HW Basic and Intermediate API

- Mohon download data dari

a. https://www.kaggle.com/olistbr/brazilian-ecommerce

- olist_order_items_dataset_csv (10 ribu pertama)

b. Buatlah sebuah script yang bertujuan untuk memasukkan data csv ke dalam database
c. Buatlah sebuah script yang bertujuan untuk membaca data dari database
d. Buatlah sebuah python script yang digunakan untuk menjalankan aplikasi berbasis flask berikut jg dengan responnya

Capture script from jupyter notebook

a. Read raw data csv 10 ribu pertama

```
#cek raw data csv
df = pd.read_csv('olist_order_items_dataset.csv')
df[:10000]
```

	order_id	order_item_id	product_id	seller_id	sh
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	20
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	20
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	20
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	20
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	20
...	...	...	...	...	...
9995	16f391bc8fc37407de41720cd92f6266	1	78b7b1ff2d3f06a589354ddf2f4f9db3	620c87c171fb2a6dd6e8bb4dec959fc6	20
9996	16f4331c56f6c5c76a5ea85c7919f087	1	19c91ef95d509ea33eda93495c4d3481	06a2c3af7b3aee5d69171b0e14f0ee87	20
9997	16f4a05a36f470dbed1f1c2ca6e2a616	1	0aabfb375647d9738ad0f7b4ea3653b1	37515688008a7a40ac93e3b2e4ab203f	20
9998	16f4c47c722704ef26d0f086cb75d213	1	aca2eb7d00ea1a7b8ebd4e68314663af	955fee9216a65b617aa5c0531780ce60	20
9999	16f4c47c722704ef26d0f086cb75d213	2	aca2eb7d00ea1a7b8ebd4e68314663af	955fee9216a65b617aa5c0531780ce60	20

10000 rows × 7 columns

b. script yang bertujuan untuk memasukkan data csv ke dalam database

```
# set up configuration to local database
host = "localhost"
port = "5432"
database = "homework"
user = "postgres"
password = "Jakarta12"
setting = "dbname=" + database + " user=" + user + " host=" + host + " port=" + port + " password=" + p
engine = pg.connect(setting)
```

```
#function insert raw data csv to table local db
def insert_into_table(conn, df, table):
    """
    Using cursor.mogrify() to build the bulk insert query
    then cursor.execute() to execute the query
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]

#     print("TUPLE" , tuples)
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))

#     print("COLUMNS", cols)

    # SQL quert to execute
    cursor = conn.cursor()
    values = [cursor.mogrify("(%s,%s,%s,%s,%s,%s,%s)", tup).decode('utf8') for tup in tuples]
    query = "INSERT INTO %s(%s) VALUES " % (table, cols) + ",".join(values)

    print("QUERY", query)
    try:
        cursor.execute(query, tuples)
        conn.commit()
    except (Exception, pg.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("INSERT DONE")
    cursor.close()
```

```
# call function to import raw data from csv file to local db


insert_into_table(engine, df[:10000], 'olist_order_items_dataset')
```

```
6','8b321bb669392f5163d04c59e235e066','2018-04-23 22:51:53',15.0,19.04),('16ec508e18e1d4949fb54daf2539f7bd',1,'e611a297d7a9df83f906
aab5c746fd06','81f89e42267213cb94da7ddc301651da','2018-08-28 22:04:30',96.0,13.33),('16ec5f5b0b5245798a6334e4c239d13a',1,'a62e25e09
e05e6faf31d90c6ec1aa3d1','634964b17796e64304cadf1ad3050fb7','2018-01-26 14:16:44',108.0,15.52),('16ed456866d66d9108a13a17374364c4',
1,'9d6580f17a0cba74324ce82874fbbe13','4e922959ae960d389249c378d1c939f5','2018-02-15 20:08:12',145.0,21.5),('16ee199eae699a717d29b71
35597f94f',1,'a7b9753ca5040193a1505f15c723c0e0','09f952a5f58d2285b0372551ae8f9b01','2018-04-17 01:10:16',330.0,19.89),('16ee19fe4df
0a090015e2fa16f3bf3e0',1,'3dd2a17168ec895c781a9191c1e95ad7','de722cd6dad950a92b7d4f82673f8833','2018-04-30 04:30:41',149.9,21.1),
('16ef87ed2e298fd5de508be5d5d29833',1,'75d6b6963340c6063f7f4cfcccfe6a30','cc419e0650a3c5ba77189a1882b7556a','2017-11-20 08:56:08',5
6.99,15.15),('16efc11ccab2e8a17656020f80cd39e3',1,'75c859dbba6358947a27d204801125a6','8759e7aedd644f487315e5860962f162','2017-12-11
04:11:45',209.0,46.93),('16f0fa8aecf907e4fe3768af883f0def',1,'99a4ae1a9ff32f696460a9295533d2e2','750303a20e9c56b2a6bc45cdce0b897
d','2018-06-20 12:19:12',59.0,71.13),('16f1d9edf9452dd60d94733527624571',1,'b73a2c0ec9eb8d2badfbc9d8d27f3ad6','dd7ddc04e1b6c2c61435
2b383efe2d36','2018-07-31 18:31:18',119.6,39.44),('16f2b04292cfdf925c7d231f98812249',1,'d1c427060a0f73f6b889a5c7c61f2ac4','a1043baf
d471dff536d0c462352beb48','2017-10-05 12:04:21',149.99,40.38),('16f3197b63d493ba1c9fa0fd6c75f985',1,'164520f853fb53ecd0b55302c564f7
ad','aa1eb17b9fdf6dfd546e25c45c7b9235','2018-04-02 03:55:27',30.0,7.39),('16f3197b63d493ba1c9fa0fd6c75f985',2,'164520f853fb53ecd0b5
5302c564f7ad','aa1eb17b9fdf6dfd546e25c45c7b9235','2018-04-02 03:55:27',30.0,7.39),('16f391bc8fc37407de41720cd92f6266',1,'78b7b1ff2d
3f06a589354ddf2f4f9db3','620c87c171fb2a6dd6e8bb4dec959fc6','2017-04-13 12:32:07',292.9,16.22),('16f4331c56f6c5c76a5ea85c7919f087',
1,'19c91ef95d509ea33eda93495c4d3481','06a2c3af7b3aee5d69171b0e14f0ee87','2018-07-02 21:27:05',122.99,36.63),('16f4a05a36f470dbed1f1
c2ca6e2a616',1,'0aabfb375647d9738ad0f7b4ea3653b1','37515688008a7a40ac93e3b2e4ab203f','2017-10-06 11:49:24',19.9,15.1),('16f4c47c722
704ef26d0f086cb75d213',1,'aca2eb7d00ea1a7b8ebd4e68314663af','955fee9216a65b617aa5c0531780ce60','2018-01-11 22:48:11',69.9,24.94),
('16f4c47c722704ef26d0f086cb75d213',2,'aca2eb7d00ea1a7b8ebd4e68314663af','955fee9216a65b617aa5c0531780ce60','2018-01-11 22:48:11',6
9.9,24.94)


INSERT DONE
```

c. script yang bertujuan untuk membaca data dari database

```
#Script to read data from local db

query = f"""
    select
        *
    from
        olist_order_items_dataset
    where
        1=1

    """
df_read = pd.read_sql(query, con=engine)
df_read
```

	order_id	order_item_id	product_id	seller_id	sh
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	20
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	20
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	20
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	20
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	20
...	...	...	...	...	...
9995	16f391bc8fc37407de41720cd92f6266	1	78b7b1ff2d3f06a589354ddf2f4f9db3	620c87c171fb2a6dd6e8bb4dec959fc6	20
9996	16f4331c56f6c5c76a5ea85c7919f087	1	19c91ef95d509ea33eda93495c4d3481	06a2c3af7b3aee5d69171b0e14f0ee87	20
9997	16f4a05a36f470dbed1f1c2ca6e2a616	1	0aabfb375647d9738ad0f7b4ea3653b1	37515688008a7a40ac93e3b2e4ab203f	20
9998	16f4c47c722704ef26d0f086cb75d213	1	aca2eb7d00ea1a7b8ebd4e68314663af	955fee9216a65b617aa5c0531780ce60	20
9999	16f4c47c722704ef26d0f086cb75d213	2	aca2eb7d00ea1a7b8ebd4e68314663af	955fee9216a65b617aa5c0531780ce60	20

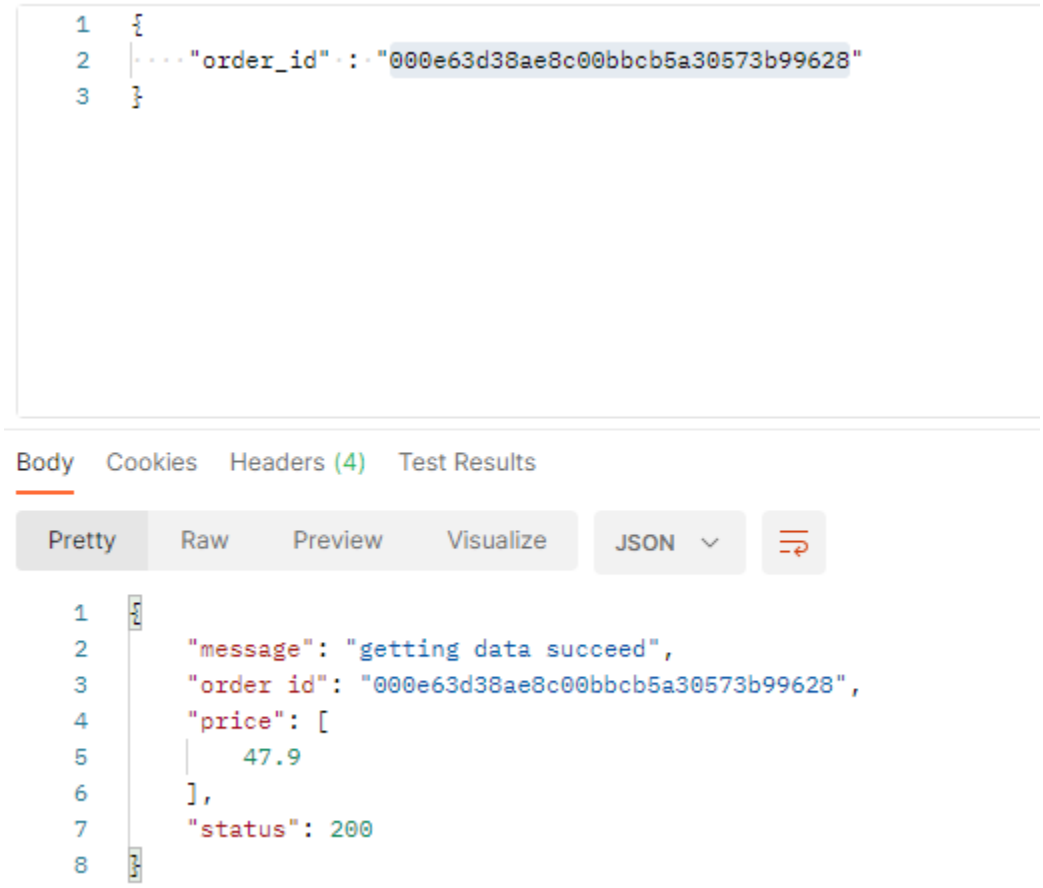
10000 rows × 7 columns

d. python script yang digunakan untuk menjalankan aplikasi berbasis flask berikut jg dengan responnya

- Flask read data

```
application = Flask(__name__)
@application.route('/digital-skola/read', methods=['GET'])
def read():
    content = request.get_json(force=True)
    order_id = content['order_id']
    query = f"""
            SELECT *
            FROM homework.public.olist_order_items_dataset a

            where
                order_id = '{order_id}'

    """
    #--and product_category_name like '%{product_name}%'
    df = pd.read_sql(query, con=engine)

    #arr = [df['product_category_name'].iloc[0]]
    #print("DF ", df)


    try:
        arr = [df['price'].iloc[0]]
        result = {
            "status" : 200,
            "message" : "getting data succeed",
            "order id" : order_id,
            "price": arr
        }
    except IndexError:
        result = {
            "status" : 204,
            "message" : "Nodata for order id = "+ order_id,

        }



    return result
```

- Respone with get available data



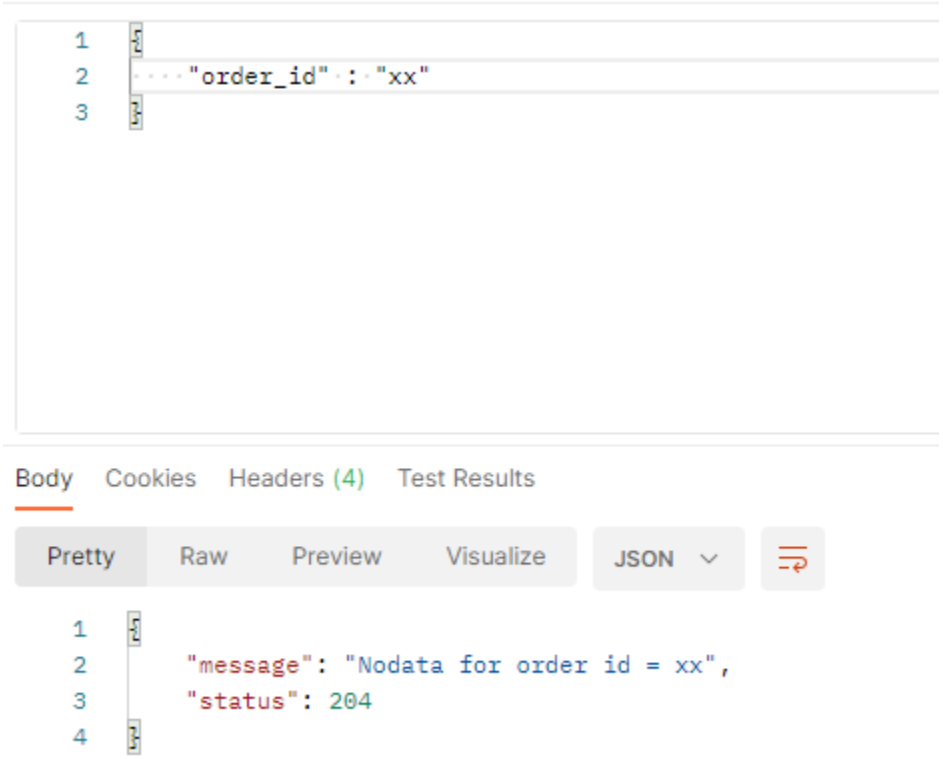
```
{
    "order_id" : "000e63d38ae8c00bbcb5a30573b99628"
}
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "message": "getting data succeed",
    "order id": "000e63d38ae8c00bbcb5a30573b99628",
    "price": [
        47.9
    ],
    "status": 200
}
```

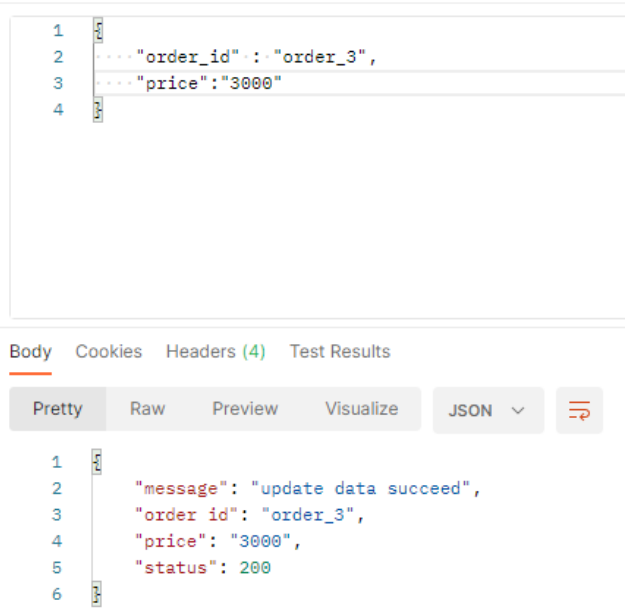
- Respone with get unavailable data



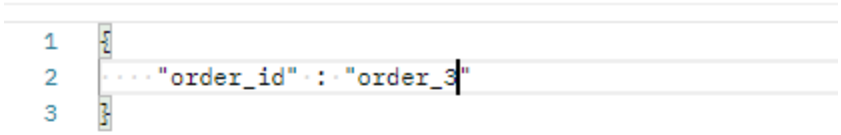
- Flask update data script

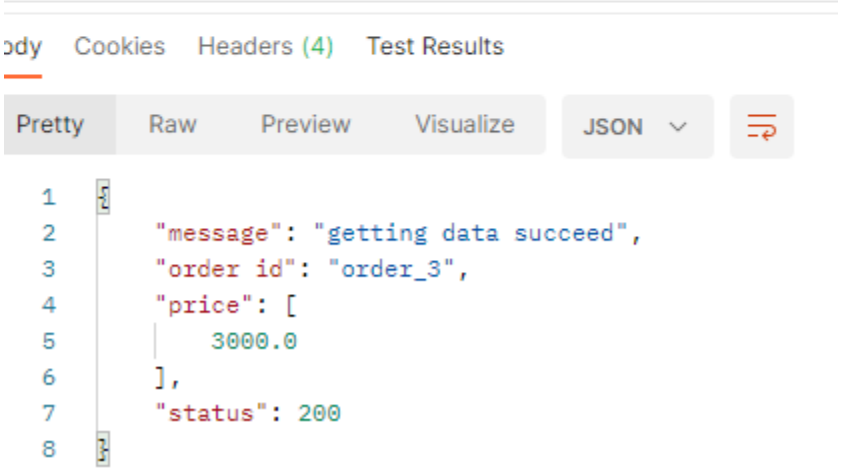
```python
@application.route('/digital-skola/edit', methods=['PUT'])
def edit():
    content = request.get_json(force=True)
    order_id = content['order_id']
    price = content['price']
    query = f"""
            update olist_order_items_dataset
            set price = '{price}'
            where
                order_id = '{order_id}'

    """
    conn=engine
    cursor = conn.cursor()
    cursor.execute(query)
    conn.commit()


    result = {
            "status" : 200,
            "message" : "update data succeed",
            "order id" : order_id,
            "price": price
        }




    return result
```

- Result update data



- After update data check on local db using get data



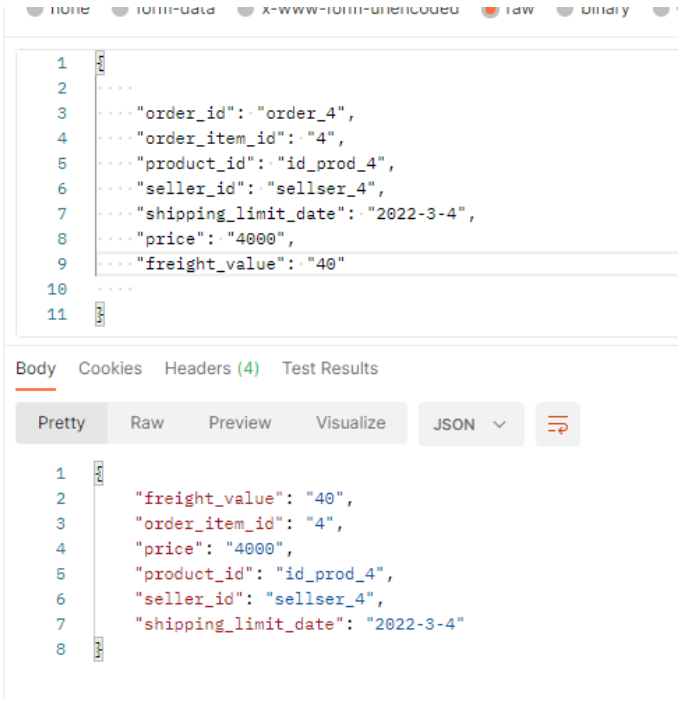


- Flask insert data script

```python
def insert_into_table(conn, df, table):
    """
    Using cursor.mogrify() to build the bulk insert query
    then cursor.execute() to execute the query
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]

    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))

    # SQL query to execute
    cursor = conn.cursor()
    values = [cursor.mogrify("(%s,%s,%s,%s,%s,%s,%s,%s)", tup).decode('utf8') for tup in tuples]
    print("VALUES", values)
    query = "INSERT INTO %s(%s) VALUES " % (table, cols) + ",".join(values)

    # print("QUERY", query)
    try:
        cursor.execute(query, tuples)
        conn.commit()
    except (Exception, pg.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("INSERT DONE")
    cursor.close()
```

```python
@application.route('/digital-skola/insert', methods=['POST'])
def insert():

    content = request.get_json(force=True)
    order_id = content['order_id']
    order_item_id= content['order_item_id']
    product_id = content['product_id']
    seller_id = content['seller_id']
    shipping_limit_date = content['shipping_limit_date']
    price = content['price']
    freight_value = content['freight_value']


    df = pd.DataFrame()

    df.at[(0,'order_id')] = order_id
    df.at[(0,'order_item_id')] = order_item_id
    df.at[(0,'product_id')] = product_id
    df.at[(0,'seller_id')] = seller_id
    df.at[(0,'shipping_limit_date')] = shipping_limit_date
    df.at[(0,'price')] = price
    df.at[(0,'freight_value')] = freight_value

    insert_into_table(engine, df, 'olist_order_items_dataset')

    print(freight_value)
    result = {

            "order_item_id": str(df['order_item_id'].iloc[0]),
            "product_id": str(df['product_id'].iloc[0]),
            "seller_id": str(df['seller_id'].iloc[0]),
            "shipping_limit_date": str(df['shipping_limit_date'].iloc[0]),
            "price": str(df['price'].iloc[0]),
            "freight_value": str(df['freight_value'].iloc[0]),

        }



    return result
```

- Result after inser data



- After inser data then check using get