```
import numpy as np
import pickle

import psycopg2 as pg
import pandas.io.sql as psql
import pandas as pd

from typing import Union, List, Tuple

connection = pg.connect(host='pgsql-196447.vipserv.org', port=5432,
dbname='wbauer_adb', user='wbauer_adb', password='adb2020');

def film_in_category(category_id:int)->pd.DataFrame:
    ''' Funkcja zwracająca wynik zapytania do bazy o tytuł filmu, język, oraz
kategorię dla zadanego id kategorii.
    Przykład wynikowej tabeli:
    |   |title          |languge    |category|
    |0  |Amadeus Holy   |English    |Action|

    Tabela wynikowa ma być posortowana po tylule filmu i języku.

    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    category_id (int): wartość id kategorii dla którego wykonujemy zapytanie

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''

    quotation =   f"""SELECT film.title, language.name AS languge, category.name
AS category FROM film
                  INNER JOIN language ON film.language_id = language.language_id
                  INNER JOIN film_category ON film.film_id =
film_category.film_id
                  INNER JOIN category ON film_category.category_id =
category.category_id
                  WHERE category.category_id = {category_id}
                  ORDER by title, language.name"""


    return pd.read_sql_query(quotation, con=connection) if isinstance(category_id,
int) else None


def number_films_in_category(category_id:int)->pd.DataFrame:
    ''' Funkcja zwracająca wynik zapytania do bazy o ilość filmów w zadanej
kategori przez id kategorii.
    Przykład wynikowej tabeli:
    |   |category   |count|
    |0  |Action     |64   |
```

```
    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    category_id (int): wartość id kategorii dla którego wykonujemy zapytanie

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''


    quotation =   f"""SELECT category.name AS category, count(film.title)
                FROM film
                LEFT JOIN film_category ON film.film_id = film_category.film_id
                LEFT JOIN category ON film_category.category_id =
category.category_id
                WHERE category.category_id = {category_id}
                GROUP BY category.name"""

    return pd.read_sql_query(quotation, con=connection) if isinstance(category_id,
int) else None

def number_film_by_length(min_length: Union[int,float] = 0, max_length:
Union[int,float] = 1e6 ) :
    ''' Funkcja zwracająca wynik zapytania do bazy o ilość filmów o dla
poszczegulnych długości pomiędzy wartościami min_length a max_length.
    Przykład wynikowej tabeli:
    |   |length     |count|
    |0  |46         |64   |

    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    min_length (int,float): wartość minimalnej długości filmu
    max_length (int,float): wartość maksymalnej długości filmu

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''
    qoutation =   f"""SELECT length, count(title)
                  FROM film
                  WHERE length BETWEEN {min_length} AND {max_length}
                  GROUP BY length"""

    return pd.read_sql_query(qoutation, con=connection) if (isinstance(min_length,
(int, float)) and isinstance(max_length, (int, float)) and (min_length <
max_length))  else None


def client_from_city(city:str)->pd.DataFrame:
    ''' Funkcja zwracająca wynik zapytania do bazy o listę klientów z zadanego
miasta przez wartość city.
```

```
    Przykład wynikowej tabeli:
    |   |city       |first_name |last_name
    |0  |Athenai    |Linda      |Williams

    Tabela wynikowa ma być posortowana po nazwisku i imieniu klienta.

    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    city (str): nazwa miaste dla którego mamy sporządzić listę klientów

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''
    qoutation =   f"""SELECT city, customer.first_name, customer.last_name
                  FROM city
                  JOIN address ON address.city_id = city.city_id
                  JOIN customer ON address.address_id = customer.address_id
                  WHERE city.city = '{city}'
                  ORDER BY  customer.last_name desc, customer.first_name desc"""

    return pd.read_sql_query(qoutation, con=connection) if isinstance(city, str)
else None

def avg_amount_by_length(length:Union[int,float])->pd.DataFrame:
    ''' Funkcja zwracająca wynik zapytania do bazy o średnią wartość wypożyczenia
filmów dla zadanej długości length.
    Przykład wynikowej tabeli:
    |   |length |avg
    |0  |48     |4.295389


    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    length (int,float): długość filmu dla którego mamy pożyczyć średnią wartość
wypożyczonych filmów

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''
    qoutation =   f"""SELECT film.length, avg(payment.amount)
                  FROM film
                  JOIN inventory ON inventory.film_id = film.film_id
                  JOIN rental ON rental.inventory_id = inventory.inventory_id
                  JOIN payment ON payment.rental_id = rental.rental_id
                  WHERE film.length = {length}
                  GROUP BY film.length"""


    return pd.read_sql_query(qoutation, con=connection) if isinstance(length,
(int,float)) else None
```

```
def client_by_sum_length(sum_min:Union[int,float])->pd.DataFrame:
    ''' Funkcja zwracająca wynik zapytania do bazy o sumaryczny czas wypożyczonych
filmów przez klientów powyżej zadanej wartości .
    Przykład wynikowej tabeli:
    |   |first_name |last_name  |sum
    |0  |Brian      |Wyman      |1265

    Tabela wynikowa powinna być posortowane według sumy, imienia i nazwiska
klienta.
    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    sum_min (int,float): minimalna wartość sumy długości wypożyczonych filmów
którą musi spełniać klient

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''
    qoutation = f"""SELECT customer.first_name, customer.last_name,
sum(film.length)
                    FROM film
                    JOIN inventory ON inventory.film_id = film.film_id
                    JOIN rental ON rental.inventory_id = inventory.inventory_id
                    JOIN customer ON customer.customer_id = rental.customer_id
                    GROUP BY customer.first_name, customer.last_name
                    HAVING sum(film.length) > {sum_min}
                    ORDER BY sum(film.length), customer.last_name,
customer.first_name"""


    return pd.read_sql_query(qoutation, con=connection) if (isinstance(sum_min,
(int,float)) and sum_min >= 0) else None

def category_statistic_length(name:str)->pd.DataFrame:
    ''' Funkcja zwracająca wynik zapytania do bazy o statystykę długości filmów w
kategorii o zadanej nazwie.
    Przykład wynikowej tabeli:
    |   |category   |avg    |sum    |min    |max
    |0  |Action     |111.60 |7143   |47     |185

    Jeżeli warunki wejściowe nie są spełnione to funkcja powinna zwracać wartość
None.

    Parameters:
    name (str): Nazwa kategorii dla której ma zostać wypisana statystyka

    Returns:
    pd.DataFrame: DataFrame zawierający wyniki zapytania
    '''
    quotation =   f"""SELECT category.name as category, avg(film.length),
sum(film.length), min(film.length), max(film.length)
                FROM film
```

```
                JOIN film_category ON film_category.film_id = film.film_id
                JOIN category ON category.category_id = film_category.category_id
                WHERE category.name = '{name}'
                GROUP BY category.name"""

    return pd.read_sql_query(quotation, con=connection) if isinstance(name, str)
else None
```