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-03
Стецюренко І. С.
Перевірив:
Петрашенко А. В.

Київ-2022

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-поданняконтролер).

Деталізоване завдання:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати вилучення рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні внесення нових даних. Унеможливити виведення програмою системних помилок на екрані

шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими не мовою програмування, а відповідним SQL-запитом!
3. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
4. Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати лише мову SQL (без ORM).

URL репозиторію з вихідним кодом

Модель “сутність - зв’язок” галузі “доставка продуктів”

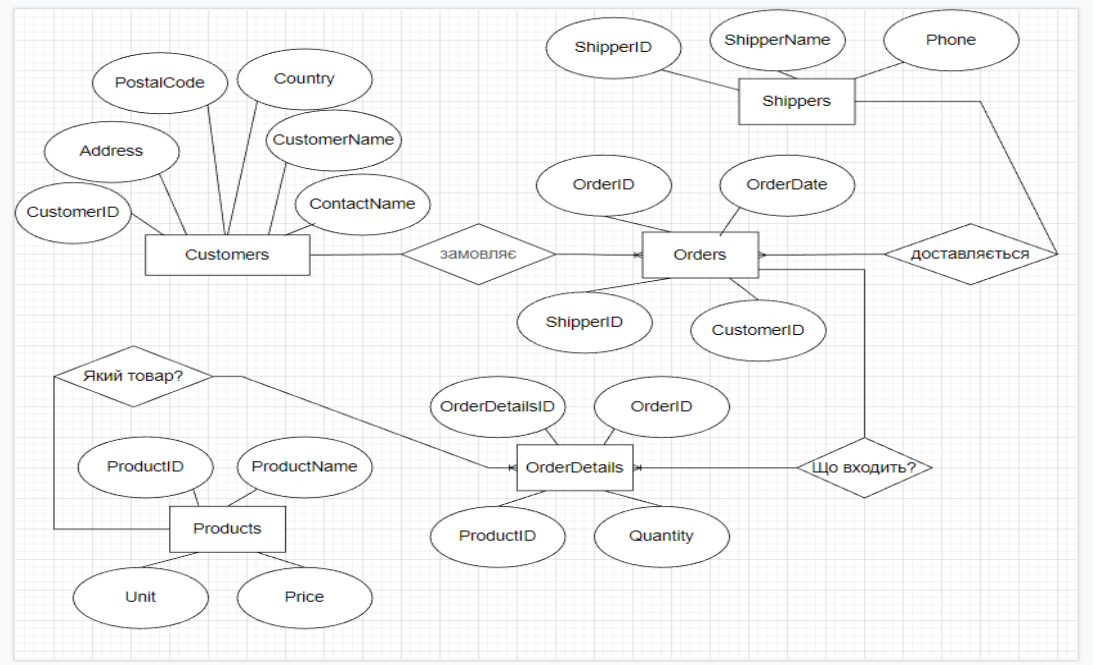


Рисунок 1. ER-діаграма, побудована за нотацією Чена

1. **Опис зв’язків**

При проектуванні бази даних “Доставка товарів” можна виділити такі сутності(Customers, Orders, OrderDetails, Shippers, Products,)

Клієнти (Customers) можуть робити безліч замовлень(Orders), тому між таблицями встановлено зв’язок 1:N.

Певну кількість замовлень(Orders) буде доставляти одна з компаній-перевізників(Shippers), тому між таблицями Shippers та Orders встановлено зв’язок 1:N.

Для реалізації зв’язку багато до багатьох (N:M) зумовлена поява таблиці OrderDetails, так ми зможем дізнатись інформацію, які саме товари замовив клієнт, за номером замовлення(OrderID) та в якій кількості(Quantity).

Кожне замовлення має деяку кількість товарів в собі, тому між сутностями Orders та OrderDetails був встановлений зв’язок 1:N.

OrderDetails буде включати в себе товари із списку Products, тому між сутностями Products та OrderDetails встановлений зв’язок 1:N.

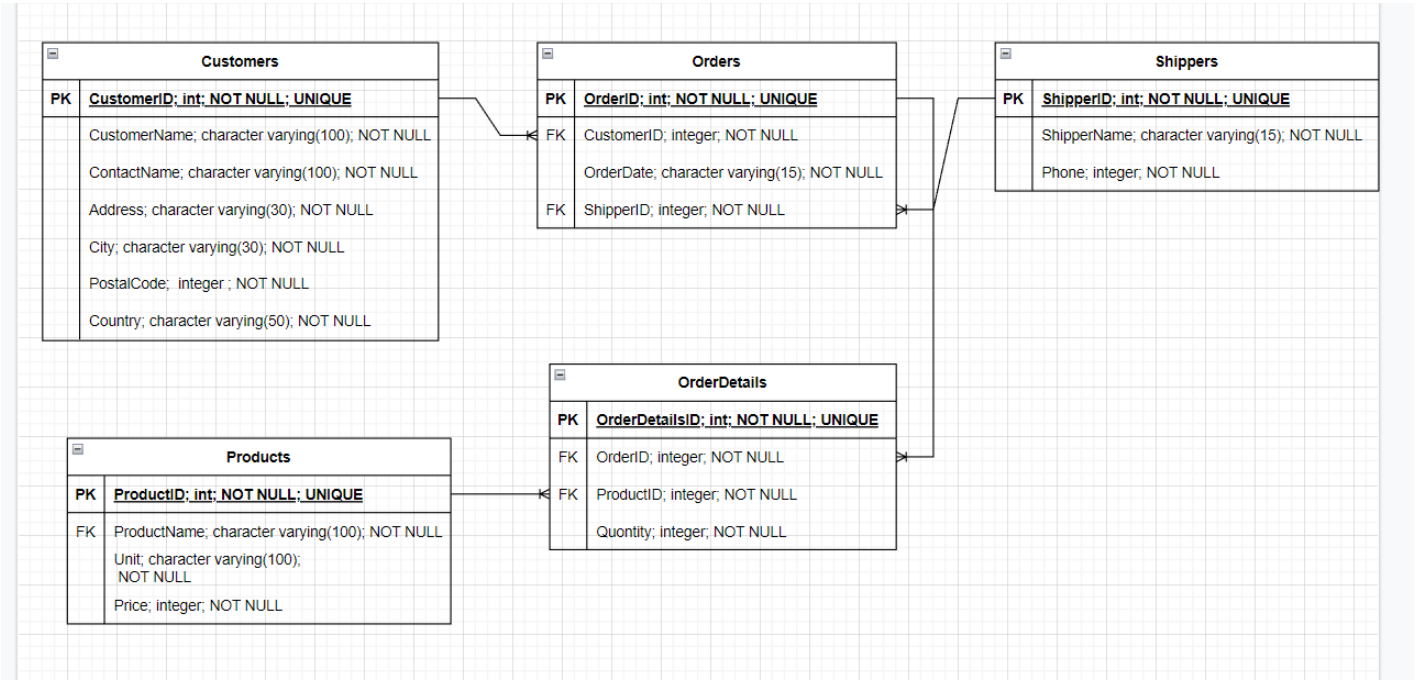


Рисунок 2 - ER-діаграма, переведена у таблиці БД

Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.

Мова програмування – Python 3.10.4

Середовище розробки програмного забезпечення – PyCharm Community Edition.

Структура меню підпрограми

```
Welcome!

                                                    Main menu
                                                    1. Show one table
                                                    2. Show all table
                                                    3. Insert data
                                                    4. Delete data
                                                    5. Update date
                                                    6. Select data
                                                    7. Randomize data
                                                    8. Exit

Make your choice =>
```

Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

Перегляд однієї таблиці

```
Make your choice => 1
PostgreSQL connect
                                    1. Shippers
                                    2. Orders
                                    3. Customers
                                    4. OrderDetails
                                    5. Products
Make your choice => 2
Orders
SQL query: "SELECT * FROM "Orders" ;
+---------+------------+------------+-----------+
| OrderID | CustomerID | OrderDate  | ShipperID |
+---------+------------+------------+-----------+
|    1    |     1      | 24.07.2022 |     3     |
|    2    |     2      | 10.08.2022 |     3     |
|    3    |     3      | 15.08.2022 |     3     |
|    4    |     4      | 19.08.2022 |     2     |
|    5    |     5      | 27.08.2022 |     1     |
|    6    |     3      | 31.08.2022 |     2     |
|    7    |     2      | 5.09.2022  |     3     |
|    8    |     4      | 12.09.2022 |     1     |
+---------+------------+------------+-----------+

PostgreSQL connection is closed
Continue work with DB? Y/N =>
```

```
Make your choice => 1
PostgreSQL connect
                                    1. Shippers
                                    2. Orders
                                    3. Customers
                                    4. OrderDetails
                                    5. Products
Make your choice => 1
Shippers
SQL query: "SELECT * FROM "Shippers" ;
+-----------+-------------+---------------+
| ShipperID | ShipperName |     Phone     |
+-----------+-------------+---------------+
|     1     | Nova Poshta | +380984500609 |
|     2     |  Ukrposhta  |   0800300545  |
|     3     |    Meest    | +380504327707 |
+-----------+-------------+---------------+

PostgreSQL connection is closed
Continue work with DB? Y/N =>
```

Перегляд всіх таблиць

```
Make your choice => 2
PostgreSQL connect
Shippers
SQL query: "SELECT * FROM "Shippers" ;
+-----------+-------------+---------------+
| ShipperID | ShipperName |     Phone     |
+-----------+-------------+---------------+
|     1     | Nova Poshta | +380984500609 |
|     2     |  Ukrposhta  |   0800300545  |
|     3     |    Meest    | +380504327707 |
+-----------+-------------+---------------+

Orders
SQL query: "SELECT * FROM "Orders" ;
+---------+------------+------------+-----------+
| OrderID | CustomerID | OrderDate  | ShipperID |
+---------+------------+------------+-----------+
|    1    |     1      | 24.07.2022 |     3     |
|    2    |     2      | 10.08.2022 |     3     |
|    3    |     3      | 15.08.2022 |     3     |
|    4    |     4      | 19.08.2022 |     2     |
|    5    |     5      | 27.08.2022 |     1     |
|    6    |     3      | 31.08.2022 |     2     |
|    7    |     2      | 5.09.2022  |     3     |
|    8    |     4      | 12.09.2022 |     1     |
+---------+------------+------------+-----------+
```

```
Customers
SQL query: "SELECT * FROM "Customers" ;
+------------+--------------------+---------------+----------------------------+--------+------------+---------+
| CustomerID |    CustomerName    |  ContactName  |          Address           |  City  | PostalCode | Country |
+------------+--------------------+---------------+----------------------------+--------+------------+---------+
|     1      | Alfreds Futterkiste |  Maria Anders  |       Obere Street 57       | Berlin |   12209    | Germany |
|     2      |   Du monde entier   | Janine Labrune | 67, rue des Cinquante Otages | Nantes |   44000    |  Frence |
|     3      |    Franchi S.p.A.   | Paolo Accorti  |     Via Monte Bianco 34     | Torino |   10100    |  Italy  |
|     4      |  Poshchyvalov Oleh  |  Larin Dmytro  |      Soborna Street 77a     |  Kyiv  |    8131    | Ukraine |
|     5      |    Dmytro Lyashuk   | Pavlenko Roman |     Korsunska Street 16     |  Lviv  |   79022    | Ukraine |
+------------+--------------------+---------------+----------------------------+--------+------------+---------+
```

OrderDetails

SQL query: "SELECT * FROM "OrderDetails" ;

OrderDetailID	OrderID	ProductID	Quantity
1	1	7	1
2	1	12	3
3	1	16	5
4	1	20	1
5	2	4	3
6	2	8	2
7	2	10	1
8	2	21	1
9	2	17	2
10	3	24	1
11	3	23	1
12	4	2	1
13	4	13	5
14	4	14	2
15	4	15	1
16	5	8	2
17	5	9	3
18	5	19	5
19	5	25	1
20	6	2	2
21	6	3	4
22	6	5	2
23	6	7	1
24	7	16	4
25	7	17	3
26	7	18	4
27	7	4	3
28	8	12	5
29	8	13	3
30	8	26	2

```
Products
SQL query: "SELECT * FROM "Products" ;
```

ProductID	ProductName	Unit	Price
1	Coca-Cola 2l	2l * 6 - 35	6
2	Coca-Cola ZERO 1.5l	1.5l * 6 - 22	4.7
3	Coca-Cola 1.5l	1.5l - 47	0.78
4	Coca-Cola ZERO 1.5l	1.5l - 47	0.82
5	Package of Morshynska mineral water 1.5l * 6	1.5l * 6 - 63	3.28
6	Morshynska mineral water 1.5l	1.5l - 156	0.55
7	Sprite 1.5l x 6	1.5l * 6 - 24	4.36
8	Coffee beans Jacobs Monarch 1 kg	67	15
9	Mountain Ceylon Pekoe black tea 1 kg	15	21.65
10	Saucep green tea 1 kg	23	26.4
11	Nestle Nesquik with whole grains and B vitamins 460 g	58	2.95
12	Nestle Lion Flakes 450 g	58	2.98
13	Nestle Fitness with vitamins and minerals 425 g	58	4.2
14	Art Foods Basmati rice 1 kg	63	4.3
15	Pere Bulgur groats 800 g	72	2.95
16	Chips Pringles Cheese Cheese 190 g	58	3.57
17	Winway roasted cashews 100 g	12	3.09
18	Winway roasted pistachios 80 g	36	2.41
19	Millennium dark chocolate with whole hazelnuts 1.1 kg	26	14.97
20	Napoleon cake 1 kg	10	15.61
21	Cake Snickers 1 kg	7	16.21
22	Trout fillet 1 kg	5	27.07
23	Freshly frozen mackerel 1 kg	12	5.14
24	Salmon steak in vacuum packaging 1 kg	3	22.19
25	Elovena galettes with gluten-free syrup 160 g	54	3.11
26	Milk 2.5% VILLA MILK 1 l	30	1.1

Внесення даних

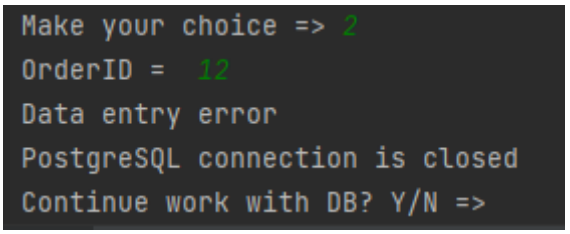
```
☑PostgreSQL connect
                            1. Shippers
                            2. Orders
                            3. Customers
                            4. OrderDetails
                            5. Products
Make your choice => 2
OrderID =  9
CustomerID =  2
OrderDate =  27.10.2022
ShipperID =  1
Orders
 SQL query: INSERT INTO "Orders" ( "OrderID"  , "CustomerID"  , "OrderDate"  , "ShipperID"  ) VALUES ( '9'  , '2'  , '27.10.2022'  , '1' );
Inserted successfully!!
PostgreSQL connection is closed
Continue work with DB? Y/N =>
```

Перевіримо чи вставились дані:

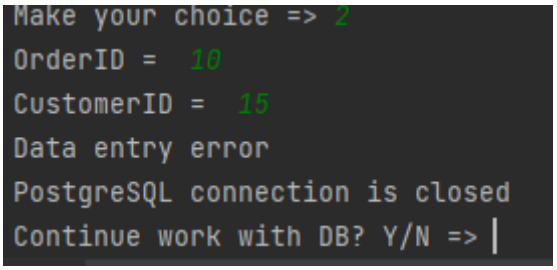
```
Make your choice => 1
PostgreSQL connect
                                1. Shippers
                                2. Orders
                                3. Customers
                                4. OrderDetails
                                5. Products
Make your choice => 2
Orders
SQL query: "SELECT * FROM "Orders" ;
+---------+------------+------------+-----------+
| OrderID | CustomerID | OrderDate  | ShipperID |
+---------+------------+------------+-----------+
|    1    |     1      | 24.07.2022 |     3     |
|    2    |     2      | 10.08.2022 |     3     |
|    3    |     3      | 15.08.2022 |     3     |
|    4    |     4      | 19.08.2022 |     2     |
|    5    |     5      | 27.08.2022 |     1     |
|    6    |     3      | 31.08.2022 |     2     |
|    7    |     2      | 5.09.2022  |     3     |
|    8    |     4      | 12.09.2022 |     1     |
|    9    |     2      | 27.10.2022 |     1     |
+---------+------------+------------+-----------+

PostgreSQL connection is closed
```

	OrderID [PK] integer	CustomerID integer	OrderDate character varying (15)	ShipperID integer
1	1	1	24.07.2022	3
2	2	2	10.08.2022	3
3	3	3	15.08.2022	3
4	4	4	19.08.2022	2
5	5	5	27.08.2022	1
6	6	3	31.08.2022	2
7	7	2	5.09.2022	3
8	8	4	12.09.2022	1
9	9	2	27.10.2022	1

Якщо було введено неправильний первинний ключ, відразу отримуємо помилку та процес зупиняються.



Якщо було введено не існуючий вторинний ключ, також отримуємо помилку та процес зупиняється.



Лістинг операції insert:

```
def menu_3():

    os.system('cls||clear')

    con = conect()

    title = title_table(con)

    show_mas(title)

    print("Make your choice =>", end=' ')

    while 1:

        number = input()

        num = int(number)


        if num > len(title) or num < 1:

            print('Incorrect number entered, please enter again =>', end=' ')

        else:

            break
```

```
    headers_selected_table = table_headlines(con, title[num - 1])
    values = []
    for i in range(len(headers_selected_table)):
        error = 0
        print(headers_selected_table[i] + " = ", end=" ")
        value = input()
        if i == 0:
            value0 = int(last_value_pk(con, title[num - 1],
headers_selected_table[i]))
            value0 = value0 + 1
            if int(value) != value0:
                error = 1
        if fkey(con, title[num - 1], headers_selected_table[i]) == 1:
            if find_value(con, headers_selected_table[i], value, title[num -
1]) == 0:
                error = 1

        if error == 1:
            print("Data entry error")
            close(con)
            return q_exit()
        else:
            values.insert(i, value)
    insert(headers_selected_table, con, title, num, values)
    con.commit()
    close(con)
    return q_exit()
```

Редагування даних

Таблиця до:

	OrderID [PK] integer	CustomerID integer	OrderDate character varying (15)	ShipperID integer
1	1	1	24.07.2022	3
2	2	2	10.08.2022	3
3	3	3	15.08.2022	3
4	4	4	19.08.2022	2
5	5	5	27.08.2022	1
6	6	3	31.08.2022	2
7	7	2	5.09.2022	3
8	8	4	12.09.2022	1
9	9	2	27.10.2022	1

```
Make your choice => 5
PostgreSQL connect
                                    1. Shippers
                                    2. Orders
                                    3. Customers
                                    4. OrderDetails
                                    5. Products
Make your choice => 2
Enter the number OrderID, which you want to change => 9

                                     1. CustomerID
                                     2. OrderDate
                                     3. ShipperID
Make your choice => 2
New value of OrderDate => 12.12.2022
Orders
 SQL query: UPDATE "Orders" SET "OrderDate" = '12.12.2022'
         WHERE "OrderID" = '9';
 Data updated successfully!!
PostgreSQL connection is closed
Continue work with DB? Y/N =>
```

Таблиця після:

	OrderID [PK] integer	CustomerID integer	OrderDate character varying (15)	ShipperID integer
1	1	1	24.07.2022	3
2	2	2	10.08.2022	3
3	3	3	15.08.2022	3
4	4	4	19.08.2022	2
5	5	5	27.08.2022	1
6	6	3	31.08.2022	2
7	7	2	5.09.2022	3
8	8	4	12.09.2022	1
9	9	2	12.12.2022	1

Якщо ввести неіснуючий первинний ключ, то програма про це повідомляє і просить ввести існуюче значення. Аналогічно буде й з вторинним ключем.

```
Make your choice => 5
PostgreSQL connect
                                1. Shippers
                                2. Orders
                                3. Customers
                                4. OrderDetails
                                5. Products
Make your choice => 2
Enter the number OrderID, which you want to change => 10
Incorrect number entered, please enter again => 9

                                1. CustomerID
                                2. OrderDate
                                3. ShipperID
Make your choice => 1
New value of CustomerID => 19
Incorrect value entered, please enter again =>
```

Лістинг операції update:

```
def menu_5():

    os.system('cls||clear')

    con = conect()

    title = title_table(con)

    show_mas(title)

    print("Make your choice =>", end=' ')

    while 1:

        number = input()

        num = int(number)



        if num > len(title) or num < 1:

            print('Incorrect number entered, please enter again =>', end=' ')

        else:

            break

    headers_selected_table = table_headlines(con, title[num - 1])

    print(f"Enter the number {headers_selected_table[0]}, which you want to
change =>", end=' ')

    while 1:



        num_id = input()

        num_id = int(num_id)



        if num_id > int(last_value_pk(con, title[num - 1],
headers_selected_table[0])) or num_id < 1:

            print('Incorrect number entered, please enter again =>', end=' ')

        else:

            break

    print()

    width = shutil.get_terminal_size().columns
```

```
    position = (width - max(map(len, headers_selected_table))) // 2
    for i in range(1, len(headers_selected_table)):  # left justtified
        print(' ' * position + f'{i}. ' + headers_selected_table[i])
    print("Make your choice =>", end=' ')
    while 1:
        num_item = input()
        num_item = int(num_item)

        if num_item > (len(headers_selected_table) - 1) or num_item < 1:
            print('Incorrect number entered, please enter again =>', end=' ')
        else:
            break
    print(f"New value of {headers_selected_table[num_item]} =>", end=' ')
    flag = 1
    while flag:
        value = input()
        if fkey(con, title[num - 1], headers_selected_table[num_item]) == 1:
            if find_value(con, headers_selected_table[num_item], value,
title[num - 1]) == 0:
                print('Incorrect value entered, please enter again =>', end='
')
            else:
                flag = 0
        else:
            flag = 0
    update(con, title, num,  headers_selected_table, num_item, num_id, value)
    con.commit()
    close(con)
    return q_exit()
```

Вилучення даних

```
Make your choice => 4
PostgreSQL connect
                                    1. Shippers
                                    2. Orders
                                    3. Customers
                                    4. OrderDetails
                                    5. Products
Make your choice => 2
Enter the number OrderID, which you want to delete => 9
You really want to remove the line where OrderID = 9? Y/N => Y
Orders
 SQL query: DELETE FROM Orders WHERE "OrderID" = '9';
 Deleted successfully!!
PostgreSQL connection is closed
Continue work with DB? Y/N =>
```

Результат видалення

	OrderID [PK] integer	CustomerID integer	OrderDate character varying (15)	ShipperID integer
1	1	1	24.07.2022	3
2	2	2	10.08.2022	3
3	3	3	15.08.2022	3
4	4	4	19.08.2022	2
5	5	5	27.08.2022	1
6	6	3	31.08.2022	2
7	7	2	5.09.2022	3
8	8	4	12.09.2022	1

Лістинг операції delete:

```
def menu_4():

    os.system('cls||clear')

    con = conect()

    title = title_table(con)
```

```
    show_mas(title)

    print("Make your choice =>", end=' ')

    while 1:

        number = input()

        num = int(number)


        if num > len(title) or num < 1:

            print('Incorrect number entered, please enter again =>', end=' ')

        else:

            break

    headers_selected_table = table_headlines(con, title[num - 1])

    print(f"Enter the number {headers_selected_table[0]}, which you want to
delete =>", end=' ')

    while 1:



        num_id = input()

        num_id = int(num_id)



        if 0 > num_id > last_value_pk(con, title[num - 1],
headers_selected_table[0]):

            print('Incorrect number entered, please enter again =>', end=' ')

        else:

            break

    print(f"You really want to remove the line where
{headers_selected_table[0]} = {num_id}? Y/N =>", end=' ')

    while 1:

        c = input()

        if c == 'Y' or c == 'y':

            delete(con, title, num, headers_selected_table, num_id)

            con.commit()
```

```
            close(con)

            return q_exit()

        elif c == 'N' or c == 'n':

            close(con)

            return q_exit()

        else:

            print("The wrong value was entered, it will be entered again. Y/N
=>", end=' ')
```

Завдання 2

Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

	ShipperID [PK] integer	ShipperName character varying (100)	Phone character varying (15)
1	1	Nova Poshta	+380984500609
2	2	Ukrposhta	0800300545
3	3	Meest	+380504327707
4	130	R	O
5	685	J	J
6	837	V	B
7	835	M	Y
8	142	J	H
9	918	F	F
10	748	K	C
11	524	H	D
12	786	V	E
13	669	M	K
14	62	I	G
15	672	U	D
16	347	L	W
17	860	G	J
18	779	V	V
19	180	E	A
20	714	E	N
21	894	J	H

```
INSERT INTO "{title[num - 1]}" ({columns}) SELECT {DATA} FROM
GENERATE_SERIES(1, {number}) ;
```

Кількість псевдовипадкових рядків визначається також за допомогою псевдогенератору.

Завдання 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.

```
Make your choice => 6
☑PostgreSQL connect
1. Information about the order by the name of the customer (you can write only the first name, only the last name or first name and last name).
2. What goods were sent via _ ? Enter the name of the shipper?
3. Calculate the cost of the order.
Make your choice => 1

Enter first name or last name, or both first and last name => Poshchyvalov Oleh
+-------------------+---------+-----------+------------+-----------------------------------------+----------+
|    CustomerName   | OrderID | OrderDate | ShipperName |                ProductName              | Quantity |
+-------------------+---------+-----------+------------+-----------------------------------------+----------+
| Poshchyvalov Oleh |    4    | 19.08.2022 |  Ukrposhta  |            Coca-Cola ZERO 1.5l          |    1     |
| Poshchyvalov Oleh |    8    | 12.09.2022 | Nova Poshta |          Nestle Lion Flakes 450 g       |    5     |
| Poshchyvalov Oleh |    8    | 12.09.2022 | Nova Poshta | Nestle Fitness with vitamins and minerals 425 g |    3     |
| Poshchyvalov Oleh |    4    | 19.08.2022 |  Ukrposhta  | Nestle Fitness with vitamins and minerals 425 g |    5     |
| Poshchyvalov Oleh |    4    | 19.08.2022 |  Ukrposhta  |         Art Foods Basmati rice 1 kg     |    2     |
| Poshchyvalov Oleh |    4    | 19.08.2022 |  Ukrposhta  |           Pere Bulgur groats 800 g      |    1     |
| Poshchyvalov Oleh |    8    | 12.09.2022 | Nova Poshta |           Milk 2.5% VILLA MILK 1 l      |    2     |
+-------------------+---------+-----------+------------+-----------------------------------------+----------+

Time of request = 16 ms
PostgreSQL connection is closed
Data selected successfully!
Continue work with DB? Y/N =>
```

```
Make your choice => 6
☑PostgreSQL connect
1. Information about the order by the name of the customer (you can write only the first name, only the last name or first name and last name).
2. What goods were sent via _ ? Enter the name of the shipper?
3. Calculate the cost of the order.
Make your choice => 2

Enter the name of the shipper => Nova Poshta
+---------+-------------+-----------------------------------------------------+----------+
| OrderID | ShipperName |                    ProductName                      | Quantity |
+---------+-------------+-----------------------------------------------------+----------+
|    5    | Nova Poshta |           Coffee beans Jacobs Monarch 1 kg          |    2     |
|    5    | Nova Poshta |         Mountain Ceylon Pekoe black tea 1 kg        |    3     |
|    5    | Nova Poshta | Millennium dark chocolate with whole hazelnuts 1.1 kg |    5     |
|    5    | Nova Poshta |     Elovena galettes with gluten-free syrup 160 g    |    1     |
|    8    | Nova Poshta |              Nestle Lion Flakes 450 g               |    5     |
|    8    | Nova Poshta |   Nestle Fitness with vitamins and minerals 425 g   |    3     |
|    8    | Nova Poshta |              Milk 2.5% VILLA MILK 1 l               |    2     |
+---------+-------------+-----------------------------------------------------+----------+

Time of request = 15 ms
PostgreSQL connection is closed
Data selected successfully!
Continue work with DB? Y/N =>
```

```
Make your choice => 6
☑PostgreSQL connect
1. Information about the order by the name of the customer (you can write only the first name, only the last name or first name and last name).
2. What goods were sent via _ ? Enter the name of the shipper?
3. Calculate the cost of the order.
Make your choice => 3

Enter the order number => 5
+---------+---------------+
| OrderID | Cost of order |
+---------+---------------+
|    5    |     172.91    |
+---------+---------------+

Time of request = 17 ms
PostgreSQL connection is closed
Data selected successfully!
Continue work with DB? Y/N =>
```

Вихід із програми

```
Continue work with DB? Y/N => N
                                Goodbye!!
```

Програмний модуль model.py

Цей програмний модуль відповідає за всю ЛОГІКУ проекта.

```
import shutil

import time



def table_headlines(connection, table_name):

    with connection.cursor() as cursor:

        cursor.execute(

            f"""SELECT column_name FROM information_schema.columns WHERE
table_name = '{table_name}' ORDER BY ordinal_position;"""

        )

        full_fetch_headlines = cursor.fetchall()

        i = 0

        headlines = []

        for record in full_fetch_headlines:

            s: str = "".join(c for c in record if c.isalnum())  # залишаються
літери та числа

            headlines.insert(i, s)

            i = i + 1

    return headlines



def show_table(connection, table_name):

    print(f"{table_name}")

    print(f"""SQL query: "SELECT * FROM "{table_name}" ;""")
```

```
    headlines = table_headlines(connection, table_name)


    from prettytable import PrettyTable
    mytable = PrettyTable()
    mytable.field_names = headlines


    with connection.cursor() as cursor:
        cursor.execute(
            f""" SELECT * FROM "{table_name}" ;"""
        )
        full_fetch = cursor.fetchall()
        for record in full_fetch:
            # print(record)
            mytable.add_row(record)
    print(mytable)
    print()



def title_table(connection):
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT "table_name" FROM information_schema.tables WHERE
table_schema='public';"""
        )
        full_fetch = cursor.fetchall()
        i = 0
        title = []
        for record in full_fetch:
```

```
            s: str = "".join(c for c in record if c.isalnum())  # залишаються
літери та числа
            title.insert(i, s)
            i = i + 1
        return title


def show_mas(lines):
    width = shutil.get_terminal_size().columns
    position = (width - max(map(len, lines))) // 2
    i = 1
    for line in lines:  # left justtified
        print(' ' * position + f'{i}. ' + line)
        i = i + 1


def mas_pk(connection):
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT a.attname FROM   pg_index i JOIN   pg_attribute a ON
a.attrelid = i.indrelid  AND a.attnum = ANY(i.indkey) AND    i.indisprimary;"""
            # https://wiki.postgresql.org/wiki/Retrieve_primary_key_columns
        )
        full_fetch = cursor.fetchall()
        i = 0
        pk = []
        for record in full_fetch:
            s: str = "".join(c for c in record if c.isalnum())  # залишаються
літери та числа
            pk.insert(i, s)
```

```
            i = i + 1
    return pk



def fkey(connection, table_name, headline):
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT constraint_name FROM information_schema.key_column_usage
WHERE  table_name = '{table_name}' ;"""
        )
        full_fetch = cursor.fetchall()
        for record in full_fetch:
            s: str = "".join(c for c in str(record) if c.isalnum())
            if s.find(headline) != -1:
                if s.find('fkey') != -1:
                    return 1  # fk
                else:
                    return 0  # not key



def last_value_pk(connection, table_name, headline):
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT "{headline}" FROM "{table_name}" ORDER BY "{headline}"
;"""
        )
        full_fetch = cursor.fetchall()
        values = []
        i = 0
        for record in full_fetch:
```

```
            s = "".join(c for c in str(record) if c.isdecimal())  # залишаються
літери та числа
            values.insert(i, s)
            i = i + 1
        return values[len(values) - 1]



def find_value(connection, headline, value, skip_table):
    title = title_table(connection)
    for record in title:
        if record == skip_table:
            continue
        # print(record)
        column_names = table_headlines(connection, record)
        for record1 in column_names:
            if record1 == headline:
                with connection.cursor() as cursor:
                    cursor.execute(
                        f"""SELECT "{headline}" FROM "{record}" ;"""
                    )
                    full_fetch = cursor.fetchall()
                    i = 0
                    values = []
                    for num in full_fetch:
                        s = "".join(c for c in str(num) if c.isdecimal())  #
залишаються літери та числа
                        values.insert(i, int(s))
                        i = i + 1
                    need_num = -1
                    for num in values:
```

```
                    if int(num) == int(value):
                        need_num = num
                        break
                if need_num == -1:
                    return 0
                else:
                    return 1


def data_type(connection, table_name, headline):
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT "data_type" FROM information_schema.columns WHERE
table_schema='public' and "table_name" = '{table_name}' and "column_name"
='{headline}';"""
        )
        full_fetch = cursor.fetchall()
        values = []
        i = 0
        for record in full_fetch:
            s: str = "".join(c for c in str(record) if c.isdecimal())  #
залишаються літери та числа
            values.insert(i, str(s))
            i = i + 1
        return full_fetch[0]


def select1(Name, connection):
    from prettytable import PrettyTable
    mytable = PrettyTable()
```

```
    mytable.field_names = ["CustomerName", "OrderID", "OrderDate",
"ShipperName", "ProductName", "Quantity"]

    beg = int(time.time() * 1000)


    with connection.cursor() as cursor:

        cursor.execute(

            f"""SELECT c."CustomerName", o."OrderID", o."OrderDate",
s."ShipperName", p."ProductName", od."Quantity"  FROM "Customers" as c

                JOIN "Orders" as o ON c."CustomerID" = o."CustomerID"

                JOIN "Shippers" as s ON o."ShipperID" = s."ShipperID"

                JOIN "OrderDetails" as od ON o."OrderID" = od."OrderID"

                JOIN "Products" as p ON p."ProductID" = od."ProductID"

                WHERE "CustomerName" LIKE '%{Name}%';"""

        )

        end = int(time.time() * 1000) - beg

        full_fetch = cursor.fetchall()

        for record in full_fetch:

            # print(record)

            mytable.add_row(record)

    print(mytable)

    print()

    print('Time of request = {} ms'.format(end))



def select2(Name, connection):

    from prettytable import PrettyTable

    mytable = PrettyTable()

    mytable.field_names = ["OrderID", "ShipperName", "ProductName", "Quantity"]

    beg = int(time.time() * 1000)
```

```
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT o."OrderID", s."ShipperName", p."ProductName",
od."Quantity"  FROM "Orders" as o
                JOIN "Shippers" as s ON o."ShipperID" = s."ShipperID"
                JOIN "OrderDetails" as od ON o."OrderID" = od."OrderID"
                JOIN "Products" as p ON p."ProductID" = od."ProductID"
                WHERE "ShipperName" LIKE '%{Name}%';;"""
        )
        end = int(time.time() * 1000) - beg
        full_fetch = cursor.fetchall()
        for record in full_fetch:
            # print(record)
            mytable.add_row(record)
    print(mytable)
    print()
    print('Time of request = {} ms'.format(end))



def select3(Number, connection):
    from prettytable import PrettyTable
    mytable = PrettyTable()
    mytable.field_names = ["OrderID", "Cost of order"]
    beg = int(time.time() * 1000)

    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT o."OrderID", SUM("Price" * "Quantity") as "Cost of
order" FROM "Orders" as o
                JOIN "OrderDetails" as od ON o."OrderID" = od."OrderID"
```

```
                JOIN "Products" as p ON p."ProductID" = od."ProductID"
                WHERE o."OrderID" = '{Number}'
                GROUP BY o."OrderID";"""
        )
        end = int(time.time() * 1000) - beg
        full_fetch = cursor.fetchall()
        for record in full_fetch:
            # print(record)
            mytable.add_row(record)
    print(mytable)
    print()
    print('Time of request = {} ms'.format(end))



def insert(headers_selected_table, con, title, num, values):
    columns = ' "' + str(headers_selected_table[0]) + '" '
    for i in range(1, len(headers_selected_table)):
        columns = str(columns) + ' , "' + str(headers_selected_table[i]) + '" '
    data = " '" + str(values[0]) + "' "
    for i in range(1, len(values)):
        data = str(data) + " , '" + str(values[i]) + "' "
    with con.cursor() as cursor:
        cursor.execute(
            f"""INSERT INTO "{title[num - 1]}" ({columns} ) VALUES ({data});"""
        )
    print(title[num - 1])
    print(f""" SQL query: INSERT INTO "{title[num - 1]}" ({columns} ) VALUES
({data});""")
    print("Inserted successfully!!")
```

```
def delete(con, title, num, headers_selected_table, num_id):

    with con.cursor() as cursor:

        cursor.execute(

            f"""DELETE FROM "{title[num - 1]}" WHERE
"{headers_selected_table[0]}" = '{num_id}';"""

        )

        print(title[num - 1])

        print(f""" SQL query: DELETE FROM {title[num - 1]} WHERE
"{headers_selected_table[0]}" = '{num_id}';""")

        print(" Deleted successfully!!")




def update(con, title, num, headers_selected_table, num_item, num_id, value):

    with con.cursor() as cursor:

        cursor.execute(

            f""" UPDATE "{title[num - 1]}" SET
"{headers_selected_table[num_item]}" = '{value}'

                WHERE "{headers_selected_table[0]}" = '{num_id}';"""

        )

    print(title[num - 1])

    print(

        f""" SQL query: UPDATE "{title[num - 1]}" SET
"{headers_selected_table[num_item]}" = '{value}'

            WHERE "{headers_selected_table[0]}" = '{num_id}';""")

    print(" Data updated successfully!!")




def rand(con, title, num, columns, data, number):

    with con.cursor() as cursor:
```

```
        cursor.execute(
            f""" INSERT INTO "{title[num - 1]}" ({columns}) SELECT {data} from
generate_series(1, {number}) ;"""
        )
    print(title[num - 1])
    print(
        f""" SQL query: INSERT INTO "{title[num - 1]}" ({columns}) SELECT
{data} from generate_series(1, {number});""")
    print("Inserted randomly")



def q_exit():
    print('Continue work with DB? Y/N =>', end=' ')
    while 1:
        c = input()
        if c == 'Y' or c == 'y':
            return 1
        elif c == 'N' or c == 'n':
            return 0
        else:
            print("The wrong value was entered, it will be entered again. Y/N
=>", end=' ')
```

Також клас Model, який включає всі потрібні функції:

table_headlines - повертає масив назв стовпців у заданій таблиці

show_table - виводить на екран задану таблицю

title_table - повертає масив назв таблиць з бази даних

show_mas - виводить масив рядків на екран у вигляді меню

fkey - перевіряє чи задана назва є вторинним ключ, якщо так то повертає 1, якщо ні - 0

last_value_pk - повертає найбільше значення для первинного ключа

find_value - перевіряє чи існує така значення, повертає 1(існує) чи 0(не існує)

data_type - повертає тип стовпця заданої таблиці

select1 - виконується перший запит на пошук інформації

select2 - виконується другий запит на пошук інформації

select3 - виконується третій запит на пошук інформації

insert - sql запит на додавання інформації

delete - sql запит на видалення інформації

update - sql запит на оновлення інформації

rand - sql запит на псевдорандомне заповнення інформацією таблиці

q_exit - запитує чи завершується робота з програмою, якщо так(“Y”), то повертає 0 у MainMenu та завершується робота програми, якщо ні(“N”) то робота продовжується