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"

Виконав:
студент 3 курсу
групи КВ-94
Поляруш Є. М.
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/EugenePoliarush/Lab2.git

Використана мова програмування: Python 3.7

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), prettytable (для гарного виводу БД), time (для виміру часу запиту пошуку для завдання 3).

Відомості про обрану предметну галузь з лабораторної роботи №1

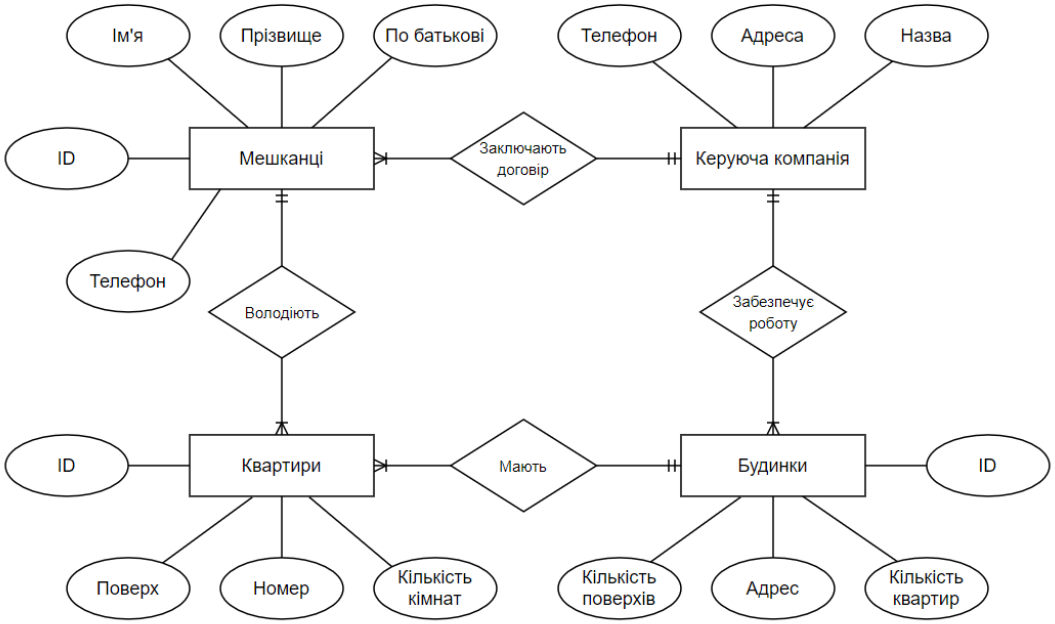


Рисунок 1. ER-діаграма, побудована за нотацією Чена

Опис предметної галузі

Дана предметна галузь передбачає роботу компанії певного житлового комплексу і її зв'язок з мешканцями.

Перетворення моделі у схему бази даних

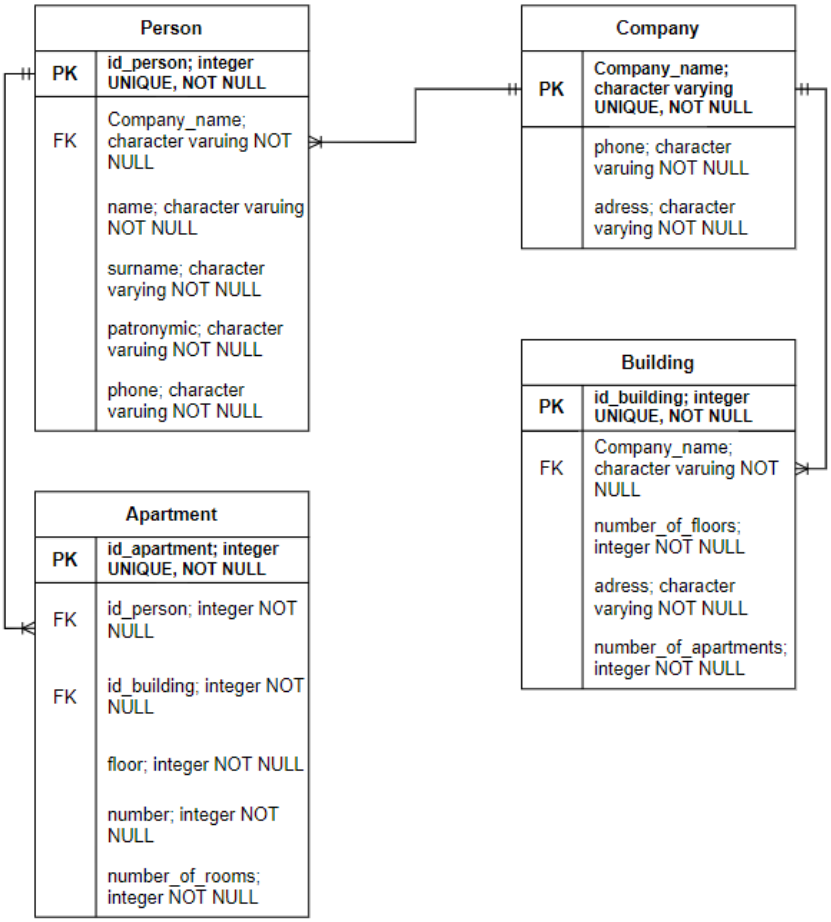


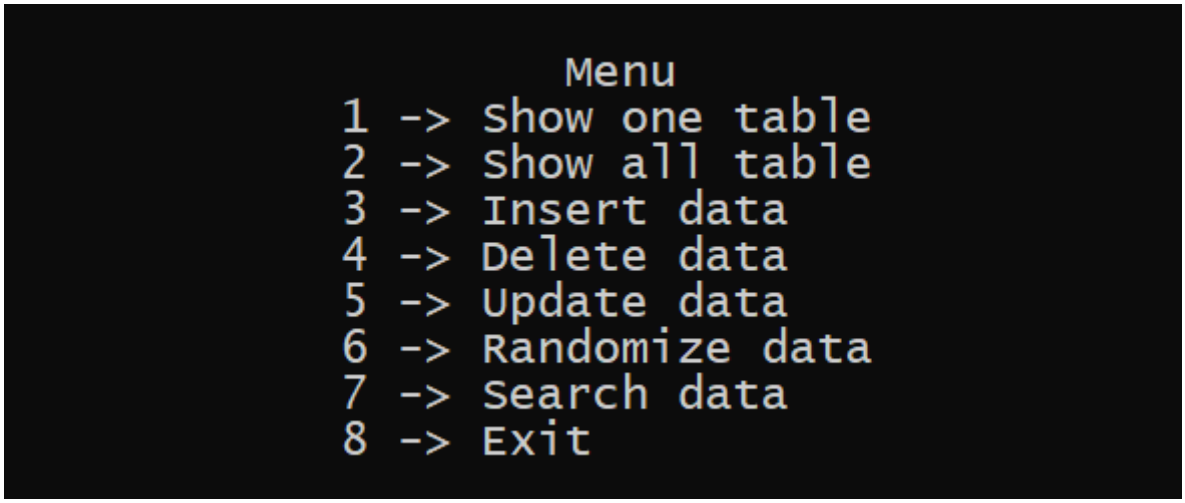
Рисунок 2. Схема бази даних

Сутність “Керуюча компанія” було перетворено у таблицю “Company”. Сутність “Будинки” було перетворено у таблицю “Building”, зв’язок 1:N цієї сутності з сутністю “Керуюча компанія” зумовив появу у ній зовнішнього ключа Company name. Сутність “Квартири” було перетворено у таблицю “Apartment”, зв’язки 1:N цієї сутності з сутностями “Будинки” і “Мешканці” зумовили появу у ній зовнішніх ключів id_person, id_building. Сутність “Мешканці” було перетворено у таблицю “Person”, зв’язок 1:N цієї сутності з сутністю “Керуюча компанія” зумовив появу у ній зовнішнього ключа Company name.

Сутність	Атрибут	Тип атрибуту
Company *- містить дані про компанію*	***Company_name*** *- унікальний ідентифікатор компанії, тобто її назва* ***phone*** *– телефон компанії* ***adress*** *– адреса компанії*	***character varying*** *(рядок)* ***character varying*** *(рядок)* ***character varying*** *(рядок)*
Buildilng *- містить дані про будинки, якими управляє компанія*	***id_building*** *- унікальний ідентифікатор будинку* ***Company_name*** *- назва компанії, яка ним керує* ***number_of_floors*** *– кількість поверхів у будинку* ***adress*** *– адреса будинку* ***number_of_apartments*** *– кількість квартир у будинку*	***integer*** *(числовий)* ***character varying*** *(рядок)* ***integer*** *(числовий)* ***character varying*** *(рядок)* ***integer*** *(числовий)*
Apartment *– містить дані про квартири*	***id_apartment*** *- унікальний ідентифікатор квартири* ***id_building*** *– ідентифікатор будинку в якому знаходиться квартира* ***id_person*** *– ідентифікатор особи, яка володіє квартирою* ***floor*** *– номер поверху де розташована квартира* ***number*** *– номер квартири* ***number_of_rooms*** *– кількість кімнат в квартирі*	***integer*** *(числовий)* ***integer*** *(числовий)* ***integer*** *(числовий)* ***integer*** *(числовий)* ***integer*** *(числовий)* ***integer*** *(числовий)*
Person *- містить дані про мешканця*	***id_person*** *- унікальний ідентифікатор мешканця* ***Company_name*** *– ідентифікатор компанії* ***name*** *– Ім’я мешканця* ***surname*** *– Прізвище мешканця* ***patronymic*** *– По батькові* ***phone*** *– телефон мешканця*	***integer*** *(числовий)* ***character varying*** *(рядок)* ***character varying*** *(рядок)* ***character varying*** *(рядок)* ***character varying*** *(рядок)* ***character varying*** *(рядок)*

Таблиця 1. Опис структури БД

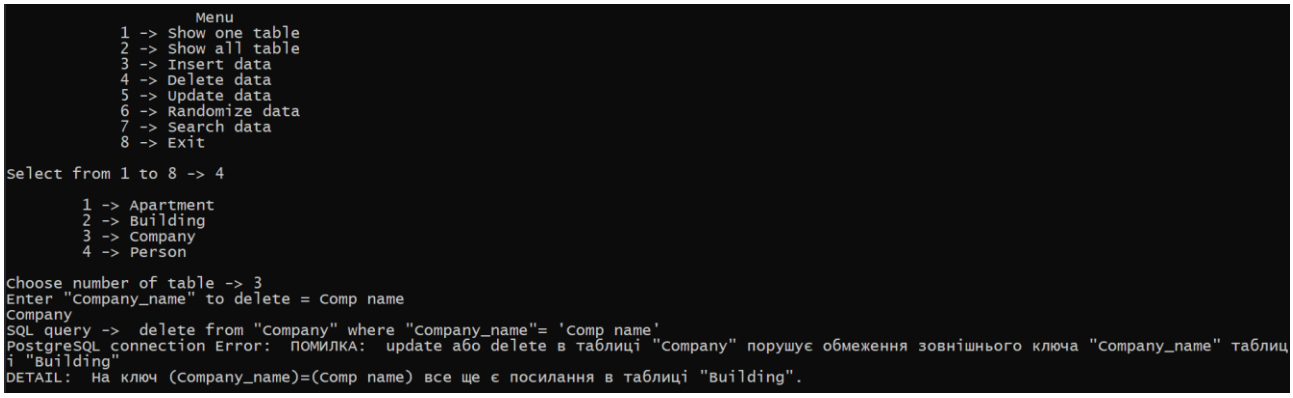
Схема меню



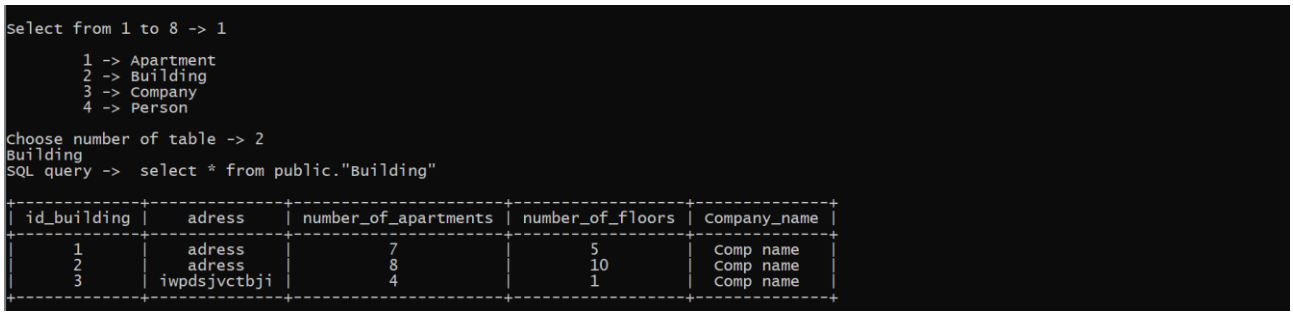
На скріншоті показано схему меню користувача. Пункти 1 та 2 призначенні для виведення даних. Пункти 3, 4 та 5 для редагувань відповідно вставлення даних, видалення, та редагування. Пункт 6 призначений для запису рандомізованих даних в певну таблицю задану користувачем. Пункт 7 забезпечує пошук за атрибутами.

Завдання 1

Запит на видалення



Як можна побачити видалення не було здійснено, оскільки на даний ключ є посилання в іншій таблиці, тому було видано повідомлення про помилку.



```
                    Menu
           1 -> Show one table
           2 -> Show all table
           3 -> Insert data
           4 -> Delete data
           5 -> Update data
           6 -> Randomize data
           7 -> Search data
           8 -> Exit

Select from 1 to 8 -> 4

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
Enter "id_building" to delete = 3
Building
SQL query ->  delete from "Building" where "id_building"=  3
Data deleted successfully!
```

Ось випадок з успішним видаленням поля.

```
Select from 1 to 8 -> 1

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
Building
SQL query ->  select * from public."Building"

+-------------+--------+----------------------+------------------+--------------+
| id_building | adress | number_of_apartments | number_of_floors | Company_name |
+-------------+--------+----------------------+------------------+--------------+
|      1      | adress |          7           |        5         |  Comp name   |
|      2      | adress |          8           |        10        |  Comp name   |
+-------------+--------+----------------------+------------------+--------------+
```

Запит на вставку поля

```
Select from 1 to 8 -> 3

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
id_building = 3
adress = fdgsfg
number_of_apartments = 9
number_of_floors = 8
Company_name = Comp name2
Building
Sql query ->  insert into "Building" ("id_building", "adress", "number_of_apartments", "number_of_floors", "Company_name") values (3, '
fdgsfg', 9, 8, 'Comp name2')
PostgreSQL connection Error:  ПОМИЛКА:  insert або update в таблиці "Building" порушує обмеження зовнішнього ключа "Company_name"
DETAIL:  Ключ (Company_name)=(Comp name2) не присутній в таблиці "Company".
```

Як можна побачити додавання даних не було здійснено, оскільки заданий ключ, не присутній в таблиці ‘Company’, тому було видано повідомлення про помилку.

```
Select from 1 to 8 -> 3

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
id_building = 3
adress = sdfsdg
number_of_apartments = 9
number_of_floors = 8
Company_name = Comp name
Building
Sql query ->  insert into "Building" ("id_building", "adress", "number_of_apartments", "number_of_floors", "Company_name") values (3, '
sdfsdg', 9, 8, 'Comp name')
Data added successfully!
```

Ось приклад успішного додавання даних.

```
Select from 1 to 8 -> 1

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
Building
SQL query ->  select * from public."Building"

+-------------+--------+----------------------+------------------+--------------+
| id_building | adress | number_of_apartments | number_of_floors | Company_name |
+-------------+--------+----------------------+------------------+--------------+
|      1      | adress |          7           |        5         |  Comp name   |
|      2      | adress |          8           |        10        |  Comp name   |
|      3      | sdfsdg |          9           |        8         |  Comp name   |
+-------------+--------+----------------------+------------------+--------------+
```

Запит на зміну полів

```
Select from 1 to 8 -> 5

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
id_building of row to update = 3

            1 -> id_building
            2 -> adress
            3 -> number_of_apartments
            4 -> number_of_floors
            5 -> Company_name

Number of attribute =>5
New value of attribute = Comp name2
Building
SQL query ->  update "Building" set "Company_name"= 'Comp name2' where "id_building"= '3'
PostgreSQL connection Error:  ПОМИЛКА:  insert або update в таблиці "Building" порушує обмеження зовнішнього ключа "Company_name"
DETAIL:  Ключ (Company_name)=(Comp name2) не присутній в таблиці "Company".
```

Як можна побачити редагування даних не було здійснено, оскільки заданий ключ, не присутній в таблиці ‘Company’, тому було видано повідомлення про помилку, аналогічна помилка при вставці.

```
Select from 1 to 8 -> 5
-
        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
id_building of row to update = 3

            1 -> id_building
            2 -> adress
            3 -> number_of_apartments
            4 -> number_of_floors
            5 -> Company_name

Number of attribute =>2
New value of attribute = New_adress
Building
SQL query ->  update "Building" set "adress"= 'New_adress' where "id_building"= '3'
Data updeted successfully!
```

Ось приклад успішного редагування даних, нижче показано нову адресу, для 3 рядка.

```
Select from 1 to 8 -> 1

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of table -> 2
Building
SQL query ->  select * from public."Building"

+-------------+------------+----------------------+-----------------+--------------+
| id_building |   adress   | number_of_apartments | number_of_floors | Company_name |
+-------------+------------+----------------------+-----------------+--------------+
|      1      |   adress   |          7           |        5        |  Comp name   |
|      2      |   adress   |          8           |        10       |  Comp name   |
|      3      | New_adress |          9           |        8        |  Comp name   |
+-------------+------------+----------------------+-----------------+--------------+
```

Завдання 2

Вставка 10 псевдорандомізованих записів у кожну з таблиць.

Початкова таблиця

```
Apartment
SQL query ->  select * from public."Apartment"

+---------------+-------+--------+------------------+------------+--------------+
| id_apartment | floor | number | number_of_rooms | id_person | id_building |
+---------------+-------+--------+------------------+------------+--------------+
|       1       |   3   |   1    |        2         |     3      |      1       |
|       2       |   2   |   4    |        1         |     2      |      2       |
|       3       |   2   |   5    |        3         |     2      |      2       |
|       4       |   5   |   1    |        3         |     3      |      1       |
|       5       |   8   |   4    |        2         |     2      |      1       |
+---------------+-------+--------+------------------+------------+--------------+
```

```
Choose number of table -> 1
How many rows to random? -> 10
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
SQL query ->  insert into "Apartment" select (select max(id_apartment)+1 from "Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil
((RANDOM()*3)),(select ceil(RANDOM()*max(id_person)) from "Person"),(select ceil(RANDOM()*max(id_building)) from "Building")
Data randomed successfully!
```

Результат

```
Apartment
SQL query ->  select * from public."Apartment"
```

id_apartment	floor	number	number_of_rooms	id_person	id_building
1	3	1	2	3	1
2	2	4	1	2	2
3	2	5	3	2	2
4	5	1	3	3	1
5	8	4	2	2	1
6	5	2	2	4	2
7	8	4	1	3	3
8	8	1	2	3	2
9	7	1	2	4	2
10	3	8	3	3	1
11	2	6	3	2	1
12	7	10	2	2	2
13	9	5	1	4	2
14	2	6	2	3	1
15	2	3	1	4	1

Початкова таблиця

```
Building
SQL query ->  select * from public."Building"
```

id_building	adress	number_of_apartments	number_of_floors	Company_name
1	adress	7	5	Comp name
2	adress	8	10	Comp name
3	New_adress	9	8	Comp name

```
Choose number of table -> 2
How many rows to random? -> 10
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
SQL query ->  insert into "Building" select (select max(id_building)+1 from "Building"),array_to_string(ARRAY(select chr((97 + round(ra
ndom()*25))::integer) From generate_series(1,floor(RANDOM()*(25-10)+10)::integer)),' '),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
 "Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)
Data randomed successfully!
```

Результат

id_building	adress	number_of_apartments	number_of_floors	Company_name
1	adress	7	5	Comp name
2	adress	8	10	Comp name
3	New_adress	9	8	Comp name
4	a x v c o l x g b a e e g y g m r j y h q s	3	3	Comp name
5	i v v o h n j d e r c c d f i g x o t	8	4	Comp name
6	m l k s x f j h t x k m l m m r z j j t d x k	5	8	Comp name
7	z e y w m r b m e q u w w	7	4	Comp name
8	c y r d g r g q r q g p	6	7	Comp name
9	m b y v u g v p c j k k r k	1	9	Comp name
10	f u f j l v w o p f d q g r k k n w q h t j	5	8	Comp name
11	t b r v p t u h k v f e p d c r t y q l n	9	8	Comp name
12	s o m c r n n x c b n q w i k f v j r w t	10	3	Comp name
13	h y s o v s i d d o u d t e c q p e h x g b s u	5	9	Comp name

Початкова таблиця

```
Company
SQL query ->  select * from public."Company"
```

Company_name	adress	phone
Comp name	some adress	09634634645

```
How many rows to random? -> 10
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
SQL query ->  insert into "Company" select 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()*(25-10)+10)::integer)),' '),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From generate_series(1,f
loor(RANDOM()*(10-8)+8)::integer)),' ')
Data randomed successfully!
```

Результат

Company_name	adress	phone
Comp name	some adress	09634634645
h w x f l u h n n h e x w x j z y	h w t u a b v v a j p e l i n v r	2 2 3 7 7 4 7 8
d t w l e m g g q l e f q n x e d w l r f x t r	x r n g i m r v k d j t r p v g b	4 9 2 5 5 7 8 1
u b q q q b n b s g g n q y q m	m g u m k l n n k k v n j h s e	5 8 3 3 4 4 5 8 7
b o y i t m t f s c b u a h v w	m u o b k q n k x v	3 1 3 4 7 3 6 4 9
f y u q t u j l d e n t p p g h q y	s h k l v t k q j r o	5 1 3 3 7 0 1 8
u m c p b o l g r q o e q f n p	c w g k e v k v r y t u	4 5 1 1 5 5 1 2 8
m s l a l i d a e d o f l x q f t k r s e	e e s t m w i f n u g g m g	2 7 4 8 9 6 9 0
c x s l r t x j c t f j d	p j k f h o c c i i p q s	4 8 4 5 6 2 1 5
t u a p u i x c z s d	a k y h s z a c w q x g s v	4 7 6 6 7 3 1 5 6
w v d k z j m p g q m f m l o d k f z	e n w d m k s j h y w j j g e p	1 2 8 3 0 4 1 1

Початкова таблиця

```
Person
SQL query ->  select * from public."Person"

+-----------+--------------------+-------------------+--------------+-------------+---------------+
| id_person |        name        |      surname      |  patronymic  |    phone    | Company_name  |
+-----------+--------------------+-------------------+--------------+-------------+---------------+
|     1     |       Name1        |     Surname1      | patronymic1  | 0954634634  |  Comp name    |
|     2     |       Name2        |     Surname2      | patronymic2  | 0936346344  |  Comp name    |
|     3     |       Name3        |     Surname3      | patronymic3  | 0964354534  |  Comp name    |
|     4     | etmzxtenbtohfbspv  | njacepuxaualttn   |  jxihzmgtef  | 029186394   |  Comp name    |
+-----------+--------------------+-------------------+--------------+-------------+---------------+
```

```
Choose number of table -> 4
How many rows to random? -> 10
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()*(25-10)+10)::integer)),' '),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((48 + round(random()
*9))::integer) From generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),' '),(SELECT "Company_name" FROM "Company" ORDER BY RANDOM()
LIMIT 1)
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()*(25-10)+10)::integer)),' '),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((48 + round(random()
*9))::integer) From generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),' '),(SELECT "Company_name" FROM "Company" ORDER BY RANDOM()
LIMIT 1)
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()*(25-10)+10)::integer)),' '),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((48 + round(random()
*9))::integer) From generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),' '),(SELECT "Company_name" FROM "Company" ORDER BY RANDOM()
LIMIT 1)
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()*(25-10)+10)::integer)),' '),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((48 + round(random()
*9))::integer) From generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),' '),(SELECT "Company_name" FROM "Company" ORDER BY RANDOM()
LIMIT 1)
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()*(25-10)+10)::integer)),' '),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((48 + round(random()
*9))::integer) From generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),' '),(SELECT "Company_name" FROM "Company" ORDER BY RANDOM()
LIMIT 1)
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()*(25-10)+10)::integer)),' '),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((48 + round(random()
*9))::integer) From generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),' '),(SELECT "Company_name" FROM "Company" ORDER BY RANDOM()
LIMIT 1)
SQL query ->  insert into "Person" select (select max(id_Person)+1 from "Person"),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()
```

Результат(Оскільки таблиця завелика в консолі відображається не зовсім гарно)

```
+----------+----------------------------------------+--------------------------------------+------------------------------
--------+----------------+-------------------------------+
| id_person |                 name                  |               surname                |          patronymic
        |     phone      |         Company_name          |
+----------+----------------------------------------+--------------------------------------+------------------------------
--------+----------------+-------------------------------+
|    1     |                Name1                   |               Surname1               |          patronymic1
        |   0954634634   |           Comp name           |
|    2     |                Name2                   |               Surname2               |          patronymic2
        |   0936346344   |           Comp name           |
|    3     |                Name3                   |               Surname3               |          patronymic3
        |   0964354534   |           Comp name           |
|    4     |          etmzxtenbtohfbspv             |            njacepuxaualttn           |           jxihzmgtef
        |   029186394    |           Comp name           |
|    5     | s y q t t l k w i f i f d r t m u o v z f f | q i u x p r m m i f c m g w j n i g c |   d h m x c q b j h t e x h y k e
q u     | 5 7 3 8 1 2 7 8 2 |              Comp name           |
|    6     |        i f l n p p p u q o w d l k c         |   w b v p f r x e b x y              |    x c s c o n h q z a h t u b
j        | 8 1 5 3 3 8 4 2 5 | f y u q t u j l d e n t p p g h q y |
|    7     |         r q l t v a q j f s p a u c          | z c o s a v f b v p m n m a x g s x x |       j o g j f j k d o i
        | 2 9 3 2 3 3 7 1 5 |   u m c p b o l g r q o e q f n p   |
|    8     |      i g g k k f p o p g e g e c e o g       |   i i v o h c j x n d o l            | h l c m a l r k j u y n p b j r l i
o o k k | 2 4 9 6 9 5 6 7 9 |      c x s l r t x j c t f j d      |
|    9     |      h n i s q m d h s v c s k x e k         |   a l b m g i w p l m h              | a w q l y v r t h m y p j f j l l y
e s v m | 4 5 1 1 6 1 3 9 5 |   b o y i t m t f s c b u a h v w   |
|    10    |       n p b i j v n y p p a w a t w          | h l g m c t w g e g d d j x o b x h  |   f i u f l p w v u s g p n o w e
t k     | 6 4 8 5 7 0 8 5 0 |   u m c p b o l g r q o e q f n p   |
|    11    | w d q o r n o j v g w e u j u n k a k h o    |   l s v n k y n d w t z s            |      z l k r f f w x d f q u
        |  7 4 7 3 1 2 7 2  |   b o y i t m t f s c b u a h v w   |
|    12    |        x q k s k g t r y l c r i             | t h w g v s w w b q q r q x q y j v  |   u t x f l r z t p c x h m x l x
k u t   |  2 6 1 3 5 3 9 2  |   u b q q q b n b s g g n q y q m   |
|    13    |         e n f m z d u p p n                  |   o m m v m a u r m i w u c u l      |      c j b h d u c b y t s q
        | 8 2 6 4 7 4 3 1 7 |      c x s l r t x j c t f j d      |
|    14    |     u r g v l w p w n v s o l k s q l        |   w h r i g u v u c f y x            | t e r c z o v v w j r d u f j v y
o x l g | 5 1 0 4 3 7 8 1 3 |              Comp name           |
+----------+----------------------------------------+--------------------------------------+------------------------------
--------+----------------+-------------------------------+
```

Завдання 3

Пошук за трьома атрибутами з двох таблиць (Person, Apartment).

```
Select from 1 to 8 -> 7
Enter numbers of tables for search from 2 to 4-> 2
Enter numbers of atribute for search -> 3

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of first table.
Choose number of table -> 4
Choose number of second table.
Choose number of table -> 1
Person

            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name

Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building

Enter key1 -> id_person
Person

            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name
```

```
Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building

Choose type of atribute
 1 -> numeric
 2 -> string 1
Enter name of first atribute in form table.artibute -> two.id_apartment
min = 2
max = 10
Person

            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name

Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building

Choose type of atribute
 1 -> numeric
 2 -> string 2
Enter name of first atribute in form table.artibute -> one.name
Text = Name2
Person
```

```
            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name

Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building

Choose type of atribute
 1 -> numeric
 2 -> string 1
Enter name of first atribute in form table.artibute -> two.floor
min = 3
max = 10
```

```
SQL query ->  select * from public."Person" as one inner join public."Apartment" as two using(id_person) where 2<two.id_apartment and t
wo.id_apartment<10 and one.name LIKE 'Name2' and 3<two.floor and two.floor<10
Time:  0.0070133209228515625
+-----------+-------+----------+-------------+------------+--------------+--------------+-------+--------+-----------------+-----------
--+
| id_person |  name | surname  |  patronymic |   phone    | Company_name | id_apartment | floor | number | number_of_rooms | id_buildin
g |
+-----------+-------+----------+-------------+------------+--------------+--------------+-------+--------+-----------------+-----------
--+
|     2     | Name2 | Surname2 | patronymic2 | 0936346344 |  Comp name   |      5       |   8   |   4    |        2        |      1
  |
+-----------+-------+----------+-------------+------------+--------------+--------------+-------+--------+-----------------+-----------
--+
Data search successfully!
```

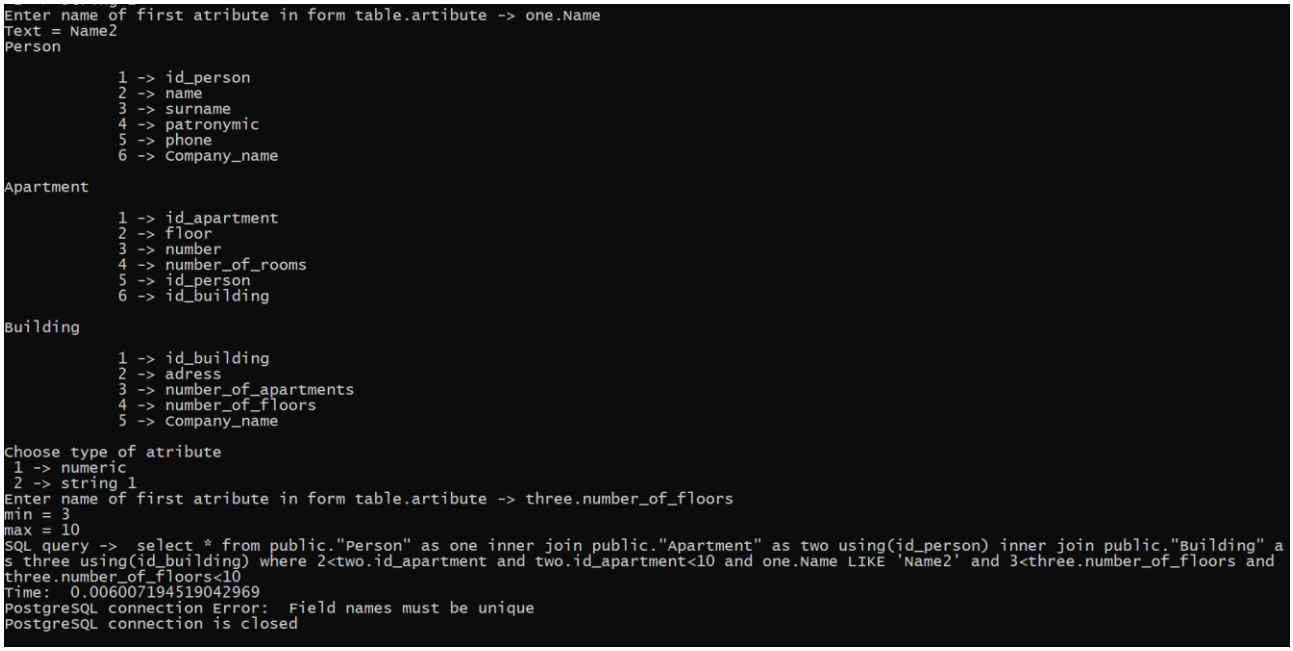
Пошук за трьома атрибутами з трьох таблиць (Person, Apartment, Building)

(В даному випадку програма працює коректно, але вивід не відбувається оскільки існують стовпці з однаковою назвою, при виконанні того ж запита в pgAdmin4 вивід відбувається коректно)

```
Select from 1 to 8 -> 7
Enter numbers of tables for search from 2 to 4-> 3
Enter numbers of atribute for search -> 3

        1 -> Apartment
        2 -> Building
        3 -> Company
        4 -> Person

Choose number of first table.
Choose number of table -> 4
Choose number of second table.
Choose number of table -> 1
Choose number of third table.
Choose number of table -> 2
Person

            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name
```

```
Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building

Building

            1 -> id_building
            2 -> adress
            3 -> number_of_apartments
            4 -> number_of_floors
            5 -> Company_name

Enter key1 -> id_person
Enter key2 -> id_building
Person

            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name

Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building
```

```
Building

            1 -> id_building
            2 -> adress
            3 -> number_of_apartments
            4 -> number_of_floors
            5 -> Company_name

Choose type of atribute
 1 -> numeric
 2 -> string 1
Enter name of first atribute in form table.artibute -> two.id_apartment
min = 2
max = 10
Person

            1 -> id_person
            2 -> name
            3 -> surname
            4 -> patronymic
            5 -> phone
            6 -> Company_name

Apartment

            1 -> id_apartment
            2 -> floor
            3 -> number
            4 -> number_of_rooms
            5 -> id_person
            6 -> id_building

Building

            1 -> id_building
            2 -> adress
            3 -> number_of_apartments
            4 -> number_of_floors
            5 -> Company_name

Choose type of atribute
 1 -> numeric
 2 -> string 2
```



Завдання 4

Код модулю model.py

```
import Connect
import time
from View import View
from prettytable import from_db_cursor

tables = {
    1: 'Apartment',
    2: 'Building',
    3: 'Company',
    4: 'Person',
}

class Model:
    @staticmethod
    def CheckTable():
        flag = True
        while flag:
            table = input('Choose number of table -> ')
            if table.isdigit():
                table = int(table)
                if table >= 1 and table <= 4:
                    flag = False
                else:
                    print('Wrong number, choose again.')
            else:
```

```
            print('Wrong number, choose again.')
    return table

@staticmethod
def showOneTable():
    View.list()
    connect = Connect.makeConnect()
    cursor = connect.cursor()

    table = Model.CheckTable()

    table_name = """" + tables[table] + """"
    print(tables[table])

    show = 'select * from public.{}'.format(table_name)

    print("SQL query -> ", show)
    print('')
    cursor.execute(show)
    printtable = from_db_cursor(cursor)
    records = cursor.fetchall()
    obj = View(table, records, printtable)
    obj.show()
    cursor.close()
    Connect.closeConnect(connect)

@staticmethod
def showAllTables():
    connect = Connect.makeConnect()
    cursor = connect.cursor()
    for table in range(1, 5):
        table_name = """" + tables[table] + """"
        print(tables[table])

        show = 'select * from public.{}'.format(table_name)

        print("SQL query -> ", show)
        print('')
        cursor.execute(show)
        printtable = from_db_cursor(cursor)
        records = cursor.fetchall()
```

```
            obj = View(table, records, printtable)
            obj.show()
        cursor.close()
        Connect.closeConnect(connect)

    @staticmethod
    def insert():
        connect = Connect.makeConnect()
        cursor = connect.cursor()
        flag = True
        while flag:
            View.list()
            table = Model.CheckTable()
            if table == 1:
                id_apartment = input("id_apartment = ")
                floor = input("floor = ")
                number = input("number = ")
                number_of_rooms = input('number_of_rooms = ')
                id_person = input('id_person = ')
                id_building = input('id_building = ')

                insert = 'insert into "Apartment" ("id_apartment", "floor", "number",
"number_of_rooms", "id_person", "id_building") values ({}, {}, {}, {}, {},
{})'.format(
                    id_apartment, floor, number, number_of_rooms, id_person, id_building)

                flag = False
            elif table == 2:
                id_building = input('id_building = ')
                adress = "'" + input('adress = ') + "'"
                number_of_apartments = input('number_of_apartments = ')
                number_of_floors = input('number_of_floors = ')
                Company_name = "'" + input('Company_name = ') + "'"

                insert = 'insert into "Building" ("id_building", "adress",
"number_of_apartments", "number_of_floors", "Company_name") values ({}, {}, {},
{}, {})'.format(
                    id_building, adress, number_of_apartments, number_of_floors,
Company_name)

                flag = False
```

```
            elif table == 3:
                Company_name = "'" + input('Company_name = ') + "'"
                adress = "'" + input('adress = ') + "'"
                phone = "'" + input('phone = ') + "'"

                insert = 'insert into "Company" ("Company_name", "adress", "phone")
values ({}, {}, {})'.format(
                    Company_name, adress, phone)

                flag = False
            elif table == 4:
                id_person = input('id_person = ') + "'"
                name = "'" + input('name = ') + "'"
                surname = "'" + input('surname = ') + "'"
                patronymic = "'" + input('patronymic = ') + "'"
                phone = "'" + input('phone = ') + "'"
                Company_name = "'" + input('Company_name = ') + "'"

                insert = 'insert into "Person" ("id_person", "name", "surname",
"patronymic", "phone", "Company_name") values ({}, {}, {}, {}, {}, {})'.format(
                    id_person, name, surname, patronymic, phone, Company_name)

                flag = False
            else:
                print('\nWrong number, choose again.')
        print(tables[table])
        print('SQl query -> ', insert)
        cursor.execute(insert)
        connect.commit()
        print('Data added successfully!')
        cursor.close()
        Connect.closeConnect(connect)

    @staticmethod
    def delete():
        connect = Connect.makeConnect()
        cursor = connect.cursor()
        flag = True
        while flag:
            View.list()
            table = Model.CheckTable()
```

```
            if table == 1:
                id_apartment = input('Enter "id_apartment" to delete = ')
                delete = 'delete from "Apartment" where "id_apartment"=
{}'.format(id_apartment)
                flag = False
            elif table == 2:
                id_building = input('Enter "id_building" to delete = ')
                delete = 'delete from "Building" where "id_building"=
{}'.format(id_building)
                flag = False
            elif table == 3:
                Company_name = "'" + input('Enter "Company_name" to delete = ') + "'"
                delete = 'delete from "Company" where "Company_name"=
{}'.format(Company_name)
                flag = False
            elif table == 4:
                id_person = input('Enter "id_person" to delete = ')
                delete = 'delete from "Person" where "id_person"= {}'.format(id_person)
            else:
                print('\nWrong number, choose again.')
        print(tables[table])
        print("SQL query -> ", delete)
        cursor.execute(delete)
        connect.commit()
        print('Data deleted successfully!')
        cursor.close()
        Connect.closeConnect(connect)

    @staticmethod
    def update():
        connect = Connect.makeConnect()
        cursor = connect.cursor()
        flag = True
        while flag:
            View.list()
            table = Model.CheckTable()
            if table == 1:
                id_apartment = "'" + input('id_apartment of row to update = ') + "'"
                View.attribute_list(1)
                second_flag = True
```

```
            while second_flag:
                num = input('Number of attribute ->')
                value = """ + input('New value of attribute = ') + """
                if num == '1':
                    set = "'id_apartment"= {}'.format(value)
                    second_flag = False
                elif num == '2':
                    set = "'floor"= {}'.format(value)
                    second_flag = False
                elif num == '3':
                    set = "'number"= {}'.format(value)
                    second_flag = False
                elif num == '4':
                    set = "'number_of_rooms"= {}'.format(value)
                    second_flag = False
                elif num == '5':
                    set = "'id_person"= {}'.format(value)
                    second_flag = False
                elif num == '6':
                    set = "'id_building"= {}'.format(value)
                    second_flag = False
                else:
                    print('\nWrong number, choose again.')
            update = 'update "Apartment" set {} where "id_apartment"= {}'.format(set,
id_apartment)
            flag = False
            pass
        elif table == 2:
            id_building = """ + input('id_building of row to update = ') + """
            View.attribute_list(2)
            second_flag = True
            while second_flag:
                num = input('Number of attribute =>')
                value = """ + input('New value of attribute = ') + """
                if num == '1':
                    set = "'id_building"= {}'.format(value)
                    second_flag = False
                elif num == '2':
                    set = "'adress"= {}'.format(value)
                    second_flag = False
                elif num == '3':
```

```
                    set = '"number_of_apartments"= {}'.format(value)
                    second_flag = False
                elif num == '4':
                    set = '"number_of_floors"= {}'.format(value)
                    second_flag = False
                elif num == '5':
                    set = '"Company_name"= {}'.format(value)
                    second_flag = False
                else:
                    print('\nWrong number, choose again.')
            update = 'update "Building" set {} where "id_building"= {}'.format(set,
id_building)
            flag = False
            pass
        elif table == 3:
            Company_name = "'" + input('Company_name of row to update = ') + "'"
            View.attribute_list(3)
            second_flag = True
            while second_flag:
                num = input('Number of attribute =>')
                value = "'" + input('New value of attribute = ') + "'"
                if num == '1':
                    set = '"Company_name"= {}'.format(value)
                    second_flag = False
                elif num == '2':
                    set = '"adress"= {}'.format(value)
                    second_flag = False
                elif num == '3':
                    set = '"phone"= {}'.format(value)
                    second_flag = False
                else:
                    print('\nWrong number, choose again.')
            update = 'update "Company" set {} where "Company_name"=
{}'.format(set, Company_name)
            flag = False
            pass
        elif table == 4:
            id_person = input('id_person of row to update = ')
            View.attribute_list(4)
            second_flag = True
            while second_flag:
```

```
                num = input('Number of attribute =>')
                value = """ + input('New value of attribute = ') + """
                if num == '1':
                    set = "'id_person"= {}'.format(value)
                    second_flag = False
                elif num == '2':
                    set = "'name"= {}'.format(value)
                    second_flag = False
                elif num == '3':
                    set = "'surname"= {}'.format(value)
                    second_flag = False
                elif num == '4':
                    set = "'patronymic"= {}'.format(value)
                    second_flag = False
                elif num == '5':
                    set = "'phone"= {}'.format(value)
                    second_flag = False
                elif num == '6':
                    set = "'Company_name"= {}'.format(value)
                    second_flag = False
                else:
                    print('\nWrong number, choose again.')
            update = 'update "Person" set {} where "id_person"= {}'.format(set,
id_person)
            flag = False
            pass
        else:
            print('\nWrong number, choose again.')
    print(tables[table])
    print("SQL query -> ", update)
    cursor.execute(update)
    connect.commit()
    print('Data updeted successfully!')
    cursor.close()
    Connect.closeConnect(connect)
    pass

@staticmethod
def random():
    connect = Connect.makeConnect()
    cursor = connect.cursor()
```

```
        flag = True
        while flag:
            View.list()
            table = Model.CheckTable()
            num = input('How many rows to random? -> ')
            if num.isdigit():
                flag = False
                for i in range(int(num)):
                    random = ''
                    if table == 1:
                        random = 'insert into "Apartment" select (select max(id_apartment)+1
from
"Apartment"),ceil((RANDOM()*10)),ceil((RANDOM()*10)),ceil((RANDOM()*3)),(
select ceil(RANDOM()*max(id_person)) from "Person"),(select
ceil(RANDOM()*max(id_building)) from "Building")'
                    elif table == 2:
                        random = 'insert into "Building" select (select max(id_building)+1
from "Building"),array_to_string(ARRAY(select chr((97 +
round(random()*25))::integer) From generate_series(1,floor(RANDOM()*(25-
10)+10)::integer)),\' \'),ceil((RANDOM()*10)),ceil((RANDOM()*10)),(SELECT
"Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)'
                    elif table == 3:
                        random = 'insert into "Company" select
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()*(25-10)+10)::integer)),\'
\'),array_to_string(ARRAY(select chr((48 + round(random()*9))::integer) From
generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),\' \')'
                    elif table == 4:
                        random = 'insert into "Person" select (select max(id_Person)+1 from
"Person"),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()*(25-10)+10)::integer)),\'
\'),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((48 + round(random()*9))::integer) From
generate_series(1,floor(RANDOM()*(10-8)+8)::integer)),\' \'),(SELECT
"Company_name" FROM "Company" ORDER BY RANDOM() LIMIT 1)'
```

```
                print("SQL query -> ", random)
                cursor.execute(random)
                connect.commit()
            else:
                print('\nWrong number, choose again.')

        print('Data randomed successfully!')
        cursor.close()
        Connect.closeConnect(connect)

    #select * from public."Person" as one inner join public."Apartment" as two USING ("id_person") where 2<two.id_apartment and two.id_apartment<10 and one.Name LIKE 'Name2'
    #select * from public."Person" as one inner join public."Apartment" as two USING ("id_person") inner join public."Building" as three USING ("id_building") where 2<two.id_apartment and two.id_apartment<10 and one.Name LIKE 'Name2' and 3<three.number_of_floors and three.number_of_floors<10
    #select * from public."Person" as one inner join public."Company" as four USING("Company_name") inner join public."Apartment" as two USING ("id_person") inner join public."Building" as three USING ("id_building") where 2<two.id_apartment and two.id_apartment<10 and one.Name LIKE 'Name2' and 3<three.number_of_floors and three.number_of_floors<10 and four."Company_name" LIKE 'Comp name'
    @staticmethod
    def search():
        connect = Connect.makeConnect()
        cursor = connect.cursor()

        flag = True
        while flag:
            num_table = input('Enter numbers of tables for search from 2 to 4-> ')
            num = input('Enter numbers of atribute for search -> ')

            table1 = None
            table2 = None
            table3 = None
            table4 = None

            if num_table == '2':
                View.list()
                print('Choose number of first table.')
```

```
    table1 = Model.CheckTable()
    print('Choose number of second table.')
    table2 = Model.CheckTable()
elif num_table == '3':
    View.list()
    print('Choose number of first table.')
    table1 = Model.CheckTable()
    print('Choose number of second table.')
    table2 = Model.CheckTable()
    print('Choose number of third table.')
    table3 = Model.CheckTable()
elif num_table == '4':
    table1 = 4
    table2 = 3
    table3 = 2
    table4 = 1

if num_table == '2':
    print(tables[table1])
    View.attribute_list(table1)
    print(tables[table2])
    View.attribute_list(table2)
elif num_table == '3':
    print(tables[table1])
    View.attribute_list(table1)
    print(tables[table2])
    View.attribute_list(table2)
    print(tables[table3])
    View.attribute_list(table3)
elif num_table == '4':
    print(tables[table1])
    View.attribute_list(table1)
    print(tables[table2])
    View.attribute_list(table2)
    print(tables[table3])
    View.attribute_list(table3)
    print(tables[table4])
    View.attribute_list(table4)

key1 = None
key2 = None
```

```
        key3 = None

        if num_table == '2':
            key1 = input('Enter key1 -> ')
        elif num_table == '3':
            key1 = input('Enter key1 -> ')
            key2 = input('Enter key2 -> ')
        elif num_table == '4':
            key1 = input('Enter key1 -> ')
            key2 = input('Enter key2 -> ')
            key3 = input('Enter key3 -> ')
        search = ''
        if num_table == '2':
            search = 'select * from public."{}"'.format(tables[table1]) + ' as one inner
join public."{}"'.format(tables[table2]) + ' as two using({})'.format(key1) + ' where '
        elif num_table == '3':
            search = 'select * from public."{}"'.format(tables[table1]) + ' as one inner
join public."{}"'.format(tables[table2]) + ' as two using({})'.format(key1) + ' inner
join public."{}"'.format(tables[table3]) + ' as three using({})'.format(key2) + ' where '
        elif num_table == '4':
            search = 'select * from public."{}"'.format(tables[table1]) + ' as one inner
join public."{}"'.format(tables[table2]) + ' as two using({})'.format(key1)  + ' inner
join public."{}"'.format(tables[table3]) + ' as three using({})'.format(key2)  + ' inner
join public."{}"'.format(tables[table4]) + ' as four using({})'.format(key3) + ' where '

        for i in range(int(num)):
            if num_table == '2':
                print(tables[table1])
                View.attribute_list(table1)
                print(tables[table2])
                View.attribute_list(table2)
            elif num_table == '3':
                print(tables[table1])
                View.attribute_list(table1)
                print(tables[table2])
                View.attribute_list(table2)
                print(tables[table3])
                View.attribute_list(table3)
            elif num_table == '4':
                print(tables[table1])
                View.attribute_list(table1)
```

```
                print(tables[table2])
                View.attribute_list(table2)
                print(tables[table3])
                View.attribute_list(table3)
                print(tables[table4])
                View.attribute_list(table4)
            type_atribute = input('Choose type of atribute\n 1 -> numeric\n 2 -> string
')
            if type_atribute == '1':
                atribute = input('Enter name of first atribute in form table.artibute -> ')
                num1 = input('min = ')
                num2 = input('max = ')
                search += '{}<{} and {}<{}'.format(num1,atribute,atribute,num2)
            elif  type_atribute == '2':
                atribute = input('Enter name of first atribute in form table.artibute -> ')
                num1 = """ + input('Text = ') + """
                search += '{} LIKE {}'.format(atribute,num1)
            if i != (int(num)-1):
                search += ' and '

        start = time.time()
        print("SQL query -> ", search)
        end = time.time()
        print("Time: ",end - start)
        cursor.execute(search)
        printtable = from_db_cursor(cursor)
        records = cursor.fetchall()
        obj = View(1, records, printtable)
        obj.show()
        connect.commit()
        flag = False

    print('Data search successfully!')
    cursor.close()
    Connect.closeConnect(connect)
```

В даному модулі реалізуються всі запити до бази даних програми. Для кожного завдання є окремий метод. Метод `CheckTable` створений для того щоб дізнатсь з якою таблицею потрібно працювати. Метод `showOneTable` призначений для виводу однієї таблиці. Метод `showAllTables` призначений для виводу всіх таблиць. Метод `insert` призначений для додавання даних. Метод `delete` призначений для

видалення даних. Метод `update` призначений для редагування даних. Метод `random` призначений для отримання рандомізованих даних. Метод `search` призначений для пошуку даних.