Learning Progress Review

By

Omicron

Anggota Kelompok Omicron

Anugrah Yazid Ghani

https://www.linkedin.com/in/anugrah-yazid-7253bb221/

Edo Mohammad Hadad Gibran

https://www.linkedin.com/in/edo-gibran-38505a142/

Fajar Achmad

https://www.linkedin.com/in/fajar-achmad-755945111/

Muhammad Fikri Fadila

https://www.linkedin.com/in/muhammad-fikri-fadila-a551161a6/

Daftar Isi

Advanced Pandas DataFrame 1

Database Programming 2

Application Programming Interfcae (API) 3

1

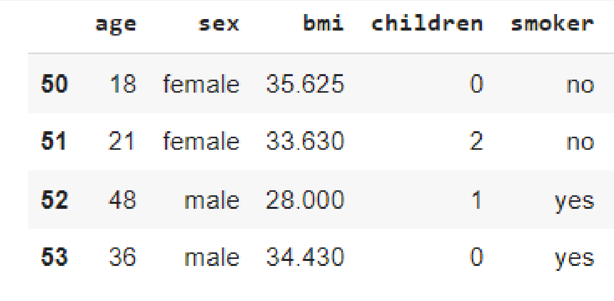
Advanced Pandas DataFrame

Indexing DataFrame

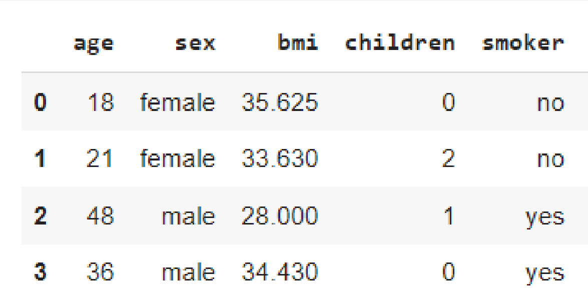
Indexing pada *dataframe* dengan menggunakan Pandas memiliki beberapa pengaplikasian di dalam dataset, antara lain :

1. Mengurutkan *index*
2. Membuat data pada variabel tertentu menjadi *index*

1. Mengurutkan *index*



	age	sex	bmi	children	smoker
50	18	female	35.625	0	no
51	21	female	33.630	2	no
52	48	male	28.000	1	yes
53	36	male	34.430	0	yes



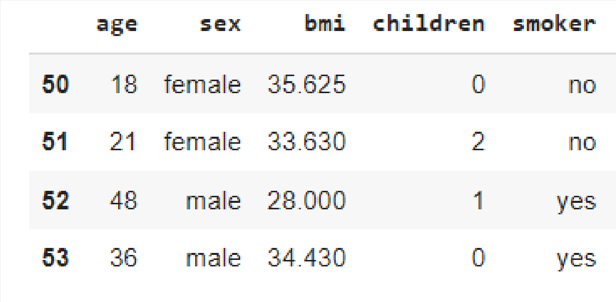
	age	sex	bmi	children	smoker
0	18	female	35.625	0	no
1	21	female	33.630	2	no
2	48	male	28.000	1	yes
3	36	male	34.430	0	yes

Mengurutkan *index* dimulai dari 0 :

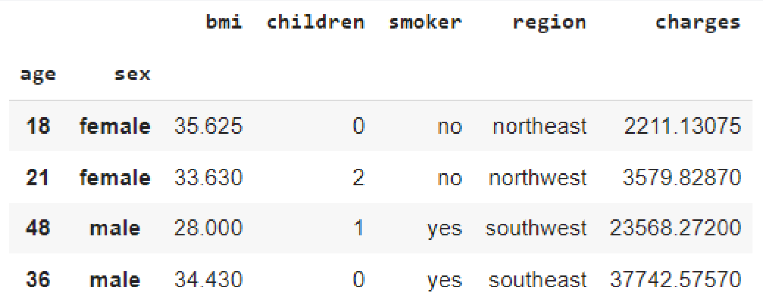
Syntax :

```
nama_dataset.reset_index(drop=True)
```

2. Membuat data pada variabel tertentu menjadi *index*



	age	sex	bmi	children	smoker
50	18	female	35.625	0	no
51	21	female	33.630	2	no
52	48	male	28.000	1	yes
53	36	male	34.430	0	yes



		bmi	children	smoker	region	charges
age	**sex**					
18	**female**	35.625	0	no	northeast	2211.13075
21	**female**	33.630	2	no	northwest	3579.82870
48	**male**	28.000	1	yes	southwest	23568.27200
36	**male**	34.430	0	yes	southeast	37742.57570

Membuat kolom 'age' dan 'sex' menjadi *index* :

Syntax :

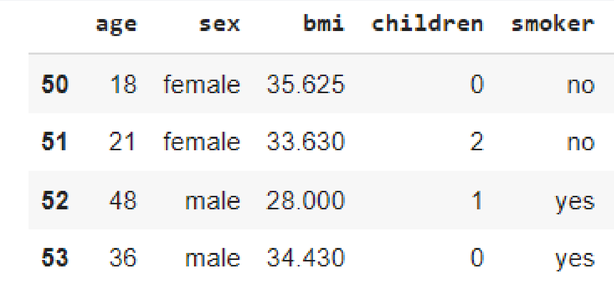
```
nama_dataset.set_index([‘age’, ‘sex’])
```

Menghapus Kolom

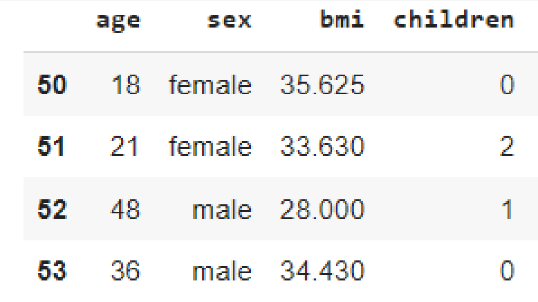
Pandas dapat menghapus kolom – kolom yang tidak diperlukan dengan tujuan :

1. Hanya memilih kolom yang diperlukan untuk dianalisa
2. Memilih kolom yang akan digunakan dalam pembuatan *machine learning model*

Menghapus Kolom



	age	sex	bmi	children	smoker
50	18	female	35.625	0	no
51	21	female	33.630	2	no
52	48	male	28.000	1	yes
53	36	male	34.430	0	yes



	age	sex	bmi	children
50	18	female	35.625	0
51	21	female	33.630	2
52	48	male	28.000	1
53	36	male	34.430	0

Menghapus kolom ‘smoker’ :

Syntax :

```
nama_dataset.drop([‘smoker’], axis = 1)
```

Menggabungkan DataFrame dengan Metode JOIN

Perbedaan JOIN dan MERGE

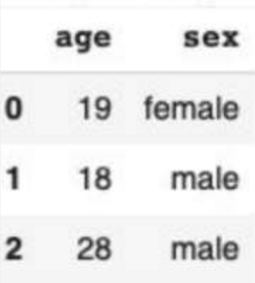
JOIN

Menggabungkan dataset berdasarkan *index*

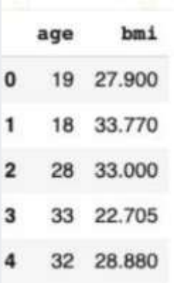
MERGE

Menggabungkan dataset lebih fleksibel dan memungkinkan untuk menentukan kolom selain *index* pada kedua dataframe

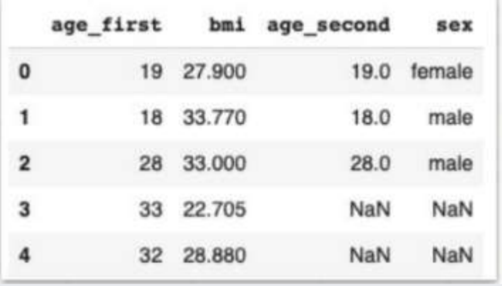
Menggabungkan DataFrame dengan Metode JOIN



	age	sex
0	19	female
1	18	male
2	28	male



	age	bmi
0	19	27.900
1	18	33.770
2	28	33.000
3	33	22.705
4	32	28.880



	age_first	bmi	age_second	sex
0	19	27.900	19.0	female
1	18	33.770	18.0	male
2	28	33.000	28.0	male
3	33	22.705	NaN	NaN
4	32	28.880	NaN	NaN

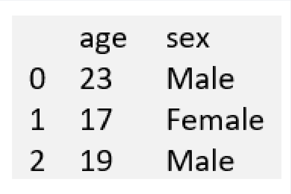
Syntax :

```
data_5.join(data_6, lsuffix = ‘_first’, rsuffix = ‘_second’)
```

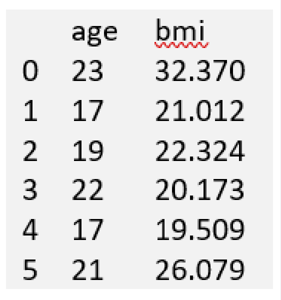
Concatenate & Append DataFrame

Menggabungkan objek dengan Pandas pada *specific axis* baik itu *x-axis* (horizontal) ataupun *y-axis* (vertikal)

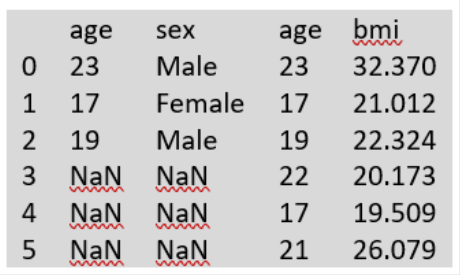
- **Concatenate (Horizontal)**



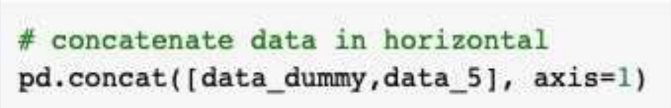
	age	sex
0	23	Male
1	17	Female
2	19	Male



	age	bmi
0	23	32.370
1	17	21.012
2	19	22.324
3	22	20.173
4	17	19.509
5	21	26.079



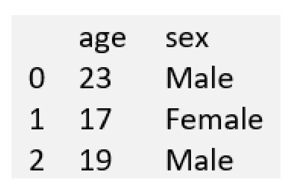
	age	sex	age	bmi
0	23	Male	23	32.370
1	17	Female	17	21.012
2	19	Male	19	22.324
3	NaN	NaN	22	20.173
4	NaN	NaN	17	19.509
5	NaN	NaN	21	26.079



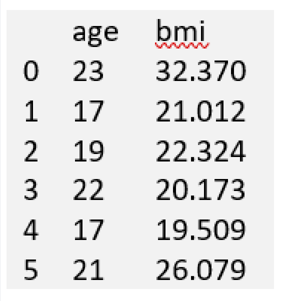
```
# concatenate data in horizontal
pd.concat([data_dummy,data_5], axis=1)
```

Concatenate & Append DataFrame

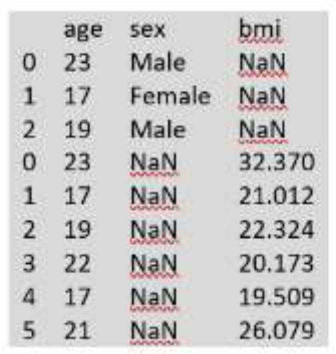
- **Concatenate *(Vertical)***



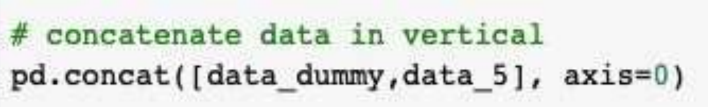
	age	sex
0	23	Male
1	17	Female
2	19	Male



	age	bmi
0	23	32.370
1	17	21.012
2	19	22.324
3	22	20.173
4	17	19.509
5	21	26.079



	age	sex	bmi
0	23	Male	NaN
1	17	Female	NaN
2	19	Male	NaN
0	23	NaN	32.370
1	17	NaN	21.012
2	19	NaN	22.324
3	22	NaN	20.173
4	17	NaN	19.509
5	21	NaN	26.079

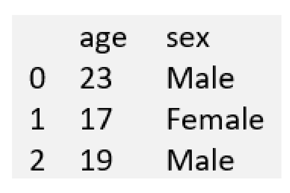


```
# concatenate data in vertical
pd.concat([data_dummy,data_5], axis=0)
```

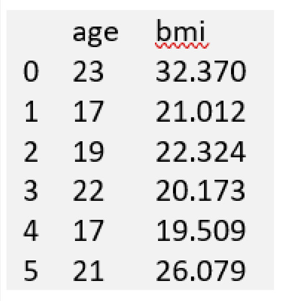
Concatenate & Append DataFrame

- **Append**

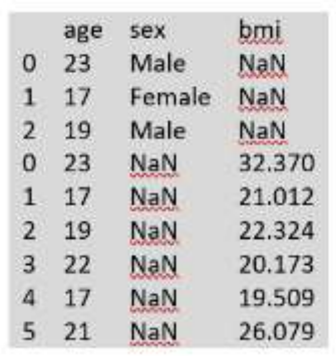
  Dalam dataframe, *append* dapat dilakukan jika terdapat nama kolom pada kedua dataset yang sama.



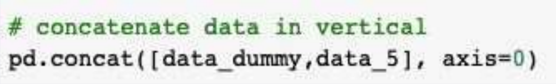
	age	sex
0	23	Male
1	17	Female
2	19	Male



	age	bmi
0	23	32.370
1	17	21.012
2	19	22.324
3	22	20.173
4	17	19.509
5	21	26.079



	age	sex	bmi
0	23	Male	NaN
1	17	Female	NaN
2	19	Male	NaN
0	23	NaN	32.370
1	17	NaN	21.012
2	19	NaN	22.324
3	22	NaN	20.173
4	17	NaN	19.509
5	21	NaN	26.079



```
# concatenate data in vertical
pd.concat([data_dummy,data_5], axis=0)
```

Pivot Table Dataframe

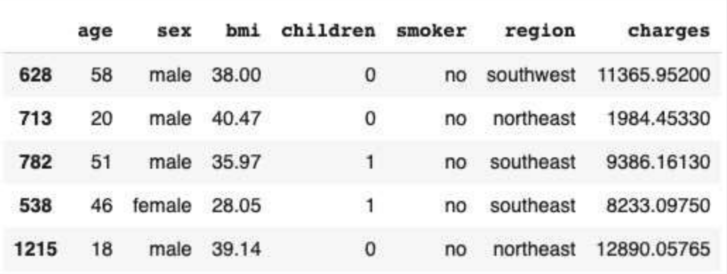
Pivot table memberikan informasi berupa agregasi suatu data dengan melampirkan isi data pada nama kolom tertentu.

Pivot Table

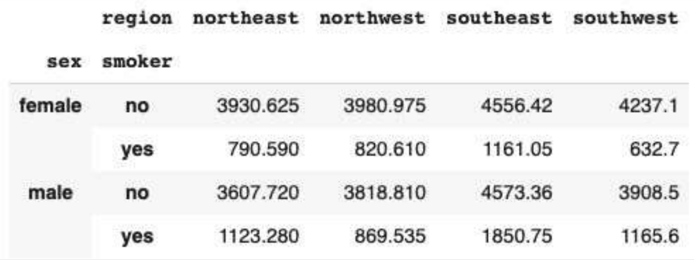
Beberapa karakteristik pivot table menggunakan pandas:

- Tampilan seperti pivot table yang ada di spreadsheet

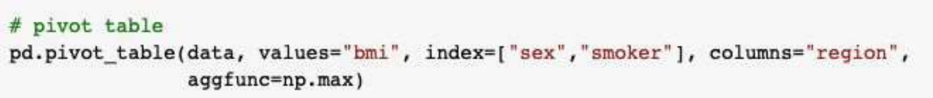
- Nama kolom sebagai level data disimpan dalam bentuk MultiIndex



	age	sex	bmi	children	smoker	region	charges
628	58	male	38.00	0	no	southwest	11365.95200
713	20	male	40.47	0	no	northeast	1984.45330
782	51	male	35.97	1	no	southeast	9386.16130
538	46	female	28.05	1	no	southeast	8233.09750
1215	18	male	39.14	0	no	northeast	12890.05765



	region	**northeast**	**northwest**	**southeast**	**southwest**
sex	**smoker**				
female	**no**	3930.625	3980.975	4556.42	4237.1
	yes	790.590	820.610	1161.05	632.7
male	**no**	3607.720	3818.810	4573.36	3908.5
	yes	1123.280	869.535	1850.75	1165.6

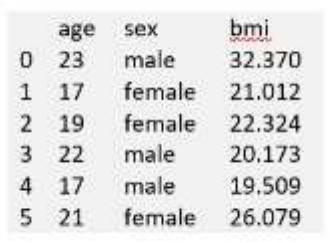


```
# pivot table
pd.pivot_table(data, values="bmi", index=["sex","smoker"], columns="region",
               aggfunc=np.max)
```

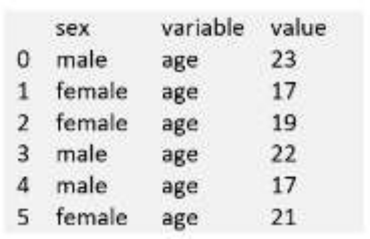
Melting Dataframe

Melting dataframe digunakan untuk memberikan informasi data dimana **nama kolom/variabel akan menjadi *datapoint*** dan tetap memberikan informasi nilai dari kolom/variabel namun di kolom yang berbeda.

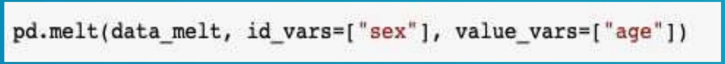
Pivot Table



	age	sex	bmi
0	23	male	32.370
1	17	female	21.012
2	19	female	22.324
3	22	male	20.173
4	17	male	19.509
5	21	female	26.079



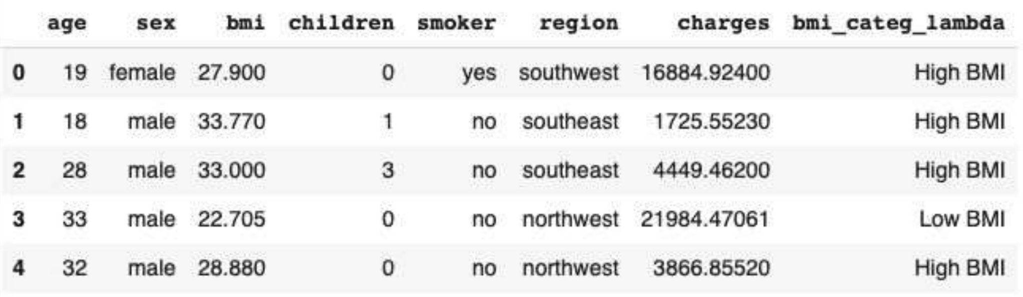
	sex	variable	value
0	male	age	23
1	female	age	17
2	female	age	19
3	male	age	22
4	male	age	17
5	female	age	21



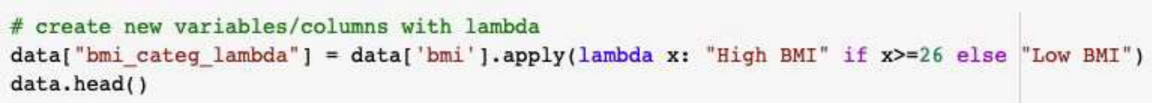
```
pd.melt(data_melt, id_vars=["sex"], value_vars=["age"])
```

Lambda Function

- *Lambda function* mempersingkat *syntax* python



	age	sex	bmi	children	smoker	region	charges	bmi_categ_lambda
0	19	female	27.900	0	yes	southwest	16884.92400	High BMI
1	18	male	33.770	1	no	southeast	1725.55230	High BMI
2	28	male	33.000	3	no	southeast	4449.46200	High BMI
3	33	male	22.705	0	no	northwest	21984.47061	Low BMI
4	32	male	28.880	0	no	northwest	3866.85520	High BMI



```python
# create new variables/columns with lambda
data["bmi_categ_lambda"] = data['bmi'].apply(lambda x: "High BMI" if x>=26 else "Low BMI")
data.head()
```

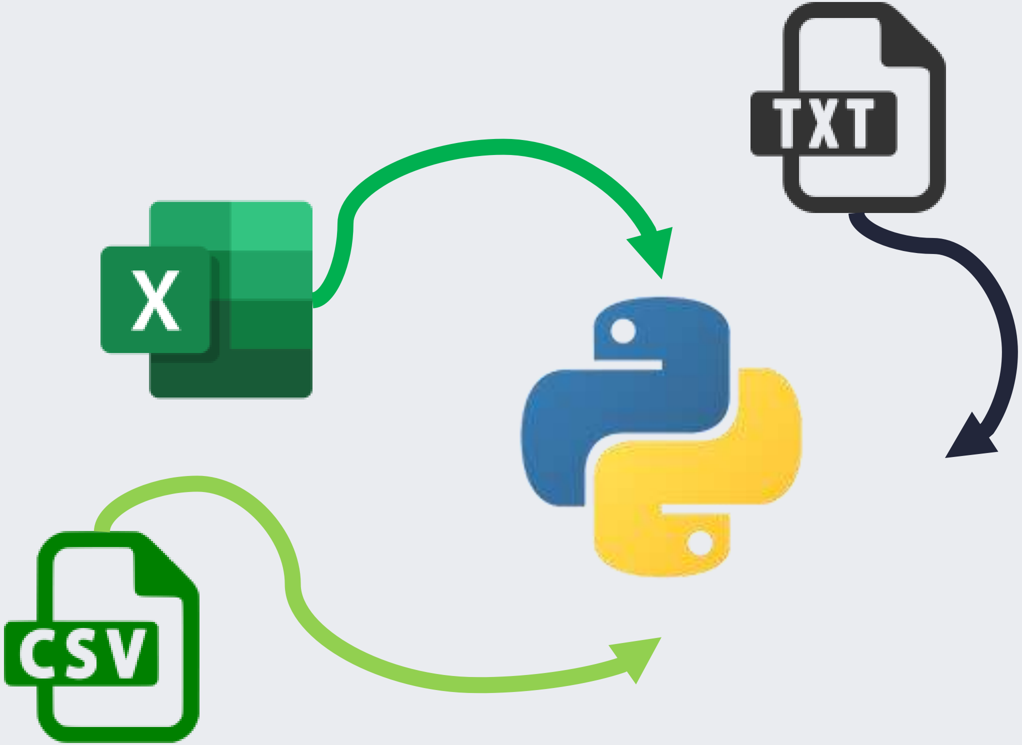
2 Database Programming

READING FILE IN PYTHON

Ada banyak tipe file yang dapat di *read* dalam PYTHON.

Diantara yang sudah dipelajari adalah:

- CSV
- Excel
- Text

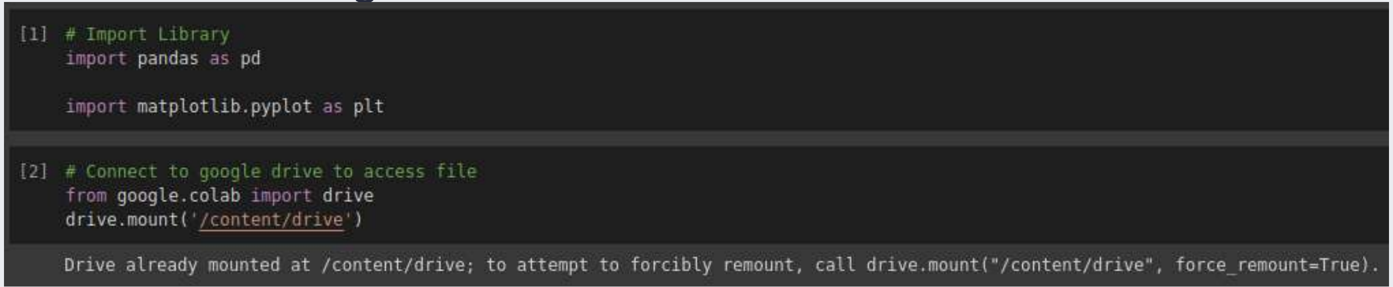


FILE HANDLING IN PYTHON

Character	Meaning
“r”	Read - Default value. Opens a file for reading, error if the file does not exist
“a”	Append - Opens a file for appending, creates the file if it does not exist
“w”	Write - Opens a file for writing, creates the file if it does not exist
“x”	Create - Creates the specified file, returns an error if the file exists
“t”	Text - Default value. Text mode
“b”	Binary - Binary mode (e.g. images)
“+”	Open for updating (reading and writing)

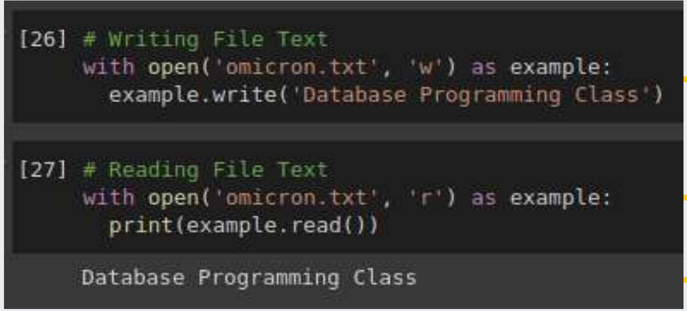
CREATE FILE IN PYTHON

Connect to Google Drive to Access File



```
[1]  # Import Library
     import pandas as pd

     import matplotlib.pyplot as plt
```

```
[2]  # Connect to google drive to access file
     from google.colab import drive
     drive.mount('/content/drive')

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```



```
[26] # Writing File Text
     with open('omicron.txt', 'w') as example:
       example.write('Database Programming Class')
```

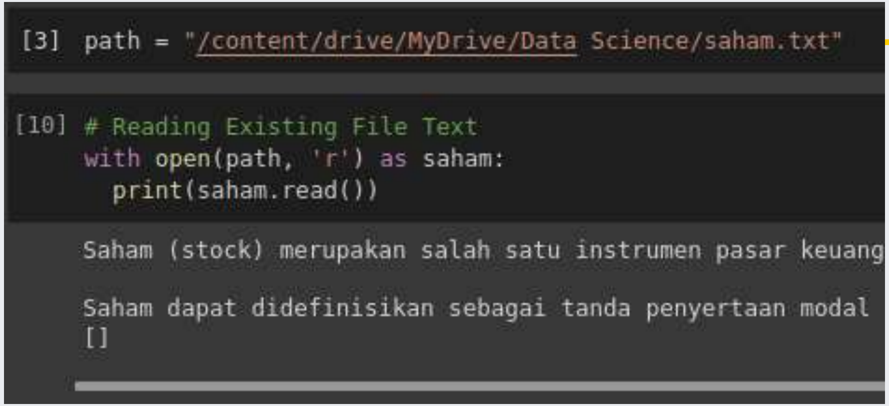
```
[27] # Reading File Text
     with open('omicron.txt', 'r') as example:
       print(example.read())

     Database Programming Class
```

Create new file using write (“w”)

Read new file using read (“r”)

Output

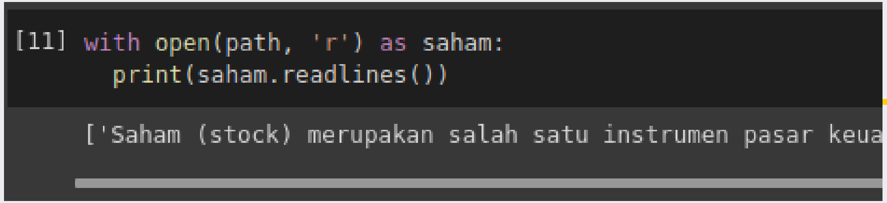
READ EXISTING FILE



```
[3]  path = "/content/drive/MyDrive/Data Science/saham.txt"

[10] # Reading Existing File Text
     with open(path, 'r') as saham:
       print(saham.read())

     Saham (stock) merupakan salah satu instrumen pasar keuang

     Saham dapat didefinisikan sebagai tanda penyertaan modal
     []
```

Create Path File

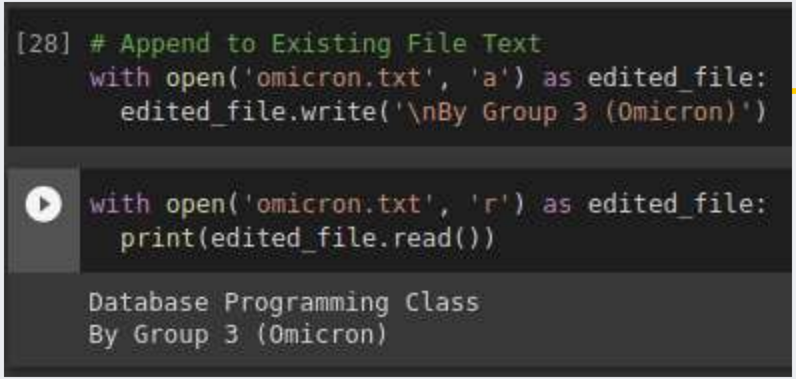
Read existing file using read ("r")



```
[11] with open(path, 'r') as saham:
       print(saham.readlines())

     ['Saham (stock) merupakan salah satu instrumen pasar keua
```

Read existing file using readlines (Make output in One LINE)

APPEND TO EXISTING FILE & DELETE FILE



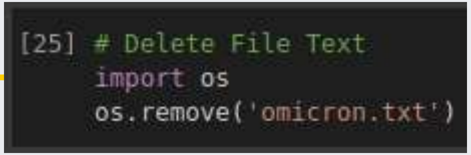
```
[28] # Append to Existing File Text
     with open('omicron.txt', 'a') as edited_file:
       edited_file.write('\nBy Group 3 (Omicron)')
```

```
with open('omicron.txt', 'r') as edited_file:
  print(edited_file.read())
```

```
Database Programming Class
By Group 3 (Omicron)
```

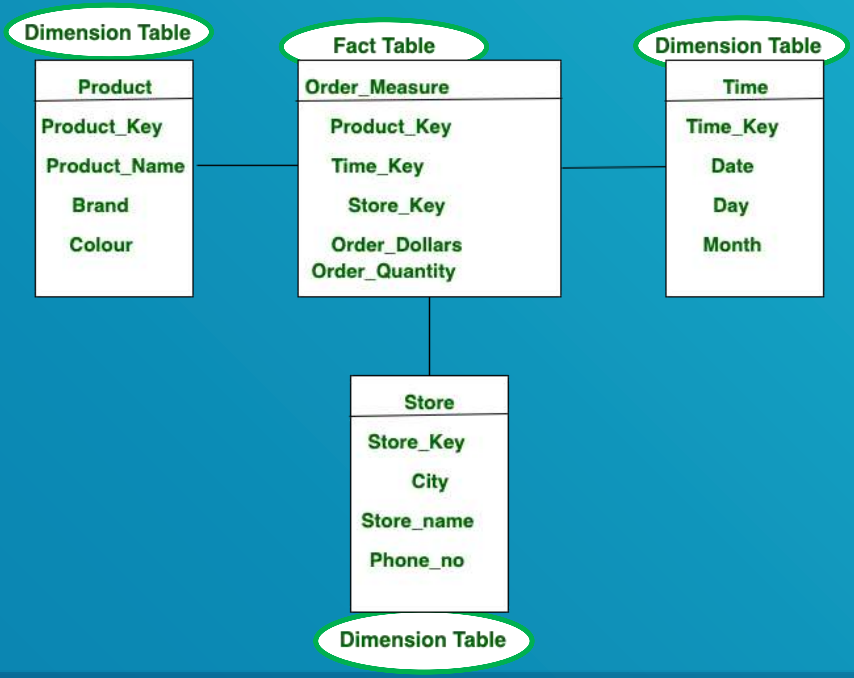
Append new text to existing file

Delete File



```
[25] # Delete File Text
     import os
     os.remove('omicron.txt')
```

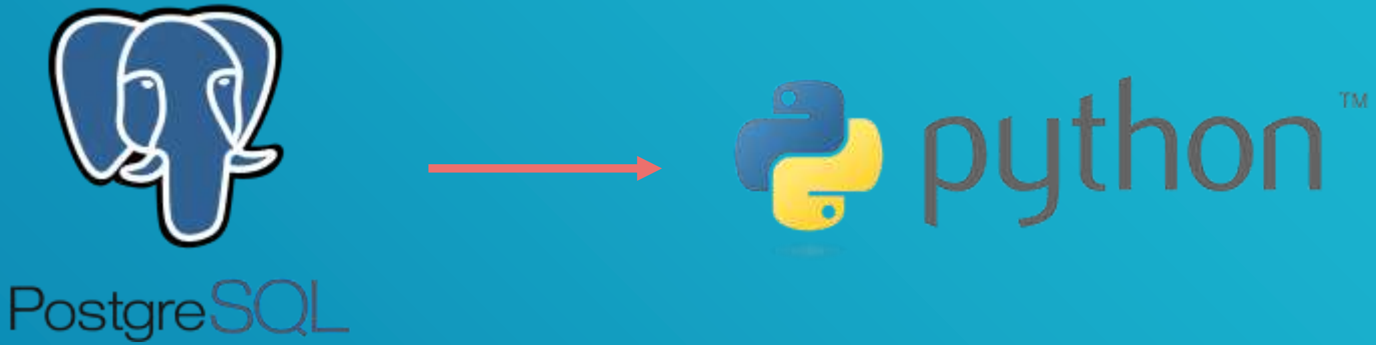
DATA MODELLING



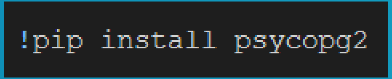
Perbedaan *Fact Table & Dimension Table*

NO.	Fact Table	Dimension Table
1.	Fact table contains the measuring on the attributes of a dimension table.	Dimension table contains the attributes on that truth table calculates the metric.
2.	In fact table, There is less attributes than dimension table.	While in dimension table, There is more attributes than fact table.
3.	In fact table, There is more records than dimension table.	While in dimension table, There is less records than fact table.
4.	Fact table forms a vertical table.	While dimension table forms a horizontal table.
5.	The attribute format of fact table is in numerical format and text format.	While the attribute format of dimension table is in text format.
6.	It comes after dimension table.	While it comes before fact table.
7.	The number of fact table is less than dimension table in a schema.	While the number of dimension is more than fact table in a schema.
8.	It is used for analysis purpose and decision making.	While the main task of dimension table is to store the information about a business and its process.

PUT POSTGRE TO PYTHON

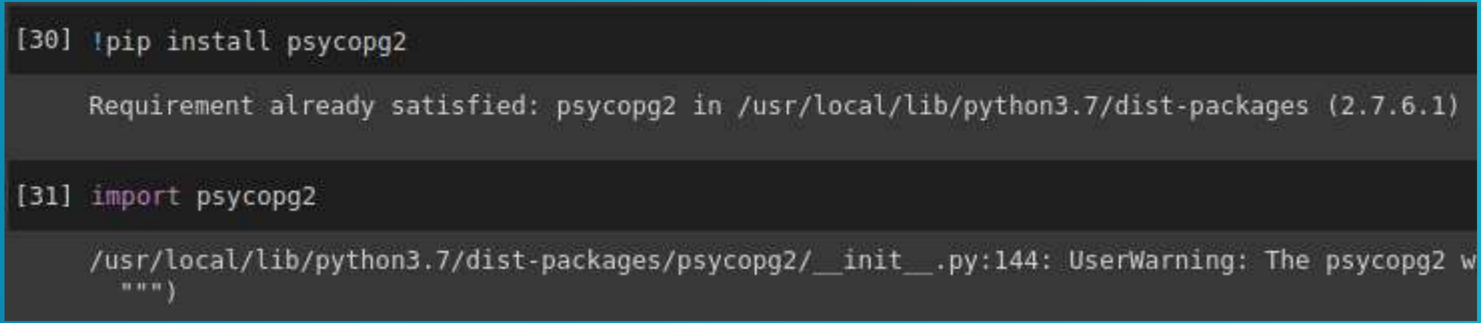


Psycopg library, adaptor database untuk digunakan di dalam pemograman Python

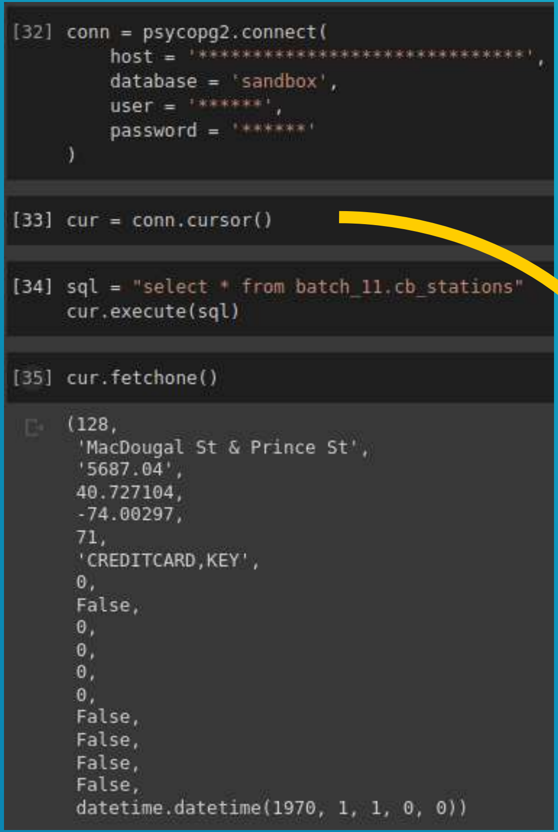


```
!pip install psycopg2
```

PREPARATION TO CONNECT POSTGRE IN PYTHON



```
[30] !pip install psycopg2

     Requirement already satisfied: psycopg2 in /usr/local/lib/python3.7/dist-packages (2.7.6.1)

[31] import psycopg2

     /usr/local/lib/python3.7/dist-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 w
       """)
```



```
[32] conn = psycopg2.connect(
         host = '******************************',
         database = 'sandbox',
         user = '******',
         password = '******'
     )

[33] cur = conn.cursor()

[34] sql = "select * from batch_11.cb_stations"
     cur.execute(sql)

[35] cur.fetchone()

     (128,
      'MacDougal St & Prince St',
      '5687.04',
      40.727104,
      -74.00297,
      71,
      'CREDITCARD,KEY',
      0,
      False,
      0,
      0,
      0,
      0,
      False,
      False,
      False,
      False,
      datetime.datetime(1970, 1, 1, 0, 0))
```

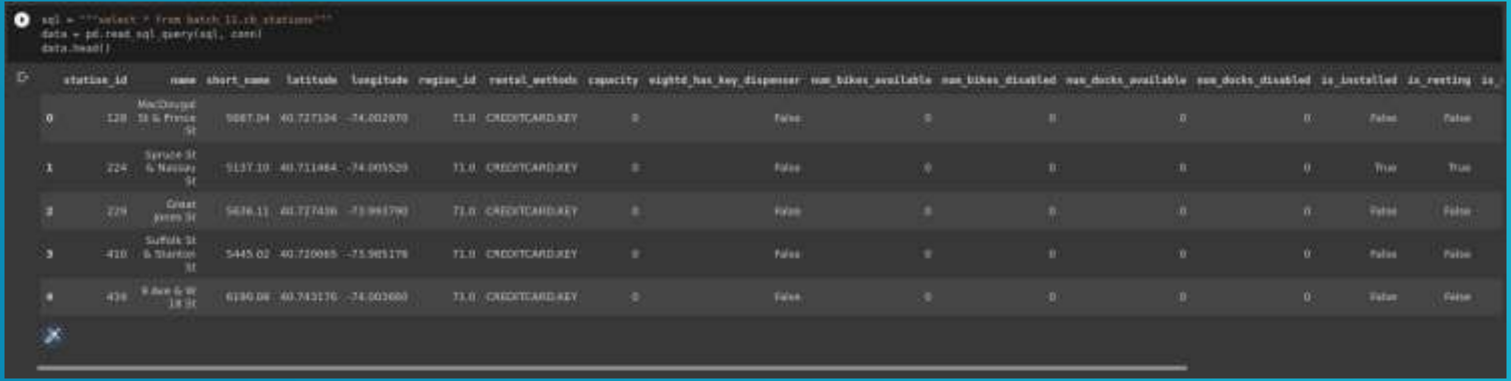
CONNECT THE DATABASE

To Connect, we need:

- Host
- User
- Database
- Password

Store the database table using cursor (Temporary Memory)

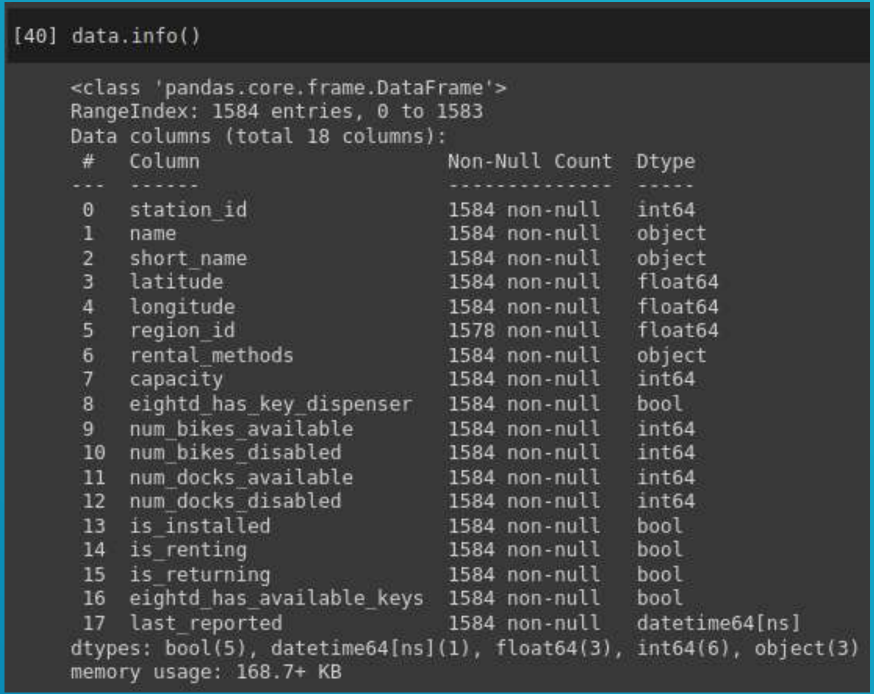
EXECUTE QUERY



```
sql = """select * from batch_11.cb_stations"""
data = pd.read_sql_query(sql, conn)
data.head()
```

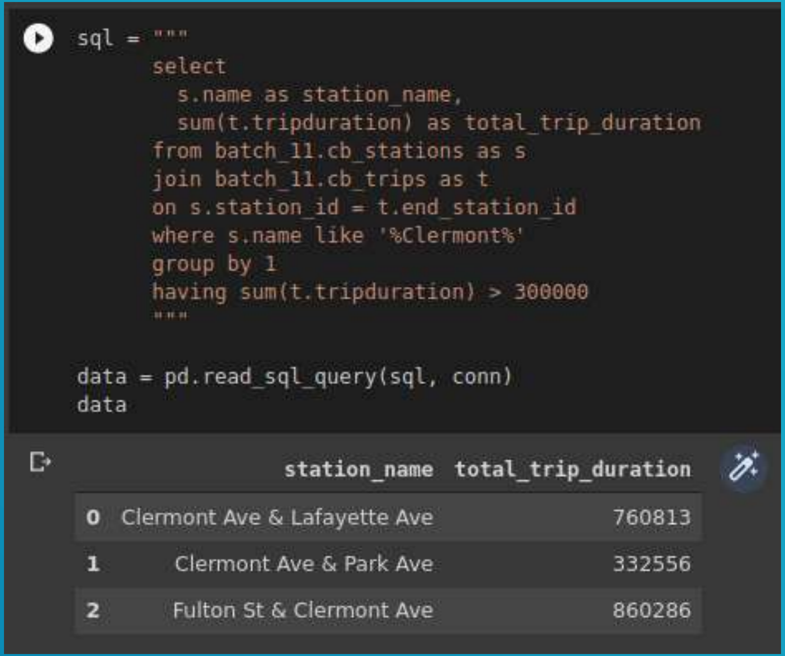
	station_id	name	short_name	latitude	longitude	region_id	rental_methods	capacity	eightd_has_key_dispenser	num_bikes_available	num_bikes_disabled	num_docks_available	num_docks_disabled	is_installed	is_renting	is_
0	128	MacDougal St & Prince St	5687.04	40.727104	-74.002970	71.0	CREDITCARD,KEY	0	False	0	0	0	0	False	False	
1	224	Spruce St & Nassau St	5137.10	40.711464	-74.005529	71.0	CREDITCARD,KEY	0	False	0	0	0	0	True	True	
2	229	Great Jones St	5636.11	40.727434	-73.993790	71.0	CREDITCARD,KEY	0	False	0	0	0	0	False	False	
3	410	Suffolk St & Stanton St	5445.02	40.720665	-73.985179	71.0	CREDITCARD,KEY	0	False	0	0	0	0	False	False	
4	436	9 Ave & W 18 St	6190.08	40.743176	-74.003660	71.0	CREDITCARD,KEY	0	False	0	0	0	0	False	False	

SEE OUTPUT INFO



```
[40] data.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 1584 entries, 0 to 1583
     Data columns (total 18 columns):
      #   Column                      Non-Null Count  Dtype
     ---  ------                      --------------  -----
      0   station_id                  1584 non-null   int64
      1   name                        1584 non-null   object
      2   short_name                  1584 non-null   object
      3   latitude                    1584 non-null   float64
      4   longitude                   1584 non-null   float64
      5   region_id                   1578 non-null   float64
      6   rental_methods              1584 non-null   object
      7   capacity                    1584 non-null   int64
      8   eightd_has_key_dispenser    1584 non-null   bool
      9   num_bikes_available         1584 non-null   int64
      10  num_bikes_disabled          1584 non-null   int64
      11  num_docks_available         1584 non-null   int64
      12  num_docks_disabled          1584 non-null   int64
      13  is_installed                1584 non-null   bool
      14  is_renting                  1584 non-null   bool
      15  is_returning                1584 non-null   bool
      16  eightd_has_available_keys   1584 non-null   bool
      17  last_reported               1584 non-null   datetime64[ns]
     dtypes: bool(5), datetime64[ns](1), float64(3), int64(6), object(3)
     memory usage: 168.7+ KB
```

MORE CHALLENGING QUERY



```
sql = """
        select
          s.name as station_name,
          sum(t.tripduration) as total_trip_duration
        from batch_11.cb_stations as s
        join batch_11.cb_trips as t
        on s.station_id = t.end_station_id
        where s.name like '%Clermont%'
        group by 1
        having sum(t.tripduration) > 300000
        """

data = pd.read_sql_query(sql, conn)
data
```

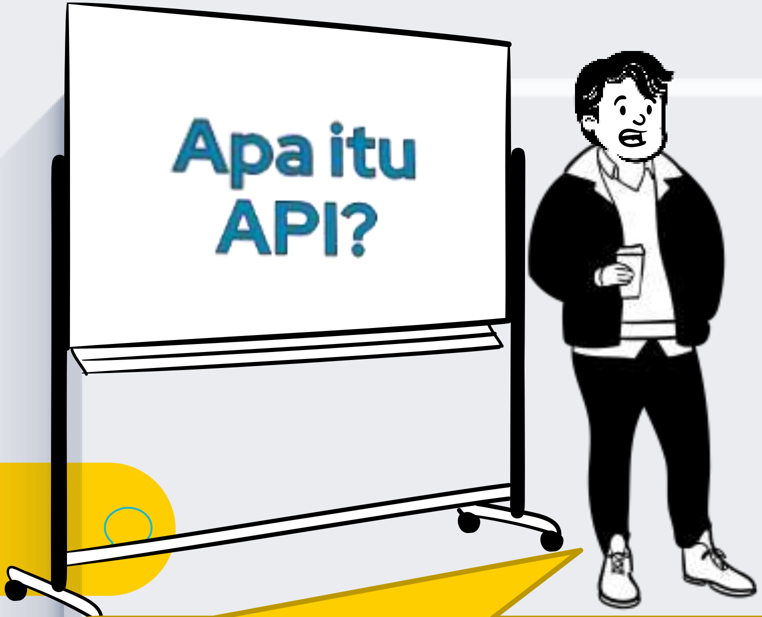
```
                  station_name  total_trip_duration
0  Clermont Ave & Lafayette Ave               760813
1       Clermont Ave & Park Ave               332556
2       Fulton St & Clermont Ave               860286
```

DigitalSkola

3

Application Programming Interface

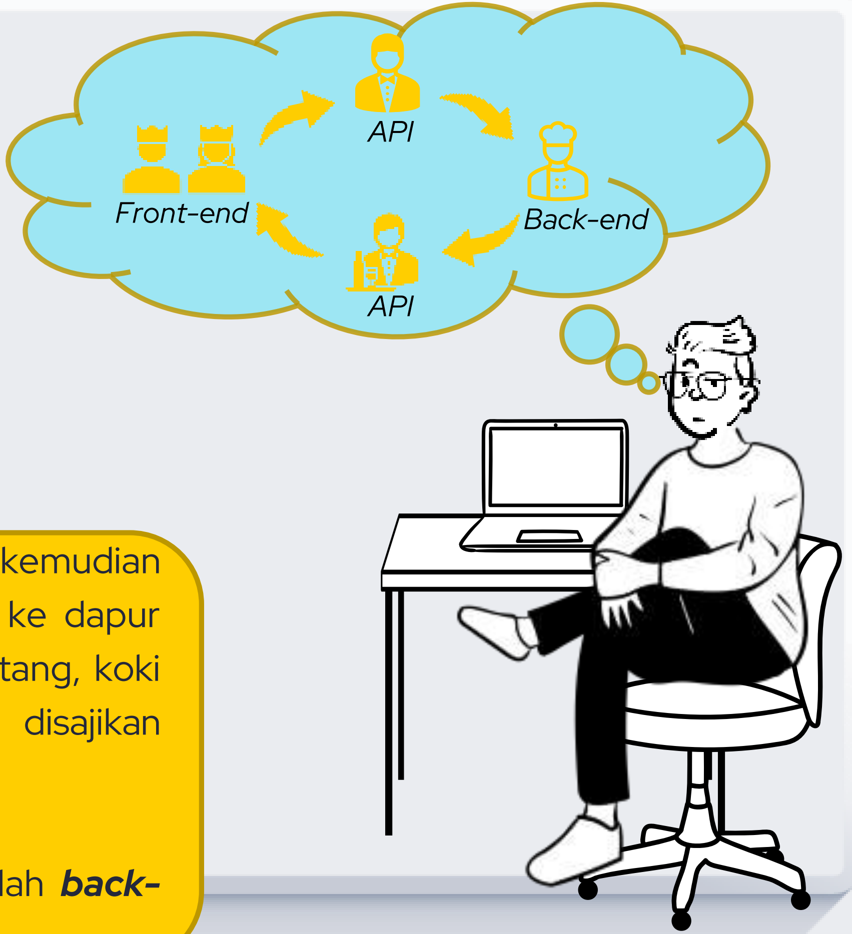
{ A P I }



Apa itu API?

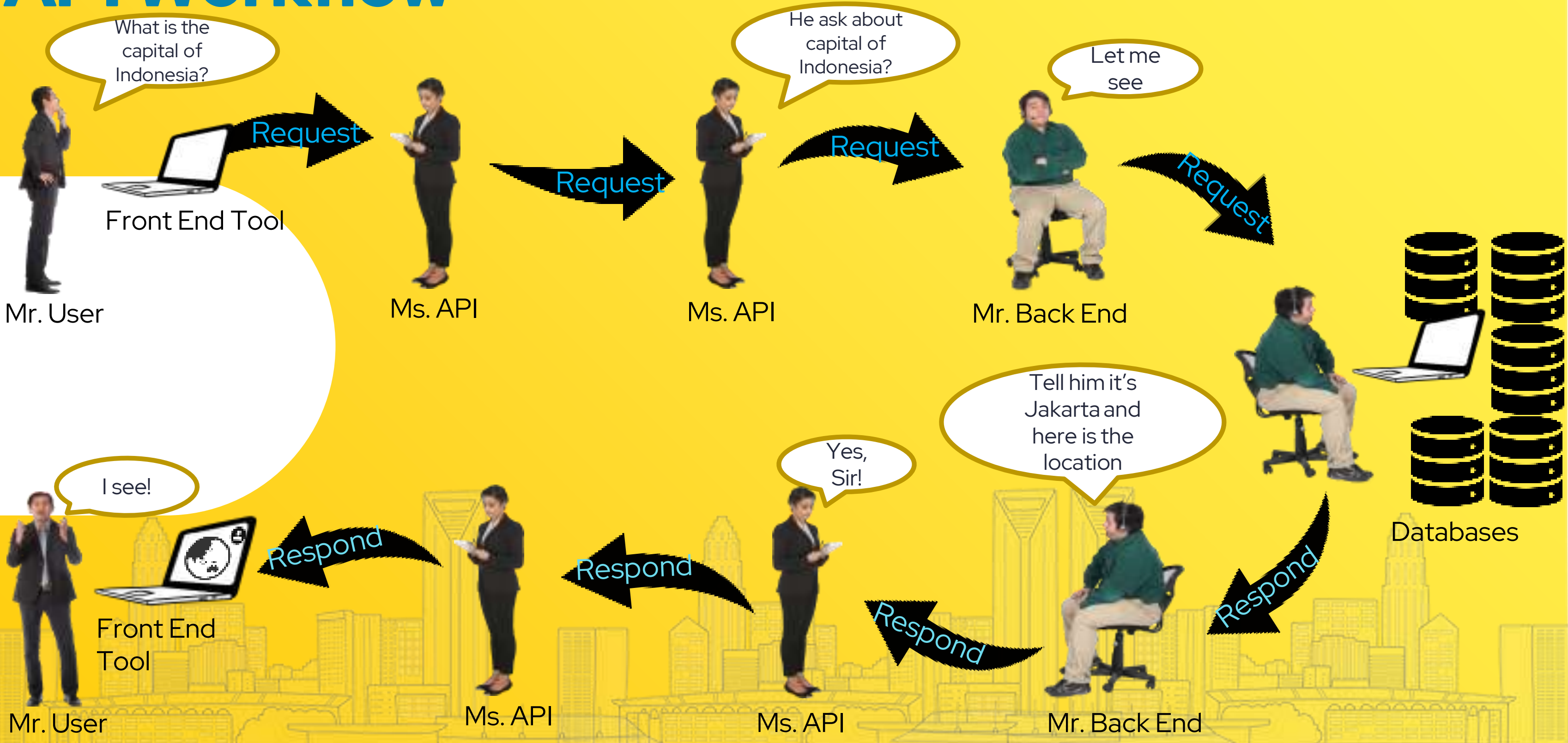
Bayangkan kamu pergi ke sebuah restoran, kemudian pramusaji mencatat pesananmu dan membawanya ke dapur untuk dimasak oleh koki. Ketika makanan sudah matang, koki memberikannya ke pramusaji untuk kemudian disajikan kepadamu.

Dalam kasus ini, kamu adalah ***front-end***, koki adalah ***back-end***, dan pramusaji adalah ***API*** .

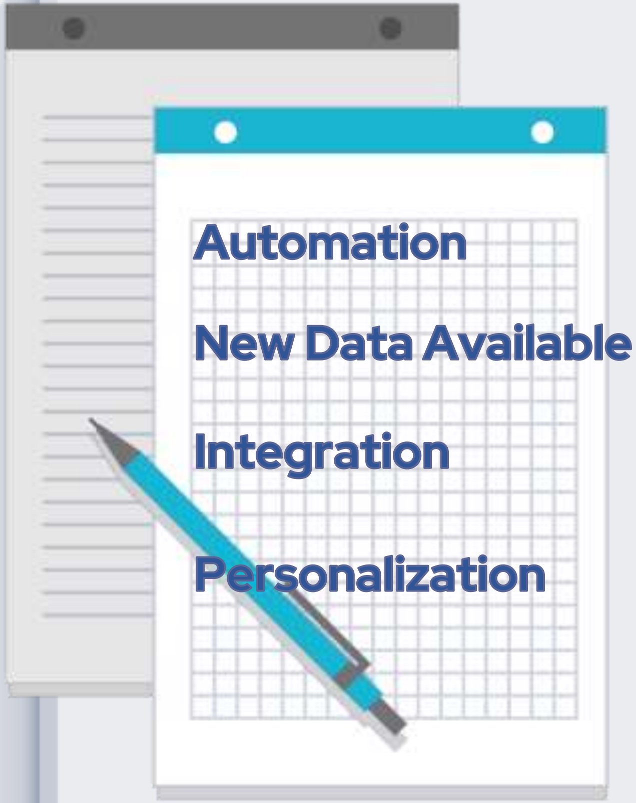


DigitalSkola

API Workflow



Keuntungan API



Automation	Agensi dapat memperbaharui alur kerja agar lebih produktif
New Data Available	Semua informasi yang dihasilkan terbuka untuk semua orang
Integration	Konten dari laman apapun atau aplikasi yang dapat dengan mudah tertanam
Personalization	*User* dapat memodifikasi konten dan pelayanan yang sering mereka gunakan

Implementasi API

Format Data API :

{;} JSON

- ✓ Bahasa pemrograman *Built on javascript*
- ✓ Sangat berguna baik di *front-end* dan *back-end*
- ✓ Format sederhana, yaitu : { *"key"* : *"values"* }

</> XML

- ✓ Format data *mature* dan *powerful*
- ✓ Blok Utama disebut ***node***
- ✓ Format : < *opening node* >*value* </ *closing node* >

API di =

django

Flask
web development,
one drop at a time

Fast API

Komponen API

01 ENDPOINT

URL yang menggambarkan data yang sedang kita gunakan. URL *endpoint* terikat dengan *resource* tertentu di dalam API.

02 DATA

Untuk menggunakan *method* yang meliputi perubahan data di dalam REST API. Kita membutuhkan *data payload* dengan *request create atau* modify data.

03 HEADERS

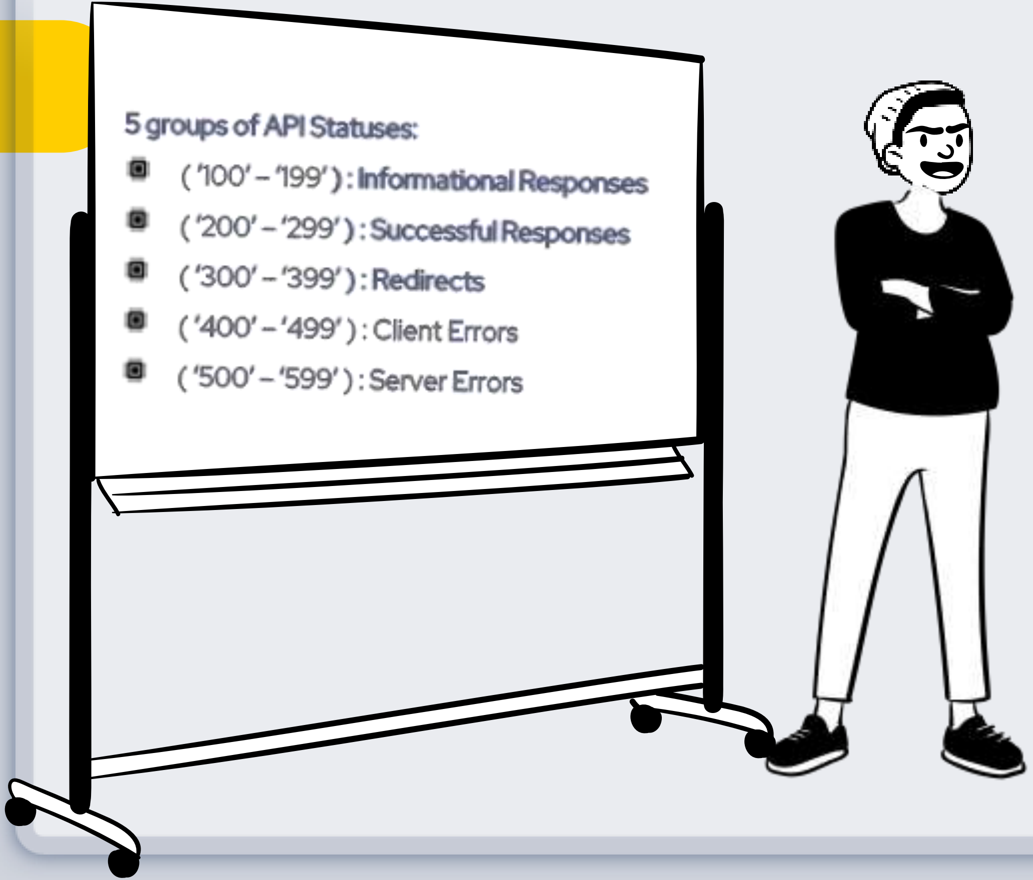
Berisi metadata yang dibutuhkan untuk memasukkan *request, seperti authentication tokens, content type returned*, dan *caching policy*.

04 METHOD

Menunjukkan bagaimana cara untuk berinteraksi dengan *resources* yang berada di *endpoint*. REST API *method meliputi :*

HTTP Verb	CRUD
POST	Create
GET	Read
PUT	Update / Replace
PATCH	Update / Modify
DELETE	Delete

Status API



5 groups of API Statuses:

- ('100' – '199') : Informational Responses
- ('200' – '299') : Successful Responses
- ('300' – '399') : Redirects
- ('400' – '499') : Client Errors
- ('500' – '599') : Server Errors

Kode status respon HTTP mengindikasikan apakah *request* HTTP tertentu telah berhasil.

Terima Kasih !

👍