

Manipulasi Data dengan Pandas 2

Penggabungan Series/Dataframe

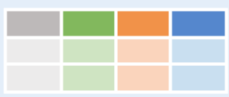
Pivot, Melt, Stack & Unstack

Topik

- M14 - Manipulasi Data Pandas - Bagian 2
 - **Penggabungan Series/Dataframe**
 - Bagaimana Cara Menggabungkan Pandas Series/Dataframe?
 - Append
 - Concat
 - Merge
 - Join
 - **Pivot, Melt, Stack & Unstack**
 - Pivot
 - Melt
 - Stack & Unstack

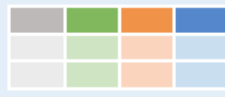
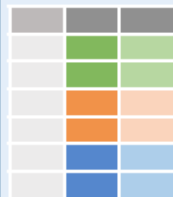
- 1. Penggabungan Series/Dataframe**
- 2. Pivot, Melt, Stack & Unstack**

Reshaping Data – Change the layout of a data set



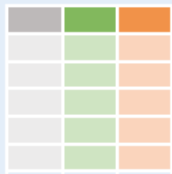
pd.melt(df)

Gather columns into rows.



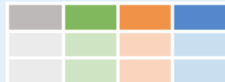
df.pivot(columns='var', values='val')

Spread rows into columns.



pd.concat([df1, df2])

Append rows of DataFrames



pd.concat([df1, df2], axis=1)

Append columns of DataFrames

df.sort_values('mpg')

Order rows by values of a column (low to high).

df.sort_values('mpg', ascending=False)

Order rows by values of a column (high to low).

df.rename(columns = {'y': 'year'})

Rename the columns of a DataFrame

df.sort_index()

Sort the index of a DataFrame

df.reset_index()

Reset index of DataFrame to row numbers, moving index to columns.

df.drop(['Length', 'Height'], axis=1)

Drop columns from DataFrame

<http://talimi.se/p/pandas/>

1. Penggabungan Series/Dataframe

- Mempelajari bagaimana cara menggabungkan dua atau lebih Pandas series/dataframe terpisah menjadi sebuah series/dataframe.
- Dengan demikian seluruh data telah menjadi satu kesatuan sehingga dapat diproses untuk dianalisis lebih lanjut.

1.2. Menggabungkan Pandas Series/Dataframe

- Sebagai seorang praktisi data, pasti sering kali bertemu dengan banyak file sekaligus dan data yang dibutuhkan tersebar di berbagai file.
- Dibutuhkan metode untuk menggabungkan semua informasi yang dibutuhkan dari setiap file tersebut.
- Dengan menggunakan excel atau tools pengolah spreadsheet lain hal itu bisa terjadi mungkin dengan menggunakan copy paste file satu ke file lainnya atau yang agak canggih menggunakan method importRange di google sheets.
- Tetapi tentu hal itu tidak bisa diandalkan ketika berurusan dengan big data yang datanya bisa milyaran rows dengan informasi yang tidak terbatas.
- Python dan Pandas adalah satu-satunya cara untuk mengatasinya.
- Terdapat beberapa metode untuk menggabungkan Series/Dataframe di Pandas, yaitu:
 - append
 - concat
 - merge
 - join

Penggabungan Series/Dataframe: 1. Append

- Method **.append()** dapat digunakan pada **dataframe/series** untuk menambah **row**.
- Jika di SQL memiliki 2 tabel atau lebih maka dapat menggabungkannya secara vertikal dengan Union.
- Jadi **SQL Union** ekuivalen dengan method **.append()** di Pandas.
- Method **.append** pada series:

```
1 import pandas as pd
2 # Buat series of int (s1) dan series of string (s2)
3 s1 = pd.Series([1,2,3,4,5,6])
4 s2 = pd.Series(["a","b","c","d","e","f"])
5 # Terapkan method append
6 s1_append_s2 = s1.append(s2)
7 print("Series - append:\n", s1_append_s2)
```

```
Series - append:
0    1
1    2
2    3
3    4
4    5
5    6
0    a
1    b
2    c
3    d
4    e
5    f
dtype: object
```

Penggabungan Series/Dataframe: 1. Append

- method .append pada dataframe

```
8 # Buat dataframe df1 dan df2
9 df1 = pd.DataFrame({'a':[1,2],
10                      'b':[3,4]})
11 df2 = pd.DataFrame({'b':[1,2],
12                      'a':[3,4]})
13 # Terapkan method append
14 df1_append_df2 = df1.append(df2)
15 print("Dataframe - append:\n", df1_append_df2)
```

Dataframe - append:

	a	b
0	1	3
1	2	4
0	3	1
1	4	2

Tugas 1. Penggabungan Series/Dataframe: Append

- Terapkanlah method append dengan series s1 dan dataframe df1 ditempatkan setelah series s2 dan dataframe df2 masing-masingnya.
- Method append dapat digunakan untuk series dan dataframe sebagai berikut:
 - * nama_series_1.append(nama_series_2)
 - * nama_dataframe_1.append(nama_dataframe_2)

Series - append:

0	a
1	b
2	c
3	d
4	e
5	f
0	1
1	2
2	3
3	4
4	5
5	6

dtype: object

Dataframe - append:

	a	b
0	3	1
1	4	2
0	1	3
1	2	4

```
import __
# Buat series of int (s1) dan series of string (s2)
s1 = __
s2 = __

# Terapkan method append
s2_append_s1 = __
print("Series - append:\n", __)

# Buat dataframe df1 dan df2
df1 = __({__:__,
          __:__})
df2 = __({__:__,
          __:__})

# Terapkan method append
df2_append_df1 = __
print("Dataframe - append:\n", __)
```

Penggabungan Series/Dataframe: 2. Concat

- Method **.concat()** dapat digunakan pada **dataframe** yang ditujukan untuk penggabungan baik dalam **row-wise** (dalam arah) atau **column-wise**.
- Method **.concat** pada row-wise:

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({'a':[1,2],
4                     'b':[3,4]})
5 df2 = pd.DataFrame({'b':[1,2],
6                     'a':[3,4]})
7 # Terapkan method concat row-wise
8 row_wise_concat = pd.concat([df1, df2])
9 print("Row-wise - concat:\n", row_wise_concat)
```

Row-wise - concat:

	a	b
0	1	3
1	2	4
0	3	1
1	4	2

- Method **.concat** pada column-wise:

```
10 # Terapkan method concat column-wise
11 col_wise_concat = pd.concat([df1, df2], axis=1)
12 print("Column-wise - concat:\n", col_wise_concat)
```

Column-wise - concat:

	a	b	a	b
0	1	3	3	1
1	2	4	4	2

Penggabungan Series/Dataframe: 2. Concat

Dapat juga menambahkan identifier dari dataframe untuk data yang ditambahkan

```
13 # Penambahan identifier --> membentuk hasil penggabungan multiindex
14 multiindex_concat = pd.concat([df1,df2], axis=0, keys=['df1','df2'])
15 print("Multiindex - concat:\n", multiindex_concat)
```

Multiindex - concat:

		a	b
df1	0	1	3
	1	2	4
df2	0	3	1
	1	4	2

Tugas 2. Penggabungan Series/Dataframe: Concat

- Balikkanlah posisi kedua dataframe yang akan digabungkan dengan concat.
- Pastikan kamu telah dengan benar menempatkan urutan kedua dataframe yang akan digabung dengan concat, yaitu [df2, df1].
- Pastikan kamu melakukan penggabungan baik row-wise, column-wise, atau menggunakan identifier sesuai dengan keyword arguments yang diperlukan.
- Syntax umum:
`pd.concat(list_of_dataframe, axis=..., keys=...)`

```
Row-wise - concat:
   a  b
0  3  1
1  4  2
0  1  3
1  2  4
Column-wise - concat:
   a  b  a  b
0  3  1  1  3
1  4  2  2  4
Multiindex - concat:
      a  b
df1  0  3  1
     1  4  2
df2  0  1  3
     1  2  4
```

```
import ____
# Buat dataframe df1 dan df2
df1 = pd.DataFrame({'a':[1,2],
                    'b':[3,4]})
df2 = pd.DataFrame({'b':[1,2],
                    'a':[3,4]})
```

```
# Terapkan method concat row-wise
row_wise_concat = ____
print("Row-wise - concat:\n", ____)
```

```
# Terapkan method concat column-wise
col_wise_concat = ____
print("Column-wise - concat:\n", ____)
```

```
# Penambahan identifier --> membentuk hasil
penggabungan multiindex
multiindex_concat = ____
print("Multiindex - concat:\n", ____)
```

Penggabungan Series/Dataframe: 3. Merge (index tunggal)

- Method **.merge()** untuk menggabungkan Series/Dataframe yang bentuknya mirip dengan syntax **join** di SQL, specify left and right tables, join key dan how to join (left, right, inner, full outer).

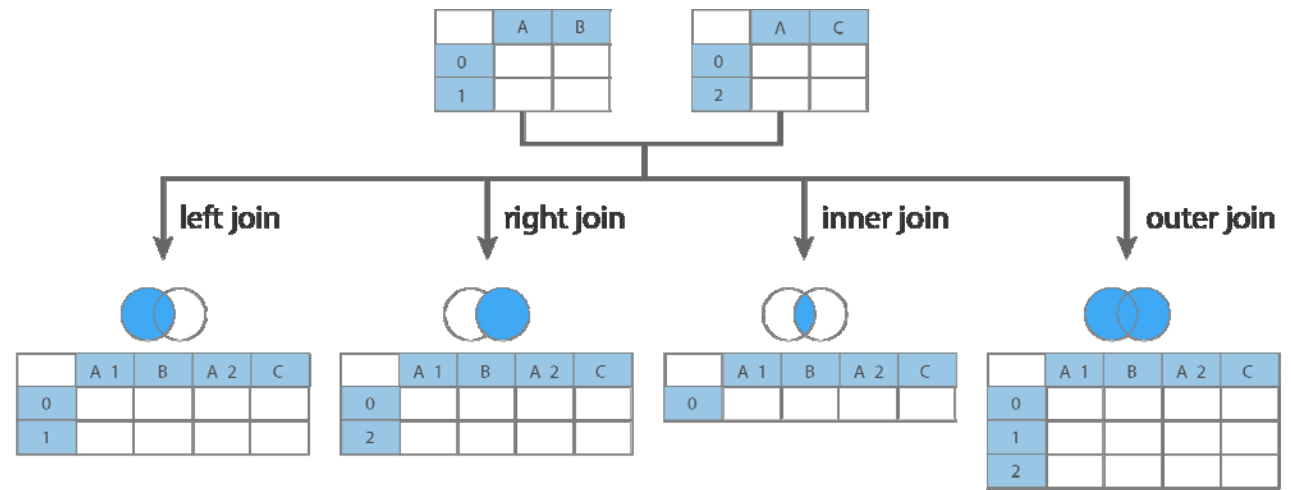
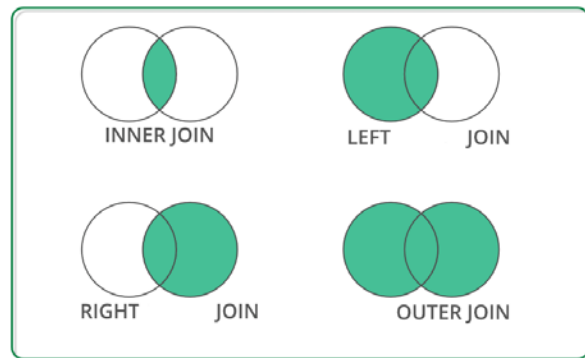
```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 })
8 df2 = pd.DataFrame({
9     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
10    'val3': [1, 2, 3, 4, 5],
11    'val4': [6, 7, 8, 8, 10]
12 })
```

Merge - Left:					
	key	val1	val2	val3	val4
0	k1	200	30	1.0	6.0
1	k2	500	50	NaN	NaN
2	k3	0	100	2.0	7.0
3	k4	500	20	NaN	NaN
4	k5	100	10	3.0	8.0

- pd.merge yang ekuivalen dengan SQL left join

```
13 # Merge yang ekuivalen dengan SQL left join
14 merge_df_left = pd.merge(left=df1, right=df2, how='left', left_on='key',
15    right_on='key')
15 print('Merge - Left:\n', merge_df_left)
```

Join in Pandas



Penggabungan Series/Dataframe: 3. Merge (index tunggal)

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 })
8 df2 = pd.DataFrame({
9     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
10    'val3': [1, 2, 3, 4, 5],
11    'val4': [6, 7, 8, 10]
12 })
```

- pd.merge yang ekuivalen dengan SQL right join

```
16 # Merge yang ekuivalen dengan SQL right join
17 merge_df_right = pd.merge(left=df1, right=df2, how='right', left_on='key',
18    right_on='key')
19 print('Merge - Right:\n', merge_df_right)
```

Merge - Right:					
	key	val1	val2	val3	val4
0	k1	200.0	30.0	1	6
1	k3	0.0	100.0	2	7
2	k5	100.0	10.0	3	8
3	k7	NaN	NaN	4	8
4	k10	NaN	NaN	5	10

- pd.merge yang ekuivalen dengan SQL inner join

```
19 # Merge yang ekuivalen dengan SQL inner join
20 merge_df_inner = pd.merge(left=df1, right=df2, how='inner', left_on='key',
21    right_on='key')
22 print('Merge - Inner:\n', merge_df_inner)
```

Merge - Inner:					
	key	val1	val2	val3	val4
0	k1	200	30	1	6
1	k3	0	100	2	7
2	k5	100	10	3	8

- pd.merge yang ekuivalen dengan SQL outer join

```
22 # Merge yang ekuivalen dengan SQL outer join
23 merge_df_outer = pd.merge(left=df1, right=df2, how='outer', left_on='key',
24    right_on='key')
25 print('Merge - Outer:\n', merge_df_outer)
```

Merge - Outer:					
	key	val1	val2	val3	val4
0	k1	200.0	30.0	1.0	6.0
1	k2	500.0	50.0	NaN	NaN
2	k3	0.0	100.0	2.0	7.0
3	k4	500.0	20.0	NaN	NaN
4	k5	100.0	10.0	3.0	8.0
5	k7	NaN	NaN	4.0	8.0
6	k10	NaN	NaN	5.0	10.0

Tugas 3. Penggabungan Series/Dataframe: Merge (index tunggal)

- Pada contoh sebelumnya, keyword argument left=df1 dan right=df2, untuk praktek gunakanlah keyword argument untuk left dan right masing-masingnya adalah df2 dan df1.
- Pastikan bahwa dalam melakukan merge df2 adalah left dataframe dan df1 merupakan right dataframe.
- Pastikan kembali bahwa keempat proses merge yang diminta secara berurutan yaitu left, right, inner dan outer.
- Untuk melakukan merge kamu dapat menggunakan pd.merge(left=__, right=__, how=__, left_on=__, right_on=__).

```
import __
# Buat dataframe df1 dan df2
df1 = pd.DataFrame({
    'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
    'val1': [200, 500, 0, 500, 100],
    'val2': [30, 50, 100, 20, 10]
})
df2 = pd.DataFrame({
    'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
    'val3': [1, 2, 3, 4, 5],
    'val4': [6, 7, 8, 8, 10]
})
# Merge yang ekuivalen dengan SQL left join
merge_df_left = __
print('Merge - Left:\n', __)
# Merge yang ekuivalen dengan SQL right join
merge_df_right = __
print('Merge - Right:\n', __)
# Merge yang ekuivalen dengan SQL inner join
merge_df_inner = __
print('Merge - Inner:\n', __)
# Merge yang ekuivalen dengan SQL outer join
merge_df_outer = __
print('Merge - Outer:\n', __)
```

Tugas 3. Penggabungan Series/Dataframe: Merge (index tunggal)

```
Merge - Left:
   key  val3  val4  val1  val2
0  k1     1     6  200.0  30.0
1  k3     2     7   0.0  100.0
2  k5     3     8  100.0   10.0
3  k7     4     8   NaN   NaN
4 k10     5    10   NaN   NaN

Merge - Right:
   key  val3  val4  val1  val2
0  k1   1.0   6.0  200    30
1  k3   2.0   7.0    0   100
2  k5   3.0   8.0  100    10
3  k2   NaN   NaN  500    50
4  k4   NaN   NaN  500    20

Merge - Inner:
   key  val3  val4  val1  val2
0  k1     1     6  200    30
1  k3     2     7    0   100
2  k5     3     8  100    10

Merge - Outer:
   key  val3  val4  val1  val2
0  k1   1.0   6.0  200.0  30.0
1  k3   2.0   7.0   0.0  100.0
2  k5   3.0   8.0  100.0   10.0
3  k7   4.0   8.0   NaN   NaN
4 k10   5.0  10.0   NaN   NaN
5  k2   NaN   NaN  500.0  50.0
6  k4   NaN   NaN  500.0  20.0
```

Penggabungan Series/Dataframe: 4. Merge (multi index)

- Penggunaan method **.merge** yang telah dipelajari sebelumnya adalah untuk **dataframe** dengan **index tunggal**.
- Bagaimana jika salah satu dataframe atau keseluruhan dataframe yang akan digabungkan tersebut memiliki **multiindex**?

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 }).set_index(['key', 'val2'])
8 print('Dataframe 1:\n', df1)
9 df2 = pd.DataFrame({
10     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
11     'val3': [1, 2, 3, 4, 5],
12     'val4': [6, 7, 8, 8, 10]
13 }).set_index(['key', 'val3'])
14 print('Dataframe 2:\n', df2)
```

Dataframe 1:

		val1
key	val2	
k1	30	200
k2	50	500
k3	100	0
k4	20	500
k5	10	100

Dataframe 2:

		val4
key	val3	
k1	1	6
k3	2	7
k5	3	8
k7	4	8
k10	5	10

- Jika digabungkan secara langsung seperti yang telah dilakukan pada bagian sebelumnya **terjadi kegagalan dalam merging kedua dataframe yang memiliki multi index**.

```
15 # Merge dataframe yang memiliki multi index
16 df_merge = pd.merge(df1, df2)
17 print(df_merge)
```

```
Traceback (most recent call last):
  File "<stdin>", line 18, in <module>
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/merge.py", line 47, in merge
    validate=validate)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/merge.py", line 524, in __init__
    self._validate_specification()
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/merge.py", line 1033, in _validate_specification
    lidx=self.left_index, ridx=self.right_index))
pandas.errors.MergeError: No common columns to perform merge on. Merge options: left_on=None, right_on=None, left_index=False, right_index=False
```

Penggabungan Series/Dataframe: 4. Merge (multi index)

- Cara mengatasinya adalah dengan **mereset index pada kedua dataframe**, kemudian merge akan mendeteksi common single/multi column di kedua dataframe dan melakukan merge,

```
15 # Merge dataframe yang memiliki multi index
16 df_merge = pd.merge(df1.reset_index(), df2.reset_index())
17 print('Merging dataframe:\n', df_merge)
```

	key	val2	val1	val3	val4
0	k1	30	200	1	6
1	k3	100	0	2	7
2	k5	10	100	3	8

Dataframe 1:		
	key	val1
	k1	30
	k2	50
	k3	100
	k4	20
	k5	10

Dataframe 2:		
	key	val3
	k1	1
	k3	2
	k5	3
	k7	4
	k10	5

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 }).set_index(['key', 'val2'])
8 print('Dataframe 1:\n', df1)
9 df2 = pd.DataFrame({
10     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
11     'val3': [1, 2, 3, 4, 5],
12     'val4': [6, 7, 8, 8, 10]
13 }).set_index(['key', 'val3'])
14 print('Dataframe 2:\n', df2)
```

Tugas 4. Penggabungan Series/Dataframe: Merge (multi index)

- Pastikan terlebih dahulu bahwa jika kedua dataframe yang akan kamu merge sudah multi index, maka harus dijadikan index tunggal terlebih dahulu. Setelah itu baru digunakan `pd.merge`.
- Syntaxnya secara umum masih sama dengan merge index tunggal, yaitu `pd.merge(left=__, right=__, how=__, left_on=__, right_on=__)`.

```
import __

# Buat dataframe df1 dan df2
df1 = pd.DataFrame({
    'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
    'val1': [200, 500, 0, 500, 100],
    'val2': [30, 50, 100, 20, 10]
}).set_index(__)
print('Dataframe 1:\n', __)

df2 = pd.DataFrame({
    'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
    'val3': [1, 2, 3, 4, 5],
    'val4': [6, 7, 8, 8, 10]
}).set_index(__)
print('Dataframe 2:\n', __)

# Merge dataframe yang memiliki multi index
df_merge = __
print('Merging dataframe:\n', __)
```

Penggabungan Series/Dataframe: 5. Join

- Method **.join()** apabila digunakan pada **dataframe** untuk menggabungkan kedua data dengan set index pada kedua tabel tersebut sebagai join key, tanpa index, hal ini **tidak akan berhasil**.

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 })
8 df2 = pd.DataFrame({
9     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
10    'val3': [1, 2, 3, 4, 5],
11    'val4': [6, 7, 8, 8, 10]
12 })
13 # Penerapan join dengan menggunakan index default di dataframe
14 df1.join(df2)
```

```
Traceback (most recent call last):
  File "<stdin>", line 16, in <module>
  File "/usr/local/lib/python3.5/site-packages/pandas/core/frame.py", line 6815, in join
    rsuffix=rsuffix, sort=sort)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/frame.py", line 6830, in _join_com
    suffixes=(lsuffix, rsuffix), sort=sort)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/merge.py", line 48, in mer
    return op.get_result()
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/merge.py", line 552, in ge
    rdata.items, rsuf)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/internals/managers.py", line 1972,
    '{rename}'.format(rename=to_rename))
ValueError: columns overlap but no suffix specified: Index(['key'], dtype='object')
```

- Terdapat error berupa:

ValueError: columns overlap but no suffix specified: Index(['key'], dtype='object')

Penggabungan Series/Dataframe: 5. Join

Solusi: secara **default**, fungsi join ini akan mengeksekusi **left join**

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 })
8 df2 = pd.DataFrame({
9     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
10    'val3': [1, 2, 3, 4, 5],
11    'val4': [6, 7, 8, 8, 10]
12 })
13 # Penerapan join dengan menggunakan set_index default 'key'
14 print(df1.set_index('key').join(df2.set_index('key')))
```

	val1	val2	val3	val4
key				
k1	200	30	1.0	6.0
k2	500	50	NaN	NaN
k3	0	100	2.0	7.0
k4	500	20	NaN	NaN
k5	100	10	3.0	8.0

Penggabungan Series/Dataframe: 5. Join

Untuk tipe join yang lain (contoh=inner), harus menspesifikasikan keyword `how='inner'` seperti yang dicontohkan berikut ini

```
1 import pandas as pd
2 # Buat dataframe df1 dan df2
3 df1 = pd.DataFrame({
4     'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
5     'val1': [200, 500, 0, 500, 100],
6     'val2': [30, 50, 100, 20, 10]
7 })
8 df2 = pd.DataFrame({
9     'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
10    'val3': [1, 2, 3, 4, 5],
11    'val4': [6, 7, 8, 8, 10]
12 })
13 # Penerapan join dengan menggunakan set_index dan keyword how
14 join_df = df1.set_index('key').join(df2.set_index('key'), how='inner')
15 print(join_df)
```

	val1	val2	val3	val4
key				
k1	200	30	1	6
k3	0	100	2	7
k5	100	10	3	8

Tugas 5. Penggabungan Series/Dataframe: Join

- Lakukanlah seperti yang dicontohkan tetapi penggabungan dengan method join berupa outer join.
- Sebelum mengaplikasikan method .join() silakan di set terlebih dahulu index dari kedua dataframe yang akan di-join.
- Langkah ini ekuivalen dengan pd.merge(..., ..., ..., left_on=__, right_on=__).

	val1	val2	val3	val4
key				
k1	200.0	30.0	1.0	6.0
k10	NaN	NaN	5.0	10.0
k2	500.0	50.0	NaN	NaN
k3	0.0	100.0	2.0	7.0
k4	500.0	20.0	NaN	NaN
k5	100.0	10.0	3.0	8.0
k7	NaN	NaN	4.0	8.0

```
import __

# Buat dataframe df1 dan df2
df1 = pd.DataFrame({
    'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
    'val1': [200, 500, 0, 500, 100],
    'val2': [30, 50, 100, 20, 10]
})
df2 = pd.DataFrame({
    'key': ['k1', 'k3', 'k5', 'k7', 'k10'],
    'val3': [1, 2, 3, 4, 5],
    'val4': [6, 7, 8, 8, 10]
})

# Penerapan join dengan menggunakan set_index dan
keyword how
join_df = __
print(__)
```

Tugas 6. Penggabungan Series/Dataframe: Merge

Diberikan dataframe sebagai berikut

```
df1 = pd.DataFrame({
    'key': ['k1', 'k2', 'k3', 'k4', 'k5'],
    'val1': [200, 500, 0, 500, 100],
    'val2': [30, 50, 100, 20, 10],
})

df2 = pd.DataFrame({
    'key': ['k1', 'k1', 'k5', 'k7', 'k10'],
    'val3': [1, 2, 3, 4, 5],
    'val4': [6, 7, 8, 8, 10]
})
```

Apa yang akan dihasilkan oleh code ini:

```
pd.merge(df1, df2, validate="1:1")
```

- 1. Penggabungan Series/Dataframe**
- 2. Pivot, Melt, Stack & Unstack**

Pivot, Melt, Stack & Unstack

mempelajari bagaimana cara memformat ulang dataset dengan teknik pivot, melt, stack dan unstack

Pivot, Melt, Stack & Unstack

- Melakukan format ulang pada dataset itu sangatlah penting, biasanya hal ini dilakukan untuk mengetahui keseluruhan data secara cepat dengan chart atau visualisasi.
- Di Pandas, ada beberapa teknik untuk melakukan pivot atau unpivot (**melt**) di Pandas
- Terdapat pula konsep **stack** yang artinya menumpuk data dengan kolom yang lebih sedikit (stack) sama seperti konsep melt dan ada pula yang memperluas data dengan kolom yang lebih banyak (unstack) sama seperti konsep **pivot**.

Row index in the new table

Columns in the new table

Cell values in the new table

<u>ix</u>	<u>Item</u>	<u>CType</u>	<u>USD</u>	<u>EU</u>
<u>0</u>	Item0	Gold	1\$	1€
<u>1</u>	Item0	Bronze	2\$	2€
<u>2</u>	Item1	Gold	3\$	3€
<u>3</u>	Item1	Silver	4\$	4€

<u>ix=Item</u>	<u>Bronze</u>	<u>Gold</u>	<u>Silver</u>
<u>Item0</u>	2\$	1\$	NaN
<u>Item1</u>	NaN	3\$	4\$

```
d.pivot(index='Item', columns='CType', values='USD')
```

<https://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/>



<https://morioh.com/p/17278219952b>

Persiapan Dataset

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 print('Data:\n', data)
10 # Ukuran dataframe
11 print('Data terdiri atas %d baris dan %d kolom.' % data.shape)
```

Data:

	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Data terdiri atas 12 baris dan 4 kolom.

Tugas 7. Persiapan dataset untuk operasi Pivot, Melt, Stack & Unstack

- Carilah unique records/value pada keempat kolom dataframe 'data'.
- Untuk mencari unique value dari setiap kolom, lakukan pengindex-an dataframe dengan nama kolomnya sehingga menjadi series dan kemudian terapkan method .unique().
- Tempatkan ke dalam looping perulangan untuk sejumlah kolom yang dimiliki oleh dataframe.
- Untuk mengambil nama-nama kolom gunakan attribut .columns pada dataframe.

```
import __
```

```
# Dataframe
```

```
data = pd.DataFrame({  
    'kelas': 6*['A'] + 6*['B'],  
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],  
    'pelajaran': 6*['math','english'],  
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]  
}, columns=__)
```

```
# Unique value pada setiap kolom data
```

```
__ in __.__:  
    print('Unique value %s: %s' % (column, data[__].__))
```

```
Unique value kelas: ['A' 'B']  
Unique value murid: ['A1' 'A2' 'A3' 'B1' 'B2' 'B3']  
Unique value pelajaran: ['math' 'english']  
Unique value nilai: [ 90  60  70  85  50 100  40  95  80  45]
```

Python | Pandas.pivot()

- `pandas.pivot(index, columns, values)` function produces pivot table based on 3 columns of the DataFrame.
- Uses unique values from index / columns and fills with values.
- **Parameters:**
 - **index[ndarray]** : Labels to use to make new frame's index
 - **columns[ndarray]** : Labels to use to make new frame's columns
 - **values[ndarray]** : Values to use for populating new frame's values
- **Returns:** Reshaped DataFrame
- **Exception:** `ValueError` raised if there are any duplicates.

Bentuk Umum

`p = d.pivot(index='column1', columns='column2', values='column3')`

Pivot

- Untuk menerapkan method **.pivot()** pada dataframe dapat dilakukan pada dataframe yang memiliki index tunggal ataupun memiliki index multiindex.

Data:

	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Data terdiri atas 12 baris dan 4 kolom.

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 print('Data:\n', data)
10 # Ukuran dataframe
11 print('Data terdiri atas %d baris dan %d kolom.' % data.shape)
```

Unique value kelas: ['A' 'B']
Unique value murid: ['A1' 'A2' 'A3' 'B1' 'B2' 'B3']
Unique value pelajaran: ['math' 'english']
Unique value nilai: [90 60 70 85 50 100 40 95 80 45]

- **Pivoting** dengan **single column** measurement

Pivoting with single column measurement:

	english	math
A1	60	90
A2	85	70
A3	60	50
B1	40	100
B2	80	95
B3	45	60

```
9 # Pivoting with single column measurement
10 pivot1 = data.pivot(index='murid',columns='pelajaran',values='nilai')
11 print('Pivoting with single column measurement:\n', pivot1)
```

Pivot

- The [pivot](#) function is used to create a new derived table out of a given one.
 - Pivot takes 3 arguments with the following names:
 - index
 - columns
 - values
- Bentuk Umum**
`p = d.pivot(index='column1', columns='column2', values='column3')`
- As a value for each of these parameters you need to specify a column name in the original table.
 - Then the pivot function will create a new table, whose row and column indices are the unique values of the respective parameters.
 - The cell values of the new table are taken from column given as the [values](#) parameter.

```
p = d.pivot(index='Item', columns='CType', values='USD')
```

<u>ix</u>	<u>Item</u>	<u>CType</u>	<u>USD</u>	<u>EU</u>
<u>0</u>	Item0	Gold	1\$	1€
<u>1</u>	Item0	Bronze	2\$	2€
<u>2</u>	Item1	Gold	3\$	3€
<u>3</u>	Item1	Silver	4\$	4€

<u>ix=Item</u>	<u>Bronze</u>	<u>Gold</u>	<u>Silver</u>
<u>Item0</u>	2\$	1\$	NaN
<u>Item1</u>	NaN	3\$	4\$

```
d.pivot(index='Item', columns='CType', values='USD')
```

```

9 # Pivoting with single column measurement
10 pivot1 = data.pivot(index='murid', columns='pelajaran', values='nilai')
11 print('Pivoting with single column measurement:\n', pivot1)

```

Pivot: multiple column

- **Pivoting dengan multiple column measurement**

```

12 # Pivoting with multiple column measurement
13 pivot2 = data.pivot(index='murid', columns='pelajaran')
14 print('Pivoting with multiple column measurement:\n', pivot2)

```

Pivoting with multiple column measurement:

	kelas		nilai	
	english	math	english	math
A1	60	90		
A2	85	70		
A3	60	50		
B1	40	100		
B2	80	95		
B3	45	60		

```

Unique value kelas: ['A' 'B']
Unique value murid: ['A1' 'A2' 'A3' 'B1' 'B2' 'B3']
Unique value pelajaran: ['math' 'english']
Unique value nilai: [ 90  60  70  85  50 100  40  95  80  45]

```

Pivoting with single column measurement:

	english	math
A1	60	90
A2	85	70
A3	60	50
B1	40	100
B2	80	95
B3	45	60

Data:

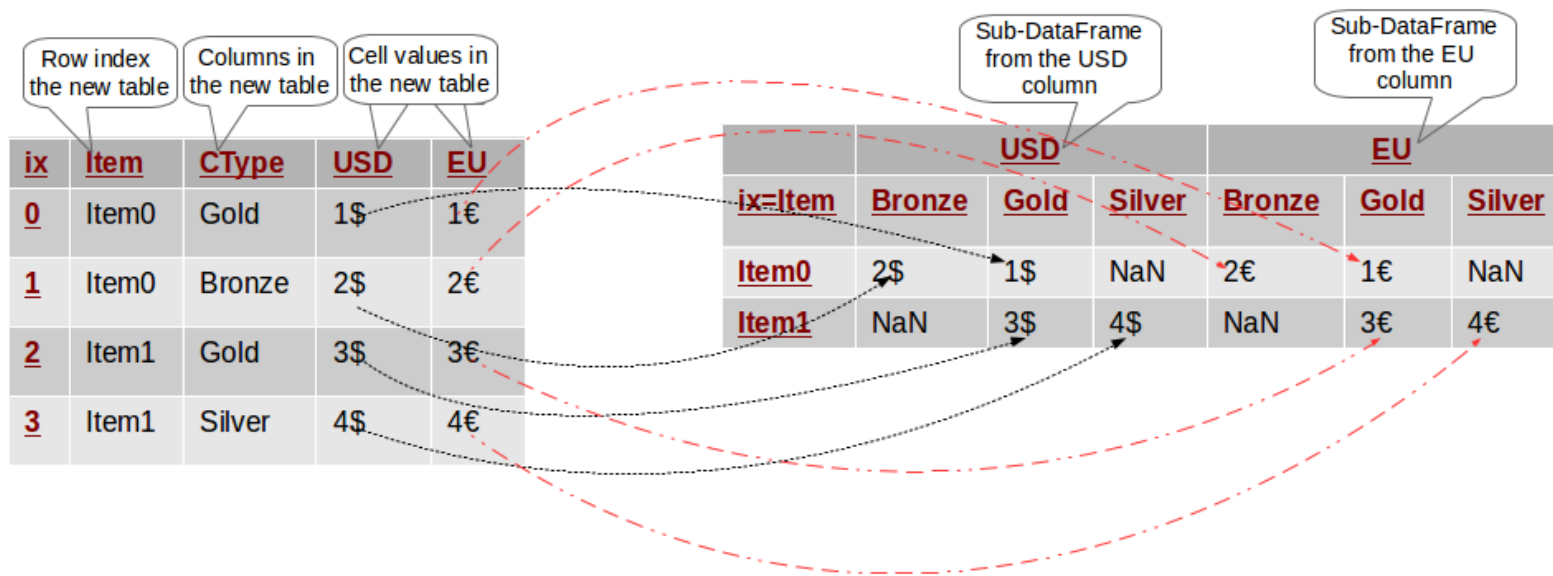
	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Data terdiri atas 12 baris dan 4 kolom.

- Apa yang **berbeda** dari kedua code di atas? Pada code pertama di **spesifikasikan values** mana yang akan dilakukan pivot sedangkan **di code kedua, values tidak spesifik** mana yang akan dilakukan pivot
- Maka Pandas secara default memperlakukan **kolom yang ada selain** yang telah di spesifikasikan sebagai index dan columns, sebagai values.

Pivoting By Multiple Columns

```
p = d.pivot(index='Item', columns='CType')
```



```
d.pivot(index='Item', columns='CType')
```

Tugas 8. Pivot single dan multiple column

- Memahami konsep pivoting.
- Perbedaan untuk membentuk dataframe hasil pivoting adalah ada atau tidaknya keyword values yang dikirimkan/diset ke dalam method .pivot().

```
import ____
# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=['kelas','murid','pelajaran','nilai'])
# Pivoting with single column measurement
____ = ____
print('Pivoting with single column measurement:\n', ____)
# Pivoting with multiple column measurement
____ = ____
print('Pivoting with multiple column measurement:\n', ____)
```

Pivot_table

- Apa yang terjadi jika output pivot tabel memiliki duplicate index?
- Seperti yang diketahui, index di dataframe adalah unique identifier untuk setiap row.
- Jadi index tidak boleh memiliki duplikat .
- Setiap kali membuat pivot tabel, harus menspesifikasikan index diset dengan kolom yang mana dan columns diset dengan kolom yang mana.

Data:

	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Data terdiri atas 12 baris dan 4 kolom.

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 # Creating pivot and assign pivot_tab
10 pivot_tab = data.pivot(index='kelas',columns='pelajaran',values='nilai')
11 print('Creating pivot table:\n', pivot_tab)
```

```
Traceback (most recent call last):
  File "<stdin>", line 12, in <module>
  File "/usr/local/lib/python3.5/site-packages/pandas/core/frame.py", line 5628, in pivot
    return pivot(self, index=index, columns=columns, values=values)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/pivot.py", line 388, in pi
    return indexed.unstack(columns)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/series.py", line 3301, in unstack
    return unstack(self, level, fill_value)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/reshape.py", line 396, in
    constructor=obj._constructor_expanddim)
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/reshape.py", line 128, in
    self._make_selectors())
  File "/usr/local/lib/python3.5/site-packages/pandas/core/reshape/reshape.py", line 166, in
    raise ValueError('Index contains duplicate entries, '
ValueError: Index contains duplicate entries, cannot reshape
```

Pivot_table

Common Mistake in Pivoting

As we saw the pivot method takes at least 2 column names as parameters - the index and the columns named parameters.

What will happen if we have multiple rows with the same values for these columns?

How will the pivot method determine the value of the corresponding cell in the pivoted table?

The following diagram depicts the problem:

<u>ix</u>	<u>Item</u>	<u>CType</u>	<u>USD</u>	<u>EU</u>
<u>0</u>	<u>Item0</u>	<u>Gold</u>	1\$	1€
<u>1</u>	Item0	Bronze	2\$	2€
<u>2</u>	<u>Item0</u>	<u>Gold</u>	3\$	3€
<u>3</u>	Item1	Silver	4\$	4€

<u>ix=Item</u>	<u>Bronze</u>	<u>Gold</u>	<u>Silver</u>
<u>Item0</u>	2\$	1 or 3\$?	NaN
<u>Item1</u>	NaN	NaN	4\$

```
d.pivot(index='Item', columns='CType', values='USD')
```

Pivot_table

- The pivot_table method comes to solve this problem.
- It works like pivot, but it aggregates the values from rows with duplicate entries for the specified columns.
- In other words, in the previous example we could have used the **mean**, the **median** or **another aggregation function** to compute a single value from the conflicting entries.

<u>ix</u>	<u>Item</u>	<u>CType</u>	<u>USD</u>	<u>EU</u>
<u>0</u>	Item0	Gold	1	1
<u>1</u>	Item0	Bronze	2	2
<u>2</u>	Item0	Gold	3	3
<u>3</u>	Item1	Silver	4	4

<u>ix=Item</u>	<u>Bronze</u>	<u>Gold</u>	<u>Silver</u>
<u>Item0</u>	2	2 = mean(1,3)	NaN
<u>Item1</u>	NaN	NaN	4

```
d.pivot_table(index='Item', columns='CType', values='USD', aggfunc=np.mean)
```

Python | Pandas.pivot_table()

- Syntax to create a spreadsheet-style pivot table as a DataFrame:

```
pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All')
```

- Levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.
- **Parameters:**
 - **data** : DataFrame
 - **values** : column to aggregate, optional
 - **index**: column, Grouper, array, or list of the previous
 - **columns**: column, Grouper, array, or list of the previous
 - **aggfunc**: function, list of functions, dict, default numpy.mean
 - > If list of functions passed, the resulting pivot table will have hierarchical columns whose top level are the function names.
 - > If dict is passed, the key is column to aggregate and value is function or list of functions
 - **fill_value[scalar, default None]** : Value to replace missing values with
 - **margins[boolean, default False]** : Add all row / columns (e.g. for subtotal / grand totals)
 - **dropna[boolean, default True]** : Do not include columns whose entries are all NaN
 - **margins_name[string, default 'All']** : Name of the row / column that will contain the totals when margins is True.
- **Returns:** DataFrame

Python | Pandas.pivot_table()

- Hal ini dapat diatasi dengan melakukan method **.pivot_table()** pada dataframe.
- Metode ini sama seperti melakukan pivot pada tabel tapi juga melakukan **groupby** dan **aggregation** (aggfunc) pada level rows sehingga dipastikan tidak ada duplicate index di rows (secara default aggfunc = 'mean').

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 # Creating pivot and assign pivot_tab dengan menggunakan keyword aggfunc
10 pivot_tab =
11     data.pivot_table(index='kelas',columns='pelajaran',values='nilai',aggfunc=sum)
12 print('Creating pivot table:\n', pivot_tab)
```

Creating pivot table:

	pelajaran	english	math
kelas			
A		205	210
B		165	255

- Keyword **aggfunc** yang digunakan pada method **.pivot_table()** dapat menggunakan nilai berikut:
 - sum
 - 'mean'
 - 'median'

Tugas 9. Python | Pandas.pivot_table()

- Seperti yang dicontohkan untuk mengcreate pivot tabel dengan method .pivot_table() tetapi aggfunc yang digunakan adalah 'mean' dan 'median'.
- Pastikan kembali bahwa keyword aggfunc yang digunakan untuk membentuk dua pivot tabel adalah nilai rata-rata dan nilai median masing-masing kelas untuk kedua mata pelajaran.
- Pastikan untuk mean dan median harus dinyatakan sebagai tipe data string, yaitu aggfunc='mean' dan aggfunc='median'.

```
Creating pivot table -- aggfunc mean:
pelajaran  english  math
kelas
A           68.333333  70.0
B           55.000000  85.0
Creating pivot table -- aggfunc median:
pelajaran  english  math
kelas
A           60      70
B           45      95
```

```
import __

# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=__)

# Creating pivot and assign pivot_tab dengan menggunakan keyword
aggfunc='mean'
pivot_tab_mean = __
print('Creating pivot table -- aggfunc mean:\n', __)

# Creating pivot and assign pivot_tab dengan menggunakan keyword
aggfunc='median'
pivot_tab_median = __
print('Creating pivot table -- aggfunc median:\n', __)
```

Python | Pandas.melt()

- To make analysis of data in table easier, we can reshape the data into a more computer-friendly form using Pandas in Python.
- Pandas.melt() is one of the function to do so.
- Pandas.melt() unpivots a DataFrame **from wide format to long format**.
- **melt()** function is useful to massage a DataFrame into a format where:
 - one or more columns are identifier variables,
 - while all other columns, considered measured variables, are unpivoted to the row axis,
 - leaving just two non-identifier columns, variable and value.

Python | Pandas.melt()

- **Syntax :**

`pandas.melt(frame, id_vars=None, value_vars=None, var_name=None, value_name='value', col_level=None)`

- **Parameters:**

- **frame** : DataFrame
- **id_vars[tuple, list, or ndarray, optional]** : Column(s) to use as identifier variables.
- **value_vars[tuple, list, or ndarray, optional]**: Column(s) to unpivot. If not specified, uses all columns that are not set as id_vars.
- **var_name[scalar]**: Name to use for the 'variable' column. If None it uses frame.columns.name or 'variable'.
- **value_name[scalar, default 'value']**: Name to use for the 'value' column.
- **col_level[int or string, optional]**: If columns are a MultiIndex then use this level to melt.

- **Returns:** DataFrame into a format where one or more columns are identifier variables

Python | Pandas.melt()

Pandas Melt: Reshape with Identifiers

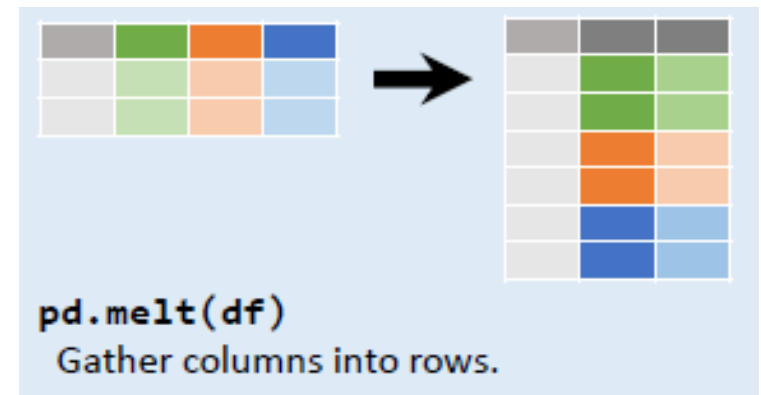
	pID	C1	C2	C3
0	h	13	15	12
1	n	16	16	16
2	w	13	22	18

**Wide
Dataframe**

→

	pID	Sample	Count
0	h	C1	13
1	n	C1	16
2	w	C1	13
3	h	C2	15
4	n	C2	16
5	w	C2	22
6	h	C3	12
7	n	C3	16
8	w	C3	18

**Tidy/Long
Dataframe**



wide			
id	x	y	z
1	a	c	e
2	b	d	f

long		
id	key	val
1	x	a
2	x	b
1	y	c
2	y	d
1	z	e
2	z	f

```
# Create a simple dataframe
```

```
# importing pandas as pd  
import pandas as pd
```

```
# creating a dataframe
```

```
df = pd.DataFrame({'Name': {0: 'John', 1: 'Bob', 2: 'Shiela'},  
                  'Course': {0: 'Masters', 1: 'Graduate', 2: 'Graduate'},  
                  'Age': {0: 27, 1: 23, 2: 21}})
```

```
df
```

```
# Name is id_vars and Course is value_vars
```

```
pd.melt(df, id_vars = ['Name'], value_vars = ['Course'])
```

```
# multiple unpivot columns
```

```
pd.melt(df, id_vars = ['Name'], value_vars = ['Course', 'Age'])
```

```
# Names of 'variable' and 'value' columns can be customized
```

```
pd.melt(df, id_vars = ['Name'], value_vars = ['Course'],  
        var_name = 'ChangedVarname', value_name = 'ChangedValname')
```

Out[15]:

	Name	ChangedVarname	ChangedValname
0	John	Course	Masters
1	Bob	Course	Graduate
2	Shiela	Course	Graduate

Out[2]:

	Name	Course	Age
0	John	Masters	27
1	Bob	Graduate	23
2	Shiela	Graduate	21

Out[13]:

	Name	variable	value
0	John	Course	Masters
1	Bob	Course	Graduate
2	Shiela	Course	Graduate

Out[14]:

	Name	variable	value
0	John	Course	Masters
1	Bob	Course	Graduate
2	Shiela	Course	Graduate
3	John	Age	27
4	Bob	Age	23
5	Shiela	Age	21

Melt

df3

	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150



df3.melt(id_vars=['first', 'last'])

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

Reshaping pandas dataframe with pd.melt (wide to long form)

Melt 1

	student	school	english	math	physics
0	Andy	Z	10	20	30
1	Bernie	Y	100	200	300
2	Cindy	Z	1000	2000	3000
3	Deb	Y	10000	20000	30000

```
pd.melt(frame=df_wide,
        id_vars=["student", "school"],
        var_name="cLaSs",
        value_name="gRaDe")
```



	student	school	cLaSs	gRaDe
0	Andy	Z	english	10
1	Bernie	Y	english	100
2	Cindy	Z	english	1000
3	Deb	Y	english	10000
4	Andy	Z	math	20
5	Bernie	Y	math	200
6	Cindy	Z	math	2000
7	Deb	Y	math	20000
8	Andy	Z	physics	30
9	Bernie	Y	physics	300
10	Cindy	Z	physics	3000
11	Deb	Y	physics	30000

Create wide dataframe

```
df_wide = pd.DataFrame(
    {"student": ["Andy", "Bernie", "Cindy", "Deb"],
     "school": ["Z", "Y", "Z", "Y"],
     "english": [10, 100, 1000, 10000], # eng grades
     "math": [20, 200, 2000, 20000], # math grades
     "physics": [30, 300, 3000, 30000] # physics grades
    })
```

Melt 2

	student	school	english	math	physics
0	Andy	Z	10	20	30
1	Bernie	Y	100	200	300
2	Cindy	Z	1000	2000	3000
3	Deb	Y	10000	20000	30000

```
pd.melt(frame=df_wide,
        id_vars="student",
        value_vars=["english", "math"],
        var_name="cLaSs",
        value_name="gRaDe")
```



	student	cLaSs	gRaDe
0	Andy	english	10
1	Bernie	english	100
2	Cindy	english	1000
3	Deb	english	10000
4	Andy	math	20
5	Bernie	math	200
6	Cindy	math	2000
7	Deb	math	20000

Melt 3

	student	school	english	math	physics
0	Andy	Z	10	20	30
1	Bernie	Y	100	200	300
2	Cindy	Z	1000	2000	3000
3	Deb	Y	10000	20000	30000

```
pd.melt(frame=df_wide,
        id_vars="student",
        var_name="cLaSs",
        value_name="gRaDe")
```



	student	cLaSs	gRaDe
0	Andy	school	Z
1	Bernie	school	Y
2	Cindy	school	Z
3	Deb	school	Y
4	Andy	english	10
5	Bernie	english	100
6	Cindy	english	1000
7	Deb	english	10000
8	Andy	math	20
9	Bernie	math	200
10	Cindy	math	2000
11	Deb	math	20000
12	Andy	physics	30
13	Bernie	physics	300
14	Cindy	physics	3000
15	Deb	physics	30000

Melt (1)

- Teknik melt melalui **pd.melt()** digunakan untuk mengembalikan kondisi data yang sudah dilakukan pivot menjadi sebelum pivot.
- Perhatikan kembali dataframe yang telah digunakan sebelumnya dan dataframanya sudah dipivot.

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 # Pivoting dataframe
10 data_pivot = data.pivot_table(index='kelas',columns='pelajaran',values='nilai',
11                               aggfunc='mean').reset_index()
11 print('Pivoting dataframe:\n', data_pivot)
```

Data:

	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Data terdiri atas 12 baris dan 4 kolom.

dengan bentuk dataframe dari output baris ke-11

```
Pivoting dataframe:
pelajaran kelas    english  math
0          A  68.333333  70.0
1          B  55.000000  85.0
```

Melt (1)

Pivoting dataframe:

	pelajaran	kelas	english	math
0		A	68.333333	70.0
1		B	55.000000	85.0

Data:

	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Data terdiri atas 12 baris dan 4 kolom.

Melakukan teknik melting pada dataframe output yang telah didapat sebelumnya.

[1] Melting dataframe

```
12 # [1] Melting dataframe data_pivot
13 data_melt_1 = pd.melt(data_pivot)
14 print('Melting dataframe:\n', data_melt_1)
```

Melting dataframe:

	pelajaran	value
0	kelas	A
1	kelas	B
2	english	68.3333
3	english	55
4	math	70
5	math	85

[2] Dengan menspesifikasi keyword argument **id_vars** yang ditujukan untuk membuat fix kolom yang berperan sebagai id tiap barisnya

```
15 # [2] Melting dataframe data_pivot dengan id_vars
16 data_melt_2 = pd.melt(data_pivot, id_vars='kelas')
17 print('Melting dataframe dengan idvars:\n', data_melt_2)
```

Melting dataframe dengan idvars:

	kelas	pelajaran	value
0	A	english	68.333333
1	B	english	55.000000
2	A	math	70.000000
3	B	math	85.000000

Tugas 10. Melt (1)

- Melting suatu dataframe hanya dapat dilakukan pada dataframe yang telah dipivot.
- Lakukan pivot terlebih dahulu data dengan index='kelas', columns='pelajaran', values='nilai', aggfunc='mean' dan kemudian reset index-nya.
- Lakukanlah melting secara default dan dengan menggunakan id_vars seperti yang telah dicontohkan.

```
import __

# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=['kelas','murid','pelajaran','nilai'])

# Pivoting dataframe
data_pivot = __
print('Pivoting dataframe:\n', __)

# [1] Melting dataframe data_pivot
data_melt_1 = __
print('Melting dataframe:\n', __)

# [2] Melting dataframe data_pivot dengan id_vars
data_melt_2 = __
print('Melting dataframe dengan idvars:\n', __)
```

Melt (2)

- Dengan menggunakan kembali dataframe yang telah diperoleh melalui pivoting.

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 # Pivoting dataframe
10 data_pivot = data.pivot_table(index='kelas',columns='pelajaran',values='nilai',
11                               aggfunc='mean').reset_index()
11 print('Pivoting dataframe:\n', data_pivot)
```

- dengan bentuk dataframe dari output baris ke-11

```
Pivoting dataframe:
  pelajaran kelas  english  math
0          A    68.333333  70.0
1          B    55.000000  85.0
```

Melt (2)

```
Pivoting dataframe:
pelajaran kelas    english  math
0           A  68.333333  70.0
1           B  55.000000  85.0
```

Lanjutkan dengan melakukan teknik melting pada dataframe output di atas untuk keyword argumen lainnya, yaitu

[3] Dengan menspesifikasikan keyword argument **value_vars** yang digunakan untuk **menampilkan variasi value apa saja yang perlu dimunculkan di kolom variable**.

```
12 # [3.a] Melting dataframe data_pivot dengan value_vars
13 data_melt_3a = pd.melt(data_pivot, value_vars=['math'])
14 print('Melting dataframe dengan value_vars:\n', data_melt_3a)
15 # [3.b] Melting dataframe data_pivot dengan id_vars dan value_vars
16 data_melt_3b = pd.melt(data_pivot, id_vars='kelas', value_vars=['math'])
17 print('Melting dataframe dengan id_vars dan value_vars:\n', data_melt_3b)
```

```
Melting dataframe dengan value_vars:
```

```
  pelajaran  value
0      math  70.0
1      math  85.0
```

```
Melting dataframe dengan id_vars dan value_vars:
```

```
  kelas pelajaran  value
0     A      math  70.0
1     B      math  85.0
```

Melt (2)

[4] Dengan spesifikasikan keyword argument **var_name** dan **value_name** yang digunakan untuk menampilkan nama kolom untuk variable dan value

```
18 # [4] Melting dataframe data_pivot dengan id_vars, value_vars, var_name. dan value_name
19 data_melt_4 = pd.melt(data_pivot, id_vars='kelas', value_vars=['english', 'math'],
    var_name='pelajaran', value_name='nilai')
20 print('Melting dataframe dengan id_vars, value_vars, var_name. dan value_name:\n',
    data_melt_4)
```

```
Melting dataframe dengan id_vars, value_vars, var_name. dan value_name:
   kelas pelajaran  nilai
0     A  english  68.333333
1     B  english  55.000000
2     A    math   70.000000
3     B    math   85.000000
```

Tugas 11. Melt (2)

- Pastikan proses melting yang dilakukan dengan `pd.melt` menggunakan keyword argument seperti yang telah dicontohkan, yaitu `value_vars` dan `id_vars`.
- Secara umum `pd.melt` akan menerima keyword arguments berupa `id_vars`, `value_vars`, `var_name`, `value_name` seperti yang dinyatakan berikut
`pd.melt(nama_dataframe, id_vars=____, value_vars=____, var_name=____, value_name=____)`

```
import ____

# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=['kelas','murid','pelajaran','nilai'])

# Pivoting dataframe
data_pivot = data.pivot_table(index='kelas',columns='pelajaran',values='nilai',
aggfunc='mean').reset_index()
print('Pivoting dataframe:\n', ____)

# [3.a] Melting dataframe data_pivot dengan value_vars
data_melt_3a = ____
print('Melting dataframe dengan value_vars:\n', ____)

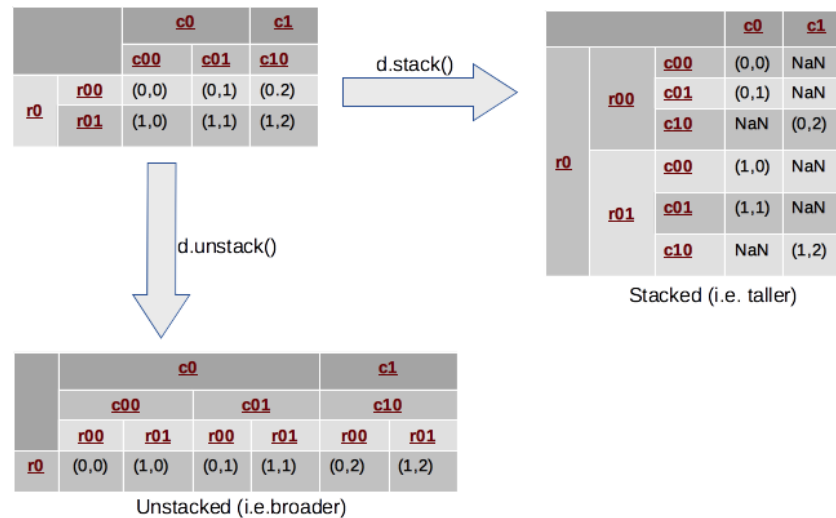
# [3.b] Melting dataframe data_pivot dengan id_vars dan value_vars
data_melt_3b = ____
print('Melting dataframe dengan id_vars dan value_vars:\n', ____)

# [4] Melting dataframe data_pivot dengan id_vars, value_vars, var_name. dan
value_name
data_melt_4 = ____
print('Melting dataframe dengan id_vars, value_vars, var_name. dan
value_name:\n', ____)
```

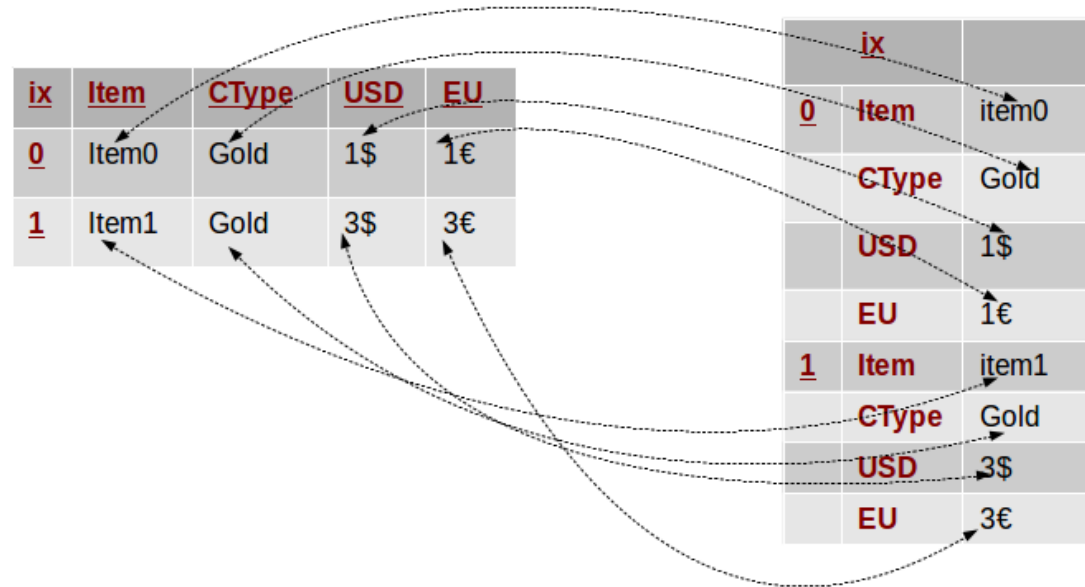
Stack & Unstack

Stack & Unstack (1)

- In fact **pivoting** a table is a **special case of stacking** a DataFrame.
- Let us assume we have a DataFrame with MultiIndices on the rows and columns.
- **Stacking** a DataFrame means moving (also rotating or pivoting) the innermost column index to become the innermost row index.
- The inverse operation is called **unstacking**. It means moving the innermost row index to become the innermost column index.



Stack and Unstack



d.stack() ↔ p.unstack()

- **Syntax:** *DataFrame.stack(level=- 1, dropna=True)*
- **Parameters –**
 - **level :** *It levels to stack from the column axis to the index axis. It either takes an int, string or list as input value. And by default is set to -1.*
 - **dropna :** *It asks whether to drop the rows into the resulting dataframe or series in case they don't have any values. It is of bool type and by default set to True.*
- **Returns** *a stacked DataFrame or series.*

Stack & Unstack (1)

- Konsep **stacking** dan **unstacking** sama dengan **melt** dan **pivot** secara **berurutan**, hanya saja tidak memasukkan index sebagai parameter di stack/unstack tapi harus set index terlebih dahulu, baru bisa melakukan stacking/unstacking dengan level yang bisa ditentukan sendiri.
- Perhatikan kembali dataframe berikut dengan multi index-nya.

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math','english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 print('Dataframe:\n', data)
10 # Set index data untuk kolom kelas, murid, dan pelajaran
11 data = data.set_index(['kelas','murid','pelajaran'])
12 print('Dataframe multi index:\n', data)
```

Dataframe:

	kelas	murid	pelajaran	nilai
0	A	A1	math	90
1	A	A1	english	60
2	A	A2	math	70
3	A	A2	english	85
4	A	A3	math	50
5	A	A3	english	60
6	B	B1	math	100
7	B	B1	english	40
8	B	B2	math	95
9	B	B2	english	80
10	B	B3	math	60
11	B	B3	english	45

Dataframe multi index:

	kelas	murid	pelajaran	nilai
A	A1	math		90
		english		60
	A2	math		70
		english		85
	A3	math		50
		english		60
B	B1	math		100
		english		40
	B2	math		95
		english		80
	B3	math		60
		english		45

Stack & Unstack (1)

kelas murid pelajaran			nilai
A	A1	math	90
		english	60
	A2	math	70
		english	85
	A3	math	50
		english	60
B	B1	math	100
		english	40
	B2	math	95
		english	80
	B3	math	60
		english	45

Mari terapkan bagaimana menggunakan teknik stacking dan unstacking ini pada dataframe multi index 'data':

- **[1] Unstacking dataframe**

```
13 # [1] Unstacking dataframe
14 data_unstack_1 = data.unstack()
15 print('Unstacking dataframe:\n', data_unstack_1)
```

Unstacking dataframe:

		nilai	
pelajaran		english	math
kelas murid			
A	A1	60	90
	A2	85	70
	A3	60	50
B	B1	40	100
	B2	80	95
	B3	45	60

- **[2] Unstacking dengan specify level name**

```
16 # [2] Unstacking dengan specify level name
17 data_unstack_2 = data.unstack(level='murid')
18 print('Unstacking dataframe dengan level name:\n', data_unstack_2)
```

Unstacking dataframe dengan level name:

		nilai					
murid		A1	A2	A3	B1	B2	B3
kelas pelajaran							
A	english	60.0	85.0	60.0	NaN	NaN	NaN
	math	90.0	70.0	50.0	NaN	NaN	NaN
B	english	NaN	NaN	NaN	40.0	80.0	45.0
	math	NaN	NaN	NaN	100.0	95.0	60.0

Stack & Unstack (1)

- **[3]** Unstacking dengan specify level position

```
19 # [3] Unstacking dengan specify level position
20 data_unstack_3 = data.unstack(level=1)
21 print('Unstacking dataframe dengan level position:\n', data_unstack_3)
```

Unstacking dataframe dengan level position:							
		nilai					
murid		A1	A2	A3	B1	B2	B3
kelas pelajaran							
A	english	60.0	85.0	60.0	NaN	NaN	NaN
	math	90.0	70.0	50.0	NaN	NaN	NaN
B	english	NaN	NaN	NaN	40.0	80.0	45.0
	math	NaN	NaN	NaN	100.0	95.0	60.0

Tugas 12. Stack & Unstack (1)

- Pastikan melakukan proses unstacking default, dengan menspesifikasi level name dan level position secara berurutan seperti yang dideskripsikan.
- Proses unstacking dapat diterapkan pada dataframe melalui method `.unstack()`.
- Jika tidak dispesifikasi keyword argument-nya maka proses unstacking dilakukan secara default.
- Keyword argument dapat dispesifikasi dengan level berupa string nama kolom atau posisi berupa integer level multiindexnya.

```
import __

# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=['kelas','murid','pelajaran','nilai'])
print('Dataframe:\n', data)

# Set index data untuk kolom kelas, murid, dan pelajaran
data = __
print('Dataframe multi index:\n', data)

# [1] Unstacking dataframe
data_unstack_1 = __
print('Unstacking dataframe:\n', __)

# [2] Unstacking dengan specify level name
data_unstack_2 = __
print('Unstacking dataframe dengan level name:\n', __)

# [3] Unstacking dengan specify level position
data_unstack_3 = __
print('Unstacking dataframe dengan level position:\n', __)
```

Stack & Unstack (2)

- Dalam bagian kedua dari Stack & Unstack ini akan dibahas stacking dataframe.
- Perhatikan dataframe berikut ini

```
1 import pandas as pd
2 # Dataframe
3 data = pd.DataFrame({
4     'kelas': 6*['A'] + 6*['B'],
5     'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
6     'pelajaran': 6*['math', 'english'],
7     'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
8 }, columns=['kelas','murid','pelajaran','nilai'])
9 data = data.set_index(['kelas','murid','pelajaran'])
10 data_unstack = data.unstack(level=1)
11 print('Dataframe:\n', data_unstack)
```

Dataframe:

		nilai					
murid		A1	A2	A3	B1	B2	B3
kelas	pelajaran						
A	english	60.0	85.0	60.0	NaN	NaN	NaN
	math	90.0	70.0	50.0	NaN	NaN	NaN
B	english	NaN	NaN	NaN	40.0	80.0	45.0
	math	NaN	NaN	NaN	100.0	95.0	60.0

Stack & Unstack (2)

- [1] Stacking dataframe

```
12 # [1] Stacking dataframe
13 data_stack = data_unstack.stack()
14 print('Stacked dataframe:\n', data_stack)
```

- [2] Tukar posisi index setelah stacking dataframe

```
15 # [2] Tukar posisi index setelah stacking dataframe
16 data_swap = data_stack.swaplevel(1,2)
17 print('Swapped data:\n', data_swap)
```

Stacked dataframe:

			nilai
kelas	pelajaran	murid	
A	english	A1	60.0
		A2	85.0
		A3	60.0
	math	A1	90.0
		A2	70.0
		A3	50.0
B	english	B1	40.0
		B2	80.0
		B3	45.0
	math	B1	100.0
		B2	95.0
		B3	60.0

Swapped data:

			nilai
kelas	murid	pelajaran	
A	A1	english	60.0
	A2	english	85.0
	A3	english	60.0
	A1	math	90.0
	A2	math	70.0
	A3	math	50.0
B	B1	english	40.0
	B2	english	80.0
	B3	english	45.0
	B1	math	100.0
	B2	math	95.0
	B3	math	60.0

Stack & Unstack (2)

- [3] Melakukan **sort_index** pada **stacking** dataframe

```
18 # [3] Melakukan sort_index pada stacking dataframe
19 data_sort = data_swap.sort_index()
20 print('Sorted data:\n', data_sort)
```

Sorted data:

			nilai
kelas	murid	pelajaran	
A	A1	english	60.0
		math	90.0
	A2	english	85.0
		math	70.0
	A3	english	60.0
		math	50.0
B	B1	english	40.0
		math	100.0
	B2	english	80.0
		math	95.0
	B3	english	45.0
		math	60.0

Tugas 13. Stack & Unstack (2)

- Stacking kembali data yang telah diunstacking melalui penerapan method `.stack()` pada dataframe `data_unstack`.
- Berdasarkan data stacking ini kamu swap level kolomnya dan kemudian lakukan sort berdasarkan index.
- Pastikan urutan yang dilakukan sesuai dengan yang diinstruksikan dan dicontohkan, yaitu stacking, swapping dan sorting.

```
import __

# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=['kelas','murid','pelajaran','nilai'])
data = data.set_index(['kelas','murid','pelajaran'])
data_unstack = data.unstack(level=1)
print('Dataframe:\n', data_unstack)

# [1] Stacking dataframe
data_stack = __
print('Stacked dataframe:\n', __)

# [2] Tukar posisi index setelah stacking dataframe
data_swap = __
print('Swapped data:\n', __)

# [3] Melakukan sort_index pada stacking dataframe
data_sort = __
print('Sorted data:\n', __)
```

Tugas 14. Stack & Unstack (2)

- Diberikan dataframe:

```
# Dataframe
data = pd.DataFrame({
    'kelas': 6*['A'] + 6*['B'],
    'murid': 2*['A1'] + 2*['A2'] + 2*['A3'] + 2*['B1'] + 2*['B2'] + 2*['B3'],
    'pelajaran': 6*['math','english'],
    'nilai': [90,60,70,85,50,60,100,40,95,80,60,45]
}, columns=['kelas','murid','pelajaran','nilai'])
```

- Bagaimana cara untuk menghasilkan output seperti di bawah ini:

	nilai					
kelas	A			B		
murid	A1	A2	A3	B1	B2	B3
	pelajaran					
english	60.0	85.0	60.0	40.0	80.0	45.0
math	90.0	70.0	50.0	100.0	95.0	60.0

Referensi

- Data Manipulation with Pandas - DQLab.
- Rebecca Vickery, Data Manipulation for Machine Learning with Pandas, <https://towardsdatascience.com/data-manipulation-for-machine-learning-with-pandas-ab23e79ba5de>, 2019.
- Soner Yıldırım, Reshaping Pandas DataFrames Melt, Stack and Pivot functions, <https://towardsdatascience.com/reshaping-pandas-dataframes-9812b3c1270e>, 2020.
- Praneel Nihar, Reshaping data using pandas, <https://medium.com/@u.praneel.nihar/reshaping-data-using-pandas-e2400be1ab2>, 2019.
- Reshaping and pivot tables, https://pandas.pydata.org/docs/user_guide/reshaping.html