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 III курсу
групи КВ-92
Корж А. А.
Перевірив:
Петрашенко А. В.

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/narcissichka/DataBase

Використана мова програмування: Python 3.9

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 отримання і обробку замовлень з різних інтернет-магазинів.

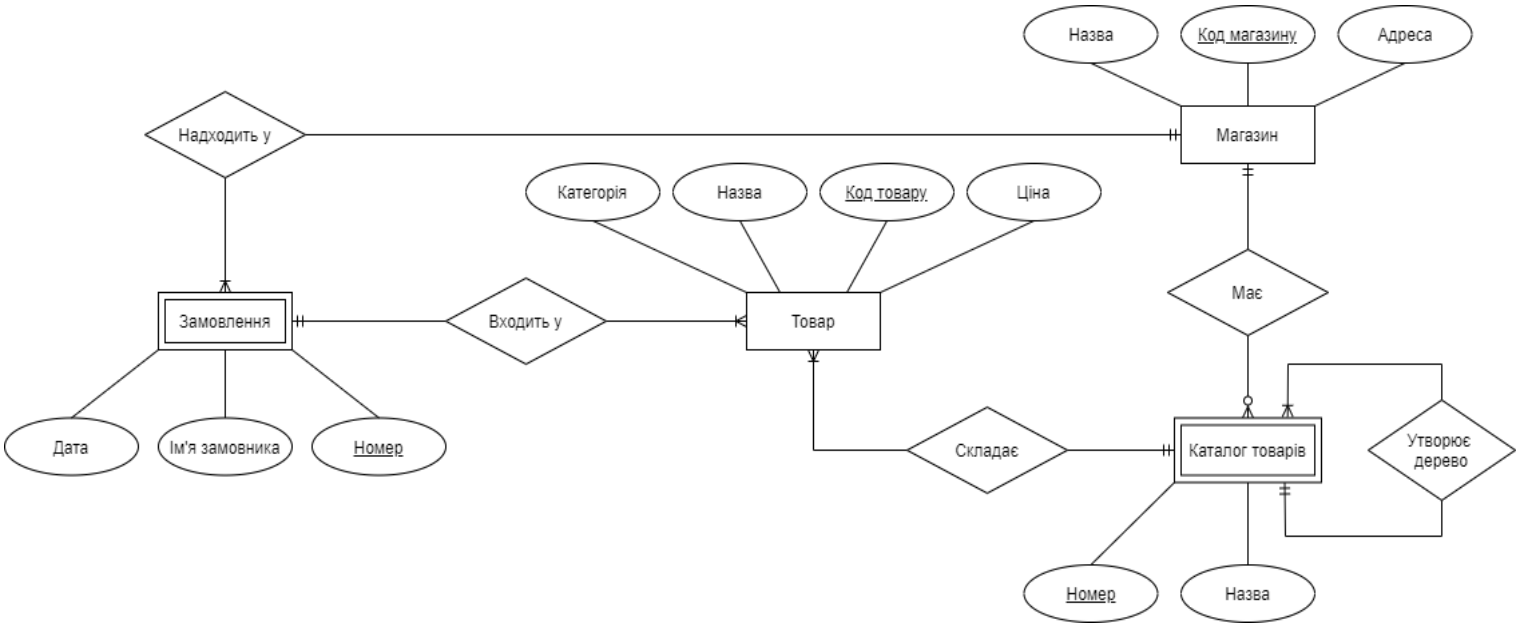



Рисунок 1. ER-діаграма, побудована за нотацією Чена

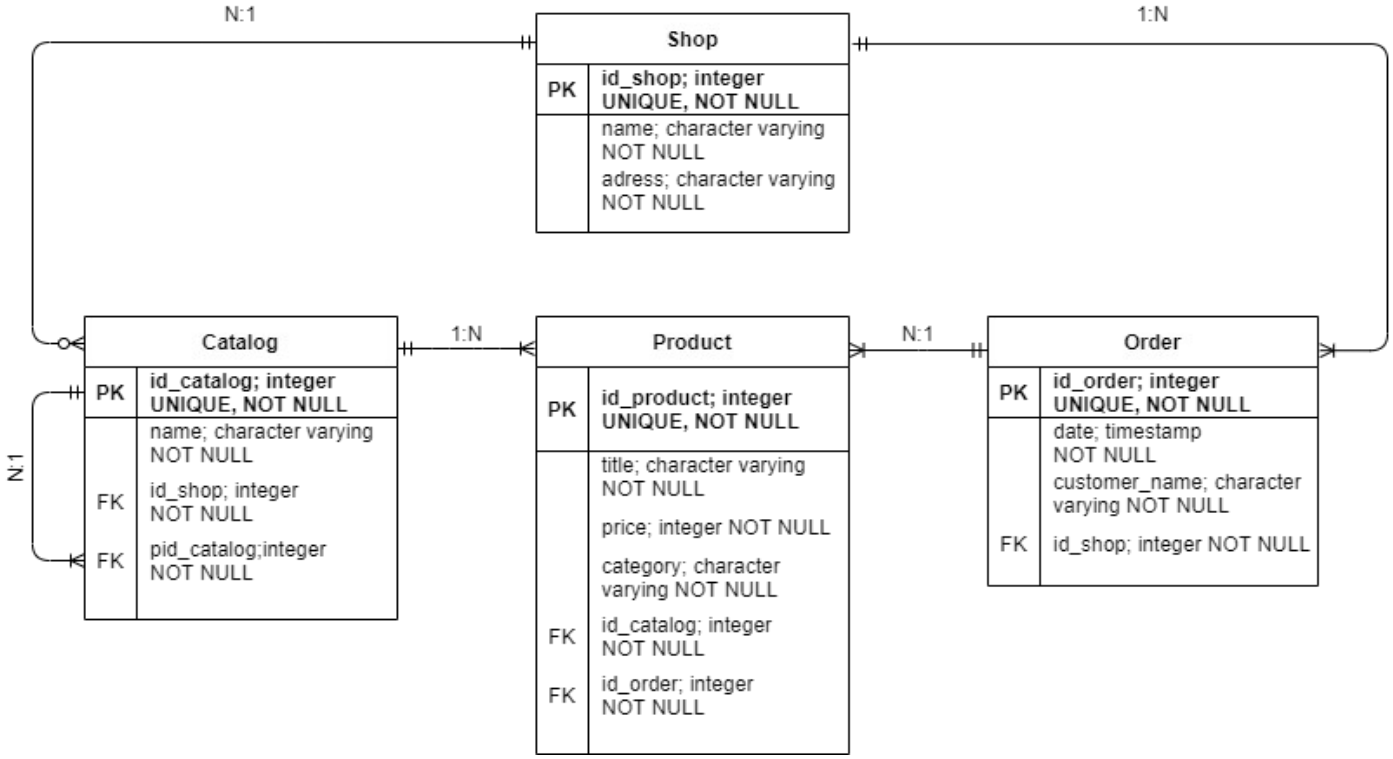



Рисунок 2. Схема бази даних

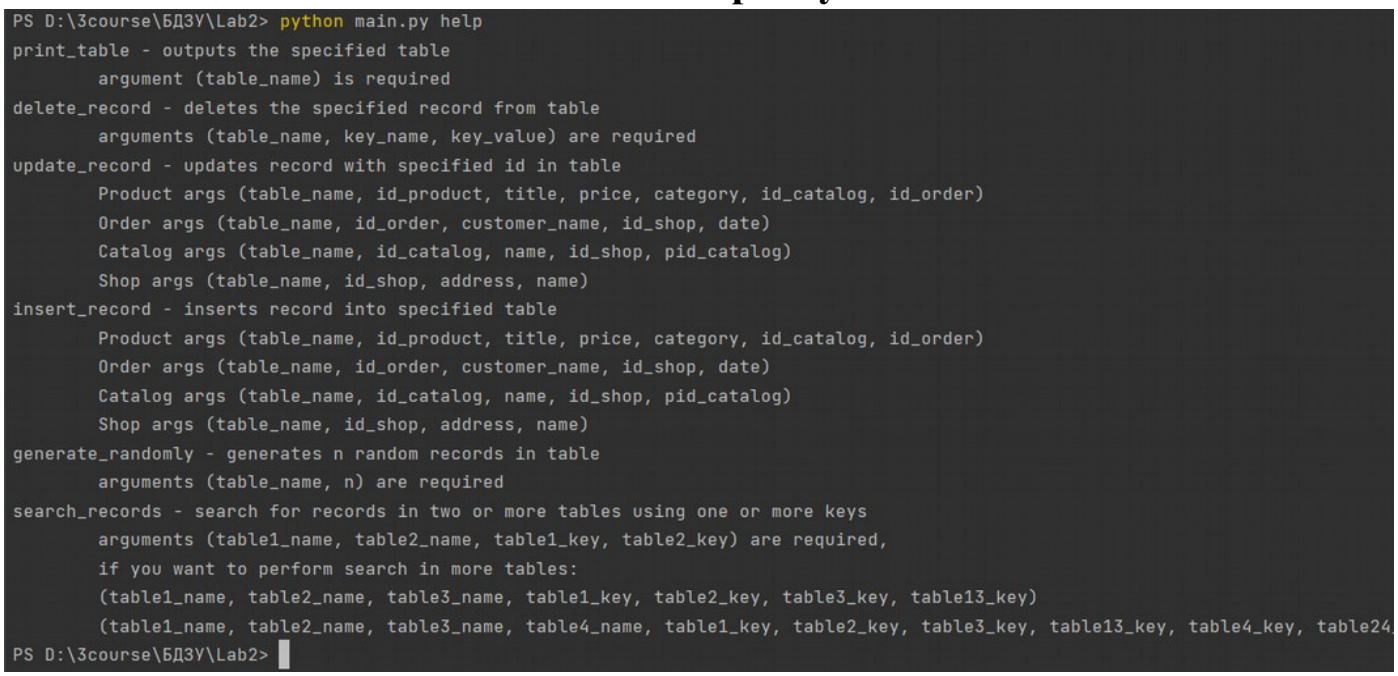
Таблиця 1. Опис структури БД

| Сутність | Атрибут | Тип атрибуту |
|---|---|---|
| **Product** - *містить дані про замовлений товар* | **id_product** - *унікальний ідентифікатор товару*<br>**title** - *назва товару*<br>**price** - *ціна товару*<br>**category** - *категорія, до якої належить товар*<br>**id_catalog** - *ідентифікатор каталогу, в якому знаходиться товар*<br>**id_order** - *ідентифікатор замовлення, до якого додано товар* | **integer** *(числовий)*<br>**character varying** *(рядок)*<br>**integer** *(числовий)*<br>**character varying** *(рядок)*<br>**integer** *(числовий)*<br>**integer** *(числовий)* |
| **Catalog** - *містить у собі всі товари магазину* | **id_catalog** - *унікальний ідентифікатор каталогу*<br>**name** - *назва каталогу*<br>**id_shop** - *ідентифікатор магазину, до якого відноситься даний каталог*<br>**pid_catalog** - *посилання на батьківський каталог* | **integer** *(числовий)*<br>**character varying** *(рядок)*<br>**integer** *(числовий)*<br>**integer** *(числовий)* |
| **Order** - *зберігає дані про замовлення* | **id_order** - *унікальний ідентифікатор замовлення*<br>**customer_name** - *ім'я замовника*<br>**date** - *мітка часу, коли було оформлено замовлення*<br>**id_shop** - *ідентифікатор магазину, в який надійшло замовлення* | **integer** *(числовий)*<br>**character varying** *(рядок)*<br>**timestamp without time zone** *(мітка часу)*<br>**integer** *(числовий)* |
| **Shop** - *містить інформацію про магазин* | **id_shop** - *унікальний ідентифікатор магазину*<br>**name** - *назва магазину*<br>**address** - *фізична адреса магазину* | **integer** *(числовий)*<br>**character varying** *(рядок)*<br>**character varying** *(рядок)* |

У обраній базі даних зберігається інформація про замовлення (таблиця Order), що надійшли до магазинів (таблиця Shop), каталог товарів (таблиця Product) яких розміщено у сутності Catalog. Виходячи з даних, що зберігаються в базі, можна визначити: коли було зроблено замовлення, ким воно було зроблено, які товари до нього входять, до якого саме з магазинів мережі надійшло.

Відповідно, кожен товар відноситься до певного каталогу з дерева каталогів, а також має інші характеристики (див. Таблицю 1). Товар зв’язаний зовнішнім ключем з замовленням та каталогом, каталог та замовлення в свою чергу зв’язані з магазином.

## Схема меню користувача




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: Product, Order, Catalog, Shop

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:

Product id_product(int) title(str) price(float) category(str) id_catalog(int) id_order(int)

Order id_order(int) customer_name(str) id_shop(int) date(datetime)

Catalog id_catalog(int) name(str) id_shop(int) pid_catalog(int)

Shop id_shop(int) address(str) name(str)

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:

Product id_product(int) title(str) price(float) category(str) id_catalog(int) id_order(int)

Order id_order(int) customer_name(str) id_shop(int) date(datetime)

Catalog id_catalog(int) name(str) id_shop(int) pid_catalog(int)

Shop id_shop(int) address(str) name(str)

***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 чотирьох) 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:

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 (обов’язково з вказанням до якої таблиці з перелічених аргументів він відноситься: one.key_name, two.key_name, three.key_name або four.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.

*Важливо: дата з часовою міткою у програмі вказується через крапку у такому форматі: year.month.day.hour.minute.second

# Завдання 1

## *Запит на видалення*

Для перевірки роботи розглянемо запити на видалення дочірньої таблиці Product та батьківської таблиці Order.

Таблиця Product до видалення запису 7:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Product
SELECT * FROM public."Product"
Product table:
id_product: 4   title: Moscino T-Shirt  price: 500      category: tops  id_catalog: 6   id_order: 1202
-------------------------------------
id_product: 5   title: Dolce&Gabbana skirt      price: 300      category: unders        id_catalog: 6   id_order: 1202
-------------------------------------
id_product: 6   title: Louis Vuitton bag        price: 100500   category: bags  id_catalog: 4   id_order: 1202
-------------------------------------
id_product: 1   title: Dior handbag     price: 10000    category: bags  id_catalog: 4   id_order: 1200
-------------------------------------
id_product: 2   title: Hogo Boss trousers       price: 20000    category: unders        id_catalog: 7   id_order: 1201
-------------------------------------
id_product: 3   title: Armani jacket    price: 15000    category: tops  id_catalog: 9   id_order: 1201
-------------------------------------
id_product: 7   title: jacket   price: 3000     category: male outerwear        id_catalog: 7   id_order: 1202
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Таблиця Product після видалення запису 7:

```
PS D:\3course\БД3У\Lab2> python main.py delete_record Product id_product 7
select count(*) from public."Product" where id_product=7
DELETE FROM public."Product" WHERE id_product=7;
PS D:\3course\БД3У\Lab2> python main.py print_table Product
SELECT * FROM public."Product"
Product table:
id_product: 4   title: Moscino T-Shirt  price: 500      category: tops  id_catalog: 6   id_order: 1202
-------------------------------------
id_product: 5   title: Dolce&Gabbana skirt      price: 300      category: unders        id_catalog: 6   id_order: 1202
-------------------------------------
id_product: 6   title: Louis Vuitton bag        price: 100500   category: bags  id_catalog: 4   id_order: 1202
-------------------------------------
id_product: 1   title: Dior handbag     price: 10000    category: bags  id_catalog: 4   id_order: 1200
-------------------------------------
id_product: 2   title: Hogo Boss trousers       price: 20000    category: unders        id_catalog: 7   id_order: 1201
-------------------------------------
id_product: 3   title: Armani jacket    price: 15000    category: tops  id_catalog: 9   id_order: 1201
-------------------------------------
PS D:\3course\БД3У\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 Order до видалення запису 1200:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id_order: 1200  customer_name: Jane Olsen       id_shop: 9657   date: 2021-09-15 21:25:00
-----------------------------------
id_order: 1201  customer_name: Victor Crump     id_shop: 9657   date: 2021-09-15 22:03:30
-----------------------------------
id_order: 1202  customer_name: Elizabeth Taylor         id_shop: 9657   date: 2021-09-16 09:15:24
-----------------------------------
id_order: 1203  customer_name: bsdutip  id_shop: 9657   date: 2019-06-16 16:50:58.639537
-----------------------------------
PS D:\3course\БД3У\Lab2>
```

Спроба видалення запису 1200 з таблиці Order:

```
PS D:\3course\БД3У\Lab2> python main.py delete_record Order id_order 1200
select count(*) from public."Order" where id_order=1200
select count(*) from public."Product" where id_order=1200
this record is connected with Product table, deleting will throw error
PS D:\3course\БД3У\Lab2>
```

### *Запит на вставку поля*

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

Таблиця Product до вставки запису 7:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Product
SELECT * FROM public."Product"
Product table:
id_product: 4   title: Moscino T-Shirt  price: 500      category: tops  id_catalog: 6   id_order: 1202
--------------------------------------
id_product: 5   title: Dolce&Gabbana skirt      price: 300      category: unders        id_catalog: 6   id_order: 1202
--------------------------------------
id_product: 6   title: Louis Vuitton bag        price: 100500   category: bags  id_catalog: 4   id_order: 1202
--------------------------------------
id_product: 1   title: Dior handbag     price: 10000    category: bags  id_catalog: 4   id_order: 1200
--------------------------------------
id_product: 2   title: Hogo Boss trousers       price: 20000    category: unders        id_catalog: 7   id_order: 1201
--------------------------------------
id_product: 3   title: Armani jacket    price: 15000    category: tops  id_catalog: 9   id_order: 1201
--------------------------------------
PS D:\3course\БД3У\Lab2>
```

Таблиця Product після вставки запису 7:

```
PS D:\3course\БД3У\Lab2> python main.py insert_record Product 7 Jeans 456.89 unders 9 1200
select count(*) from public."Product" where id_product=7
select count(*) from public."Catalog" where id_catalog=9
select count(*) from public."Order" where id_order=1200
insert into public."Product" (id_product, title, price, category, id_catalog, id_order) VALUES (7, 'Jeans', 456.89, 'unders', 9, 1200);
PS D:\3course\БД3У\Lab2> python main.py print_table Product
SELECT * FROM public."Product"
Product table:
id_product: 4   title: Moscino T-Shirt  price: 500      category: tops  id_catalog: 6   id_order: 1202
--------------------------------------
id_product: 5   title: Dolce&Gabbana skirt      price: 300      category: unders        id_catalog: 6   id_order: 1202
--------------------------------------
id_product: 6   title: Louis Vuitton bag        price: 100500   category: bags  id_catalog: 4   id_order: 1202
--------------------------------------
id_product: 1   title: Dior handbag     price: 10000    category: bags  id_catalog: 4   id_order: 1200
--------------------------------------
id_product: 2   title: Hogo Boss trousers       price: 20000    category: unders        id_catalog: 7   id_order: 1201
--------------------------------------
id_product: 3   title: Armani jacket    price: 15000    category: tops  id_catalog: 9   id_order: 1201
--------------------------------------
id_product: 7   title: Jeans    price: 457      category: unders        id_catalog: 9   id_order: 1200
--------------------------------------
PS D:\3course\БД3У\Lab2>
```

Записи у батьківській таблиці Catalog:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Catalog
SELECT * FROM public."Catalog"
Catalog table:
id_catalog: 1   name: main      id_shop: 9657   pid_catalog: 1
------------------------------------
id_catalog: 5   name: clothes   id_shop: 9657   pid_catalog: 1
------------------------------------
id_catalog: 6   name: woman     id_shop: 9657   pid_catalog: 5
------------------------------------
id_catalog: 7   name: men       id_shop: 9657   pid_catalog: 5
------------------------------------
id_catalog: 8   name: kids      id_shop: 9657   pid_catalog: 5
------------------------------------
id_catalog: 2   name: accessories       id_shop: 9657   pid_catalog: 1
------------------------------------
id_catalog: 9   name: unisex    id_shop: 9657   pid_catalog: 5
------------------------------------
id_catalog: 3   name: hats      id_shop: 9657   pid_catalog: 2
------------------------------------
id_catalog: 4   name: bags      id_shop: 9657   pid_catalog: 2
------------------------------------
PS D:\3course\БД3У\Lab2>
```

Спроба вставки запису у дочірню таблицю Product з неіснуючим зовнішнім ключем 10:

```
PS D:\3course\БД3У\Lab2> python main.py insert_record Product 8 Toy 456.89 unders 10 1202
select count(*) from public."Product" where id_product=8
select count(*) from public."Catalog" where id_catalog=10
select count(*) from public."Order" where id_order=1202
Something went wrong (record with such id exists or inappropriate foreign key values)
PS D:\3course\БД3У\Lab2>
```

### *Запит на зміну полів*

Для перевірки роботи розглянемо запити на зміну значеннння в дочірній таблиці Order. Спочатку коректний, потім з неіснуючим значенням зовнішнього ключа батьківської таблиці Shop.

Таблиця Order до зміни запису 1203:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id_order: 1200  customer_name: Jane Olsen       id_shop: 9657   date: 2021-09-15 21:25:00
-------------------------------------
id_order: 1201  customer_name: Victor Crump     id_shop: 9657   date: 2021-09-15 22:03:30
-------------------------------------
id_order: 1202  customer_name: Elizabeth Taylor         id_shop: 9657   date: 2021-09-16 09:15:24
-------------------------------------
id_order: 1203  customer_name: bsdutip  id_shop: 9657   date: 2019-06-16 16:50:58.639537
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Таблиця Order після зміни запису 1203:

```
PS D:\3course\БД3У\Lab2> python main.py update_record Order 1203 Anna 9658 2020.12.31.00.00.01
select count(*) from public."Shop" where id_shop=9658
select count(*) from public."Order" where id_order=1203
UPDATE public."Order" SET customer_name='Anna', id_shop=9658, date='2020-12-31 00:00:01' WHERE id_order=1203;
PS D:\3course\БД3У\Lab2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id_order: 1200  customer_name: Jane Olsen       id_shop: 9657   date: 2021-09-15 21:25:00
-------------------------------------
id_order: 1201  customer_name: Victor Crump     id_shop: 9657   date: 2021-09-15 22:03:30
-------------------------------------
id_order: 1202  customer_name: Elizabeth Taylor         id_shop: 9657   date: 2021-09-16 09:15:24
-------------------------------------
id_order: 1203  customer_name: Anna     id_shop: 9658   date: 2020-12-31 00:00:01
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Записи у батьківській таблиці Shop:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Shop
SELECT * FROM public."Shop"
Shop table:
id_shop: 9657   address: Barlby Road, Selby, YO85BL, UK         name: Asos
-------------------------------------
id_shop: 9658   address: lmlvptugpgowegryfcgvnv         name: zgdcqes
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

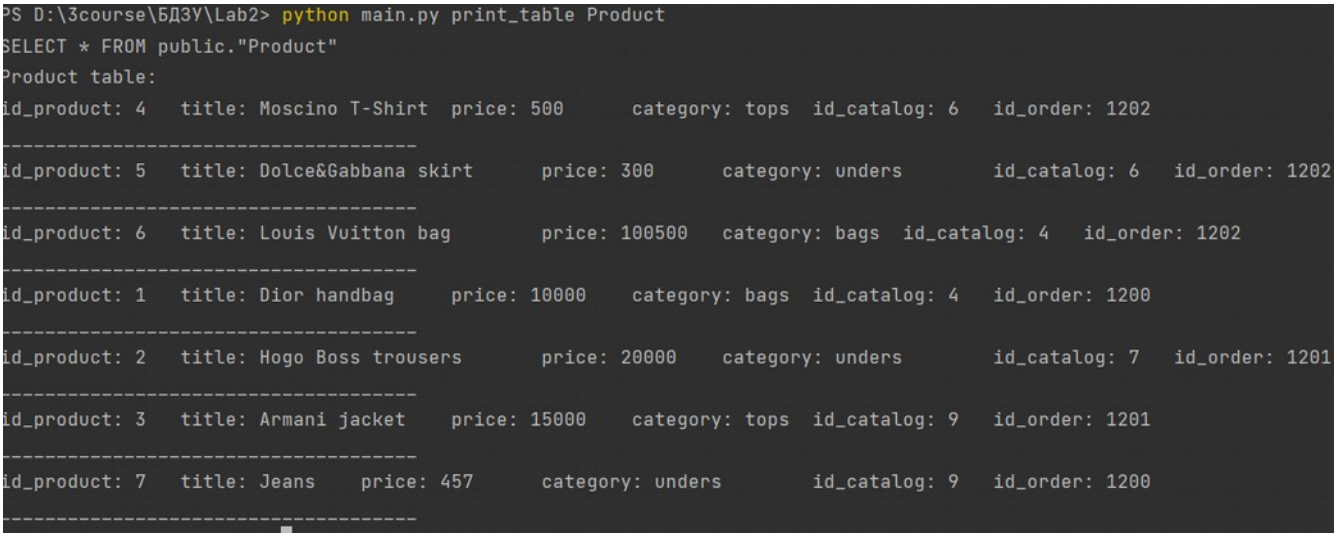
Спроба зміни запису у дочірній таблиці Order з неіснуючим зовнішнім ключем 9:

```
PS D:\3course\БД3У\Lab2> python main.py update_record Order 1203 Anna 9 2020.12.31.00.00.01
select count(*) from public."Shop" where id_shop=9
select count(*) from public."Order" where id_order=1203
Something went wrong (record with such id does not exist or inappropriate foreign key value)
PS D:\3course\БД3У\Lab2>
```

# Завдання 2

Вставка 5 псевдорандомізованих записів у кожну з таблиць.

Початкова таблиця Product:




Запит (скопійовано з терміналу, тому що скріншот нечитабельний):

```
PS D:\3course\БДЗУ\Lab2> python main.py generate_randomly Product 5
insert into public."Product"select (SELECT MAX(id_product)+1 FROM public."Product"),
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),array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)
      FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), ''), (SELECT
id_catalog FROM public."Catalog" LIMIT 1 OFFSET (round(random() * ((SELECT
COUNT(id_catalog) FROM public."Cat
alog")-1)))),(SELECT id_order FROM public."Order" LIMIT 1 OFFSET (round(random() *
((SELECT COUNT(id_order) FROM public."Order")-1))));
```

Модифікована таблиця Product:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Product
SELECT * FROM public."Product"
Product table:
id_product: 4   title: Moscino T-Shirt  price: 500      category: tops  id_catalog: 6   id_order: 1202
-------------------------------------
id_product: 5   title: Dolce&Gabbana skirt      price: 300      category: unders        id_catalog: 6   id_order: 1202
-------------------------------------
id_product: 6   title: Louis Vuitton bag        price: 100500   category: bags  id_catalog: 4   id_order: 1202
-------------------------------------
id_product: 1   title: Dior handbag     price: 10000    category: bags  id_catalog: 4   id_order: 1200
-------------------------------------
id_product: 2   title: Hogo Boss trousers       price: 20000    category: unders        id_catalog: 7   id_order: 1201
-------------------------------------
id_product: 3   title: Armani jacket    price: 15000    category: tops  id_catalog: 9   id_order: 1201
-------------------------------------
id_product: 7   title: Jeans    price: 457      category: unders        id_catalog: 9   id_order: 1200
-------------------------------------
id_product: 8   title: lsjxcu   price: 32381    category: gqgnqluho     id_catalog: 5   id_order: 1201
-------------------------------------
id_product: 9   title: cvcz     price: 2347     category: fqbuy         id_catalog: 8   id_order: 1202
-------------------------------------
id_product: 10  title: gdui     price: 20645    category: qouphh        id_catalog: 6   id_order: 1203
-------------------------------------
id_product: 11  title: hyfeoomgv        price: 54845    category: xejzb         id_catalog: 3   id_order: 1202
-------------------------------------
id_product: 12  title: phsfdgfg         price: 3884     category: lzejsie       id_catalog: 7   id_order: 1201
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Початкова таблиця Order:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id_order: 1200  customer_name: Jane Olsen       id_shop: 9657   date: 2021-09-15 21:25:00
-------------------------------------
id_order: 1201  customer_name: Victor Crump     id_shop: 9657   date: 2021-09-15 22:03:30
-------------------------------------
id_order: 1202  customer_name: Elizabeth Taylor         id_shop: 9657   date: 2021-09-16 09:15:24
-------------------------------------
id_order: 1203  customer_name: Anna     id_shop: 9658   date: 2020-12-31 00:00:01
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Запит:

```
PS D:\3course\БД3У\Lab2> python main.py generate_randomly Order 5
insert into public."Order" select (SELECT (MAX(id_order)+1) FROM public."Order"),
array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM
generate_series(1, FLOOR(RA
NDOM()*(10-3)+3):: integer)), ''), (SELECT id_shop FROM public."Shop" LIMIT 1 OFFSET
(round(random()  *((SELECT  COUNT(id_shop)  FROM  public."Shop")-1)))),  (SELECT
to_timestamp(1549634400+ra
ndom()*70071999));
```

Модифікована таблиця Order:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Order
SELECT * FROM public."Order"
Order table:
id_order: 1200  customer_name: Jane Olsen       id_shop: 9657   date: 2021-09-15 21:25:00
-------------------------------------
id_order: 1201  customer_name: Victor Crump     id_shop: 9657   date: 2021-09-15 22:03:30
-------------------------------------
id_order: 1202  customer_name: Elizabeth Taylor         id_shop: 9657   date: 2021-09-16 09:15:24
-------------------------------------
id_order: 1203  customer_name: Anna     id_shop: 9658   date: 2020-12-31 00:00:01
-------------------------------------
id_order: 1204  customer_name: pfuufcr  id_shop: 9658   date: 2020-01-24 05:13:07.997146
-------------------------------------
id_order: 1205  customer_name: ktvkuceq         id_shop: 9658   date: 2019-03-10 00:33:02.442442
-------------------------------------
id_order: 1206  customer_name: hmbvy    id_shop: 9657   date: 2020-06-29 12:28:35.424963
-------------------------------------
id_order: 1207  customer_name: ytrfx    id_shop: 9657   date: 2020-08-21 15:59:27.449535
-------------------------------------
id_order: 1208  customer_name: hrxlzbp  id_shop: 9658   date: 2019-04-18 02:37:39.638047
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Початкова таблиця Catalog:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Catalog
SELECT * FROM public."Catalog"
Catalog table:
id_catalog: 1   name: main      id_shop: 9657   pid_catalog: 1
-------------------------------------
id_catalog: 5   name: clothes   id_shop: 9657   pid_catalog: 1
-------------------------------------
id_catalog: 6   name: woman     id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 7   name: men       id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 8   name: kids      id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 2   name: accessories       id_shop: 9657   pid_catalog: 1
-------------------------------------
id_catalog: 9   name: unisex    id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 3   name: hats      id_shop: 9657   pid_catalog: 2
-------------------------------------
id_catalog: 4   name: bags      id_shop: 9657   pid_catalog: 2
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Запит:

```
PS D:\3course\БДЗУ\Lab2> python main.py generate_randomly Catalog 5
insert into public."Catalog" select (SELECT MAX(id_catalog)+1 FROM public."Catalog"),
array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM
generate_series(1, FLOO
R(RANDOM()*(15-5)+5):: integer)), ''), (SELECT id_shop FROM public."Shop" LIMIT 1
OFFSET (round(random() *((SELECT COUNT(id_shop) FROM public."Shop")-1)))), (SELECT
id_catalog FROM public
."Catalog" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id_catalog) FROM
public."Catalog")-1))));
```

Модифікована таблиця Catalog:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Catalog
SELECT * FROM public."Catalog"
Catalog table:
id_catalog: 1   name: main      id_shop: 9657   pid_catalog: 1
-------------------------------------
id_catalog: 5   name: clothes   id_shop: 9657   pid_catalog: 1
-------------------------------------
id_catalog: 6   name: woman     id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 7   name: men       id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 8   name: kids      id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 2   name: accessories       id_shop: 9657   pid_catalog: 1
-------------------------------------
id_catalog: 9   name: unisex    id_shop: 9657   pid_catalog: 5
-------------------------------------
id_catalog: 3   name: hats      id_shop: 9657   pid_catalog: 2
-------------------------------------
id_catalog: 4   name: bags      id_shop: 9657   pid_catalog: 2
-------------------------------------
id_catalog: 10  name: ubkvph    id_shop: 9658   pid_catalog: 7
-------------------------------------
id_catalog: 11  name: pfyldq    id_shop: 9657   pid_catalog: 7
-------------------------------------
id_catalog: 12  name: lvrusuuxqh        id_shop: 9658   pid_catalog: 4
-------------------------------------
id_catalog: 13  name: bevco     id_shop: 9658   pid_catalog: 12
-------------------------------------
id_catalog: 14  name: sdlnx     id_shop: 9657   pid_catalog: 13
```

Початкова таблиця Shop:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Shop
SELECT * FROM public."Shop"
Shop table:
id_shop: 9657   address: Barlby Road, Selby, YO85BL, UK         name: Asos
-------------------------------------
id_shop: 9658   address: lmlvptugpgowegryfcgvnv         name: zgdcqes
-------------------------------------
PS D:\3course\БД3У\Lab2>
```

Запит:

```
PS D:\3course\БД3У\Lab2> python main.py generate_randomly Shop 5
insert into public."Shop" select (SELECT MAX(id_shop)+1 FROM public."Shop"),
array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM
generate_series(1, FLOOR(RANDOM(
)*(25-10)+10):: integer)), ''), array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
```

Модифікована таблиця Shop:

```
PS D:\3course\БД3У\Lab2> python main.py print_table Shop
SELECT * FROM public."Shop"
Shop table:
id_shop: 9657   address: Barlby Road, Selby, YO85BL, UK         name: Asos
-------------------------------------
id_shop: 9658   address: lmlvptugpgowegryfcgvnv         name: zgdcqes
-------------------------------------
id_shop: 9659   address: lptqjkhoyhwxugsgxinj   name: vxhsrnc
-------------------------------------
id_shop: 9660   address: rknxevxsjcpm   name: xgmee
-------------------------------------
id_shop: 9661   address: rithcbqhyulkmew        name: mwwl
-------------------------------------
id_shop: 9662   address: snwtwuetczvsjjvgmfqckohh       name: uxxccxcb
-------------------------------------
id_shop: 9663   address: lgldtjclbfeduqwleuizg  name: skmtedb
-------------------------------------
```

## Завдання 3

Пошук за трьома атрибутами з двох таблиць (Product, Catalog).

Формування запиту:

```
PS D:\3course\БД3У\Lab2> python main.py search_records Product Catalog id_catalog id_catalog
specify the number of attributes you`d like to search by: 3
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.id_product
specify the left end of search interval: 0
specify the right end of search interval: 10
specify the type of data you want to search for (numeric, string or date): string
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.name
specify the string you`d like to search for: woman
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.id_shop
specify the left end of search interval: 9656
specify the right end of search interval: 9658
```

Запит:

```
select * from public."Product" as one inner join public."Catalog" as two on one."id_catalog"=two."id_catalog" where 0<one.id_product and one.id_product<10 and two.name LIKE 'woman' and 9656<two.id_shop and two.id_shop<9658
```

Результат:

```
search result:
4
Moscino T-Shirt
500
tops
6
1202
6
woman
9657
5
-------------------------------------
5
Dolce&Gabbana skirt
300
unders
6
1202
6
woman
9657
5
-------------------------------------
--- 0.015571355819702148 seconds ---
PS D:\3course\БД3У\Lab2>
```

Пошук за трьома атрибутами з трьох таблиць (Product, Catalog, Order)
Формування запиту:

```
PS D:\3course\БД3У\Lab2> python main.py search_records Product Catalog Order id_catalog id_catalog id_order id_order
specify the number of attributes you`d like to search by: 3
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.price
specify the left end of search interval: 400
specify the right end of search interval: 20000
specify the type of data you want to search for (numeric, string or date): string
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.name
specify the string you`d like to search for: unisex
specify the type of data you want to search for (numeric, string or date): date
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.date
specify the left end of search interval in form: year.month.day.hour.minute.second: 2021.08.31.00.09.23
specify the right end of search interval in form: year.month.day.hour.minute.second: 2021.10.1.00.00.00
```

Запит:

```
select * from public."Product" as one inner join public."Catalog" as two on one."id_catalog"=two."id_catalog" inner join public."Order" as three on three."id_order"=one."id_order"where 400<one.price and one.price<20000 and two.name LIKE 'unisex' and three.date BETWEEN '2021-08-31 00:09:23' AND '2021-10-01 00:00:00'
```

Результат:

```
search result:
3
Armani jacket
15000
tops
9
1201
9
unisex
9657
5
1201
Victor Crump
9657
2021-09-15 22:03:30
--------------------------------
7
Jeans
457
unders
9
1200
9
unisex
9657
5
1200
Jane Olsen
9657
2021-09-15 21:25:00
```

Пошук за чотирма атрибутами з чотирьох таблиць (Product, Catalog, Order, Shop)
Формування запиту:

```
PS D:\3course\БД3У\Lab2> python main.py search_records Product Catalog Order Shop  id_catalog id_catalog id_order id_order id_shop id_shop
specify the number of attributes you`d like to search by: 4
specify the type of data you want to search for (numeric, string or date): date
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.date
specify the left end of search interval in form: year.month.day.hour.minute.second: 2021.09.01.04.50.55
specify the right end of search interval in form: year.month.day.hour.minute.second: 2021.10.15.06.07.08
specify the type of data you want to search for (numeric, string or date): string
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.name
specify the string you`d like to search for: bags
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: four.id_shop
specify the left end of search interval: 9656
specify the right end of search interval: 9659
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.price
specify the left end of search interval: 100
specify the right end of search interval: 1000000
```

Запит:

```
select * from public."Product" as one inner join public."Catalog" as two on one."id_catalog"=two."id_catalog" inner join public."Order" as three on three."id_order"=one."id_order" inner join public."Shop" as four on four."id_shop"=two."id_shop"where three.date BETWEEN '2021-09-01 04:50:55' AND '2021-10-15 06:07:08' and two.name LIKE 'bags' and 9656<four.id_shop and four.id_shop<9659 and 100<one.price and one.price<1000000
```

Результат:

```
search result:
6
Louis Vuitton bag
100500
bags
4
1202
4
bags
9657
2
1202
Elizabeth Taylor
9657
2021-09-16 09:15:24
9657
Barlby Road, Selby, YO85BL, UK
Asos
-------------------------------------
```

```
-----------------------------------
1
Dior handbag
10000
bags
4
1200
4
bags
9657
2
1200
Jane Olsen
9657
2021-09-15 21:25:00
9657
Barlby Road, Selby, YO85BL, UK
Asos
-----------------------------------
--- 0.017673730850219727 seconds ---
PS D:\3course\БД3У\Lab2>
```

# Завдання 4

Код програмного модулю “model.py”

```
import datetime
import psycopg2 as ps


class Model:
    def __init__(self):
        self.conn = None
        try:
            self.conn = ps.connect(
                database="postgres",
                user='postgres',
                password="********",
                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_products(self) -> None:
        table = self.get(f"SELECT * FROM public.\"Product\"")
        print('Product table:')
        for row in table:
            print('id_product:', row[0], '\ttitle:', row[1], '\tprice:', row[2], '\
tcategory:', row[3],
                  '\tid_catalog:', row[4], '\tid_order:', row[5])
            print('_____________________________________')

    def print_order(self) -> None:
        table = self.get(f"SELECT * FROM public.\"Order\"")
        print('Order table:')
        for row in table:
            print('id_order:', row[0], '\tcustomer_name:', row[1], '\tid_shop:',
row[2], '\tdate:', row[3])
            print('_____________________________________')

    def print_catalog(self) -> None:
        table = self.get(f"SELECT * FROM public.\"Catalog\"")
        print('Catalog table:')
        for row in table:
            print('id_catalog:', row[0], '\tname:', row[1], '\tid_shop:', row[2], '\
tpid_catalog:', row[3])
            print('_____________________________________')

    def print_shop(self) -> None:
        table = self.get(f"SELECT * FROM public.\"Shop\"")
        print('Shop table:')
        for row in table:
            print('id_shop:', row[0], '\taddress:', row[1], '\tname:', row[2])
            print('_____________________________________')

    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_product(self, key_value: int, title: str, price: float, category:
str,
                            id_catalog: int, id_order: int) -> None:
```

```
        self.request(f"UPDATE public.\"Product\" SET title=\'{title}\', price={price},
category=\'{category}\', "
                     f"id_catalog={id_catalog}, id_order={id_order} WHERE
id_product={key_value};")

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

    def update_data_catalog(self, key_value: int, name: str, id_shop: int,
pid_catalog: int) -> None:
        self.request(f"UPDATE public.\"Catalog\" SET name=\'{name}\',
id_shop={id_shop}, "
                     f"pid_catalog={pid_catalog} WHERE id_order={key_value};")

    def update_data_shop(self, key_value: int, address: str, name: str) -> None:
        self.request(f"UPDATE public.\"Shop\" SET address=\'{address}\', "
                     f"name=\'{name}\' WHERE id_shop={key_value};")

    def insert_data_product(self, id_product: int, title: str, price: float, category:
str, id_catalog: int,
                            id_order: int) -> None:
        self.request(f"insert into public.\"Product\" (id_product, title, price,
category, id_catalog, id_order) "
                     f"VALUES ({id_product}, \'{title}\', {price}, \'{category}\',
{id_catalog}, {id_order});")

    def insert_data_order(self, id_order: int, customer_name: str, id_shop: int, date:
datetime.datetime) -> None:
        self.request(f"insert into public.\"Order\" (id_order, customer_name, id_shop,
date) "
                     f"VALUES ({id_order}, \'{customer_name}\', {id_shop},
\'{date}\');")

    def insert_data_catalog(self, id_catalog: int, name: str, id_shop: int,
pid_catalog: int) -> None:
        self.request(f"insert into public.\"Catalog\" (id_catalog, name, id_shop,
pid_catalog) "
                     f"VALUES ({id_catalog}, \'{name}\', {id_shop}, {pid_catalog});")

    def insert_data_shop(self, id_shop: int, address: str, name: str) -> None:
        self.request(f"insert into public.\"Shop\" (id_shop, address, name) "
                     f"VALUES ({id_shop}, \'{address}\', \'{name}\');")

    def product_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Product\""
                         "select (SELECT MAX(id_product)+1 FROM public.\"Product\"), "
                         "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),"
```

```
                          "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) \
                          FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)),
''), "
                          "(SELECT id_catalog FROM public.\"Catalog\" LIMIT 1 OFFSET
(round(random() * "
                          "((SELECT COUNT(id_catalog) FROM public.\"Catalog\")-1)))),"
                          "(SELECT id_order FROM public.\"Order\" LIMIT 1 OFFSET "
                          "(round(random() * ((SELECT COUNT(id_order) FROM
public.\"Order\")-1))));")

    def order_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Order\" select (SELECT (MAX(id_order)
+1) FROM public.\"Order\"), "
                          "array_to_string(ARRAY(SELECT chr((97 + round(random() * 25))
:: integer) "
                          "FROM generate_series(1, FLOOR(RANDOM()*(10-3)+3)::
integer)), ''), "
                          "(SELECT id_shop FROM public.\"Shop\" LIMIT 1 OFFSET "
                          "(round(random() *((SELECT COUNT(id_shop) FROM
public.\"Shop\")-1)))), "
                          "(SELECT to_timestamp(1549634400+random()*70071999));")

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

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

    def search_data_two_tables(self, table1_name: str, table2_name: str, table1_key,
table2_key,
                               search: str):
```

```
        found = 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}")
        print('search result:')
        for row in found:
            for i in range(0, len(row)):
                print(row[i])
            print('_____________________________________')

    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):
        found = 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}")
        print('search result:')
        for row in found:
            for i in range(0, len(row)):
                print(row[i])
            print('_____________________________________')

    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):
        found = 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}")
        print('search result:')
        for row in found:
            for i in range(0, len(row)):
                print(row[i])
            print('_____________________________________')

    def numeric_search(self, a: int, b: int, key: str):
        return f"{a}<{key} and {key}<{b}"

    def string_search(self, string: str, key: str):
        return f"{key} LIKE \'{string}\'"

    def date_search(self, datetime1: datetime.datetime, datetime2: datetime.datetime,
key: str):
        return f"{key} BETWEEN \'{datetime1}\' AND \'{datetime2}\'"
```

Даний модуль є точкою доступу до бази даних з програми. Саме в ньому реалізуються всі запити. Для цього в ньому 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 вони всі повертають False якщо виникла помилка і запит не відбувся, а якщо вдалося зробити запит, request повертає True, get повертає усі дані що було взято з запитів SELECT (масив кортежів з записами таблиць), get_el повертає тільки перший запис.

Метод count повертає кількість усіх записів таблиці.

Метод find повертає кількість записів таблиці, що відповідають певній умові (або False, якщо записів не знайдено).

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 та виведення на екран відповідних таблиць.

Метод delete_data реалізує запит на видалення відповідного запису таблиці.

Методи update_data_(table name) відправляють до БД запит на оновлення даних у певному полі таблиці.

Методи insert_data_(table name) відправляють до БД запит на вставку запису в таблицю.

Методи (table name)_data_generator реалізують запит до БД на вставку псевдорандомізованих даних (для первинного ключа максимальне вже існуюче значення +1, для зовнішніх ключів — одне із можливих значень для цього ключа).

Методи search_data_(number)_tables реалізують запит на отримання результату пошуку серед number таблиць за рядком, що генерується в контроллері методами: numeric_search, string_search, date_search.

Методи numeric_search, string_search, date_search приймають на вхід параметри пошуку (діапазони, рядковий шаблон, ключі) та повертають рядок пошуку за відповідним типом атрибуту.