Data Wrangling: Join, Combine,

In [1]:
```
import numpy as np
import pandas as pd
pd.options.display.max_rows = 20
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)
```

Hierarchical Indexing

In [2]:
```
data = pd.Series(np.random.randn(9),
                 index=[['a', 'a', 'a', 'b', 'b', 'c', 'c', 'd', 'd'],
                        [1, 2, 3, 1, 3, 1, 2, 2, 3]])
data
```

Out[2]:
```
a  1   -0.204708
   2    0.478943
   3   -0.519439
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
d  2    0.281746
   3    0.769023
dtype: float64
```

In [3]:
```
data.index
```

Out[3]:
```
MultiIndex([('a', 1),
            ('a', 2),
            ('a', 3),
            ('b', 1),
            ('b', 3),
            ('c', 1),
            ('c', 2),
            ('d', 2),
            ('d', 3)],
           )
```

In [4]:
```
data['b']
```

Out[4]:
```
1   -0.555730
3    1.965781
dtype: float64
```

In [5]:
```
data['b':'c']
```

Out[5]:
```
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
dtype: float64
```

In [6]:
```
data.loc[['b', 'd']]
```

Out[6]:
```
b  1   -0.555730
   3    1.965781
d  2    0.281746
   3    0.769023
dtype: float64
```

In [7]:
```
data.loc[:, 2]
```

Out[7]:
```
a    0.478943
c    0.092908
d    0.281746
dtype: float64
```

In [8]:
```
data.unstack()
```

Out[8]:

	1	2	3
a	-0.204708	0.478943	-0.519439
b	-0.555730	NaN	1.965781
c	1.393406	0.092908	NaN
d	NaN	0.281746	0.769023

In [9]:
```
data.unstack().stack()
```

Out[9]:
```
a  1   -0.204708
   2    0.478943
   3   -0.519439
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
d  2    0.281746
   3    0.769023
dtype: float64
```

In [10]:
```
frame = pd.DataFrame(np.arange(12).reshape((4, 3)),
                     index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],
                     columns=[['Ohio', 'Ohio', 'Colorado'],
                              ['Green', 'Red', 'Green']])
frame
```

Out[10]:

		Ohio		Colorado
		Green	Red	Green
a	**1**	0	1	2
	2	3	4	5
b	**1**	6	7	8
	2	9	10	11

In [12]:
```
frame.index.names = ['key1', 'key2']
frame.columns.names = ['state', 'color']
frame
```

Out[12]:

	state	Ohio	Ohio	Colorado
	color	Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

In [13]:
```
frame['Ohio']
```

Out[13]:

	color	Green	Red
key1	key2		
a	1	0	1
	2	3	4
b	1	6	7
	2	9	10

In [15]:
```
pd.MultiIndex.from_arrays([['Ohio', 'Ohio', 'Colorado'], ['Green', 'Red', 'Green']]
                          names=['state', 'color'])
```

Out[15]:
```
MultiIndex([(    'Ohio', 'Green'),
            (    'Ohio',   'Red'),
            ('Colorado', 'Green')],
           names=['state', 'color'])
```

Reordering and Sorting Levels

In [16]:
```
frame.swaplevel('key1', 'key2')
```

Out[16]:

	state	Ohio	Ohio	Colorado
	color	Green	Red	Green
key2	key1			
1	a	0	1	2
2	a	3	4	5
1	b	6	7	8
2	b	9	10	11

In [17]:
```
frame.sort_index(level=1)
```

Out[17]:

	state	Ohio	Colorado
	color	Green	Red	Green
key1	key2			
a	1	0	1	2
b	1	6	7	8
a	2	3	4	5
b	2	9	10	11

In [19]:
```
frame.sort_index(level=0)
```

Out[19]:

	state	Ohio	Colorado
	color	Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

In [20]:
```
frame.swaplevel(0, 1).sort_index(level=0)
```

Out[20]:

	state	Ohio	Colorado
	color	Green	Red	Green
key2	key1			
1	a	0	1	2
	b	6	7	8
2	a	3	4	5
	b	9	10	11

Summary Statistics by Level

In [21]:
```
frame.sum(level='key2')
```

```
C:\Users\Usuário\AppData\Local\Temp\ipykernel_21488\2004046222.py:1: FutureWarnin
g: Using the level keyword in DataFrame and Series aggregations is deprecated and
will be removed in a future version. Use groupby instead. df.sum(level=1) should u
se df.groupby(level=1).sum().
  frame.sum(level='key2')
```

Out[21]:

state	Ohio		Colorado
color	Green	Red	Green
key2			
1	6	8	10
2	12	14	16

In [22]:
```
frame.groupby(level='key2').sum()
```

Out[22]:

state	Ohio		Colorado
color	Green	Red	Green
key2			
1	6	8	10
2	12	14	16

In [23]:
```
frame.groupby(level='key1').sum()
```

Out[23]:

state	Ohio		Colorado
color	Green	Red	Green
key1			
a	3	5	7
b	15	17	19

In [24]:
```
frame.sum(level='color', axis=1)
```

```
C:\Users\Usuário\AppData\Local\Temp\ipykernel_21488\4133796543.py:1: FutureWarnin
g: Using the level keyword in DataFrame and Series aggregations is deprecated and
will be removed in a future version. Use groupby instead. df.sum(level=1) should u
se df.groupby(level=1).sum().
  frame.sum(level='color', axis=1)
```

Out[24]:

	color	Green	Red
key1	key2		
a	1	2	1
	2	8	4
b	1	14	7
	2	20	10

In [27]:
```
frame.groupby(level='color', axis=1).sum()
```

Out[27]:

key1	key2	color	Green	Red
a	1	2	1	
	2	8	4	
b	1	14	7	
	2	20	10	

Indexing with a DataFrame's columns

In [28]:
```
frame = pd.DataFrame({'a': range(7), 'b': range(7, 0, -1),
                      'c': ['one', 'one', 'one', 'two', 'two',
                            'two', 'two'],
                      'd': [0, 1, 2, 0, 1, 2, 3]})
frame
```

Out[28]:

	a	b	c	d
0	0	7	one	0
1	1	6	one	1
2	2	5	one	2
3	3	4	two	0
4	4	3	two	1
5	5	2	two	2
6	6	1	two	3

In [29]:
```
frame2 = frame.set_index(['c', 'd'])
frame2
```

Out[29]:

		a	b
c	d		
one	0	0	7
	1	1	6
	2	2	5
two	0	3	4
	1	4	3
	2	5	2
	3	6	1

In [30]:
```
frame.set_index(['c', 'd'], drop=False)
```

Out[30]:

		a	b	c	d
c	d				
one	0	0	7	one	0
	1	1	6	one	1
	2	2	5	one	2
two	0	3	4	two	0
	1	4	3	two	1
	2	5	2	two	2
	3	6	1	two	3

In [31]:
```
frame2.reset_index()
```

Out[31]:

	c	d	a	b
0	one	0	0	7
1	one	1	1	6
2	one	2	2	5
3	two	0	3	4
4	two	1	4	3
5	two	2	5	2
6	two	3	6	1

Combining and Merging Datasets

Database-Style DataFrame Joins

In [32]:
```
df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
                    'data1': range(7)})
df2 = pd.DataFrame({'key': ['a', 'b', 'd'],
                    'data2': range(3)})
df1
```

Out[32]:

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

In [33]:
```
df2
```

Out[33]:

	key	data2
0	a	0
1	b	1
2	d	2

In [34]:
```
pd.merge(df1, df2)
```

Out[34]:

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

In [35]:
```
pd.merge(df1, df2, on='key')
```

Out[35]:

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

In [36]:
```
df3 = pd.DataFrame({'lkey': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
                    'data1': range(7)})
df4 = pd.DataFrame({'rkey': ['a', 'b', 'd'],
                    'data2': range(3)})
df3
```

Out[36]:

	lkey	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

In [37]:
```
df4
```

Out[37]:

	rkey	data2
0	a	0
1	b	1
2	d	2

In [38]:
```
pd.merge(df3, df4, left_on='lkey', right_on='rkey')
```

Out[38]:

	lkey	data1	rkey	data2
0	b	0	b	1
1	b	1	b	1
2	b	6	b	1
3	a	2	a	0
4	a	4	a	0
5	a	5	a	0

In [40]:
```
pd.merge(df1, df2, how='outer')
```

Out[40]:

	key	data1	data2
0	b	0.0	1.0
1	b	1.0	1.0
2	b	6.0	1.0
3	a	2.0	0.0
4	a	4.0	0.0
5	a	5.0	0.0
6	c	3.0	NaN
7	d	NaN	2.0

In [41]:
```
pd.merge(df1, df2, how='inner')
```

Out[41]:

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

In [42]:
```
pd.merge(df1, df2, how='left')
```

Out[42]:

	key	data1	data2
0	b	0	1.0
1	b	1	1.0
2	a	2	0.0
3	c	3	NaN
4	a	4	0.0
5	a	5	0.0
6	b	6	1.0

In [43]:
```
pd.merge(df1, df2, how='right')
```

Out[43]:

	key	data1	data2
0	a	2.0	0
1	a	4.0	0
2	a	5.0	0
3	b	0.0	1
4	b	1.0	1
5	b	6.0	1
6	d	NaN	2

In [44]:
```
df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'b'],
                    'data1': range(6)})
df2 = pd.DataFrame({'key': ['a', 'b', 'a', 'b', 'd'],
                    'data2': range(5)})
df1
```

Out[44]:

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	b	5

In [45]:
```
df2
```

Out[45]:

	key	data2
0	a	0
1	b	1
2	a	2
3	b	3
4	d	4

In [46]:
```
pd.merge(df1, df2, on='key', how='left')
```

Out[46]:

	key	data1	data2
0	b	0	1.0
1	b	0	3.0
2	b	1	1.0
3	b	1	3.0
4	a	2	0.0
5	a	2	2.0
6	c	3	NaN
7	a	4	0.0
8	a	4	2.0
9	b	5	1.0
10	b	5	3.0

In [47]:
```
pd.merge(df1, df2, how='inner')
```

Out[47]:

	key	data1	data2
0	b	0	1
1	b	0	3
2	b	1	1
3	b	1	3
4	b	5	1
5	b	5	3
6	a	2	0
7	a	2	2
8	a	4	0
9	a	4	2

In [48]:
```
left = pd.DataFrame({'key1': ['foo', 'foo', 'bar'],
                     'key2': ['one', 'two', 'one'],
                     'lval': [1, 2, 3]})
right = pd.DataFrame({'key1': ['foo', 'foo', 'bar', 'bar'],
                      'key2': ['one', 'one', 'one', 'two'],
```

```
                       'rval': [4, 5, 6, 7]})
left
```

Out[48]:

	key1	key2	lval
0	foo	one	1
1	foo	two	2
2	bar	one	3

In [49]:
```
right
```

Out[49]:

	key1	key2	rval
0	foo	one	4
1	foo	one	5
2	bar	one	6
3	bar	two	7

In [50]:
```
pd.merge(left, right, on=['key1', 'key2'], how='outer')
```

Out[50]:

	key1	key2	lval	rval
0	foo	one	1.0	4.0
1	foo	one	1.0	5.0
2	foo	two	2.0	NaN
3	bar	one	3.0	6.0
4	bar	two	NaN	7.0

In [51]:
```
pd.merge(left, right, on='key1')
```

Out[51]:

	key1	key2_x	lval	key2_y	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

In [52]:
```
pd.merge(left, right, on='key1', suffixes=('_left', '_right'))
```

Out[52]:

	key1	key2_left	lval	key2_right	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

Merging on Index

In [53]:
```
left1 = pd.DataFrame({'key': ['a', 'b', 'a', 'a', 'b', 'c'],
                      'value': range(6)})
right1 = pd.DataFrame({'group_val': [3.5, 7]}, index=['a', 'b'])
left1
```

Out[53]:

	key	value
0	a	0
1	b	1
2	a	2
3	a	3
4	b	4
5	c	5

In [54]:
```
right1
```

Out[54]:

	group_val
a	3.5
b	7.0

In [55]:
```
pd.merge(left1, right1, left_on='key', right_index=True)
```

Out[55]:

	key	value	group_val
0	a	0	3.5
2	a	2	3.5
3	a	3	3.5
1	b	1	7.0
4	b	4	7.0

In [56]:
```
pd.merge(left1, right1, left_on='key', right_index=True, how='outer')
```

Out[56]:

	key	value	group_val
0	a	0	3.5
2	a	2	3.5
3	a	3	3.5
1	b	1	7.0
4	b	4	7.0
5	c	5	NaN

In [57]:
```
lefth = pd.DataFrame({'key1': ['Ohio', 'Ohio', 'Ohio',
                               'Nevada', 'Nevada'],
                      'key2': [2000, 2001, 2002, 2001, 2002],
                      'data': np.arange(5.)})
righth = pd.DataFrame(np.arange(12).reshape((6, 2)),
                      index=[['Nevada', 'Nevada', 'Ohio', 'Ohio',
                              'Ohio', 'Ohio'],
                             [2001, 2000, 2000, 2000, 2001, 2002]],
                      columns=['event1', 'event2'])
lefth
```

Out[57]:

	key1	key2	data
0	Ohio	2000	0.0
1	Ohio	2001	1.0
2	Ohio	2002	2.0
3	Nevada	2001	3.0
4	Nevada	2002	4.0

In [58]:
```
righth
```

Out[58]:

		event1	event2
Nevada	**2001**	0	1
	2000	2	3
Ohio	**2000**	4	5
	2000	6	7
	2001	8	9
	2002	10	11

In [59]:
```
pd.merge(lefth, righth, left_on=['key1', 'key2'], right_index=True)
```

Out[59]:

	key1	key2	data	event1	event2
0	Ohio	2000	0.0	4	5
0	Ohio	2000	0.0	6	7
1	Ohio	2001	1.0	8	9
2	Ohio	2002	2.0	10	11
3	Nevada	2001	3.0	0	1

In [60]:
```
pd.merge(lefth, righth, left_on=['key1', 'key2'],
         right_index=True, how='outer')
```

Out[60]:

	key1	key2	data	event1	event2
0	Ohio	2000	0.0	4.0	5.0
0	Ohio	2000	0.0	6.0	7.0
1	Ohio	2001	1.0	8.0	9.0
2	Ohio	2002	2.0	10.0	11.0
3	Nevada	2001	3.0	0.0	1.0
4	Nevada	2002	4.0	NaN	NaN
4	Nevada	2000	NaN	2.0	3.0

In [61]:
```
left2 = pd.DataFrame([[1., 2.], [3., 4.], [5., 6.]],
                     index=['a', 'c', 'e'],
                     columns=['Ohio', 'Nevada'])
right2 = pd.DataFrame([[7., 8.], [9., 10.], [11., 12.], [13, 14]],
                      index=['b', 'c', 'd', 'e'],
                      columns=['Missouri', 'Alabama'])
left2
```

Out[61]:

	Ohio	Nevada
a	1.0	2.0
c	3.0	4.0
e	5.0	6.0

In [62]:
```
right2
```

Out[62]:

	Missouri	Alabama
b	7.0	8.0
c	9.0	10.0
d	11.0	12.0
e	13.0	14.0

In [63]:
```
pd.merge(left2, right2, how='outer', left_index=True, right_index=True)
```

Out[63]:

	Ohio	Nevada	Missouri	Alabama
a	1.0	2.0	NaN	NaN
b	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0
d	NaN	NaN	11.0	12.0
e	5.0	6.0	13.0	14.0

In [64]:
```
left2.join(right2, how='outer')
```

Out[64]:

	Ohio	Nevada	Missouri	Alabama
a	1.0	2.0	NaN	NaN
b	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0
d	NaN	NaN	11.0	12.0
e	5.0	6.0	13.0	14.0

In [66]:
```
left1
```

Out[66]:

	key	value
0	a	0
1	b	1
2	a	2
3	a	3
4	b	4
5	c	5

In [67]:
```
right1
```

Out[67]:

	group_val
a	3.5
b	7.0

In [65]:
```
left1.join(right1, on='key')
```

Out[65]:

	key	value	group_val
0	a	0	3.5
1	b	1	7.0
2	a	2	3.5
3	a	3	3.5
4	b	4	7.0
5	c	5	NaN

In [68]:
```
another = pd.DataFrame([[7., 8.], [9., 10.], [11., 12.], [16., 17.]],
                       index=['a', 'c', 'e', 'f'],
                       columns=['New York', 'Oregon'])
another
```

Out[68]:

	New York	Oregon
a	7.0	8.0
c	9.0	10.0
e	11.0	12.0
f	16.0	17.0

In [70]:
```
left2
```

Out[70]:

	Ohio	Nevada
a	1.0	2.0
c	3.0	4.0
e	5.0	6.0

In [71]:
```
right2
```

Out[71]:

	Missouri	Alabama
b	7.0	8.0
c	9.0	10.0
d	11.0	12.0
e	13.0	14.0

In [69]:
```
left2.join([right2, another])
```

Out[69]:

	Ohio	Nevada	Missouri	Alabama	New York	Oregon
a	1.0	2.0	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0	9.0	10.0
e	5.0	6.0	13.0	14.0	11.0	12.0

In [72]:
```
left2.join([right2, another], how='outer')
```

Out[72]:

	Ohio	Nevada	Missouri	Alabama	New York	Oregon
a	1.0	2.0	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0	9.0	10.0
e	5.0	6.0	13.0	14.0	11.0	12.0
b	NaN	NaN	7.0	8.0	NaN	NaN
d	NaN	NaN	11.0	12.0	NaN	NaN
f	NaN	NaN	NaN	NaN	16.0	17.0

Concatenating Along an Axis

In [73]:
```
arr = np.arange(12).reshape((3, 4))
arr
```

Out[73]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [74]:
```
np.concatenate([arr, arr], axis=1)
```

Out[74]:
```
array([[ 0,  1,  2,  3,  0,  1,  2,  3],
       [ 4,  5,  6,  7,  4,  5,  6,  7],
       [ 8,  9, 10, 11,  8,  9, 10, 11]])
```

In [75]:
```
s1 = pd.Series([0, 1], index=['a', 'b'])
s2 = pd.Series([2, 3, 4], index=['c', 'd', 'e'])
s3 = pd.Series([5, 6], index=['f', 'g'])
```

In [76]:
```
pd.concat([s1, s2, s3])
```

Out[76]:
```
a    0
b    1
c    2
d    3
e    4
f    5
g    6
dtype: int64
```

In [77]:
```
pd.concat([s1, s2, s3], axis=1)
```

Out[77]:

	0	1	2
a	0.0	NaN	NaN
b	1.0	NaN	NaN
c	NaN	2.0	NaN
d	NaN	3.0	NaN
e	NaN	4.0	NaN
f	NaN	NaN	5.0
g	NaN	NaN	6.0

In [78]:
```
s4 = pd.concat([s1, s3])
s4
```

Out[78]:
```
a    0
b    1
f    5
g    6
dtype: int64
```

In [80]:
```
pd.concat([s1, s4], axis=1)
```

Out[80]:

	0	1
a	0.0	0
b	1.0	1
f	NaN	5
g	NaN	6

In [79]:
```
pd.concat([s1, s4], axis=1, join='inner')
```

Out[79]:

	0	1
a	0	0
b	1	1

In [81]:
```
pd.concat([s1, s4], axis=1, join_axes=[['a', 'c', 'b', 'e']])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_21488\215449900.py in <module>
----> 1 pd.concat([s1, s4], axis=1, join_axes=[['a', 'c', 'b', 'e']])

C:\PythonDSA\anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

TypeError: concat() got an unexpected keyword argument 'join_axes'
```

In [83]:
```
pd.concat([s1, s4], axis=1).reindex(['a', 'c', 'b', 'e'])
```

Out[83]:

	0	1
a	0.0	0.0
c	NaN	NaN
b	1.0	1.0
e	NaN	NaN

In [84]:
```
result = pd.concat([s1, s1, s3], keys=['one', 'two', 'three'])
result
```

Out[84]:
```
one    a    0
       b    1
two    a    0
       b    1
three  f    5
       g    6
dtype: int64
```

In [85]:
```
result.unstack()
```

Out[85]:

	a	b	f	g
one	0.0	1.0	NaN	NaN
two	0.0	1.0	NaN	NaN
three	NaN	NaN	5.0	6.0

In [86]:
```
pd.concat([s1, s2, s3], axis=1, keys=['one', 'two', 'three'])
```

Out[86]:

	one	two	three
a	0.0	NaN	NaN
b	1.0	NaN	NaN
c	NaN	2.0	NaN
d	NaN	3.0	NaN
e	NaN	4.0	NaN
f	NaN	NaN	5.0
g	NaN	NaN	6.0

In [87]:
```
df1 = pd.DataFrame(np.arange(6).reshape(3, 2), index=['a', 'b', 'c'],
                   columns=['one', 'two'])
df2 = pd.DataFrame(5 + np.arange(4).reshape(2, 2), index=['a', 'c'],
                   columns=['three', 'four'])
df1
```

Out[87]:

	one	two
a	0	1
b	2	3
c	4	5

In [88]:
```
df2
```

Out[88]:

	three	four
a	5	6
c	7	8

In [89]:
```
pd.concat([df1, df2], axis=1, keys=['level1', 'level2'])
```

Out[89]:

	level1		level2	
	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

In [90]:
```
pd.concat([df1, df2], axis=1)
```

Out[90]:

	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

In [91]:
```
pd.concat({'level1': df1, 'level2': df2}, axis=1)
```

Out[91]:

	level1		level2	
	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

In [92]:
```
pd.concat([df1, df2], axis=1, keys=['level1', 'level2'],
          names=['upper', 'lower'])
```

Out[92]:

upper	level1		level2	
lower	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

In [93]:
```
df1 = pd.DataFrame(np.random.randn(3, 4), columns=['a', 'b', 'c', 'd'])
df2 = pd.DataFrame(np.random.randn(2, 3), columns=['b', 'd', 'a'])
df1
```

Out[93]:

	a	b	c	d
0	1.246435	1.007189	-1.296221	0.274992
1	0.228913	1.352917	0.886429	-2.001637
2	-0.371843	1.669025	-0.438570	-0.539741

In [94]:
```
df2
```

Out[94]:

	b	d	a
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614

In [95]:
```
pd.concat([df1, df2], ignore_index=True)
```

Out[95]:

	a	b	c	d
0	1.246435	1.007189	-1.296221	0.274992
1	0.228913	1.352917	0.886429	-2.001637
2	-0.371843	1.669025	-0.438570	-0.539741
3	-1.021228	0.476985	NaN	3.248944
4	0.302614	-0.577087	NaN	0.124121

Combining Data with Overlap

In [96]:
```
a = pd.Series([np.nan, 2.5, np.nan, 3.5, 4.5, np.nan],
              index=['f', 'e', 'd', 'c', 'b', 'a'])
b = pd.Series(np.arange(len(a), dtype=np.float64),
              index=['f', 'e', 'd', 'c', 'b', 'a'])
b[-1] = np.nan
a
```

Out[96]:
```
f    NaN
e    2.5
d    NaN
c    3.5
b    4.5
a    NaN
dtype: float64
```

In [97]:
```
b
```

Out[97]:
```
f    0.0
e    1.0
d    2.0
c    3.0
b    4.0
a    NaN
dtype: float64
```

In [98]:
```
np.where(pd.isnull(a), b, a)
```

Out[98]:
```
array([0. , 2.5, 2. , 3.5, 4.5, nan])
```

In [99]:
```
b.combine_first(a)
```

Out[99]:
```
f    0.0
e    1.0
d    2.0
c    3.0
b    4.0
a    NaN
dtype: float64
```

In [100…
```
b[:-2].combine_first(a[2:])
```

Out[100]:
```
a    NaN
b    4.5
c    3.0
d    2.0
e    1.0
f    0.0
dtype: float64
```

In [101…
```
df1 = pd.DataFrame({'a': [1., np.nan, 5., np.nan],
                    'b': [np.nan, 2., np.nan, 6.],
                    'c': range(2, 18, 4)})
df2 = pd.DataFrame({'a': [5., 4., np.nan, 3., 7.],
                    'b': [np.nan, 3., 4., 6., 8.]})
df1
```

Out[101]:

	a	b	c
0	1.0	NaN	2
1	NaN	2.0	6
2	5.0	NaN	10
3	NaN	6.0	14

In [102…
```
df2
```

Out[102]:

	a	b
0	5.0	NaN
1	4.0	3.0
2	NaN	4.0
3	3.0	6.0
4	7.0	8.0

In [103…
```
df1.combine_first(df2)
```

Out[103]:

	a	b	c
0	1.0	NaN	2.0
1	4.0	2.0	6.0
2	5.0	4.0	10.0
3	3.0	6.0	14.0
4	7.0	8.0	NaN

Reshaping and Pivoting

Reshaping with Hierarchical Indexing

In [104…
```
data = pd.DataFrame(np.arange(6).reshape((2, 3)),
                    index=pd.Index(['Ohio', 'Colorado'], name='state'),
                    columns=pd.Index(['one', 'two', 'three'],
                    name='number'))
data
```

Out[104]:

number	one	two	three
state			
Ohio	0	1	2
Colorado	3	4	5

In [105…
```
result = data.stack()
result
```

Out[105]:
```
state     number
Ohio      one       0
          two       1
          three     2
Colorado  one       3
          two       4
          three     5
dtype: int32
```

In [106…
```
result.unstack()
```

Out[106]:

number	one	two	three
state			
Ohio	0	1	2
Colorado	3	4	5

In [107…
```
result.unstack(0)
```

Out[107]:

state	Ohio	Colorado
number		
one	0	3
two	1	4
three	2	5

In [108…
```
result.unstack('state')
```

Out[108]:

state	Ohio	Colorado
number		
one	0	3
two	1	4
three	2	5

In [110…
```
s1 = pd.Series([0, 1, 2, 3], index=['a', 'b', 'c', 'd'])
s2 = pd.Series([4, 5, 6], index=['c', 'd', 'e'])
s1
```

Out[110]:
```
a    0
b    1
c    2
d    3
dtype: int64
```

In [111…
```
s2
```

Out[111]:
```
c    4
d    5
e    6
dtype: int64
```

In [112…
```
data2 = pd.concat([s1, s2], keys=['one', 'two'])
data2
```

Out[112]:
```
one  a    0
     b    1
     c    2
     d    3
two  c    4
     d    5
     e    6
dtype: int64
```

In [113…
```
data2.unstack()
```

Out[113]:

	a	b	c	d	e
one	0.0	1.0	2.0	3.0	NaN
two	NaN	NaN	4.0	5.0	6.0

In [114…
```
data2.unstack()
```

Out[114]:

	a	b	c	d	e
one	0.0	1.0	2.0	3.0	NaN
two	NaN	NaN	4.0	5.0	6.0

In [115…
```
data2.unstack().stack()
```

Out[115]:
```
one  a    0.0
     b    1.0
     c    2.0
     d    3.0
two  c    4.0
     d    5.0
     e    6.0
dtype: float64
```

In [116…
```
data2.unstack().stack(dropna=False)
```

Out[116]:
```
one  a    0.0
     b    1.0
     c    2.0
     d    3.0
     e    NaN
two  a    NaN
     b    NaN
     c    4.0
     d    5.0
     e    6.0
dtype: float64
```

In [117…
```
df = pd.DataFrame({'left': result, 'right': result + 5},
                  columns=pd.Index(['left', 'right'], name='side'))
df
```

Out[117]:

	side	left	right
state	number		
Ohio	one	0	5
	two	1	6
	three	2	7
Colorado	one	3	8
	two	4	9
	three	5	10

In [118…
```
df.unstack('state')
```

Out[118]:

side	left		right	
state	Ohio	Colorado	Ohio	Colorado
number				
one	0	3	5	8
two	1	4	6	9
three	2	5	7	10

In [119…
```
df.unstack('state').stack('side')
```

Out[119]:

	state	Colorado	Ohio
number	side		
one	left	3	0
	right	8	5
two	left	4	1
	right	9	6
three	left	5	2
	right	10	7

Pivoting “Long” to “Wide” Format

In [120…
```
data = pd.read_csv('examples/macrodata.csv')
data.head()
```

Out[120]:

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate	unemp
0	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82	5.8
1	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08	5.1
2	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82	5.3
3	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33	5.6
4	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50	5.2

In [122…
```
periods = pd.PeriodIndex(year=data.year, quarter=data.quarter,
                         name='date')
periods
```

Out[122]:
```
PeriodIndex(['1959Q1', '1959Q2', '1959Q3', '1959Q4', '1960Q1', '1960Q2',
             '1960Q3', '1960Q4', '1961Q1', '1961Q2',
             ...
             '2007Q2', '2007Q3', '2007Q4', '2008Q1', '2008Q2', '2008Q3',
             '2008Q4', '2009Q1', '2009Q2', '2009Q3'],
            dtype='period[Q-DEC]', name='date', length=203)
```

In [123…
```
columns = pd.Index(['realgdp', 'infl', 'unemp'], name='item')
columns
```

Out[123]:
```
Index(['realgdp', 'infl', 'unemp'], dtype='object', name='item')
```

In [125…
```
data = data.reindex(columns=columns)
data
```

Out[125]:

item	realgdp	infl	unemp
0	2710.349	0.00	5.8
1	2778.801	2.34	5.1
2	2775.488	2.74	5.3
3	2785.204	0.27	5.6
4	2847.699	2.31	5.2
…	…	…	…
198	13324.600	-3.16	6.0
199	13141.920	-8.79	6.9
200	12925.410	0.94	8.1
201	12901.504	3.37	9.2
202	12990.341	3.56	9.6

203 rows × 3 columns

In [126…
```
data.index = periods.to_timestamp('D', 'end')
```

In [127…
```
ldata = data.stack().reset_index().rename(columns={0: 'value'})
```

In [128…
```
ldata[:10]
```

Out[128]:

	date	item	value
0	1959-03-31 23:59:59.999999999	realgdp	2710.349
1	1959-03-31 23:59:59.999999999	infl	0.000
2	1959-03-31 23:59:59.999999999	unemp	5.800
3	1959-06-30 23:59:59.999999999	realgdp	2778.801
4	1959-06-30 23:59:59.999999999	infl	2.340
5	1959-06-30 23:59:59.999999999	unemp	5.100
6	1959-09-30 23:59:59.999999999	realgdp	2775.488
7	1959-09-30 23:59:59.999999999	infl	2.740
8	1959-09-30 23:59:59.999999999	unemp	5.300
9	1959-12-31 23:59:59.999999999	realgdp	2785.204

In [129…
```
pivoted = ldata.pivot('date', 'item', 'value')
pivoted
```

Out[129]:

item	infl	realgdp	unemp
date			
1959-03-31 23:59:59.999999999	0.00	2710.349	5.8
1959-06-30 23:59:59.999999999	2.34	2778.801	5.1
1959-09-30 23:59:59.999999999	2.74	2775.488	5.3
1959-12-31 23:59:59.999999999	0.27	2785.204	5.6
1960-03-31 23:59:59.999999999	2.31	2847.699	5.2
…	…	…	…
2008-09-30 23:59:59.999999999	-3.16	13324.600	6.0
2008-12-31 23:59:59.999999999	-8.79	13141.920	6.9
2009-03-31 23:59:59.999999999	0.94	12925.410	8.1
2009-06-30 23:59:59.999999999	3.37	12901.504	9.2
2009-09-30 23:59:59.999999999	3.56	12990.341	9.6

203 rows × 3 columns

In [130…
```
ldata['value2'] = np.random.randn(len(ldata))
ldata[:10]
```

Out[130]:

	date	item	value	value2
0	1959-03-31 23:59:59.999999999	realgdp	2710.349	0.523772
1	1959-03-31 23:59:59.999999999	infl	0.000	0.000940
2	1959-03-31 23:59:59.999999999	unemp	5.800	1.343810
3	1959-06-30 23:59:59.999999999	realgdp	2778.801	-0.713544
4	1959-06-30 23:59:59.999999999	infl	2.340	-0.831154
5	1959-06-30 23:59:59.999999999	unemp	5.100	-2.370232
6	1959-09-30 23:59:59.999999999	realgdp	2775.488	-1.860761
7	1959-09-30 23:59:59.999999999	infl	2.740	-0.860757
8	1959-09-30 23:59:59.999999999	unemp	5.300	0.560145
9	1959-12-31 23:59:59.999999999	realgdp	2785.204	-1.265934

In [131…

```
pivoted = ldata.pivot('date', 'item')
pivoted[:5]
```

Out[131]:

	value			value2		
item	**infl**	**realgdp**	**unemp**	**infl**	**realgdp**	**unemp**
date						
1959-03-31 23:59:59.999999999	0.00	2710.349	5.8	0.000940	0.523772	1.343810
1959-06-30 23:59:59.999999999	2.34	2778.801	5.1	-0.831154	-0.713544	-2.370232
1959-09-30 23:59:59.999999999	2.74	2775.488	5.3	-0.860757	-1.860761	0.560145
1959-12-31 23:59:59.999999999	0.27	2785.204	5.6	0.119827	-1.265934	-1.063512
1960-03-31 23:59:59.999999999	2.31	2847.699	5.2	-2.359419	0.332883	-0.199543

In [132…

```
pivoted['value'][:5]
```

Out[132]:

item	infl	realgdp	unemp
date			
1959-03-31 23:59:59.999999999	0.00	2710.349	5.8
1959-06-30 23:59:59.999999999	2.34	2778.801	5.1
1959-09-30 23:59:59.999999999	2.74	2775.488	5.3
1959-12-31 23:59:59.999999999	0.27	2785.204	5.6
1960-03-31 23:59:59.999999999	2.31	2847.699	5.2

In [133…

```
unstacked = ldata.set_index(['date', 'item']).unstack('item')
unstacked[:7]
```

Out[133]:

	value			value2		
item	infl	realgdp	unemp	infl	realgdp	unemp
date						
1959-03-31 23:59:59.999999999	0.00	2710.349	5.8	0.000940	0.523772	1.343810
1959-06-30 23:59:59.999999999	2.34	2778.801	5.1	-0.831154	-0.713544	-2.370232
1959-09-30 23:59:59.999999999	2.74	2775.488	5.3	-0.860757	-1.860761	0.560145
1959-12-31 23:59:59.999999999	0.27	2785.204	5.6	0.119827	-1.265934	-1.063512
1960-03-31 23:59:59.999999999	2.31	2847.699	5.2	-2.359419	0.332883	-0.199543
1960-06-30 23:59:59.999999999	0.14	2834.390	5.2	-0.970736	-1.541996	-1.307030
1960-09-30 23:59:59.999999999	2.70	2839.022	5.6	0.377984	0.286350	-0.753887

Pivoting “Wide” to “Long” Format

In [134…

```
df = pd.DataFrame({'key': ['foo', 'bar', 'baz'],
                   'A': [1, 2, 3],
                   'B': [4, 5, 6],
                   'C': [7, 8, 9]})
df
```

Out[134]:

	key	A	B	C
0	foo	1	4	7
1	bar	2	5	8
2	baz	3	6	9

In [135…

```
melted = pd.melt(df, ['key'])
melted
```

Out[135]:

	key	variable	value
0	foo	A	1
1	bar	A	2
2	baz	A	3
3	foo	B	4
4	bar	B	5
5	baz	B	6
6	foo	C	7
7	bar	C	8
8	baz	C	9

In [136…

```
reshaped = melted.pivot('key', 'variable', 'value')
reshaped
```

Out[136]:

variable	A	B	C
key			
bar	2	5	8
baz	3	6	9
foo	1	4	7

In [137…
```
reshaped2 = melted.pivot('variable', 'key', 'value')
reshaped2
```

Out[137]:

key	bar	baz	foo
variable			
A	2	3	1
B	5	6	4
C	8	9	7

In [138…
```
reshaped.reset_index()
```

Out[138]:

variable	key	A	B	C
0	bar	2	5	8
1	baz	3	6	9
2	foo	1	4	7

In [139…
```
pd.melt(df, id_vars=['key'], value_vars=['A', 'B'])
```

Out[139]:

	key	variable	value
0	foo	A	1
1	bar	A	2
2	baz	A	3
3	foo	B	4
4	bar	B	5
5	baz	B	6

In [140…
```
pd.melt(df, value_vars=['A', 'B', 'C'])
```

Out[140]:

	variable	value
0	A	1
1	A	2
2	A	3
3	B	4
4	B	5
5	B	6
6	C	7
7	C	8
8	C	9

In [141…

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

Out[141]:

	variable	value
0	key	foo
1	key	bar
2	key	baz
3	A	1
4	A	2
5	A	3
6	B	4
7	B	5
8	B	6

Conclusion