НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота №2

з дисципліни **Бази даних і засоби управління**
на тему: "Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL"

Виконала: студентка 3 курсу
групи КВ-93
Баранова Є.В.
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 консольного інтерфейсу;
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі;
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/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/BaranovaEugenia/DB_Lab2

Використана мова програмування: Python 3.9.

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), time (для виміру часу запиту пошуку для завдання 3), sys (для реалізації консольного інтерфейсу).

Відомості про обрану предметну галузь з лабораторної роботи №1

Модель «сутність-зв’язок» галузі продажу напоїв в кав’ярні.

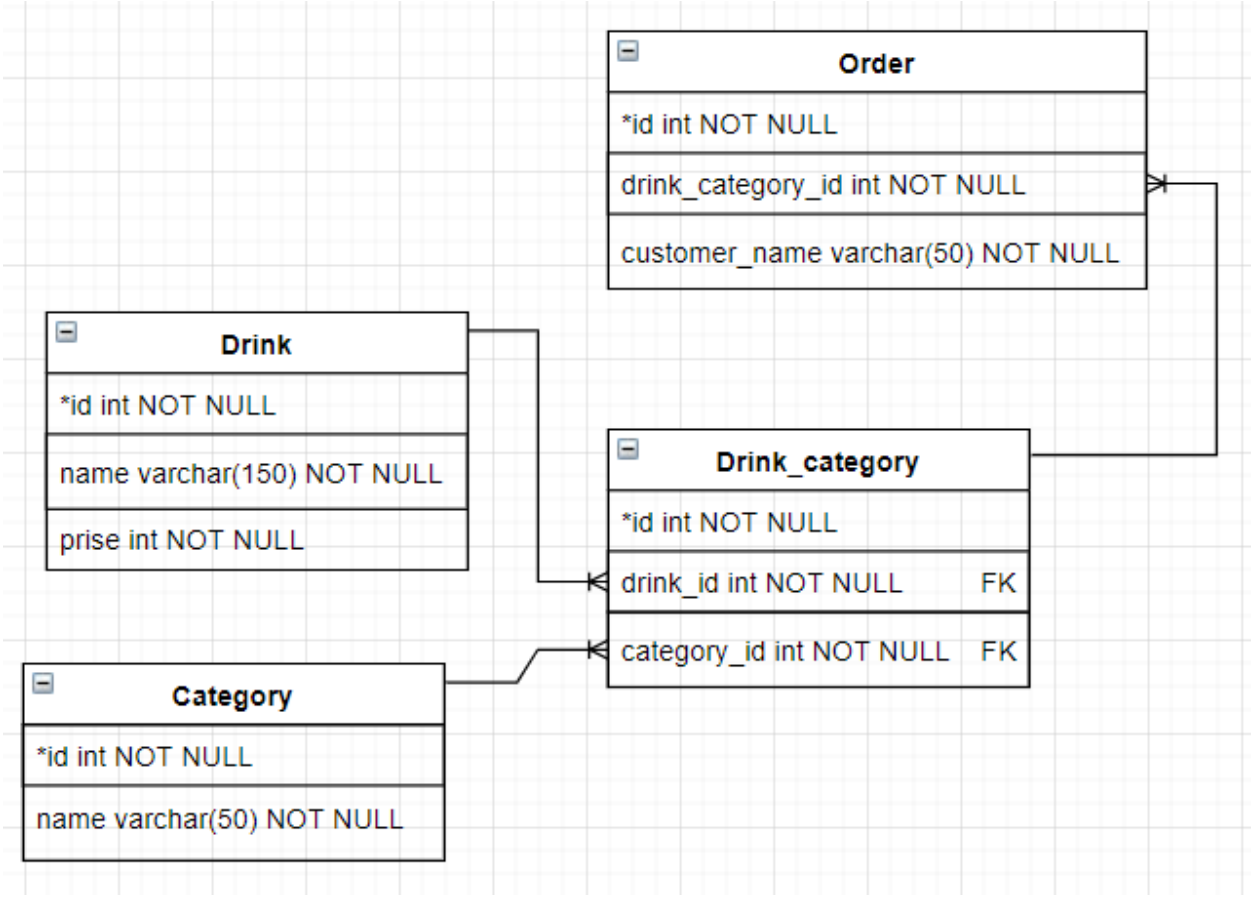


Рисунок 1 – Логічна модель

Таблиця 1 – Опис структури

Відношення	Атрибут (кожен атрибут не допускає NULL)	Тип
Drink (Інформація про конкретний напій)	id – унікальний ID напою в БД name – назва напою price – ціна напою	Числовий Текстовий (150) Числовий
Order (Інформація про замовлення)	id – унікальний ID замовлення в БД drink_category_id – ID конкретного напою customer_name – ім’я клієнта	Числовий Числовий Текстовий (50)
Drink_category (Інформація про категорію та напій)	id – унікальний ID клієнта в БД drink_id – ID напою category_id – ID категорії	Числовий Числовий Числовий
Category (Інформація про категорії напоїв)	id – унікальний ID категорії напою в БД name – назва типу напою	Числовий Текстовий (50)

Модель має чотири сутності: order, drink, drink_category, category.

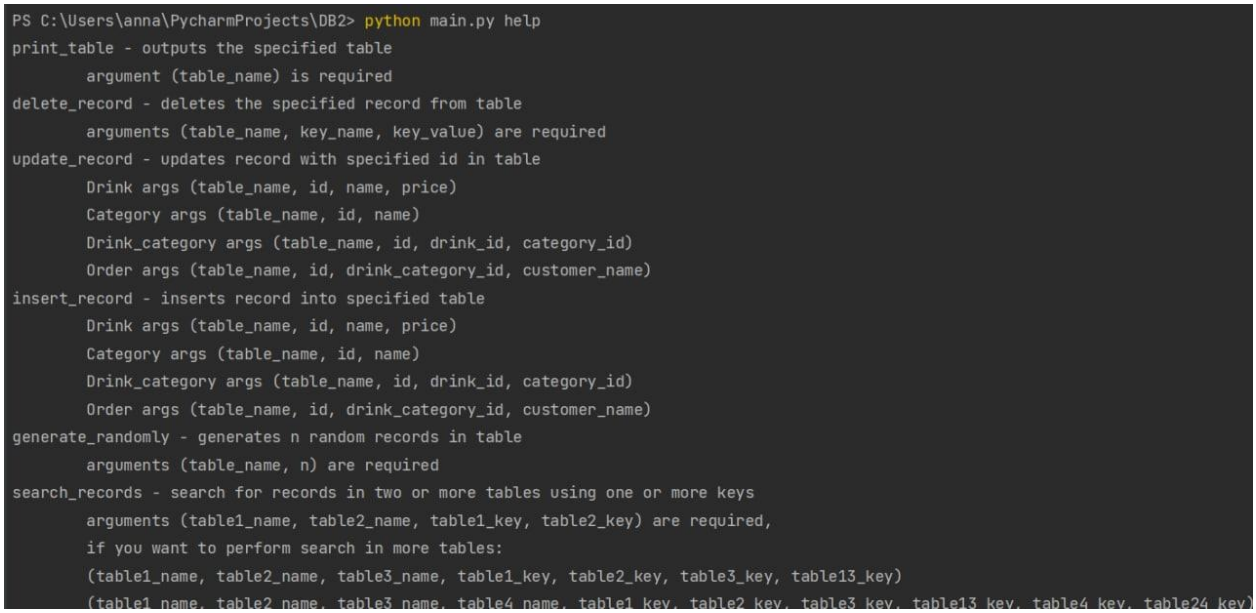
Drink: містить інформацію про напій. Кожен напій має власний id, назву і ціну.

Order: містить інформацію про клієнта і обраний ним напій. Кожне замовлення має власне id, напій і власника.

Drink_category: містить інформацію про конкретній напій. Містить id напою і id категорії. Слугує для зв'язування напою та категорії.

Category: містить інформацію про категорії напоїв. Кожна категорія має власний id та назву.

Схема меню користувача



Команда help показує усі доступні користувачу команди, коротко описує їх та надає список обов'язкових аргументів.

Методи реалізовані до пункту 1 завдання лабораторної роботи:

print_table – виводить вміст цієї таблиці у вікно терміналу. Аргументом може бути одне із імен: Drink, Category, Drink_category, Order.

delete_record – видаляє запис з вказаним первинним ключем. Аргументами є table_name, key_name, key_value

update_record – змінює усі поля, окрім первинного ключа у обраному записі. Аргументи різні для кожної таблиці:

Drink id (int) name(str) price(int)

Category id (int) name(str)

Drink_category id (int) drink_id(int) category_id(int)

Order id (int) drink_category_id(int) customer_name(str)

insert_record – вставляє новий рядок у таблицю з обраними значеннями полів. Аргументи також різні для кожної таблиці.

Метод реалізований до пункту 2 завдання лабораторної роботи:

generate_randomly – здійснює генерування n псевдорандомізованих записів у обраній таблиці. Аргументами є ім’я таблиці та число записів, що мають бути створені.

Метод реалізований до пункту 3 завдання лабораторної роботи:

search_records – реалізує пошук за 1 та більше атрибутами з вказаних таблиць і виводить у вікно терміналу результат пошуку та час, за який було проведено запит. Початково потрібно вказати агрументи:

table1_name table2_name table1_key table2_key або
table1_name table2_name table3_name table1_key table2_key table3_key table13_key або
table1_name table2_name table3_name table4_name table1_key table2_key table3_key table13_key table4_key table24_key
де table13_key, table24_key – це зовнішні ключі, що зв’язують 1 та 3 таблицю, або 2 та 4.

Після вказання цієї інформації потрібно буде вказати кількість атрибутів для пошуку, а тип пошуку, ім’я атрибуту, та значення.

Завдання 1

Запит на видалення

Для перевірки роботи розглянемо запити на видалення дочірньої таблиці Drink_category та батьківської таблиці Drink.

Таблиця Drink_category до видалення запису 10:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink_category
SELECT * FROM public."Drink_category"
Drink_category table:
id: 2   drink_id: 1     category_id: 2
-----------------------------------
id: 3   drink_id: 1     category_id: 3
-----------------------------------
id: 4   drink_id: 1     category_id: 4
-----------------------------------
id: 5   drink_id: 2     category_id: 3
-----------------------------------
id: 7   drink_id: 3     category_id: 4
-----------------------------------
id: 8   drink_id: 4     category_id: 1
-----------------------------------
id: 10  drink_id: 5     category_id: 4
-----------------------------------
id: 9   drink_id: 2     category_id: 3
-----------------------------------
id: 1   drink_id: 2     category_id: 3
-----------------------------------
```

Таблиця Drink_category після видалення запису 10:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py delete_record Drink_category id 10
select count(*) from public."Drink_category" where id=10
select count(*) from public."Order" where id=10
DELETE FROM public."Drink_category" WHERE id=10;
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink_category
SELECT * FROM public."Drink_category"
Drink_category table:
id: 2   drink_id: 1     category_id: 2
-----------------------------------
id: 3   drink_id: 1     category_id: 3
-----------------------------------
id: 4   drink_id: 1     category_id: 4
-----------------------------------
id: 5   drink_id: 2     category_id: 3
-----------------------------------
id: 7   drink_id: 3     category_id: 4
-----------------------------------
id: 8   drink_id: 4     category_id: 1
-----------------------------------
id: 9   drink_id: 2     category_id: 3
-----------------------------------
id: 1   drink_id: 2     category_id: 3
-----------------------------------
```

Таблиця Drink до видалення запису 1:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink
SELECT * FROM public."Drink"
Drink table:
id: 2   name: Espresso  price: 30
------------------------------------
id: 3   name: Hot chocolate     price: 25
------------------------------------
id: 5   name: Cacao     price: 25
------------------------------------
id: 1   name: Amerikano         price: 30
------------------------------------
id: 4   name: Green_tea         price: 15
------------------------------------
id: 6   name: Latte     price: 20
------------------------------------
```

Спроба видалення запису 1 з таблиці Drink:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py delete_record Drink id 1
select count(*) from public."Drink" where id=1
select count(*) from public."Drink_category" where id=1
this record is connected with another table, deleting will throw error
```

Запит на вставку поля

Для перевірки роботи розглянемо запити на вставки в дочірню таблицю Order.

Таблиця Order до вставки запису 6:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id: 3   drink_category_id: 1    customer_name: Sonya
-----------------------------------
id: 5   drink_category_id: 3    customer_name: Vova
-----------------------------------
id: 4   drink_category_id: 9    customer_name: Masha
-----------------------------------
id: 1   drink_category_id: 2    customer_name: Vanya
-----------------------------------
id: 2   drink_category_id: 11   customer_name: Anna
-----------------------------------
```

Таблиця Order після вставки запису 6:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py insert_record Order 6 11 Katya
select count(*) from public."Order" where id=6
select count(*) from public."Drink_category" where id=11
insert into public."Order" (id, drink_category_id, customer_name) VALUES (6, '11', 'Katya');
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id: 3   drink_category_id: 1    customer_name: Sonya
-----------------------------------
id: 5   drink_category_id: 3    customer_name: Vova
-----------------------------------
id: 4   drink_category_id: 9    customer_name: Masha
-----------------------------------
id: 1   drink_category_id: 2    customer_name: Vanya
-----------------------------------
id: 2   drink_category_id: 11   customer_name: Anna
-----------------------------------
id: 6   drink_category_id: 11   customer_name: Katya
-----------------------------------
```

Записи у батьківській таблиці Drink_category:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink_category
SELECT * FROM public."Drink_category"
Drink_category table:
id: 2   drink_id: 1     category_id: 2
-----------------------------------
id: 4   drink_id: 1     category_id: 4
-----------------------------------
id: 5   drink_id: 2     category_id: 3
-----------------------------------
id: 7   drink_id: 3     category_id: 4
-----------------------------------
id: 8   drink_id: 4     category_id: 1
-----------------------------------
id: 9   drink_id: 2     category_id: 3
-----------------------------------
id: 1   drink_id: 2     category_id: 3
-----------------------------------
id: 3   drink_id: 4     category_id: 2
-----------------------------------
id: 6   drink_id: 1     category_id: 3
-----------------------------------
```

Спроба вставки запису у дочірню таблицю Order з неіснуючим зовнішнім ключем 110:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py insert_record Order 12 110 Anna
select count(*) from public."Order" where id=12
select count(*) from public."Drink_category" where id=110
Something went wrong (record with such id exists or inappropriate foreign key values)
```

Запит на зміну полів

Для перевірки роботи розглянемо запити на вставки в дочірню таблицю Order.

Таблиця Order до зміни запису 2:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id: 2   drink_category_id: 10   customer_name: Bogdan
-----------------------------------
id: 3   drink_category_id: 1    customer_name: Sonya
-----------------------------------
id: 5   drink_category_id: 3    customer_name: Vova
-----------------------------------
id: 4   drink_category_id: 9    customer_name: Masha
-----------------------------------
id: 1   drink_category_id: 2    customer_name: Vanya
-----------------------------------
```

Таблиця Order після зміни запису 2:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py update_record Order 2 11 Anna
select count(*) from public."Order" where id=2
select count(*) from public."Drink_category" where id=11
UPDATE public."Order" SET drink_category_id='11', customer_name='Anna' WHERE id=2;
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id: 3   drink_category_id: 1    customer_name: Sonya
-----------------------------------
id: 5   drink_category_id: 3    customer_name: Vova
-----------------------------------
id: 4   drink_category_id: 9    customer_name: Masha
-----------------------------------
id: 1   drink_category_id: 2    customer_name: Vanya
-----------------------------------
id: 2   drink_category_id: 11   customer_name: Anna
-----------------------------------
```

Записи у батьківській таблиці Drink_category:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink_category
SELECT * FROM public."Drink_category"
Drink_category table:
id: 2   drink_id: 1     category_id: 2
-----------------------------------
id: 4   drink_id: 1     category_id: 4
-----------------------------------
id: 5   drink_id: 2     category_id: 3
-----------------------------------
id: 7   drink_id: 3     category_id: 4
-----------------------------------
id: 8   drink_id: 4     category_id: 1
-----------------------------------
id: 9   drink_id: 2     category_id: 3
-----------------------------------
id: 1   drink_id: 2     category_id: 3
-----------------------------------
id: 3   drink_id: 4     category_id: 2
-----------------------------------
id: 6   drink_id: 1     category_id: 3
-----------------------------------
```

Спроба вставки запису у дочірню таблицю Order з неіснуючим зовнішнім ключем 12:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py update_record Order 1 12 Kostya
select count(*) from public."Order" where id=1
select count(*) from public."Drink_category" where id=12
Something went wrong (record with such id does not exist or inappropriate foreign key value)
```

Завдання 2

Запит на видалення

Вставка 2 псевдорандомізованих записів у кожну з таблиць.
Початкова таблиця Category:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Category
SELECT * FROM public."Category"
Category table:
id: 2   name: Without sugar, cold
-----------------------------------
id: 4   name: Without sugar, hot
-----------------------------------
id: 1   name: With_sugar_Cold
-----------------------------------
id: 3   name: With_sugar_Hot
-----------------------------------
```

Модифікована таблиця Category:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py generate_randomly Category 2
insert into public."Category"select (SELECT MAX(id)+1 FROM public."Category"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                    FROM generate_series(1, FLOOR(
RANDOM()*(10-4)+4):: integer)), '');
insert into public."Category"select (SELECT MAX(id)+1 FROM public."Category"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                    FROM generate_series(1, FLOOR(
RANDOM()*(10-4)+4):: integer)), '');
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Category
SELECT * FROM public."Category"
Category table:
id: 2   name: Without sugar, cold
-----------------------------------
id: 4   name: Without sugar, hot
-----------------------------------
id: 1   name: With_sugar_Cold
-----------------------------------
id: 3   name: With_sugar_Hot
-----------------------------------
id: 5   name: supe
-----------------------------------
id: 6   name: niupfeyw
```

Початкова таблиця Drink:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink
SELECT * FROM public."Drink"
Drink table:
id: 2   name: Espresso  price: 30
-----------------------------------
id: 3   name: Hot chocolate     price: 25
-----------------------------------
id: 5   name: Cacao     price: 25
-----------------------------------
id: 1   name: Amerikano         price: 30
-----------------------------------
id: 4   name: Green_tea         price: 15
-----------------------------------
id: 6   name: Latte     price: 20
```

Модифікована таблиця Drink:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py generate_randomly Drink 2
insert into public."Drink"select (SELECT MAX(id)+1 FROM public."Drink"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                    FROM generate_series(1, FLOOR(RANDOM
()*(10-4)+4):: integer)), ''), FLOOR(RANDOM()*(100000-1)+1);
insert into public."Drink"select (SELECT MAX(id)+1 FROM public."Drink"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                    FROM generate_series(1, FLOOR(RANDOM
()*(10-4)+4):: integer)), ''), FLOOR(RANDOM()*(100000-1)+1);
```

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink
SELECT * FROM public."Drink"
Drink table:
id: 2   name: Espresso  price: 30

-------------------------------------
id: 3   name: Hot chocolate     price: 25

-------------------------------------
id: 5   name: Cacao     price: 25

-------------------------------------
id: 1   name: Amerikano         price: 30

-------------------------------------
id: 4   name: Green_tea         price: 15

-------------------------------------
id: 6   name: Latte     price: 20

-------------------------------------
id: 7   name: gpjdb     price: 41637

-------------------------------------
id: 8   name: lwctluvms         price: 63794
```

Початкова таблиця Drink_category:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink_category
SELECT * FROM public."Drink_category"
Drink_category table:
id: 2   drink_id: 1     category_id: 2

---------------------------------------
id: 4   drink_id: 1     category_id: 4

---------------------------------------
id: 5   drink_id: 2     category_id: 3

---------------------------------------
id: 7   drink_id: 3     category_id: 4

---------------------------------------
id: 8   drink_id: 4     category_id: 1

---------------------------------------
id: 9   drink_id: 2     category_id: 3

---------------------------------------
id: 1   drink_id: 2     category_id: 3

---------------------------------------
id: 3   drink_id: 4     category_id: 2

---------------------------------------
id: 6   drink_id: 1     category_id: 3
```

Модифікована таблиця Drink_category:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py generate_randomly Drink_category 2
LECT id FROM public."Category" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Category")-1)));
insert into public."Drink_category" select (SELECT (MAX(id)+1) FROM public."Drink_category"), (SELECT id FROM public."Drink" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Drink")-1)))), (SE
LECT id FROM public."Category" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Category")-1))));
```

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Drink_category
SELECT * FROM public."Drink_category"
Drink_category table:
id: 2   drink_id: 1     category_id: 2
--------------------------------------
id: 4   drink_id: 1     category_id: 4
--------------------------------------
id: 5   drink_id: 2     category_id: 3
--------------------------------------
id: 7   drink_id: 3     category_id: 4
--------------------------------------
id: 8   drink_id: 4     category_id: 1
--------------------------------------
id: 9   drink_id: 2     category_id: 3
--------------------------------------
id: 1   drink_id: 2     category_id: 3
--------------------------------------
id: 3   drink_id: 4     category_id: 2
--------------------------------------
id: 6   drink_id: 1     category_id: 3
--------------------------------------
id: 10  drink_id: 4     category_id: 4
--------------------------------------
id: 11  drink_id: 6     category_id: 4
```

Початкова таблиця Order:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id: 3   drink_category_id: 1    customer_name: Sonya
--------------------------------------
id: 5   drink_category_id: 3    customer_name: Vova
--------------------------------------
id: 4   drink_category_id: 9    customer_name: Masha
--------------------------------------
id: 1   drink_category_id: 4    customer_name: Vanya
--------------------------------------
id: 2   drink_category_id: 2    customer_name: Anna
```

Модифікована таблиця Order:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py generate_randomly Order 2
insert into public."Order" select (SELECT MAX(id)+1 FROM public."Order"), (SELECT id FROM public."Drink_category" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Drink_category")-1)))), array
_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(15-5)+5):: integer)), '');
insert into public."Order" select (SELECT MAX(id)+1 FROM public."Order"), (SELECT id FROM public."Drink_category" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Drink_category")-1)))), array
_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(15-5)+5):: integer)), '');
```

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id: 3   drink_category_id: 1    customer_name: Sonya
--------------------------------------
id: 5   drink_category_id: 3    customer_name: Vova
--------------------------------------
id: 4   drink_category_id: 9    customer_name: Masha
--------------------------------------
id: 1   drink_category_id: 4    customer_name: Vanya
--------------------------------------
id: 2   drink_category_id: 2    customer_name: Anna
--------------------------------------
id: 6   drink_category_id: 5    customer_name: srioymdyly
--------------------------------------
id: 7   drink_category_id: 5    customer_name: lubkxdgqrxg
```

Завдання 3

Пошук за двома атрибутами з двох таблиць (Drink_category, Drink).

Формування запиту:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py search_records Drink_category Drink 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.drink_id
specify the left end of search interval: 0
specify the right end of search interval: 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: two.price
specify the left end of search interval: 20
specify the right end of search interval: 35
```

Результат:

```
select * from public."Drink_category" as one inner join public."Drink" as two on
one."id"=two."id" where 0<one.drink_id and one.drink_id<4 and 20<two.price and two.price<35
```

```
--- 0.004950284957885742 seconds ---
search result:
2
1
2
2
Espresso
30
------------------------------
5
2
3
5
Cacao
25
------------------------------
1
2
3
1
Amerikano
30
```

Пошук за трьома атрибутами з трьох таблиць (Drink_category, Drink, Category).

Формування запиту:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py search_records Drink_category Drink Category 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.category_id
specify the left end of search interval: 0
specify the right end of search interval: 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: two.price
specify the left end of search interval: 20
specify the right end of search interval: 40
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.id
specify the left end of search interval: 1
specify the right end of search interval: 5
```

Результат:

```
select * from public."Drink_category" as one inner join public."Drink" as two on one."id"=two."id" inner join
public."Category" as three on three."id"=one."id"where 0<one.category_id and one.category_id<3 and 20<two.price and two.price<40 and 1<thre
e.id and three.id<5
--- 0.006752967834472656 seconds ---
search result:
2
1
2
2
Espresso
30
2
Without sugar, cold
_____
3
4
2
3
Hot chocolate
25
3
With_sugar_Hot
```

Пошук за чотирма атрибутами з чотирьох таблиць (Drink_category, Drink, Category, Order).

Формування запиту:

```
PS C:\Users\anna\PycharmProjects\DB2> python main.py search_records Drink_category Drink Category Order 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.drink_id
specify the left end of search interval: 0
specify the right end of search interval: 6
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.price
specify the left end of search interval: 10
specify the right end of search interval: 40
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.id
specify the left end of search interval: 0
specify the right end of search interval: 6
specify the type of data you want to search for (numeric or string): string
specify the name of key by which you`d like to perform search in form: table_number.key_name: four.customer_name
specify the string you`d like to search for: Anna
```

Результат:

```
select * from public."Drink_category" as one inner join public."Drink" as two on one."id"=two."id" inner join
public."Category" as three on three."id"=one."id" inner join public."Order" as four on four."id"=two."id"where 0<one.drink_id and one.drink
_id<6 and 10<two.price and two.price<40 and 0<three.id and three.id<6 and four.customer_name LIKE 'Anna'
--- 0.011001825332641602 seconds ---
search result:
2
1
2
2
Espresso
30
2
Without sugar, cold
2
2
Anna
```

Завдання 4

model.py

```
import psycopg2 as ps


class Model:
    def __init__(self):
        self.conn = None
        try:
            self.conn = ps.connect(
                dbname="coffee_shop",
                user='postgres',
                password="090902",
                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_category(self) -> None:
        return self.get(f"SELECT * FROM public.\"Category\"")

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

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

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

    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_category(self, key_value: int, name: str) -> None:
        self.request(f"UPDATE public.\"Category\" SET name=\'{name}\' WHERE
id={key_value};")

    def update_data_drink(self, key_value: int, name: str, price: int) -> None:
        self.request(f"UPDATE public.\"Drink\" SET name=\'{name}\',
price=\'{price}\' WHERE id={key_value};")

    def update_data_drink_category(self, key_value: int, drink_id: int,
category_id: int) -> None:
        self.request(f"UPDATE public.\"Drink_category\" SET
drink_id=\'{drink_id}\', category_id=\'{category_id}\' "
                     f"WHERE id={key_value};")
```

```
    def update_data_order(self, key_value: int, drink_category_id: int,
customer_name: str) -> None:
        self.request(f"UPDATE public.\"Order\" SET
drink_category_id=\'{drink_category_id}\', "
                     f"customer_name=\'{customer_name}\' WHERE
id={key_value};")

    def insert_data_category(self, key_value: int, name: str) -> None:
        self.request(f"insert into public.\"Category\" (id, name) "
                     f"VALUES ({key_value}, \'{name}\');")

    def insert_data_drink(self, key_value: int, name: str, price: int) -> None:
        self.request(f"insert into public.\"Drink\" (id, name, price) "
                     f"VALUES ({key_value}, \'{name}\', \'{price}\');")

    def insert_data_drink_category(self, key_value: int, drink_id: int,
category_id: int) -> None:
        self.request(f"insert into public.\"Drink_category\" (id, drink_id,
category_id) "
                     f"VALUES ({key_value}, \'{drink_id}\',
\'{category_id}\');")

    def insert_data_order(self, key_value: int, drink_category_id: int,
customer_name: str) -> None:
        self.request(f"insert into public.\"Order\" (id, drink_category_id,
customer_name) "
                     f"VALUES ({key_value}, \'{drink_category_id}\',
\'{customer_name}\');")

    def drink_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Drink\""
                         "select (SELECT MAX(id)+1 FROM public.\"Drink\"), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) \
                         FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), ''), "
                         "FLOOR(RANDOM()*(100000-1)+1);")

    def order_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Order\" select (SELECT MAX(id)+1
FROM public.\"Order\"), "
                         "(SELECT id FROM public.\"Drink_category\" LIMIT 1
OFFSET "
                         "(round(random() *((SELECT COUNT(id) FROM
public.\"Drink_category\")-1)))), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) "
                         "FROM generate_series(1, FLOOR(RANDOM()*(15-5)+5)::
integer)), '');")
```

```
    def category_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Category\""
                         "select (SELECT MAX(id)+1 FROM public.\"Category\"), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) \
                         FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), '');")


    def drink_category_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Drink_category\" "
                         "select (SELECT (MAX(id)+1) FROM
public.\"Drink_category\"), "
                         "(SELECT id FROM public.\"Drink\" LIMIT 1 OFFSET "
                         "(round(random() *((SELECT COUNT(id) FROM
public.\"Drink\")-1)))), "
                         "(SELECT id FROM public.\"Category\" LIMIT 1 OFFSET "
                         "(round(random() *((SELECT COUNT(id) FROM
public.\"Category\")-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_all_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}")
```

В даному модулі реалізуються всі запити. Для цього використовується бібліотека мови Python – psycopg2.

Конструктор класу Model налагоджує зв’язок із сервером і видає повідомлення про помилку, якщо зв’язок не було встановлено.

Методи request, get, get_el здійснюють запити до бази даних за допомогою cursor; якщо виникла помилка і запит не відбувся, вони всі повертають False; якщо вдалося зробити запит: request повертає True, get - усі дані що було взято з запитів SELECT, get_el - тільки перший запис.

Метод count повертає кількість усіх записів таблиці.

Метод find повертає кількість записів таблиці, що відповідають певній умові.

Методи max, min повертають відповідно максимальне і мінімальне значення зазначеного ключа у таблиці.

Методи print_products, print_order, print_catalog, print_shop здійснюють отримання з БД відповідних таблиць.

Метод delete_data реалізує запит на видалення відповідного запису таблиці.

Метод update_data відправляє до БД запит на оновлення даних у певному полі таблиці.

Метод insert_data відправляє до БД запит на вставку запису в таблицю.

Метод data_generator реалізує запит до БД на вставку псевдорандомізованих даних.

Метод search_data_tables реалізує запит на отримання результату пошуку серед number таблиць за рядком пошуку, що генерується в view методами: numeric_search, string_search.