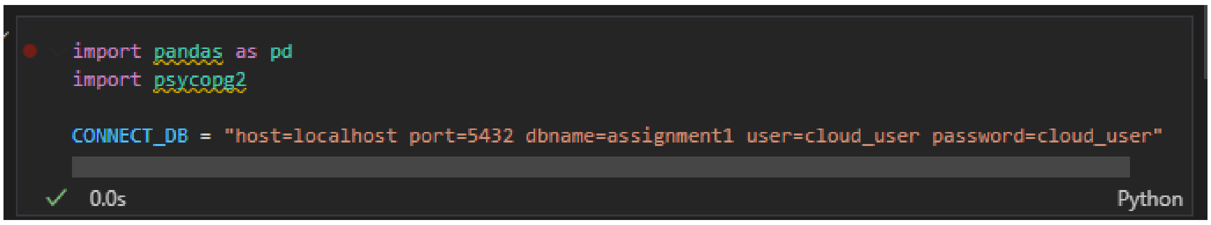
Assignment 1

Connect to postgre local db

Melakukan import pandas dan psycopg2 library, dan membuat koneksi db ke posgresql pada ubuntu local.

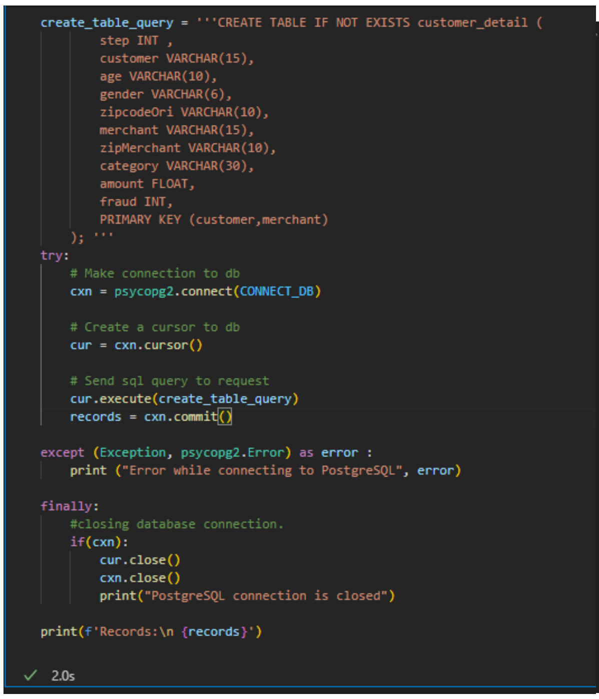


```python
import pandas as pd
import psycopg2

CONNECT_DB = "host=localhost port=5432 dbname=assignment1 user=cloud_user password=cloud_user"
```

✓ 0.0s Python

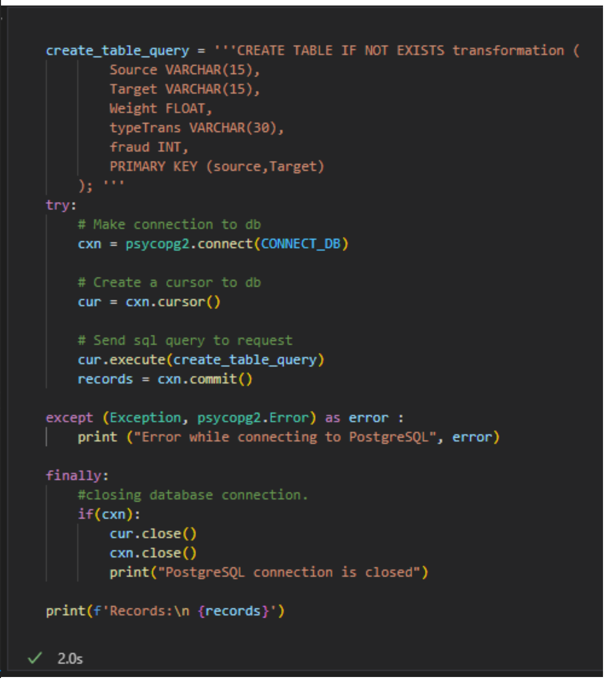
Create table for both csv file (customer_detail and transformation)

Melakukan create table query dengan menambahkan primary key pada column customer dan merchant pada table customer_detail, dan primary key source dan target pada table transformation. Primary key pada column customer / source saja akan menimbulkan error (duplicate value) sehingga memilih untuk membuat 2 primary key.



```python
create_table_query = '''CREATE TABLE IF NOT EXISTS customer_detail (
        step INT ,
        customer VARCHAR(15),
        age VARCHAR(10),
        gender VARCHAR(6),
        zipcodeOri VARCHAR(10),
        merchant VARCHAR(15),
        zipMerchant VARCHAR(10),
        category VARCHAR(30),
        amount FLOAT,
        fraud INT,
        PRIMARY KEY (customer,merchant)
    ); '''
try:
    # Make connection to db
    cxn = psycopg2.connect(CONNECT_DB)

    # Create a cursor to db
    cur = cxn.cursor()

    # Send sql query to request
    cur.execute(create_table_query)
    records = cxn.commit()

except (Exception, psycopg2.Error) as error :
    print ("Error while connecting to PostgreSQL", error)

finally:
    #closing database connection.
    if(cxn):
        cur.close()
        cxn.close()
        print("PostgreSQL connection is closed")

print(f'Records:\n {records}')
```

✓ 2.0s

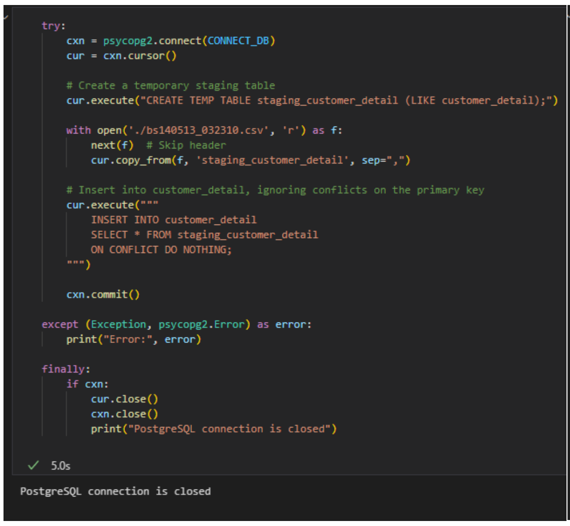


```python
create_table_query = '''CREATE TABLE IF NOT EXISTS transformation (
        Source VARCHAR(15),
        Target VARCHAR(15),
        Weight FLOAT,
        typeTrans VARCHAR(30),
        fraud INT,
        PRIMARY KEY (source,Target)
    ); '''
try:
    # Make connection to db
    cxn = psycopg2.connect(CONNECT_DB)

    # Create a cursor to db
    cur = cxn.cursor()

    # Send sql query to request
    cur.execute(create_table_query)
    records = cxn.commit()

except (Exception, psycopg2.Error) as error :
    print ("Error while connecting to PostgreSQL", error)

finally:
    #closing database connection.
    if(cxn):
        cur.close()
        cxn.close()
        print("PostgreSQL connection is closed")

print(f'Records:\n {records}')
```

✓ 2.0s

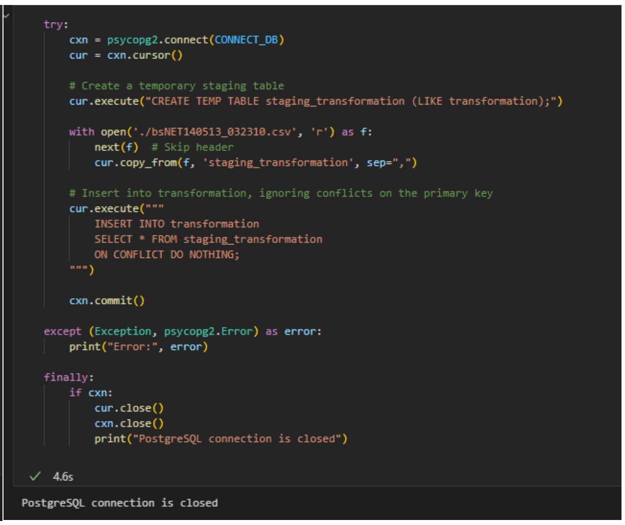
Add data to table

Menambahkan data pada kedua table dengan data csv yang telah diberikan. Inserting data dilakukan dengan membuat temporary table terlebih dahulu kemudian meng copy data dari temporary table ke main table dan Ketika ada conflik akan di skip.



```
try:
    cxn = psycopg2.connect(CONNECT_DB)
    cur = cxn.cursor()

    # Create a temporary staging table
    cur.execute("CREATE TEMP TABLE staging_customer_detail (LIKE customer_detail);")

    with open('./bs140513_032310.csv', 'r') as f:
        next(f)  # Skip header
        cur.copy_from(f, 'staging_customer_detail', sep=",")

    # Insert into customer_detail, ignoring conflicts on the primary key
    cur.execute("""
        INSERT INTO customer_detail
        SELECT * FROM staging_customer_detail
        ON CONFLICT DO NOTHING;
    """)

    cxn.commit()

except (Exception, psycopg2.Error) as error:
    print("Error:", error)

finally:
    if cxn:
        cur.close()
        cxn.close()
        print("PostgreSQL connection is closed")
```

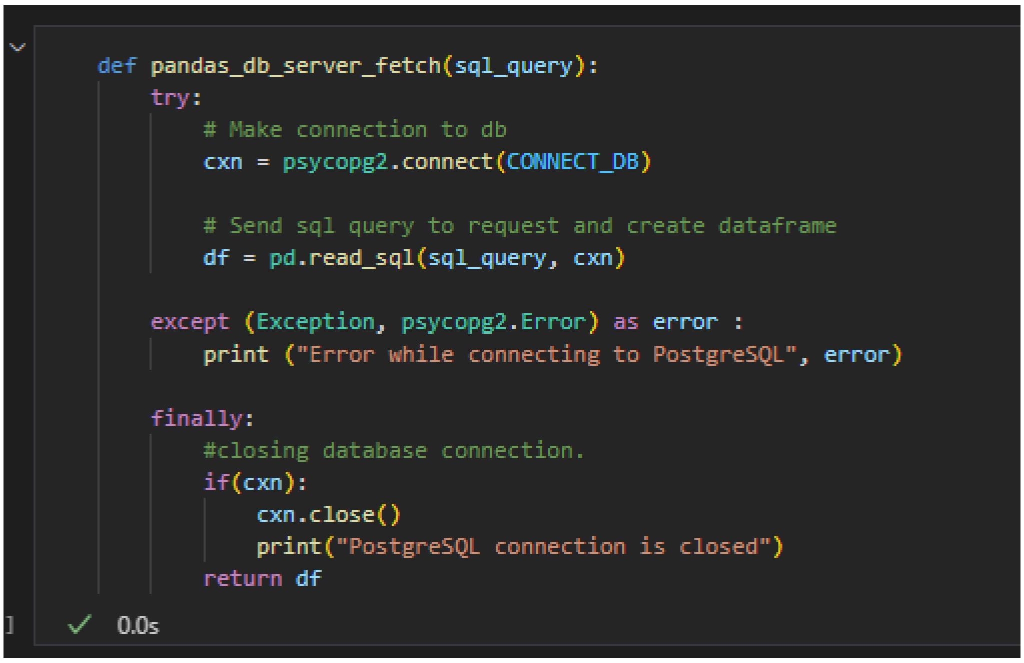
```
✓ 5.0s
PostgreSQL connection is closed
```



```
try:
    cxn = psycopg2.connect(CONNECT_DB)
    cur = cxn.cursor()

    # Create a temporary staging table
    cur.execute("CREATE TEMP TABLE staging_transformation (LIKE transformation);")

    with open('./bsNET140513_032310.csv', 'r') as f:
        next(f)  # Skip header
        cur.copy_from(f, 'staging_transformation', sep=",")

    # Insert into transformation, ignoring conflicts on the primary key
    cur.execute("""
        INSERT INTO transformation
        SELECT * FROM staging_transformation
        ON CONFLICT DO NOTHING;
    """)

    cxn.commit()

except (Exception, psycopg2.Error) as error:
    print("Error:", error)

finally:
    if cxn:
        cur.close()
        cxn.close()
        print("PostgreSQL connection is closed")
```

```
✓ 4.6s
PostgreSQL connection is closed
```

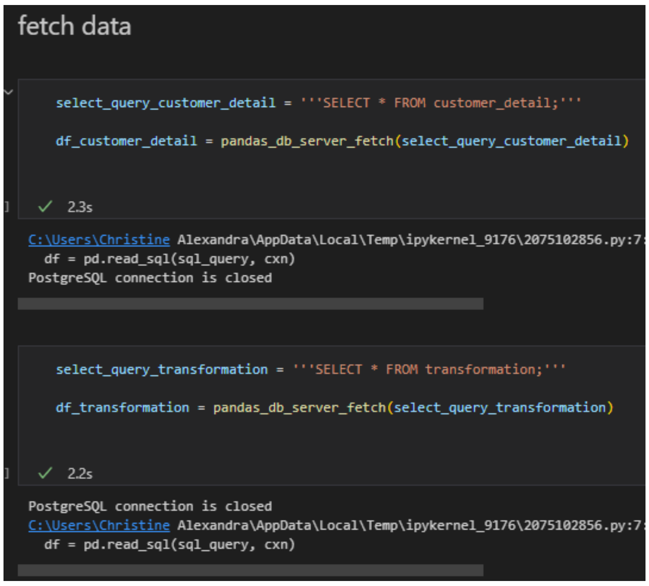
SQL fetching data function



```
def pandas_db_server_fetch(sql_query):
    try:
        # Make connection to db
        cxn = psycopg2.connect(CONNECT_DB)

        # Send sql query to request and create dataframe
        df = pd.read_sql(sql_query, cxn)

    except (Exception, psycopg2.Error) as error :
        print ("Error while connecting to PostgreSQL", error)

    finally:
        #closing database connection.
        if(cxn):
            cxn.close()
            print("PostgreSQL connection is closed")
        return df
```

```
✓ 0.0s
```

Data fetching to dataframe



fetch data

```python
select_query_customer_detail = '''SELECT * FROM customer_detail;'''

df_customer_detail = pandas_db_server_fetch(select_query_customer_detail)
```

✓ 2.3s

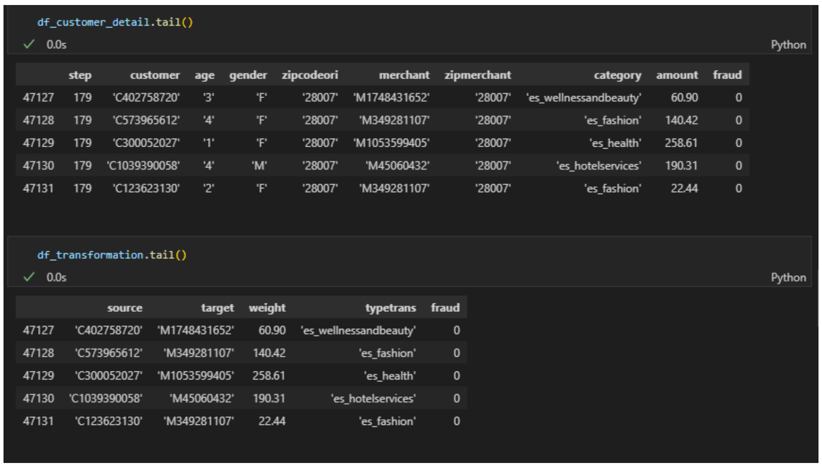
```
C:\Users\Christine Alexandra\AppData\Local\Temp\ipykernel_9176\2075102856.py:7:
  df = pd.read_sql(sql_query, cxn)
PostgreSQL connection is closed
```

```python
select_query_transformation = '''SELECT * FROM transformation;'''

df_transformation = pandas_db_server_fetch(select_query_transformation)
```

✓ 2.2s

```
PostgreSQL connection is closed
C:\Users\Christine Alexandra\AppData\Local\Temp\ipykernel_9176\2075102856.py:7:
  df = pd.read_sql(sql_query, cxn)
```

Data pada dataframe



```python
df_customer_detail.tail()
```

✓ 0.0s Python

	step	customer	age	gender	zipcodeori	merchant	zipmerchant	category	amount	fraud
47127	179	'C402758720'	'3'	'F'	'28007'	'M1748431652'	'28007'	'es_wellnessandbeauty'	60.90	0
47128	179	'C573965612'	'4'	'F'	'28007'	'M349281107'	'28007'	'es_fashion'	140.42	0
47129	179	'C300052027'	'1'	'F'	'28007'	'M1053599405'	'28007'	'es_health'	258.61	0
47130	179	'C1039390058'	'4'	'M'	'28007'	'M45060432'	'28007'	'es_hotelservices'	190.31	0
47131	179	'C123623130'	'2'	'F'	'28007'	'M349281107'	'28007'	'es_fashion'	22.44	0

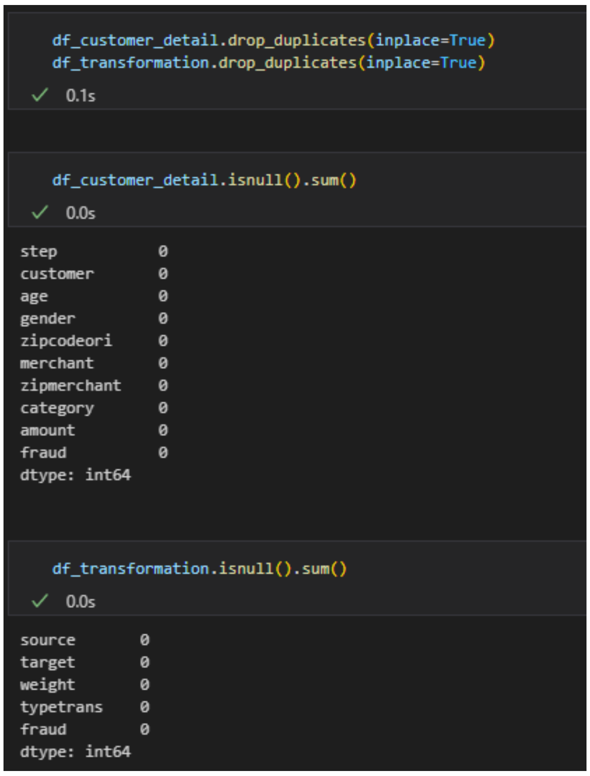
```python
df_transformation.tail()
```

✓ 0.0s Python

	source	target	weight	typetrans	fraud
47127	'C402758720'	'M1748431652'	60.90	'es_wellnessandbeauty'	0
47128	'C573965612'	'M349281107'	140.42	'es_fashion'	0
47129	'C300052027'	'M1053599405'	258.61	'es_health'	0
47130	'C1039390058'	'M45060432'	190.31	'es_hotelservices'	0
47131	'C123623130'	'M349281107'	22.44	'es_fashion'	0

Drop duplicate and null data

Menghapus data duplikat dan kosong. Dilakukan karena terdapat beberapa data duplikat pada primary key(customer_detail dan source).



```
df_customer_detail.drop_duplicates(inplace=True)
df_transformation.drop_duplicates(inplace=True)
```
✓ 0.1s

```
df_customer_detail.isnull().sum()
```
✓ 0.0s

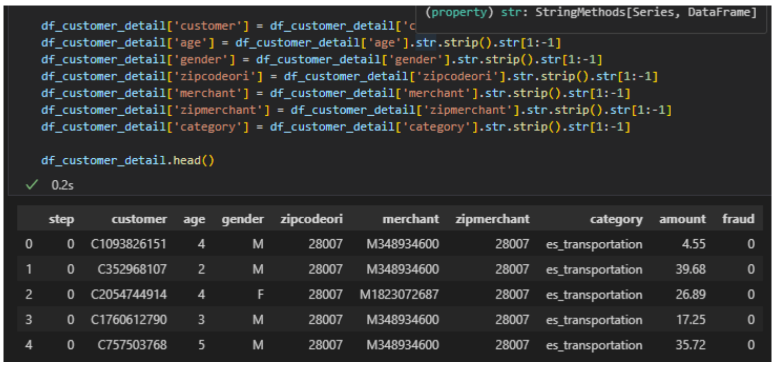
```
step           0
customer       0
age            0
gender         0
zipcodeori     0
merchant       0
zipmerchant    0
category       0
amount         0
fraud          0
dtype: int64
```

```
df_transformation.isnull().sum()
```
✓ 0.0s

```
source       0
target       0
weight       0
typetrans    0
fraud        0
dtype: int64
```

Removing whitespace and singlequote

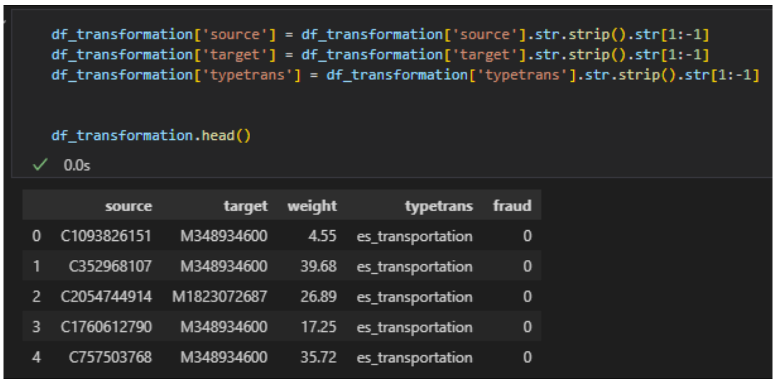
Menghapus data yang memiliki spasi kosong (` `) dan menghapus tanda kutip satu , sehingga data pada kolom terlihat lebih rapih dan mudah di identifikasi.



```
df_customer_detail['customer'] = df_customer_detail['c
df_customer_detail['age'] = df_customer_detail['age'].str.strip().str[1:-1]
df_customer_detail['gender'] = df_customer_detail['gender'].str.strip().str[1:-1]
df_customer_detail['zipcodeori'] = df_customer_detail['zipcodeori'].str.strip().str[1:-1]
df_customer_detail['merchant'] = df_customer_detail['merchant'].str.strip().str[1:-1]
df_customer_detail['zipmerchant'] = df_customer_detail['zipmerchant'].str.strip().str[1:-1]
df_customer_detail['category'] = df_customer_detail['category'].str.strip().str[1:-1]

df_customer_detail.head()
```
(property) str: StringMethods[Series, DataFrame]

✓ 0.2s

	step	customer	age	gender	zipcodeori	merchant	zipmerchant	category	amount	fraud
0	0	C1093826151	4	M	28007	M348934600	28007	es_transportation	4.55	0
1	0	C352968107	2	M	28007	M348934600	28007	es_transportation	39.68	0
2	0	C2054744914	4	F	28007	M1823072687	28007	es_transportation	26.89	0
3	0	C1760612790	3	M	28007	M348934600	28007	es_transportation	17.25	0
4	0	C757503768	5	M	28007	M348934600	28007	es_transportation	35.72	0



```
df_transformation['source'] = df_transformation['source'].str.strip().str[1:-1]
df_transformation['target'] = df_transformation['target'].str.strip().str[1:-1]
df_transformation['typetrans'] = df_transformation['typetrans'].str.strip().str[1:-1]


df_transformation.head()
```
✓ 0.0s

	source	target	weight	typetrans	fraud
0	C1093826151	M348934600	4.55	es_transportation	0
1	C352968107	M348934600	39.68	es_transportation	0
2	C2054744914	M1823072687	26.89	es_transportation	0
3	C1760612790	M348934600	17.25	es_transportation	0
4	C757503768	M348934600	35.72	es_transportation	0

Github repo : https://github.com/ChristopherRsl/df_assignment1