

Manipulasi Data dengan Pandas 3

Aggregation & GroupBy
Time Series in Pandas

Topik

M15 - Manipulasi Data Pandas - Bagian 3

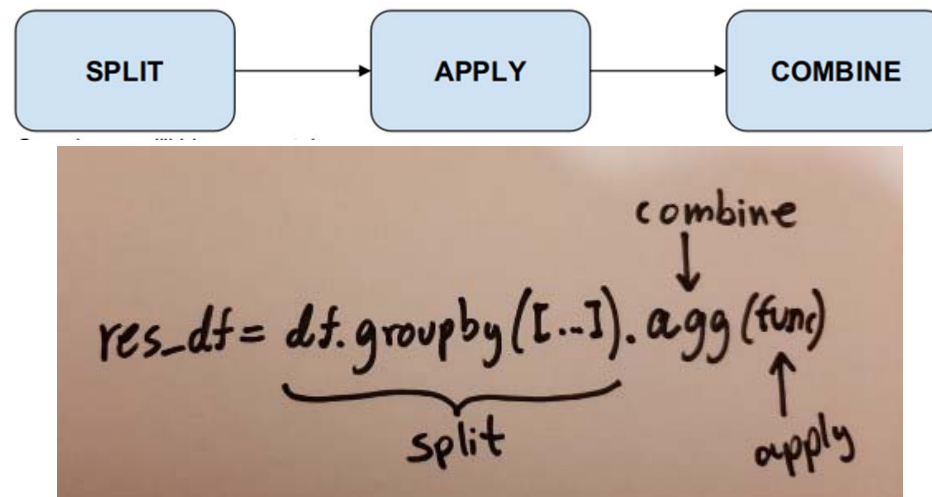
- **Aggregation & GroupBy**
 - Review Inspeksi Data
 - Groupby dan Agregasi dengan Fungsi Statistik Dasar
 - Groupby dengan Multiple Aggregations
 - Groupby dengan Custom Aggregations
 - Groupby dengan Custom Aggregations by dict
- **Time Series in Pandas**
 - Load Dataset as Time Series
 - Convert to Datetime
 - Resampling untuk Time Series Data
 - Downsampling Data
 - Upsampling Data
 - Resampling by Frequency
 - Visualisasi

- 1. Aggregation & GroupBy**
- 2. Time Series in Pandas**

Aggregation & GroupBy

- Teknik agregasi dan groupby digunakan untuk melihat dataset dari point of view yang berbeda.
- Dapat digunakan untuk mengetahui apakah set data tersebut dapat dikelompokkan berdasarkan aturan tertentu, yang kemudian juga bisa menerapkan beberapa fungsi atau metode statistik ke hasil group dataset itu untuk mengetahui behavior dari data tersebut secara summary/overview.
- Misalnya kita memerlukan pengelompokan data atau menentukan resume dari data secara keseluruhan atau masing-masing kelompok datanya, misal dari sisi rata-rata, nilai tengah, nilai maksimum, nilai minimum dan yang lainnya.

Basic Concept of Groupby & Aggregation



1. **Split:** membagi data menjadi kelompok-kelompok dengan cara membuat sebuah groupby object dari original DataFrame
2. **Apply:** menerapkan sebuah fungsi pada masing-masing kelompok tersebut. Fungsi ini bisa berupa perhitungan summary statistic (transform atau filter)
3. **Combine:** menggabungkan semua hasil fungsi dari tiap kelompok menjadi dataframe baru.

Groupby, split-apply-combine and pandas

Case study: NetFlix

- We will learn how to use the pandas groupby operation, which draws from the well-known split-apply-combine strategy, on Netflix movie data.
- Netflix recently released some user ratings data.
- We wanted to ask a straightforward question: *do Netflix subscribers prefer older or newer movies?*
- Intuitively:
 - we want to split the dataset into groups, one for each year, and then to compute a summary statistic, such as the mean or the median,
 - and then to see whether this statistic increases over the years
 - after this, you may want to perform a statistical test.

Data Exploration with pandas

Case study: NetFlix

Import the necessary packages, load the data, and check out the first five rows of the data

<https://raw.githubusercontent.com/datacamp/community-groupby/master/data/chasewillden-netflix-shows/data/netflix.csv>

```
# Import packages and set visualization style
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline

# Import data and check out head of DataFrame
df = pd.read_csv('data/chasewillden-netflix-shows/data/netflix.csv')
df.head()
```

Data Exploration with pandas

Case study: NetFlix

	title	rating	ratinglevel	ratingdescription	release_year	user_rating_score	user_rating_size
0	White Chicks	PG-13	crude and sexual humor, language and some drug...	80	2004	82.0	80
1	Lucky Number Slevin	R	strong violence, sexual content and adult lang...	100	2006	NaN	82
2	Grey's Anatomy	TV-14	Parents strongly cautioned. May be unsuitable ...	90	2016	98.0	80
3	Prison Break	TV-14	Parents strongly cautioned. May be unsuitable ...	90	2008	98.0	80
4	How I Met Your Mother	TV-PG	Parental guidance suggested. May not be suitable...	70	2014	94.0	80

- Before performing groupby and split-apply-combine procedure, let's look a bit more closely at the data to make sure it's what we think it is and to deal with missing values.
- Note that there is a missing value NaN in the user_rating_score of the second row (row 1).

Summarising data with plots and statistics

Case study: Netflix

- Applying the pandas DataFrame `.info()` shows that we have 1000 rows and 7 columns of data.
- Also that the column of interest, `user_rating_score`, has only 605 non-null values.
- This means that there are 395 missing values.

```
# Check out info of DataFrame  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1000 entries, 0 to 999  
Data columns (total 7 columns):  
title                1000 non-null object  
rating               1000 non-null object  
ratinglevel          941 non-null object  
ratingdescription    1000 non-null int64  
release_year         1000 non-null int64  
user_rating_score    605 non-null float64  
user_rating_size     1000 non-null int64  
dtypes: float64(1), int64(3), object(3)  
memory usage: 54.8+ KB
```

Summarising data with plots and statistics

Case study: Netflix

- We can drop rows that have any missing values, drop any duplicate rows and build a pairplot of the DataFrame using seaborn in order to get a visual sense of the data.
- We'll color the data by the 'rating' column.
- Check out the plots and see what information you can get from them.

```
# Drop rows with missing values and drop duplicate
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)

# Visualize pairplot of df
sns.pairplot(df, hue='rating');
```

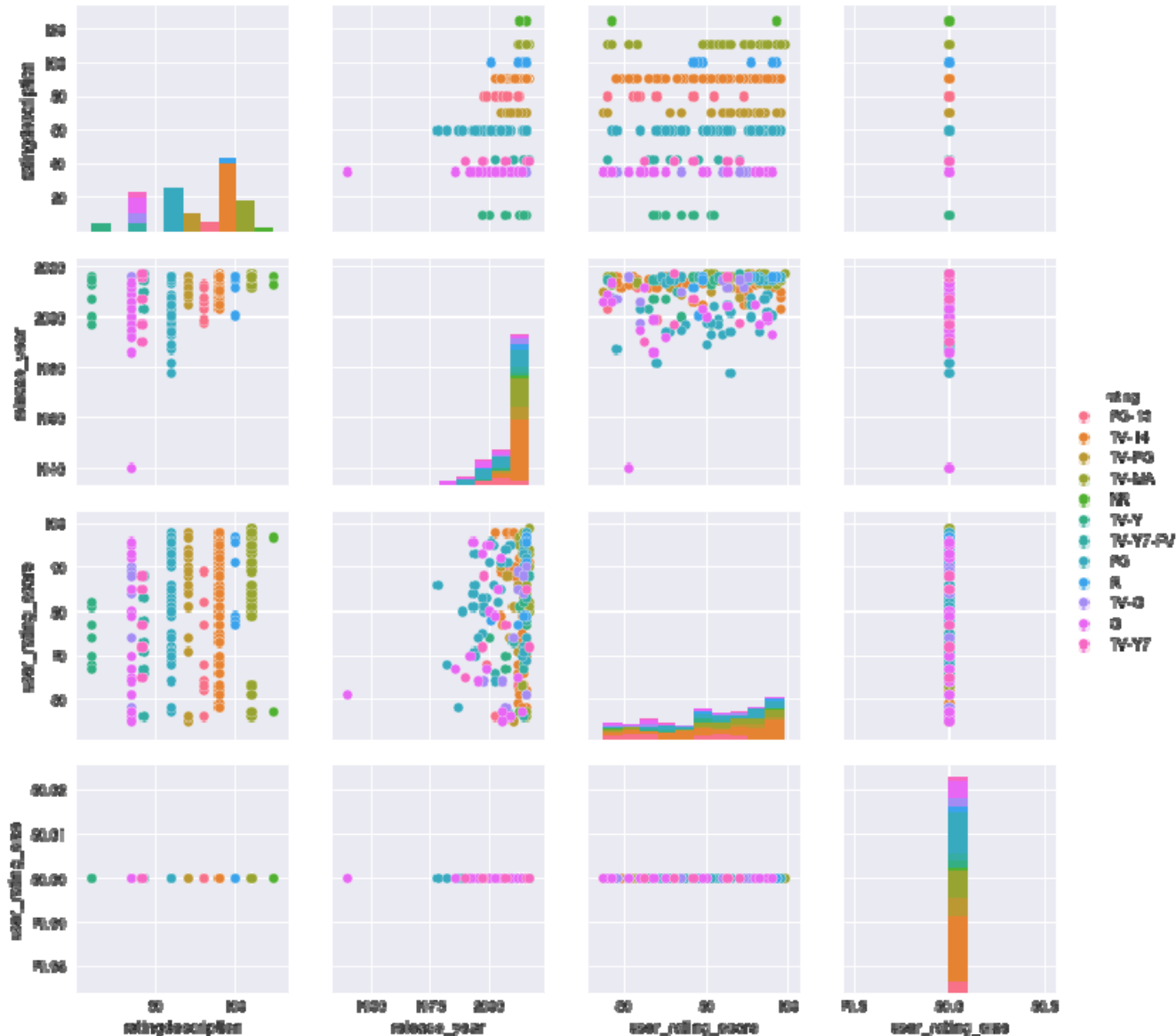
Summarising data with plots and statistics

Case study: Netflix

```
sns.pairplot(df, hue='rating');
```

Data columns (total 7 columns):

- title 1000 non-null object
- rating 1000 non-null object
- ratinglevel 941 non-null object
- ratingdescription 1000 non-null int64
- release_year 1000 non-null int64
- user_rating_score 605 non-null float64
- user_rating_size 1000 non-null int64



Summarising data with plots and statistics

Case study: Netflix

Check out several summary statistics of the DataFrame using the `.describe()` method:

```
# Get summary stats of df  
df.describe()
```

	ratingdescription	release_year	user_rating_score	user_rating_size
count	246.000000	246.000000	246.000000	246.0
mean	73.556911	2010.272358	81.390244	80.0
std	26.616145	8.887219	12.677883	0.0
min	10.000000	1940.000000	55.000000	80.0
25%	60.000000	2007.000000	71.000000	80.0
50%	80.000000	2015.000000	83.500000	80.0
75%	90.000000	2016.000000	92.750000	80.0
max	124.000000	2017.000000	99.000000	80.0

Groupby and split-apply-combine to answer the question

Step 1. Split

- Now that we've checked out data, it's time for the main part.
- We'll first use a *groupby* method to split the data into groups, *where each group is the set of movies released in a given year*.
- This is the *split* in *split-apply-combine*:

```
# Group by year  
df_by_year = df.groupby('release_year')
```

- This creates a groupby object:

```
# Check type of GroupBy object  
type(df_by_year)
```

- Output:

```
pandas.core.groupby.DataFrameGroupBy
```

Groupby and split-apply-combine to answer the question

Step 2. Apply

- Remember that the `.describe()` method for a `DataFrame` returns summary statistics for numeric columns.
- Well, the `.describe()` method for `DataFrameGroupBy` objects returns summary statistics for each group in the split.
- In our case, it's for each **release_year**.
- This is an example of the apply in split-apply-combine: we're applying the `.describe()` method to each group in the groupby.
- Do this and print the first 5 rows of the result:

Groupby and split-apply-combine to answer the question

```
# Summary stats over years
df_by_year.describe().head()
```

	ratingdescription								user_rating_score					user_rating_size							
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	max
release_year																					
1940	1	35	NaN	35	35	35	35	35	1	61...		61	61	1	80	NaN	80	80	80	80	80
1978	1	60	NaN	60	60	60	60	60	1	86...		86	86	1	80	NaN	80	80	80	80	80
1982	1	60	NaN	60	60	60	60	60	1	68...		68	68	1	80	NaN	80	80	80	80	80
1986	1	35	NaN	35	35	35	35	35	1	67...		67	67	1	80	NaN	80	80	80	80	80
1987	1	60	NaN	60	60	60	60	60	1	58...		58	58	1	80	NaN	80	80	80	80	80
5 rows × 24 columns																					

Groupby and split-apply-combine to answer the question

If we want to see what the grouping looks like, you can pass the groupby object to the function `list()`:

```
# Cast grouping as a list and check out one year  
list(df_by_year)[10]
```

```
(1995,      title rating      ratinglevel \  
766      Balto    G General Audiences. Suitable for all ages.  
967 Heavyweights  PG      some rude language and pranks  
  
Rating  description  release_year  user_rating_score  user_rating_size  
766      35      1995      64.0      80  
967      60      1995      74.0      80 )
```

Groupby and split-apply-combine to answer the question

Step 3. Combine

- Let's say that we wanted the mean or median user_rating_score for each year.
- Then we can apply the .mean() or .median() method, respectively, to the groupby object and **'combine'** these into a new DataFrame.

```
# Get median values by year and print first 5 rows
df_med_by_year = df_by_year.median()
df_med_by_year.head()
```

	ratingdescription	user_rating_score	user_rating_size
release_year			
1940	35	61	80
1978	60	86	80
1982	60	68	80
1986	35	67	80
1987	60	58	80

Groupby and split-apply-combine to answer the question

- There's an important subtlety concerning the index of the DataFrame `df_med_by_year`.
- Recall that the index of a DataFrame consists of the row labels.
- Check out the index of the original DataFrame `df`:

```
# Print index of df
print(df.index)
```

```
Int64Index([ 0,  2,  3,  4,  5,  6,  7,  8,  9, 10,
            ...
            908, 911, 917, 931, 962, 966, 967, 972, 973, 979],
           dtype='int64', length=246)
```

- This index consists of the original row numbers, labelled by integers.
- '1' is missing as we dropped some rows above.

Groupby and split-apply-combine to answer the question

- The index of `df_med_by_year` consists of the values in the original column that we grouped by, the years from `release_year`:

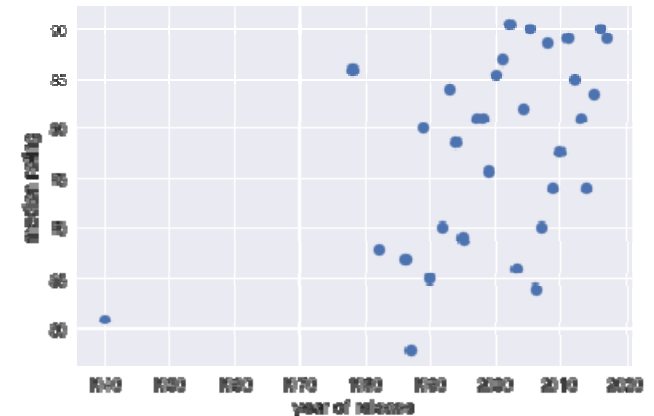
```
# Print index  
print(df_med_by_year.index)
```

```
Int64Index([1940, 1978, 1982, 1986, 1987, 1989, 1990, 1992, 1993, 1994, 1995,  
            1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007,  
            2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017],  
            dtype='int64', name='release_year')
```

- We're interested in the `user_rating_score` column, which contains the median rating for each year.
- We can slice out the `user_rating_score` column of `df_med_by_year` and plot it as a function of the year (given by the index of the DataFrame `df_rat_by_year`):

Groupby and split-apply-combine to answer the question

```
# Slice out user rating and plot
df_rat_by_year = df_med_by_year['user_rating_score']
plt.scatter(df_rat_by_year.index, df_rat_by_year)
plt.xlabel('year of release')
plt.ylabel('median rating');
```



Looking at the figure, the median rating definitely increases over time.

Review Inspeksi Data

- Mari review kembali terkait dengan inspeksi data yang pernah dilakukan pada pembahasan sebelumnya.
- Menggunakan dataset **global_air_quality_4000rows.csv**

[1] Load data dari csv

```
import pandas as pd

# Load data global_air_quality.csv
global_air_quality = pd.read_csv('/path/global_air_quality_4000rows.csv')
print('Lima data teratas:\n', global_air_quality.head())
```

```
Lima data teratas:
   location  ... averaged_over_in_hours
0  MOBILE-KICKAPOO  ...             1.00
1  Oxford St Ebbes  ...             1.00
2  BROADWAY (South)  ...             1.00
3  Deen Dayal Nagar, Sagar - MPPCB  ...             0.25
4  Manglerud  ...             1.00

[5 rows x 11 columns]
```

[2] Melakukan pengecekan terhadap data

```
5 # Melakukan pengecekan terhadap data
6 print('Info global_air_quality:\n', global_air_quality.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 11 columns):
location          3997 non-null object
city              3966 non-null object
country           4000 non-null object
pollutant         4000 non-null object
value             4000 non-null float64
timestamp         4000 non-null object
unit              4000 non-null object
source_name       4000 non-null object
latitude          4000 non-null float64
longitude         4000 non-null float64
averaged_over_in_hours 3634 non-null float64
dtypes: float64(4), object(7)
memory usage: 343.8+ KB
```

[3] Melakukan count tanpa groupby

```
7 # Melakukan count tanpa groupby
8 print('Count tanpa groupby:\n', global_air_quality.count())
```

```
Count tanpa groupby:
location          3997
city              3966
country           4000
pollutant         4000
value             4000
timestamp         4000
unit              4000
source_name       4000
latitude          4000
longitude         4000
averaged_over_in_hours 3634
dtype: int64
```

[4] Melakukan count dengan groupby

```
9 # Melakukan count dengan groupby
10 gaq_groupby_count = global_air_quality.groupby('source_name').count()
11 print('Count dengan groupby (5 data teratas):\n',
      gaq_groupby_count.head())
```

```
Count dengan groupby (5 data teratas):
      location  city  ...  longitude  averaged_over_in_hours
source_name
ARPALAZIO      72   72  ...         72                    72
Agaar.mn       27   27  ...         27                     0
AirNow      1712  1681  ...      1715                1715
Andalucia      71   71  ...         71                    71
Anqing         4    4  ...          4                     4
```

```
[5 rows x 10 columns]
```

Terdapat perbedaan antara melakukan count dengan groupby dan tanpa groupby,

1. Terdapat index berdasarkan apa yang di specify as groupby (source_name)
2. Perhitungan data menjadi berdasarkan apa yang di specify as groupby
3. Overall, lebih mudah untuk membaca data summary yang telah di groupby

Tugas 1. Aggregation & GroupBy (1)

- Pastikan telah membaca dataset `global_air_quality.csv`, mencetak 5 data teratasnya, mengecek informasi dataset, menghitung jumlah data yang non-missing value, dan mengelompokkannya berdasarkan kolom `'source_name'`.
- Langkah secara berurutan sesuai dengan deskripsi adalah:
 1. `import pandas` sebagai aliasnya,
 2. load data dengan `pd.read_csv(...)`,
 3. tampilkan 5 data teratas dengan `.head()`,
 4. cek informasi dataset tersebut dengan `.info()`
 5. hitung jumlah non-missing value dengan `.count()`
 6. kelompokkan dengan menggunakan `.groupby('source_name').count()` dan cetak lima baris teratasnya.

```
import ____

# Load data global_air_quality.csv
global_air_quality =
pd.read_csv('/path/global_air_quality_4000rows.csv')
print('Lima data teratas:\n', ____)

# Melakukan pengecekan terhadap data
print('Info global_air_quality:\n', ____)

# Melakukan count tanpa groupby
print('Count tanpa groupby:\n', ____)

# Melakukan count dengan groupby
gaq_groupby_count = ____
print('Count dengan groupby (5 data teratas):\n', ____)
```

Groupby dan Agregasi dengan Fungsi Statistik Dasar (1)

- Pada bagian ini dibahas penerapan groupby dan fungsi agregasi **mean** dan **std** untuk menentukan nilai rata-rata dan standar deviasi dari masing-masing kelompok data dari dataset **global_air_quality_4000rows.csv** dan diassign sebagai variabel **gaq**.
- Dibuat variabel pollutant

```
import pandas as pd
```

```
# Load data global_air_quality.csv
```

```
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')
```

```
# Create variabel pollutant
```

```
pollutant = gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=[  
    'country', 'city'], columns='pollutant').fillna(0)
```

```
print('Data pollutant (5 teratas):\n', pollutant.head())
```

```
Data pollutant (5 teratas):
```

		value						
pollutant		bc	co	no2	o3	pm10	pm25	so2
country city								
AR	Buenos Aires	0.0	0.0	0.0	0.0	0.0	18.1	0.0
AU	Townsville	0.0	0.0	0.0	0.0	0.0	3.9	0.0
BA	Goražde	0.0	141.0	19.0	8.0	0.0	0.0	0.0
	Ilijaš	0.0	0.0	0.0	0.0	100.0	0.0	0.0
	Jajce	0.0	1508.0	25.0	6.0	9.0	0.0	0.0

[1] Group berdasarkan country dan terapkan agregasi mean.

Method `.mean()` yang diaplikasikan setelah penerapan method `.groupby()` ini digunakan untuk **mencari rata-rata dari tiap kelompok**.

```
7 # [1] Group berdasarkan country dan terapkan agregasi mean
8 pollutant_mean = pollutant.groupby('country').mean()
9 print('Rata-rata pollutant (5 teratas):\n', pollutant_mean.head())
```

```
Rata-rata pollutant (5 teratas):
      value
pollutant bc      co      no2  ...      pm10      pm25      so2
country
AR      0.0      0.000000  0.000000  ...  0.000000  18.100000  0.000000
AU      0.0      0.000000  0.000000  ...  0.000000   3.900000  0.000000
BA      0.0  475.833333  19.500000  ...  40.333333  0.000000  18.000000
CA      0.0      0.036818  0.000355  ...   0.836364   3.433601  0.000091
CL      0.0      0.000000  21.000000  ...  0.000000  0.000000  0.000000

[5 rows x 7 columns]
```

[2] Group berdasarkan **country** dan terapkan agregasi **std**.

Method **.std()** yang diaplikasikan setelah penerapan method **.groupby()** digunakan untuk **mencari standard deviasi (penyimpangan) dari tiap kelompok**.

```
10 # [2] Group berdasarkan country dan terapkan agregasi std
11 pollutant_std = pollutant.groupby('country').std().fillna(0)
12 print('Standar deviasi pollutant (5 teratas):\n', pollutant_std.head())
```

Standar deviasi pollutant (5 teratas):

	value						
pollutant	bc	co	no2	...	pm10	pm25	so2
country	...						
AR	0.0	0.000000	0.000000	...	0.000000	0.000000	0.000000
AU	0.0	0.000000	0.000000	...	0.000000	0.000000	0.000000
BA	0.0	536.925476	11.945711	...	46.701891	0.000000	30.403947
CA	0.0	0.089841	0.000750	...	1.713052	2.286509	0.000302
CL	0.0	0.000000	0.000000	...	0.000000	0.000000	0.000000

[5 rows x 7 columns]

Tugas 2. Aggregation & GroupBy (2)

- Pastikan telah meng-create variabel polutan seperti yang dicontohkan, kemudian melakukan pengelompokkan (groupby) berdasarkan kolom 'country' dan menerapkan agregasi mean dan std, serta mencetak 5 baris teratas masing-masingnya.
- Dataframe yang dikelompokkan dengan groupby memiliki syntax umum berupa:
[nama_dataframe].groupby([nama_kolom]).[nama_fungsi_agregasi]
dengan
[nama_dataframe] --> nama variabel dataframe
[nama_kolom] --> nama atau nama-mama kolom dari dataframe untuk di-groupby
[nama_fungsi_agregasi] --> nama fungsi agregasi yang diaplikasikan

```
import __

# Load data global_air_quality.csv
gaq = pd.read_csv(__)

# Create variabel pollutant
pollutant =
gaq[____].pivot_table(index=____,columns=____).fillna(0)
print('Data pollutant (5 teratas):\n', __)

# [1] Group berdasarkan country dan terapkan agregasi
mean
pollutant_mean = ____
print('Rata-rata pollutant (5 teratas):\n', __)

# [2] Group berdasarkan country dan terapkan agregasi
std
pollutant_std = ____
print('Standar deviasi pollutant (5 teratas):\n', __)
```

Groupby dan Agregasi dengan Fungsi Statistik Dasar (2)

[3] Group berdasarkan country dan terapkan agregasi sum

Penggunaan method `.sum()` setelah penerapan method `.groupby()` digunakan untuk mencari total nilai dari tiap kelompok.

```
7 # [3] Group berdasarkan country dan terapkan agregasi sum
8 pollutant_sum = pollutant.groupby('country').sum()
9 print('Total pollutant (5 teratas):\n', pollutant_sum.head())
```

```
Total pollutant (5 teratas):
value
pollutant  bc      co      no2      o3  pm10      pm25      so2
country
AR         0.0    0.000    0.0000    0.000000    0.0  18.100000    0.000
AU         0.0    0.000    0.0000    0.000000    0.0   3.900000    0.000
BA         0.0 2855.000  117.0000  35.000000  242.0   0.000000  108.000
CA         0.0    0.405    0.0039    0.285593    9.2  37.769608    0.001
CL         0.0    0.000   21.0000    0.000000    0.0   0.000000    0.000
```

Groupby dan Agregasi dengan Fungsi Statistik Dasar (2)

[4] Group berdasarkan **country** dan terapkan agregasi **nunique**.

Penggunaan method `.nunique()` setelah penerapan method `.groupby()` digunakan untuk mencari berapakah jumlah unique value dari tiap kelompok.

```
10 # [4] Group berdasarkan country dan terapkan agregasi nunique
11 pollutant_nunique = pollutant.groupby('country').nunique()
12 print('Jumlah unique value pollutant (5 teratas):\n', pollutant_nunique.head())
```

```
Jumlah unique value pollutant (5 teratas):
      value
country bc co no2 o3 pm10 pm25 so2
AR      1  1  1  1   1    1    1
AU      1  1  1  1   1    1    1
BA      1  6  6  4   5    1    4
CA      1  3  4  9   4   10    2
CL      1  1  1  1   1    1    1
```

Tugas 3. Aggregation & GroupBy (3)

- Pastikan dataset telah kamu baca dan kemudian create variabel pollutant dengan tipe DataFrame.
- Berdasarkan dataframe ini kita dapat mengelompokkan dengan nama kolom yang diinstruksikan dan melakukan fungsi agregasi:
 1. penjumlahan data, dan
 2. menghitung jumlah data yang uniktentunya dalam kelompok yang sama.
- Dataframe yang dikelompokkan dengan groupby memiliki syntax umum berupa
[nama_dataframe].groupby([nama_kolom]).[nama_fungsi_agregasi]
dengan
[nama_dataframe] --> nama variabel dataframe
[nama_kolom] --> nama atau nama-mama kolom dari dataframe untuk di-groupby
[nama_fungsi_agregasi] --> nama fungsi agregasi yang diaplikasikan untuk sub bab ini kita menggunakan sum dan nunique

```
import ____

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('____')

# Create variabel pollutant
pollutant =
gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=['country', 'city'], columns='pollutant').fillna(0)
print('Data pollutant (5 teratas):\n', pollutant.head())

# [3] Group berdasarkan country dan terapkan agregasi sum
pollutant_sum = ____
print('Total pollutant (5 teratas):\n',
pollutant_sum.head())

# [4] Group berdasarkan country dan terapkan agregasi
nunique
pollutant_nunique = ____
print('Jumlah unique value pollutant (5 teratas):\n',
pollutant_nunique.head())
```

Groupby dan Agregasi dengan Fungsi Statistik Dasar (3)

[5] Group berdasarkan **country** dan terapkan agregasi **min**.

Penggunaan method **.min()** setelah penerapan method **.groupby()** digunakan untuk **memunculkan nilai terkecil dari tiap kelompok**.

```
7 # [5] Group berdasarkan country dan terapkan agregasi min
8 pollutant_min = pollutant.groupby('country').min()
9 print('Nilai min pollutant (5 teratas):\n', pollutant_min.head())
```

```
Nilai min pollutant (5 teratas):
      value
pollutant  bc  co  no2  o3 pm10  pm25  so2
country
AR         0.0  0.0  0.0  0.0  0.0  18.1  0.0
AU         0.0  0.0  0.0  0.0  0.0   3.9  0.0
BA         0.0  0.0  0.0  0.0  0.0   0.0  0.0
CA         0.0  0.0  0.0  0.0  0.0   0.0  0.0
CL         0.0  0.0  21.0  0.0  0.0   0.0  0.0
```

Groupby dan Agregasi dengan Fungsi Statistik Dasar (3)

[6] Group berdasarkan **country** dan terapkan agregasi **max**.

Penggunaan method **.max()** setelah penerapan method **.groupby()** digunakan untuk memunculkan nilai terbesar dari tiap kelompok.

```
10 # [6] Group berdasarkan country dan terapkan agregasi max
11 pollutant_max = pollutant.groupby('country').max()
12 print('Nilai max pollutant (5 teratas):\n', pollutant_max.head())
```

```
Nilai max pollutant (5 teratas):
      value
pollutant bc      co      no2      o3  pm10  pm25  so2
country
AR        0.0    0.000  0.0000  0.000000  0.0   18.1  0.000
AU        0.0    0.000  0.0000  0.000000  0.0    3.9  0.000
BA        0.0  1508.000  32.0000  21.000000  100.0   0.0  78.000
CA        0.0    0.285  0.0024  0.048333   5.2    6.0  0.001
CL        0.0    0.000  21.0000  0.000000   0.0    0.0  0.000
```

Tugas 3. Aggregation & GroupBy (4)

- Gunakanlah method `.first()` dan `.last()` untuk aggregasi setelah penerapan `.groupby()` yang masing-masingnya bertujuan untuk memunculkan item pertama dan item terakhir dari tiap kelompok.
- Pastikan telah membuat dataframe yang diassign sebagai variabel pollutant dari dataset 'global_air_quality.csv'. Kemudian, kelompokkan data berdasarkan kolom yang diperintahkan dan menggunakan data pertama dan data terakhir untuk satu kelompok data yang sama sebagai fungsi aggregasinya.
- Dataframe yang dikelompokkan dengan groupby memiliki syntax umum berupa
[nama_dataframe].groupby([nama_kolom]).[nama_fungsi_aggregasi]
dengan
[nama_dataframe] --> nama variabel dataframe
[nama_kolom] --> nama atau nama-mama kolom dari dataframe untuk di-groupby
[nama_fungsi_aggregasi] --> nama fungsi aggregasi yang diaplikasikan untuk sub bab ini kita menggunakan first dan last.

```
import __

# Load data https://storage.googleapis.com/dqlab-
dataset/L04/global_air_quality_4000rows.csv
gaq = pd.read_csv(__)

# Create variabel pollutant
pollutant =
gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=['country', 'city'],
columns='pollutant').fillna(0)

print('Data pollutant (5 teratas):\n', pollutant.head())

# Group berdasarkan country dan terapkan aggregasi first
pollutant_first = __
print('Item pertama pollutant (5 teratas):\n', __)

# Group berdasarkan country dan terapkan aggregasi last
pollutant_last = __
print('Item terakhir pollutant (5 teratas):\n', __)
```

Tugas 4. Aggregation & GroupBy (4)

Output yang diharapkan

```
Item pertama pollutant (5 teratas):
      value
pollutant bc      co      no2      o3 pm10 pm25 so2
country
AR        0.0     0.000  0.0000  0.000000 0.0  18.1  0.0
AU        0.0     0.000  0.0000  0.000000 0.0   3.9  0.0
BA        0.0  141.000  19.0000  8.000000 0.0   0.0  0.0
CA        0.0   0.285  0.0024  0.048333 0.0   3.9  0.0
CL        0.0     0.000  21.0000  0.000000 0.0   0.0  0.0
Item terakhir pollutant (5 teratas):
      value
pollutant bc      co      no2      o3 pm10 pm25 so2
country
AR        0.0     0.0     0.0  0.000    0.0  18.1  0.0
AU        0.0     0.0     0.0  0.000    0.0   3.9  0.0
BA        0.0  292.0    29.0  0.000   96.0   0.0  78.0
CA        0.0     0.0     0.0  0.036    3.0   0.0  0.0
CL        0.0     0.0    21.0  0.000    0.0   0.0  0.0
```

Groupby dengan Multiple Aggregations

Menggunakan **groupby** dengan **multiple aggregations** yang berupa kombinasi antara beberapa fungsi.

```
import pandas as pd

# Load data global_air_quality.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')

# Create variabel pollutant
pollutant = gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=
['country', 'city'], columns='pollutant').fillna(0)
print('Data pollutant (5 teratas):\n', pollutant.head())

# Group berdasarkan country dan terapkan agregasi: min, median, mean, max
multiagg = pollutant.groupby('country').agg(['min', 'median', 'mean', 'max'])
print('Multiple aggregations (5 teratas):\n', multiagg.head())
```

```
Multiple aggregations (5 teratas):
value
bc          co          ...          pm25          so2
max first last max first ... first last max first last
country
AR    0.0  0.0  0.0    0.000  0.000  ...  18.1  18.1    0.000  0.0  0.0
AU    0.0  0.0  0.0    0.000  0.000  ...   3.9   3.9    0.000  0.0  0.0
BA    0.0  0.0  0.0  1508.000  141.000  ...   0.0   0.0   78.000  0.0  78.0
CA    0.0  0.0  0.0    0.285  0.285  ...   3.9   0.0    0.001  0.0  0.0
CL    0.0  0.0  0.0    0.000  0.000  ...   0.0   0.0    0.000  0.0  0.0

[5 rows x 21 columns]
```

Tugas 5. Aggregation & GroupBy (5)

- Terapkanlah multiple aggregations pada dataframe pollutant dengan fungsi aggregasinya adalah 'min', 'median', 'mean', 'max'.
- Pastikan telah menggunakan fungsi aggregasi sesuai dengan urutan yang diinstruksikan yaitu 'min', 'median', 'mean', dan 'max' setelah melakukan proses groupby.
- Dataframe yang dikelompokkan dengan groupby memiliki syntax umum berupa [nama_dataframe].groupby([nama_kolom]).agg([list_fungsi_aggregasi]) dengan [nama_dataframe] --> nama variabel dataframe [nama_kolom] --> nama atau nama-mama kolom dari dataframe untuk di-groupby [list_fungsi_aggregasi] --> list dari beberapa buah nama fungsi aggregasi yang akan diaplikasikan.

```
import __

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv(__)

# Create variabel pollutant
pollutant =
gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=
['country', 'city'], columns='pollutant').fillna(0)
print('Data pollutant (5 teratas):\n', pollutant.head())

# Group berdasarkan country dan terapkan aggregasi: min,
median, mean, max
multiagg = __
print('Multiple aggregations (5 teratas):\n', __)
```

```
Multiple aggregations (5 teratas):
      value      ...      pm25  so2
      bc      ...      max  min median      mean      max
country min median mean max min ...
AR      0.0      0.0 0.0 0.0 0.0 ... 18.1 0.0      0.0 0.000000 0.000
AU      0.0      0.0 0.0 0.0 0.0 ...  3.9 0.0      0.0 0.000000 0.000
BA      0.0      0.0 0.0 0.0 0.0 ...  0.0 0.0      5.5 18.000000 78.000
CA      0.0      0.0 0.0 0.0 0.0 ...  6.0 0.0      0.0 0.000091 0.001
CL      0.0      0.0 0.0 0.0 0.0 ...  0.0 0.0      0.0 0.000000 0.000

[5 rows x 28 columns]
```

Groupby dengan Custom Aggregations

- Dengan membuat sebuah **Python function (user defined)**, kita dapat menggunakannya sebagai **custom aggregation** pada dataframe yang telah di groupby.

```
import pandas as pd

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')
# Create variabel pollutant
pollutant = gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=
['country', 'city'], columns='pollutant').fillna(0)
# Create sebuah function: data_range
def data_range(series):
    return series.max() - series.min()
# Group berdasarkan country dan terapkan aggregasi dari function: data_range
custom_agg = pollutant.groupby('country').agg(data_range)
print('Custom aggregation (5 teratas):\n', custom_agg.head())
```

Custom aggregation (5 teratas):							
pollutant	value						
	bc	co	no2	o3	pm10	pm25	so2
country							
AR	0.0	0.000	0.0000	0.000000	0.0	0.0	0.000
AU	0.0	0.000	0.0000	0.000000	0.0	0.0	0.000
BA	0.0	1508.000	32.0000	21.000000	100.0	0.0	78.000
CA	0.0	0.285	0.0024	0.048333	5.2	6.0	0.001
CL	0.0	0.000	0.0000	0.000000	0.0	0.0	0.000

- di sini dibuat sebuah fungsi untuk menentukan range pada setiap kelompok.

Tugas 6. Aggregation & GroupBy (6)

- Tentukanlah inter quartile range (IQR) pada setiap kelompok data, dan kemudian tampilkanlah 5 data teratas saja.
- Pastikan telah menyelesaikan membuat sebuah fungsi iqr yang digunakan nanti sebagai fungsi agregasi setelah menerapkan proses pengelompokan dengan groupby.
- Fungsi iqr menggunakan series sebagai input argument-nya dan outputnya adalah IQR.
- Gunakan fungsi ini pada agregasi setelah pengelompokan, yaitu
[nama_dataframe].groupby([nama_kolom]).agg(iqr)

Custom aggregation (5 teratas):

	value						
pollutant	bc	co	no2	o3	pm10	pm25	so2
country							
AR	0.0	0.00	0.00000	0.000000	0.0	0.000000	0.0
AU	0.0	0.00	0.00000	0.000000	0.0	0.000000	0.0
BA	0.0	297.75	14.25000	7.500000	79.0	0.000000	17.0
CA	0.0	0.00	0.00025	0.035708	0.5	3.876471	0.0
CL	0.0	0.00	0.00000	0.000000	0.0	0.000000	0.0

```
import ____

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv(____)

# Create variabel pollutant
pollutant =
gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=
['country', 'city'], columns='pollutant').fillna(0)

# Create sebuah function: iqr
def iqr(____):
    Q1 = series.__(____)
    Q3 = series.__(____)
    return ____

# Group berdasarkan country dan terapkan agregasi dari
function: iqr
custom_agg = ____
print('Custom aggregation (5 teratas):\n', ____)
```

Groupby dengan Custom Aggregations by dict

- Penggunaan custom aggregation lainnya pada dataframe yang telah di groupby dapat dilakukan dengan mengirimkan argument berupa sebuah dict yang berisi 'key' dict sebagai nama kolomnya dan 'value' dict adalah fungsi untuk aggregasi, baik user defined function atau function yang telah tersedia.

- Persiapan dataset:

```
import pandas as pd

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')

# Create variabel pollutant
pollutant = gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=
['country', 'city'], columns='pollutant').fillna(0)
print('Data pollutant (5 teratas):\n', pollutant.head())
```

- Berikut ini dataset yang akan dikenakan teknik custom aggregation dengan menggunakan dict ini yaitu

```
Data pollutant (5 teratas):
value
pollutant      bc      co      no2      o3      pm10      pm25      so2
country city
AR      Buenos Aires  0.0      0.0      0.0      0.0      0.0      18.1      0.0
AU      Townsville    0.0      0.0      0.0      0.0      0.0      3.9      0.0
BA      Goražde        0.0     141.0     19.0      8.0      0.0      0.0      0.0
      Ilijaš          0.0      0.0      0.0      0.0     100.0      0.0      0.0
      Jajce           0.0     1508.0     25.0      6.0      9.0      0.0      0.0
```

Groupby dengan Custom Aggregations by dict

- Akan mengaplikasikan teknik custom aggregation pada kolom 'o3' dan 'so' dengan fungsi aggregasi masing-masingnya adalah 'max' dan 'data_range'.
- Fungsi 'data_range' ini merupakan fungsi yang didefinisikan sendiri (user-defined) untuk menentukan jangkauan (range) data.

```
7 # Function range of data
8 def data_range(series):
9     return series.max() - series.min()
10 # Create custom aggregation using dict
11 custom_agg_dict = pollutant['value'][['o3', 'so2']].groupby('country').agg({
12     'o3': 'max',
13     'so2': data_range
14 })
15 print('\nCetak 5 data teratas custom_agg_dict:\n', custom_agg_dict.head())
```

```
Cetak 5 data teratas custom_agg_dict:
              so2              o3
country
AR           0.000    0.000000
AU           0.000    0.000000
BA          78.000   21.000000
CA           0.001    0.048333
CL           0.000    0.000000
```

Tugas 7. Aggregation & GroupBy (6)

- Dengan dataset yang masih sama seperti tersedia di code editor, tentukanlah median untuk kolom 'pm10' serta iqr untuk kolom 'pm25' dan 'so2'.
- Tampilkan pula 5 data teratas saja.
- Untuk fungsi iqr, dapat dibuat return sebagai hasil pengurangan method .quantile(0.75) dan .quantile(0.25) pada argument series.
- Pastikan sudah membaca dataset, membuat fungsi untuk menghitung inter-quartile range, dan mencetak 5 data teratas dari output custom aggregation using dict.

```
import pandas as pd

# Load data https://storage.googleapis.com/dqlab-
dataset/L04/global_air_quality_4000rows.csv
gaq = pd.read_csv(____)

# Create variabel pollutant
pollutant =
gaq[['country', 'city', 'pollutant', 'value']].pivot_table(index=
['country', 'city'], columns='pollutant').fillna(0)
print('Data pollutant (5 teratas):\n', pollutant.head())

# Function IQR
def iqr(____):
    return series.____ - ____

# Create custom aggregation using dict
custom_agg_dict =
pollutant['value'][[____]].groupby('country').agg({
    ____:____,
    ____:____,
    ____:____
})
print('\nCetak 5 data teratas custom_agg_dict:\n', ____)
```

Tugas 8. Aggregation & GroupBy (6)

- Diberikan file 'global_air_quality.csv' jika kode berikut dieksekusi

```
1 import pandas as pd
2 # Load dataset global_air_quality.csv
3 gaq = pd.read_csv('global_air_quality.csv')
4 gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
5 gaq = gaq.set_index('timestamp')
6 pollutant = gaq.groupby('pollutant')['value'].mean().sort_values(ascending=False)
7 print(pollutant)
```

```
pollutant
pm10      39.667325
co        23.576613
pm25      12.248441
bc         7.086669
so2        6.017561
o3         2.368129
no2        -4.874572
Name: value, dtype: float64
```

- Pollutant manakah yang mempunyai rata-rata nilai emisi tertinggi di dunia?
- Asumsi: negara yang termasuk di file tersebut sudah semua negara di dunia.

1. Aggregation & GroupBy

2. Time Series in Pandas

Pendahuluan: Time Series in Pandas

- Time series merupakan kumpulan dataset yang indexnya berupa waktu.
- Data seperti ini umum dijumpai pada proses yang melibatkan suatu event.
- Pada bagian ini akan dipelajari dataset dengan index waktu ini atau dikenal dengan time series.
- Ketika berurusan dengan big data, biasanya akan selalu terdapat entitas waktu kapan suatu event itu terjadi.
- Waktu ini digunakan untuk mengenali event itu terjadi pada suatu frame tertentu dan dapat dijadikan index yang membuat setiap event menjadi unique.

Basic format datetime menurut ISO 8601 yaitu **YYYY-mm-dd HH:MM:SS**.

4.1. Time Series in Pandas:

Load Dataset as Time Series

- Terkadang Pandas salah mengenali **object datetime** menjadi **object string** dan pandas menjadi tidak bisa memanfaatkan full benefit dari time index.

Basic format datetime menurut ISO 8601: **YYYY-mm-dd HH:MM:SS**.

- Terdapat beberapa cara untuk mengubah kolom waktu menjadi format yang benar, yaitu **datetime object** di Pandas.

4.1. Time Series in Pandas:

Load Dataset as Time Series

[1] **read_csv**, terdapat keyword argument 'parse_dates', yang jika di set True dan set index untuk kolom waktu tersebut maka kolom datetime tersebut akan ditransformasikan sebagai datetime Pandas dan menjadi index

```
import pandas as pd

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv',
parse_dates=True, index_col='timestamp')

# Cetak 5 data teratas
print(gaq.head())
# Cetak info dari dataframe gaq
print('info')
print(gaq.info())
```

```
timestamp location ... averaged_over_in_hours
2017-01-18 16:00:00+00:00 MOBILE-KICKAPOO ... 1.00
2020-04-07 20:00:00+00:00 Oxford St Ebbes ... 1.00
2020-04-07 19:00:00+00:00 BROADWAY (South) ... 1.00
2020-04-07 18:30:00+00:00 Deen Dayal Nagar, Sagar - MPPCB ... 0.25
2020-04-07 20:00:00+00:00 Manglerud ... 1.00

[5 rows x 10 columns]
info
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4000 entries, 2017-01-18 16:00:00+00:00 to 2017-01-08 08:00:00+00:00
Data columns (total 10 columns):
location          3997 non-null object
city              3966 non-null object
country           4000 non-null object
pollutant         4000 non-null object
value             4000 non-null float64
unit              4000 non-null object
source_name       4000 non-null object
latitude          4000 non-null float64
longitude         4000 non-null float64
averaged_over_in_hours 3634 non-null float64
dtypes: float64(4), object(6)
memory usage: 343.8+ KB
```

Tugas 9. Load Dataset as Time Series (1)

- Pastikan telah melakukan proses sesuai dengan yang telah diinstruksikan,
 1. memanggil library pandas sebagai aliasnya
 2. membaca dataset dengan `pd.read_csv()`
 3. mencetak 5 baris teratas dan info dari dataframe.
- Untuk membaca data dengan `pd.read_csv` agar secara langsung dikenali sebagai dataframe dengan index berupa time series, keyword argument `parse_dates=True`, `index_col='timestamp'` dapat digunakan.

```
import ____

# Load dataset
global_air_quality_4000rows.csv
gaq = pd.read_csv(____, ____=____, ____=____)

# Cetak 5 data teratas
print(____)

# Cetak info dari dataframe gaq
print('info')
print(____)
```

4.2. Time Series in Pandas: Convert to Datetime

- Jika dataset yang dibaca **terlanjur di load dengan `pd.read_csv` dan Pandas salah mengenali object datetime menjadi object string**, maka tipe data object tersebut **dapat diubah menjadi format datetime**.
- **[2] `pd.to_datetime`** digunakan untuk mentransform salah satu kolom di dataframe menjadi datetime Pandas dan kemudian set menjadi index.
- Dataframe awal indexnya masih berupa urutan bilangan bulat dari nol. Kemudian dengan menerapkan `pd.to_datetime` dan `set_index`, dataframe sudah memiliki index berupa datetime.

```
import pandas as pd
```

```
# Load data global_air_quality_4000rows.csv
```

```
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')
```

```
# Cetak 5 data teratas
```

```
print('Sebelum diubah dalam format datetime:\n', gaq.head())
```

```
# Ubah menjadi datetime
```

```
gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
```

```
gaq = gaq.set_index('timestamp')
```

```
# Cetak 5 data teratas
```

```
print('Sesudah diubah dalam format datetime:\n', gaq.head())
```

Sebelum diubah dalam format datetime:

	location	...	averaged_over_in_hours
0	MOBILE-KICKAPOO	...	1.00
1	Oxford St Ebbes	...	1.00
2	BROADWAY (South)	...	1.00
3	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
4	Manglerud	...	1.00

[5 rows x 11 columns]

Sesudah diubah dalam format datetime:

timestamp	location	...	averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...	1.00
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...	1.00
2020-04-07 19:00:00+00:00	BROADWAY (South)	...	1.00
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
2020-04-07 20:00:00+00:00	Manglerud	...	1.00

[5 rows x 10 columns]

Tugas 10. Load Dataset as Time Series (2)

- Pastikan membaca data secara default (tidak menggunakan keyword argument seperti pada subbab sebelumnya).
- Setelah itu, kolom dataframe yang memiliki data berformat datetime diubah agar bertipe datetime64 dan kemudian dijadikan index.
- Untuk kasus ini dataframe yang telah dibaca diubah kolom 'timestamp'-nya menjadi datetime64 melalui
[nama_dataframe][nama_kolom] =
pd.to_datetime([nama_dataframe][nama_kolom])
dan [nama_kolom] tersebut dijadikan sebagai index dari dataframe melalui
[nama_dataframe] =
[nama_dataframe].set_index([nama_kolom])

```
import ____

# Load dataset global_air_quality_4000rows.csv
gaq = pd.read_csv(____)

# Cetak 5 data teratas
print('Sebelum diubah dalam format datetime:\n', ____)
```



```
# Ubah menjadi datetime
gaq['timestamp'] = ____
gaq = ____

# Cetak 5 data teratas
print('Sesudah diubah dalam format datetime:\n', ____)
```

4.3. Time Series in Pandas:

Resampling untuk Time Series Data

Pandas time series juga memiliki fungsi **resampling** yang dapat berguna untuk:

[1] Downsampling

- Mengurangi baris datetime menjadi frekuensi yang lebih lambat, bisa dibilang juga mengurangi rows dataset menjadi lebih sedikit
- Contoh: mengubah kolom datetime yang awalnya **daily** menjadi **monthly**

[2] Upsampling

- Kebalikan dari downsampling, menambah baris datetime menjadi frekuensi yang lebih cepat, menambah rows dataset dengan membuat kolom datetime menjadi lebih detail
- Contoh: mengubah kolom datetime yang awalnya **daily** menjadi **hourly**

4.3. Time Series in Pandas:

Resampling untuk Time Series Data

- Dari Resampling ini, kemudian dapat menerapkan metode statistik untuk transform value data yang ada (ex: mean, sum, count, etc)
- Tabel resampling untuk time series:

Input	Description
'Min', 'T'	minute
'H'	hour
'D'	day
'B'	Business day
'W'	week
'M'	month
'Q'	quarter
'A'	year

4.4. Time Series in Pandas: Downsampling Data

- Sekarang akan dicoba melakukan proses downsampling pada dataset 'global_air_quality_4000rows.csv' yang telah di load sebelumnya.
- Perhatikan dataset awal:

```
import pandas as pd
```

```
# Load data global_air_quality_4000rows.csv
```

```
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')
```

```
gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
```

```
gaq = gaq.set_index('timestamp')
```

```
print('Dataset sebelum di-downsampling (5 teratas):\n', gaq.head())
```

Dataset sebelum di-downsampling (5 teratas):

timestamp	location	...	averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...	1.00
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...	1.00
2020-04-07 19:00:00+00:00	BROADWAY (South)	...	1.00
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
2020-04-07 20:00:00+00:00	Manglerud	...	1.00

[5 rows x 10 columns]

Melakukan downsampling dari

[1] Daily to monthly

```
7 # [1] Downsampling dari daily to monthly dan kita hitung rata-rata untuk sebulan
```

```
8 gaq_monthly = gaq.resample('M').mean()
```

```
9 print('Downsampling daily to monthly - mean (5 teratas):\n', gaq_monthly.head())
```

Downsampling daily to monthly - mean (5 teratas):

timestamp	value	latitude	longitude	averaged_over_in_hours
2014-05-31 00:00:00+00:00	9.7	14.635	121.080	1.0
2014-06-30 00:00:00+00:00	NaN	NaN	NaN	NaN
2014-07-31 00:00:00+00:00	NaN	NaN	NaN	NaN
2014-08-31 00:00:00+00:00	7.1	37.132	-86.148	1.0
2014-09-30 00:00:00+00:00	NaN	NaN	NaN	NaN

4.4. Time Series in Pandas: [Downsampling Data](#)

[2] Daily to yearly

```
10 # [2] Downsampling dari daily to yearly dan kita hitung totalnya untuk setahun
11 gaq_yearly = gaq.resample('A').sum()
12 print('Downsampling daily to yearly - sum (5 teratas):\n', gaq_yearly.head())
```

```
Downsampling daily to yearly - sum (5 teratas):
      value  ... averaged_over_in_hours
timestamp  ...
2014-12-31 00:00:00+00:00    16.800000  ...          2.00
2015-12-31 00:00:00+00:00   319.209800  ...         30.00
2016-12-31 00:00:00+00:00  11356.389400  ...        172.25
2017-12-31 00:00:00+00:00  20181.246425  ...        147.75
2018-12-31 00:00:00+00:00 -282811.729944  ...       1284.50

[5 rows x 4 columns]
```

Tugas 11. Downsampling Data

- Kerjakanlah proses downsampling
 1. Daily to weekly dan apply max
 2. Daily to quaterly dan apply min
- Bacalah dataset secara default, kemudian ubah ke kolom 'timestamp' agar bertipe datetime64 dan gunakan kolom 'timestamp' sebagai index dari dataframe.
- Pastikan telah melakukan downsampling dari interval waktu dalam time series dataframe yang awalnya daily menjadi weekly dan quarterly.
- `pd.to_datetime` dapat digunakan untuk merubah kolom menjadi kolom bertipe datetime64
- method `.resample` dapat digunakan pada time series dataframe baik untuk downsampling atau upsampling berdasarkan string waktu yang telah ditunjukkan.
- Agregasi diperlukan setelah proses resampling yaitu sama halnya setelah proses grouping dengan `groupby`.

Tugas 12. Downsampling Data

```
import __

# Load dataset global_air_quality_4000rows.csv
gaq = pd.read_csv(__)
gaq['timestamp'] = pd.to_datetime(__)
gaq = gaq.set_index(__)
print('Dataset sebelum di-downsampling (5 teratas):\n', gaq.head())

# [1] Downsampling dari daily to weekly dan kita hitung maksimum untuk
seminggu
gaq_weekly = __
print('Downsampling daily to weekly - max (5 teratas):\n', __)

# [2] Downsampling dari daily to quaterly dan kita hitung minimumnya
untuk tiap quarter
gaq_quaterly = __
print('Downsampling daily to quaterly - min (5 teratas):\n', __)
```

```
Dataset sebelum di-downsampling (5 teratas):
```

timestamp	location	... averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...
2020-04-07 19:00:00+00:00	BROADWAY (South)	...
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...
2020-04-07 20:00:00+00:00	Manglerud	...

```
[5 rows x 10 columns]
Downsampling daily to weekly - max (5 teratas):
```

timestamp	country	pollutant	... longitude	averaged_over_in_hours
2014-05-18 00:00:00+00:00	PH	pm25	...	121.08
2014-05-25 00:00:00+00:00	NaN	NaN	...	NaN
2014-06-01 00:00:00+00:00	NaN	NaN	...	NaN
2014-06-08 00:00:00+00:00	NaN	NaN	...	NaN
2014-06-15 00:00:00+00:00	NaN	NaN	...	NaN

```
[5 rows x 8 columns]
Downsampling daily to quaterly - min (5 teratas):
```

timestamp	country	pollutant	... longitude	averaged_over_in_hours
2014-06-30 00:00:00+00:00	PH	pm25	...	121.080
2014-09-30 00:00:00+00:00	US	pm25	...	-86.148
2014-12-31 00:00:00+00:00	NaN	NaN	...	NaN
2015-03-31 00:00:00+00:00	NaN	NaN	...	NaN
2015-06-30 00:00:00+00:00	AR	pm25	...	-58.506

```
[5 rows x 8 columns]
```

4.5. Time Series in Pandas: Upsampling Data

- Di bagian sebelumnya kita telah melakukan proses downsampling.
- Proses upsampling untuk dataset yang sama.

```
import pandas as pd

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')

gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
gaq = gaq.set_index('timestamp')
print('Dataset sebelum di-upsampling (5 teratas):\n', gaq.head())
# Upsampling dari daily to hourly dan kita hitung reratanya
gaq_hourly = gaq.resample('H').mean()
print('Upsampling daily to hourly - mean (5 teratas):\n', gaq_hourly.head())
```

```
Dataset sebelum di-upsampling (5 teratas):
```

timestamp	location	...	averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...	1.00
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...	1.00
2020-04-07 19:00:00+00:00	BROADWAY (South)	...	1.00
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
2020-04-07 20:00:00+00:00	Manglerud	...	1.00

```
[5 rows x 10 columns]
Upsampling daily to hourly - count (5 teratas):
```

timestamp	value	latitude	longitude	averaged_over_in_hours
2014-05-13 03:00:00+00:00	9.7	14.635	121.08	1.0
2014-05-13 04:00:00+00:00	0.0	0.000	0.00	0.0
2014-05-13 05:00:00+00:00	0.0	0.000	0.00	0.0
2014-05-13 06:00:00+00:00	0.0	0.000	0.00	0.0
2014-05-13 07:00:00+00:00	0.0	0.000	0.00	0.0

Tugas 13. Upsampling Data

- Kerjakanlah proses upsampling daily to hourly dan apply mean di code editor dengan jalan mengisi kode yang tidak lengkap (___).
- Pastikan telah
 - membaca dataset secara default, kemudian ubah ke kolom 'timestamp' agar bertipe datetime64 dan gunakan kolom 'timestamp' sebagai index dari dataframe.
 - melakukan upsampling dari interval waktu dalam time series dataframe yang awalnya daily menjadi hourly dan kemudian terapkan fungsi agregasi seperti yang diinstruksikan.
 - pd.to_datetime dapat kamu gunakan untuk merubah kolom menjadi kolom bertipe datetime64
 - method .resample dapat kamu gunakan pada time series dataframe baik untuk downsampling atau upsampling berdasarkan string waktu yang telah ditunjukkan sebagai argumentnya.
 - Agregasi diperlukan setelah proses resampling yaitu sama halnya setelah proses grouping dengan groupby.

```
import __
# Load dataset global_air_quality_4000rows.csv
gaq = pd.read_csv(__)
gaq['timestamp'] = pd.to_datetime(__)
gaq = __
print('Dataset sebelum di-upsampling (5 teratas):\n', __)

# Upsampling dari daily to hourly dan kita hitung reratanya
gaq_hourly = __
print('Upsampling daily to hourly - mean (5 teratas):\n',
__)
```

```
Dataset sebelum di-upsampling (5 teratas):
```

timestamp	location	...	averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...	1.00
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...	1.00
2020-04-07 19:00:00+00:00	BROADWAY (South)	...	1.00
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
2020-04-07 20:00:00+00:00	Manglerud	...	1.00

```
[5 rows x 10 columns]
Upsampling daily to hourly - mean (5 teratas):
```

timestamp	value	latitude	longitude	averaged_over_in_hours
2014-05-13 03:00:00+00:00	9.7	14.635	121.08	1.0
2014-05-13 04:00:00+00:00	NaN	NaN	NaN	NaN
2014-05-13 05:00:00+00:00	NaN	NaN	NaN	NaN
2014-05-13 06:00:00+00:00	NaN	NaN	NaN	NaN
2014-05-13 07:00:00+00:00	NaN	NaN	NaN	NaN

4.6. Time Series in Pandas: Resampling by Frequency

- Melakukan resampling data (baik upsampling atau downsampling) berdasarkan frekuensi, misalnya sekali 2 minggu, tiap 12 jam, dsb.
- Kondisi awal data yang dimiliki :

```
import pandas as pd

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')

gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
gaq = gaq.set_index('timestamp')
print('Dataset sebelum di-resampling (5 teratas):\n', gaq.head())
```

```
Dataset sebelum di-resampling (5 teratas):
```

timestamp	location	...	averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...	1.00
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...	1.00
2020-04-07 19:00:00+00:00	BROADWAY (South)	...	1.00
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
2020-04-07 20:00:00+00:00	Manglerud	...	1.00

[5 rows x 10 columns]

Untuk memahaminya, perhatikan contoh berikut ini:

[1] Data ini downsampling dari daily to 2 weekly, kemudian dihitung rata-ratanya, jika ada nilai NaN maka dapat diisi dengan fillna method = 'ffill'

```
7 # [1] Downsampling dari daily to 2 weekly, hitung reratanya, dan fillna = 'ffill'
8 gaq_2weekly = gaq.resample('2W').mean().fillna(method='ffill')
9 print('Downsampling daily to 2 weekly - mean - ffill (5 teratas):\n',
    gaq_2weekly.head())
```

```
Downsampling daily to 2 weekly - mean - ffill (5 teratas):
```

timestamp	value	latitude	longitude	averaged_over_in_hours
2014-05-18 00:00:00+00:00	9.7	14.635	121.08	1.0
2014-06-01 00:00:00+00:00	9.7	14.635	121.08	1.0
2014-06-15 00:00:00+00:00	9.7	14.635	121.08	1.0
2014-06-29 00:00:00+00:00	9.7	14.635	121.08	1.0
2014-07-13 00:00:00+00:00	9.7	14.635	121.08	1.0

4.6. Time Series in Pandas: Resampling by Frequency

[2] Selanjutnya, data awal di upsampling dari daily to 8 hourly, kemudian hitung rata-ratanya, jika ada nilai NaN maka dapat di isi dengan fillna method = 'bfill'

```
10 # [2] Upsampling dari daily to 8 hourly, hitung reratanya, dan fillna = 'bfill'
11 gaq_8hourly = gaq.resample('8H').mean().fillna(method='bfill')
12 print('Upsampling daily to 8 hourly - mean - bfill (5 teratas):\n',
      gaq_8hourly.head())
```

```
Rata-rata pollutant (5 teratas):
      value
pollutant bc      co      no2  ...      pm10      pm25      so2
country
AR      0.0    0.000000  0.000000  ...    0.000000  18.100000  0.000000
AU      0.0    0.000000  0.000000  ...    0.000000   3.900000  0.000000
BA      0.0  475.833333  19.500000  ...   40.333333  0.000000  18.000000
CA      0.0    0.036818  0.000355  ...    0.836364   3.433601  0.000091
CL      0.0    0.000000  21.000000  ...    0.000000  0.000000  0.000000

[5 rows x 7 columns]
```

Tugas 14. Time Series in Pandas: Resampling by Frequency

- Kerjakanlah proses data awal resampling dari daily to bi-monthly, kemudian hitung rata-ratanya, jika ada nilai NaN maka dapat diisi dengan fillna method = 'bfill' serta mengisi kode yang tidak lengkap (___) di kode..
- Pastikan telah
 - membaca dataset secara default, kemudian ubah ke kolom 'timestamp' agar bertipe datetime64 dan gunakan kolom 'timestamp' sebagai index dari dataframe.
 - melakukan resampling dari interval waktu dalam time series dataframe yang awalnya daily menjadi bimonthly dan kemudian terapkan fungsi agregasi seperti yang diinstruksikan. Aplikasikan juga method .fillna untuk mengisi data-data hasil resampling berupa NaN.
 - pd.to_datetime dapat kamu gunakan untuk merubah kolom menjadi kolom bertipe datetime64
 - method .resample dapat kamu gunakan pada time series dataframe baik untuk downsampling atau upsampling berdasarkan string waktu yang telah ditunjukkan sebagai argumentnya dan juga berdasarkan pengali waktu yang disebut dengan frekuensi. Agregasi diperlukan setelah proses resampling yaitu sama halnya setelah proses grouping dengan groupby.
Catatan: kamu dapat mengisi datum yang hilang (missing value).

```
import __

# Load dataset global_air_quality_4000rows.csv
gaq = pd.read_csv(__)
gaq['timestamp'] = pd.to_datetime(__)
gaq = __
print('Dataset sebelum di-resampling (5 teratas):\n', __)

# Resample dari daily to 2 monthly, hitung reratanya, dan fillna = 'bfill'
gaq_2monthly = __
print('Resampling daily to 2 monthly - mean - ffill (5 teratas):\n', __)
```

```
Dataset sebelum di-resampling (5 teratas):
```

timestamp	location	...	averaged_over_in_hours
2017-01-18 16:00:00+00:00	MOBILE-KICKAPOO	...	1.00
2020-04-07 20:00:00+00:00	Oxford St Ebbes	...	1.00
2020-04-07 19:00:00+00:00	BROADWAY (South)	...	1.00
2020-04-07 18:30:00+00:00	Deen Dayal Nagar, Sagar - MPPCB	...	0.25
2020-04-07 20:00:00+00:00	Manglerud	...	1.00

```
[5 rows x 10 columns]
Resampling daily to 2 monthly - mean - ffill (5 teratas):
```

timestamp	value	latitude	longitude	averaged_over_in_hours
2014-05-31 00:00:00+00:00	9.7	14.635	121.080	1.0
2014-07-31 00:00:00+00:00	7.1	37.132	-86.148	1.0
2014-09-30 00:00:00+00:00	7.1	37.132	-86.148	1.0
2014-11-30 00:00:00+00:00	18.1	-34.560	-58.506	1.0
2015-01-31 00:00:00+00:00	18.1	-34.560	-58.506	1.0

4.7. Time Series in Pandas: Visualisasi

- Memvisualisasikan dataframe yang telah diolah (seperti yang telah dipelajari dari bagian sebelumnya) dapat menggunakan library matplotlib.
- Terdapat beberapa plot styles yang dapat diterapkan dalam pembuatan visualisasi di matplotlib.

Color	Marker	Line
b: blue	o: circle	: dotted
g: green	*: star	--: dashed
r: red	s: square	
c: cyan	+: plus	

- Visualisasi average jumlah masing-masing polutan berdasarkan bulan waktunya. Siapkan dataframe:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load data global_air_quality_4000rows.csv
gaq = pd.read_csv('/path/global_air_quality_4000rows.csv')

gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
gaq = gaq.set_index('timestamp')
```

4.7. Time Series in Pandas: Visualisasi

[1] Membuat pivot table yang menunjukkan waktu di baris nya dan masing-masing value dari pollutant nya dalam kolom

```
7 # [1] Membuat pivot table yang menunjukkan waktu di baris nya dan masing-masing
  value dari pollutant nya dalam kolom
8 gaq_viz =
  gaq[['pollutant', 'value']].reset_index().set_index(['timestamp', 'pollutant'])
9 gaq_viz = gaq_viz.pivot_table(index='timestamp', columns='pollutant',
  aggfunc='mean').fillna(0)
10 gaq_viz.columns = gaq_viz.columns.droplevel(0)
11 print('Data (5 teratas):\n', gaq_viz.head())
```

Data (5 teratas):

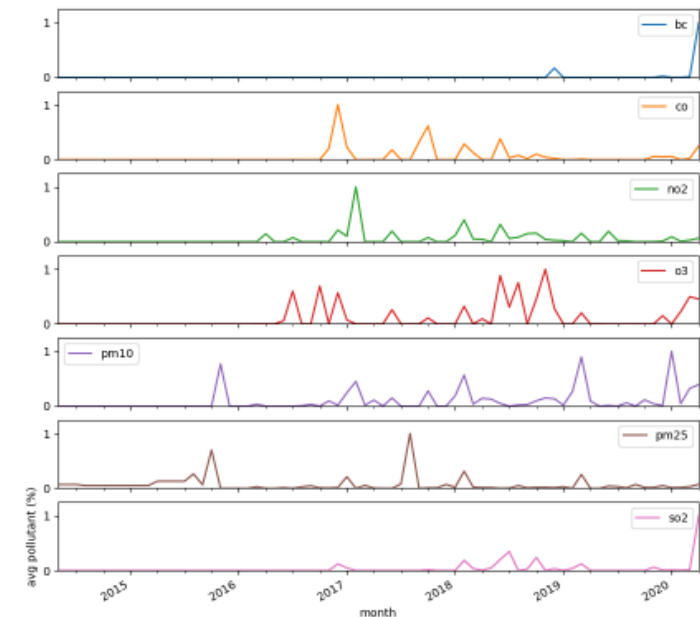
timestamp	bc	co	no2	o3	pm10	pm25	so2
2014-03-13 12:00:00+00:00	0.0	0.0	0.0	0.0	0.0	5.0	0.0
2014-05-13 03:00:00+00:00	0.0	0.0	0.0	0.0	0.0	9.7	0.0
2014-08-12 20:00:00+00:00	0.0	0.0	0.0	0.0	0.0	7.1	0.0
2014-09-26 00:30:00+00:00	0.0	0.0	0.0	0.0	0.0	27.8	0.0
2015-04-15 00:00:00+00:00	0.0	0.0	0.0	0.0	0.0	18.1	0.0

4.7. Time Series in Pandas: Visualisasi

[2] Membuat fungsi yang memberikan default value 0 ketika value nya di bawah 0 dan apply ke setiap elemen dari dataset tersebut, kemudian menampilkannya sebagai chart

```
12 # [2] Membuat fungsi yang memberikan default value 0 ketika value nya di bawah 0
    dan apply ke setiap elemen dari dataset tersebut, kemudian menampilkannya sebagai
    chart
13 def default_val(val):
14     if val < 0:
15         return 0
16     else:
17         return val
18 line1 = gaq_viz.resample('M').mean().ffill().applymap(lambda x:
    default_val(x)).apply(lambda x: x/x.max()) # default value if value < 0 then 0,
    kemudian menghasilkan % value = value/max(value)
19 line1.plot(
20     title = 'average value of each pollutant over months',
21     figsize = (10,10), #ukuran canvas 10px x 10px
22     ylim = (0,1.25), #memberikan batas tampilan y-axis hanya 0 sampai 125%
23     subplots = True #memecah plot menjadi beberapa bagian sesuai dengan jumlah
    kolom
24 )
25 plt.ylabel('avg pollutant (%)')
26 plt.xlabel('month')
27 plt.show()
```

Average value of each pollutant over months



Tugas 15. Time Series in Pandas: Visualisasi

- Pastikan telah mengikuti urutan-urutan seperti yang telah dicontohkan pada section Lesson.
Untuk membuat plot yang diminta, pastikan juga bahwa dataframanya telah diolah sedemikian rupa sehingga dapat menjawab persoalan dalam bentuk visualisasi, sehingga dapat dimengerti interpretasi dari data yang dimiliki.
- Pandas telah menyediakan akses untuk plotting langsung berdasarkan method yang bersumber dataframe yang diinstantiate.
`[nama_dataframe].plot(**kwargs)`
dengan kwargs merupakan keyword argument yang diperlukan untuk melayout plotting.
- Bisa merujuk ke <https://pandas.pydata.org/pandas-docs/version/0.24/reference/api/pandas.DataFrame.plot.html>

```

import ____
import ____ as plt

# Load dataset global_air_quality_4000rows.csv
gaq = pd.read_csv(____)
gaq['timestamp'] = pd.to_datetime(gaq['timestamp'])
gaq = gaq.set_index('timestamp')

# [1] Membuat pivot table yang menunjukkan waktu di baris nya dan masing-masing value dari pollutant nya dalam kolom
gaq_viz = gaq[____].reset_index().____(____)
gaq_viz = gaq_viz.____(index=____, columns=____, ____).____
gaq_viz.columns = gaq_viz.____
print('Data (5 teratas):\n', gaq_viz.head())

# [2] Membuat fungsi yang memberikan default value 0 ketika value nya di bawah 0 dan apply ke setiap elemen dari dataset
tersebut, kemudian menampilkannya sebagai chart
def ____ (val):
    if ____ < 0:
        return 0
    else:
        return ____
line1 = gaq_viz.resample(____).____.____.applymap(lambda x: ____).apply(lambda x: ____) # default value if value < 0 then 0, kemudian
menghasilkan % value = value/max(value)
line1.plot(
    title = ____,
    figsize = ____, #ukuran canvas 10px x 10px
    ylim = ____, #memberikan batas tampilan y-axis hanya 0 sampai 125%
    subplots = ____ #memecah plot menjadi beberapa bagian sesuai dengan jumlah kolom
)
plt.ylabel(____)
plt.xlabel(____)
plt.____

```

Tugas 16. Time Series in Pandas: Visualisasi

- Diberikan csv file: 'global_air_quality.csv'
- Bagaimana statement (kode) untuk mengetahui bagaimana trend dari nilai rata-rata pollutant SO2 (kolom pollutant) yang dihitung dari perhitungan value SO2 (kolom value) dalam tahunan ?

Referensi

- Data Manipulation with Pandas - DQLab.
- Groupby, split-apply-combine and pandas,
<https://www.datacamp.com/community/tutorials/pandas-split-apply-combine-groupby>, 2017