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»**

**Тема****: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»**

Виконав: студент IIІ курсу

ФПМ групи КВ-93

Рудницький Павло

Київ – 2021

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

Постановка задачі

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 GitHub: <https://github.com/Pasha-freelance/DataBase-Lab2>

Мова програмування: Python 3.9

Використані бібліотеки: psycopg2 (для встановлення зв’язку з СУБД), time (для виміру часу пошуку), sys (для реалізації інтерфейсу)

**Модель «сутність-зв’язок» мережі магазинів**

Обрана предметна галузь передбачає зберігання товарів в кожному окремому філіалу мережі магазинів. Згідно цієї області для побудови бази даних було виділено наступні сутності:

1. Сутність “Shop” містить такі атрибути як ID, адреса магазину та контакти управляючого філіалом. Слугує для зберігання інформації про філіал
2. Сутність “Discount” містить такі атрибути як ID, відсоток знижки та термін її дії. Слугує зберігання інформації про знижку.
3. Сутність “Product\_discount” містить такі атрибути як ID, ідентифікатор продукту та ідентифікатор знижки. Слугує для зв’язування знижки та продукту.
4. Сутність “Product” містить такі атрибути як ID, ідентифікатор відділу, до якого належить продукт та назву продукту. Слугує для зберігання інформації про конкретний продукт.

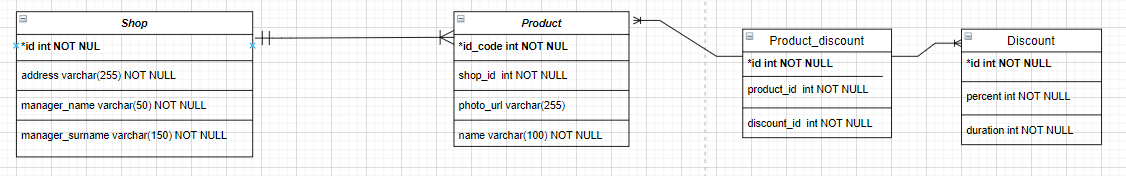
**

Рисунок 1 - Логічна модель предметної області «Мережа магазинів

***Опис структури БД «Мережа магазинів»***

|  |  |  |
| --- | --- | --- |
| Відношення | Атрибут | Розмір (тип) |
| ***Відношення “Shop”***  Вміщує інформацію про філіал магазину | **\*id** – унікальний ID філіалу магазина в БД, не допускає NULL  **address** – адреса магазину, не допускає NULL  **manager\_name** – ім’я управляючого магазином, не допускає NULL  **manager\_surname** – прізвище управляючого магазином, не допускає NULL | Числовий  Текстовий(255)  Текстовий(50)  Текстовий(150) |
| ***Відношення “Discount”***  Вміщує інформацію про знижку | **\*id** – унікальний ID знижки в БД, не допускає NULL  **percent –** відсоток знижки, не допускає NULL  **duration –** термін дії знижки, не допускає NULL | Числовий  Числовий  Числовий |
| ***Відношення “Product\_discount”***  Вміщує інформацію про продукт та знижку | **\*id** – унікальний ID таблиці в БД, не допускає NULL  **product\_id –** ідентифікатор продукту, не допускає NULL  **discount\_id –** ідентифікатор знижки, не допускає NULL | Числовий  Числовий  Числовий |
| ***Відношення “Product”***  Вміщує інформацію про конкретний продукт | **\*id** – унікальний ID продукту в БД, не допускає NULL  **photo\_url** – посилання на фото продукту  **name** – назва продукту, не допускає NULL | Числовий  Текстовий(255)  Текстовий(100) |

Меню користувача

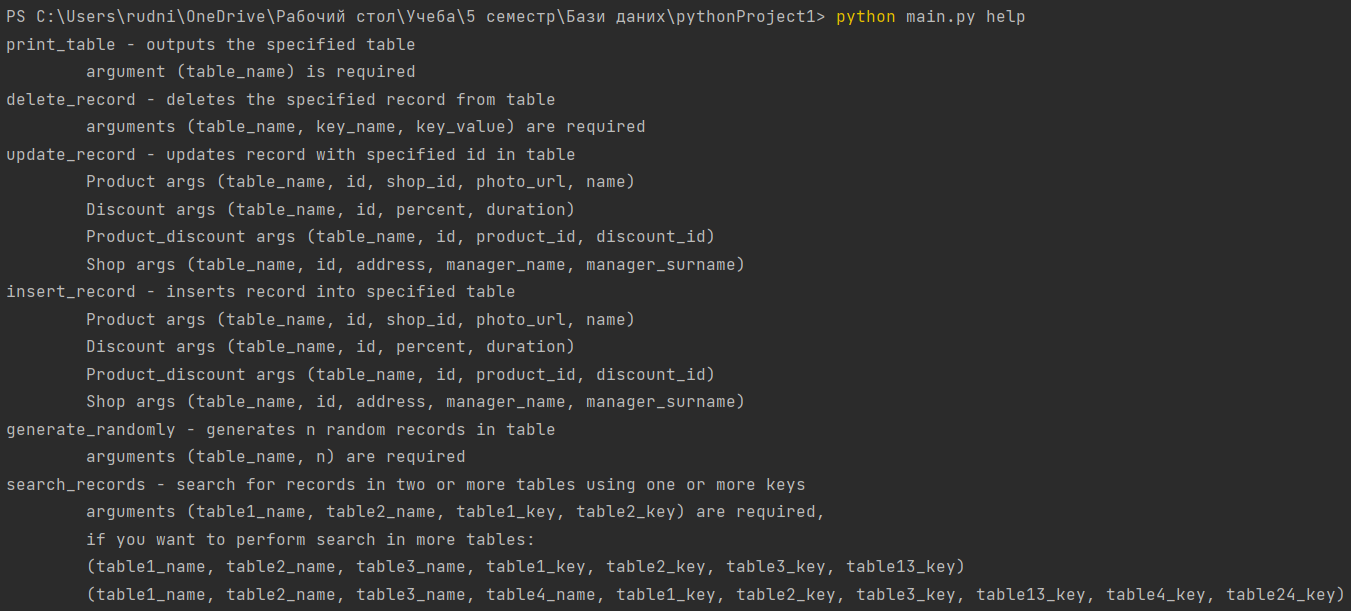


Рисунок 2 – меню користувача додатку БД «Магазин»

На рисунку 2 зображено виконання команди “menu”, яка показує всі доступні для виконання команди та аргументи, які вони приймають.

При виклику команда запускає відповідний метод у файлі “Controller.py”, який в свою чергу передає значення у файл “View”. Після перевірки на правильність внесення даних користувачем, запит передається у файл “Model.py”, де здійснюється запит до бази даних.

**Завдання 1**

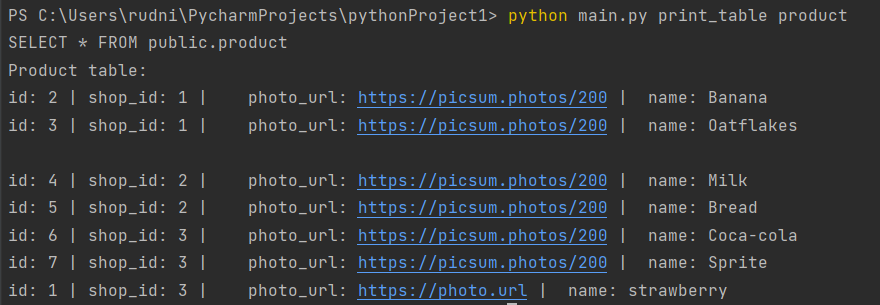
Методи реалізації:

1. Print\_table – виводить вміст таблиці у вікно терміналу. За аргумент приймає назву таблиці.
2. Delete\_record – видаляє запис з таблиці. За аргументи приймає назву таблиці, первинний ключ та його значення.
3. Update\_record – змінює усі поля запису (окрім первинного ключа). Аргументи різні для для кожної таблиці:
4. для таблиці “Product”: id, photo\_url, name;
5. для таблиці “ Shop”: id, address, manager\_name, manager\_surname;
6. для таблиці “Discount”: id, percent,duration;
7. для таблиці “Product\_discount”: id, product\_id, discount\_id.
8. Insert\_record – вствавляє новий рядок у обрану таблицю. Аргументи різні для для кожної таблиці:
9. для таблиці “Product”: id, photo\_url, name;
10. для таблиці “ Shop”: id, address, manager\_name, manager\_surname
11. для таблиці “Discount”: id, percent,duration;
12. для таблиці “Product\_discount”: id, product\_id, discount\_id.

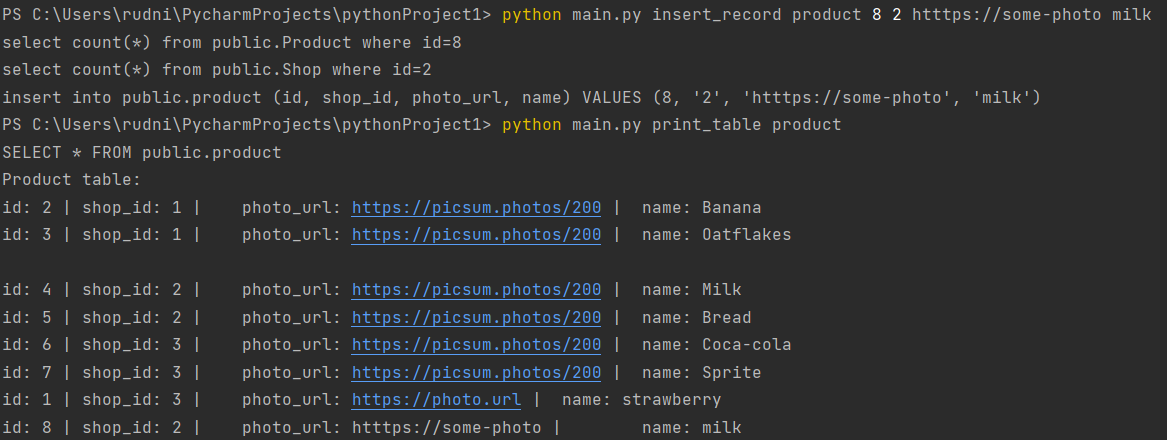
Запит на внесення даних

Внесення запису до таблиці “Product”

Таблиця до вставки поля:

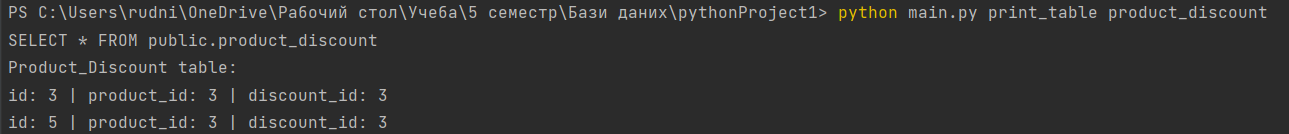


Таблиця після вставки поля з id 8:

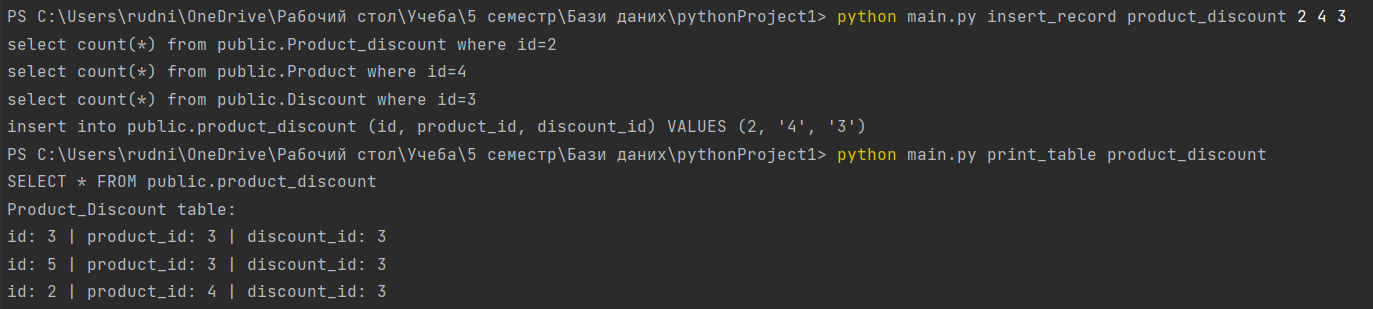


Внесення запису до таблиці “Product\_discount”

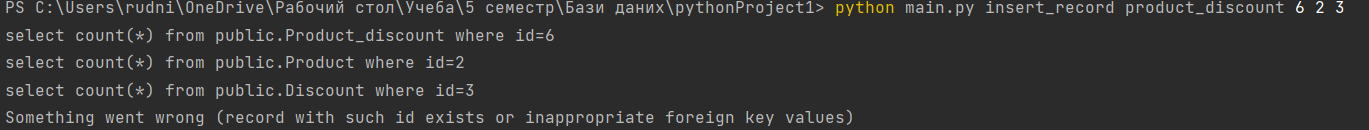
Таблиця до вставки поля:



Таблиця після вставки поля з id 2:



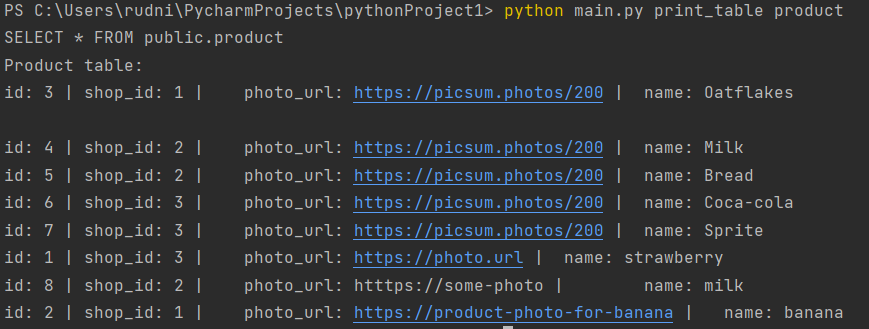
Спроба вставки запису з неіснуючими даними:



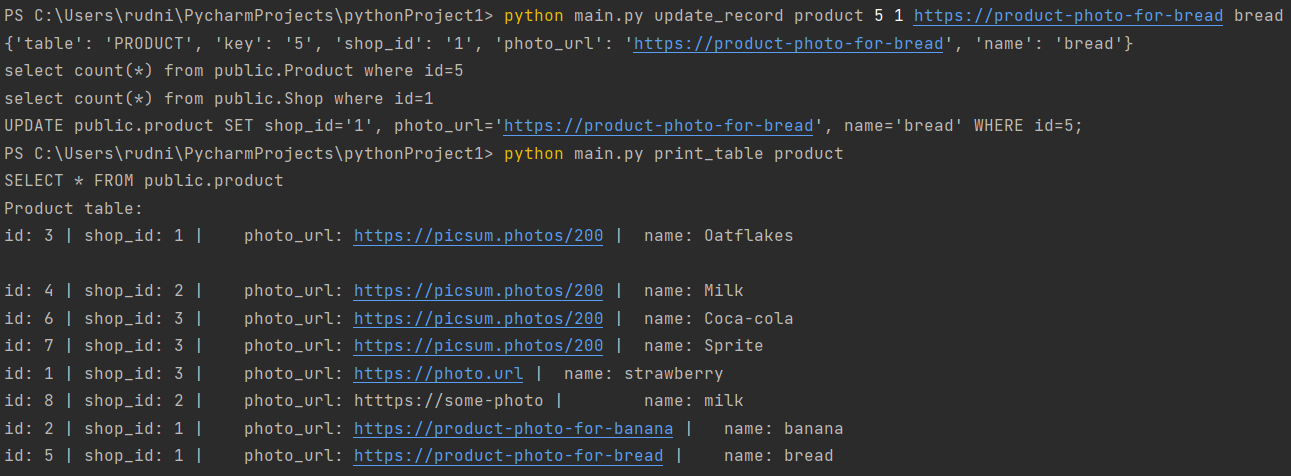
Запит на зміну даних

Внесення змін до запису у таблиці “Product”

Таблиця до внесення змін:

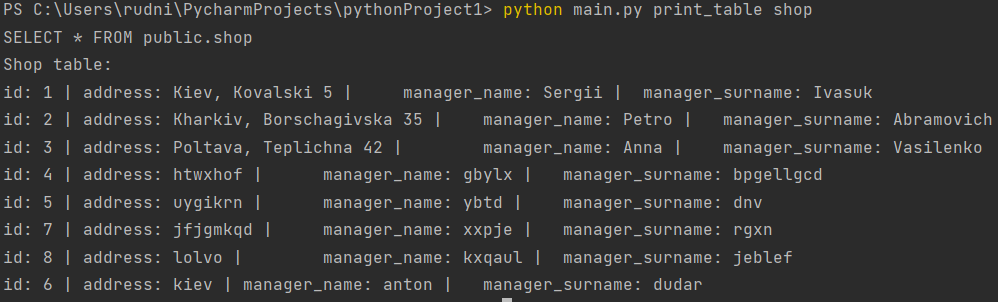


Таблиця після зміни запису з id 5:

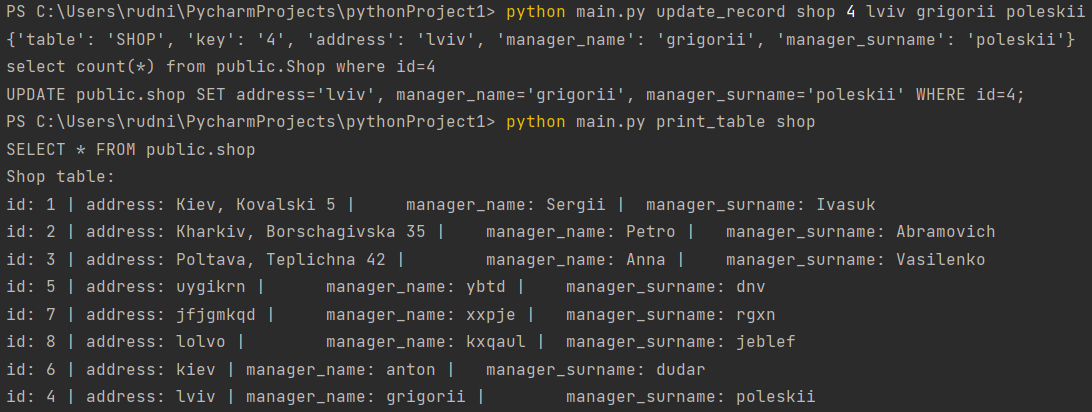


Внесення змін у таблицю “Shop”

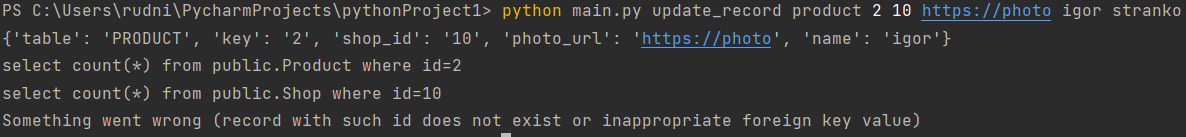
Таблиця до внесення змін:



Таблиця після внесення змін у запис з id 4:



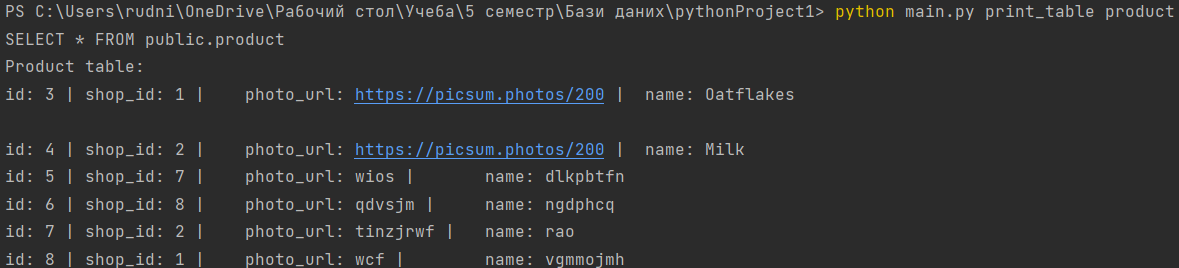
Спроба внесення змін до таблиці з неіснуючими даними:



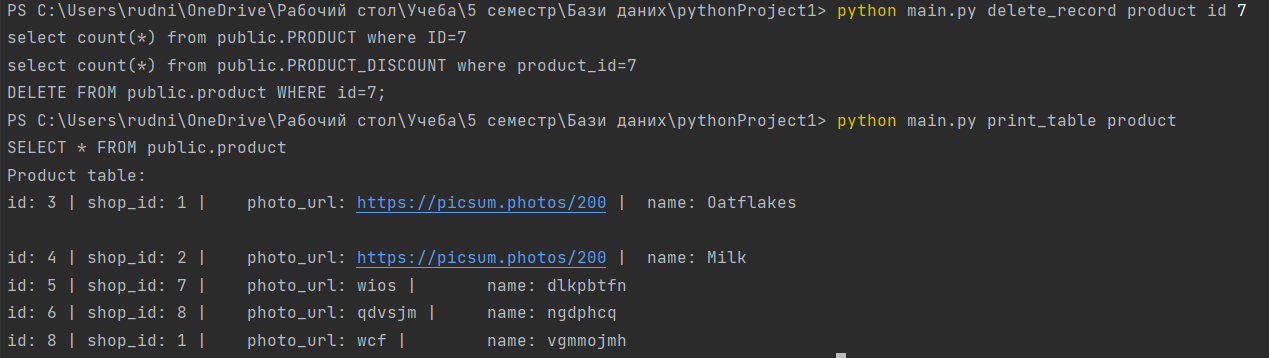
Запит на видалення:

Видалення запису з таблиці “Product”

Таблиця до видалення запису:

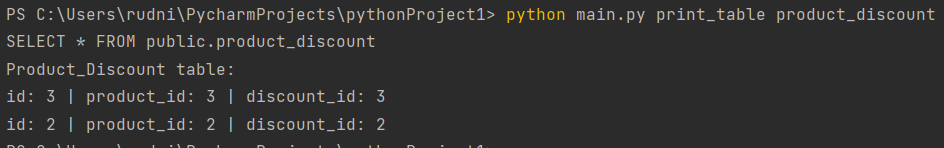


Таблиця після видалення запису з id 7:

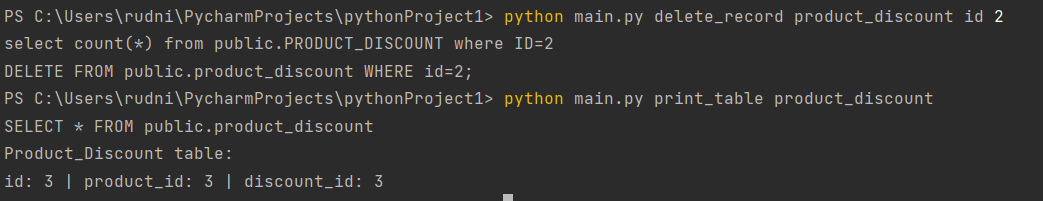


Видалення запису з таблиці “Product\_discount”

Таблиця до видалення запису:

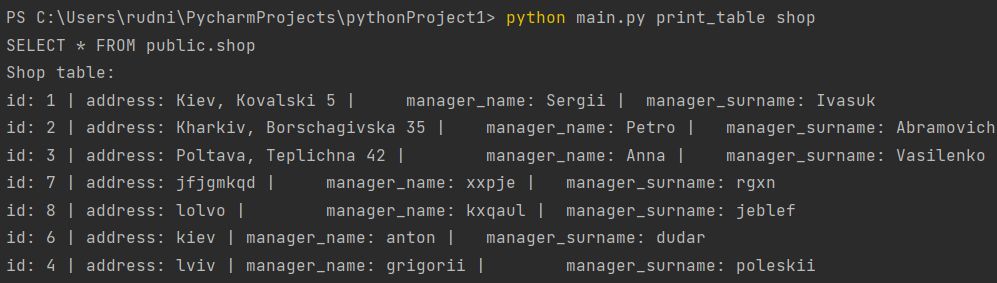


Таблиця після видалення запиту з id 2:

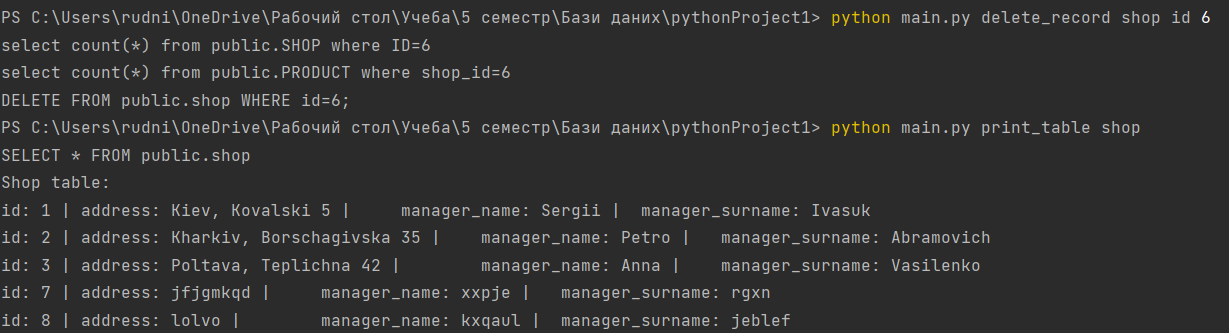


Видалення запису з таблиці “Shop”

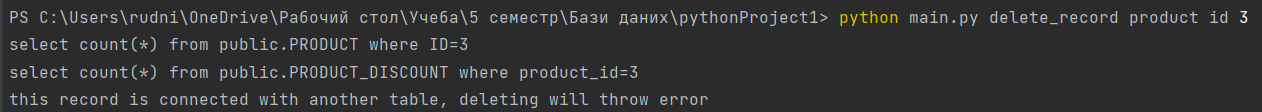
Таблиця до видалення запису:



Таблиця після видалення запису з id 6:



Спроба видалити запис Product, який використовюється у таблиці Product\_Discount:



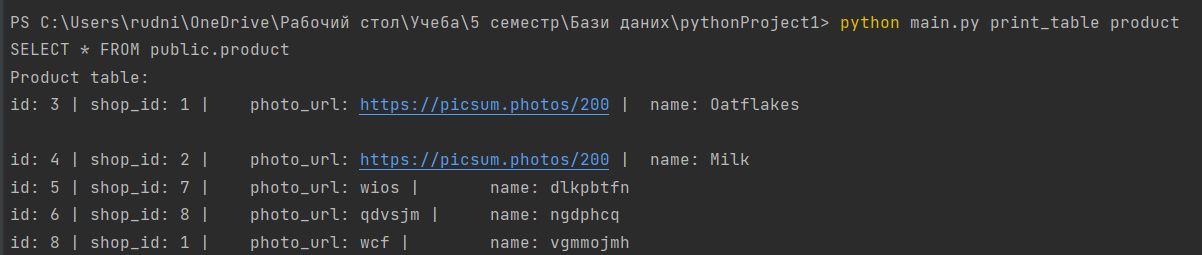
**Завдання 2**

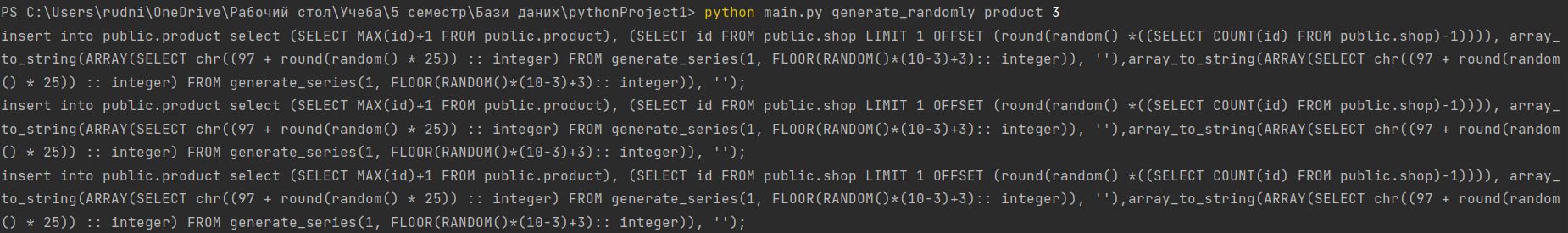
Методи реалізації:

1. Generate\_randomly – здійснює додавання до таблиці обраної користувачем кількісті рандомізованих записів. За аргументи приймає назву таблиці та число рандомізованих записів.

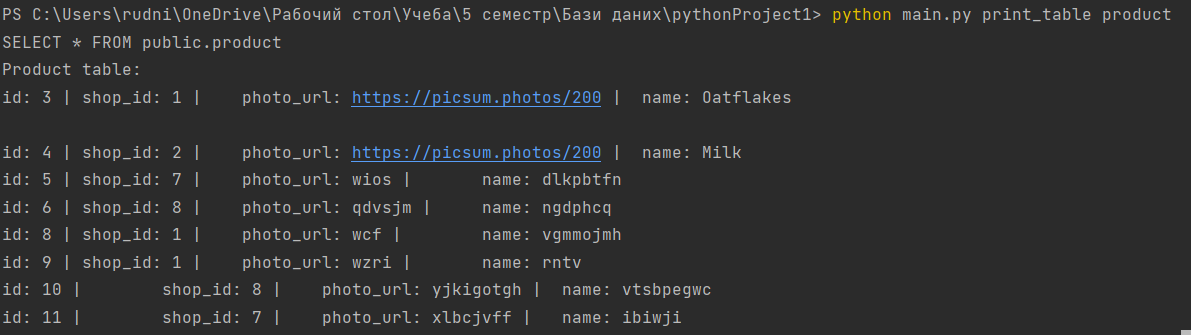
Вставка 3 рандомізованих записів до таблиці “Product”

Таблиця до вставки рандомізованих записів:



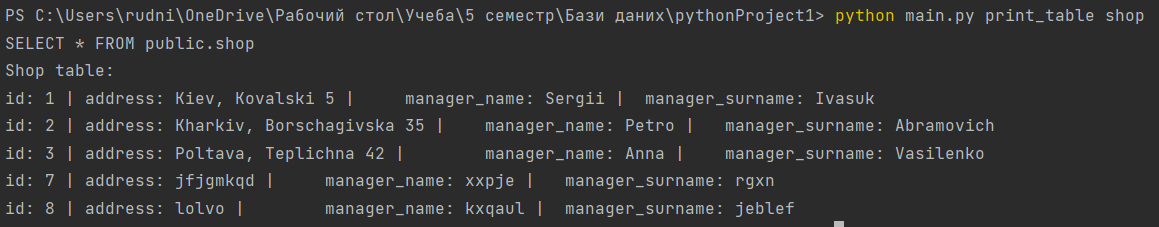
Запит:

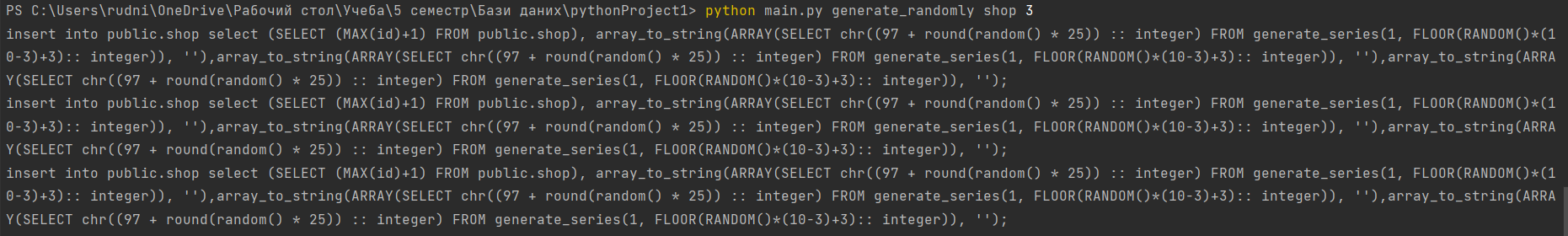
Таблиця після вставки трьох рандомізованих записів:



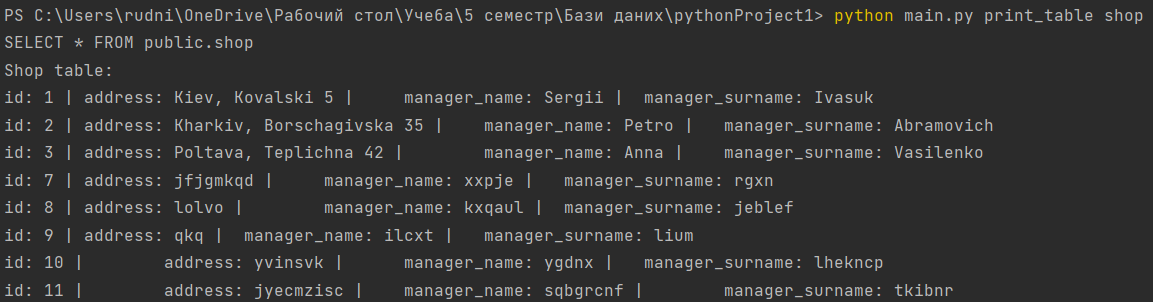
Вставка 4 рандомізованих записів до таблиці “Shop”

Таблиця до вставки рандомізованих записів:



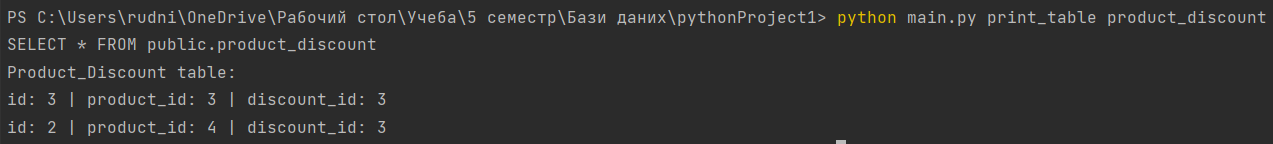
Запит:

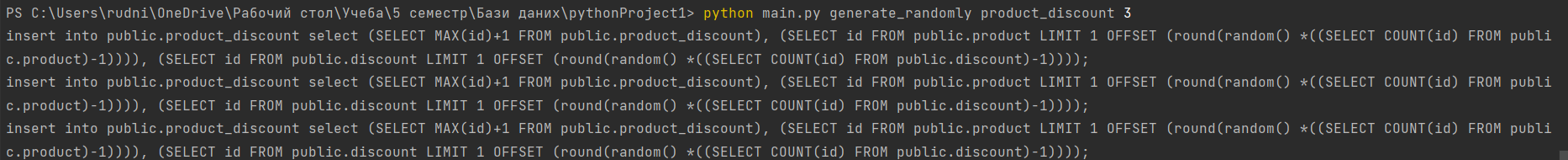
Таблиця після вставки чотирьох рандомізованих записів:



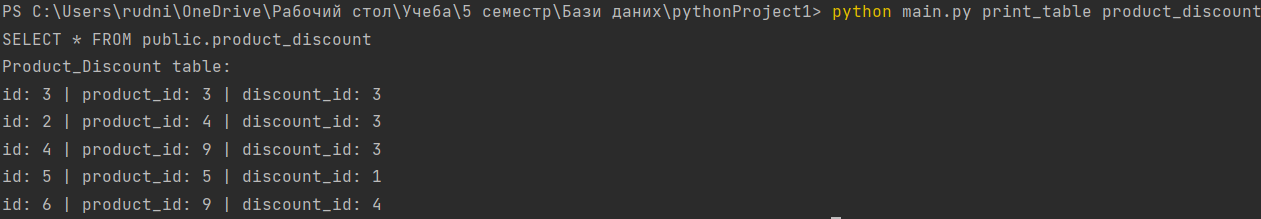
Вставка 4 рандомізованих записів до таблиці “Product\_discount”

Таблиця до вставки рандомізованих записів:



Запит:

Таблиця після вставки чотирьох рандомізованих записів:



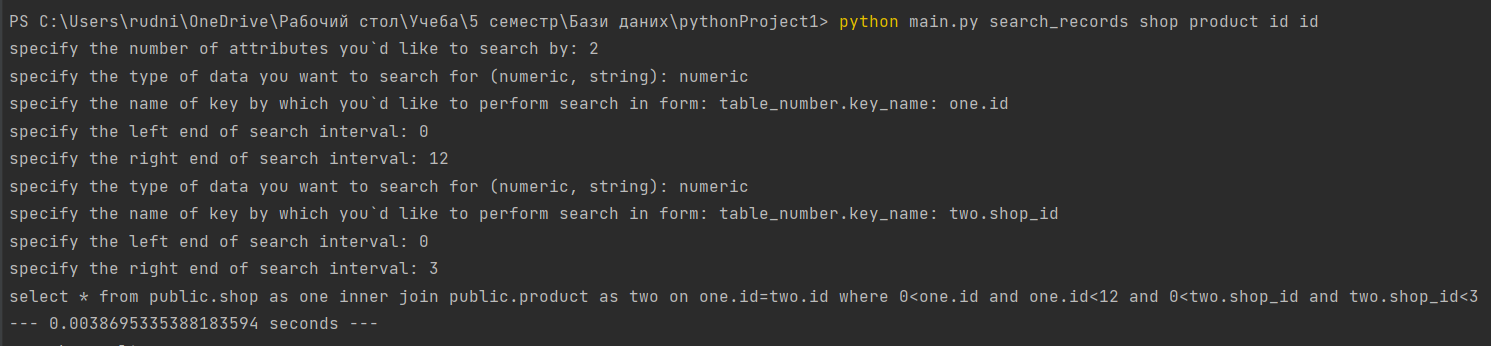
**Завдання 3**

Метод реалізації:

1. Search\_records – реалізує пошук за атрибутами введиних користувачем таблиць та виводить у вікно терміналу результат пошуку та час, за який він виконався. За аргументи функція приймає:
2. table1\_name, table2\_name, table1\_key, table2\_key;
3. table1\_name, table2\_name, table3\_name, table1\_key, table2\_key, table3\_key, table13\_key;
4. 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).

Пошук за двома атрибутами з двох таблиць (Product, Shop)

Формування запиту:

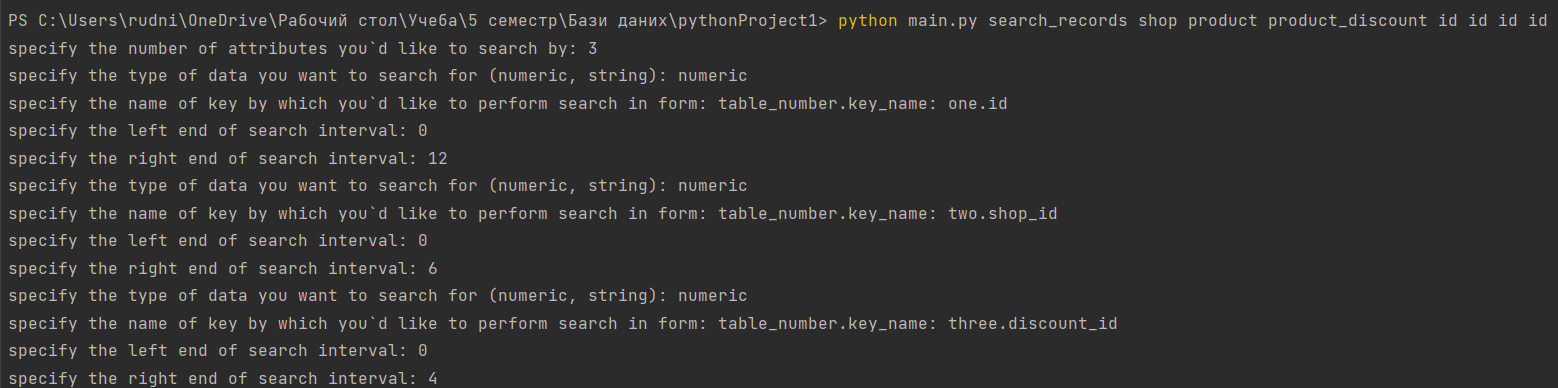


Запит:

select \* from public.shop as one inner join public.product as two on one.id=two.id where 0<one.id and one.id<12 and 0<two.shop\_id and two.shop\_id<3

Результат пошуку:

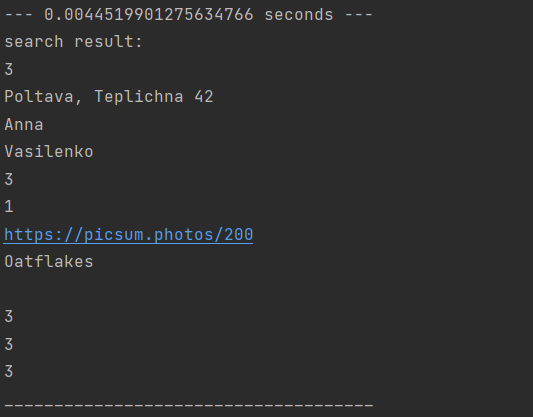
Пошук за трьома атрибутами з трьох таблиць (Shop, Product, Product\_discount) Формування запиту:



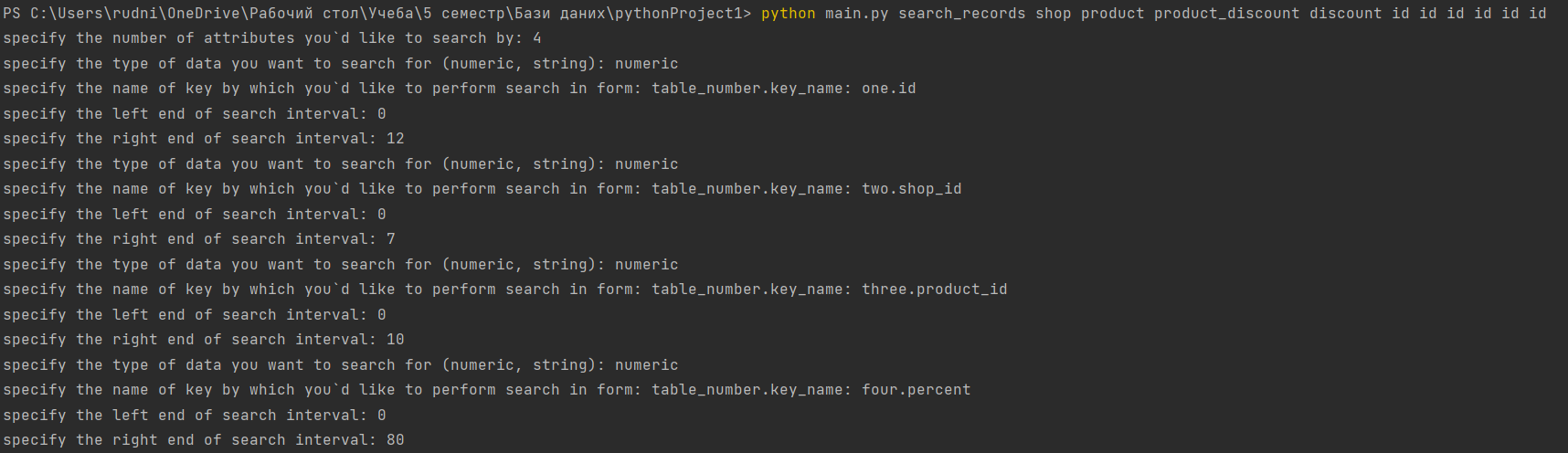
Запит:

select \* from public.shop as one inner join public.product as two on one.id=two.id inner join public.product\_discount as three on three.id=one.id where 0<one.id and one.id<12 and 0<two.shop\_id and two.shop\_id<6 and 0<three.discount\_id and three.discount\_id<4

Результат пошуку:



Пошук за чотирьма атрибутами з чотирьох таблиць (Shop, Product, Product\_discount, Discount)

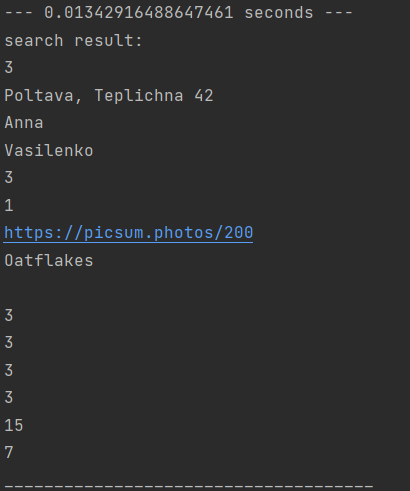
Формування запиту: 

Запит:

select \* from public.shop as one inner join public.product as two on one.id=two.id inner join public.product\_discount as three on three.id=one.id inner join public.discount as four on

four.id=two.id where 0<one.id and one.id<12 and 0<two.shop\_id and two.shop\_id<7 and 0<three.product\_id and three.product\_id<10 and 0<four.percent and four.percent<80

Результат пошуку:



**Завдання 4**

Програмний код модулю “Model”:

import psycopg2 as ps

class Model:

def \_\_init\_\_(self):

self.conn = None

try:

self.conn = ps.connect(

dbname="\*\*\*\*\*\*\*\*",

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:

return self.get(f"SELECT \* FROM public.product")

def print\_product\_discount(self) -> None:

return self.get(f"SELECT \* FROM public.product\_discount")

def print\_discount(self) -> None:

return self.get(f"SELECT \* FROM public.discount")

def print\_shop(self) -> None:

return self.get(f"SELECT \* FROM public.shop")

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, shop\_id: int, photo\_url: str, name: str) -> None:

self.request(f"UPDATE public.product SET shop\_id=\'{shop\_id}\', photo\_url=\'{photo\_url}\', name=\'{name}\' "

f"WHERE id={key\_value};")

def update\_data\_discount(self, key\_value: int, percent: int, duration: int) -> None:

self.request(f"UPDATE public.discount SET percent=\'{percent}\', duration=\'{duration}\' "

f"WHERE id={key\_value};")

def update\_data\_product\_discount(self, key\_value: int, product\_id: int, discount\_id: int) -> None:

self.request(f"UPDATE public.product\_discount SET product\_id=\'{product\_id}\', discount\_id=\'{discount\_id}\' "

f"WHERE id={key\_value};")

def update\_data\_shop(self, key\_value: int, address: str, manager\_name: str, manager\_surname: str) -> None:

self.request(f"UPDATE public.shop SET address=\'{address}\', manager\_name=\'{manager\_name}\', "

f"manager\_surname=\'{manager\_surname}\' WHERE id={key\_value};")

def insert\_data\_discount(self, id\_code: int, percent: int, duration: int) -> None:

self.request(f"insert into public.discount (id, percent, duration) "

f"VALUES ({id\_code}, \'{percent}\', \'{duration}\')")

def insert\_data\_product\_discount(self, id\_code: int, product\_id: int, discount\_id: int) -> None:

self.request(f"insert into public.product\_discount (id, product\_id, discount\_id) "

f"VALUES ({id\_code}, \'{product\_id}\', \'{discount\_id}\')")

def insert\_data\_product(self, id\_code: int, shop\_id: int, photo\_url: str, name: str) -> None:

self.request(f"insert into public.product (id, shop\_id, photo\_url, name) "

f"VALUES ({id\_code}, \'{shop\_id}\', \'{photo\_url}\', \'{name}\')")

def insert\_data\_shop(self, id\_code: int, address: str, manager\_name: str, manager\_surname: str) -> None:

self.request(f"insert into public.shop (id, address, manager\_name, manager\_surname) "

f"VALUES ({id\_code}, \'{address}\', \'{manager\_name}\', \'{manager\_surname}\')")

def product\_data\_generator(self, times: int) -> None:

for i in range(times):

self.request("insert into public.product select (SELECT MAX(id)+1 FROM public.product), "

"(SELECT id FROM public.shop LIMIT 1 OFFSET "

"(round(random() \*((SELECT COUNT(id) FROM public.shop)-1)))), "

"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)), '');")

def shop\_data\_generator(self, times: int) -> None:

for i in range(times):

self.request("insert into public.shop select (SELECT (MAX(id)+1) FROM public.shop), "

"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)), ''); ")

def discount\_data\_generator(self, times: int) -> None:

for i in range(times):

self.request("insert into public.discount select (SELECT MAX(id)+1 FROM public.discount), "

"FLOOR(RANDOM()\*(100000-1)+1),"

"FLOOR(RANDOM()\*(100000-1)+1); ")

def product\_discount\_data\_generator(self, times: int) -> None:

for i in range(times):

self.request("insert into public.product\_discount select (SELECT MAX(id)+1 FROM public.product\_discount), "

"(SELECT id FROM public.product LIMIT 1 OFFSET "

"(round(random() \*((SELECT COUNT(id) FROM public.product)-1)))), "

"(SELECT id FROM public.discount LIMIT 1 OFFSET "

"(round(random() \*((SELECT COUNT(id) FROM public.discount)-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}")

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\_product – отримання з бази даних та виведення у консоль користувача таблиці “Product”;
6. Print\_product\_discount– отримання з бази даних та виведення у консоль користувача таблиці “ Product\_discount”;
7. Print\_discount – отримання з бази даних та виведення у консоль користувача таблиці “Discount”;
8. Print\_shop – отримання з бази даних та виведення у консоль користувача таблиці “Shop”;
9. Delete\_data – реалізує видалення запису з обраної користувачем таблиці;
10. Update\_data\_(назва таблиці) – реалізує запит за зміну даних у обраній користувачем таблиці;
11. Insert\_data\_(назва таблиці) – реалізує запит на вставку запису до обраної користувачем таблиці;
12. (назва таблиці)\_data\_generator – реалізує запит на вставку рандомізованих записів до обраної користувачем таблиці;
13. Search\_data\_(кількість таблиць)\_tables – реалізує пошук даних у вибраній користувачем кількості таблиць;