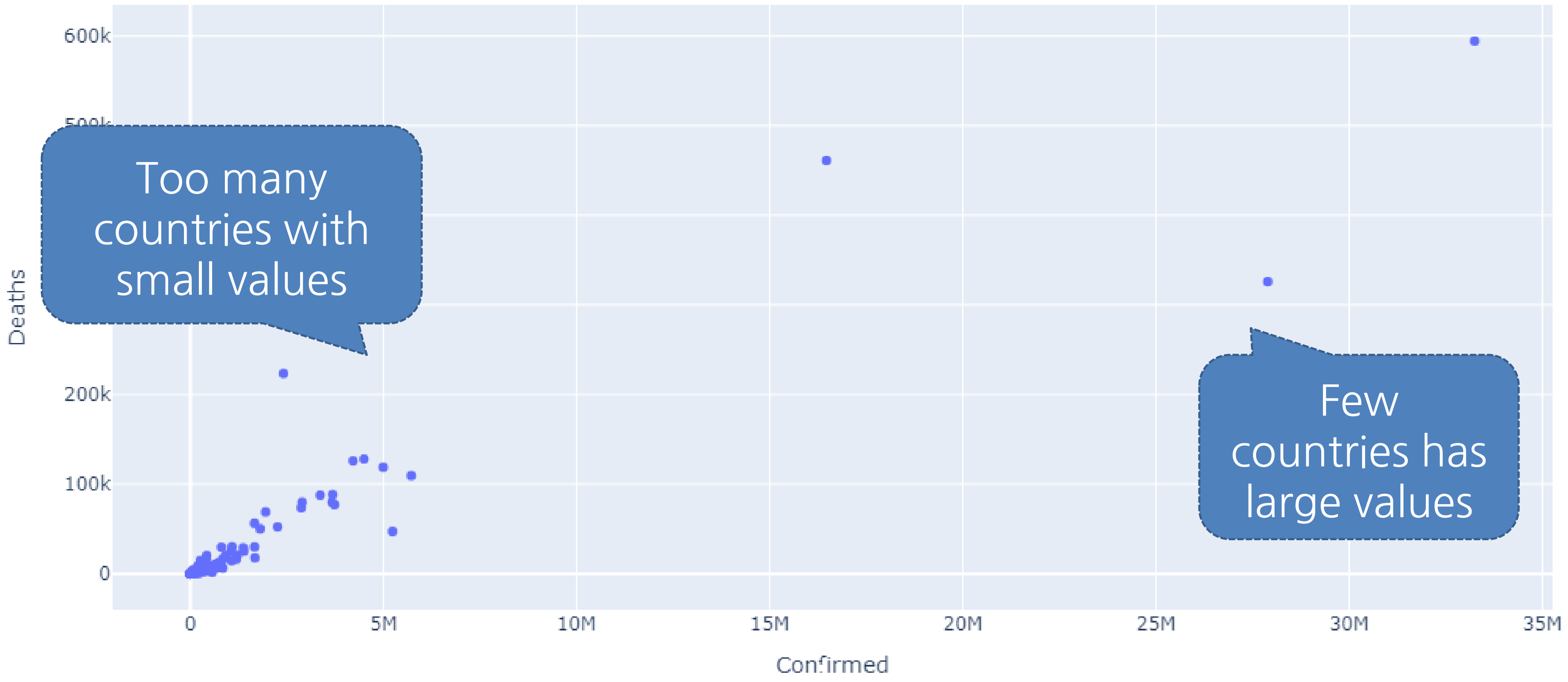
```
df_lastday = df.loc[df['Date'] == '05/29/2021’]\
                .groupby(['Country'], as_index=False).sum()
```



	Country	SNo	Confirmed	Deaths	Recovered
0	Afghanistan	305665	70111.0	2899.0	57281.0
1	Albania	305666	132297.0	2449.0	129215.0
2	Algeria	305667	128456.0	3460.0	89419.0
3	Andorra	305668	13693.0	127.0	13416.0
4	Angola	305669	34180.0	757.0	27646.0
...	...	...	...	...	...
190	Vietnam	305832	6908.0	47.0	2896.0
191	West Bank and Gaza	305833	307838.0	3492.0	300524.0

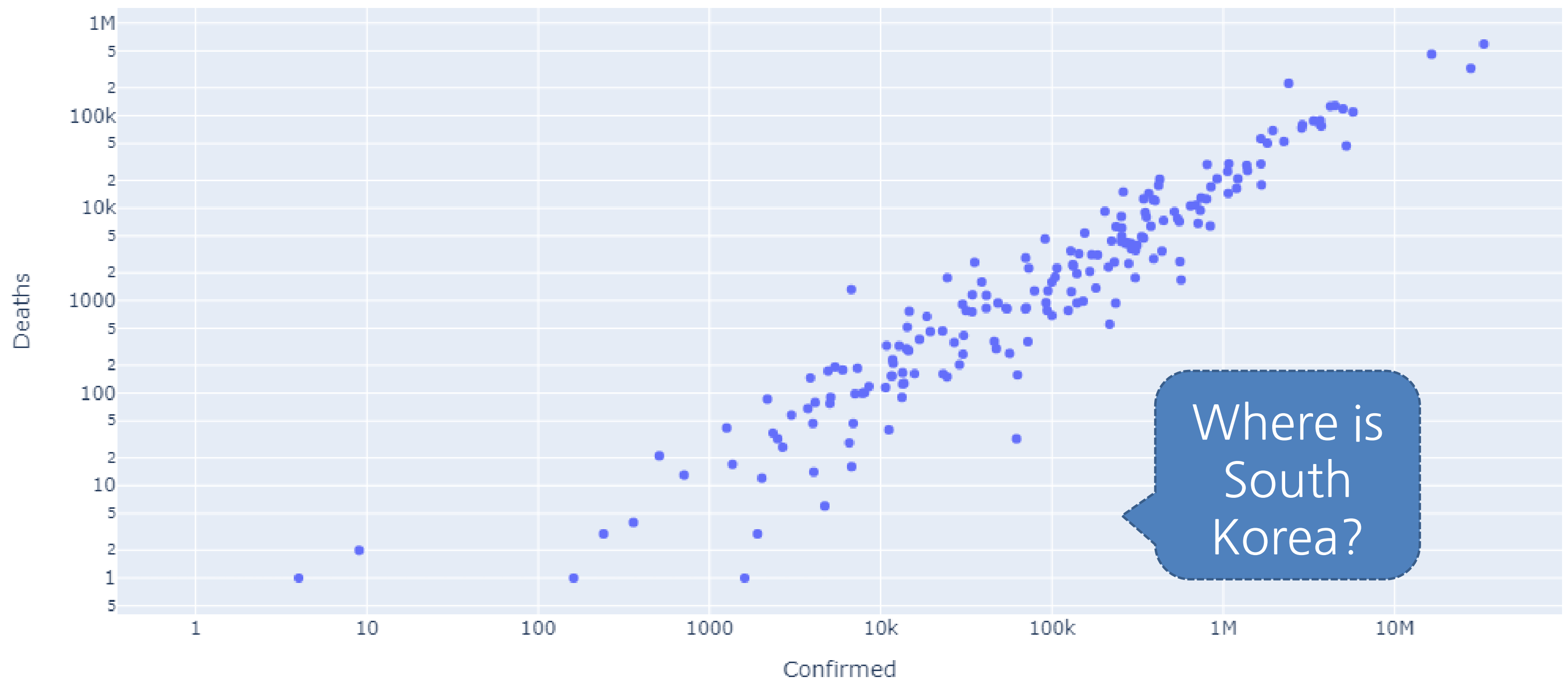
Plotting Scatter Graphs

```
import plotly.express as px
fig = px.scatter(df_lastday, x='Confirmed', y='Deaths')
fig.show()
```



Log-scale for x- and y-axis

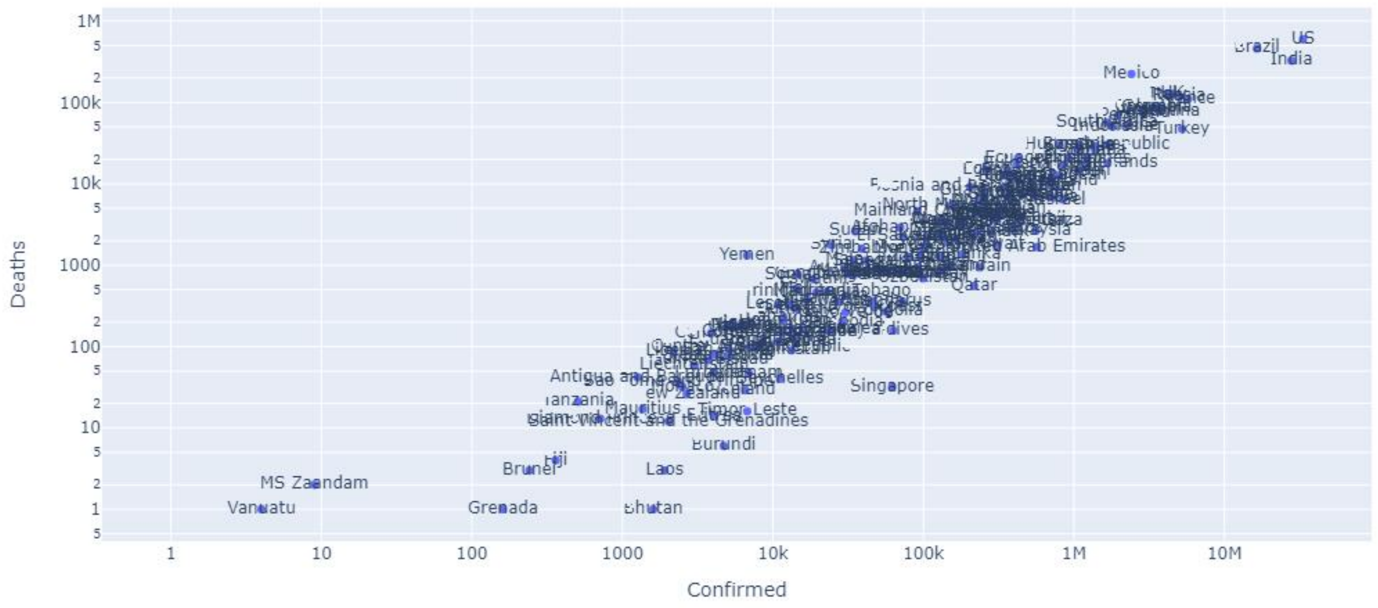
```
import plotly.express as px
fig = px.scatter(df_lastday, x='Confirmed', y='Deaths',
      log_x=True, log_y=True)
fig.show()
```



Where is South Korea?

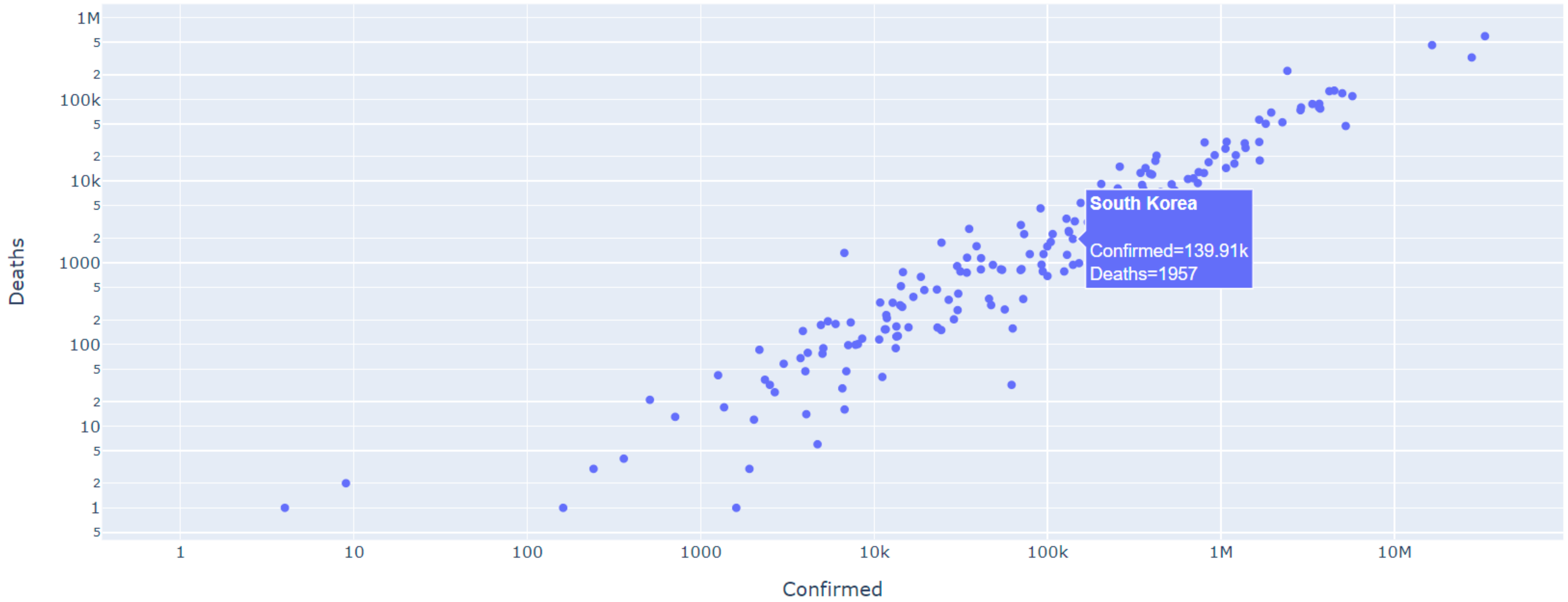
Plotting Labels

```
import plotly.express as px
fig = px.scatter(df_lastday, x='Confirmed', y='Deaths',
        text='Country', log_x=True, log_y=True)
fig.show()
```



Labelling by Mouse Hover

```
import plotly.express as px
fig = px.scatter(df_lastday, x='Confirmed', y='Deaths’,
          hover_name='Country', log_x=True, log_y=True)
fig.show()
```



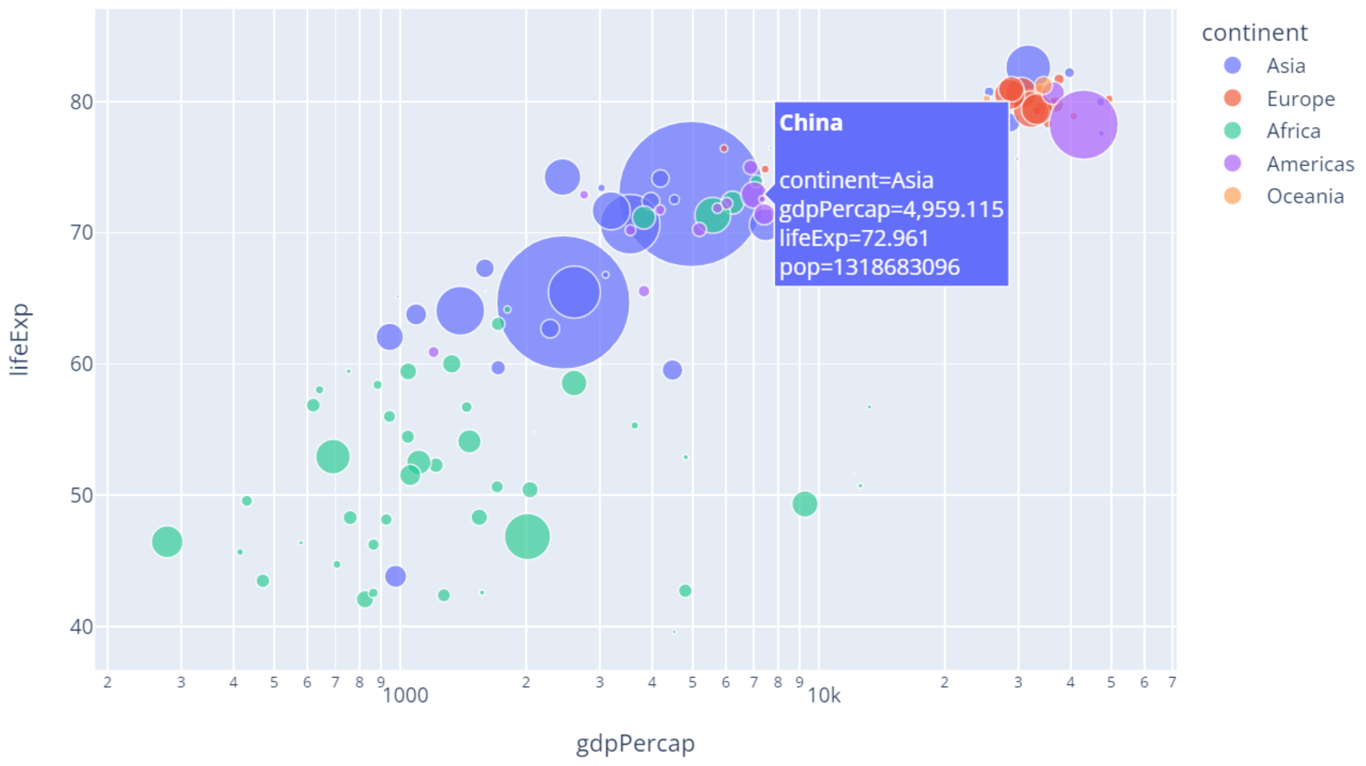
JOIN TWO DATAFRAMES

Correlation Between Confirmed, Death and GDP

- The positive correlation between Confirmed and Death
- Q: Does a country with large income shows lower death rate, compared those with small income, in terms of GDP?
- ➔ Need to combine GDP with the COVID-10 data

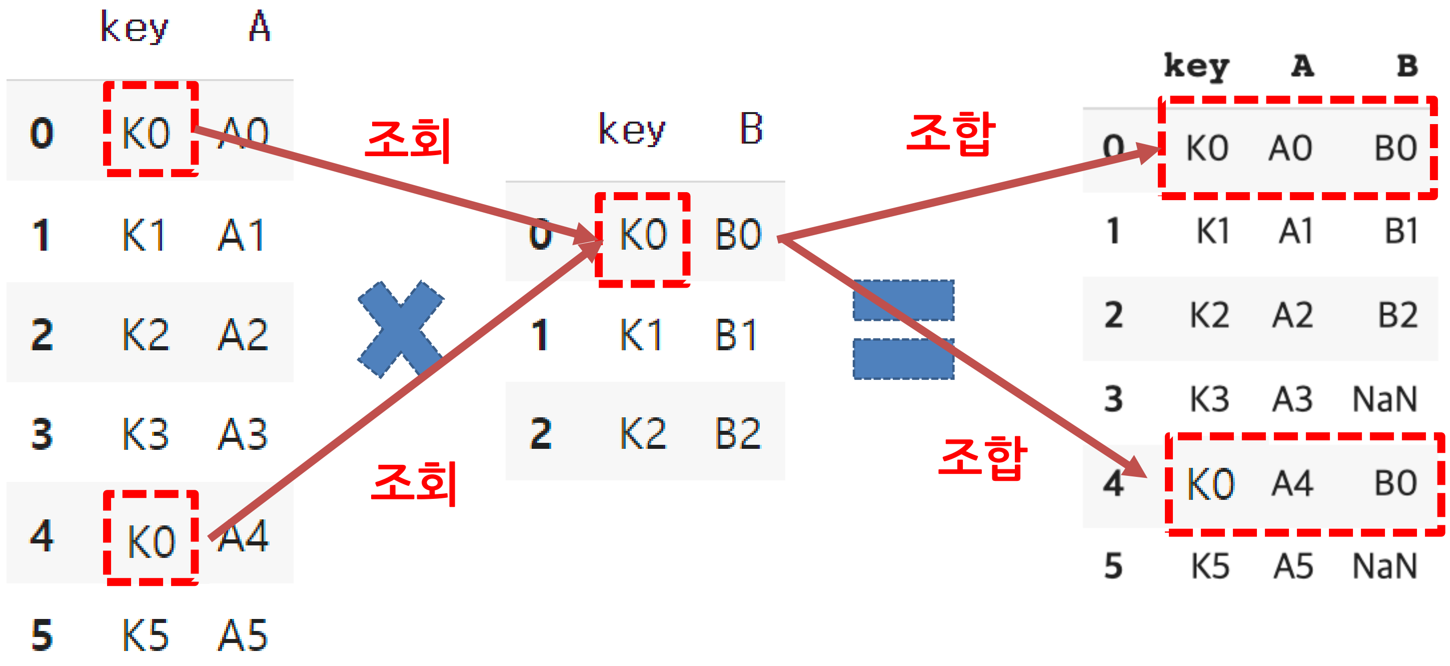
Correlations between Three Variables

- The most efficient graph to show the correlation between 3 variables in a 2-dim plane
  ➔ Bubble chart



Join

- Join DataFrames test1 and test 2
  - Want to append B values with test1 data frame where test1.key is identical to test2.key



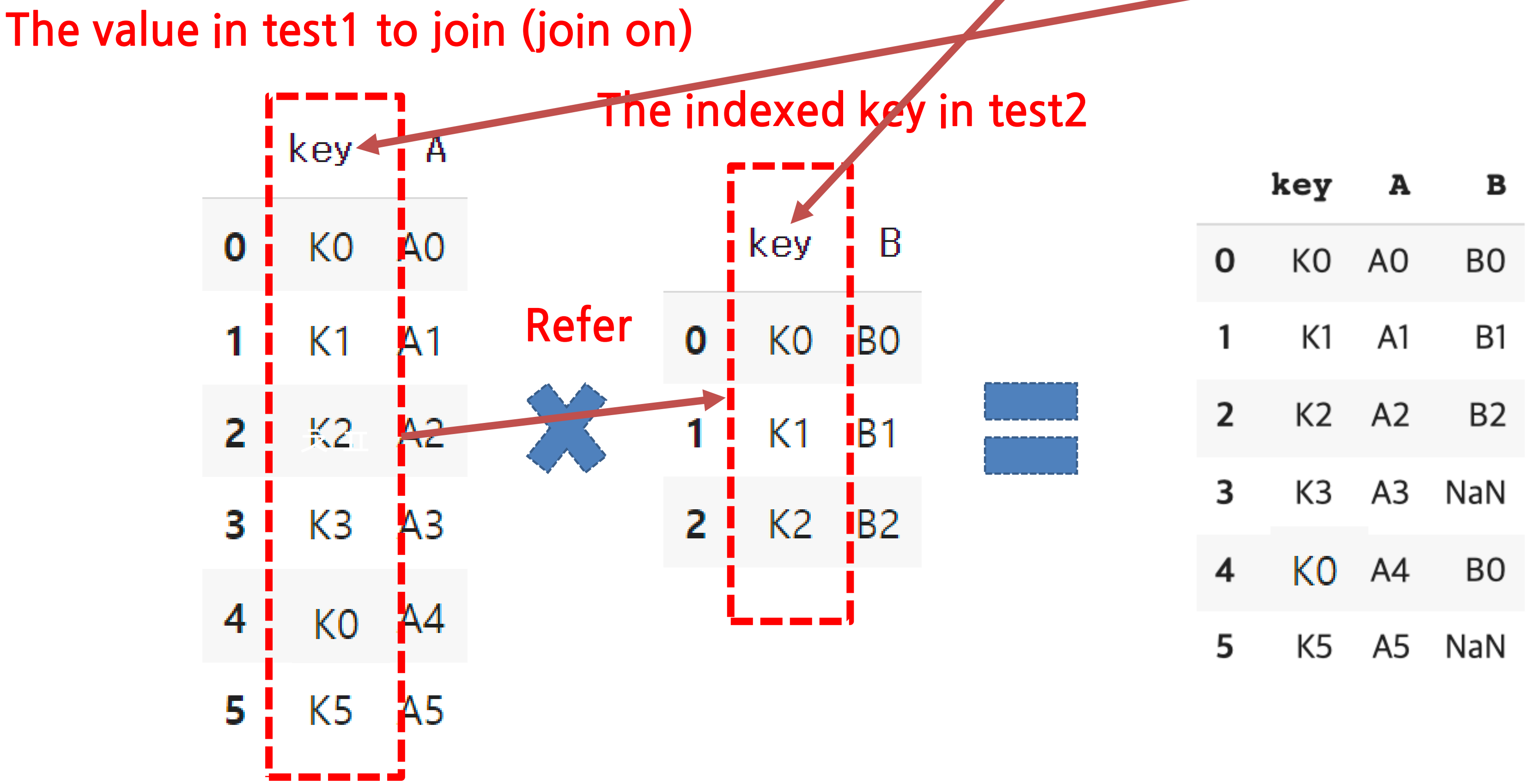
	key	A
0	K0	A0
1	K1	A1
2	K2	A2
3	K3	A3
4	K0	A4
5	K5	A5

	key	B
0	K0	B0
1	K1	B1
2	K2	B2

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	NaN
4	K0	A4	B0
5	K5	A5	NaN

Join

```
test1.join(test2.set_index('key'), on='key')
```



	key	A
0	K0	A0
1	K1	A1
2	K2	A2
3	K3	A3
4	K0	A4
5	K5	A5

	key	B
0	K0	B0
1	K1	B1
2	K2	B2

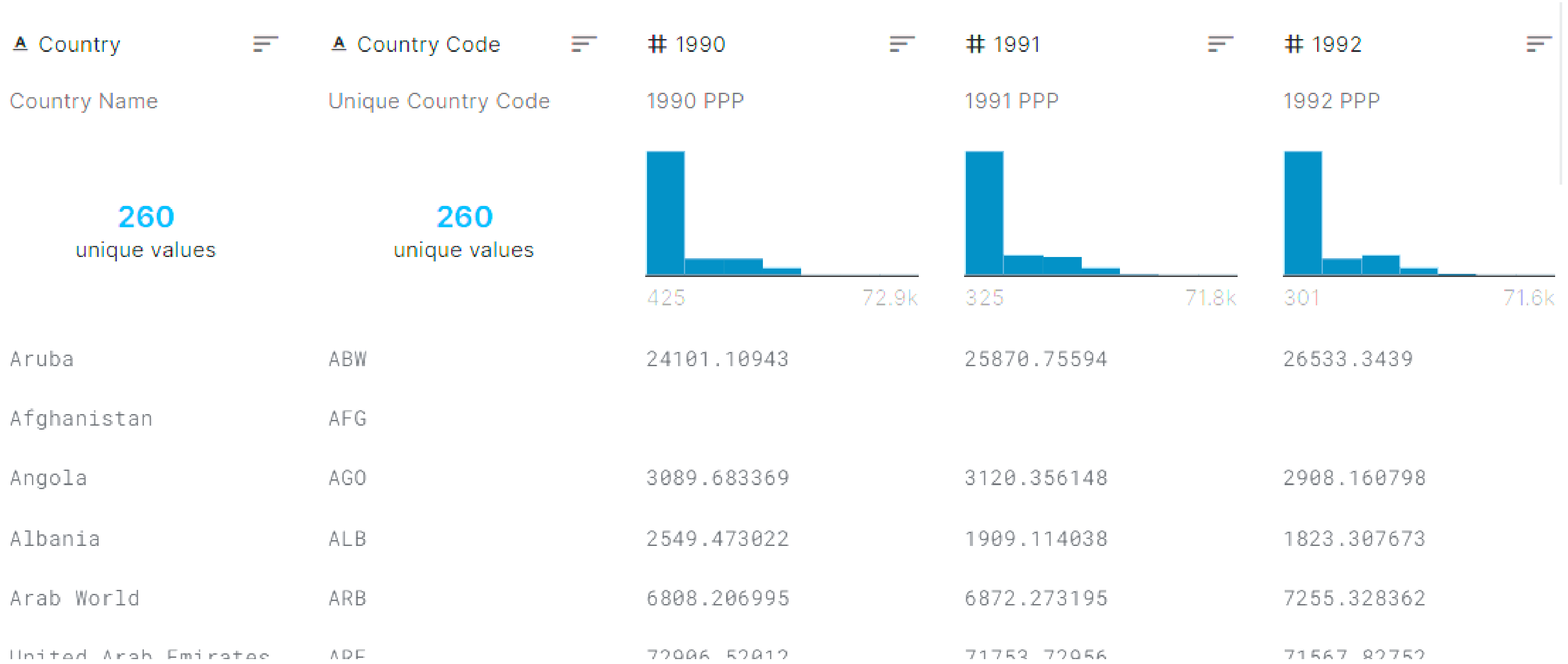
	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	NaN
4	K0	A4	B0
5	K5	A5	NaN

Practice

- Test the join operation with toy data

GDP Data

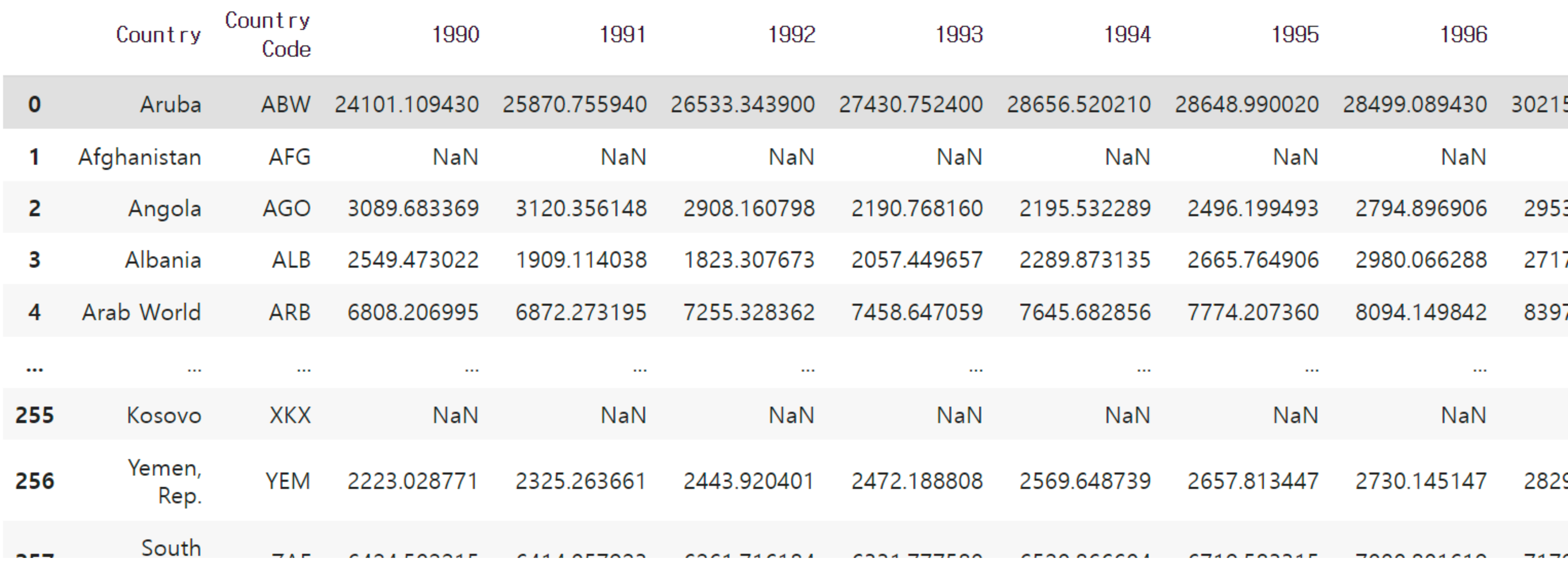
- Download from LMS
  - Size: 44KB



Country	Country Code	1990	1991	1992
Country Name	Unique Country Code	1990 PPP	1991 PPP	1992 PPP
260 unique values	260 unique values	425 – 72.9k	325 – 71.8k	301 – 71.6k
Aruba	ABW	24101.10943	25870.75594	26533.3439
Afghanistan	AFG			
Angola	AGO	3089.683369	3120.356148	2908.160798
Albania	ALB	2549.473022	1909.114038	1823.307673
Arab World	ARB	6808.206995	6872.273195	7255.328362
United Arab Emirates	ARE	72906.52012	71753.72956	71567.82752

Reading Data Files

```
gdp = pd.read_csv('drive/MyDrive/GDP.csv')
gdp
```



	Country	Country Code	1990	1991	1992	1993	1994	1995	1996	
0	Aruba	ABW	24101.109430	25870.755940	26533.343900	27430.752400	28656.520210	28648.990020	28499.089430	30215
1	Afghanistan	AFG	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	Angola	AGO	3089.683369	3120.356148	2908.160798	2190.768160	2195.532289	2496.199493	2794.896906	2953
3	Albania	ALB	2549.473022	1909.114038	1823.307673	2057.449657	2289.873135	2665.764906	2980.066288	2717
4	Arab World	ARB	6808.206995	6872.273195	7255.328362	7458.647059	7645.682856	7774.207360	8094.149842	8397
...	...	...	...	...	...	...	...	...	...	
255	Kosovo	XKX	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
256	Yemen, Rep.	YEM	2223.028771	2325.263661	2443.920401	2472.188808	2569.648739	2657.813447	2730.145147	2829
[illegible]	South	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Drop Columns

- Will use Country and 2018 columns only

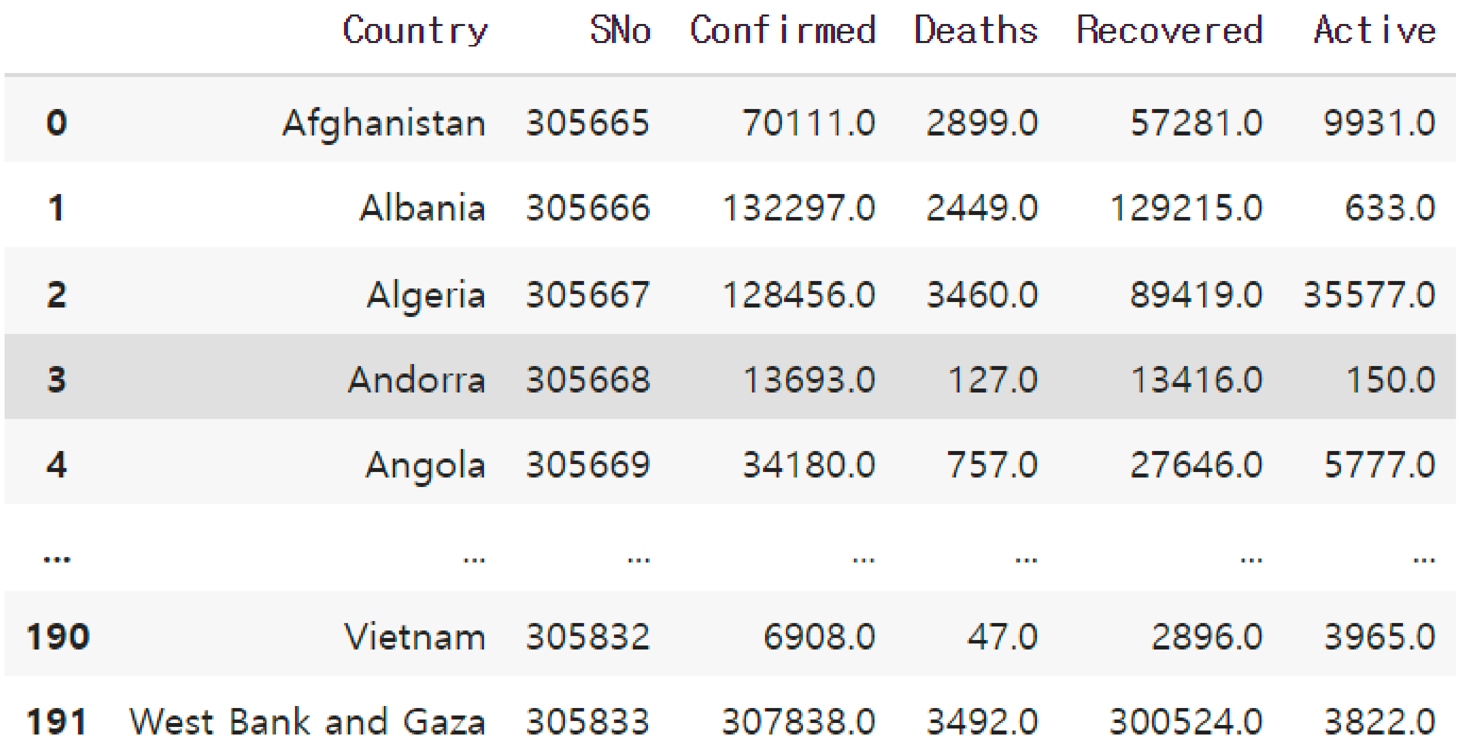
```
gdp = gdp.loc[:, ['Country', '2018']]
gdp
```

	Country	2018
0	Aruba	NaN
1	Afghanistan	1955.006208
2	Angola	6452.355165
3	Albania	13364.155400
4	Arab World	17570.137600
...	...	...
255	Kosovo	11348.363450
256	Yemen, Rep.	2575.126385

Data Preprocessing

- We already has df_lastday

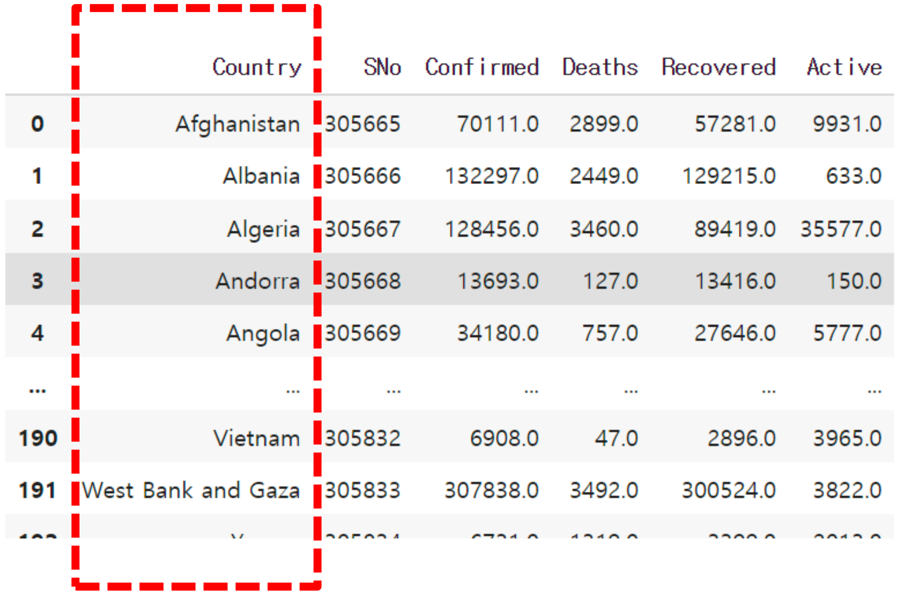
```
df_lastday = df.loc[df['Date'] == '05/29/2021’]\
         .groupby(['Country'], as_index=False).sum()
df_lastday
```



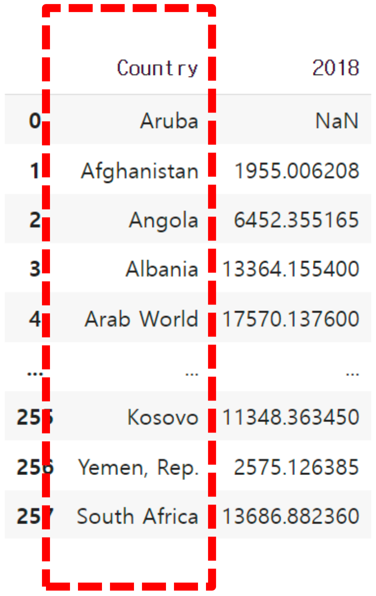
```
                Country     SNo  Confirmed  Deaths  Recovered   Active
  0         Afghanistan  305665    70111.0  2899.0    57281.0   9931.0
  1             Albania  305666   132297.0  2449.0   129215.0    633.0
  2             Algeria  305667   128456.0  3460.0    89419.0  35577.0
  3             Andorra  305668    13693.0   127.0    13416.0    150.0
  4              Angola  305669    34180.0   757.0    27646.0   5777.0
...                 ...     ...        ...     ...        ...      ...
190             Vietnam  305832     6908.0    47.0     2896.0   3965.0
191  West Bank and Gaza  305833   307838.0  3492.0   300524.0   3822.0
```

Join

- Add 2018 (=gdp) into df_lastday by joining on Country columns



	Country	SNo	Confirmed	Deaths	Recovered	Active
0	Afghanistan	305665	70111.0	2899.0	57281.0	9931.0
1	Albania	305666	132297.0	2449.0	129215.0	633.0
2	Algeria	305667	128456.0	3460.0	89419.0	35577.0
3	Andorra	305668	13693.0	127.0	13416.0	150.0
4	Angola	305669	34180.0	757.0	27646.0	5777.0
...	...	...	...	...	...	...
190	Vietnam	305832	6908.0	47.0	2896.0	3965.0
191	West Bank and Gaza	305833	307838.0	3492.0	300524.0	3822.0



	Country	2018
0	Aruba	NaN
1	Afghanistan	1955.006208
2	Angola	6452.355165
3	Albania	13364.155400
4	Arab World	17570.137600
...	...	...
255	Kosovo	11348.363450
256	Yemen, Rep.	2575.126385
257	South Africa	13686.882360

Join

```
df_lastday_with_gdp = df_lastday.join(
        gdp.set_index('Country'), on='Country')
df_lastday_with_gdp
```

	Country	SNo	Confirmed	Deaths	Recovered	Active	2018
0	Afghanistan	305665	70111.0	2899.0	57281.0	9931.0	1955.006208
1	Albania	305666	132297.0	2449.0	129215.0	633.0	13364.155400
2	Algeria	305667	128456.0	3460.0	89419.0	35577.0	15481.787620
3	Andorra	305668	13693.0	127.0	13416.0	150.0	NaN
4	Angola	305669	34180.0	757.0	27646.0	5777.0	6452.355165
...	...	...	...	...	...	...	...
190	Vietnam	305832	6908.0	47.0	2896.0	3965.0	7447.814334
191	West Bank and Gaza	305833	307838.0	3492.0	300524.0	3822.0	5157.568578
192	Yemen	305834	6731.0	1319.0	3399.0	2013.0	NaN
193	Zambia	305835	94751.0	1276.0	91594.0	1881.0	4223.906936

Bubble Chart

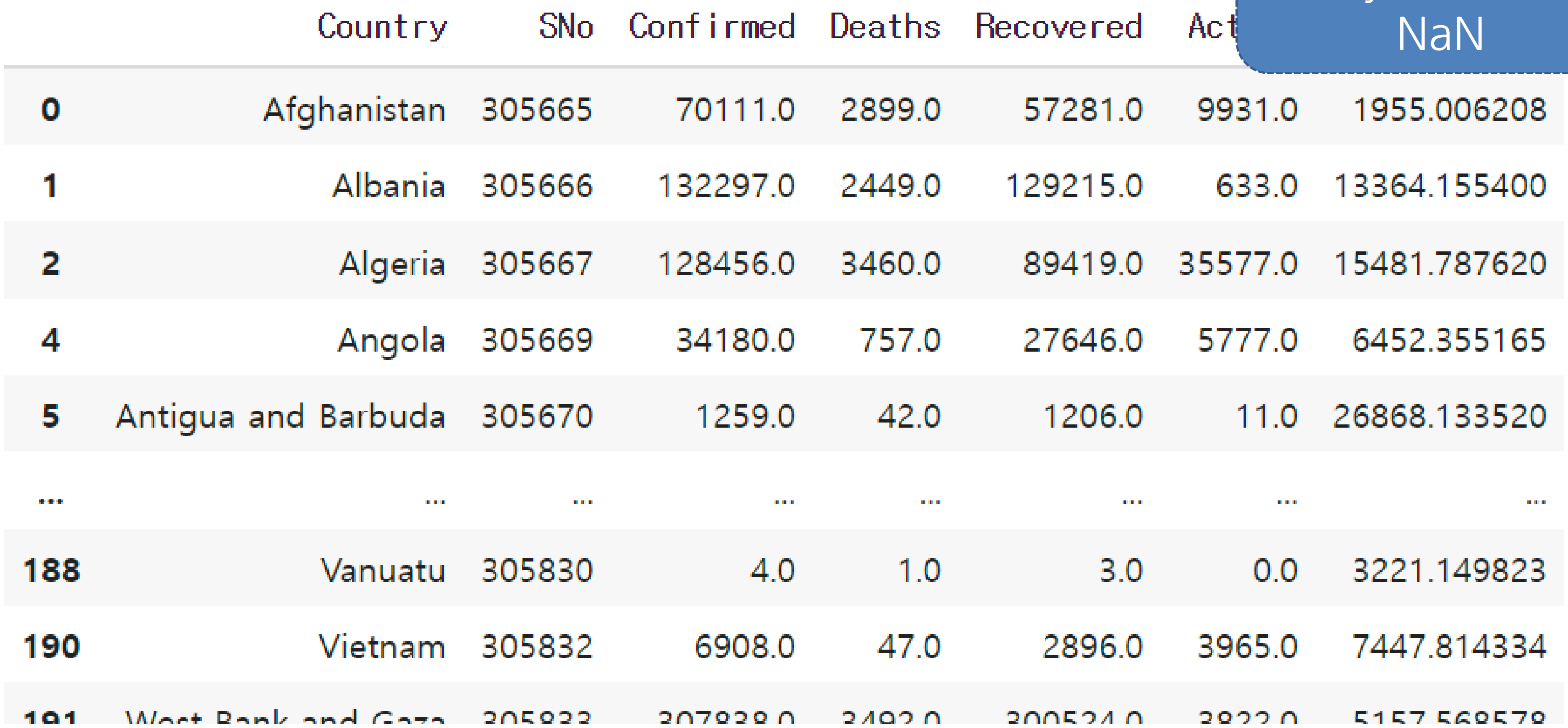
```python
import plotly.express as px

fig = px.scatter(df_lastday_with_gdp,
    x='Confirmed', y='Deaths', hover_name='Country',
    log_x=True, log_y=True, size="2018")
fig.show()
```

```
ValueError: Invalid element(s) received for the 'size'
property of scatter.marker Invalid elements include: [nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan]
```

Cleansing with NaN

```
df_lastday_with_gdp = df_lastday_with_gdp.dropna()
```

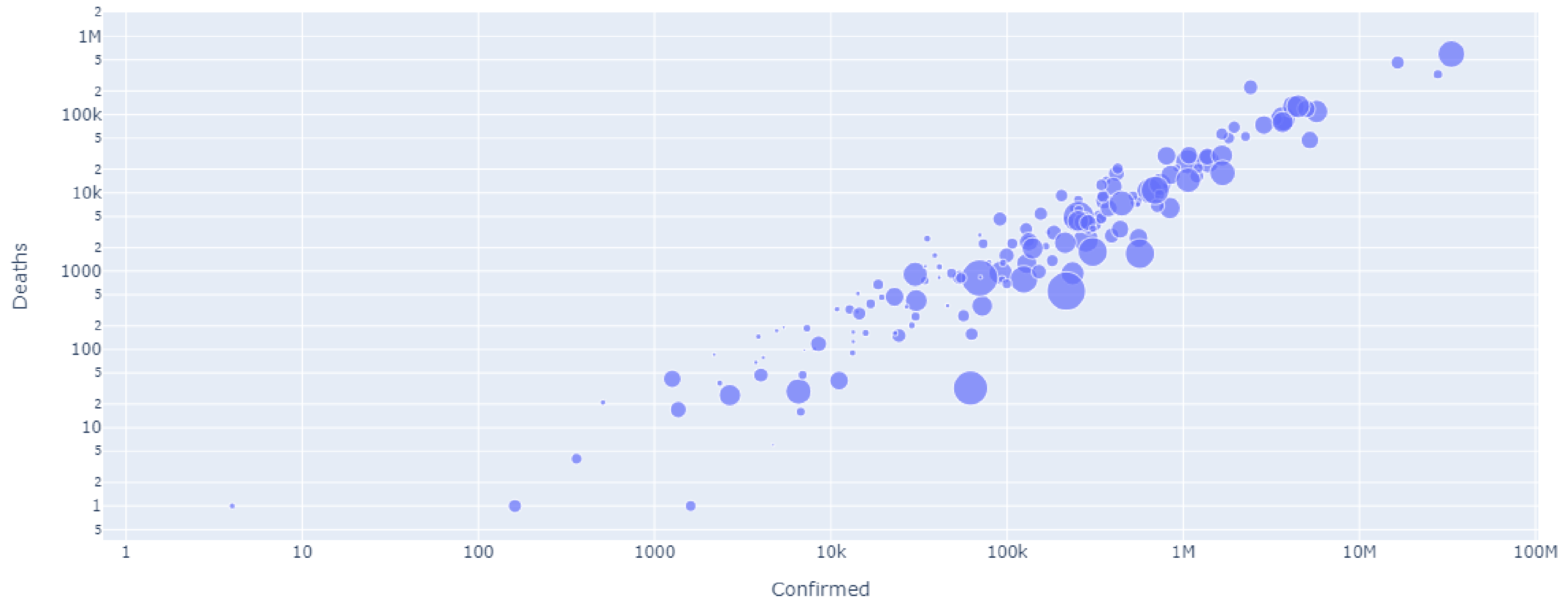
Delete a row if any field has NaN



```
                 Country    SNo  Confirmed  Deaths  Recovered     Act
  0          Afghanistan 305665    70111.0  2899.0    57281.0   9931.0   1955.006208
  1              Albania 305666   132297.0  2449.0   129215.0    633.0  13364.155400
  2              Algeria 305667   128456.0  3460.0    89419.0  35577.0  15481.787620
  4               Angola 305669    34180.0   757.0    27646.0   5777.0   6452.355165
  5  Antigua and Barbuda 305670     1259.0    42.0     1206.0     11.0  26868.133520
...                  ...    ...        ...     ...        ...      ...           ...
188              Vanuatu 305830        4.0     1.0        3.0      0.0   3221.149823
190              Vietnam 305832     6908.0    47.0     2896.0   3965.0   7447.814334
191   West Bank and Gaza 305833   307838.0  3492.0   300524.0   3822.0   5157.568578
```

Bubble Chart

```python
import plotly.express as px

fig = px.scatter(df_lastday_with_gdp,
    x='Confirmed', y='Deaths', hover_name='Country',
    log_x=True, log_y=True, size="2018")
fig.show()
```



Summary

- DataFrame selection
  - .loc
- DataFrame manipulation
  - .groupby
  - .join