НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря

СІКОРСЬКОГО»ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Лабораторна робота №2
з дисципліни
«Бази даних і засоби управління»
*Тема**:** «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»*

Виконав:
студент III курсу
групи КВ-93
Вітковський В.Б.
Перевірив:
Павловський В.І.

Київ 2021

Постановка задачі

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольногоінтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних убазі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більшесутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових
   – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значенняTrue/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Інформація про програму

Посилання на репозиторій у GitHub з вихідним кодом програми та звітом: https://github.com/Vvold/DataBase_lab

Використана мова програмування: Python 3.10

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), datetime (для роботи з датою і передачею її у запити до БД), time (для виміру часу запиту пошуку для завдання 3), sys (для реалізації консольного інтерфейсу).

Відомості про обрану предметну галузь з лабораторної роботи №1

Обрана предметна галузь передбачає можливість користувачеві вибрати зі списку пісні, та додавати їх у власну бібліотеку.

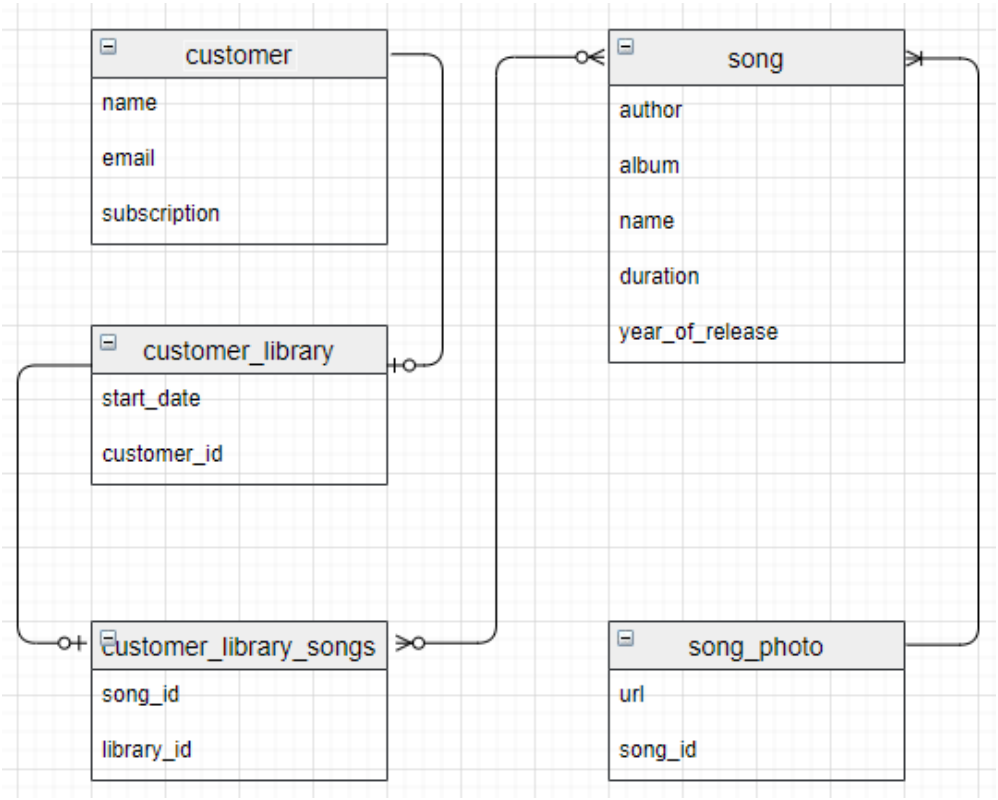


Рисунок 1. Схема бази даних

Таблиця 1. Опис структури БД

Відношення	Атрибут	Тип (розмір)
Відношення "customer" Вміщує інформацію про користувача сервісу	**id** - унікальний ID користувача **name** - ім'я користувача **email** - електронна пошта користувача **subscription** - інформація про те, має користувач підписку на сервіс, чи немає	Serial primary key Текстовий (255) Текстовий (255) Булевий тип (0,1)
Відношення "song" Вміщує інформацію про пісню, доступну на сервісі	**id** - унікальний ID пісні **author** - ім'я автора пісні **album** - назва альбому, з якого пісня **name** - назва пісні **duration** - тривалість пісні (в секундах) **year_of_release** - рік виходу пісні	Serial primary key Текстовий (255) Текстовий (255) Текстовий (255) Числовий Числовий
Відношення "song_photo" Вміщує інформацію про фото до пісні	**id** - унікальний ID фото пісні **url** - адреса зображення **song_id** - атрибут, який посилається на ID пісні	Serial primary key Текстовий (255) Числовий
Відношення "customer_library" Вміщує інформацію про бібліотеку пісень користувача	**id** - унікальний ID бібліотеки **start_date** - дата коли була додана перша пісня до бібліотеки **customer_id** - атрибут, який посилається на ID користувача	Serial primary key Числовий Числовий
Відношення "customer_library_songs" Вміщує інформацію про пісні, які є у бібліотеці користувача	**id** - унікальний ID користувача **library_id** - атрибут, який посилається на ID бібліотеки **song_id** - атрибут, який посилається на ID пісні	Serial primary key Числовий Числовий

Схема меню користувача

```
PS C:\my\Python\BD_lab2>  python main.py help
print_table - outputs the specified table
        argument (table_name) is required
delete_record - deletes the specified record from table
        arguments (table_name, key_name, key_value) are required
update_record - updates record with specified id in table
        customer args (table_name, id, name, email, subscription)
        customer_library args (table_name, id, start_date, customer_id)
        customer_library_songs args (table_name, id, library_id, song_id)
        song args (table_name, id, author, album, name, duration, year_of_release)
        song_photo args (table_name, id, url, song_id)
insert_record - inserts record into specified table
        customer args (table_name, id, name, email, subscription)
        customer_library args (table_name, id, start_date, customer_id)
        customer_library_songs args (table_name, id, library_id, song_id)
        song args (table_name, id, author, album, name, duration, year_of_release)
        song_photo args (table_name, id, url, song_id)
generate_randomly - generates n random records in table
        arguments (table_name, n) are required
search_records - search for records in two or more tables using one or more keys
        arguments

PS C:\my\Python\BD_lab2>
```

На знімку екрану терміналу продемонстовано виконання команди help, що показує усі доступні користувачу команди, коротко описує їх та надає список обов'язкових аргументів. Кожна команда запускає відповідний метод об'єкту класу Controller, який реалізує передачу аргументів у клас View на перевірку і за умови їх коректності, Controller далі передає ці аргументи у клас Model, що здійснює запит до бази даних.

Методи реалізовані до пункту 1 завдання лабораторної роботи:

print_table – за умови коректності імені таблиці виводить вміст цієї таблиці у вікно терміналу. Аргументом може бути одне із імен:

customer, customer_library, customer_library_songs, song, song_photo

delete_record – за умови правильності введених аргументів, наявності відповідного запису (з вказаним значенням первинного ключа) та незалежності інших таблиць від цього запису (до цього запису немає зовнішнього ключа з іншої таблиці), видаляє запис з вказаним первинним ключем. Аргументами є:

table_name, key_name, key_value

update_record – за умови правильності введених аргументів, наявності відповідного запису (з вказаним значенням первинного ключа) та записів інших таблиць (на які хочемо змінити поточні), змінює усі поля, окрім первинного ключа у обраному записі. Аргументи різні для кожної таблиці:

customer: id(int), name(str), email(str), subscription(bool)

customer_library: id(int), start_date(int), customer_id(int)

customer_library_songs: id(int), library_id(int), song_id(int)

song: id(int), author(str), album(str), name(str), duration(int),
year_of release(int)

song_photo: id(int), url(str), song_id(int)

insert_record – за умови правильності введених аргументів, відсутності відповідного запису (з вказаним значенням первинного ключа) та наявності записів інших таблиць (на які хочемо посилатись зі створеного запису), вставляє новий рядок у таблицю з обраними значеннями полів. Аргументи різні для кожної таблиці:

customer: id(int), name(str), email(str), subscription(str)

customer_library: id(int), start_date(int), customer_id(int)

customer_library_songs: id(int), library_id(int), song_id(int)

song: id(int), author(str), album(str), name(str), duration(int),
year_of release(str)

song_photo: id(int), url(str), song_id(int)

Метод реалізований до пункту 2 завдання лабораторної роботи:

generate_randomly – за умови введення правильного імені таблиці та числа n відмінного від нуля, здійснює генерування n псевдорандомізованих записів у обраній таблиці. Аргементами є ім’я таблиці та часло записів, що мають бути створені.

Метод реалізований до пункту 3 завдання лабораторної роботи:

search_records – за умови введення потрібної кількості аргументів та правильного задання умов пошуку, реалізує пошук за 1 та більше атрибутами з вказаних таблиць (від двох до п’яти) і виводить у вікно терміналу результат пошуку (або нічого, якщо

пошук не дав результатів) та час, за який було проведено запит. Початково потрібно вказати агрументи.

Після вказання цієї інформації потрібно буде вказати кількість атрибутів для пошуку, а тип пошуку, ім’я атрибуту (обов’язково з вказанням до якої таблиці з перелічених аргументів він відноситься: one.key_name, two.key_name, three.key_name, four.key_name або five.key_name), та значення (спочатку лівий кінець інтервалу, потім правий для числового пошуку та пошуку за датою, або рядок для пошуку за ключовим словом). Спочатку вказуються всі дані для першого атрибуту, потім для другого і т.д. до введеної кількості атрибутів.

Завдання 1

Запит на видалення

Для перевірки роботи розглянемо запити на видалення дочірньої таблиці customer та батьківської таблиці customer_library.

Таблиця song_photo до видалення запису 4:

```
PS C:\my\Python\BD_lab2> python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
------------------------------------------------------------------------------------------
id: 2   url: song_2 photo       song_id: 2
------------------------------------------------------------------------------------------
id: 3   url: song_3 photo       song_id: 3
------------------------------------------------------------------------------------------
id: 4   url: song_4 photo       song_id: 4
------------------------------------------------------------------------------------------
```

Таблиця song_photo після видалення запису 4:

```
PS C:\my\Python\BD_lab2> python main.py delete_record song_photo id 4
select count(*) from public."song_photo" where id=4
DELETE FROM public."song_photo" WHERE id=4;
PS C:\my\Python\BD_lab2> python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
------------------------------------------------------------------------------------------
id: 2   url: song_2 photo       song_id: 2
------------------------------------------------------------------------------------------
id: 3   url: song_3 photo       song_id: 3
------------------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

У даній програмній реалізації видалення запису з батьківської таблиці, який зв’язаний з дочірньою таблицею, не буде здійснено, а буде видано повідомлення про помилку.

Таблиця song до видалення запису 4

```
PS C:\my\Python\BD_lab2> python main.py print_table song
SELECT * FROM public."song"
song table:
id: 1   author: Author_1        album: Album_1  name: song_1    duration: 240   year_of_release: 1999
--------------------------------------------------------------------------------
id: 2   author: Author_2        album: Album_2  name: song_2    duration: 250   year_of_release: 2000
--------------------------------------------------------------------------------
id: 3   author: Author_3        album: Album_3  name: song_3    duration: 260   year_of_release: 2001
--------------------------------------------------------------------------------
id: 4   author: Author_4        album: Album_4  name: song_4    duration: 270   year_of_release: 2002
--------------------------------------------------------------------------------
```

Спроба видалення запису 4 з таблиці song:

```
PS C:\my\Python\BD_lab2> python main.py delete_record song id 4
select count(*) from public."song" where id=4
select count(*) from public."customer_library_songs" where id=4
select count(*) from public."song_photo" where id=4
this record is connected with another table, deleting will throw error
PS C:\my\Python\BD_lab2>
```

Запит на вставку поля

Для перевірки роботи розглянемо запити на вставки в дочірню таблицю song_photo. Спочатку коректний, потім з неіснуючим значенням зовнішнього ключа батьківської таблиці song.

Таблиця song_photo до вставки запису 4:

```
PS C:\my\Python\BD_lab2> python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo      song_id: 1
--------------------------------------------------------------------------------
id: 2   url: song_2 photo      song_id: 2
--------------------------------------------------------------------------------
id: 3   url: song_3 photo      song_id: 3
--------------------------------------------------------------------------------
```

Таблиця song_phoro після вставки запису 4:

```
PS C:\my\Python\BD_lab2> python main.py insert_record song_photo 4 song_4_photo 4
select count(*) from public."song_photo" where id=4
select count(*) from public."song" where id=4
insert into public."song_photo" (id, url, song_id) VALUES (4, 'song_4_photo', '4');
PS C:\my\Python\BD_lab2> python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
--------------------------------------------------------------------------------
id: 2   url: song_2 photo       song_id: 2
--------------------------------------------------------------------------------
id: 3   url: song_3 photo       song_id: 3
--------------------------------------------------------------------------------
id: 4   url: song_4_photo       song_id: 4
--------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Записи у батьківській таблиці song:

```
PS C:\my\Python\BD_lab2> python main.py print_table song
SELECT * FROM public."song"
song table:
id: 1   author: Author_1        album: Album_1  name: song_1    duration: 240   year_of_release: 1999
--------------------------------------------------------------------------------
id: 2   author: Author_2        album: Album_2  name: song_2    duration: 250   year_of_release: 2000
--------------------------------------------------------------------------------
id: 3   author: Author_3        album: Album_3  name: song_3    duration: 260   year_of_release: 2001
--------------------------------------------------------------------------------
id: 4   author: Author_4        album: Album_4  name: song_4    duration: 270   year_of_release: 2002
--------------------------------------------------------------------------------
```

Спроба вставки запису у дочірню таблицю song_photo з неіснуючим зовнішнім ключем 5:

```
PS C:\my\Python\BD_lab2> python main.py insert_record song_photo 4 song_4_photo 5
select count(*) from public."song_photo" where id=4
select count(*) from public."song" where id=5
Something went wrong (record with such id exists or inappropriate foreign key values)
PS C:\my\Python\BD_lab2>
```

Запит на зміну полів

Для перевірки роботи розглянемо запити на зміну значеннння в дочірній таблиці Song_photo. Спочатку коректний, потім з неіснуючим значенням зовнішнього ключа батьківської таблиці song.
Таблиця song_photo до зміни запису 4:

```
PS C:\my\Python\BD_lab2> python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
_____________________________________________________________________________________________
id: 2   url: song_2 photo       song_id: 2
_____________________________________________________________________________________________
id: 3   url: song_3 photo       song_id: 3
_____________________________________________________________________________________________
id: 4   url: song_4_photo       song_id: 4
_____________________________________________________________________________________________
```

Таблиця song_photo після зміни запису 4:

```
PS C:\my\Python\BD_lab2>  python main.py update_record song_photo 4 song_5_photo 3
select count(*) from public."song_photo" where id=4
select count(*) from public."song" where id=3
UPDATE public."song_photo" SET url='song_5_photo', song_id=3 WHERE id=4;
PS C:\my\Python\BD_lab2>  python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
_____________________________________________________________________________________________
id: 2   url: song_2 photo       song_id: 2
_____________________________________________________________________________________________
id: 3   url: song_3 photo       song_id: 3
_____________________________________________________________________________________________
id: 4   url: song_5_photo       song_id: 3
_____________________________________________________________________________________________
PS C:\my\Python\BD_lab2>
```

Записи у батьківській таблиці song:

```
PS C:\my\Python\BD_lab2> python main.py print_table song
SELECT * FROM public."song"
song table:
id: 1   author: Author_1        album: Album_1  name: song_1    duration: 240   year_of_release: 1999
_____________________________________________________________________________________
id: 2   author: Author_2        album: Album_2  name: song_2    duration: 250   year_of_release: 2000
_____________________________________________________________________________________
id: 3   author: Author_3        album: Album_3  name: song_3    duration: 260   year_of_release: 2001
_____________________________________________________________________________________
id: 4   author: Author_4        album: Album_4  name: song_4    duration: 270   year_of_release: 2002
_____________________________________________________________________________________
PS C:\my\Python\BD_lab2>
```

Спроба зміни запису у дочірній таблиці song_photo з неіснуючим зовнішнім ключем 5:

```
PS C:\my\Python\BD_lab2> python main.py update_record song_photo 4 song_5_photo 5
select count(*) from public."song_photo" where id=4
select count(*) from public."song" where id=5
Something went wrong (record with such id does not exist or inappropriate foreign key value)
PS C:\my\Python\BD_lab2>
```

Завдання 2

Вставка 5 псевдорандомізованих записів у кожну з таблиць.

Початкова таблиця customer:

```
PS C:\my\Python\BD_lab2> python main.py print_table customer
SELECT * FROM public."customer"
customer table:
id: 1   name: Andriy    email: andriy@gmail.com         subscription: True
-----------------------------------------------------------------------------------------
id: 2   name: Vitaliy   email: vitaliy@gmail.com        subscription: False
-----------------------------------------------------------------------------------------
id: 3   name: Bogdan    email: bogdan@gmail.com         subscription: True
-----------------------------------------------------------------------------------------
id: 4   name: Oleg      email: oleg@gmail.com   subscription: True
-----------------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Запит:

```
PS C:\my\Python\BD_lab2> python main.py generate_randomly customer 5
insert into public.customer select (SELECT (MAX(id)+1) FROM public.customer), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: i
nteger)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT TRUE );
insert into public.customer select (SELECT (MAX(id)+1) FROM public.customer), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: i
nteger)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT TRUE );
insert into public.customer select (SELECT (MAX(id)+1) FROM public.customer), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: i
nteger)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT TRUE );
insert into public.customer select (SELECT (MAX(id)+1) FROM public.customer), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: i
nteger)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT TRUE );
insert into public.customer select (SELECT (MAX(id)+1) FROM public.customer), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: i
nteger)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT TRUE );
```

Модифікована таблиця customer:

```
PS C:\my\Python\BD_lab2> python main.py print_table customer
SELECT * FROM public."customer"
customer table:
id: 1   name: Andriy    email: andriy@gmail.com         subscription: True
-----------------------------------------------------------------------------------------
id: 2   name: Vitaliy   email: vitaliy@gmail.com        subscription: False
-----------------------------------------------------------------------------------------
id: 3   name: Bogdan    email: bogdan@gmail.com         subscription: True
-----------------------------------------------------------------------------------------
id: 4   name: Oleg      email: oleg@gmail.com   subscription: True
-----------------------------------------------------------------------------------------
id: 5   name: nohx      email: gxrflnkc         subscription: True
-----------------------------------------------------------------------------------------
id: 6   name: nedbuv    email: dkhddjel         subscription: True
-----------------------------------------------------------------------------------------
id: 7   name: hzh       email: orddmt   subscription: True
-----------------------------------------------------------------------------------------
id: 8   name: extypqmgm         email: tdjt     subscription: True
-----------------------------------------------------------------------------------------
id: 9   name: carrb     email: sqirrfwe         subscription: True
-----------------------------------------------------------------------------------------
```

Початкова таблиця customer_library:

```
PS C:\my\Python\BD_lab2> python main.py print_table customer_library
SELECT * FROM public."customer_library"
customer_library table:
id: 1   start_date: 191832      customer_id: 1
-----------------------------------------------------------------------------------------
id: 2   start_date: 191900      customer_id: 3
-----------------------------------------------------------------------------------------
id: 3   start_date: 193333      customer_id: 2
-----------------------------------------------------------------------------------------
```

Запит:

```
PS C:\my\Python\BD_lab2> python main.py generate_randomly customer_library 5
insert into public.customer_library select (SELECT MAX(id)+1 FROM public.customer_library), FLOOR(RANDOM()*(200000-1)+1),(SELECT id FROM public.customer LIM
IT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer)-1))));
insert into public.customer_library select (SELECT MAX(id)+1 FROM public.customer_library), FLOOR(RANDOM()*(200000-1)+1),(SELECT id FROM public.customer LIM
IT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer)-1))));
insert into public.customer_library select (SELECT MAX(id)+1 FROM public.customer_library), FLOOR(RANDOM()*(200000-1)+1),(SELECT id FROM public.customer LIM
IT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer)-1))));
insert into public.customer_library select (SELECT MAX(id)+1 FROM public.customer_library), FLOOR(RANDOM()*(200000-1)+1),(SELECT id FROM public.customer LIM
IT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer)-1))));
insert into public.customer_library select (SELECT MAX(id)+1 FROM public.customer_library), FLOOR(RANDOM()*(200000-1)+1),(SELECT id FROM public.customer LIM
IT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer)-1))));
```

Модифікована таблиця customer_library:

```
PS C:\my\Python\BD_lab2> python main.py print_table customer_library
SELECT * FROM public."customer_library"
customer_library table:
id: 1   start_date: 191832      customer_id: 1
------------------------------------------------------------------------------------------
id: 2   start_date: 191900      customer_id: 3
------------------------------------------------------------------------------------------
id: 3   start_date: 193333      customer_id: 2
------------------------------------------------------------------------------------------
id: 4   start_date: 124468      customer_id: 2
------------------------------------------------------------------------------------------
id: 5   start_date: 100263      customer_id: 2
------------------------------------------------------------------------------------------
id: 6   start_date: 78273       customer_id: 3
------------------------------------------------------------------------------------------
id: 7   start_date: 159248      customer_id: 2
------------------------------------------------------------------------------------------
id: 8   start_date: 55696       customer_id: 3
------------------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Початкова таблиця customer_library_songs:

```
PS C:\my\Python\BD_lab2>  python main.py print_table customer_library_songs
SELECT * FROM public."customer_library_songs"
customer_library_songs table:
id: 1   library_id: 1   song_id: 1
------------------------------------------------------------------------------------------
id: 2   library_id: 1   song_id: 2
------------------------------------------------------------------------------------------
id: 3   library_id: 1   song_id: 3
------------------------------------------------------------------------------------------
id: 4   library_id: 1   song_id: 4
------------------------------------------------------------------------------------------
id: 5   library_id: 2   song_id: 1
------------------------------------------------------------------------------------------
id: 6   library_id: 2   song_id: 2
------------------------------------------------------------------------------------------
id: 7   library_id: 3   song_id: 2
------------------------------------------------------------------------------------------
id: 8   library_id: 3   song_id: 3
------------------------------------------------------------------------------------------
id: 9   library_id: 3   song_id: 4
------------------------------------------------------------------------------------------
```

Запит:

```
PS C:\my\Python\BD_lab2> python main.py generate_randomly customer_library_songs 5
insert into public.customer_library_songs select (SELECT MAX(id)+1 FROM public.customer_library_songs), (SELECT id FROM public.customer_library LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer_library)-1)))), (SELECT id FROM public.song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.customer_library_songs select (SELECT MAX(id)+1 FROM public.customer_library_songs), (SELECT id FROM public.customer_library LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer_library)-1)))), (SELECT id FROM public.song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.customer_library_songs select (SELECT MAX(id)+1 FROM public.customer_library_songs), (SELECT id FROM public.customer_library LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer_library)-1)))), (SELECT id FROM public.song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.customer_library_songs select (SELECT MAX(id)+1 FROM public.customer_library_songs), (SELECT id FROM public.customer_library LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer_library)-1)))), (SELECT id FROM public.song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.customer_library_songs select (SELECT MAX(id)+1 FROM public.customer_library_songs), (SELECT id FROM public.customer_library LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.customer_library)-1)))), (SELECT id FROM public.song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
```

Модифікована таблиця customer_library_songs:

```
PS C:\my\Python\BD_lab2>  python main.py print_table customer_library_songs
SELECT * FROM public."customer_library_songs"
customer_library_songs table:
id: 1   library_id: 1   song_id: 1
------------------------------------------------------------------------------------------
id: 2   library_id: 1   song_id: 2
------------------------------------------------------------------------------------------
id: 3   library_id: 1   song_id: 3
------------------------------------------------------------------------------------------
id: 4   library_id: 1   song_id: 4
------------------------------------------------------------------------------------------
id: 5   library_id: 2   song_id: 1
------------------------------------------------------------------------------------------
id: 6   library_id: 2   song_id: 2
------------------------------------------------------------------------------------------
id: 7   library_id: 3   song_id: 2
------------------------------------------------------------------------------------------
id: 8   library_id: 3   song_id: 3
------------------------------------------------------------------------------------------
id: 9   library_id: 3   song_id: 4
------------------------------------------------------------------------------------------
id: 10  library_id: 1   song_id: 3
------------------------------------------------------------------------------------------
id: 11  library_id: 3   song_id: 3
------------------------------------------------------------------------------------------
id: 12  library_id: 2   song_id: 4
------------------------------------------------------------------------------------------
id: 13  library_id: 3   song_id: 1
------------------------------------------------------------------------------------------
id: 14  library_id: 1   song_id: 4
------------------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Початкова таблиця song:

```
PS C:\my\Python\BD_lab2> python main.py print_table song
SELECT * FROM public."song"
song table:
id: 1   author: Author_1        album: Album_1  name: song_1    duration: 240   year_of_release: 1999
------------------------------------------------------------------------------------------
id: 2   author: Author_2        album: Album_2  name: song_2    duration: 250   year_of_release: 2000
------------------------------------------------------------------------------------------
id: 3   author: Author_3        album: Album_3  name: song_3    duration: 260   year_of_release: 2001
------------------------------------------------------------------------------------------
id: 4   author: Author_4        album: Album_4  name: song_4    duration: 270   year_of_release: 2002
------------------------------------------------------------------------------------------
```

Запит:

```
PS C:\my\Python\BD_lab2> python main.py generate_randomly song 5
insert into public.song select (SELECT (MAX(id)+1) FROM public.song), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: i
nteger) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 2
5)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''), FLOOR(RANDOM()*(200000-1)+1),FLOOR(RANDOM()*(2021-1)
+1);
insert into public.song select (SELECT (MAX(id)+1) FROM public.song), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: i
nteger) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 2
5)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''), FLOOR(RANDOM()*(200000-1)+1),FLOOR(RANDOM()*(2021-1)
+1);
insert into public.song select (SELECT (MAX(id)+1) FROM public.song), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: i
nteger) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 2
5)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''), FLOOR(RANDOM()*(200000-1)+1),FLOOR(RANDOM()*(2021-1)
+1);
insert into public.song select (SELECT (MAX(id)+1) FROM public.song), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: i
nteger) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 2
5)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''), FLOOR(RANDOM()*(200000-1)+1),FLOOR(RANDOM()*(2021-1)
+1);
insert into public.song select (SELECT (MAX(id)+1) FROM public.song), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: i
nteger) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),array_to_string(ARRAY(SELECT chr((97 + round(random() * 2
5)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''), FLOOR(RANDOM()*(200000-1)+1),FLOOR(RANDOM()*(2021-1)
+1);
```

Модифікована таблиця song:

```
PS C:\my\Python\BD_lab2> python main.py print_table song
SELECT * FROM public."song"
song table:
id: 1   author: Author_1        album: Album_1  name: song_1    duration: 240   year_of_release: 1999
-----------------------------------------------------------------------------------------
id: 2   author: Author_2        album: Album_2  name: song_2    duration: 250   year_of_release: 2000
-----------------------------------------------------------------------------------------
id: 3   author: Author_3        album: Album_3  name: song_3    duration: 260   year_of_release: 2001
-----------------------------------------------------------------------------------------
id: 4   author: Author_4        album: Album_4  name: song_4    duration: 270   year_of_release: 2002
-----------------------------------------------------------------------------------------
id: 5   author: ujrit   album: hfwwqb   name: firrodg   duration: 144626        year_of_release: 594
-----------------------------------------------------------------------------------------
id: 6   author: ueaskcyh        album: fxhhszx  name: cusat     duration: 54720         year_of_release: 312
-----------------------------------------------------------------------------------------
id: 7   author: ckivkijik       album: cdbrdcp  name: huxsjcr   duration: 175775        year_of_release: 1057
-----------------------------------------------------------------------------------------
id: 8   author: rgvshsifk       album: wtqwvftd         name: vnnkfpe   duration: 151762        year_of_release: 1026
-----------------------------------------------------------------------------------------
id: 9   author: gnjyxpbom       album: nkmelgss         name: cuuvhvxqx         duration: 130528        year_of_release: 507
-----------------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Початкова таблиця song_photo:

```
PS C:\my\Python\BD_lab2>  python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
-----------------------------------------------------------------------------------------
id: 2   url: song_2 photo       song_id: 2
-----------------------------------------------------------------------------------------
id: 3   url: song_3 photo       song_id: 3
-----------------------------------------------------------------------------------------
id: 4   url: song_5_photo       song_id: 3
-----------------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Запит:

```
PS C:\my\Python\BD_lab2> python main.py generate_randomly song_photo 5
insert into public.song_photo select (SELECT MAX(id)+1 FROM public.song_photo), array_to_string(ARRAY(SELECT chr((97 + ro
und(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT id FROM public.
song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.song_photo select (SELECT MAX(id)+1 FROM public.song_photo), array_to_string(ARRAY(SELECT chr((97 + ro
und(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT id FROM public.
song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.song_photo select (SELECT MAX(id)+1 FROM public.song_photo), array_to_string(ARRAY(SELECT chr((97 + ro
und(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT id FROM public.
song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.song_photo select (SELECT MAX(id)+1 FROM public.song_photo), array_to_string(ARRAY(SELECT chr((97 + ro
und(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT id FROM public.
song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
insert into public.song_photo select (SELECT MAX(id)+1 FROM public.song_photo), array_to_string(ARRAY(SELECT chr((97 + ro
und(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),(SELECT id FROM public.
song LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public.song)-1))));
```

Модифікована таблиця song_photo:

```
PS C:\my\Python\BD_lab2> python main.py print_table song_photo
SELECT * FROM public."song_photo"
song_photo table:
id: 1   url: song_1 photo       song_id: 1
-------------------------------------------------------------------------------
id: 2   url: song_2 photo       song_id: 2
-------------------------------------------------------------------------------
id: 3   url: song_3 photo       song_id: 3
-------------------------------------------------------------------------------
id: 4   url: song_5_photo       song_id: 3
-------------------------------------------------------------------------------
id: 5   url: fybse      song_id: 3
-------------------------------------------------------------------------------
id: 6   url: vlj        song_id: 1
-------------------------------------------------------------------------------
id: 7   url: bwlbvvfp   song_id: 4
-------------------------------------------------------------------------------
id: 8   url: duihuoqiv  song_id: 4
-------------------------------------------------------------------------------
id: 9   url: jnoqvh     song_id: 2
-------------------------------------------------------------------------------
PS C:\my\Python\BD_lab2>
```

Завдання 3

Пошук за двома атрибутами з двох таблиць(customer, customer_library)
Формування запиту:

```
PS C:\my\Python\BD_lab2>  python main.py search_records customer customer_library id id
specify the number of attributes you`d like to search by: 2
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.customer_id
specify the left end of search interval: 0
specify the right end of search interval: 5
select * from public.customer as one inner join public.customer_library as two on one.id=two.id where 0<one.id and one.id<5 and 0<two.c
ustomer_id and two.customer_id<5
--- 0.007371664047241211 seconds ---
```

Запит:

```
select * from public.customer as one inner join public.customer_library as two on one.id=two.id where 0<one.id and one.id<5 and 0<two.customer_id and two.customer_id<5;
```

Результат:

```
search result:
1
Andriy
andriy@gmail.com
True
1
191832
1
--------------------------------------------------------------------------------
2
Vitaliy
vitaliy@gmail.com
False
2
191900
3
--------------------------------------------------------------------------------
3
Bogdan
bogdan@gmail.com
True
3
193333
2
--------------------------------------------------------------------------------
4
Oleg
oleg@gmail.com
True
4
124468
2
--------------------------------------------------------------------------------
```

Пошук за трьома атрибутами з трьох таблиць (customer, customer_library, customer_library_songs)

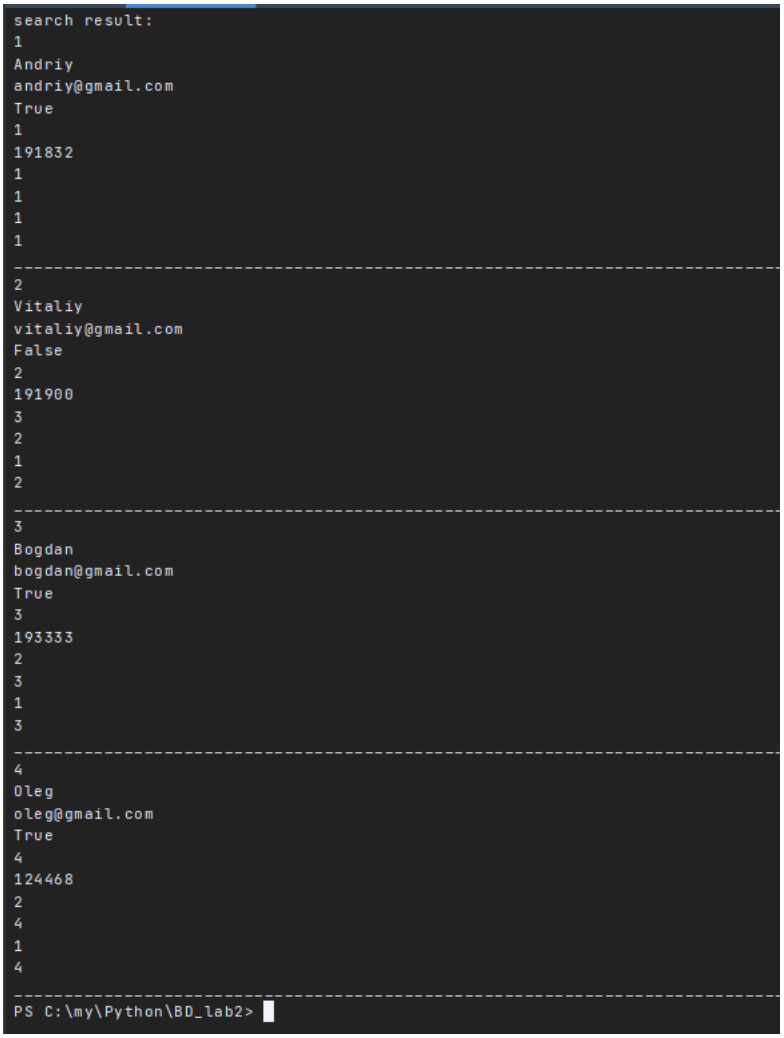
Формування запиту:

```
PS C:\my\Python\BD_lab2>  python main.py search_records customer customer_library customer_library_songs id id id id
specify the number of attributes you`d like to search by: 3
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.customer_id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.song_id
specify the left end of search interval: 0
specify the right end of search interval: 5
select * from public.customer as one inner join public.customer_library as two on one.id=two.id inner join public.customer_library_songs as three on three.id=one.id where 0<one.id and one.id<5 and 0<two.customer_id and two.customer_id<5 and 0<three.song_id and three.song_id<5
--- 0.009779930114746094 seconds ---
```

Запит:

```
select * from public.customer as one inner join public.customer_library as two on one.id=two.id inner join public.customer_library_songs as three on three.id=one.id where 0<one.id and one.id<5 and 0<two.customer_id and two.customer_id<5 and 0<three.song_id and three.song_id<5;
```

Результат:



Пошук за чотирьма атрибутами з чотирьох таблиць (customer, customer_library, customer_library_songs, song)

Формування запиту:

```
PS C:\my\Python\BD_lab2>  python main.py search_records customer customer_library customer_library_songs song id id id id id id
specify the number of attributes you`d like to search by: 4
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.customer_id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.library_id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric or string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: four.duration
specify the left end of search interval: 0
specify the right end of search interval: 260
select * from public.customer as one inner join public.customer_library as two on one.id=two.id inner join public.customer_library_song
s as three on three.id=one.id inner join public.song as four on four.id=two.id where 0<one.id and one.id<5 and 0<two.customer_id and tw
o.customer_id<5 and 0<three.library_id and three.library_id<5 and 0<four.duration and four.duration<260
--- 0.02150726318359375 seconds ---
```

Запит:

```
select * from public.customer as one inner join public.customer_library as two on one.id=two.id inner join public.customer_library_songs as three on three.id=one.id inner join public.song as four on four.id=two.id where 0<one.id and one.id<5 and 0<two.customer_id and two.customer_id<5 and 0<three.library_id and three.library_id<5 and 0<four.duration and four.duration<260;
```

Результат:

```
search result:
1
Andriy
andriy@gmail.com
True
1
191832
1
1
1
1
1
Author_1
Album_1
song_1
240
1999
-----------------------------------------------
2
Vitaliy
vitaliy@gmail.com
False
2
191900
3
2
1
2
2
Author_2
Album_2
song_2
250
2000
-----------------------------------------------
PS C:\my\Python\BD_lab2>
```

Завдання 4

Програмний код модулю “model.py”:

```
import psycopg2 as ps


class Model:
    def __init__(self):
        self.conn = None
        try:
            self.conn = ps.connect(
                database="music_service",
                user='postgres',
                password="Vb09gh54ty12",
                host='127.0.0.1',
                port="5432",
            )
        except(Exception, ps.DatabaseError) as error:
```

```
            print("[INFO] Error while working with Postgresql", error)

    def request(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return True
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as error:
            print(error)
            self.conn.rollback()
            return False

    def get(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return cursor.fetchall()
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as error:
            print(error)
            self.conn.rollback()
            return False

    def get_el(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return cursor.fetchone()
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as error:
            print(error)
            self.conn.rollback()
            return False

    def count(self, table_name: str):
        return self.get_el(f"select count(*) from public.\"{table_name}\"")

    def find(self, table_name: str, key_name: str, key_value: int):
        return self.get_el(f"select count(*) from public.\"{table_name}\" where
{key_name}={key_value}")

    def max(self, table_name: str, key_name: str):
        return self.get_el(f"select max({key_name}) from public.\"{table_name}\"")

    def min(self, table_name: str, key_name: str):
        return self.get_el(f"select min({key_name}) from public.\"{table_name}\"")

    def print_customer(self) -> None:
        return self.get(f"SELECT * FROM public.\"customer\"")

    def print_customer_library(self) -> None:
        return self.get(f"SELECT * FROM public.\"customer_library\"")

    def print_customer_library_songs(self) -> None:
        return self.get(f"SELECT * FROM public.\"customer_library_songs\"")

    def print_song(self) -> None:
        return self.get(f"SELECT * FROM public.\"song\"")

    def print_song_photo(self) -> None:
        return self.get(f"SELECT * FROM public.\"song_photo\"")
```

```
    def delete_data(self, table_name: str, key_name: str, key_value) -> None:
        self.request(f"DELETE FROM public.\"{table_name}\" WHERE {key_name}={key_value};")

    def update_data_customer(self, key_value: int, name: str, email: str, subscription: bool) ->
None:
        self.request(
            f"UPDATE public.\"customer\" SET name=\'{name}\', email=\'{email}\',
subscription={subscription} WHERE id ={key_value};")

    def update_data_customer_library(self, key_value: int, start_date: int, customer_id: int) ->
None:
        self.request(f"UPDATE public.\"customer_library\" SET start_date= {start_date},
customer_id={customer_id}"
                     f"WHERE id ={key_value};")

    def update_data_customer_library_songs(self, key_value: int, library_id: int, song_id: int)
-> None:
        self.request(f"UPDATE public.\"customer_library_songs\" SET library_id={library_id},
song_id={song_id}"
                     f"WHERE id={key_value};")

    def update_data_song(self, key_value: int, author: str, album: str, name: str, duration:
int,
                         year_of_release: str) -> None:
        self.request(
            f"UPDATE public.\"song\" SET author=\'{author}\', album=\'{album}\',
name=\'{name}\', duration= {duration},"
            f"year_of_release = \'{year_of_release}\' WHERE id={key_value};")

    def update_data_song_photo(self, key_value: int, url: str, song_id: int) -> None:
        self.request(f"UPDATE public.\"song_photo\" SET url=\'{url}\', song_id={song_id} WHERE
id={key_value};")

    def insert_data_customer(self, id: int, name: str, email: str, subscription: str) -> None:
        self.request(f"insert into public.\"customer\" (id, name, email, subscription) "
                     f"VALUES ({id}, \'{name}\', \'{email}\', \'{subscription})\';")

    def insert_data_customer_library(self, id: int, start_date: int, customer_id: int) -> None:
        self.request(f"insert into public.\"customer_library\" (id, start_date, customer_id) "
                     f"VALUES ({id}, {start_date}, {customer_id});")

    def insert_data_customer_library_songs(self, id: int, library_id: int, song_id: int) ->
None:
        self.request(f"insert into public.\"customer_library_songs\" (id, library_id, song_id)
"
                     f"VALUES ({id}, {library_id}, {song_id});")

    def insert_data_song(self, id: int, author: str, album: str, name: str, duration: int,
                         year_of_release: int) -> None:
        self.request(f"insert into public.\"song\" (id, author, album, name, duration,
year_of_release) "
                     f"VALUES ({id}, \'{author}\', \'{album}\', \'{name}\', {duration},
{year_of_release});")

    def insert_data_song_photo(self, id: int, url: str, song_id: str) -> None:
        self.request(f"insert into public.\"song_photo\" (id, url, song_id) "
                     f"VALUES ({id}, \'{url}\', \'{song_id}\');")

    # поміняти цю функцію
    def customer_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.customer select (SELECT (MAX(id)+1) FROM
public.customer), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) ::
```

```
integer) "
                        "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),"
                        "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) ::
integer) "
                        "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),"
                        "(SELECT TRUE  or FALSE );")

    def customer_library_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.customer_library select (SELECT MAX(id)+1 FROM
public.customer_library), "
                        "FLOOR(RANDOM()*(200000-1)+1),"
                        "(SELECT id FROM public.customer LIMIT 1 OFFSET "
                        "(round(random() *((SELECT COUNT(id) FROM public.customer)-1))));")

    def customer_library_songs_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request(
                "insert into public.customer_library_songs select (SELECT MAX(id)+1 FROM
public.customer_library_songs), "
                "(SELECT id FROM public.customer_library LIMIT 1 OFFSET "
                "(round(random() *((SELECT COUNT(id) FROM public.customer_library)-1)))), "
                "(SELECT id FROM public.song LIMIT 1 OFFSET "
                "(round(random() *((SELECT COUNT(id) FROM public.song)-1))));")

    def song_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.song select (SELECT (MAX(id)+1) FROM public.song),
"
                        "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) ::
integer) "
                        "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),"
                        "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) ::
integer) "
                        "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),"
                        "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) ::
integer) "
                        "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''), "
                        "FLOOR(RANDOM()*(200000-1)+1),"
                        "FLOOR(RANDOM()*(2021-1)+1); ")

    def song_photo_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.song_photo select (SELECT MAX(id)+1 FROM
public.song_photo), "
                        "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) ::
integer) "
                        "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3):: integer)), ''),"
                        "(SELECT id FROM public.song LIMIT 1 OFFSET "
                        "(round(random() *((SELECT COUNT(id) FROM public.song)-1))));")

    def search_data_two_tables(self, table1_name: str, table2_name: str, table1_key, table2_key,
                               search: str):
        return self.get(f"select * from public.{table1_name} as one inner join
public.{table2_name} as two "
                        f"on one.{table1_key}=two.{table2_key} "
                        f"where {search}")

    def search_data_three_tables(self, table1_name: str, table2_name: str, table3_name: str,
                                 table1_key, table2_key, table3_key, table13_key,
                                 search: str):
        return self.get(f"select * from public.{table1_name} as one inner join
public.{table2_name} as two "
                        f"on one.{table1_key}=two.{table2_key} inner join public.{table3_name}
as three "
```

```
                        f"on three.{table3_key}=one.{table13_key} "
                        f"where {search}")

    def search_data_four_tables(self, table1_name: str, table2_name: str, table3_name: str,
table4_name: str,
                                table1_key, table2_key, table3_key, table13_key,
                                table4_key, table24_key,
                                search: str):
        return self.get(f"select * from public.{table1_name} as one inner join
public.{table2_name} as two "
                        f"on one.{table1_key}=two.{table2_key} inner join public.{table3_name}
as three "
                        f"on three.{table3_key}=one.{table13_key} inner join
public.{table4_name} as four "
                        f"on four.{table4_key}=two.{table24_key} "
                        f"where {search}")
```

Опис функцій модуля:

Модуль “model.py” слугує точкую доступу до бази даних. Для реалізації запитів користувача до бази даних використовується бібліотека psycopg2.

У модулі використані такі функції:

1. request, get, get_el – здійснюють запити до бази даних. При правильному запиті request повертає True, get повертає усі дані що було взято з запитів SELECT (масив кортежів з записами таблиць), get_el повертає тільки перший запис. У разі помилки вони повертають False;
2. max, min – повертають максимальне і мінімальне значення зазначеного ключа у таблиці;
3. count – повертає кількість усіх записів у таблиці;
4. find – повертає кількість записів таблиці, що відповідають заданій користувачем умові
5. print_customer – отримання з бази даних та виведення у консоль користувача таблиці “customer”;
6. print_customer_library– отримання з бази даних та виведення у консоль користувача таблиці “ Product_discount”;
7. print_customer_library_songs – отримання з бази даних та виведення у консоль користувача таблиці “customer_library_songs”;
8. print_song – отримання з бази даних та виведення у консоль користувача таблиці “song”;
9. print_song_photo - отримання з бази даних та виведення у консоль користувача таблиці “song_photo”;
10. delete_data – реалізує видалення запису з обраної користувачем таблиці;
11. update_data_(назва таблиці) – реалізує запит за зміну даних у обраній користувачем таблиці;
12. insert_data_(назва таблиці) – реалізує запит на вставку запису до обраної користувачем таблиці;
13. (назва таблиці)_data_generator – реалізує запит на вставку рандомізованих записів до обраної користувачем таблиці;
14. search_data_(кількість таблиць)_tables – реалізує пошук даних у вибраній користувачем кількості таблиць;