



# **Session 20**

## **Advanced DataFrame**



# Table of Content

## What will We Learn Today?

1. Join Dataframe
2. Concat Dataframe
3. Append Dataframe
4. Indexing in DF
5. Pivoting Table
6. Melting Table
7. Lambda functions





# DataFrame

- DataFrame adalah struktur data dua dimensi, yaitu, data disejajarkan dalam mode tabular dalam baris dan kolom.
- Fitur DataFrame
  - Kolom berpotensi memiliki tipe berbeda
  - Ukuran – Bisa Diubah
  - Sumbu berlabel (baris dan kolom)
  - Dapat Melakukan operasi Aritmatika pada baris dan kolom



	Name	Age	Address	Job
1	Tom	28	Surabaya	Programmer
3	Jack	34	Jakarta	Data Scientist
2	Steve	29	Jakarta	Web developer
0	Ricky	42	Solo	Data Engineer



# Combining Data in Pandas

- Dengan Pandas, Kita dapat melakukan *merge*, *join*, dan *concatenate* untuk menyatukan DataFrame
- *merge()* untuk menggabungkan data pada kolom atau indeks tertentu
- *.join()* untuk menggabungkan data pada kolom atau indeks
- *concat()* untuk menggabungkan DataFrames di seluruh baris atau kolom







# Merging DataFrame

- Pandas menyediakan fungsi, **merge()**, untuk menggabungkan object-object DataFrame.

```
pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,
left_index=False, right_index=False, sort=True)
```

Parameter	Description
left	Object DataFrame
right	Object DataFrame yang lain
on	Kolom yg akan digabung. Harus ditemukan di objek DataFrame kiri dan kanan.
how	Diisi dengan salah satu dari : 'left', 'right', 'outer', 'inner'. Defaults : 'inner'.
sort	Mengurutkan 'join keys' berdasarkan alphabet. Defaults = False

```
import pandas as pd
kiri = pd.DataFrame({
    'Id':[1,2,3],
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat':['Jakarta', 'Surabaya', 'Medan']})
kanan = pd.DataFrame({
    'Id':[1,2,3],
    'Nama': ['Jane', 'Mike', 'Dave'],
    'Alamat':['Semarang', 'Yogyakarta', 'Solo']})

gabung = pd.merge(kiri, kanan, on='Id')
gabung
```

	Id	Nama_x	Alamat_x	Nama_y	Alamat_y
0	1	Budi	Jakarta	Jane	Semarang
1	2	Joko	Surabaya	Mike	Yogyakarta
2	3	Maya	Medan	Dave	Solo



# Merging DataFrame

- Menggabungkan dengan menggunakan argumen 'how'
- Argumen 'how' bisa diisi dengan 'left', 'right', 'inner', atau 'outer'

```
import pandas as pd
karyawan = pd.DataFrame({
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan'],
    'Id Pekerjaan': [1, 7, 2]})
pekerjaan = pd.DataFrame({
    'Id Pekerjaan': [2, 1, 4, 3],
    'Nama': ['Programmer', 'Data Engineer',
            'Manager', 'Web Developer'],
    'Gaji': [5000, 4000, 3000, 6000]})
print(karyawan)
print('-----')
print(pekerjaan)
```

	Nama	Alamat	Id Pekerjaan
0	Budi	Jakarta	1
1	Joko	Surabaya	7
2	Maya	Medan	2

-----

	Id Pekerjaan	Nama	Gaji
0	2	Programmer	5000
1	1	Data Engineer	4000
2	4	Manager	3000
3	3	Web Developer	6000

```
gabung_merge = pd.merge(karyawan, pekerjaan, on='Id Pekerjaan',
                        how='left', suffixes=('_pegawai', '_pekerjaan'), sort=True)
gabung_merge
```

	Nama_pegawai	Alamat	Id Pekerjaan	Nama_pekerjaan	Gaji
0	Budi	Jakarta	1	Programmer	5000.0
1	Joko	Surabaya	2	Data Engineer	4000.0
2	Maya	Medan	7	NaN	NaN

Silahkan buat kode baru dengan mengisi argument how dengan 'right', 'inner' atau 'outer'



# Joining DataFrame

- Object DataFrame menyediakan fungsi, **join()**, sehingga bisa digunakan untuk menggabungkan object DataFrame lain.

```
DataFrame.join(other, on=None, how='left', lsuffix='', rsuffix='', sort=False)
```

Parameter	Description
other	Object DataFrame lain
on	Kolom (nama) atau index yang akan digabung
how	Diisi dengan salah satu dari : 'left', 'right', 'outer', 'inner'. Defaults : 'left'.
sort	Mengurutkan 'join keys' berdasarkan alphabet. Defaults = False
lsuffix	Akhiran untuk digunakan dari kolom yg sama sebelah kiri.







# Joining DataFrame

```
[24] import pandas as pd
karyawan = pd.DataFrame({
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan'],
    'Id Pekerjaan': [1, 2, 7]})
pekerjaan = pd.DataFrame({
    'Id Pekerjaan': [1, 2, 3, 4],
    'Nama': ['Programmer', 'Data Engineer',
            'Manager', 'Web Developer'],
    'Gaji': [5000, 4000, 3000, 6000]})
print(karyawan)
print('-----')
print(pekerjaan)
```

	Nama	Alamat	Id Pekerjaan
0	Budi	Jakarta	1
1	Joko	Surabaya	2
2	Maya	Medan	7

-----

	Id Pekerjaan	Nama	Gaji
0	1	Programmer	5000
1	2	Data Engineer	4000
2	3	Manager	3000
3	4	Web Developer	6000

```
gabung_join = karyawan.join(pekerjaan, on='Id Pekerjaan',
                             how='inner', lsuffix=' pegawai')
gabung_join
```

	Nama pegawai	Alamat	Id Pekerjaan pegawai	Id Pekerjaan	Nama	Gaji
0	Budi	Jakarta	1	2	Data Engineer	4000
1	Joko	Surabaya	2	3	Manager	3000

Silahkan buat kode baru dengan mengisi argument how dengan 'right', 'left' atau 'outer'. Silahkan juga menggunakan argumen rsuffix.





# Concatenate DataFrame

- Pandas menyediakan fungsi, **concat()**, untuk menggabungkan object-object DataFrame, berdasarkan sumbu baris atau kolom.

```
pandas.concat(objs, axis=0, join='outer', ignore_index=False, keys=None, levels=None,
names=None, verify_integrity=False, sort=False, copy=True)
```

Parameter	Description
objs	Object-object DataFrame
axis	0=baris, 1= kolom
join	Diisi dengan salah satu dari : 'outer', 'inner'. Defaults : 'outer'.





# Concatenate DataFrame

- Pandas menyediakan fungsi, **concat()**, untuk menggabungkan object-object DataFrame, berdasarkan sumbu baris atau kolom.

```
import pandas as pd
kiri = pd.DataFrame({
    'Nama': ['Yudi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan']})
kanan = pd.DataFrame({
    'Nama': ['Jane', 'Mike', 'Dave'],
    'Alamat': ['Semarang', 'Yogyakarta', 'Solo']})

gabung_concat = pd.concat([kiri, kanan], axis=0, ignore_index=True)
gabung_concat
```

	Nama	Alamat
0	Yudi	Jakarta
1	Joko	Surabaya
2	Maya	Medan
3	Jane	Semarang
4	Mike	Yogyakarta
5	Dave	Solo

```
import pandas as pd
kiri = pd.DataFrame({
    'Nama': ['Yudi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan']})
kanan = pd.DataFrame({
    'Nama': ['Dian', 'Andi', 'Jack'],
    'Gaji': [5000, 4000, 3000]})

gabung_concat = pd.concat([kiri, kanan], axis=0,
                           ignore_index=True, join='outer')
gabung_concat
```

	Nama	Alamat	Gaji
0	Yudi	Jakarta	NaN
1	Joko	Surabaya	NaN
2	Maya	Medan	NaN
3	Dian	NaN	5000.0
4	Andi	NaN	4000.0
5	Jack	NaN	3000.0



# Concatenate DataFrame

- Axis = 1, menggabungkan kolom.

```
import pandas as pd
kiri = pd.DataFrame({
    'Nama': ['Yudi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan']})
kanan = pd.DataFrame({
    'Pekerjaan': ['Programmer', 'Data Engineer',
                 'Manager', 'Web Developer'],
    'Gaji': [5000, 4000, 3000, 6000]})

gabung_concat = pd.concat([kiri, kanan], axis=1, join='outer')
gabung_concat
```

	Nama	Alamat	Pekerjaan	Gaji
0	Yudi	Jakarta	Programmer	5000
1	Joko	Surabaya	Data Engineer	4000
2	Maya	Medan	Manager	3000
3	NaN	NaN	Web Developer	6000





# Append DataFrame

- Object DataFrame menyediakan fungsi, **append()**, sehingga bisa digunakan untuk menambahkan baris lainnya di akhir object.

```
DataFrame.append(other, ignore_index=False, verify_integrity=False, sort=False)
```

```
import pandas as pd
kiri = pd.DataFrame({
    'Nama': ['Yudi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan']})
kanan = pd.DataFrame({
    'Nama': ['Jane', 'Mike', 'Dave'],
    'Alamat': ['Semarang', 'Yogyakarta', 'Solo']})

gabung_append = kiri.append(kanan)
gabung_append
```

	Nama	Alamat
0	Yudi	Jakarta
1	Joko	Surabaya
2	Maya	Medan
0	Jane	Semarang
1	Mike	Yogyakarta
2	Dave	Solo



# Indexing in DataFrame (reset\_index)

- Ketika kita menggunakan *join*, *concat* atau *append* untuk object DataFrames, indeks akan berulang. Sehingga perlu untuk di-reset.

```
DataFrame.reset_index(level=None, drop=False, inplace=False, col_level=0, col_fill='')
```

```
import pandas as pd
kiri = pd.DataFrame({
    'Nama': ['Yudi', 'Joko'],
    'Alamat': ['Jakarta', 'Surabaya']})
kanan = pd.DataFrame({
    'Nama': ['Jane'],
    'Alamat': ['Semarang']})
gabung_append = kiri.append(kanan)
new=gabung_append.reset_index(drop=True)
new
```

	Nama	Alamat
0	Yudi	Jakarta
1	Joko	Surabaya
2	Jane	Semarang



# Indexing in DataFrame (set\_index)

- Kita bisa mengganti index DataFrame dengan kolom yang ada, menggunakan fungsi *set\_index*

```
DataFrame.set_index(keys, drop=True, append=False, inplace=False, verify_integrity=False)
```

```
[87] import pandas as pd
      karyawan = pd.DataFrame({
          'Emp_id' : ['001','003','005'],
          'Nama': ['Budi', 'Joko', 'Maya'],
          'Alamat':['Jakarta', 'Surabaya', 'Medan'],
          'Pekerjaan': ['Programmer', 'Data Engineer',
                        'Manager'],
          'Gaji':[5000, 4000, 3000]})
      print(karyawan)
```

	Emp_id	Nama	Alamat	Pekerjaan	Gaji
0	001	Budi	Jakarta	Programmer	5000
1	003	Joko	Surabaya	Data Engineer	4000
2	005	Maya	Medan	Manager	3000

```
karyawan.set_index('Emp_id', inplace=True)
karyawan
```

	Nama	Alamat	Pekerjaan	Gaji
Emp_id				
001	Budi	Jakarta	Programmer	5000
003	Joko	Surabaya	Data Engineer	4000
005	Maya	Medan	Manager	3000





# Pivoting Table

- Kita bisa membuat tabel pivot bergaya spreadsheet sebagai DataFrame, menggunakan fungsi ***pivot\_table***
- Tabel Pivot digunakan untuk meringkas, mengurutkan, mengatur ulang, mengelompokkan, menghitung, total, atau rata-rata data yang disimpan dalam tabel.

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

	Emp_id	Nama	Gender	Pekerjaan	Gaji
0	001	Budi	Pria	Programmer	5000
1	002	Joko	Pria	Data Engineer	4000
2	003	Maya	Perempuan	Programmer	5000
3	004	Andi	Pria	Manager	6000
4	005	Dewi	Perempuan	Manager	8000
5	006	Dian	Perempuan	Manager	6000



Gender	Perempuan		Pria		
Pekerjaan	Manager	Programmer	Data Engineer	Manager	Programmer
Gaji	7000	5000	4000	6000	5000
Pekerjaan	2	1	1	1	1



# Pivoting Table

```
import pandas as pd

karyawan = pd.DataFrame({
    'Emp_id' : ['001','002','003','004','005','006'],
    'Nama': ['Budi', 'Joko', 'Maya', 'Andi', 'Dewi', 'Dian'],
    'Gender':['Pria', 'Pria', 'Perempuan', 'Pria', 'Perempuan', 'Perempuan'],
    'Pekerjaan': ['Programmer', 'Data Engineer', 'Programmer', 'Manager',
                  'Manager', 'Manager'],
    'Gaji':[5000, 4000, 5000, 6000, 8000, 6000]})
karyawan
```

	Emp_id	Nama	Gender	Pekerjaan	Gaji
0	001	Budi	Pria	Programmer	5000
1	002	Joko	Pria	Data Engineer	4000
2	003	Maya	Perempuan	Programmer	5000
3	004	Andi	Pria	Manager	6000
4	005	Dewi	Perempuan	Manager	8000
5	006	Dian	Perempuan	Manager	6000

```
pivot_table = pd.pivot_table(karyawan, values=['Gaji'],
                               columns=['Gender','Pekerjaan'],
                               aggfunc={'Gaji': 'mean', 'Pekerjaan': 'count'})
pivot_table
```

Gender	Perempuan		Pria		
Pekerjaan	Manager	Programmer	Data Engineer	Manager	Programmer
Gaji	7000	5000	4000	6000	5000
Pekerjaan	2	1	1	1	1



# Melting Table

- Merubah Pivot DataFrame dari format lebar ke format panjang, menggunakan *melt*
- Artinya, satu atau lebih kolom digunakan sebagai pengidentifikasi dan semua kolom lainnya digunakan sebagai nilai.

```
DataFrame.melt(id_vars=None, value_vars=None, var_name=None, value_name='value',  
col_level=None, ignore_index=True)
```





# Melting Table

- Merubah Pivot DataFrame dari format lebar ke format panjang.

```
[144] import pandas as pd
mahasiswa = pd.DataFrame({
    'Mhs_id' : ['001','002','003'],
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Bhs Inggris':[100, 90, 98],
    'Matematika': [90, 87, 72],
    'Ekonomi':[90, 80, 78]})
mahasiswa
```

	Mhs_id	Nama	Bhs Inggris	Matematika	Ekonomi
0	001	Budi	100	90	90
1	002	Joko	90	87	80
2	003	Maya	98	72	78

```
0s ▶ melt_table = pd.melt(mahasiswa, id_vars=['Mhs_id','Nama'],
    value_vars=['Bhs Inggris', 'Matematika','Ekonomi'])
melt_table
```

	Mhs_id	Nama	variable	value
0	001	Budi	Bhs Inggris	100
1	002	Joko	Bhs Inggris	90
2	003	Maya	Bhs Inggris	98
3	001	Budi	Matematika	90
4	002	Joko	Matematika	87
5	003	Maya	Matematika	72
6	001	Budi	Ekonomi	90
7	002	Joko	Ekonomi	80
8	003	Maya	Ekonomi	78



# Lambda Functions

- Fungsi lambda adalah fungsi anonim kecil.
- Fungsi lambda dapat mengambil sejumlah argumen, tetapi hanya dapat memiliki satu ekspresi

```
lambda arguments : expression
```

```
[36] x = lambda a, b : a * b  
      print(x(5, 10))
```

```
50
```

```
▶ x = lambda a : a + 10  
   print(x(5))
```

```
15
```

```
▶ status = lambda umur : 'remaja' if umur > 10 and umur < 20 else 'bukan remaja'  
      print(status(15))  
remaja
```



# Lambda in DataFrame

- Kita bisa menggunakan fungsi *apply*, untuk menjalankan *lambda function*

```
DataFrame.apply(func, axis=0, raw=False, result_type=None, args=(), **kwargs)
```

```
import pandas as pd
mahasiswa = pd.DataFrame({
    'Mhs_id' : ['001', '002', '003'],
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Matematika': [90, 47, 72]})
mahasiswa
```

	Mhs_id	Nama	Matematika
0	001	Budi	90
1	002	Joko	47
2	003	Maya	72

```
mahasiswa['Grade'] = mahasiswa['Matematika'].apply(lambda nilai:
    ('A' if nilai >=90 else ('B' if nilai >=70 and nilai <90 else 'C')))
mahasiswa
```

	Mhs_id	Nama	Matematika	Grade
0	001	Budi	90	A
1	002	Joko	47	C
2	003	Maya	72	B



Thank  
YOU