MultiIndex Series & DataFrames — Cheatsheet

1️⃣ MultiIndex Series

Theory

- A MultiIndex Series has **multiple hierarchical index levels**.
- Useful to represent **higher-dimensional data** in a **1-D Series**.

Syntax

```
s = pd.Series(data, index=[level1, level2])
```

2️⃣ MultiIndex DataFrame

Theory

- A DataFrame with **multiple levels in rows and/or columns**.
- Efficient for handling **grouped**, **hierarchical**, or **panel** data.

Syntax

```
df = pd.DataFrame(data, index=[L1, L2], columns=[C1, C2])
```

3️⃣ Stack

Theory

- Converts **columns → row index level**.
- Produces a **longer** and deeper Series/DataFrame.

Syntax

```
df.stack()
```

```
## 4️⃣ Unstack
### **Theory**

* Opposite of stack. Converts **row index level → columns**.
* Produces a **wider** DataFrame.

### **Syntax**

```python
```
```

```
df.unstack()

## 5️⃣ Multiple Level Stack / Unstack

### **Theory**

* You can stack/unstack a **specific index level**.
* Useful for reorganizing complex MultiIndex structures.

### **Syntax**

```python
df.stack(level=1)
df.unstack(level=0)
```

6️⃣ Working with MultiIndex (Indexing & Slicing)

Theory

- Use `.loc`, `.xs()` (cross-section), and slicing for hierarchical selection.
- Makes filtering and accessing nested data easier.

Syntax

```
df.loc[('A', 'x')]
df.xs('A', level=0)
```

7️⃣ Melt (Unpivot)

Theory

- Converts **wide** → **long** data format.
- Good for tidying data before analysis or visualization.

Syntax

```
pd.melt(df, id_vars=['id'], value_vars=['A','B'])
```

In [1]:
```
import numpy as np
import pandas as pd
```

Series is 1D and DataFrames are 2D objects

- But why?
- And what exactly is index?

In [6]:
```
# can we have multiple index? Let's try
index_val=[('cse',2019),('cse',2020),('cse',2021),('cse',2022),('ece',2019),('ec
a=pd.Series([1,2,3,4,5,6,7,8,],index=index_val)
a
```
```

Out[6]:
```
(cse, 2019)    1
(cse, 2020)    2
(cse, 2021)    3
(cse, 2022)    4
(ece, 2019)    5
(ece, 2020)    6
(ece, 2021)    7
(ece, 2022)    8
dtype: int64
```

In [7]:
```
a[('cse',2020)]
```

Out[7]:
```
np.int64(2)
```

In [9]:
```
# The problem?
a['cse']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get
_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.Py
ObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.Py
ObjectHashTable.get_item()

KeyError: 'cse'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[9], line 2
      1 # The problem?
----> 2 a['cse']

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:1121, in Series.__getite
m__(self, key)
   1118     return self._values[key]
   1120 elif key_is_scalar:
-> 1121     return self._get_value(key)
   1123 # Convert generator to list before going through hashable part
   1124 # (We will iterate through the generator there to check for slices)
   1125 if is_iterator(key):

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:1237, in Series._get_val
ue(self, label, takeable)
   1234     return self._values[label]
   1236 # Similar to Index.get_value, but we do not fall back to positional
-> 1237 loc = self.index.get_loc(label)
   1239 if is_integer(loc):
   1240     return self._values[loc]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get
_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'cse'
```

In [10]:
```
# The solution -> multiindex series(also known as Hierarchical Indexing)
# multiple index levels within a single index
```

In [12]:
```
# how to create multiindex object
# 1. pd.MultiIndex.from_tuples()
index_val=[('cse',2019),('cse',2020),('cse',2021),('cse',2022),('ece',2019),('ec
multiindex=pd.MultiIndex.from_tuples(index_val)
multiindex
```

Out[12]:
```
MultiIndex([('cse', 2019),
            ('cse', 2020),
            ('cse', 2021),
            ('cse', 2022),
            ('ece', 2019),
            ('ece', 2020),
            ('ece', 2021),
            ('ece', 2022)],
           )
```

In [16]:
```
multiindex.levels[0]
```

Out[16]:
```
Index(['cse', 'ece'], dtype='object')
```

In [17]:
```
multiindex.levels[1]
```

Out[17]:
```
Index([2019, 2020, 2021, 2022], dtype='int64')
```

In [18]:
```
# 2. pd.MultiIndex.from_product()
pd.MultiIndex.from_product([['cse','ece'],[2019,2020,2021,2022]])
```

Out[18]:
```
MultiIndex([('cse', 2019),
            ('cse', 2020),
            ('cse', 2021),
            ('cse', 2022),
            ('ece', 2019),
            ('ece', 2020),
            ('ece', 2021),
            ('ece', 2022)],
           )
```

In [19]:
```
# level inside multiindex object
```

In [20]:
```
# creating a series with multiindex object
s=pd.Series([1,2,3,4,5,6,7,8],index=multiindex)
s
```

Out[20]:
```
cse  2019    1
     2020    2
     2021    3
     2022    4
ece  2019    5
     2020    6
     2021    7
     2022    8
dtype: int64
```

In [22]:
```
# how to fetch items from such a series
s['cse']
```

Out[22]:
```
2019    1
2020    2
2021    3
2022    4
dtype: int64
```

In [23]:
```
# unstack
temp=s.unstack()
temp
```

Out[23]:

	2019	2020	2021	2022
cse	1	2	3	4
ece	5	6	7	8

In [24]:
```
# stack
temp.stack()
```

Out[24]:
```
cse  2019    1
     2020    2
     2021    3
     2022    4
ece  2019    5
     2020    6
     2021    7
     2022    8
dtype: int64
```

In [25]:
```
## multindex dataframe
```

In [26]:
```
branch_df1=pd.DataFrame(
    [
        [1,2],
        [3,4],
        [5,6],
        [7,8],
        [9,10],
        [11,12],
        [13,14],
        [15,16],
    ],
    index=multiindex,
    columns=['avg_package','students']
)
branch_df1
```

Out[26]:

		avg_package	students
cse	**2019**	1	2
	2020	3	4
	2021	5	6
	2022	7	8
ece	**2019**	9	10
	2020	11	12
	2021	13	14
	2022	15	16

In [27]:
```
branch_df1['students']
```

Out[27]:
```
cse  2019     2
     2020     4
     2021     6
     2022     8
ece  2019    10
     2020    12
     2021    14
     2022    16
Name: students, dtype: int64
```

In [28]:
```
# Are columns really different from index?
```

In [29]:
```
# multiindex df from columns perspective
branch_df2=pd.DataFrame(
    [
        [1,2,0,0],
        [3,4,0,0],
        [5,6,0,0],
        [7,8,0,0],
    ],
    index=[2019,2020,2021,2022],
    columns=pd.MultiIndex.from_product([['delhi','mumbai'],['avg_package','stude
)
branch_df2
```

Out[29]:

	delhi		mumbai	
	avg_package	students	avg_package	students
2019	1	2	0	0
2020	3	4	0	0
2021	5	6	0	0
2022	7	8	0	0

In [30]:
```
branch_df2.loc[2019]
```

Out[30]:
```
delhi   avg_package    1
        students       2
mumbai  avg_package    0
        students       0
Name: 2019, dtype: int64
```

In [31]:
```
# Multiindex df in terms of both cols and index
branch_df3=pd.DataFrame(
    [
        [1,2,0,0],
        [3,4,0,0],
        [5,6,0,0],
        [7,8,0,0],
        [9,10,0,0],
        [11,12,0,0],
        [13,14,0,0],
        [15,16,0,0],
    ],
    index=multiindex,
    columns=pd.MultiIndex.from_product([['delhi','mumbai'],['avg_packages','stud
)
branch_df3
```

Out[31]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	2019	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0
ece	2019	9	10	0	0
	2020	11	12	0	0
	2021	13	14	0	0
	2022	15	16	0	0

In [32]:
```
## stacking and Unstacking
branch_df1
```

Out[32]:

		avg_package	students
cse	**2019**	1	2
	2020	3	4
	2021	5	6
	2022	7	8
ece	**2019**	9	10
	2020	11	12
	2021	13	14
	2022	15	16

In [33]:
```
branch_df1.stack()
```

Out[33]:
```
cse  2019  avg_package     1
           students        2
      2020  avg_package     3
           students        4
      2021  avg_package     5
           students        6
      2022  avg_package     7
           students        8
ece  2019  avg_package     9
           students       10
      2020  avg_package    11
           students       12
      2021  avg_package    13
           students       14
      2022  avg_package    15
           students       16
dtype: int64
```

In [36]:
```
branch_df1.stack().unstack()
```

Out[36]:

		avg_package	students
cse	**2019**	1	2
	2020	3	4
	2021	5	6
	2022	7	8
ece	**2019**	9	10
	2020	11	12
	2021	13	14
	2022	15	16

In [39]:
```
branch_df1.unstack()
```

Out[39]:

	avg_package				students			
	2019	2020	2021	2022	2019	2020	2021	2022
cse	1	3	5	7	2	4	6	8
ece	9	11	13	15	10	12	14	16

In [40]:
```
branch_df3
```

Out[40]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	2019	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0
ece	2019	9	10	0	0
	2020	11	12	0	0
	2021	13	14	0	0
	2022	15	16	0	0

In [41]:
```
branch_df3.stack()
```

```
C:\Users\ARIF RAZA\AppData\Local\Temp\ipykernel_20096\4148153360.py:1: FutureWarn
ing: The previous implementation of stack is deprecated and will be removed in a
future version of pandas. See the What's New notes for pandas 2.1.0 for details.
Specify future_stack=True to adopt the new implementation and silence this warnin
g.
  branch_df3.stack()
```

Out[41]:

			delhi	mumbai
cse	2019	avg_packages	1	0
		students	2	0
	2020	avg_packages	3	0
		students	4	0
	2021	avg_packages	5	0
		students	6	0
	2022	avg_packages	7	0
		students	8	0
ece	2019	avg_packages	9	0
		students	10	0
	2020	avg_packages	11	0
		students	12	0
	2021	avg_packages	13	0
		students	14	0
	2022	avg_packages	15	0
		students	16	0

In [42]:
```
branch_df3.unstack()
```

Out[42]:

	delhi												
	avg_packages				students				avg_packages				
	2019	2020	2021	2022	2019	2020	2021	2022	2019	2020	2021	2022	201
cse	1	3	5	7	2	4	6	8	0	0	0	0	
ece	9	11	13	15	10	12	14	16	0	0	0	0	

In [43]:
```
branch_df3.stack().stack()
```

```
C:\Users\ARIF RAZA\AppData\Local\Temp\ipykernel_20096\4023844418.py:1: FutureWarn
ing: The previous implementation of stack is deprecated and will be removed in a
future version of pandas. See the What's New notes for pandas 2.1.0 for details.
Specify future_stack=True to adopt the new implementation and silence this warnin
g.
  branch_df3.stack().stack()
```

Out[43]:
```
cse  2019  avg_packages  delhi      1
                         mumbai     0
           students      delhi      2
                         mumbai     0
      2020  avg_packages  delhi      3
                         mumbai     0
           students      delhi      4
                         mumbai     0
      2021  avg_packages  delhi      5
                         mumbai     0
           students      delhi      6
                         mumbai     0
      2022  avg_packages  delhi      7
                         mumbai     0
           students      delhi      8
                         mumbai     0
ece  2019  avg_packages  delhi      9
                         mumbai     0
           students      delhi     10
                         mumbai     0
      2020  avg_packages  delhi     11
                         mumbai     0
           students      delhi     12
                         mumbai     0
      2021  avg_packages  delhi     13
                         mumbai     0
           students      delhi     14
                         mumbai     0
      2022  avg_packages  delhi     15
                         mumbai     0
           students      delhi     16
                         mumbai     0
dtype: int64
```

In [44]:
```
## working with multiindex dataFrames
```

In [45]:
```
branch_df3
```

Out[45]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	**2019**	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0
ece	**2019**	9	10	0	0
	2020	11	12	0	0
	2021	13	14	0	0
	2022	15	16	0	0

In [46]:
```
# head and tail
branch_df3.head()
```

Out[46]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	2019	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0
ece	2019	9	10	0	0

In [47]:
```
# shape
branch_df3.shape
```

Out[47]:
```
(8, 4)
```

In [48]:
```
# info
branch_df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 8 entries, ('cse', np.int64(2019)) to ('ece', np.int64(2022))
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   (delhi, avg_packages)   8 non-null      int64
 1   (delhi, students)       8 non-null      int64
 2   (mumbai, avg_packages)  8 non-null      int64
 3   (mumbai, students)      8 non-null      int64
dtypes: int64(4)
memory usage: 632.0+ bytes
```

In [49]:
```
# duplicated->isnull
branch_df3.duplicated()
```

Out[49]:
```
cse  2019    False
     2020    False
     2021    False
     2022    False
ece  2019    False
     2020    False
     2021    False
     2022    False
dtype: bool
```

In [50]:
```
branch_df3.isnull()
```

Out[50]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	2019	False	False	False	False
	2020	False	False	False	False
	2021	False	False	False	False
	2022	False	False	False	False
ece	2019	False	False	False	False
	2020	False	False	False	False
	2021	False	False	False	False
	2022	False	False	False	False

In [51]:
```
# exracting rows single
branch_df3.loc[('cse',2019)]
```

Out[51]:
```
delhi   avg_packages    1
        students        2
mumbai  avg_packages    0
        students        0
Name: (cse, 2019), dtype: int64
```

In [53]:
```
  # multiple
branch_df3.loc[('cse',2019):('ece',2020):2]
```

Out[53]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	2019	1	2	0	0
	2021	5	6	0	0
ece	2019	9	10	0	0

In [54]:
```
# using iloc
branch_df3.iloc[0:5:2]
```

Out[54]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
cse	2019	1	2	0	0
	2021	5	6	0	0
ece	2019	9	10	0	0

In [55]:
```
# extracting columns
branch_df3['delhi']['students']
```

Out[55]:
```
cse  2019     2
     2020     4
     2021     6
     2022     8
ece  2019    10
     2020    12
     2021    14
     2022    16
Name: students, dtype: int64
```

In [56]:
```
branch_df3.iloc[:,1:3]
```

Out[56]:

		delhi	mumbai
		students	avg_packages
cse	2019	2	0
	2020	4	0
	2021	6	0
	2022	8	0
ece	2019	10	0
	2020	12	0
	2021	14	0
	2022	16	0

In [57]:
```
# extracting both
branch_df3.iloc[[0,4],[1,2]]
```

Out[57]:

		delhi	mumbai
		students	avg_packages
cse	2019	2	0
ece	2019	10	0

In [59]:
```
# sort index
# both->descending->diff order
# based on one level
branch_df3.sort_index(ascending=False)
```

Out[59]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
ece	2022	15	16	0	0
	2021	13	14	0	0
	2020	11	12	0	0
	2019	9	10	0	0
cse	2022	7	8	0	0
	2021	5	6	0	0
	2020	3	4	0	0
	2019	1	2	0	0

In [60]:
```
branch_df3.sort_index(ascending=[False,True])
```

Out[60]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
ece	2019	9	10	0	0
	2020	11	12	0	0
	2021	13	14	0	0
	2022	15	16	0	0
cse	2019	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0

In [61]:
```
branch_df3.sort_index(level=0,ascending=[False])
```

Out[61]:

		delhi		mumbai	
		avg_packages	students	avg_packages	students
ece	2019	9	10	0	0
	2020	11	12	0	0
	2021	13	14	0	0
	2022	15	16	0	0
cse	2019	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0

In [62]:
```
# multiindex dataframe(col)-> transpose
branch_df3.transpose()
```

Out[62]:

		cse				ece			
		2019	2020	2021	2022	2019	2020	2021	2022
delhi	avg_packages	1	3	5	7	9	11	13	15
	students	2	4	6	8	10	12	14	16
mumbai	avg_packages	0	0	0	0	0	0	0	0
	students	0	0	0	0	0	0	0	0

In [63]:
```
# swaplevel
branch_df3.swaplevel(axis=1)
```

Out[63]:

		avg_packages	students	avg_packages	students
		delhi	delhi	mumbai	mumbai
cse	2019	1	2	0	0
	2020	3	4	0	0
	2021	5	6	0	0
	2022	7	8	0	0
ece	2019	9	10	0	0
	2020	11	12	0	0
	2021	13	14	0	0
	2022	15	16	0	0

Long Vs Wide Data

Name	Height	Weight
John	160	67
Christopher	182	78

Name	Attribute	Value
John	Height	160
John	Weight	67
Christopher	Height	182
Christopher	Weight	78

Wide format is where we have a single row for every data point with multiple columns to hold the values of various attributes.

Long format is where, for each data point we have as many rows as the number of attributes and each row contains the value of a particular attribute for a given data point.

In [64]:
```
# melt -> simple exapmle branch
# wide to long
pd.DataFrame({'cse':[120]})
```

Out[64]:

	cse
0	120

In [65]:
```
pd.DataFrame({'cse':[120]}).melt()
```

Out[65]:

	variable	value
0	cse	120

In [66]:
```
# melt -> branch with year
pd.DataFrame({'cse':[120],'ece':[100],'mech':[50]})
```

Out[66]:

	cse	ece	mech
0	120	100	50

In [67]:
```
pd.DataFrame({'cse':[120],'ece':[100],'mech':[50]}).melt()
```

Out[67]:

	variable	value
0	cse	120
1	ece	100
2	mech	50

In [68]:
```
pd.DataFrame({'cse':[120],'ece':[100],'mech':[50]}).melt(var_name='branch',value
```

Out[68]:

	branch	num_students
0	cse	120
1	ece	100
2	mech	50

In [69]:
```
pd.DataFrame(
    {
        'branch':['cse','ece','mech'],
        '2020':[100,150,60],
        '2021':[120,130,80],
        '2022':[150,140,70]
    }

)
```

Out[69]:

	branch	2020	2021	2022
0	cse	100	120	150
1	ece	150	130	140
2	mech	60	80	70

In [70]:
```
pd.DataFrame(
    {
        'branch':['cse','ece','mech'],
        '2020':[100,150,60],
        '2021':[120,130,80],
        '2022':[150,140,70]
    }

).melt()
```

Out[70]:

	variable	value
0	branch	cse
1	branch	ece
2	branch	mech
3	2020	100
4	2020	150
5	2020	60
6	2021	120
7	2021	130
8	2021	80
9	2022	150
10	2022	140
11	2022	70

In [71]:
```
pd.DataFrame(
    {
        'branch':['cse','ece','mech'],
        '2020':[100,150,60],
        '2021':[120,130,80],
        '2022':[150,140,70]
    }

).melt(id_vars=['branch'],var_name='year',value_name='students')
```

Out[71]:

	branch	year	students
0	cse	2020	100
1	ece	2020	150
2	mech	2020	60
3	cse	2021	120
4	ece	2021	130
5	mech	2021	80
6	cse	2022	150
7	ece	2022	140
8	mech	2022	70

In [82]:
```
#melt->real world examples
death=pd.read_csv('time_series_covid19_deaths_global.csv')
confirm=pd.read_csv('time_series_covid19_confirmed_global.csv')
```

In [83]:
```
death.head()
```

Out[83]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	

5 rows × 1081 columns

In [84]:
```
confirm.head()
```

Out[84]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	

5 rows × 1081 columns

In [85]:
```
death=death.melt(id_vars=['Province/State','Country/Region','Lat','long'],var_na
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[85], line 1
----> 1 death=death.melt(id_vars=['Province/State','Country/Region','Lat','lon
g'],var_name='date',value_name='num_deaths')

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:9942, in DataFrame.melt(s
elf, id_vars, value_vars, var_name, value_name, col_level, ignore_index)
   9932 @Appender(_shared_docs["melt"] % {"caller": "df.melt(", "other": "melt"})
   9933 def melt(
   9934     self,
   (...)
   9940     ignore_index: bool = True,
   9941 ) -> DataFrame:
-> 9942     return melt(
   9943         self,
   9944         id_vars=id_vars,
   9945         value_vars=value_vars,
   9946         var_name=var_name,
   9947         value_name=value_name,
   9948         col_level=col_level,
   9949         ignore_index=ignore_index,
   9950     ).__finalize__(self, method="melt")

File ~\anaconda3\Lib\site-packages\pandas\core\reshape\melt.py:74, in melt(frame,
id_vars, value_vars, var_name, value_name, col_level, ignore_index)
     70 if missing.any():
     71     missing_labels = [
     72         lab for lab, not_found in zip(labels, missing) if not_found
     73     ]
---> 74     raise KeyError(
     75         "The following id_vars or value_vars are not present in "
     76         f"the DataFrame: {missing_labels}"
     77     )
     78 if value_vars_was_not_none:
     79     frame = frame.iloc[:, algos.unique(idx)]

KeyError: "The following id_vars or value_vars are not present in the DataFrame:
['long']"
```

In [86]:
```
death = death.melt(id_vars=['Province/State','Country/Region','Lat','Long'],var_
confirm = confirm.melt(id_vars=['Province/State','Country/Region','Lat','Long'],
```

In [87]:
```
death.head()
```

Out[87]:

	Province/State	Country/Region	Lat	Long	date	num_deaths
0	NaN	Afghanistan	33.93911	67.709953	1/22/20	0
1	NaN	Albania	41.15330	20.168300	1/22/20	0
2	NaN	Algeria	28.03390	1.659600	1/22/20	0
3	NaN	Andorra	42.50630	1.521800	1/22/20	0
4	NaN	Angola	-11.20270	17.873900	1/22/20	0

In [88]:
```
confirm.head()
```

Out[88]:

	Province/State	Country/Region	Lat	Long	date	num_cases
0	NaN	Afghanistan	33.93911	67.709953	1/22/20	0
1	NaN	Albania	41.15330	20.168300	1/22/20	0
2	NaN	Algeria	28.03390	1.659600	1/22/20	0
3	NaN	Andorra	42.50630	1.521800	1/22/20	0
4	NaN	Angola	-11.20270	17.873900	1/22/20	0

In [90]:
```
confirm.merge(death,on=['Province/State','Country/Region','Lat','Long','date'])
```

Out[90]:

	Province/State	Country/Region	Lat	Long	date	num_cases	n
0	NaN	Afghanistan	33.939110	67.709953	1/22/20	0	
1	NaN	Albania	41.153300	20.168300	1/22/20	0	
2	NaN	Algeria	28.033900	1.659600	1/22/20	0	
3	NaN	Andorra	42.506300	1.521800	1/22/20	0	
4	NaN	Angola	-11.202700	17.873900	1/22/20	0	
...	...	...	...	...	...	...	
311248	NaN	West Bank and Gaza	31.952200	35.233200	1/2/23	703228	
311249	NaN	Winter Olympics 2022	39.904200	116.407400	1/2/23	535	
311250	NaN	Yemen	15.552727	48.516388	1/2/23	11945	
311251	NaN	Zambia	-13.133897	27.849332	1/2/23	334661	
311252	NaN	Zimbabwe	-19.015438	29.154857	1/2/23	259981	

311253 rows × 7 columns

In [91]:
```
confirm.merge(death,on=['Province/State','Country/Region','Lat','Long','date'])[
```

Out[91]:

	Country/Region	date	num_cases	num_deaths
0	Afghanistan	1/22/20	0	0
1	Albania	1/22/20	0	0
2	Algeria	1/22/20	0	0
3	Andorra	1/22/20	0	0
4	Angola	1/22/20	0	0
...	...	...	...	...
311248	West Bank and Gaza	1/2/23	703228	5708
311249	Winter Olympics 2022	1/2/23	535	0
311250	Yemen	1/2/23	11945	2159
311251	Zambia	1/2/23	334661	4024
311252	Zimbabwe	1/2/23	259981	5637

311253 rows × 4 columns

In [ ]: