Learning Progress Review

25 - 30 Oktober 2025

Group 5

TIM SERU

Table of contents

01

Introduction and Basic Dataframe

02

Basic Dataframe

03

Exploring Dataframe

01

Introduction and Basic Dataframe

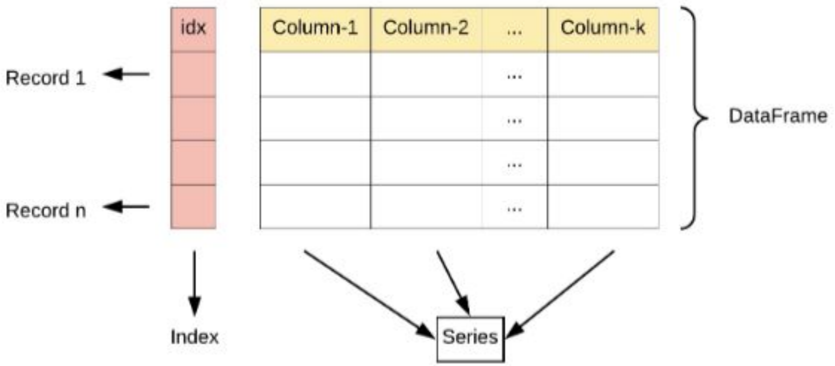
Pengantar Pandas

- Pandas merupakan library open source Python yang mendukung struktur dan analisis data yang mudah digunakan serta berkinerja tinggi.

- Library Pandas memungkinkan untuk dapat mengolah data dalam bentuk tabular, data time-series, matriks, dan lain-lain.

- Beberapa tujuan library Pandas:
  - Menganalisa data
  - Membersihkan data
  - Eksplorasi data
  - Manipulasi data

Struktur Data Pandas: Series

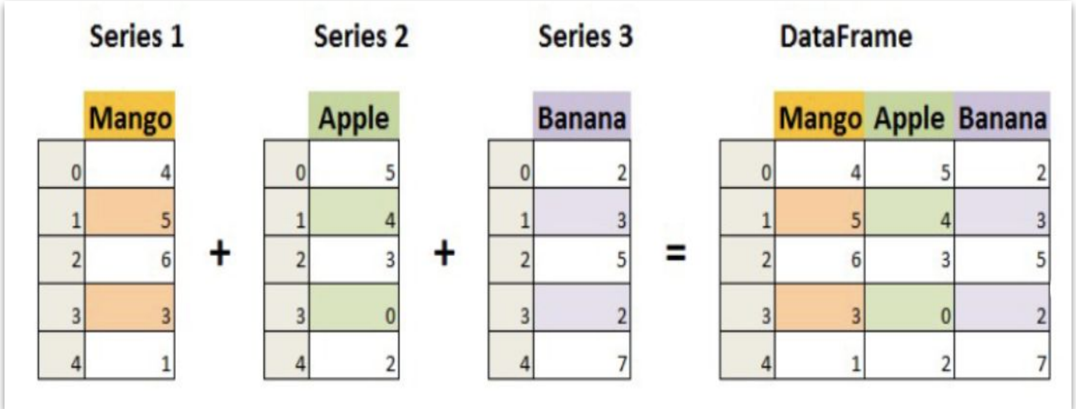


- **Series** merupakan struktur data dasar pada Pandas, yang berupa sebuah kolom berisi data atau sebuah array satu dimensi, seperti NumPy array.
- **pandas.Series**(data, index, dtype, name, copy, fastpath)

Sejumlah parameter yang dapat diterapkan pada Series:

- Parameter **Data**: Tipe data yang dapat ditampung berupa integer, float dan juga string
- Parameter **Index**: Indeks dari Series. Jumlah indeks harus sama dengan jumlah data
- Parameter **dtype**: Diisi dengan tipe data dari Series
- Parameter **copy**: dapat digunakan untuk menyalin data

Struktur Data Pandas: DataFrame



- **DataFrame** merupakan struktur data utama pada Pandas, yang berupa sebuah tabel/data tabular dengan array dua dimensi, yaitu baris dan kolom.
- Setiap kolom pada **DataFrame** merupakan objek dari Series, dan baris terdiri dari elemen yang ada pada Series.
- **pandas.DataFrame**(data, index, columns, dtype, copy)

Sejumlah parameter yang dapat diterapkan pada DataFrame:

- Parameter **Index**: merupakan label untuk baris
- Parameter **Column**: merupakan label untuk kolom
- Parameter **dtype**: Diisi dengan tipe data per kolom
- Parameter **copy**: dapat digunakan untuk menyalin data

Series pada Pandas

- Membuat Series dari data berupa list

  data = pd.Series([2, 3, 5])

```
data = pd.Series([2, 3, 5])
data
```

```
0    2
1    3
2    5
dtype: int64
```

- Membuat Series dengan memodifikasi indeks

  data = pd.Series([‘Rahmat’, 3, False], index=[‘a’, ‘b’, ‘c’])

```
data = pd.Series(['Rahmat', 3, False], index=['a', 'b', 'c'])
data
```

```
a    Rahmat
b         3
c     False
dtype: object
```

- Mengakses nilai dari indeks pada Series yang dibuat dari NumPy array

  array = np.array([2, 3, 5])

  data = pd.Series(array)
  data[2]

```
array = np.array([2, 3, 5])
data = pd.Series(array)
data[2]
```

```
5
```

- Membuat Series dari data berupa NumPy array

  array = np.array([2, 3, 5])

  data = pd.Series(array)

```
array = np.array([2, 3, 5])
data = pd.Series(array)
data
```

```
0    2
1    3
2    5
dtype: int32
```

- Mengakses nilai dari indeksnya pada Series yang dibuat dari list

  data = pd.Series([‘Rahmat’, 3, False], index=[‘a’, ‘b’, ‘c’])

  data[‘a’]

```
data = pd.Series(['Rahmat', 3, False], index=['a', 'b', 'c'])
data['a']
```

```
'Rahmat'
```

pandas

DataFrame pada Pandas

a. Membuat DataFrame dari Tipe Data berupa Dictionary

```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]}

df = pd.DataFrame(data)
df
```

	calories	duration
0	420	50
1	380	40
2	390	45

b. Membuat DataFrame dari Tipe Data berupa List

```
import pandas as pd

data = [[420, 50],
        [380, 40],
        [390, 45]]

df = pd.DataFrame(data, columns=["calories","duration"])
df
```

	calories	duration
0	420	50
1	380	40
2	390	45

pandas

Read File pada Pandas

a. Membuat DataFrame dari file Excel

```
import pandas as pd

df = pd.read_excel('data.xlsx')
df
```

	calories	duration
0	420	50
1	380	40
2	390	45

b. Membuat DataFrame dari file CSV

```
import pandas as pd

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

	calories	duration
0	420	50
1	380	40
2	390	45

c. Membuat DataFrame dari file JSON

```
import pandas as pd

df = pd.read_json('data.json')
df
```

	calories	duration
0	420	50
1	380	40
2	390	45

Data Analyzing pada Pandas

- **head()**
  Fungsi ini digunakan untuk mengembalikan sejumlah baris awal dari suatu Series atau DataFrame.

  Secara default akan mengembalikan lima baris pertama.

```python
import pandas as pd

data = {
    "calories": [420, 380, 390, 500, 340, 410, 530],
    "duration": [50, 40, 45, 53, 50, 48, 33]}

df = pd.DataFrame(data)
df.head()
```

	calories	duration
0	420	50
1	380	40
2	390	45
3	500	53
4	340	50

pandas

Data Analyzing pada Pandas

- **tail()**
  Fungsi ini digunakan untuk mengembalikan sejumlah baris akhir dari suatu Series atau DataFrame.

  Secara default akan mengembalikan lima baris terakhir.

```
import pandas as pd

data = {
    "calories": [420, 380, 390, 500, 340, 410, 530],
    "duration": [50, 40, 45, 53, 50, 48, 33]}

df = pd.DataFrame(data)
df.tail()
```

	calories	duration
2	390	45
3	500	53
4	340	50
5	410	48
6	530	33

Data Analyzing pada Pandas

- **info()**
  Fungsi ini digunakan untuk mengembalikan informasi-informasi penting terkait DataFrame, seperti nama-nama kolom, nilai non-null, dtype dan penggunaan memori.

```python
import pandas as pd

data = {
    "calories": [420, 380, 390, 500, 340, 410, 530],
    "duration": [50, 40, 45, 53, 50, 48, 33]}

df = pd.DataFrame(data)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   calories  7 non-null      int64
 1   duration  7 non-null      int64
dtypes: int64(2)
memory usage: 240.0 bytes
```

Data Analyzing pada Pandas

- **describe()**
  Fungsi ini digunakan untuk menampilkan/ mengembalikan atribut-atribut numerik DataFrame, seperti jumlah, rata-rata, nilai maksimum, nilai minimum, dll.

```python
import pandas as pd

data = {
    "calories": [420, 380, 390, 500, 340, 410, 530],
    "duration": [50, 40, 45, 53, 50, 48, 33]}

df = pd.DataFrame(data)
df.describe()
```

	calories	duration
count	7.000000	7.000000
mean	424.285714	45.571429
std	67.541874	6.948792
min	340.000000	33.000000
25%	385.000000	42.500000
50%	410.000000	48.000000
75%	460.000000	50.000000
max	530.000000	53.000000

pandas

Create, Choose, Read, Write Dataframe

A. Untuk membuat DataFrame di Pandas Python, ada beberapa tahapan yang harus dilakukan

1. Import Library Pandas

```
>>> pip install pandas
```

atau

```
>>> import pandas as pd
```

Di Google Collab :

2. Create DataFrame.

DataFrame dapat dibuat dari **berbagai sumber data :**

a. Membuat DataFrame dari Dictionary

```python
import pandas as pd

data = {
    'Nama': ['Raka', 'Gilang', 'Rafi', 'Adzka'],
    'Umur': [21, 22, 23, 20],
    'Kota': ['Jakarta', 'Bandung', 'Surabaya', 'Yogyakarta']
}

df = pd.DataFrame(data)
print(df)
```

```
     Nama  Umur        Kota
0    Raka    21     Jakarta
1  Gilang    22     Bandung
2    Rafi    23    Surabaya
3   Adzka    20  Yogyakarta
```

b. **Membuat DataFrame Dari list of lists**

```python
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

c. Membuat DataFrame Menggunakan Array

```
data = np.array([['','Col1','Col2'],
                 ['Row1',1,2],
                 ['Row2',3,4]])

print(pd.DataFrame(data=data[1:,1:],
                   index=data[1:,0],
                   columns=data[0,1:]))
```

```
     Col1 Col2
Row1    1    2
Row2    3    4
```

B. Pilih (Select) Data dari DataFrame:

B.1. Memilih Kolom

1. **Memilih 1 kolom**

```python
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

```python
df ['Nama']
```

	Nama
0	Raka
1	Gilang
2	Rafi

dtype: object

2. Memilih beberapa kolom

```
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

```
df[['Nama','Umur']]
```

	Nama	Umur
0	Raka	21
1	Gilang	22
2	Rafi	23

B. Pilih (Select) Data dari DataFrame:

B.2. Memilih Baris

a. Berdasarkan index

```
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

```
df.loc[1]
```

	1
Nama	Gilang
Umur	22
Kota	Bandung

dtype: object

b. Memilih beberapa baris

```python
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

```python
df.loc[[0, 2]]      # ambil baris ke-0 dan ke-2
```

	Nama	Umur	Kota
0	Raka	21	Jakarta
2	Rafi	23	Surabaya

```python
df.iloc[0:3]        # ambil baris ke-0 sampai sebelum ke-3
```

	Nama	Umur	Kota
0	Raka	21	Jakarta
1	Gilang	22	Bandung
2	Rafi	23	Surabaya

B. Pilih (Select) Data dari DataFrame:

B.3. Pilih baris dan kolom sekaligus

a. Dengan `.loc[]` (berdasarkan label)

```
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

```
df.loc[0:2, ['Nama', 'Kota']]
```

	Nama	Kota
0	Raka	Jakarta
1	Gilang	Bandung
2	Rafi	Surabaya

b. Dengan `.iloc[]` (berdasarkan posisi)

```
data = [
    ['Raka', 21, 'Jakarta'],
    ['Gilang', 22, 'Bandung'],
    ['Rafi', 23, 'Surabaya']
]

df = pd.DataFrame(data, columns=['Nama', 'Umur', 'Kota'])
print(df)
```

```
     Nama  Umur      Kota
0    Raka    21   Jakarta
1  Gilang    22   Bandung
2    Rafi    23  Surabaya
```

```
df.iloc[0:2, 0:2]    # baris 0-1, kolom 0-1
```

	Nama	Umur
0	Raka	21
1	Gilang	22

B. Membaca (Read) Data ke dalam DataFrame :

Contoh : Membaca file california_housing_train.csv dari drive samplesample

```
df = pd.read_csv("/content/sample_data/california_housing_train.csv")
```

```
print(df.head())
```

Membaca 5 baris teratas dari file tersebut

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -114.31     34.19                15.0       5612.0          1283.0
1    -114.47     34.40                19.0       7650.0          1901.0
2    -114.56     33.69                17.0        720.0           174.0
3    -114.57     33.64                14.0       1501.0           337.0
4    -114.57     33.57                20.0       1454.0           326.0

   population  households  median_income  median_house_value
0      1015.0       472.0         1.4936             66900.0
1      1129.0       463.0         1.8200             80100.0
2       333.0       117.0         1.6509             85700.0
3       515.0       226.0         3.1917             73400.0
4       624.0       262.0         1.9250             65500.0
```

```
df.shape
```

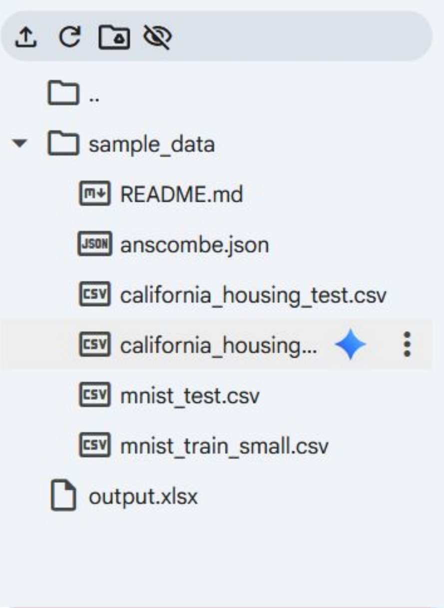
```
(17000, 9)
```

Info jumlah baris 17000, jumlah kolom 9

B. Menulis (Write) DataFrame ke File

Contoh : Menulis file california_housing_train.csv ke drive dengan format excel

```python
df.to_excel('output.xlsx', index=False)
```



longitude	latitude	ng_media	otal_room	total_bedrooms	populatio	ouseholds	median_income	an_house_value
-114.31	34.19	15	5612	1283	1015	472	1.4936	66900
-114.47	34.4	19	7650	1901	1129	463	1.82	80100
-114.56	33.69	17	720	174	333	117	1.6509	85700
-114.57	33.64	14	1501	337	515	226	3.1917	73400
-114.57	33.57	20	1454	326	624	262	1.925	65500
-114.58	33.63	29	1387	236	671	239	3.3438	74000
-114.58	33.61	25	2907	680	1841	633	2.6768	82400
-114.59	34.83	41	812	168	375	158	1.7083	48500
-114.59	33.61	34	4789	1175	3134	1056	2.1782	58400
-114.6	34.83	46	1497	309	787	271	2.1908	48100
-114.6	33.62	16	3741	801	2434	824	2.6797	86500
-114.6	33.6	21	1988	483	1182	437	1.625	62000
-114.61	34.84	48	1291	248	580	211	2.1571	48600
-114.61	34.83	31	2478	464	1346	479	3.212	70400
-114.63	32.76	15	1448	378	949	300	0.8585	45000
-114.65	34.89	17	2556	587	1005	401	1.6991	69100
-114.65	33.6	28	1678	322	666	256	2.9653	94900
-114.65	32.79	21	44	33	64	27	0.8571	25000
-114.66	32.74	17	1388	386	775	320	1.2049	44000
-114.67	33.92	17	97	24	29	15	1.2656	27500
-114.68	33.49	20	1491	360	1135	303	1.6395	44400
-114.73	33.43	24	796	243	227	139	0.8964	59200
-114.94	34.55	20	350	95	119	58	1.625	50000
-114.98	33.82	15	644	129	137	52	3.2097	71300
-115.22	33.54	18	1706	397	3424	283	1.625	53500
-115.32	32.82	34	591	139	327	89	3.6528	100000

Sheet1

02

Basic Dataframe

2.1. Data Exploration

1) Head
Mengembalikan baris n pertama. Secara default baris 5 teratas.

2) Tail
Mengembalikan baris n terakhir. Secara default baris 5 terbawah.

3) Info
Memberikan ringkasan singkat df.

4) Shape & Columns
Menampilkan dimensi - nama kolom.

5) Describe
Menampilkan rata-rata, standar deviasi, min, max, jumlah persentil tiap kolom.

```
import pandas as pd

data = {
    'name': ['Rizky', 'Siti', 'Andi', 'Putri', 'Budi',
             'Dewi', 'Agus', 'Nadia', 'Fajar', 'Intan'],
    'location': ['Jakarta', 'Bandung', 'Surabaya', 'Yogyakarta', 'Medan',
                 'Makassar', 'Semarang', 'Palembang', 'Denpasar', 'Malang'],
    'age': [24, 28, 22, 25, 30, 27, 29, 23, 26, 24]
}

df = pd.DataFrame(data)
print(df)
```

```
    name    location  age
0  Rizky     Jakarta   24
1   Siti     Bandung   28
2   Andi    Surabaya   22
3  Putri  Yogyakarta   25
4   Budi       Medan   30
5   Dewi    Makassar   27
6   Agus    Semarang   29
7  Nadia   Palembang   23
8  Fajar    Denpasar   26
9  Intan      Malang   24
```

```
print(df.head(3))
```

```
    name  location  age
0  Rizky   Jakarta   24
1   Siti   Bandung   28
2   Andi  Surabaya   22
```

```
print(df.tail(3))
```

```
    name   location  age
7  Nadia  Palembang   23
8  Fajar   Denpasar   26
9  Intan     Malang   24
```

```
print(df.shape)
```

```
(10, 3)
```

```
print(df.columns)
```

```
Index(['name', 'location', 'age'], dtype='object')
```

```
print(df.info)
```

```
<bound method DataFrame.info of     name    location  age
0  Rizky     Jakarta   24
1   Siti     Bandung   28
2   Andi    Surabaya   22
3  Putri  Yogyakarta   25
4   Budi       Medan   30
5   Dewi    Makassar   27
6   Agus    Semarang   29
7  Nadia   Palembang   23
8  Fajar    Denpasar   26
9  Intan      Malang   24>
```

```
print(df.describe())
```

```
             age
count   10.00000
mean    25.80000
std      2.65832
min     22.00000
25%     24.00000
50%     25.50000
75%     27.75000
max     30.00000
```

2.2. Data Selection

1) Single Column

2) Label Based

3) Index Based

```
   name    location  age
0  Rizky    Jakarta   24
1  Siti     Bandung   28
2  Andi    Surabaya   22
3  Putri  Yogyakarta  25
4  Budi       Medan   30
5  Dewi    Makassar   27
6  Agus    Semarang   29
7  Nadia  Palembang   23
8  Fajar   Denpasar   26
9  Intan     Malang   24
```

1)

```
print(df.age)

0    24
1    28
2    22
3    25
4    30
5    27
6    29
7    23
8    26
9    24
Name: age, dtype: int64
```

```
print(df['age'])

0    24
1    28
2    22
3    25
4    30
5    27
6    29
7    23
8    26
9    24
Name: age, dtype: int64
```

2) df.loc[row,column]

```
print(df.loc[:,'age'])

0    24
1    28
2    22
3    25
4    30
5    27
6    29
7    23
8    26
9    24
Name: age, dtype: int64
```

```
print(df.loc[0:2,'age'])

0    24
1    28
2    22
Name: age, dtype: int64
```

3) df.iloc[row,column]

```
print(df.iloc[:,2])

0    24
1    28
2    22
3    25
4    30
5    27
6    29
7    23
8    26
9    24
Name: age, dtype: int64
```

```
print(df.iloc[:2, 0:2])

    name location
0  Rizky  Jakarta
1   Siti  Bandung
```

2.3. Sorting Data Frame

Tujuan utama sorting adalah untuk mengurutkan data dari rendah ke tinggi maupun sebaliknya. Pandas menyortir Dataframe berdasarkan nilai maupun indeks.

Values based sorting

DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False, key=None)

Index based sorting

DataFrame.sort_values(by, axis=0, level=None, ascending=True, inplace=False, kind='quicksort', na_position='last', sort_remaining=True, ignore_index=False, key=None)

Keterangan Parameter

- **by**: nama atau daftar nama yang akan disortir.
- **axis**: aksis yang akan disortir.
- **level**: defaultnya None. Jika tidak, akan mengurutkan nilai sesuai indeks yang ditentukan.
- **ascending**: True berarti mengurutkan dari kecil ke besar, False berarti besar ke kecil.
- **inplace**: True untuk menerapkan inplace
- **kind**: untuk memilih 'quicksort';'mergesort';'heapsort'; dan 'stable'.
- **na_position**: opsi meletakkan NaN di awal (first) atau di akhir (last). Defaultnya, last.
- **ignore-index**: mengabaikan indeks asli yang disortir.

2.4. Filtering Data Frame

Filtering data frame dapat dilakukan melalui:

- Mendapat satu atau sejumlah kolom
- Memfilter berdasarkan kondisi single
- Memfilter berdasarkan beberapa kondisi
- Memilih kolom yang diperlukan dengan kondisi
- Memperbarui nilai pada kolom

1) Filter menggunakan nama kolom

```
df[['Nama','Kota']]
```

	Nama	Kota
0	Andi	Jakarta
1	Budi	Bandung
2	Caca	Surabaya
3	Dedi	Semarang
4	Euis	Yogyakarta

2) Filter menggunakan metode items

```
df.filter(items=['Nama','Kota'])
```

	Nama	Kota
0	Andi	Jakarta
1	Budi	Bandung
2	Caca	Surabaya
3	Dedi	Semarang
4	Euis	Yogyakarta

```
print("Original DataFrame:")
display(df)

# Filter using 'and' (&)
filtered_df_and = df[(df['Category'] == 'A') & (df['Value'] > 15)]
print("\nFiltered DataFrame using 'and':")
display(filtered_df_and)
```

Original DataFrame:

	Category	Value	Another_Value
0	A	10	100
1	B	20	200
2	A	30	300
3	C	40	400
4	B	50	500
5	C	60	600

Filtered DataFrame using 'and':

	Category	Value	Another_Value
2	A	30	300

3) Filter menggunakan iloc

```
df.iloc[0,:]
```

	0
Nama	Andi
Usia	25
Kota	Jakarta

4) Filter menggunakan loc

2.5. Adding Columns

1) Metode List

```
data = pd.DataFrame ({
    'Nama': ['Andi', 'Budi', 'Caca', 'Dedi', 'Eka'],
    'Usia': [25, 30, 35, 40, 45],
    'Tinggi badan': ['167', '169', '158', '165', '159']
})
Gender = ['L', 'L', 'P', 'L', 'P']
data['Gender'] = Gender
data
```

	Nama	Usia	Tinggi badan	Gender
0	Andi	25	167	L
1	Budi	30	169	L
2	Caca	35	158	P
3	Dedi	40	165	L
4	Eka	45	159	P

2) Metode Insert

```
data = pd.DataFrame ({
    'Nama': ['Andi', 'Budi', 'Caca', 'Dedi', 'Eka'],
    'Usia': [25, 30, 35, 40, 45],
    'Tinggi badan': ['167', '169', '158', '165', '159']
})
data.insert(3, 'Gender', ['L', 'L', 'P', 'L', 'P'])
data
```

	Nama	Usia	Tinggi badan	Gender
0	Andi	25	167	L
1	Budi	30	169	L
2	Caca	35	158	P
3	Dedi	40	165	L
4	Eka	45	159	P

3) Metode Assign

```
data = pd.DataFrame ({
    'Nama': ['Andi', 'Budi', 'Caca', 'Dedi', 'Eka'],
    'Usia': [25, 30, 35, 40, 45],
    'Tinggi badan': ['167', '169', '158', '165', '159']
})
data.assign(Gender = ['L', 'L', 'P', 'L', 'P'])
```

	Nama	Usia	Tinggi badan	Gender
0	Andi	25	167	L
1	Budi	30	169	L
2	Caca	35	158	P
3	Dedi	40	165	L
4	Eka	45	159	P

4) Metode Dictionary

```
data = pd.DataFrame ({
    'Nama': ['Andi', 'Budi', 'Caca', 'Dedi', 'Eka'],
    'Usia': [25, 30, 35, 40, 45],
    'Tinggi badan': ['167', '169', '158', '165', '159']
})

gender_dict = {0: 'L', 1: 'L', 2: 'P', 3: 'L', 4: 'P'}

data['Gender'] = pd.Series(gender_dict)
data
```

	Nama	Usia	Tinggi badan	Gender
0	Andi	25	167	L
1	Budi	30	169	L
2	Caca	35	158	P
3	Dedi	40	165	L
4	Eka	45	159	P

2.6. Grouping

- Grouping merupakan teknik pengelompokan dataframe menurut kriteria tertentu.
- Grouping memudahkan analisis data sesuai kriteria spesifik.
- Grouping di Pandas menggunakan fungsi groupby().

1) Grouping dengan agregasi rerata

```
data = pd.DataFrame({
    'Category': ['A', 'B', 'A', 'B', 'A', 'C', 'C'],
    'Value': [10, 15, 12, 18, 11, 20, 22]
    })
grouped_data = data.groupby('Category')['Value'].mean()
grouped_data
```

Category	Value
A	11.0
B	16.5
C	21.0

2) Grouping dengan agregasi sum

```
data = pd.DataFrame({
    'Category': ['A', 'B', 'A', 'B', 'A', 'C', 'C'],
    'Value': [10, 15, 12, 18, 11, 20, 22]
    })
grouped_data = data.groupby('Category')['Value'].sum()
grouped_data
```

Category	Value
A	33
B	33
C	42

3) Grouping dengan agregasi count

```
data = pd.DataFrame({
    'Category': ['A', 'B', 'A', 'B', 'A', 'C', 'C'],
    'Value': [10, 15, 12, 18, 11, 20, 22]
    })
grouped_data = data.groupby('Category').count()
grouped_data
```

Category	Value
A	3
B	2
C	2

2.7. Merging Data Frame

Merupakan teknik untuk menggabungkan beberapa DataFrame menjadi satu. Merging pada Pandas dilakukan dengan fungsi merge(). Konsep merge pada Pandas secara umum sama dengan konsep join pada SQL.

1) Left and Right

```python
df1 = pd.DataFrame({'key': ['A', 'B', 'C', 'D'],
                    'value1': [1, 2, 3, 4]})
df2 = pd.DataFrame({'key': ['C', 'D', 'E', 'F'],
                    'value2': [5, 6, 7, 8]})
merged_df_left = pd.merge(df1, df2, on='key', how='left')
merged_df_right = pd.merge(df1, df2, on='key', how='right')
print("Left Merge:")
display(merged_df_left)
print("\nRight Merge:")
display(merged_df_right)
```

Left Merge:

	key	value1	value2
0	A	1	NaN
1	B	2	NaN
2	C	3	5.0
3	D	4	6.0

Right Merge:

	key	value1	value2
0	C	3.0	5
1	D	4.0	6
2	E	NaN	7
3	F	NaN	8

2) Inner and Outer

```python
df1 = pd.DataFrame({'key': ['A', 'B', 'C', 'D'],
                    'value1': [1, 2, 3, 4]})
df2 = pd.DataFrame({'key': ['C', 'D', 'E', 'F'],
                    'value2': [5, 6, 7, 8]})
merged_df_inner = pd.merge(df1, df2, on='key', how='inner')
merged_df_outer = pd.merge(df1, df2, on='key', how='outer')
print("Left Inner:")
display(merged_df_inner)
print("\nRight Outer:")
display(merged_df_outer)
```

Left Inner:

	key	value1	value2
0	C	3	5
1	D	4	6

Right Outer:

	key	value1	value2
0	A	1.0	NaN
1	B	2.0	NaN
2	C	3.0	5.0
3	D	4.0	6.0
4	E	NaN	7.0
5	F	NaN	8.0

03

Exploring Dataframe

Data Cleansing

Pengertian : proses mengidentifikasi, memperbaiki, atau menghapus data yang tidak akurat, tidak lengkap, duplikat, atau tidak relevan dari sebuah dataset.

Tujuan : untuk memastikan bahwa data yang akan dianalisis menjadi bersih, konsisten, dan valid, sehingga hasil analisis atau model yang dibangun menjadi lebih akurat dan dapat dipercaya.

Konsep

Column Rename

- Header dan isi tidak sesuai
- Nama kolom tidak konsisten
- *Case inconsistency*
- *Strange character (symbol, space, punctuation)*
- Campuran singkatan, istilah,, ejaan, bahasa.

Missing value inspection and handling

- Data kosong/hilang:

- Selama akuisisi data
- Tidak sengaja
- Data rusak
- Unmatched data
- Value tidak ada

Data type inspection and correction

Tipe data setiap kolom pada DataFrame harus sesuai dengan jenis data dan kebutuhan analisisnya.

Contoh:

- Kolom “Tanggal Transaksi” harus terbaca **datetime**.
- Kolom “Harga” harus terbaca **float/int.**
- Kolom “Kode Barang” harus terbaca **string**

Implementasi

Column rename

df.rename(columns=str.lower, inplace=True)

Fungsi untuk mengubah nama kolom untuk diaplikasikan ke seluruh atau sebagian kolom yang ada.

Missing value

1. Inspection :

   a. .isnull()
      mengembalikan DataFrame data yang hilang.
   b. .notnull()
      mengembalikan DataFrame data yang tidak hilang.

2. Handling
   a. .dropna()
      menghilangkan nilai yang hilang.
   b. .fillna()
      mengisi nilai yang hilang.

Data type

1. Inspection:
   Tipe data dalam dataframe
   (String, Integer, Float, Boolean)

2. Cek:
   df.dtypes

3. Konversi kolom dengan tipe data yang sesuai.
   Contoh:
   from datetime import datetime as dtime
   data['columnC'] = pd.to_datetime(data['columnC'], format="%Y-%m-%d")

Data Blending

Pengertian

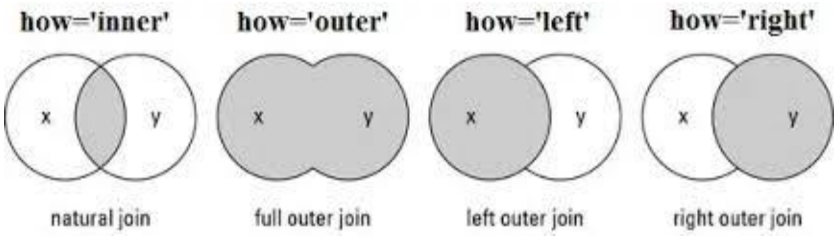
Data Blending adalah proses menggabungkan beberapa sumber data menjadi satu dataset yang lebih lengkap, untuk analisis dan visualisasi. Biasanya dilakukan saat data tersebar di beberapa tabel atau file

2 Teknik Utama di Pandas

Metode	Fungsi	Cara Kerja	Kapan Digunakan
pd.merge()	Join data berdasarkan kolom kunci (key)	Mirip SQL JOIN	Jika dua DataFrame memiliki kolom yang berelasi
pd.concat()	Menggabungkan baris atau kolom antar DataFrame	Mirip UNION / APPEND	Jika struktur kolom sama dan ingin menumpuk data

Data Blending

Jenis-Jenis Merge Join



Contoh Syntax Merge Join

```
# Parameter "how" bisa disesuaikan: 'inner', 'left', 'right', atau 'outer'

merged_df = pd.merge(df_seller, df_trans, on='Seller_ID', how='inner')
```

Contoh Syntax Concat

```
concat_df = pd.concat([df1, df2], axis=0)
print(concat_df)
```

Axis = 0 – Menambahkan Baris

Axis = 1 – Menambahkan Kolom

Data Transformasi

Pengertian

Data transformasi adalah suatu cara untuk mengubah data dari satu format ke format lain sehingga data menjadi lebih konsisten dan siap dianalisis.

Jenis Transformasi Data

1. With Map
   Cocok digunakan mengubah nilai string ke angka atau sebaliknya.
   df['nama_kolom_baru'] =df['nama_kolom_lama'].map((values_lama : values baru, ...))

```
df['origin_name'] = df ['origin'].map((1: 'North America', 2: 'Europe', 3: 'Asia'))
display(df[0:50])
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	origin_name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	North America
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	North America
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite	North America
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst	North America
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino	North America

2. With Lambda Function
   Cocok digunakan untuk menerapkan suatu operasi setiap elemen dalam kolom.
   df['nama_kolom_baru'] =df['nama_kolom_lama'].apply(lambda values_lama : operasi)

```
df
```

	Name	Total_Marks
0	Rohan	455
1	Elvish	250
2	Deepak	495
3	Soni	400
4	Radhika	350
5	Vansh	450

```
df['Percentage'] = df['Total_Marks'].apply(lambda x: x/500*100)
df
```

	Name	Total_Marks	Percentage
0	Rohan	455	91.0
1	Elvish	250	50.0
2	Deepak	495	99.0
3	Soni	400	80.0
4	Radhika	350	70.0
5	Vansh	450	90.0

Data Grouping dan Aggregation

Group by

Mengelompokkan suatu baris kemudian menetapkan fungsi di dalamnya lalu menggabungkan hasilnya.

nama_dataframe.groupby([‘nama_kolom’]).mean()

```
   Animal  Max Speed
0  Falcon      380.0
1  Falcon      370.0
2  Parrot       24.0
3  Parrot       26.0
>>> df.groupby(['Animal']).mean()
        Max Speed
Animal
Falcon      375.0
Parrot       25.0
```

Pivot

Alat untuk merangkum dan menganalisis data yang besar dari tabel dengan cara mengubah baris menjadi kolom atau sebaliknya.

pd.pivot_table(nama_dataframe, index=[‘nama_kolom’, ...], values = ‘nama_kolom’, aggfunc='sum')

```
data = data[data['transactionType']=='PURCHASE']
pd.pivot_table(data, index=['merchantCategoryCode', 'merchantCountryCode'], values='transactionAmount', aggfunc='mean').head(11)
```

		transactionAmount
merchantCategoryCode	merchantCountryCode	
airline	CAN	120.453043
	MEX	178.147949
	PR	136.571667
	US	148.552881
auto	CAN	143.973704
	MEX	172.448085
	PR	205.022105
	US	149.902006
cable/phone	CAN	140.793750
	MEX	102.668462

Data Grouping dan Aggregation

Cross Tabulation

Digunakan untuk membuat tabel silang dengan menghitung dua lebih faktor yang menunjukkan jumlah kemunculan untuk berbagai kombinasi variabel tersebut.

pd.crosstab(nama_dataframe['nama_kolom'], nama_dataframe['nama_kolom'])

```
pd.crosstab(data.merchantCategoryCode, data.posEntryMode, margins=True)
```

posEntryMode	2.0	5.0	9.0	80.0	90.0	All
merchantCategoryCode						
airline	2473	3954	3032	212	266	9937
auto	2535	4060	3035	194	259	10083
cable/phone	298	561	536	19	45	1459
entertainment	17395	27618	20565	1428	1748	68754
fastfood	25214	40546	30440	2010	2517	100727
food	17168	27158	20490	1396	1701	67913
food_delivery	1365	1902	1571	0	98	4936
fuel	5483	8822	7009	449	660	22423
furniture	1905	3183	2314	166	200	7768
gym	869	1103	691	38	173	2874
health	3479	5870	4269	287	369	14274
hotels	5666	9156	6894	453	578	22747
mobileapps	3694	5839	4344	290	353	14520
online_gifts	8322	13059	9977	684	840	32882
online_retail	40275	64219	48869	3312	4043	160718
online_subscriptions	2969	4398	3338	221	235	11161
personal care	4242	6779	4999	338	462	16820
rideshare	12547	20207	15233	1036	1269	50292
subscriptions	4690	7181	5587	388	435	18281
All	160589	255615	193193	12921	16251	638569

Thank You!