

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 ІII курсу

групи КВ-01

Черичка М. А.

Перевірив:

Павловський В. І.

Київ – 2022

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. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі. 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).

Деталізоване завдання:

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.

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!

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.

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.

## Логічна мод**ель** предметної області «Інтернет-кафе»

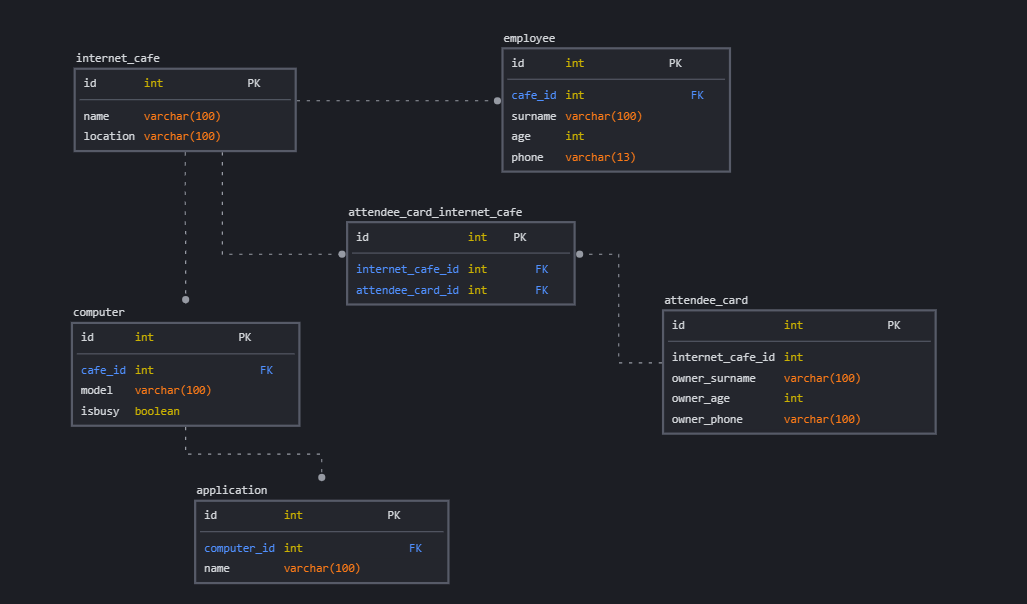


Рисунок 1 - Схема бази даних (інструмент: sqldbm.com)

## Середовище та компоненти розробки

Для розробки використовувалась мова програмування Python, середовище розробки PyCharm, а також стороння бібліотека, що надає API для доступу до PostgreSQL – psycopg2.

## Шаблон проектування

MVC - Шаблон проектування, який використаний у програмі.

Model – представляє набір функцій що описують логіку використовуваних даних. Згідно компоненту моделі, у моїй програмі відповідають всі компоненти які знаходять у файлі model.py.

View – в нашому випадку консольний інтерфейс з яким буде взаємодіяти наш користувач. Згідно компоненту представлення, то їй відповідають такі компоненти, згідно яким користувач бачить необхідні дані, що є представленням даних у вигляді консольного інтерфейсу.

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок.

## **Структура програми та її** опис

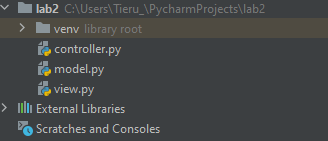


Рисунок 2. Структура програми.

Програма умовно поділена на 3 модулі: model.py, view.py, controller.py. Класи, як видно з їх назв, повністю відповідають використаному патерну MVC.

У файлі model.py описано клас Model, що займається регулювання підключення до бази даних, та виконанням низькорівневих запитів до неї.

У файлах view.py описано клас View, що виводить результати виконання тієї чи іншої дії.

У файлі controller.py описано клас Controller, які оброблюються завдяки взаємодії з користувачем, запит бажаної дії, виконання пошуку, тощо. Також в цьому файлі створюється екземпляр класу Controller, якому користувач передає вказівки щодо виконання програми

## Структура меню програми

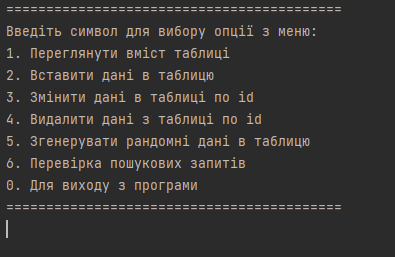


Рисунок 3. Меню для вибору операції над БД.

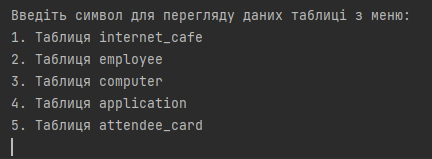


Рисунок 4. Меню для вибору таблиці, над якою проводитиметься обрана операція.

## Меню операцій складається з семи пунктів (Рисунок 3)

1. Виведення вибраної користувачем таблиці
2. Додавання нового рядку даних до таблиці.
3. Оновлення рядку даних у таблиці по заданому ідентифікатору.
4. Видалення рядку даних з таблиці по заданому ідентифікатору.
5. Додавання заданої кількості нових випадкових даних до вибраної таблиці БД.
6. Демонстрація запитів з певними параметрами, та виведення заміряної швидкості їх виконання.
7. Завершення програми.

## Меню таблиць складається з 7 пунктів (Рисунок 3)

1. Виконати обрану операцію над таблицею “internet\_cafe”.
2. Виконати обрану операцію над таблицею “employee”.
3. Виконати обрану операцію над таблицею “computer”.
4. Виконати обрану операцію над таблицею “application”.
5. Виконати обрану операцію над таблицею “attendee\_card”.

## Результати та виконання операцій

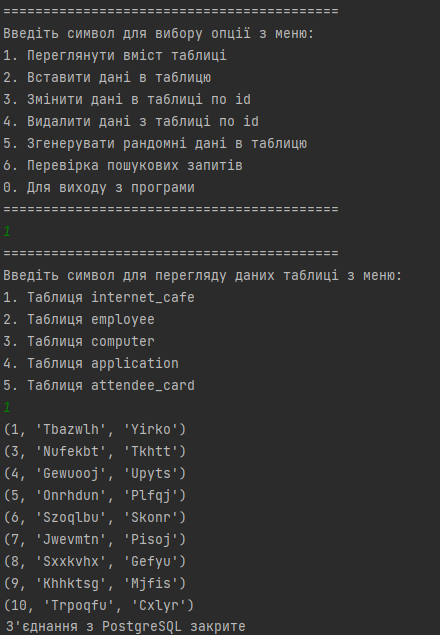


Рисунок 5. Перегляд таблиці “internet\_cafe”.

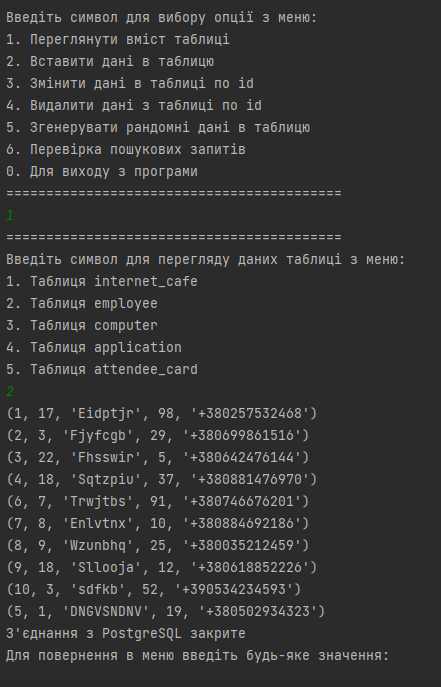


Рисунок 6. Перегляд таблиці “attendee”.

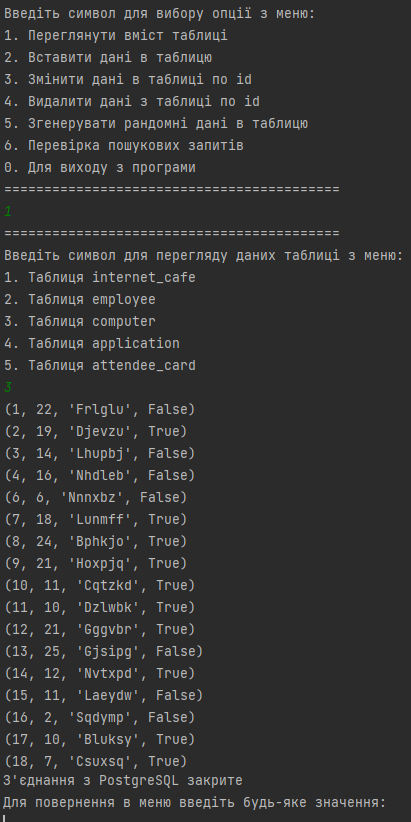


Рисунок 7. Перегляд таблиці “Computer”.

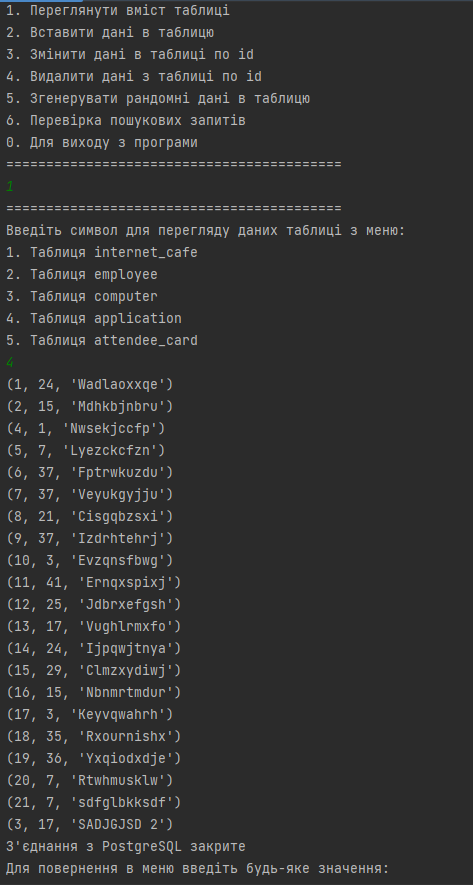


Рисунок 8. Перегляд таблиці “application”.

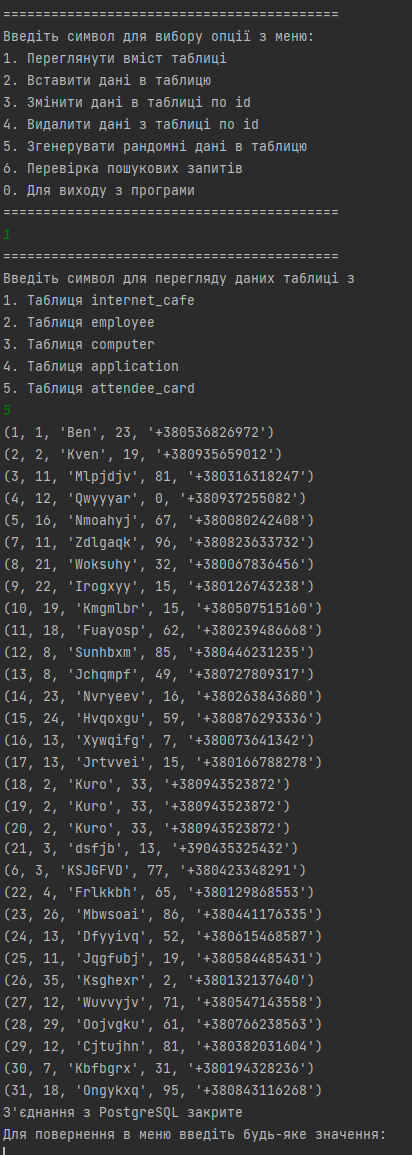
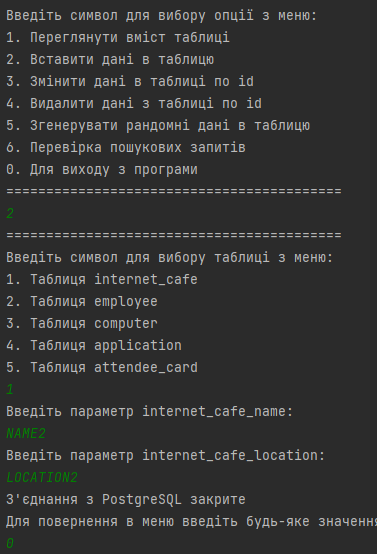
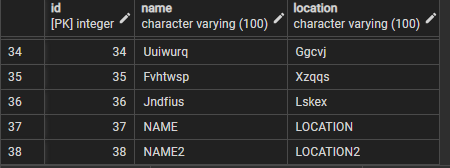
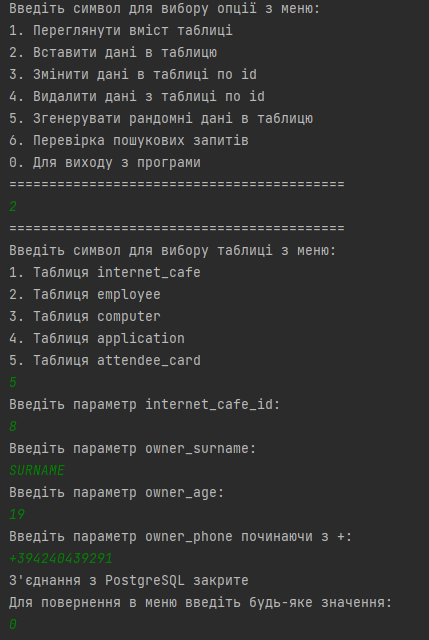


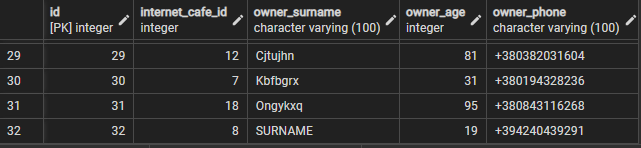
Рисунок 9. Перегляд таблиці “attendee\_card”.

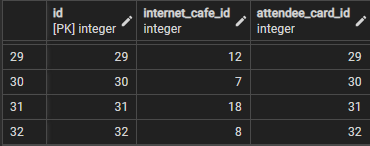




Рисунки 10-11. Вставлення даних до таблиці “internet\_cafe”.







Рисунки 12-14. Вставлення даних до таблиці “attendee\_card” та автоматичне заповнення таблиці “attendee\_card\_internet\_cafe”.

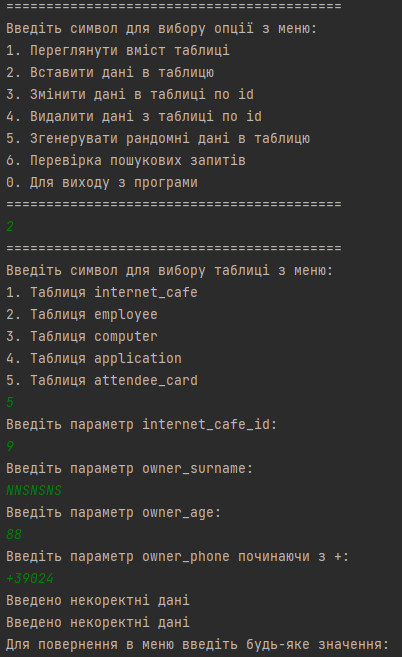
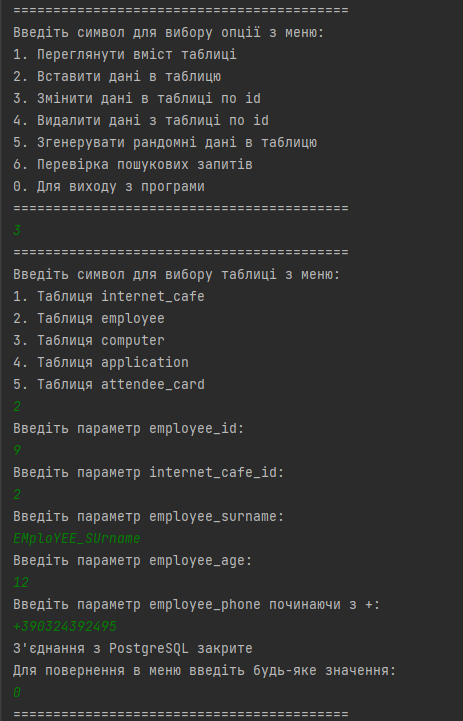
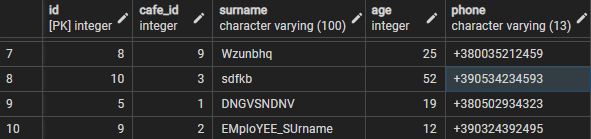


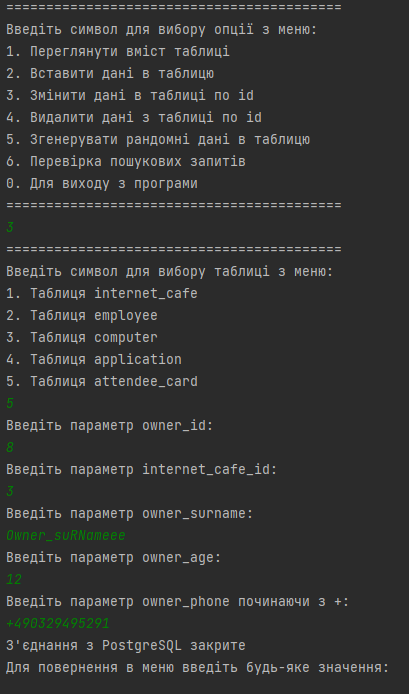
Рисунок 15. Спроба ввести некоректні дані в таблицю “attendee\_card”

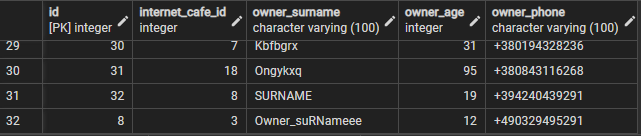
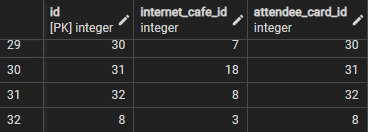
(валідація даних за кількістю символів в параметрі “owner\_phone”та перевірка параметру “internet\_cafe\_id”).





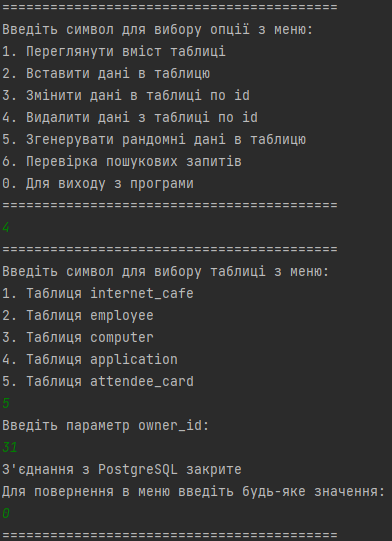
Рисунки 16-17. Оновлення даних в таблиці “employee” за параметром “employee\_id”.

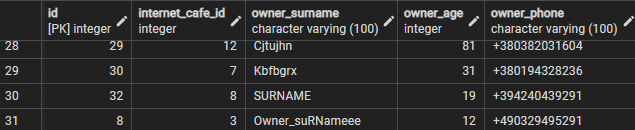


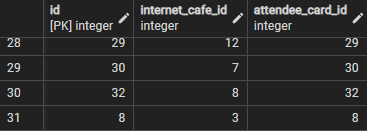
 

Рисунки 18-20. Оновлення даних в таблиці “attendee\_card” та автоматичне оновлення таблиці “attendee\_card\_internet\_cafe”

за параметром “attendee\_card \_ owner\_id”.

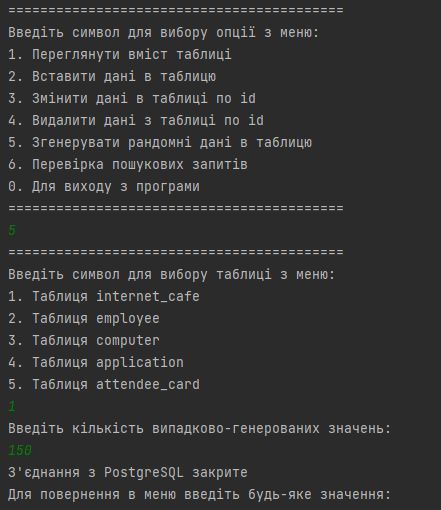


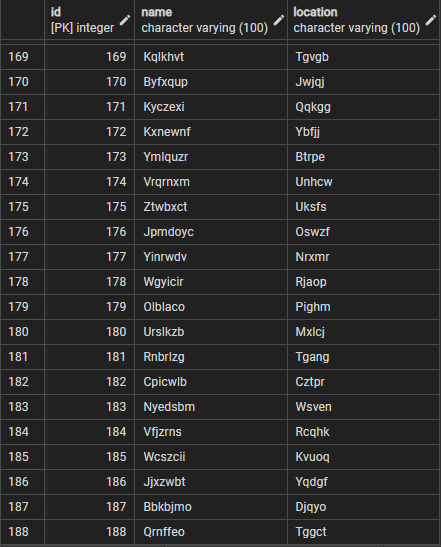




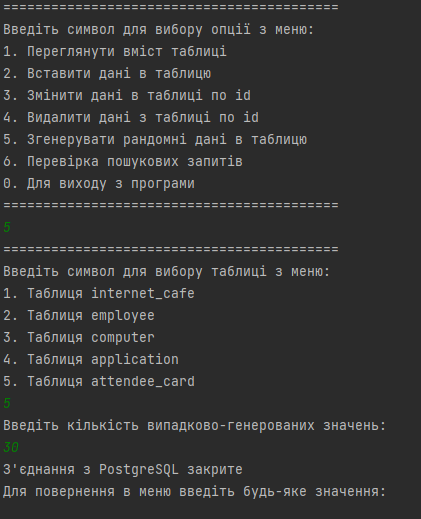
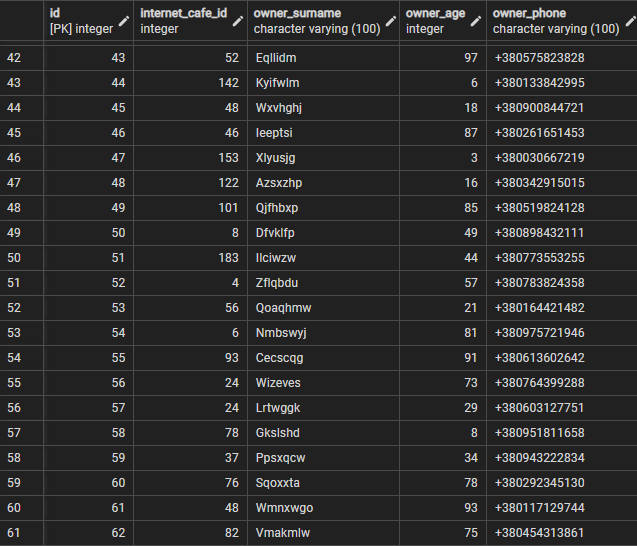
Рисунки 21-23. Видалення даних з таблиці “attendee\_card” та автоматичне оновлення таблиці “attendee\_card\_internet\_cafe”

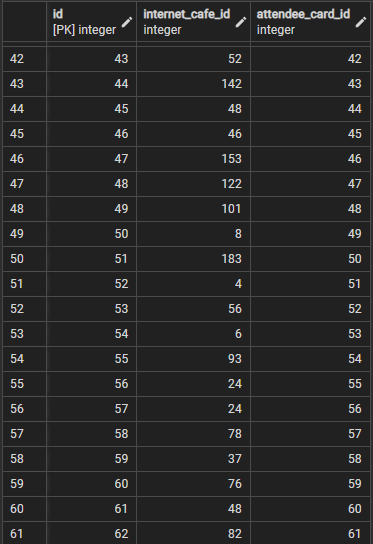
за параметром “attendee\_card\_owner\_id”.



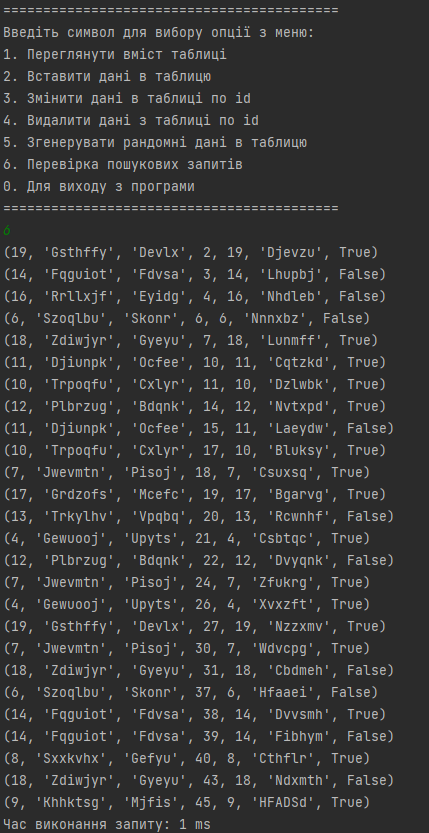
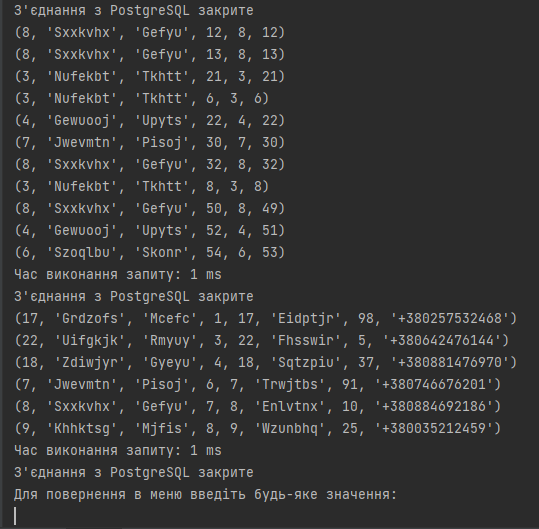


Рисунки 24-25. Додавання заданої кількості випадково-генерованих значень до таблиці “internet\_cafe”.



Рисунки 26-28. Додавання заданої кількості випадково-генерованих значень до таблиць “attendee\_card” та “attendee\_card\_internet\_cafe” відповідно.

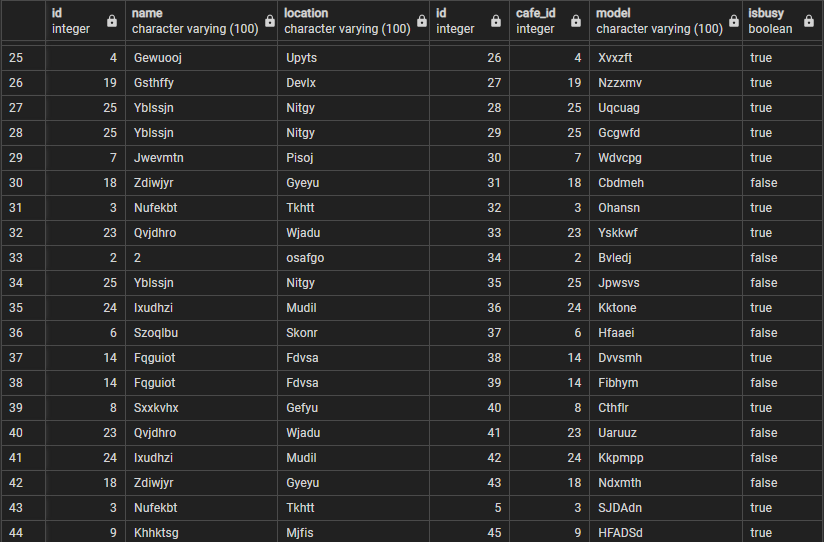
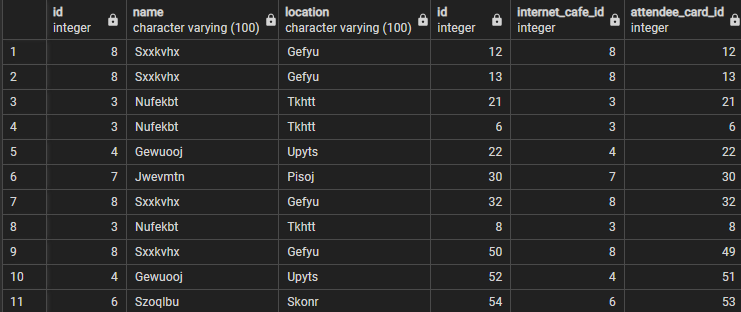
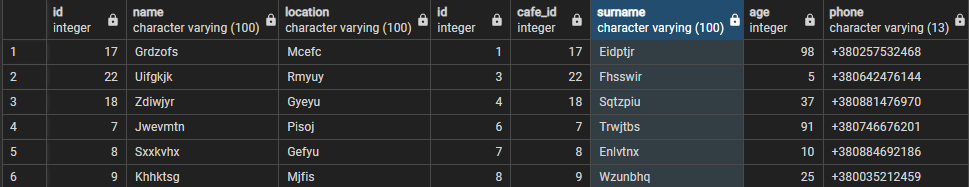
\ 

Рисунки 29-30. Демонстрація виконання запитів за заданими параметрами з двох таблиць.

f"""select \* from internet\_cafe INNER JOIN computer ON computer.cafe\_id = internet\_cafe.id   
WHERE internet\_cafe.id <= 20 AND internet\_cafe.id > 3"""

f"""select \* from internet\_cafe INNER JOIN attendee\_card\_internet\_cafe ON internet\_cafe.id = attendee\_card\_internet\_cafe.internet\_cafe\_id   
WHERE internet\_cafe.id <= 10 AND internet\_cafe.id >2"""

f"""select \* from internet\_cafe INNER JOIN employee ON employee.cafe\_id = internet\_cafe.id   
WHERE internet\_cafe.id <= 200 AND internet\_cafe.id >3"""

Рисунки 31-33. Демонстрація виконання запитів в pgAdmin4 за заданими параметрами з двох таблиць.

## Код програми

**model.py**

import psycopg2  
import random  
import self  
import time  
from view import view  
  
def current\_milli\_time():  
 return round(time.time() \* 1000)  
  
class data:  
 tableName = "attendee\_card"  
 internet\_cafe\_name = "Veselyy Gorikh"  
 internet\_cafe\_location = "Irpin"  
 internet\_cafe\_id = 2  
 employee\_id = 2  
 employee\_surname = "Bateman"  
 employee\_age = 22  
 employee\_phone = "+479218865456"  
 computer\_id = 1  
 computer\_model = "Gaming"  
 computer\_isbusy = False  
 application\_id = 3  
 application\_name = "Earthworm Jim"  
 attendee\_card\_owner\_id = 2  
 attendee\_card\_owner\_phone = "+380943523872"  
 attendee\_card\_owner\_surname = "Kuro"  
 attendee\_card\_owner\_age = 33  
 quant = 4  
  
def checkTable():  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 try:  
 # створення та виконання запиту  
 cursor.execute(f"SELECT \*FROM {data.tableName}")  
 # отримання результату запиту  
 row = cursor.fetchone()  
 # обробка результату запиту  
 while row is not None:  
 print(row)  
 row = cursor.fetchone()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 # Закриття з'єднання з БД.  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
def insertIntoTable(validation\_status=False):  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
  
 if data.tableName == "internet\_cafe":  
 try:  
 cursor.execute("INSERT INTO internet\_cafe (name,location) VALUES (%s,%s)", (data.internet\_cafe\_name, data.internet\_cafe\_location))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 # Закриття з'єднання з БД.  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "employee":  
 try:  
 cursor.execute(f"SELECT \*FROM internet\_cafe")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.employee\_phone) == 13:  
 if row[0] == data.internet\_cafe\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("INSERT INTO employee (cafe\_id,surname,age,phone) VALUES (%s,%s,%s,%s)", (data.internet\_cafe\_id, data.employee\_surname, data.employee\_age, data.employee\_phone))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "computer":  
 try:  
 cursor.execute(f"SELECT \*FROM internet\_cafe")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.computer\_model) == 6:  
 if row[0] == data.internet\_cafe\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("INSERT INTO computer (cafe\_id,model,isbusy) VALUES (%s,%s,%s)", (data.internet\_cafe\_id, data.computer\_model, data.computer\_isbusy))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "application":  
 try:  
 cursor.execute(f"SELECT \*FROM computer")  
 row = cursor.fetchone()  
 while row is not None:  
 if row[0] == data.computer\_id:  
 validation\_status = True  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("INSERT INTO application (computer\_id,name) VALUES (%s,%s)", (data.computer\_id, data.application\_name))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "attendee\_card":  
 try:  
 cursor.execute(f"SELECT \*FROM internet\_cafe")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.attendee\_card\_owner\_phone) == 13:  
 if row[0] == data.internet\_cafe\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("INSERT INTO attendee\_card (internet\_cafe\_id,owner\_surname,owner\_age,owner\_phone) VALUES (%s,%s,%s,%s)", (data.internet\_cafe\_id, data.attendee\_card\_owner\_surname, data.attendee\_card\_owner\_age, data.attendee\_card\_owner\_phone))  
 cursor.execute("SELECT \*FROM attendee\_card\_internet\_cafe")  
 row = cursor.fetchone()  
 cnt = 1  
 while row is not None:  
 cnt = cnt+1  
 row = cursor.fetchone()  
 cursor.execute("INSERT INTO attendee\_card\_internet\_cafe (internet\_cafe\_id,attendee\_card\_id) VALUES (%s,%s)", (data.internet\_cafe\_id, cnt))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 else:  
 view.incorrectTableName(self)  
  
def updateTable(validation\_status=False):  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
  
 if data.tableName == "internet\_cafe":  
 try:  
 cursor.execute("SELECT \*FROM internet\_cafe")  
 row = cursor.fetchone()  
 while row is not None:  
 if row[0] == data.internet\_cafe\_id:  
 validation\_status = True  
 break  
 row = cursor.fetchone()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == False:  
 view.incorrectData(self)  
 if validation\_status == True:  
 try:  
 cursor.execute("UPDATE internet\_cafe SET name =%s,location=%s WHERE id=%s ", (data.internet\_cafe\_name, data.internet\_cafe\_location, data.internet\_cafe\_id))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 # Закриття з'єднання з БД.  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "employee":  
 try:  
 cursor.execute("SELECT \*FROM employee")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.employee\_phone) == 13:  
 if row[0] == data.employee\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("Update employee SET cafe\_id =%s, surname =%s, age = %s, phone = %s WHERE id = %s", (data.internet\_cafe\_id, data.employee\_surname, data.employee\_age, data.employee\_phone, data.employee\_id))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "computer":  
 try:  
 cursor.execute("SELECT \*FROM computer")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.computer\_model) == 6:  
 if row[0] == data.computer\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("UPDATE computer SET cafe\_id =%s, model =%s, isbusy =%s WHERE id =%s ", (data.internet\_cafe\_id, data.computer\_model, data.computer\_isbusy, data.computer\_id))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "application":  
 try:  
 cursor.execute("SELECT \*FROM application")  
 row = cursor.fetchone()  
 while row is not None:  
 if row[0] == data.application\_id:  
 validation\_status = True  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("UPDATE application SET computer\_id =%s, name =%s WHERE id =%s", (data.computer\_id, data.application\_name, data.application\_id))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "attendee\_card":  
 try:  
 cursor.execute("SELECT \*FROM attendee\_card")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.attendee\_card\_owner\_phone) == 13:  
 if row[0] == data.attendee\_card\_owner\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("UPDATE attendee\_card SET internet\_cafe\_id =%s, owner\_surname =%s ,owner\_age =%s ,owner\_phone =%s WHERE id = %s", (data.internet\_cafe\_id, data.attendee\_card\_owner\_surname, data.attendee\_card\_owner\_age, data.attendee\_card\_owner\_phone, data.attendee\_card\_owner\_id))  
 cursor.execute("SELECT \*FROM attendee\_card\_internet\_cafe")  
 row = cursor.fetchone()  
 while row is not None:  
 row = cursor.fetchone()  
 cursor.execute("UPDATE attendee\_card\_internet\_cafe SET internet\_cafe\_id=%s WHERE id= %s", (data.internet\_cafe\_id, data.attendee\_card\_owner\_id))  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 else:  
 view.incorrectTableName(self)  
  
def deleteFromTable(validation\_status=False):  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
  
 if data.tableName == "internet\_cafe":  
 try:  
 cursor.execute("SELECT \*FROM internet\_cafe")  
 row = cursor.fetchone()  
 while row is not None:  
 if row[0] == data.internet\_cafe\_id:  
 validation\_status = True  
 break  
 row = cursor.fetchone()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == False:  
 view.incorrectData(self)  
 if validation\_status == True:  
 try:  
 cursor.execute("DELETE FROM internet\_cafe WHERE id =%s", [data.internet\_cafe\_id])  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "employee":  
 try:  
 cursor.execute("SELECT \*FROM employee")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.employee\_phone) == 13:  
 if row[0] == data.employee\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("DELETE FROM employee WHERE id = %s", [data.employee\_id])  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.incorrectData(self)  
  
 elif data.tableName == "computer":  
 try:  
 cursor.execute("SELECT \*FROM computer")  
 row = cursor.fetchone()  
 while row is not None:  
 if len(data.computer\_model) == 6:  
 if row[0] == data.computer\_id:  
 validation\_status = True  
 break  
 else:  
 view.incorrectData(self)  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("DELETE FROM computer WHERE id =%s ", [data.computer\_id])  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "application":  
 try:  
 cursor.execute("SELECT \*FROM application")  
 row = cursor.fetchone()  
 while row is not None:  
 if row[0] == data.application\_id:  
 validation\_status = True  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("DELETE FROM application WHERE id =%s", [data.application\_id])  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "attendee\_card":  
 try:  
 cursor.execute("SELECT \*FROM attendee\_card")  
 row = cursor.fetchone()  
 while row is not None:  
 if row[0] == data.attendee\_card\_owner\_id:  
 validation\_status = True  
 break  
 row = cursor.fetchone()  
 if validation\_status == False:  
 view.incorrectData(self)  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 if validation\_status == True:  
 try:  
 cursor.execute("DELETE FROM attendee\_card S WHERE id = %s", [data.attendee\_card\_owner\_id])  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
 else:  
 view.incorrectTableName(self)  
  
def genRandIntoTable(quant):  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
  
 if data.tableName == "internet\_cafe":  
 try:  
 length = 7  
 uppercase\_letter = "chr(ascii('A') + (random() \* 25)::int)"  
 lowercase\_letter = "chr(ascii('a') + (random() \* 25)::int)"  
 for i in range(0, quant):  
 cursor.execute(f"""INSERT INTO internet\_cafe (name,location)   
 VALUES ({uppercase\_letter}{(" || " + lowercase\_letter) \* (length - 1)},  
 {uppercase\_letter}{(" || " + lowercase\_letter) \* (length - 3)})""")  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "employee":  
 try:  
 cursor.execute("select id from internet\_cafe ORDER BY id DESC limit 1")  
 maxCafeId = cursor.fetchone()[0]  
 length = 7  
 uppercase\_letter = "chr(ascii('A') + (random() \* 25)::int)"  
 lowercase\_letter = "chr(ascii('a') + (random() \* 25)::int)"  
 digit = "chr(ascii('0') + (random() \* 9)::int)"  
 for i in range(0, quant):  
 cursor.execute(f"""INSERT INTO employee (cafe\_id,surname,age,phone)   
 VALUES(random()\*{maxCafeId-1}+1,   
 {uppercase\_letter}{(" || " + lowercase\_letter) \* (length - 1)},  
 random()\*100::int,  
 '+380'{(" || "+digit)\*9})""")  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "computer":  
 try:  
 cursor.execute("select id from internet\_cafe ORDER BY id DESC limit 1")  
 maxCafeId = cursor.fetchone()[0]  
 length = 7  
 uppercase\_letter = "chr(ascii('A') + (random() \* 25)::int)"  
 lowercase\_letter = "chr(ascii('a') + (random() \* 25)::int)"  
 digit = "chr(ascii('0') + (random() \* 9)::int)"  
 for i in range(0, quant):  
 cursor.execute(f"""INSERT INTO computer (cafe\_id,model,isbusy)   
 VALUES (random()\*{maxCafeId-1}+1,  
 {uppercase\_letter}{(" || " + lowercase\_letter) \* 5},  
 {bool(random.randint(0,2))})""")  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "application":  
 try:  
 cursor.execute("select id from computer ORDER BY id DESC limit 1")  
 maxComputerId = cursor.fetchone()[0]  
 length = 10  
 uppercase\_letter = "chr(ascii('A') + (random() \* 25)::int)"  
 lowercase\_letter = "chr(ascii('a') + (random() \* 25)::int)"  
 for i in range(0, quant):  
 cursor.execute(f"""INSERT INTO application (computer\_id,name)   
 VALUES (random()\*{maxComputerId-1}+1,  
 {uppercase\_letter}{(" || " + lowercase\_letter) \* (length - 1)})""")  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 # Закриття з'єднання з БД.  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
 elif data.tableName == "attendee\_card":  
 try:  
 cursor.execute("select id from internet\_cafe ORDER BY id DESC limit 1")  
 maxCafeId = cursor.fetchone()[0]  
 length = 7  
 uppercase\_letter = "chr(ascii('A') + (random() \* 25)::int)"  
 lowercase\_letter = "chr(ascii('a') + (random() \* 25)::int)"  
 digit = "chr(ascii('0') + (random() \* 9)::int)"  
 for i in range(0, quant):  
 var = random.randint(0, maxCafeId-1) + 1  
 cursor.execute(f"""INSERT INTO attendee\_card (internet\_cafe\_id,owner\_surname,owner\_age,owner\_phone)   
 VALUES ({var},  
 {uppercase\_letter}{(" || " + lowercase\_letter) \* (length - 1)},  
 random()\*100::int,  
 '+380'{(" || "+digit)\*9})""")  
 cursor.execute("SELECT \*FROM attendee\_card\_internet\_cafe")  
 row = cursor.fetchone()  
 cnt = 1  
 while row is not None:  
 row = cursor.fetchone()  
 cnt = cnt + 1  
 cursor.execute(f"""INSERT INTO attendee\_card\_internet\_cafe (internet\_cafe\_id,attendee\_card\_id)   
 VALUES ({var},  
 {cnt})""")  
 connection.commit()  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 # Закриття з'єднання з БД.  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
 else:  
 view.incorrectTableName(self)  
  
def atrSearch1():  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres",  
 password="Kjujc329!")  
 cursor = connection.cursor()  
 time\_start = current\_milli\_time()  
 cursor.execute(f"""select \* from internet\_cafe INNER JOIN computer ON computer.cafe\_id = internet\_cafe.id   
 WHERE internet\_cafe.id <= 20 AND internet\_cafe.id > 3""")  
 row = cursor.fetchone()  
 while row is not None:  
 print(row)  
 row = cursor.fetchone()  
 time\_end = current\_milli\_time()  
 print(f"Час виконання запиту: {time\_end - time\_start} ms ")  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
def atrSearch2():  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 time\_start = current\_milli\_time()  
 cursor.execute(f"""select \* from internet\_cafe INNER JOIN attendee\_card\_internet\_cafe ON internet\_cafe.id = attendee\_card\_internet\_cafe.internet\_cafe\_id   
 WHERE internet\_cafe.id <= 10 AND internet\_cafe.id >2""")  
 row = cursor.fetchone()  
 while row is not None:  
 print(row)  
 row = cursor.fetchone()  
 time\_end = current\_milli\_time()  
 print(f"Час виконання запиту: {time\_end-time\_start} ms ")  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
def atrSearch3():  
 try:  
 connection = psycopg2.connect(host="localhost", port="5432", database="lab1", user="postgres", password="Kjujc329!")  
 cursor = connection.cursor()  
 time\_start = current\_milli\_time()  
 cursor.execute(f"""select \* from internet\_cafe INNER JOIN employee ON employee.cafe\_id = internet\_cafe.id   
 WHERE internet\_cafe.id <= 200 AND internet\_cafe.id >3""")  
 row = cursor.fetchone()  
 while row is not None:  
 print(row)  
 row = cursor.fetchone()  
 time\_end = current\_milli\_time()  
 print(f"Час виконання запиту: {time\_end-time\_start} ms ")  
 except (Exception, psycopg2.Error) as error:  
 view.errorMessage(self, error)  
 finally:  
 if connection:  
 cursor.close()  
 connection.close()  
 view.connectionClose(self)  
  
  
# checkTable()  
# deleteFromTable()  
# updateTable()  
# insertIntoTable()  
# checkTable()  
# genRandIntoTable(15)  
# checkTable()  
# atrSearch()

## view.py

class view:  
  
 def menuView(self):  
 print("==========================================")  
 print("Введіть символ для вибору опції з меню:")  
 print("1. Переглянути вміст таблиці")  
 print("2. Вставити дані в таблицю")  
 print("3. Змінити дані в таблиці по id")  
 print("4. Видалити дані з таблиці по id")  
 print("5. Згенерувати рандомні дані в таблицю")  
 print("6. Перевірка пошукових запитів ")  
 print("0. Для виходу з програми ")  
 print("==========================================")  
  
 def connectionClose(self):  
 print("З'єднання з PostgreSQL закрите")  
  
 def incorrectData(self):  
 print("Введено некоректні дані")  
  
 def incorrectTableName(self):  
 print("Введено некоректну назву таблиці")  
  
 def errorMessage(self, error):  
 print("Виникла помилка при виконанні запиту", error)  
  
 def insertTableName(self):  
 print("==========================================")  
 print("Введіть символ для вибору таблиці з меню:")  
 print("1. Таблиця internet\_cafe")  
 print("2. Таблиця employee")  
 print("3. Таблиця computer")  
 print("4. Таблиця application")  
 print("5. Таблиця attendee\_card")  
  
 def uncorrectSymbol(self):  
 print("Невірно введений символ")  
  
 def insertInto\_internet\_cafe\_name(self):  
 print("Введіть параметр internet\_cafe\_name:")  
  
 def insertInto\_internet\_cafe\_location(self):  
 print("Введіть параметр internet\_cafe\_location:")  
  
 def insertInto\_internet\_cafe\_id(self):  
 print("Введіть параметр internet\_cafe\_id:")  
  
 def insertInto\_employee\_id(self):  
 print("Введіть параметр employee\_id:")  
  
 def insertInto\_employee\_surname(self):  
 print("Введіть параметр employee\_surname:")  
  
 def insertInto\_employee\_age(self):  
 print("Введіть параметр employee\_age:")  
  
 def insertInto\_employee\_phone(self):  
 print("Введіть параметр employee\_phone починаючи з +:")  
  
 def insertInto\_computer\_model(self):  
 print("Введіть параметр computer\_model:")  
  
 def insertInto\_computer\_isbusy(self):  
 print("Введіть параметр computer\_isbusy(0=False,1=True):")  
  
 def insertInto\_computer\_id(self):  
 print("Введіть параметр computer\_id:")  
  
 def insertInto\_application\_id(self):  
 print("Введіть параметр application\_id:")  
  
 def insertInto\_application\_name(self):  
 print("Введіть параметр application\_name:")  
  
 def insertInto\_attendee\_card\_owner\_id(self):  
 print("Введіть параметр owner\_id:")  
  
 def insertInto\_attendee\_card\_owner\_surname(self):  
 print("Введіть параметр owner\_surname:")  
  
 def insertInto\_attendee\_card\_owner\_age(self):  
 print("Введіть параметр owner\_age:")  
  
 def insertInto\_attendee\_card\_owner\_phone(self):  
 print("Введіть параметр owner\_phone починаючи з +:")  
  
 def insertQuantityOfRandomValues(self):  
 print("Введіть кількість випадково-генерованих значень:")  
  
 def waitingButtonToPress(self):  
 print("Для повернення в меню введіть будь-яке значення:")

## controller.py

from model import data, checkTable, insertIntoTable, updateTable, deleteFromTable, genRandIntoTable, atrSearch1, atrSearch2, atrSearch3  
import self  
from view import view  
  
class controller:  
 def menu(self):  
 view.menuView(self)  
 option = input()  
 if option == '1':  
 view.insertTableName(self)  
 option = input()  
 if option == '1':  
 data.tableName = "internet\_cafe"  
 checkTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '2':  
 data.tableName = "employee"  
 checkTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '3':  
 data.tableName = "computer"  
 checkTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '4':  
 data.tableName = "application"  
 checkTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '5':  
 data.tableName = "attendee\_card"  
 checkTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 else:  
 view.uncorrectSymbol(self)  
 controller.menu(self)  
 elif option == '2':  
 view.insertTableName(self)  
 option = input()  
 if option == '1':  
 data.tableName = "internet\_cafe"  
 view.insertInto\_internet\_cafe\_name(self)  
 data.internet\_cafe\_name = str(input())  
 view.insertInto\_internet\_cafe\_location(self)  
 data.internet\_cafe\_location = str(input())  
 insertIntoTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '2':  
 data.tableName = "employee"  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_employee\_surname(self)  
 data.employee\_surname = str(input())  
 view.insertInto\_employee\_age(self)  
 data.employee\_age = int(input())  
 view.insertInto\_employee\_phone(self)  
 data.employee\_phone = str(input())  
 insertIntoTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '3':  
 data.tableName = "computer"  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_computer\_model(self)  
 data.computer\_model = str(input())  
 view.insertInto\_computer\_isbusy(self)  
 data.computer\_isbusy = bool(input())  
 insertIntoTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '4':  
 data.tableName = "application"  
 view.insertInto\_computer\_id(self)  
 data.computer\_id = int(input())  
 view.insertInto\_application\_name(self)  
 data.application\_name = str(input())  
 insertIntoTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '5':  
 data.tableName = "attendee\_card"  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_attendee\_card\_owner\_surname(self)  
 data.attendee\_card\_owner\_surname = str(input())  
 view.insertInto\_attendee\_card\_owner\_age(self)  
 data.attendee\_card\_owner\_age = int(input())  
 view.insertInto\_attendee\_card\_owner\_phone(self)  
 data.attendee\_card\_owner\_phone = str(input())  
 insertIntoTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 else:  
 view.uncorrectSymbol(self)  
 controller.menu(self)  
 elif option == '3':  
 view.insertTableName(self)  
 option = input()  
 if option == '1':  
 data.tableName = "internet\_cafe"  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_internet\_cafe\_name(self)  
 data.internet\_cafe\_name = str(input())  
 view.insertInto\_internet\_cafe\_location(self)  
 data.internet\_cafe\_location = str(input())  
 updateTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '2':  
 data.tableName = "employee"  
 view.insertInto\_employee\_id(self)  
 data.employee\_id = int(input())  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_employee\_surname(self)  
 data.employee\_surname = str(input())  
 view.insertInto\_employee\_age(self)  
 data.employee\_age = int(input())  
 view.insertInto\_employee\_phone(self)  
 data.employee\_phone = str(input())  
 updateTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '3':  
 data.tableName = "computer"  
 view.insertInto\_computer\_id(self)  
 data.computer\_id = int(input())  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_computer\_model(self)  
 data.computer\_model = str(input())  
 view.insertInto\_computer\_isbusy(self)  
 data.computer\_isbusy = bool(input())  
 updateTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '4':  
 data.tableName = "application"  
 view.insertInto\_application\_id(self)  
 data.application\_id = int(input())  
 view.insertInto\_computer\_id(self)  
 data.computer\_id = int(input())  
 view.insertInto\_application\_name(self)  
 data.application\_name = str(input())  
 updateTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '5':  
 data.tableName = "attendee\_card"  
 view.insertInto\_attendee\_card\_owner\_id(self)  
 data.attendee\_card\_owner\_id = int(input())  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 view.insertInto\_attendee\_card\_owner\_surname(self)  
 data.attendee\_card\_owner\_surname = str(input())  
 view.insertInto\_attendee\_card\_owner\_age(self)  
 data.attendee\_card\_owner\_age = int(input())  
 view.insertInto\_attendee\_card\_owner\_phone(self)  
 data.attendee\_card\_owner\_phone = str(input())  
 updateTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 else:  
 view.uncorrectSymbol(self)  
 controller.menu(self)  
 elif option == '4':  
 view.insertTableName(self)  
 option = input()  
 if option == '1':  
 data.tableName = "internet\_cafe"  
 view.insertInto\_internet\_cafe\_id(self)  
 data.internet\_cafe\_id = int(input())  
 deleteFromTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '2':  
 data.tableName = "employee"  
 view.insertInto\_employee\_id(self)  
 data.employee\_id = int(input())  
 deleteFromTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '3':  
 data.tableName = "computer"  
 view.insertInto\_computer\_id(self)  
 data.computer\_id = int(input())  
 deleteFromTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '4':  
 data.tableName = "application"  
 view.insertInto\_application\_id(self)  
 data.application\_id = int(input())  
 deleteFromTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '5':  
 data.tableName = "attendee\_card"  
 view.insertInto\_attendee\_card\_owner\_id(self)  
 data.attendee\_card\_owner\_id = int(input())  
 deleteFromTable()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 else:  
 view.uncorrectSymbol(self)  
 controller.menu(self)  
 elif option == '5':  
 view.insertTableName(self)  
 option = input()  
 if option == '1':  
 data.tableName = "internet\_cafe"  
 view.insertQuantityOfRandomValues(self)  
 data.quant = int(input())  
 genRandIntoTable(data.quant)  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '2':  
 data.tableName = "employee"  
 view.insertQuantityOfRandomValues(self)  
 data.quant = int(input())  
 genRandIntoTable(data.quant)  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '3':  
 data.tableName = "computer"  
 view.insertQuantityOfRandomValues(self)  
 data.quant = int(input())  
 genRandIntoTable(data.quant)  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '4':  
 data.tableName = "application"  
 view.insertQuantityOfRandomValues(self)  
 data.quant = int(input())  
 genRandIntoTable(data.quant)  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '5':  
 data.tableName = "attendee\_card"  
 view.insertQuantityOfRandomValues(self)  
 data.quant = int(input())  
 genRandIntoTable(data.quant)  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 else:  
 view.uncorrectSymbol(self)  
 controller.menu(self)  
 elif option == '6':  
 atrSearch1()  
 atrSearch2()  
 atrSearch3()  
 view.waitingButtonToPress(self)  
 option1 = input()  
 controller.menu(self)  
 elif option == '0':  
 return 0  
 else:  
 view.uncorrectSymbol(self)  
 controller.menu(self)  
  
controller.menu(self)