

Міністерство освіти та науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет прикладної математики

Розрахунково-графічна робота
із дисципліни
“Бази даних та засоби управління”
Тема: «Створення додатку бази даних, орієнтованого
на взаємодію з СУБД PostgreSQL»

Виконав: Пилипенко Тимофій.

Студент групи КВ-32

Telegram: @NeLern

Київ 2025

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)

Репозиторій: <https://github.com/LernL/BD>;

Мал. 1.1. Діаграма сутність-зв'язок бази даних



В базі даних є 5 таблиці:

- Клієнт, який взаємодіє зі столиком й офіціантом. Він має такі атрибути: Номер телефону(pk), email й номер телефону офіціанта, який його обслуговує.

- Контакти клієнтів, які має email клієнта(pk) й його ім'я.
- Столик, який має номер столика(pk), к-ть стільців, форма стола, матеріал, з якого він зроблен.

- Офіціант має такі атрибути: ПІБ, номер телефону(pk).
- Бронювання має час бронювання, час кінця бронювання, номер стола, номер клієнта, кількість людей й id(pk) бронювання

Меню має 7 пунктів:

1. Waiter:

- 1.1 Add
- 1.2 List
- 1.3 Update
- 1.4 Delete
- 1.5 Back

2. Clients:

- 2.1 Add
- 2.2 List
- 2.3 Update
- 2.4 Delete
- 2.5 Back

3. Tables:

- 3.1 Add
- 3.2 List
- 3.3 Update
- 3.4 Delete
- 3.5 Back

4. Booking:

- 4.1 Add
- 4.2 List
- 4.3 Update
- 4.4 Delete
- 4.5 Back

5. Contacts:

- 5.1 Add

- 5.2 List
- 5.3 Update
- 5.4 Delete
- 5.5 Back
- 6. Analytics:
 - 6.1 By table/hour
 - 6.2 By waiter
 - 6.3 Show booking for table
 - 6.4 Exit
- 7. Exit

Мова програмування: Python;

Використані бібліотеки: math, psycorg2;

1. Дочірня таблиця Client для Waiter(атрибут phone_num_w):

Мал. 1.2. Додавання запису в таблицю Waiter

```
Waiters: 1) Add 2) List 3) Update 4) Delete 0) Back
> 2
-----
Employee number | Name
-----
111             | Sam
```

Мал. 1.3. Додавання запису в таблицю Client

```
Clients: 1) Add 2) List 3) Update 4) Delete 0) Back
> 2
-----
Phone number client | Email | Phone number waiter
-----
222                 | aaa@gmail.com | 111
1002161011          | gomev@bikqwwg.com | 292924517865
```

Мал. 1.4. Спроба видалення зв'язаного запису з таблиці Waiter

```
Waiters: 1) Add 2) List 3) Update 4) Delete 0) Back
> 4
Phone number: 111
Error: ПОМИЛКА: update або delete в таблиці "waiter" порушує обмеження зовнішнього ключа "client_phone_num_w_fkey3" таблиці "client"
DETAIL: На ключ (phone_num_w)=(111) все ще є посилання в таблиці "client".
```

Мал. 1.5. Запис в таблиці Client залишилась

```
> 2
```

Phone number client	Email	Phone number waiter
222	aaa@gmail.com	111
1002161011	gomev@bikqwwg.com	292924517865

Видалення запису з батьківської таблиці неможливе, оскільки в дочірній таблиці існують записи, що посилаються на нього через зовнішній ключ. Для цього зв'язку не встановлено жодної дії при видаленні (ON DELETE NO ACTION або ON DELETE RESTRICT), тому СУБД забороняє видалення, щоб не порушити цілісність даних.

Мал. 1.6. Запис в таблиці Client не існуючого запису з таблиці Waiter
НЕМОЖЛИВЕ

```
Clients: 1) Add 2) List 3) Update 4) Delete 0) Back
> 1
Phone number client: 444
Email: bbb@gmail.com
Phone number waiter: 333
Error: ПОМИЛКА: insert або update в таблиці "client" порушує обмеження зовнішнього ключа "client_phone_num_w_fkey3"
DETAIL: Ключ (phone_num_w)=(333) не присутній в таблиці "waiter".
```

2. Ілюстрації згенерованих таблиць й відповідних sql записів наведено далі:

Мал. 2.1. Згенерована таблиця waiter

Query: `SELECT * FROM public.waiter ORDER BY phone_num_w ASC`

	name	phone_num_w
1	Sam	111
2	Kwemasrl	1016408524
3	Fdlldhdy	1031875267
4	Juvkfahf	1033636464
5	Guryebpl	1042213880
6	Vaxneraf	1043778215
7	Dbugpkke	1045038202
8	Tetehler	1047532003
9	Rshfuply	1048337961

Total rows: 100001 Query complete 00:00:00.141

Sql запит:

```
BEGIN;
TRUNCATE TABLE public.waiter RESTART IDENTITY CASCADE;
INSERT INTO public.waiter (name, phone_num_w)
```

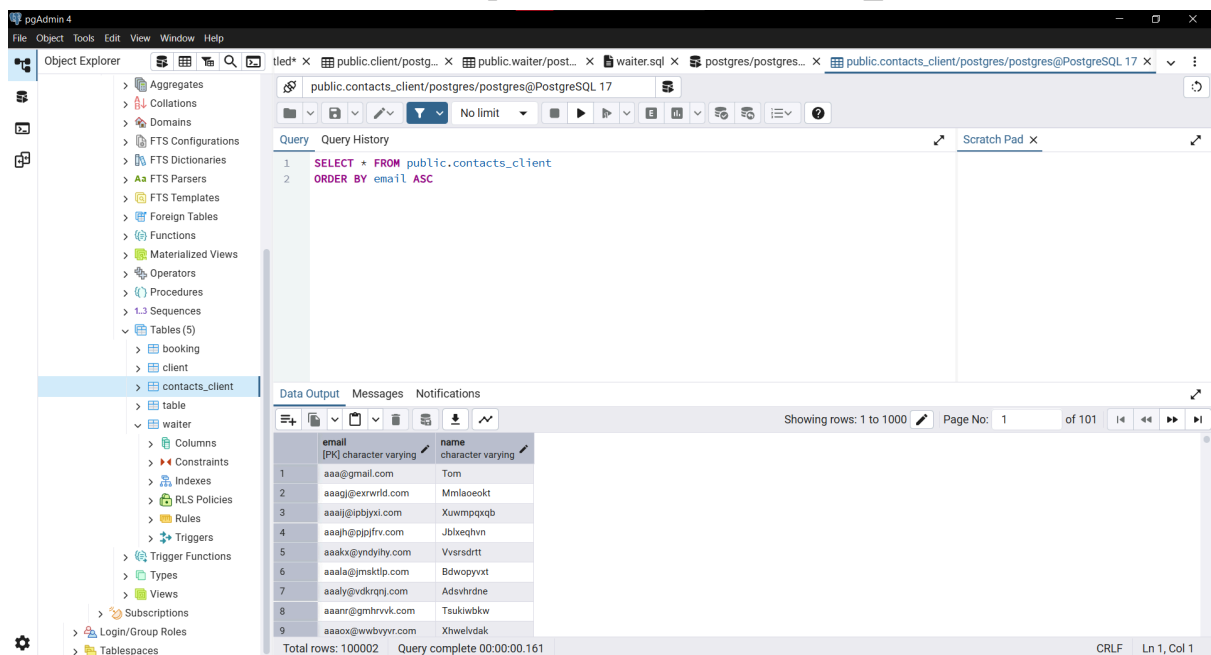
```

SELECT          chr(trunc(65+random()*25)::int)           ||
Lower(chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ) AS name,
      (random()*10000000000000)+1000000000::bigint
FROM generate_series(1,100000);

```

COMMIT;

Мал. 2.2. Згенерована таблиця contacts_client



Sql запит:

```

BEGIN;
TRUNCATE TABLE public.contacts_client RESTART IDENTITY CASCADE;
INSERT INTO public.contacts_client (email, name)
SELECT          Lower(          chr(trunc(65+random()*25)::int)           ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) || '@' ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || '.com' ) AS name,
      chr(trunc(65+random()*25)::int)           || Lower
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||

```

```
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) )
FROM generate_series(1,100000);
```

COMMIT;

Мал. 2.3. Згенерована таблиця client

	email character varying (30)	phone_num_cl (PK) bigint	phone_num_w bigint
1	aaa@gmail.com	222	111
2	gomev@bikqwwg.com	1002161011	292924517865
3	rwkdm@eqksjos.com	1002742588	253162632629
4	yrtks@wabwvta.com	1005664755	362492191356
5	kstvt@funmalh.com	1005665542	422423427089
6	hsygf@oocgtmm.com	1015926995	360901120613
7	kcdwh@qbhtffn.com	1025227999	977295459465
8	cvnco@cujgpto.com	1045211955	307311229075
9	wsht@mgoljug.com	1058358980	770882859544

Sql запит:

BEGIN;

TRUNCATE TABLE public.client RESTART IDENTITY CASCADE;

```
WITH emails AS (
  SELECT email, row_number() OVER () AS id
  FROM public.contacts_client
  ORDER BY random()
  LIMIT 100000
),
```

```
waiters AS (
  SELECT phone_num_w, row_number() OVER () AS id
  FROM public.waiter
  ORDER BY random()
  LIMIT 100000
)
```

```
INSERT INTO public.client (phone_num_w, email, phone_num_cl)
```

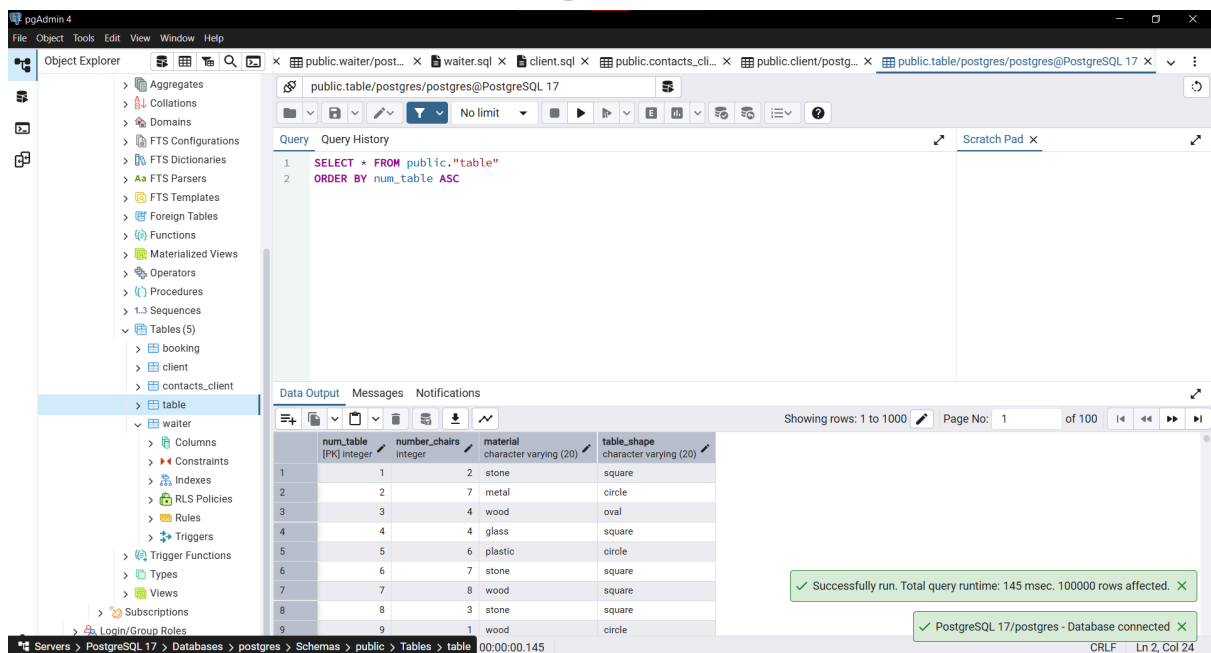
```

SELECT
    w.phone_num_w,
    e.email,
    (random() * 1000000000000 + 1000000000)::bigint
FROM emails e
JOIN waiters w USING (id);

COMMIT;

```

Мал. 2.4. Згенерована таблиця table



The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure, with the 'table' table selected under the 'public' schema. The right pane shows the query results for the query: `SELECT * FROM public."table" ORDER BY num_table ASC`. The results are displayed in a table with 4 columns: `num_table` (integer), `number_chairs` (integer), `material` (character varying (20)), and `table_shape` (character varying (20)). The table contains 9 rows of data. A status message at the bottom right indicates: "Successfully run. Total query runtime: 145 msec. 100000 rows affected."

num_table	number_chairs	material	table_shape
1	1	stone	square
2	2	metal	circle
3	3	wood	oval
4	4	glass	square
5	5	plastic	circle
6	6	stone	square
7	7	wood	square
8	8	stone	square
9	9	wood	circle

Sql запит:

```

BEGIN;

TRUNCATE TABLE public.table RESTART IDENTITY CASCADE;

WITH params AS (
    SELECT
        ARRAY['wood','metal','plastic','glass','stone'] AS materials,
        ARRAY['square','circle','rectangle','oval'] AS shapes
)

INSERT INTO public.table (num_table, number_chairs, material, table_shape)
SELECT
    gs AS num_table,
    (floor(random()*8) + 1)::int AS number_chairs,

```



```

params.materials[(floor(random()*array_length(params.materials,1)) + 1)::int]
AS material,
    params.shapes[(floor(random()*array_length(params.shapes,1)) + 1)::int]
AS table_shape
FROM generate_series(1, 100000) AS gs
CROSS JOIN params;

COMMIT;

```

Мал. 2.5. Згенерована таблиця booking

	time_book integer	end_time integer	num_table integer	phone_num_cl bigint	people_num integer	id [PK] integer	
1		4	5	24100	499152000918	5	1
2		16	17	99396	716125163114	1	2
3		24	24	47869	683444338163	4	3
4		8	9	24094	992766750537	8	4
5		10	11	95494	407182335224	8	5
6		12	13	41700	586612591211	1	6
7		7	8	87895	465241162057	5	7
8		6	7	68018	865950411614	4	8
9		15	16	96439	873945226802	7	9

Sql запит:

```

BEGIN;

TRUNCATE TABLE public.booking RESTART IDENTITY CASCADE;

WITH numtb AS (
    SELECT num_table, row_number() OVER (ORDER BY random()) AS rn
    FROM public."table"
    LIMIT 100000
),
numcl AS (
    SELECT phone_num_cl, row_number() OVER (ORDER BY random()) AS rn
    FROM public.client
    LIMIT 100000
)

```

```

INSERT INTO public.booking (id, time_book, end_time, num_table,
phone_num_cl, people_num)
SELECT
    row_number() OVER (ORDER BY random()) AS id,
    s.time_book,
    LEAST(24, s.time_book + (floor(random() * (24 - s.time_book))::int + 1)) AS
end_time,
    s.num_table,
    s.phone_num_cl,
    s.people_num
FROM (
    SELECT nt.num_table, nc.phone_num_cl,
        (floor(random()*24)::int + 1) AS time_book,
        (floor(random()*8)::int + 1) AS people_num,
        nt.rn
    FROM numtb nt
    JOIN numcl nc USING (rn)
) s;

COMMIT;

```

3. Ілюстрації запитів й відповідних sql записів наведено далі:

Мал. 3.1 Пошуковий запит бронювання з 1 до 5 години

```

Analytics: 1) By table/hour 2) By waiter 3) Show bookings for table 0) Back
> 1
people_num min (enter to skip):
people_num max (enter to skip):
hour from (0-23, enter to skip): 1
hour to (0-23, enter to skip): 5
num_table (enter to skip):

```

num_table	table_shape	material	start_hour	end_hour	avg_people
53	oval	wood	1	4	1
117	square	stone	1	5	3
139	square	stone	1	2	4
210	oval	glass	1	5	2
459	oval	metal	1	5	4
514	circle	glass	1	4	4
570	oval	stone	1	4	4
633	oval	plastic	1	4	4
819	square	stone	1	3	5
841	rectangle	stone	1	2	2
1233	rectangle	wood	1	2	2
1420	rectangle	stone	1	5	6
1453	rectangle	metal	1	3	7
1525	oval	plastic	1	3	7
1542	oval	plastic	1	4	7
1570	square	plastic	1	3	7
1785	rectangle	glass	1	4	8

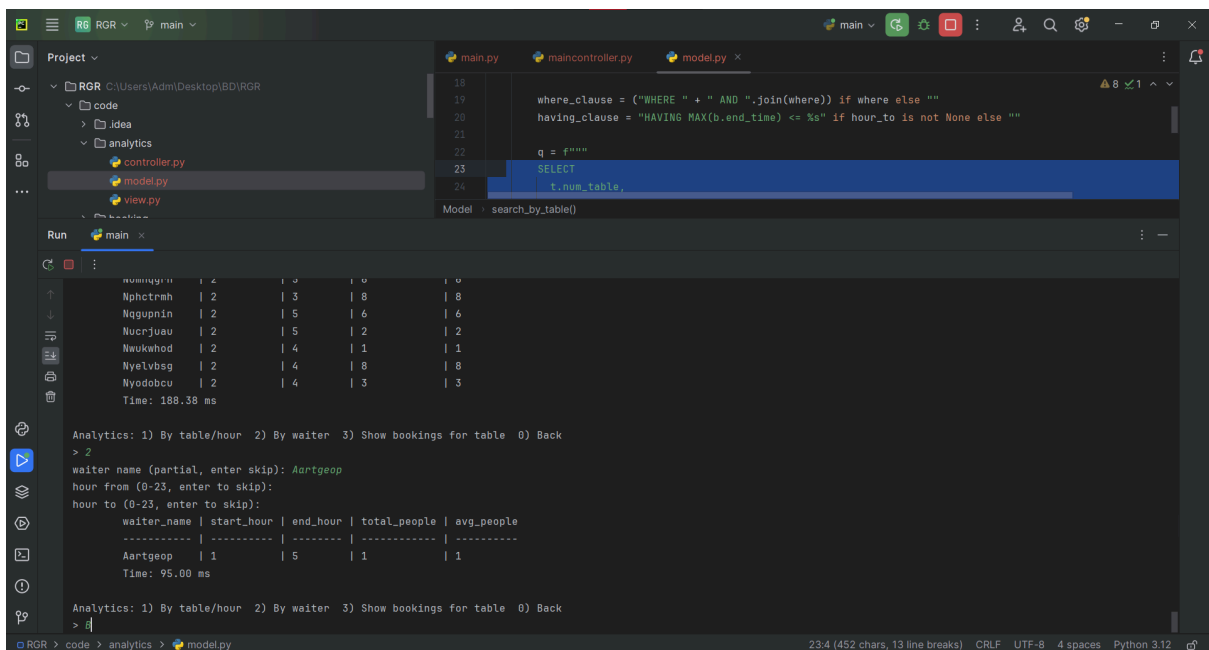
Sql запит:

```

SELECT
    t.num_table,
    t.table_shape,
    t.material,
    MIN(b.time_book) AS start_hour,
    MAX(b.end_time) AS end_hour,
    ROUND(AVG(b.people_num))::int AS avg_people
FROM booking b
JOIN "table" t ON b.num_table = t.num_table
{where_clause}
GROUP BY t.num_table, t.table_shape, t.material
{having_clause}
ORDER BY start_hour, t.num_table
LIMIT %s

```

Мал. 3.2. Пошуковий запит по імені офіціанта



Sql запит:

```

SELECT
    w.name AS waiter_name,
    MIN(b.time_book) AS start_hour,
    MAX(b.end_time) AS end_hour,
    SUM(b.people_num) AS total_people,
    ROUND(AVG(b.people_num))::int AS avg_people
FROM booking b
JOIN "table" t ON b.num_table = t.num_table

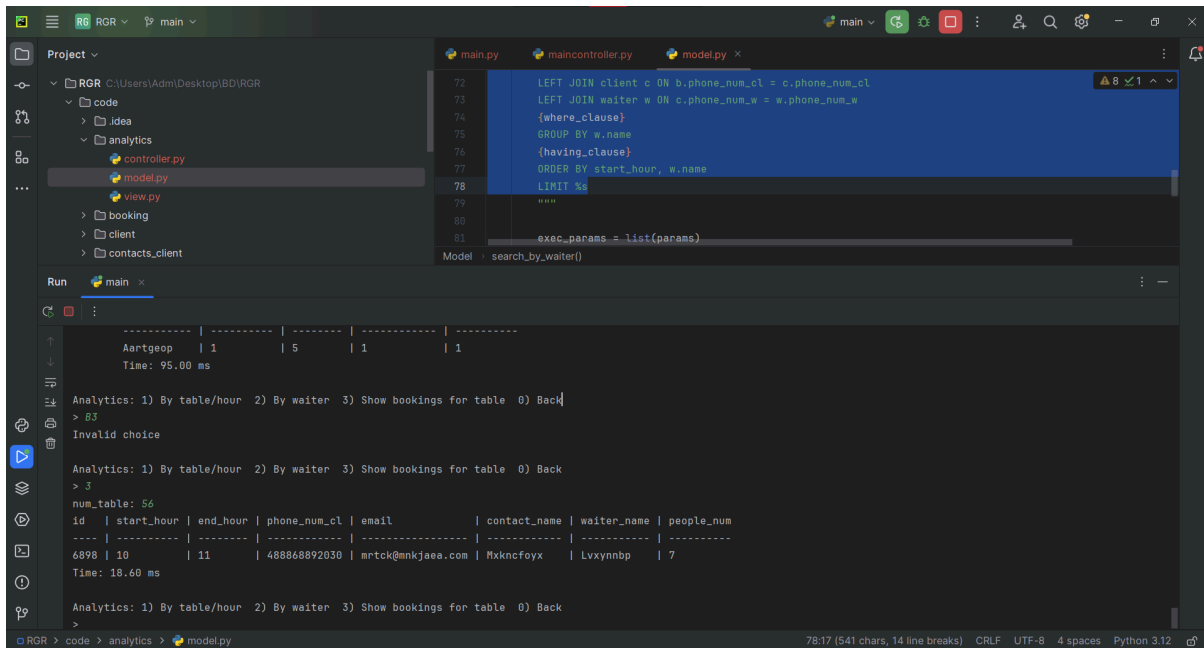
```

```

LEFT JOIN client c ON b.phone_num_cl = c.phone_num_cl
LEFT JOIN waiter w ON c.phone_num_w = w.phone_num_w
{where_clause}
GROUP BY w.name
{having_clause}
ORDER BY start_hour, w.name
LIMIT %s

```

Мал. 3.3. Пошуковий запит по номеру стола



Sql запит:

```

SELECT
b.id,
b.time_book AS start_hour,
b.end_time AS end_hour,
b.phone_num_cl,
c.email,
co.name AS contact_name,
w.name AS waiter_name,
b.people_num
FROM booking b
LEFT JOIN client c ON b.phone_num_cl = c.phone_num_cl
LEFT JOIN contacts_client co ON c.email = co.email
LEFT JOIN waiter w ON c.phone_num_w = w.phone_num_w
WHERE b.num_table = %s
ORDER BY b.time_book, b.id

```

LIMIT %s

4. Ілюстрація програмного коду модуля “Model” й його опис:

```
import psycopg2
```

```
class Model:
```

```
    def __init__(self):
```

```
        self.conn = psycopg2.connect("dbname=postgres user=postgres  
password=1234adminL host=localhost port=5432")
```

```
        self.conn.autocommit = True
```

```
        self.create_table()
```

```
    def create_table(self):
```

```
        with self.conn.cursor() as c:
```

```
            c.execute("""
```

```
                CREATE TABLE IF NOT EXISTS waiter (
```

```
                    name varchar(20) NOT NULL,
```

```
                    phone_num_w integer NOT NULL PRIMARY KEY,
```

```
                    UNIQUE (name)
```

```
                );
```

```
            """)
```

```
        self.conn.commit()
```

```
    def add(self, phone_num_w, name):
```

```
        with self.conn.cursor() as c:
```

```
            c.execute("INSERT INTO waiter (phone_num_w, name) VALUES  
(%s,%s)", (int(phone_num_w), name))
```

```
        self.conn.commit()
```

```
    def list_all(self):
```

```
        with self.conn.cursor() as c:
```

```
            c.execute("SELECT phone_num_w, name FROM waiter ORDER BY  
phone_num_w")
```

```
            return c.fetchall()
```

```
    def update(self, phone_num_w, new_name):
```

```
        with self.conn.cursor() as c:
```

```
            c.execute(
```

```

        "UPDATE waiter SET name=%s WHERE phone_num_w=%s",
        (new_name, int(phone_num_w))
    )
    self.conn.commit()

def delete(self, phone_num_w):
    with self.conn.cursor() as c:
        c.execute("DELETE FROM waiter WHERE phone_num_w=%s",
        (int(phone_num_w),))
    self.conn.commit()

def get_input(self):
    name = input("Name: ").strip()
    return name

def get_key(self):
    return input("Phone number: ").strip()

```

Короткий опис:

`__init__(self)`

Ініціалізує підключення до бази даних PostgreSQL та викликає метод створення таблиці.

`create_table(self)`

Створює таблицю, якщо вона ще не існує.

`add(self, ...)`

Додає новий запис у таблицю з указаними параметрами.

`list_all(self)`

Повертає всі записи з таблиці.

`update(self)`

Оновлює запис таблиці.

`delete(self, phone_num_w)`

Видаляє запис із таблиці.

`get_input(self)`

Зчитує з клавіатури значення для додавання або оновлення запису.

`get_key(self)`

Зчитує з клавіатури значення ключового поля.