Изображение выглядит как текст

Автоматически созданное описание

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**”

Виконала:

студентка ІII курсу

групи КВ-94

Романенко М. В.

Перевірив:

Петрашенко А. В.

Київ – 2021

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

*Загальне завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

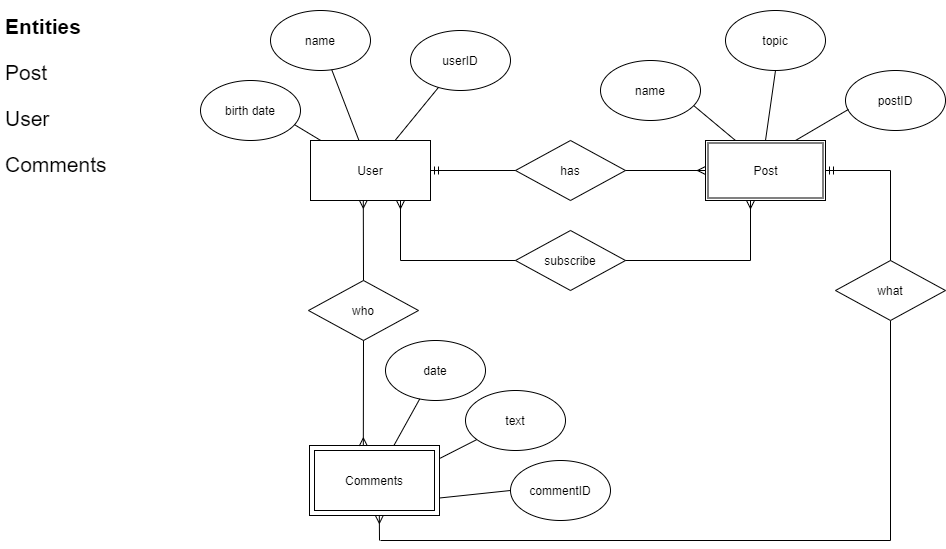
*Деталізоване завдання*:

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](https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/). 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**

<https://github.com/marromanenko/data-base-course>

**Модель «сутність-зв’язок» галузі постів якоїсь соціальної мережи**



*Рисунок 1. ER-діаграма, побудована за нотацією "Пташиної лапки"*

**Сутності, їх призначення та опис зв'язків:**

Маємо три сутності: User, Post, Comments. Сутність User описує користувачів соціальної мережи. Кожен користувач має ім’я, дату народження та свій ID. Сутність Post – це пости цих самих користувачів. Тобто кожен окремий користувач може мати скільки завгодно постів в своєму блозі, але скільки завгодно користувачів можуть бути підписаними на ці пости. Кожний пост також має свій ID, назву поста та тему, на яку написано пост. Третя сутність – це Comments. Це коментарі під постами. Коментар має два зв’язки: чий коментар (користувача) та що це за коментар (тобто під яким саме постом). Багато користувачів можуть написати скільки завгодно коментарів, а також під одним постом може бути багато коментарів. Коментар має свій ID, свою дату та сам текст.

Изображение выглядит как текст, снимок экрана, квитанция

Автоматически созданное описание

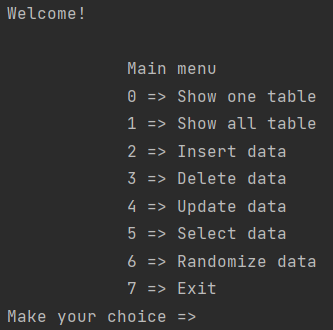
*Рисунок 2. Схема бази даних у графічному вигляді*

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

Мова програмування – Python 3.7

Середовище розробки програмного забезпечення – PyCharm Community Edition.

**Структура меню програми**

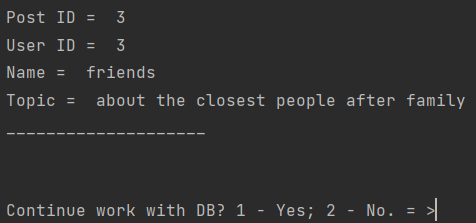


**Завдання 1**

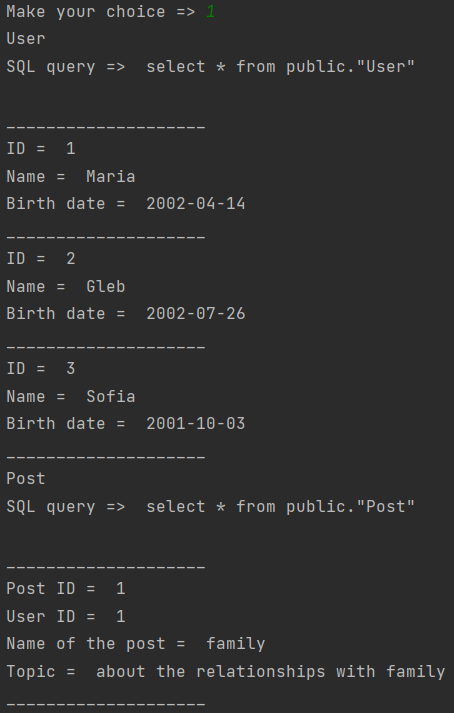
Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

**Перегляд однієї таблиці**

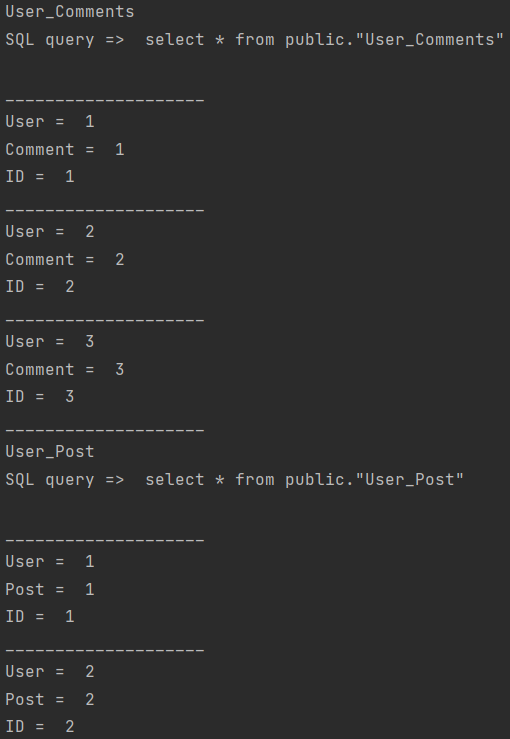


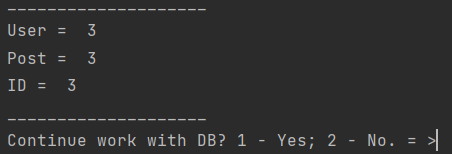


**Перегляд всіх таблиць**



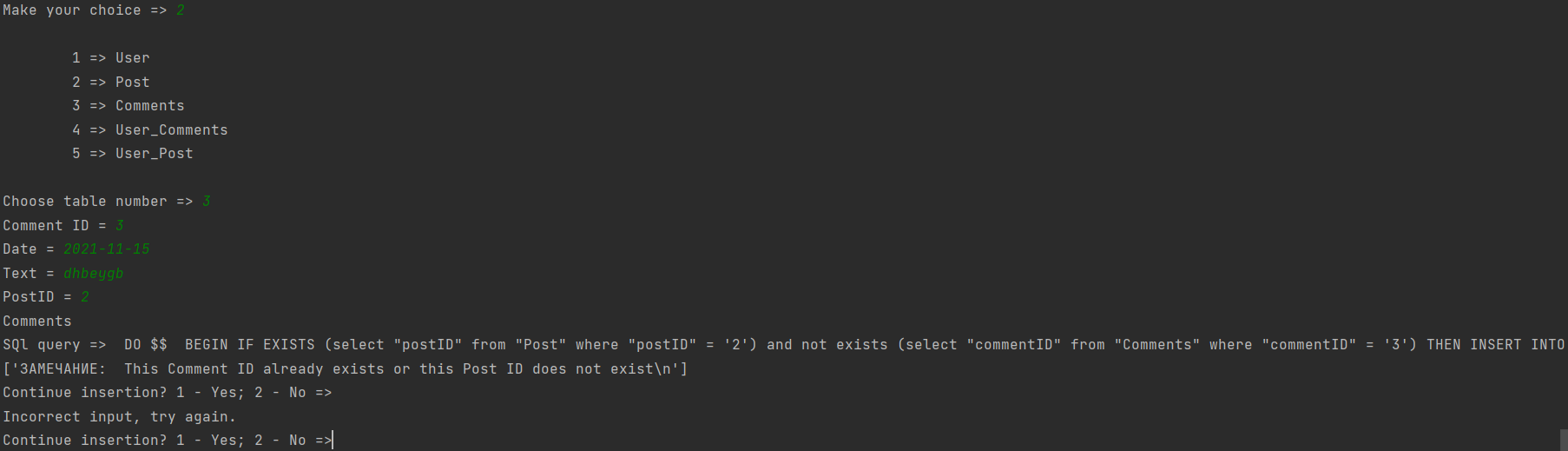




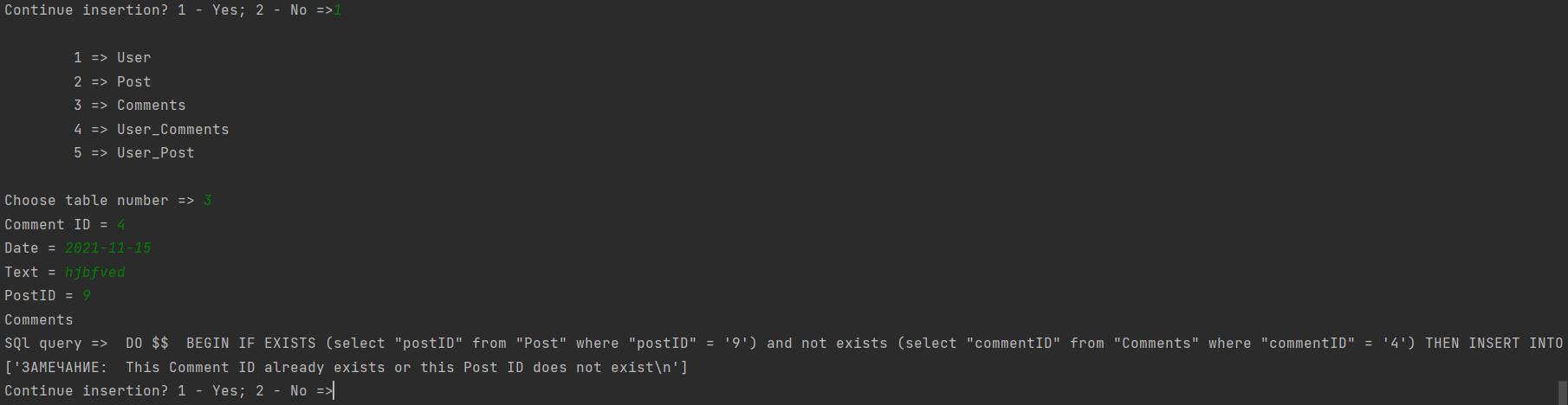


**Внесення даних**

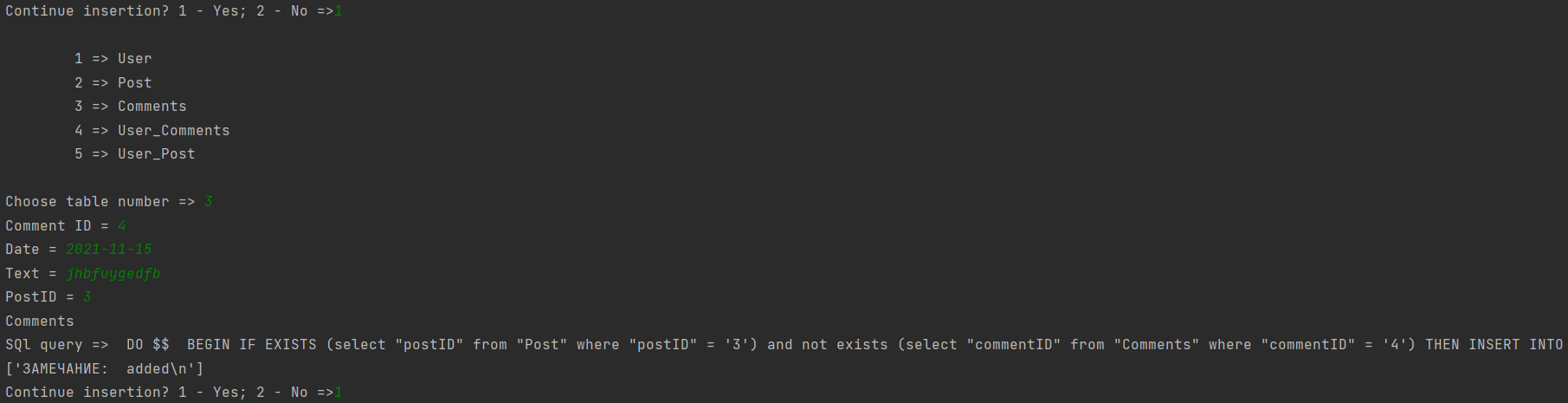
**Якщо такий первинний ключ вже є, виводиться помилка, яка каже користувачу про це:**

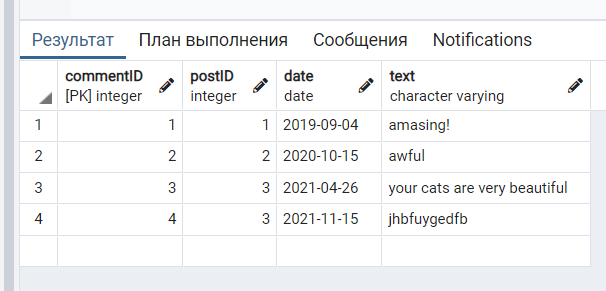


**Якщо такого первічного ключа немає, але також немає такого вторинного ключа, то також виводиться помилка, яка каже користувачеві про це (після виведення помилок програма не зупиняється):**



**Якщо з ключами все добре, програма працює без помилок:**



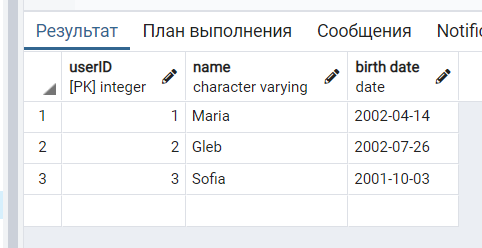


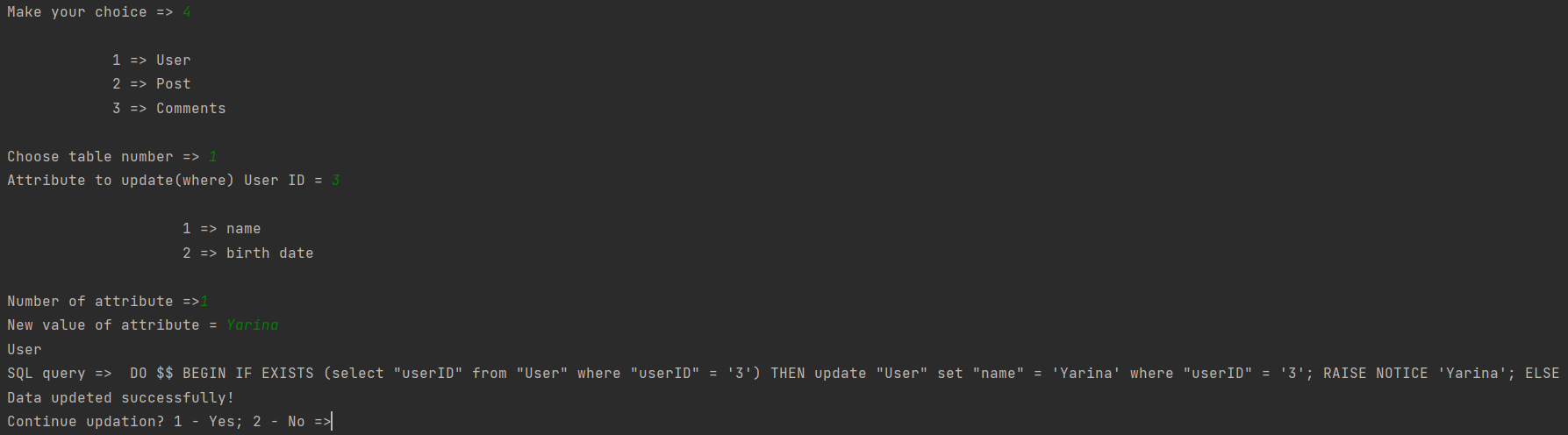
**Лістинг операції insert:**

@staticmethod  
def insert():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 restart = True  
 while restart:  
 View.list()  
 table = Model.validTable()  
 if table == 1:  
 usname = "'" + input("Name = ") + "'"  
 usbirth\_date = "'" + input("Birth date = ") + "'"  
 usid = "'" + input('User ID = ') + "'"  
 notice = "'This User ID already exists'"  
 insert = 'DO $$ BEGIN if not exists (select "userID" from "User" where "userID" = {}) then INSERT ' \  
 'INTO "User"("userID", "name", "birth date") VALUES ({},{},{}); ' \  
 'raise notice {}; else raise notice {}; ' \  
 'end if; end $$;'.format(usid, usid, usname, usbirth\_date, "'added'", notice)  
 restart = False  
 elif table == 2:  
 poid = "'" + input('ID = ') + "'"  
 poname = "'" + input('Name of the post = ') + "'"  
 potopic = "'" + input('Topic = ') + "'"  
 pouser = "'" + input('User ID = ') + "'"  
 notice = "'This Post ID already exists or this User ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) and not exists ' \  
 '(select "postID" from "Post" where "postID" = {}) THEN ' \  
 'INSERT INTO "Post"("postID", "namepost", "topic", "userID") values ({}, {}, {}, {}); RAISE NOTICE {};' \  
 ' ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(pouser, poid, poid, poname, potopic, pouser, "'added'", notice)  
 restart = False  
 elif table == 3:  
 coid = "'" + input('Comment ID = ') + "'"  
 codate = "'" + input('Date = ') + "'"  
 cotext = "'" + input('Text = ') + "'"  
 copost = "'" + input('PostID = ') + "'"  
 notice = "'This Comment ID already exists or this Post ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "postID" from "Post" where "postID" = {}) and not exists ' \  
 '(select "commentID" from "Comments" where "commentID" = {}) THEN ' \  
 'INSERT INTO "Comments"("commentID", "date", "text", "postID") values ({}, {}, {}, {}); RAISE NOTICE {};' \  
 ' ELSE RAISE NOTICE {}; END IF; END $$;'.format(copost, coid, coid, codate, cotext, copost, "'added'", notice)  
 restart = False  
 elif table == 4:  
 ucid = "'" + input('ID = ') + "'"  
 uccomment = "'" + input('Comment = ') + "'"  
 ucuser = "'" + input('User = ') + "'"  
 notice = "'This ID already exists or this User ID/Comment ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) and ' \  
 'EXISTS (select "commentID" from "Comments" where "commentID" = {}) and ' \  
 'not exists (select "ID" from "User\_Comments" where "ID" = {}) THEN ' \  
 'INSERT INTO "User\_Comments"("ID", "User\_userID", "Comments\_commentID") VALUES ({},{},{}); ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(ucuser, uccomment, ucid, ucid, ucuser, uccomment, "'added'", notice)  
 restart = False  
 elif table == 5:  
 upid = "'" + input('ID = ') + "'"  
 upuser = "'" + input('User = ') + "'"  
 uppost = "'" + input('Post = ') + "'"  
 notice = "'This ID already exists or this User ID/Comment ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) and ' \  
 'EXISTS (select "postID" from "Post" where "postID" = {}) and ' \  
 'not exists (select "ID" from "User\_Post" where "ID" = {}) THEN ' \  
 'INSERT INTO "User\_Post"("ID", "User\_userID", "Post\_postID") VALUES ({},{},{}); ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(upuser, uppost, upid, upid, upuser, uppost, "'added'", notice)  
 restart = False  
 else:  
 print('Incorrect input, try again.')  
 print(tables[table])  
 print('SQl query => ', insert)  
 cursor.execute(insert)  
 connection.commit()  
 print(connection.notices)  
 cursor.close()  
 controller.closeConnect(connection)

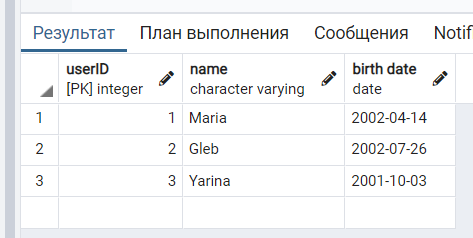
**Редагування даних**

**Таблиця до:**

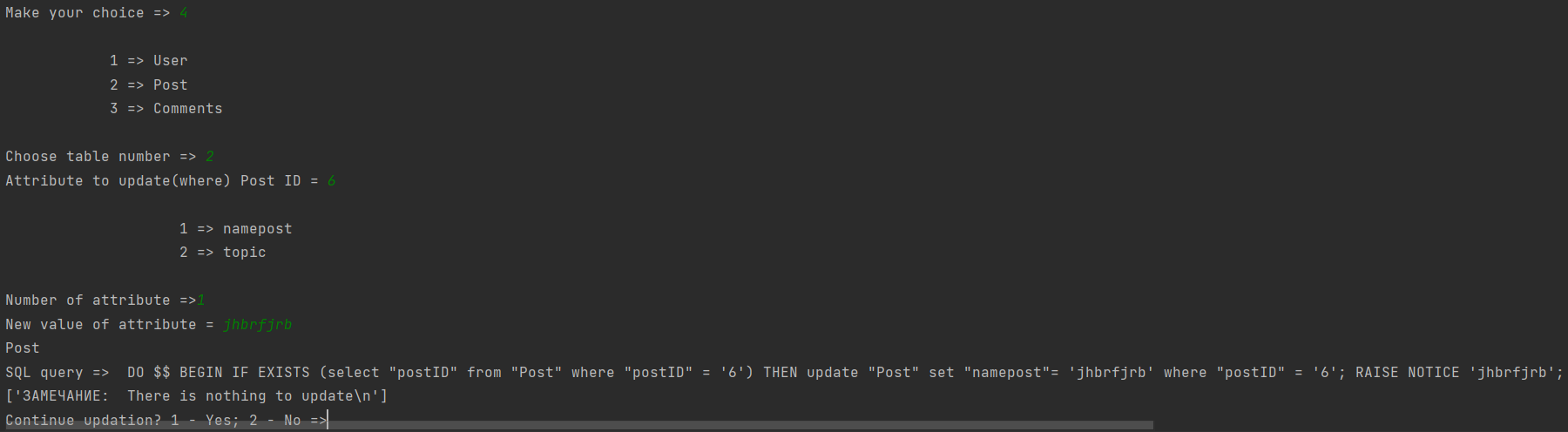




**Таблиця після:**



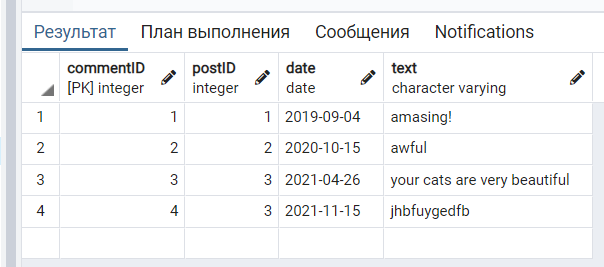
**Якщо заданого користувачем айді немає, виводиться помилка.**

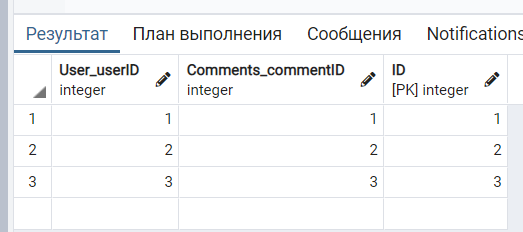


**Вилучення даних**

**В мене є таблиця Comments та одна її дочірня таблиця User\_Comments**

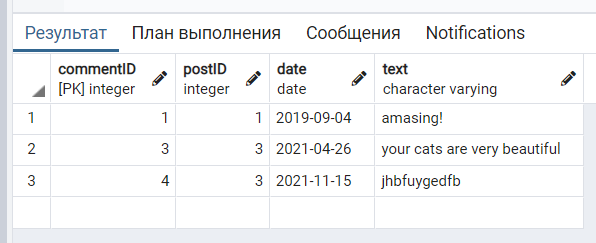
**Таблиці до:**

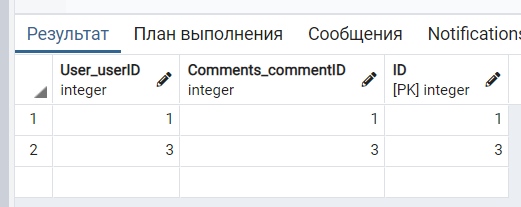






**Таблиця після:**





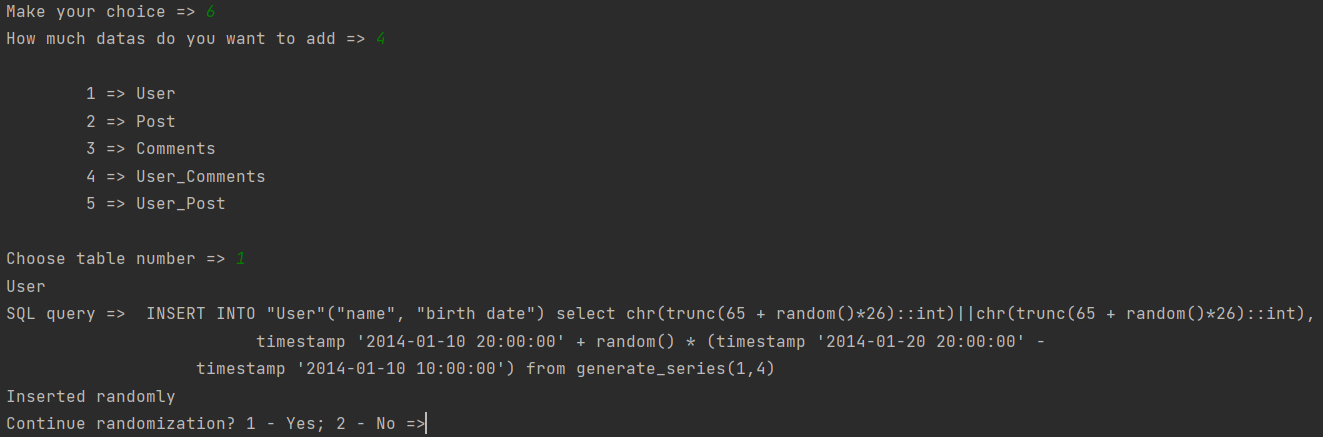
**Так само програма працює для всіх зв’язків між батьківськими та дочірніми таблицями.**

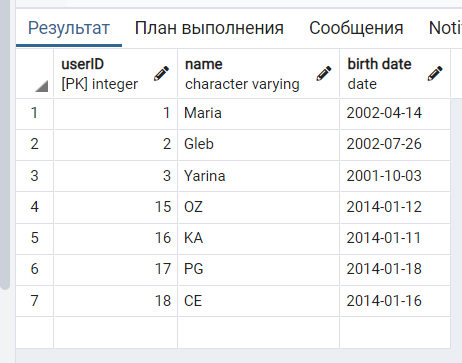
**Лістинг операції delete:**

@staticmethod  
def delete():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 restart = True  
 while restart:  
 View.list()  
 table = Model.validTable()  
 if table == 1:  
 usid = "'" + input('Attribute to delete User ID = ') + "'"  
 delete = 'delete from "User\_Post" where "User\_userID" = {};' \  
 'delete from "User\_Comments" where "User\_userID" = {};' \  
 'delete from "Comments" where "postID" in (select "postID" from "Post" where "userID" = {});' \  
 'delete from "Post" where "userID" = {};' \  
 'delete from "User" where "userID" = {};'.format(usid, usid, usid, usid, usid)  
 restart = False  
 elif table == 2:  
 poid = "'" + input('Attribute to delete Post ID = ') + "'"  
 delete = 'delete from "User\_Comments" where "Comments\_commentID" in (select "commentID" from "Comments" where "postID" = {});' \  
 'delete from "User\_Post" where "Post\_postID" = {};'\  
 'delete from "Comments" where "postID" = {};' \  
 'delete from "Post" where "postID" = {};'.format(poid, poid, poid, poid)  
 restart = False  
 elif table == 3:  
 coid = "'" + input('Attribute to delete Comment ID = ') + "'"  
 delete = 'delete from "User\_Comments" where "Comments\_commentID" = {};' \  
 'delete from "Comments" where "commentID" = {};'.format(coid, coid)  
 restart = False  
 elif table == 4:  
 ucid = "'" + input('Attribute to delete ID = ') + "'"  
 delete = 'delete from "User\_Comments" where "ID" = {}'.format(ucid)  
 restart = False  
 elif table == 5:  
 upid = "'" + input('Attribute to delete ID = ') + "'"  
 delete = 'delete from "User\_Post" where "ID" = {}'.format(upid)  
 restart = False  
 else:  
 print('Incorrect input, try again.')  
 print(tables[table])  
 print("SQL query => ", delete)  
 cursor.execute(delete)  
 connection.commit()  
 print('Data deleted successfully!')  
 cursor.close()  
 controller.closeConnect(connection)

**Завдання 2**

Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.





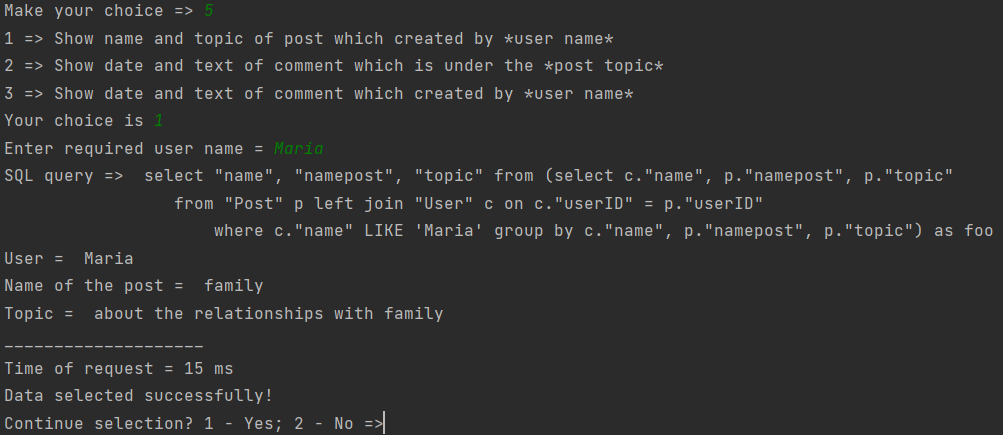
SQL query => INSERT INTO "User"("name", "birth date") select chr(trunc(65 + random()\*26)::int)||chr(trunc(65 + random()\*26)::int),

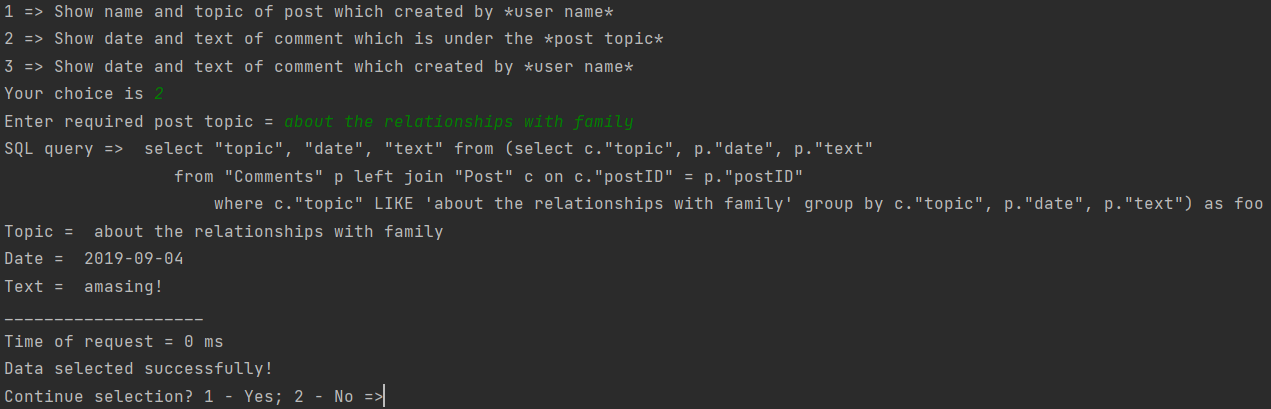
timestamp '2014-01-10 20:00:00' + random() \* (timestamp '2014-01-20 20:00:00' -

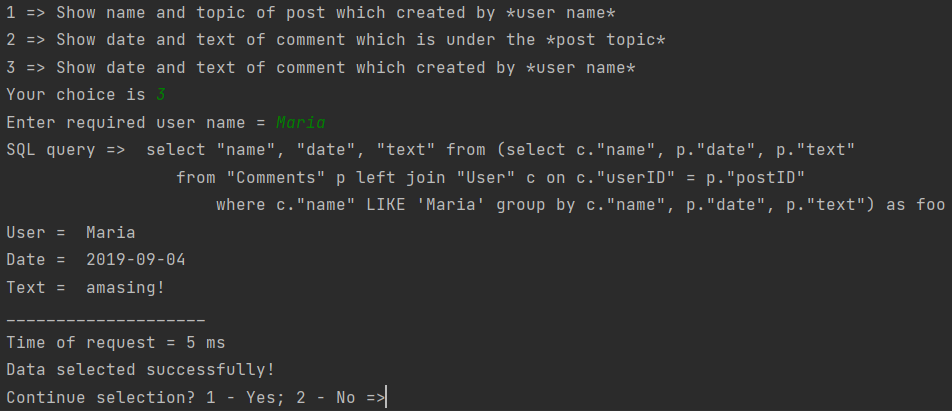
timestamp '2014-01-10 10:00:00') from generate\_series(1,4)

**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.

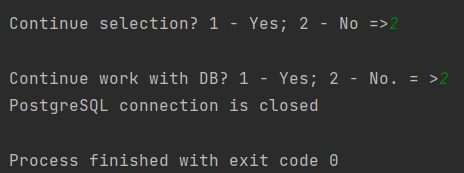






**Всі SQL-запити прописані в консолі на ілюстраціях.**

**Вихід із програми**



**Програмний модуль model.py**

**Цей програмний модуль відповідає за всю ЛОГІКУ проекта.**

import controller  
import time  
from view import View  
  
tables = {  
 1: 'User',  
 2: 'Post',  
 3: 'Comments',  
 4: 'User\_Comments',  
 5: 'User\_Post',  
}  
  
  
class Model:  
 @staticmethod  
 def validTable():  
 incorrect = True  
 while incorrect:  
 table = input('Choose table number => ')  
 if table.isdigit():  
 table = int(table)  
 if table >= 1 and table <= 5:  
 incorrect = False  
 else:  
 print('Incorrect input, try again.')  
 else:  
 print('Incorrect input, try again.')  
 return table  
  
 @staticmethod  
 def showAllTables():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 for table in range(1, 6):  
 table\_name = '''"''' + tables[table] + '''"'''  
 print(tables[table])  
 show = 'select \* from public.{}'.format(table\_name)  
 print("SQL query => ", show)  
 print('')  
 cursor.execute(show)  
 records = cursor.fetchall()  
 obj = View(table, records)  
 obj.show()  
 cursor.close()  
 controller.closeConnect(connection)  
  
 @staticmethod  
 def showOneTable():  
 View.list()  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 table = Model.validTable()  
 table\_name = '''"''' + tables[table] + '''"'''  
 print(tables[table])  
 show = 'select \* from public.{}'.format(table\_name)  
 print("SQL query => ", show)  
 print('')  
 cursor.execute(show)  
 records = cursor.fetchall()  
 obj = View(table, records)  
 obj.show()  
 cursor.close()  
 controller.closeConnect(connection)  
  
 @staticmethod  
 def insert():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 restart = True  
 while restart:  
 View.list()  
 table = Model.validTable()  
 if table == 1:  
 usname = "'" + input("Name = ") + "'"  
 usbirth\_date = "'" + input("Birth date = ") + "'"  
 usid = "'" + input('User ID = ') + "'"  
 notice = "'This User ID already exists'"  
 insert = 'DO $$ BEGIN if not exists (select "userID" from "User" where "userID" = {}) then INSERT ' \  
 'INTO "User"("userID", "name", "birth date") VALUES ({},{},{}); ' \  
 'raise notice {}; else raise notice {}; ' \  
 'end if; end $$;'.format(usid, usid, usname, usbirth\_date, "'added'", notice)  
 restart = False  
 elif table == 2:  
 poid = "'" + input('ID = ') + "'"  
 poname = "'" + input('Name of the post = ') + "'"  
 potopic = "'" + input('Topic = ') + "'"  
 pouser = "'" + input('User ID = ') + "'"  
 notice = "'This Post ID already exists or this User ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) and not exists ' \  
 '(select "postID" from "Post" where "postID" = {}) THEN ' \  
 'INSERT INTO "Post"("postID", "namepost", "topic", "userID") values ({}, {}, {}, {}); RAISE NOTICE {};' \  
 ' ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(pouser, poid, poid, poname, potopic, pouser, "'added'", notice)  
 restart = False  
 elif table == 3:  
 coid = "'" + input('Comment ID = ') + "'"  
 codate = "'" + input('Date = ') + "'"  
 cotext = "'" + input('Text = ') + "'"  
 copost = "'" + input('PostID = ') + "'"  
 notice = "'This Comment ID already exists or this Post ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "postID" from "Post" where "postID" = {}) and not exists ' \  
 '(select "commentID" from "Comments" where "commentID" = {}) THEN ' \  
 'INSERT INTO "Comments"("commentID", "date", "text", "postID") values ({}, {}, {}, {}); RAISE NOTICE {};' \  
 ' ELSE RAISE NOTICE {}; END IF; END $$;'.format(copost, coid, coid, codate, cotext, copost, "'added'", notice)  
 restart = False  
 elif table == 4:  
 ucid = "'" + input('ID = ') + "'"  
 uccomment = "'" + input('Comment = ') + "'"  
 ucuser = "'" + input('User = ') + "'"  
 notice = "'This ID already exists or this User ID/Comment ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) and ' \  
 'EXISTS (select "commentID" from "Comments" where "commentID" = {}) and ' \  
 'not exists (select "ID" from "User\_Comments" where "ID" = {}) THEN ' \  
 'INSERT INTO "User\_Comments"("ID", "User\_userID", "Comments\_commentID") VALUES ({},{},{}); ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(ucuser, uccomment, ucid, ucid, ucuser, uccomment, "'added'", notice)  
 restart = False  
 elif table == 5:  
 upid = "'" + input('ID = ') + "'"  
 upuser = "'" + input('User = ') + "'"  
 uppost = "'" + input('Post = ') + "'"  
 notice = "'This ID already exists or this User ID/Comment ID does not exist'"  
 insert = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) and ' \  
 'EXISTS (select "postID" from "Post" where "postID" = {}) and ' \  
 'not exists (select "ID" from "User\_Post" where "ID" = {}) THEN ' \  
 'INSERT INTO "User\_Post"("ID", "User\_userID", "Post\_postID") VALUES ({},{},{}); ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(upuser, uppost, upid, upid, upuser, uppost, "'added'", notice)  
 restart = False  
 else:  
 print('Incorrect input, try again.')  
 print(tables[table])  
 print('SQl query => ', insert)  
 cursor.execute(insert)  
 connection.commit()  
 print(connection.notices)  
 cursor.close()  
 controller.closeConnect(connection)  
  
 @staticmethod  
 def delete():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 restart = True  
 while restart:  
 View.list()  
 table = Model.validTable()  
 if table == 1:  
 usid = "'" + input('Attribute to delete User ID = ') + "'"  
 delete = 'delete from "User\_Post" where "User\_userID" = {};' \  
 'delete from "User\_Comments" where "User\_userID" = {};' \  
 'delete from "Comments" where "postID" in (select "postID" from "Post" where "userID" = {});' \  
 'delete from "Post" where "userID" = {};' \  
 'delete from "User" where "userID" = {};'.format(usid, usid, usid, usid, usid)  
 restart = False  
 elif table == 2:  
 poid = "'" + input('Attribute to delete Post ID = ') + "'"  
 delete = 'delete from "User\_Comments" where "Comments\_commentID" in (select "commentID" from "Comments" where "postID" = {});' \  
 'delete from "User\_Post" where "Post\_postID" = {};'\  
 'delete from "Comments" where "postID" = {};' \  
 'delete from "Post" where "postID" = {};'.format(poid, poid, poid, poid)  
 restart = False  
 elif table == 3:  
 coid = "'" + input('Attribute to delete Comment ID = ') + "'"  
 delete = 'delete from "User\_Comments" where "Comments\_commentID" = {};' \  
 'delete from "Comments" where "commentID" = {};'.format(coid, coid)  
 restart = False  
 elif table == 4:  
 ucid = "'" + input('Attribute to delete ID = ') + "'"  
 delete = 'delete from "User\_Comments" where "ID" = {}'.format(ucid)  
 restart = False  
 elif table == 5:  
 upid = "'" + input('Attribute to delete ID = ') + "'"  
 delete = 'delete from "User\_Post" where "ID" = {}'.format(upid)  
 restart = False  
 else:  
 print('Incorrect input, try again.')  
 print(tables[table])  
 print("SQL query => ", delete)  
 cursor.execute(delete)  
 connection.commit()  
 print('Data deleted successfully!')  
 cursor.close()  
 controller.closeConnect(connection)  
  
 @staticmethod  
 def update():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 restart = True  
 while restart:  
 View.list\_for\_update()  
 table = Model.validTable()  
 if table == 1:  
 usid = "'" + input('Attribute to update(where) User ID = ') + "'"  
 View.attribute\_list\_for\_update(1)  
 in\_restart = True  
 while in\_restart:  
 num = input('Number of attribute =>')  
 if num == '1':  
 value = "'" + input('New value of attribute = ') + "'"  
 set = '"name" = {}'.format(value)  
 in\_restart = False  
 elif num == '2':  
 value = "'" + input('New value of attribute = ') + "'"  
 set = '"birth date" = {}'.format(value)  
 in\_restart = False  
 else:  
 print('Incorrect input, try again.')  
 notice = "'There is nothing to update'"  
 update = 'DO $$ BEGIN IF EXISTS (select "userID" from "User" where "userID" = {}) THEN ' \  
 'update "User" set {} where "userID" = {}; ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(usid, set, usid, value, notice)  
 restart = False  
 pass  
 elif table == 2:  
 poid = "'" + input('Attribute to update(where) Post ID = ') + "'"  
 View.attribute\_list\_for\_update(2)  
 in\_restart = True  
 while in\_restart:  
 num = input('Number of attribute =>')  
 if num == '1':  
 value = "'" + input('New value of attribute = ') + "'"  
 set = '"namepost"= {}'.format(value)  
 in\_restart = False  
 elif num == '2':  
 value = "'" + input('New value of attribute = ') + "'"  
 set = '"topic"= {}'.format(value)  
 in\_restart = False  
 else:  
 print('Incorrect input, try again.')  
 notice = "'There is nothing to update'"  
 update = 'DO $$ BEGIN IF EXISTS (select "postID" from "Post" where "postID" = {}) THEN ' \  
 'update "Post" set {} where "postID" = {}; ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(poid, set, poid, value, notice)  
 restart = False  
 pass  
 elif table == 3:  
 coid = "'" + input('Attribute to update(where) Comment ID = ') + "'"  
 View.attribute\_list\_for\_update(3)  
 in\_restart = True  
 while in\_restart:  
 num = input('Number of attribute =>')  
 if num == '1':  
 value = "'" + input('New value of attribute = ') + "'"  
 set = '"date"= {}'.format(value)  
 in\_restart = False  
 elif num == '2':  
 value = "'" + input('New value of attribute = ') + "'"  
 set = '"text"= {}'.format(value)  
 in\_restart = False  
 else:  
 print('Incorrect input, try again.')  
 notice = "'There is nothing to update'"  
 update = 'DO $$ BEGIN IF EXISTS (select "commentID" from "Comments" where "commentID" = {}) THEN ' \  
 'update "Comments" set {} where "commentID" = {}; ' \  
 'RAISE NOTICE {}; ELSE RAISE NOTICE {}; END IF; ' \  
 'END $$;'.format(coid, set, coid, value, notice)  
 restart = False  
 pass  
 else:  
 print('Incorrect input, try again.')  
 print(tables[table])  
 print("SQL query => ", update)  
 cursor.execute(update)  
 connection.commit()  
 print('Data updeted successfully!')  
 cursor.close()  
 controller.closeConnect(connection)  
 pass  
  
 @staticmethod  
 def select():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 print('1 => Show name and topic of post which created by \*user name\*')  
 print('2 => Show date and text of comment which is under the \*post topic\*')  
 print('3 => Show date and text of comment which created by \*user name\*')  
 choice = input('Your choice is ')  
 choice = int(choice)  
 if choice == 1:  
 user = input('Enter required user name = ')  
 select = """select "name", "namepost", "topic" from (select c."name", p."namepost", p."topic"  
 from "Post" p left join "User" c on c."userID" = p."userID"  
 where c."name" LIKE '{}' group by c."name", p."namepost", p."topic") as foo""".format(user)  
 elif choice == 2:  
 post = input('Enter required post topic = ')  
 select = """select "topic", "date", "text" from (select c."topic", p."date", p."text"  
 from "Comments" p left join "Post" c on c."postID" = p."postID"  
 where c."topic" LIKE '{}' group by c."topic", p."date", p."text") as foo""".format(post)  
 elif choice == 3:  
 user = input('Enter required user name = ')  
 select = """select "name", "date", "text" from (select c."name", p."date", p."text"  
 from "Comments" p left join "User" c on c."userID" = p."postID"  
 where c."name" LIKE '{}' group by c."name", p."date", p."text") as foo""".format(user)  
 else:  
 print('Try again')  
 print("SQL query => ", select)  
 beg = int(time.time() \* 1000)  
 cursor.execute(select)  
 end = int(time.time() \* 1000) - beg  
 records = cursor.fetchall()  
 obj = View(choice, records)  
 obj.showSelect()  
 print('Time of request = {} ms'.format(end))  
 print('Data selected successfully!')  
 cursor.close()  
 controller.closeConnect(connection)  
  
 @staticmethod  
 def random():  
 connection = controller.makeConnect()  
 cursor = connection.cursor()  
 incorrect = True  
 while incorrect:  
 num = input('How much datas do you want to add => ')  
 num = int(num)  
 View.list()  
 table = Model.validTable()  
 if table == 1:  
 insert = """INSERT INTO "User"("name", "birth date") select chr(trunc(65 + random()\*26)::int)||chr(trunc(65 + random()\*26)::int),   
 timestamp '2014-01-10 20:00:00' + random() \* (timestamp '2014-01-20 20:00:00' -  
 timestamp '2014-01-10 10:00:00') from generate\_series(1,{})""".format(num)  
 incorrect = False  
 elif table == 2:  
 insert = """INSERT INTO "Post" ("namepost", "topic") select chr(trunc(65 + random()\*26)::int)||chr(trunc(65 + random()\*26)::int),   
 chr(trunc(65 + random()\*26)::int)||chr(trunc(65 + random()\*26)::int)  
 from generate\_series(1,{})""".format(num)  
 incorrect = False  
 elif table == 3:  
 insert = """INSERT INTO "Comments"("text", "date") select chr(trunc(65 + random()\*26)::int)||chr(trunc(65 + random()\*26)::int),   
 timestamp '2014-01-10 20:00:00' + random() \* (timestamp '2014-01-20 20:00:00' -  
 timestamp '2014-01-10 10:00:00') from generate\_series(1, {})""".format(num)  
 incorrect = False  
 elif table == 4:  
 insert = "INSERT INTO User\_Comments select " \  
 "from generate\_series(1,{})".format(num)  
 incorrect = False  
 elif table == 5:  
 insert = "INSERT INTO User\_Post select " \  
 "from generate\_series(1,{})".format(num)  
 incorrect = False  
 else:  
 print('Incorrect input, try again.')  
 print(tables[table])  
 print("SQL query => ", insert)  
 cursor.execute(insert)  
 connection.commit()  
 print('Inserted randomly')  
 cursor.close()  
 controller.closeConnect(connection)

**Маємо словник tables для виведення та обирання таблиць за певним числом. Також клас Model, який включає всі потрібні функції:**

**validTable – функція, яка переводить рядок у число, перевіряє наявність такої таблиці та повертає її;**

**showAllTables – функція, яка виводить всі таблиці;**

**showOneTable – функція, яка виводить тільки задану таблицю;**

**insert – функція, яка виконує операцію внесення даних;**

**delete – функція, яка виконує операцію вилучення даних;**

**update – функція, яка виконує операцію редагування даних;**

**select – функція, яка виконує 3 пошукові запити на вибір користувача;**

**random – функція, яка рандомно генерує дані таблиці.**