**Національний технічний університет України “Київський політехнічний інститут ім. Ігоря Сікорського”**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп’ютерних систем**

**Розрахункова графічна робота**

**“Створення додатку бази даних,**

**орієнтованого на взаємодію з СУБД PostgreSQL”**

**Група: КВ-11**

**Виконав:Овчінніков Д.С.**

**Телеграм @dmytro\_ovchinnikov**

**Оцінка:**

**Київ – 2023**

Посилання на репозиторій <https://github.com/OvchinnikovDmytro/DataBase>

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

Загальне завдання роботи полягає у наступному:

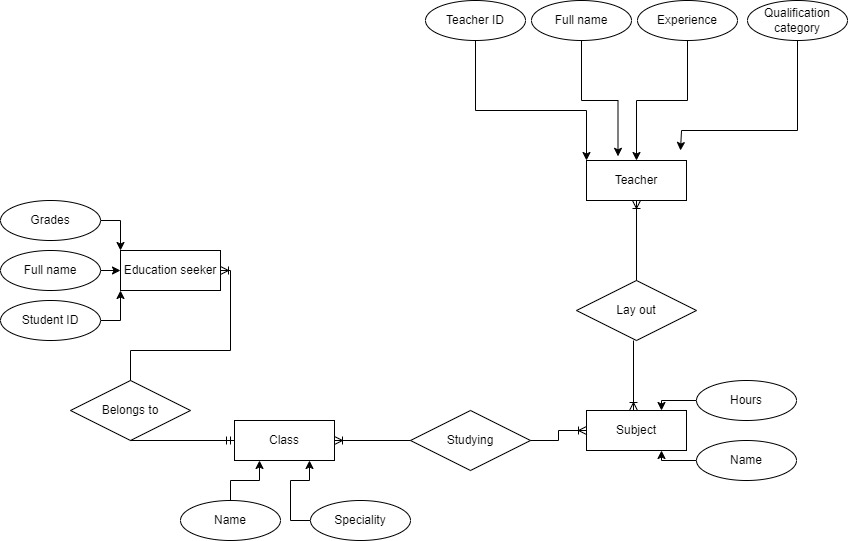
1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами 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).

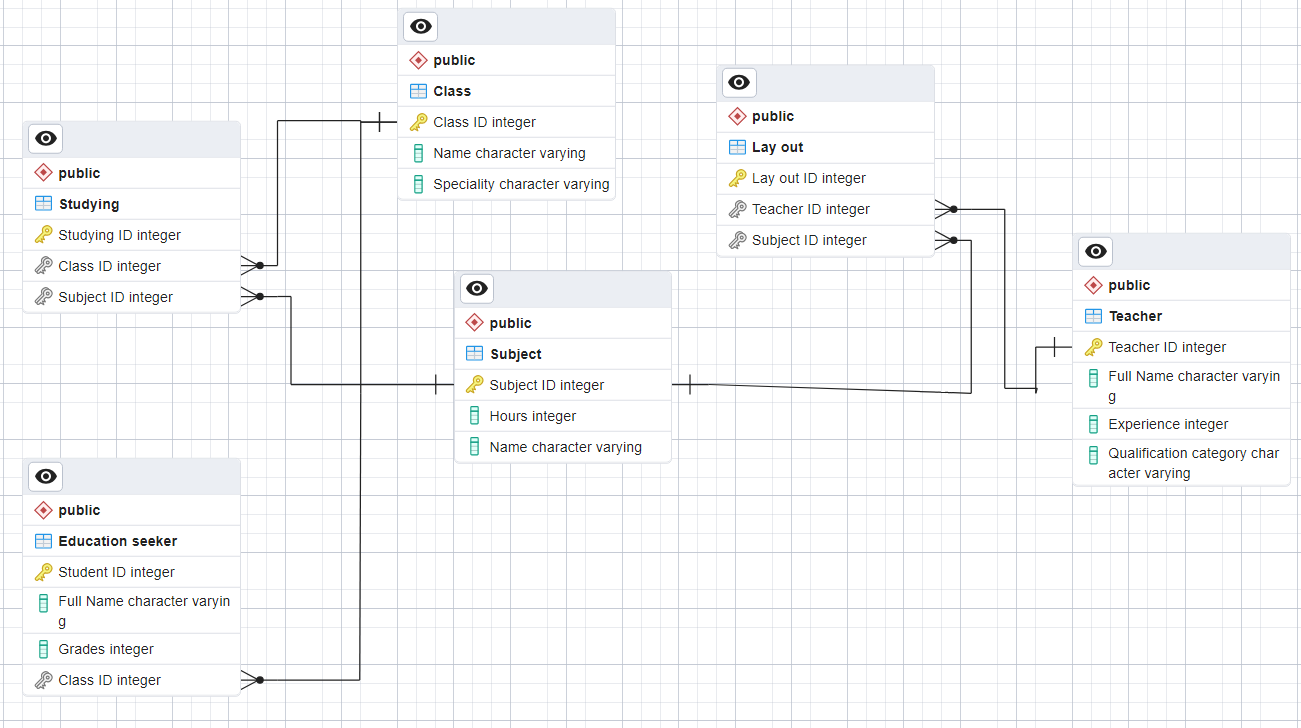
ER модель «Шкільна система управління навчанням.»

Перелік сутностей і опис їх призначення:   
***Здобувач освіти (Education seeker)***, сутність призначена для збереження даних про здобувача освіти – ПІБ, середня оцінка, та ID учнівського,ID класу.  
***Клас (Class)***, сутність призначена для збереження даних про клас здобувачів освіти для відслідковування року їх навчання та предмету їх поглибленого вивчення.  
***Предмет (Subject)***, сутність призначена для збереження даних про предмети що вивчаються в школі – назву предмету, годин вивчення предмета.

***Вчитель (Teacher)***, сутність призначена для збереження даних про вчителя – ПІБ, Стаж, та Кваліфікаційна категорія.

Здобувач може навчатися тільки в одному класі. Для одного класу викладається декілька предметів і предмети викладаються для декількох класів. Вчитель може викладати декілька предметів і викладати ці ж предмети може і інший вчитель.



****

**Меню користувача**

1. Class – взаємодія з таблицею Class
   1. Add Class – додати значення до таблиці
   2. View Class – вивести всі значення таблиці
   3. Update Class – оновити значення таблиці
   4. Delete Class – видалити значення таблиці
   5. Generate Class – згенерувати випадкові дані в таблицю
   6. Find Class – знайти деякий клас
   7. Quit - вийти
2. Education seeker – взаємодія з таблицею Education seeker
   1. Add Education seeker – додати значення до таблиці
   2. View Education seeker – вивести всі значення таблиці
   3. Update Education seeker – оновити значення таблиці
   4. Delete Education seeker – видалити значення таблиці
   5. Generate Education seeker – згенерувати випадкові дані в таблицю
   6. Find Education seeker – знайти деякого учня
   7. Quit – вийти
3. Subject – взаємодія з таблицею Subject
   1. Add Subject – додати значення до таблиці
   2. View Subject – вивести всі значення таблиці
   3. Update Subject – оновити значення таблиці
   4. Delete Subject – видалити значення таблиці
   5. Generate Subject – згенерувати випадкові дані в таблицю
   6. Find Subject – знайти деякий предмет
   7. Quit – вийти
4. Teacher – взаємодія з таблицею Teacher
   1. Add Teacher – додати значення до таблиці
   2. View Teacher – вивести всі значення таблиці
   3. Update Teacher – оновити значення таблиці
   4. Delete Teacher – видалити значення таблиці
   5. Generate Teacher – згенерувати випадкові дані в таблицю
   6. Find Teacher – знайти деякого вчителя
   7. Quit – вийти
5. Lay out – взаємодія з таблицею Lay out
   1. Add Lay out – додати значення до таблиці
   2. View Lay out – вивести всі значення таблиці
   3. Update Lay out – оновити значення таблиці
   4. Delete Lay out – видалити значення таблиці
   5. Generate Lay out – згенерувати випадкові дані в таблицю
   6. Quit – вийти
6. Studying – взаємодія з таблицею Studying
   1. Add Studying – додати значення до таблиці
   2. View Teacher – вивести всі значення таблиці
   3. Update Studying – оновити значення таблиці
   4. Delete Studying – видалити значення таблиці
   5. Generate Studying – згенерувати випадкові дані в таблицю
   6. Quit – вийти
7. Find – знайти записи за певними критеріями
8. Exit – вихід з програми

**Пункт №1**

Початкові таблиці

Classes:

ID: 1, Name: 11, Speciality: Math

Education seekers:

ID: 28, Full Name: Oleksa Oleksandr Olga, Grade: 9, Class ID 1

Операція вилучення запису батьківської таблиці

Menu:

1. Class

2. Education Seeker

3. Subject

4. Teacher

5. Lay out

6. Studying

7. Find

8. Exit

Enter your choice: 1

Class Menu:

1. Add Class

2. View Class

3. Update Class

4. Delete Class

5. Generate Class

6. Find Class

7. Quit

Enter your choice: 4

Enter class ID: 1

Class is not deleted.Delete all student from it first

Клас не видалився, бо до класу був прив'язаний здобувач освіти,після видалення його можна буде видалити і клас.

Спроба додати запис в дочірню таблицю, без відповідного запису в батьківській:

Menu:

1. Class

2. Education Seeker

3. Subject

4. Teacher

5. Lay out

6. Studying

7. Find

8. Exit

Enter your choice: 2

Education seeker Menu:

1. Add Education seeker

2. View Education seeker

3. Update Education seeker

4. Delete Education seeker

5. Generate Education seeker

6. Find Education seeker

7. Quit

Enter your choice: 1

Enter Student Full Name: Vasyl Vovchenko

Enter Student grades: 10

Enter Student class: 3

Education seeker isnt added.There is no such a class id

Здобувач освіти не був доданий так як не існує класу з заданим айді

**Пункт №2**

Генерація 100 записів в таблиці Class

Classes:

ID: 1, Name: 11, Speciality: Math

ID: 2, Name: 9, Speciality: Art

ID: 3, Name: 7, Speciality: Ukrainian

ID: 4, Name: 12, Speciality: Literature

ID: 5, Name: 12, Speciality: History

ID: 6, Name: 9, Speciality: Art

ID: 7, Name: 12, Speciality: Ukrainian

ID: 8, Name: 7, Speciality: Ukrainian

ID: 9, Name: 8, Speciality: Ukrainian

ID: 10, Name: 10, Speciality: Ukrainian

ID: 11, Name: 7, Speciality: Science

ID: 12, Name: 11, Speciality: Science

ID: 13, Name: 7, Speciality: Math

ID: 14, Name: 7, Speciality: Science

ID: 15, Name: 8, Speciality: History

ID: 16, Name: 8, Speciality: Science

ID: 17, Name: 11, Speciality: Art

ID: 18, Name: 9, Speciality: Literature

ID: 19, Name: 7, Speciality: Literature

ID: 20, Name: 9, Speciality: Science

ID: 21, Name: 9, Speciality: Art

ID: 22, Name: 12, Speciality: Science

……………………………………………

ID: 90, Name: 12, Speciality: Literature

ID: 91, Name: 12, Speciality: History

ID: 92, Name: 9, Speciality: Science

ID: 93, Name: 9, Speciality: History

ID: 94, Name: 10, Speciality: Science

ID: 95, Name: 11, Speciality: History

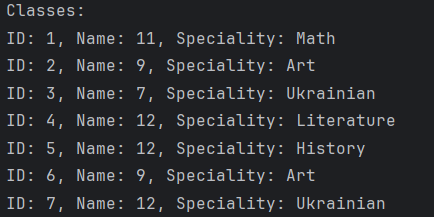
ID: 96, Name: 10, Speciality: Literature

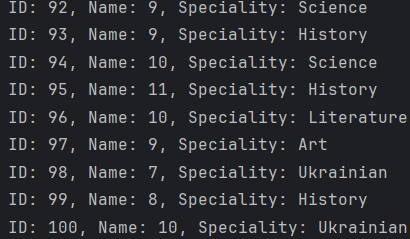
ID: 97, Name: 9, Speciality: Art

ID: 98, Name: 7, Speciality: Ukrainian

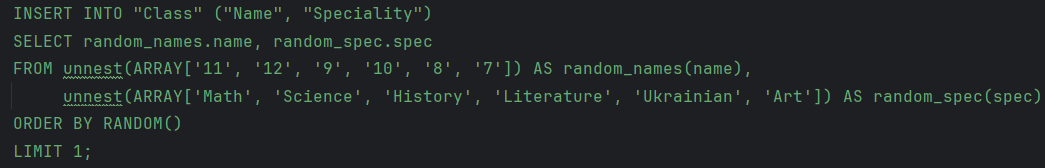
ID: 99, Name: 8, Speciality: History

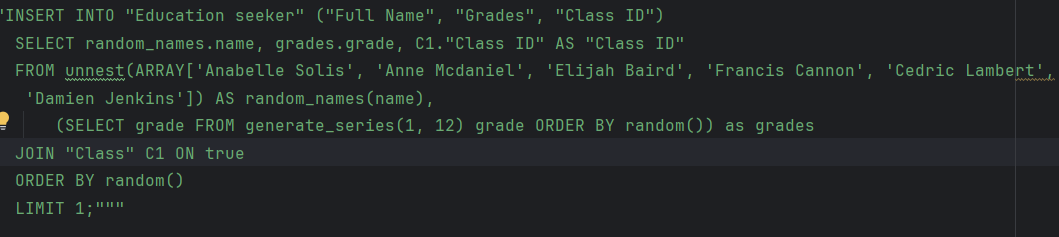
ID: 100, Name: 10, Speciality: Ukrainian

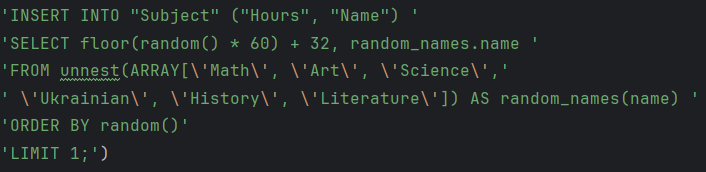


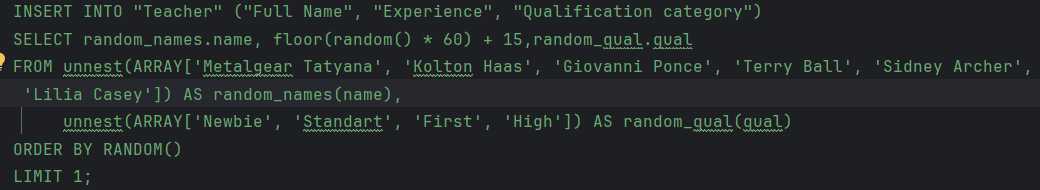


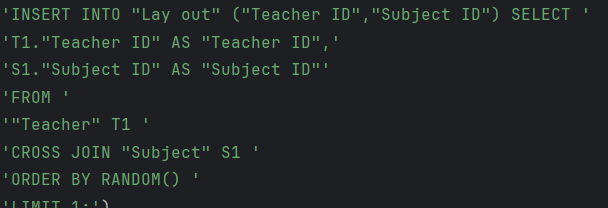
Копії SQL-запитів для генерації записів таблиць:

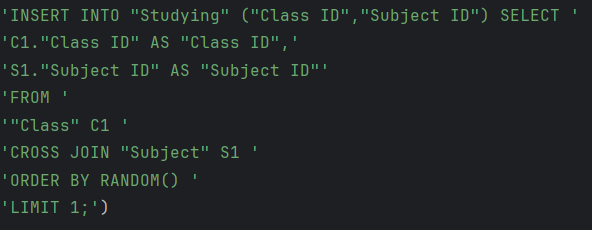




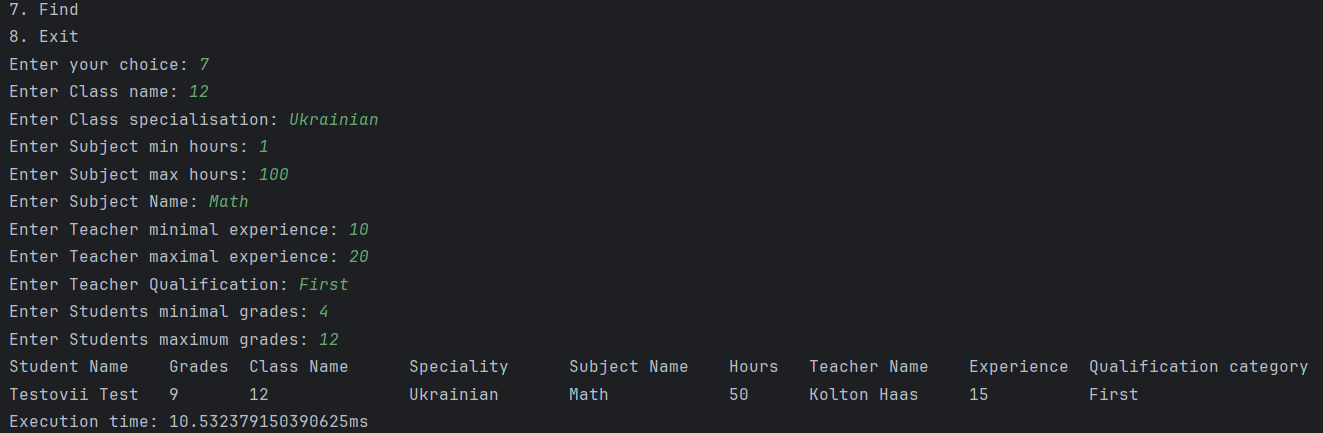




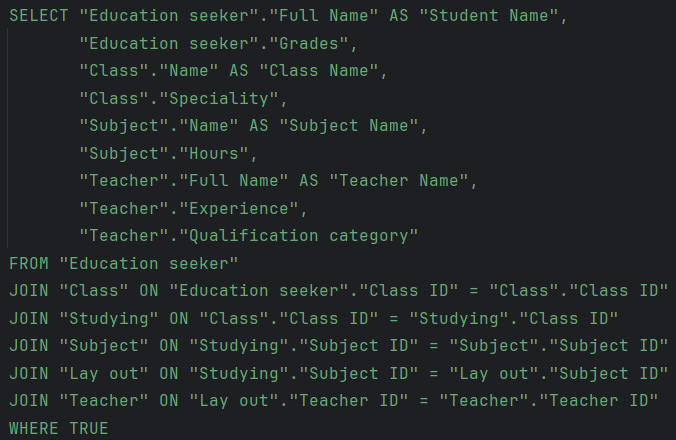


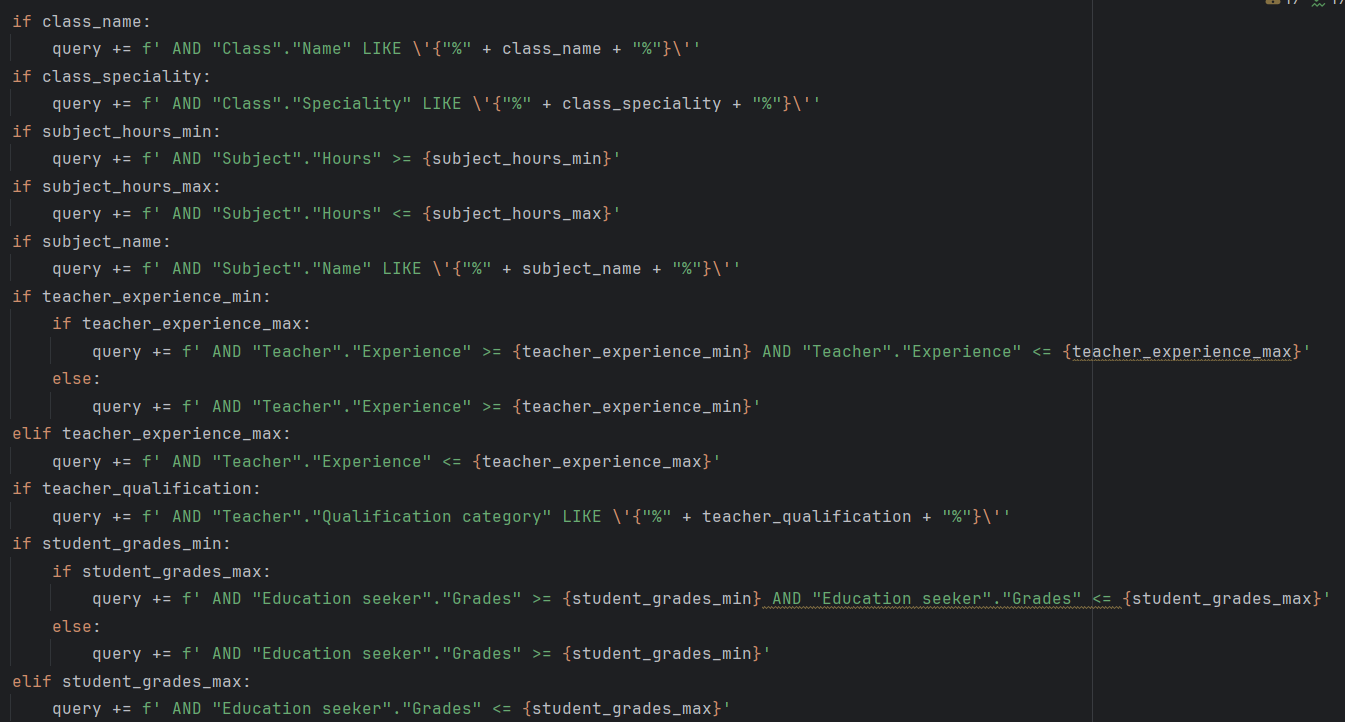


**Пункт №3**



Копія SQL-запиту для пошуку:





**Пункт №4**

Програмний код модуля model:

import psycopg2

import psycopg2.extensions

import **time**

**class** Model:

**def** \_\_init\_\_(self):

self.**conn** = psycopg2.**connect**(

dbname='postgres',

**user**='postgres',

**password**='admin',

**host**='localhost',

port=5434

)

self.create\_table\_class()

self.create\_table\_education\_seeker()

self.create\_table\_subject()

self.create\_table\_teacher()

self.create\_table\_laying()

self.create\_table\_studying()

**def** create\_table\_class(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Class" (

"Class ID" SERIAL PRIMARY KEY,

"Name" CHARACTER VARYING NOT NULL,

"Speciality" CHARACTER VARYING NOT NULL

)

''')

# **Check** **if** **the** **table** **exists**

**c**.**execute**("SELECT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'Class')")

table\_exists = **c**.fetchone()[0]

**if** **not** table\_exists:

# **Table** does **not** exist, so **create** it

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Class" (

"Class ID" SERIAL PRIMARY KEY,

"Name" CHARACTER VARYING NOT NULL,

"Speciality" CHARACTER VARYING NOT NULL

)

''')

self.**conn**.**commit**()

**def** add\_class(self, name, spec):

**c** = self.**conn**.**cursor**()

**c**.**execute**('INSERT INTO "Class" ("Name", "Speciality") VALUES (%s, %s)', (name, spec))

self.**conn**.**commit**()

**def** get\_all\_class(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('SELECT \* FROM "Class"')

**return** **c**.fetchall()

**def** update\_class(self, class\_id, name, spec):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Class" WHERE "Class ID" = {class\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('UPDATE "Class" SET "Name"=%s, "Speciality"=%s WHERE "Class ID"=%s', (name, spec, class\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** delete\_class(self, class\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Education seeker" WHERE "Class ID" = {class\_id})')

student\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Class" WHERE "Class ID" = {class\_id})')

id\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Studying" WHERE "Class ID" = {class\_id})')

studying\_id = **c**.fetchone()[0]

**if** id\_exist:

**if** student\_exist:

**return** "Class is not deleted.Delete all student from it first"

**if** studying\_id:

**return** "Clas is not deleted. Delete it from Studying table first"

**c**.**execute**('DELETE FROM "Class" WHERE "Class ID"=%s', (class\_id,))

self.**conn**.**commit**()

**c**.**execute**('SELECT MAX("Class ID") FROM "Class"')

max\_id = **c**.fetchone()[0]

**if** max\_id **is** **None**:

**c**.**execute**('ALTER SEQUENCE "Class\_Class ID\_seq" RESTART WITH %s', (1,))

self.**conn**.**commit**()

**return** "Class deleted successfuly"

**else**:

**return** "There is no such class ID"

**def** generate\_class(self, **count**):

**c** = self.**conn**.**cursor**()

**for** **i** **in** **range**(**count**):

**c**.**execute**('''

INSERT INTO "Class" ("Name", "Speciality")

SELECT random\_names.name, random\_spec.spec

FROM unnest(ARRAY['11', '12', '9', '10', '8', '7']) AS random\_names(name),

unnest(ARRAY['Math', 'Science', 'History', 'Literature', 'Ukrainian', 'Art']) AS random\_spec(spec)

ORDER BY RANDOM()

LIMIT 1;

''')

self.**conn**.**commit**()

**def** find\_class(self, name, spec):

**c** = self.**conn**.**cursor**()

**c**.**execute**("""SELECT \* FROM "**Class**" WHERE "Name" LIKE %s AND "Speciality" LIKE %s """, (name, spec))

**return** **c**.fetchall()

**def** create\_table\_education\_seeker(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Education seeker" (

"Student ID" SERIAL PRIMARY KEY,

"Full Name" CHARACTER VARYING NOT NULL,

"Grades" INTEGER NOT NULL,

"Class ID" INTEGER NOT NULL

)

''')

**c**.**execute**("SELECT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'Education seeker')")

table\_exists = **c**.fetchone()[0]

**if** **not** table\_exists:

# **Table** does **not** exist, so **create** it

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Education seeker" (

"Student ID" SERIAL PRIMARY KEY,

"Full Name" CHARACTER VARYING NOT NULL,

"Grades" INTEGER NOT NULL,

"Class ID" INTEGER NOT NULL

)

''')

self.**conn**.**commit**()

**def** add\_education\_seeker(self, name, grades, class\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Class" WHERE "Class ID" = {class\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('INSERT INTO "Education seeker" ("Full Name", "Grades", "Class ID") VALUES (%s, %s, %s)', (name, grades, class\_id))

self.**conn**.**commit**()

**return** 1

**return** 0

**def** get\_all\_education\_seekers(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('SELECT \* FROM "Education seeker"')

**return** **c**.fetchall()

**def** update\_education\_seeker(self, student\_id, name, grades, class\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Class" WHERE "Class ID" = {class\_id})')

class\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Student ID" FROM "Education seeker" WHERE "Student ID" = {student\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist **and** class\_exist:

**c**.**execute**('UPDATE "Education seeker" SET "Full Name"=%s, "Grades"=%s, "Class ID"=%s WHERE "Student ID"=%s', (name, grades, class\_id, student\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** delete\_education\_seeker(self, student\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Student ID" FROM "Education seeker" WHERE "Student ID" = {student\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('DELETE FROM "Education seeker" WHERE "Student ID"=%s', (student\_id,))

self.**conn**.**commit**()

**c**.**execute**('SELECT MAX("Student ID") FROM "Education seeker"')

max\_id = **c**.fetchone()[0]

**if** max\_id **is** **None**:

**c**.**execute**('ALTER SEQUENCE "Student\_Education seeker ID\_seq" RESTART WITH %s', (1,))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** generate\_education\_seeker(self, **count**):

**c** = self.**conn**.**cursor**()

**for** **i** **in** **range**(**count**):

**c**.**execute**(

"""INSERT INTO "Education seeker" ("**Full** Name", "Grades", "**Class** **ID**")

**SELECT** random\_names.name, grades.grade, C1."Class ID" **AS** "Class ID"

**FROM** unnest(**ARRAY**['Anabelle Solis', 'Anne Mcdaniel', 'Elijah Baird', 'Francis Cannon', 'Cedric Lambert', 'Damien Jenkins']) **AS** random\_names(name),

(**SELECT** grade **FROM** generate\_series(1, 12) grade **ORDER** **BY** **random**()) **as** grades

**JOIN** "Class" C1 **ON** **true**

**ORDER** **BY** **random**()

**LIMIT** 1;"""

)

self.**conn**.**commit**()

**def** find\_education\_seeker(self, grades\_min, grades\_max, class\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**("""SELECT \* FROM "Education seeker" WHERE "Grades" BETWEEN %s AND %s AND "**Class** **ID**" = %s""", (grades\_min, grades\_max, class\_id))

**return** **c**.fetchall()

**def** create\_table\_subject(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Subject" (

"Subject ID" SERIAL PRIMARY KEY,

"Hours" INTEGER NOT NULL,

"Name" CHARACTER VARYING NOT NULL

)

''')

**c**.**execute**("SELECT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'Subject')")

table\_exists = **c**.fetchone()[0]

**if** **not** table\_exists:

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Subject" (

"Subject ID" SERIAL PRIMARY KEY,

"Hours" INTEGER NOT NULL,

"Name" CHARACTER VARYING NOT NULL

)

''')

self.**conn**.**commit**()

**def** add\_subject(self, hours, name):

**c** = self.**conn**.**cursor**()

**c**.**execute**('INSERT INTO "Subject" ("Hours", "Name") VALUES (%s, %s)', (hours, name))

self.**conn**.**commit**()

**def** get\_all\_subject(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('SELECT \* FROM "Subject"')

**return** **c**.fetchall()

**def** update\_subject(self, subject\_id, hours, name):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Subject" WHERE "Subject ID" = {subject\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('UPDATE "Subject" SET "Hours"=%s, "Name"=%s WHERE "Subject ID"=%s', (hours, name, subject\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** delete\_subject(self, subject\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Subject" WHERE "Subject ID" = {subject\_id})')

id\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Lay out" WHERE "Subject ID" = {subject\_id})')

layout\_id = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Studying" WHERE "Subject ID" = {subject\_id})')

studying\_id = **c**.fetchone()[0]

**if** id\_exist:

**if** layout\_id:

**return** "Subject is not deleted. Delete this subject from Lay out table first"

**if** studying\_id:

**return** "Subject is not deleted. Delete this subject from Studying table first"

**c**.**execute**('DELETE FROM "Subject" WHERE "Subject ID"=%s', (subject\_id,))

self.**conn**.**commit**()

**c**.**execute**('SELECT MAX("Subject ID") FROM "Subject"')

max\_id = **c**.fetchone()[0]

**if** max\_id **is** **None**:

**c**.**execute**('ALTER SEQUENCE "Subject\_Subject ID\_seq" RESTART WITH %s', (1,))

self.**conn**.**commit**()

**return** "Subject deleted"

**else**:

**return** "Subject is not deleted there is no such ID"

**def** find\_subject(self, hours\_min, hours\_max, name):

**c** = self.**conn**.**cursor**()

**c**.**execute**("""SELECT \* FROM "Subject" WHERE "Hours" BETWEEN %s AND %s AND "Name" LIKE %s""", (hours\_min, hours\_max, name))

**return** **c**.fetchall()

**def** generate\_subject(self, **count**):

**c** = self.**conn**.**cursor**()

**for** **i** **in** **range**(**count**):

**c**.**execute**('INSERT INTO "Subject" ("Hours", "Name") '

'SELECT floor(random() \* 60) + 32, random\_names.name '

'FROM unnest(ARRAY[\'Math\', \'Art\', \'Science\','

' \'Ukrainian\', \'History\', \'Literature\']) AS random\_names(name) '

'ORDER BY random()'

'LIMIT 1;')

self.**conn**.**commit**()

**def** create\_table\_teacher(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Teacher" (

"Teacher ID" SERIAL PRIMARY KEY,

"Full Name" CHARACTER VARYING NOT NULL,

"Experience" INTEGER NOT NULL,

"Qualification category" CHARACTER VARYING NOT NULL

)

''')

**c**.**execute**("SELECT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'Teacher')")

table\_exists = **c**.fetchone()[0]

**if** **not** table\_exists:

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Teacher" (

"Teacher ID" SERIAL PRIMARY KEY,

"Full Name" CHARACTER VARYING NOT NULL,

"Experience" INTEGER NOT NULL,

"Qualification category" CHARACTER VARYING NOT NULL

)

''')

self.**conn**.**commit**()

**def** add\_teacher(self, name, **exp**, qual):

**c** = self.**conn**.**cursor**()

**c**.**execute**('INSERT INTO "Teacher" ("Full Name", "Experience", "Qualification category") VALUES (%s, %s, %s)', (name, **exp**, qual))

self.**conn**.**commit**()

**def** get\_all\_teacher(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('SELECT \* FROM "Teacher"')

**return** **c**.fetchall()

**def** update\_teacher(self, teacher\_id, name, **exp**, qual):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Teacher ID" FROM "Teacher" WHERE "Teacher ID" = {teacher\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('UPDATE "Teacher" SET "Full Name"=%s, "Experience"=%s, "Qualification category"=%s WHERE "Teacher ID"=%s', (name, **exp**, qual, teacher\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** delete\_teacher(self, teacher\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Teacher ID" FROM "Teacher" WHERE "Teacher ID" = {teacher\_id})')

id\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Teacher ID" FROM "Lay out" WHERE "Teacher ID" = {teacher\_id})')

layout\_id = **c**.fetchone()[0]

**if** id\_exist:

**if** layout\_id:

**return** "Teacher is not deleted. delete it from Lay out table first"

**c**.**execute**('DELETE FROM "Teacher" WHERE "Teacher ID"=%s', (teacher\_id,))

self.**conn**.**commit**()

**c**.**execute**('SELECT MAX("Teacher ID") FROM "Teacher"')

max\_id = **c**.fetchone()[0]

**if** max\_id **is** **None**:

**c**.**execute**('ALTER SEQUENCE "Teacher\_Teacher ID\_seq" RESTART WITH %s', (1,))

self.**conn**.**commit**()

**return** "Teacher deleted"

**else**:

**return** "Teacher is not deleted.There is no such id"

**def** find\_teacher(self, exp\_min, exp\_max, qual, hours\_min, hours\_max, subject\_name):

**c** = self.**conn**.**cursor**()

**c**.**execute**("""SELECT \* FROM "Teacher" WHERE "Experience" BETWEEN %s AND %s AND "Qualification **category**" LIKE %s""", (exp\_min, exp\_max, qual))

**return** **c**.fetchall()

**def** generate\_teacher(self, **count**):

**c** = self.**conn**.**cursor**()

**for** **i** **in** **range**(**count**):

**c**.**execute**('''

INSERT INTO "Teacher" ("Full Name", "Experience", "Qualification category")

SELECT random\_names.name, floor(random() \* 60) + 15,random\_qual.qual

FROM unnest(ARRAY['Metalgear Tatyana', 'Kolton Haas', 'Giovanni Ponce', 'Terry Ball', 'Sidney Archer', 'Lilia Casey']) AS random\_names(name),

unnest(ARRAY['Newbie', 'Standart', '**First**', 'High']) AS random\_qual(qual)

ORDER BY RANDOM()

LIMIT 1;

''')

self.**conn**.**commit**()

**def** create\_table\_laying(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Lay out" (

"Lay out ID" SERIAL PRIMARY KEY,

"Teacher ID" INTEGER NOT NULL,

"Subject ID" INTEGER NOT NULL

)

''')

**c**.**execute**("SELECT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'Lay out')")

table\_exists = **c**.fetchone()[0]

**if** **not** table\_exists:

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Lay out" (

"Lay out ID" SERIAL PRIMARY KEY,

"Teacher ID" INTEGER NOT NULL,

"Subject ID" INTEGER NOT NULL

)

''')

self.**conn**.**commit**()

**def** add\_laying(self, teach\_id, subject\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Teacher ID" FROM "Teacher" WHERE "Teacher ID" = {teach\_id})')

teach\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Subject" WHERE "Subject ID" = {subject\_id})')

subject\_exist = **c**.fetchone()[0]

**if** teach\_exist **and** subject\_exist:

**c**.**execute**('INSERT INTO "Lay out" ("Teacher ID","Subject ID") VALUES (%s,%s)',

(teach\_id, subject\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** get\_all\_laying(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('SELECT \* FROM "Lay out"')

**return** **c**.fetchall()

**def** update\_laying(self, laying\_id, teach\_id, subject\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Teacher ID" FROM "Teacher" WHERE "Teacher ID" = {teach\_id})')

teach\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Subject" WHERE "Subject ID" = {subject\_id})')

subject\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Lay out ID" FROM "Lay out" WHERE "Lay out ID" = {laying\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist **and** teach\_exist **and** subject\_exist:

**c**.**execute**(

'UPDATE "Lay out" SET "Teacher ID"=%s, "Subject ID"=%s WHERE "Lay out ID"=%s',

(teach\_id, subject\_id, laying\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** delete\_laying(self, laying\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Lay out ID" FROM "Lay out" WHERE "Lay out ID" = {laying\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('DELETE FROM "Lay out" WHERE "Lay out ID"=%s', (laying\_id,))

self.**conn**.**commit**()

**c**.**execute**('SELECT MAX("Lay out ID") FROM "Lay out"')

max\_id = **c**.fetchone()[0]

**if** max\_id **is** **None**:

**c**.**execute**('ALTER SEQUENCE "Laying\_Lay out ID\_seq" RESTART WITH %s', (1,))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** generate\_laying(self, **count**):

**c** = self.**conn**.**cursor**()

**for** **i** **in** **range**(**count**):

**c**.**execute**('INSERT INTO "Lay out" ("Teacher ID","Subject ID") SELECT '

'T1."Teacher ID" AS "Teacher ID",'

'S1."Subject ID" AS "Subject ID"'

'FROM '

'"Teacher" T1 '

'CROSS JOIN "Subject" S1 '

'ORDER BY RANDOM() '

'LIMIT 1;')

self.**conn**.**commit**()

**def** create\_table\_studying(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Studying" (

"Studying ID" SERIAL PRIMARY KEY,

"Class ID" INTEGER NOT NULL,

"Subject ID" INTEGER NOT NULL

)

''')

**c**.**execute**("SELECT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'Studying')")

table\_exists = **c**.fetchone()[0]

**if** **not** table\_exists:

**c**.**execute**('''

CREATE TABLE IF NOT EXISTS "Studying" (

"Studying ID" SERIAL PRIMARY KEY,

"Class ID" INTEGER NOT NULL,

"Subject ID" INTEGER NOT NULL

)

''')

self.**conn**.**commit**()

**def** add\_studying(self, class\_id, subject\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Class" WHERE "Class ID" = {class\_id})')

class\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Subject" WHERE "Subject ID" = {subject\_id})')

subject\_exist = **c**.fetchone()[0]

**if** class\_exist **and** subject\_exist:

**c**.**execute**('INSERT INTO "Studying" ("Class ID","Subject ID") VALUES (%s,%s)',

(class\_id, subject\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** get\_all\_studying(self):

**c** = self.**conn**.**cursor**()

**c**.**execute**('SELECT \* FROM "Studying"')

**return** **c**.fetchall()

**def** update\_studying(self, studying\_id, class\_id, subject\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Class ID" FROM "Class" WHERE "Class ID" = {class\_id})')

class\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Subject ID" FROM "Subject" WHERE "Subject ID" = {subject\_id})')

subject\_exist = **c**.fetchone()[0]

**c**.**execute**(f'SELECT EXISTS(SELECT "Studying ID" FROM "Studying" WHERE "Studying ID" = {studying\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist **and** class\_exist **and** subject\_exist:

**c**.**execute**(

'UPDATE "Studying" SET "Class ID"=%s, "Subject ID"=%s WHERE "Studying ID"=%s',

(class\_id, subject\_id, studying\_id))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** delete\_studying(self, studying\_id):

**c** = self.**conn**.**cursor**()

**c**.**execute**(f'SELECT EXISTS(SELECT "Studying ID" FROM "Studying" WHERE "Studying ID" = {studying\_id})')

id\_exist = **c**.fetchone()[0]

**if** id\_exist:

**c**.**execute**('DELETE FROM "Studying" WHERE "Studying ID"=%s', (studying\_id,))

self.**conn**.**commit**()

**c**.**execute**('SELECT MAX("Studying ID") FROM "Studying"')

max\_id = **c**.fetchone()[0]

**if** max\_id **is** **None**:

**c**.**execute**('ALTER SEQUENCE "Studying\_Studying ID\_seq" RESTART WITH %s', (1,))

self.**conn**.**commit**()

**return** 1

**else**:

**return** 0

**def** generate\_studying(self, **count**):

**c** = self.**conn**.**cursor**()

**for** **i** **in** **range**(**count**):

**c**.**execute**('INSERT INTO "Studying" ("Class ID","Subject ID") SELECT '

'C1."Class ID" AS "Class ID",'

'S1."Subject ID" AS "Subject ID"'

'FROM '

'"Class" C1 '

'CROSS JOIN "Subject" S1 '

'ORDER BY RANDOM() '

'LIMIT 1;')

self.**conn**.**commit**()

**def** find(self, class\_name, class\_speciality, subject\_hours\_min, subject\_hours\_max, subject\_name,

teacher\_experience\_min, teacher\_experience\_max, teacher\_qualification, student\_grades\_min,

student\_grades\_max):

**query** = ('''

SELECT "Education seeker"."Full Name" AS "Student Name",

"Education seeker"."Grades",

"Class"."Name" AS "Class Name",

"Class"."Speciality",

"Subject"."Name" AS "Subject Name",

"Subject"."Hours",

"Teacher"."Full Name" AS "Teacher Name",

"Teacher"."Experience",

"Teacher"."Qualification category"

FROM "Education seeker"

JOIN "Class" ON "Education seeker"."Class ID" = "Class"."Class ID"

JOIN "Studying" ON "Class"."Class ID" = "Studying"."Class ID"

JOIN "Subject" ON "Studying"."Subject ID" = "Subject"."Subject ID"

JOIN "Lay out" ON "Studying"."Subject ID" = "Lay out"."Subject ID"

JOIN "Teacher" ON "Lay out"."Teacher ID" = "Teacher"."Teacher ID"

WHERE TRUE

''')

**if** class\_name:

**query** += f' AND "Class"."Name" LIKE \'{"%" + class\_name + "%"}\''

**if** class\_speciality:

**query** += f' AND "Class"."Speciality" LIKE \'{"%" + class\_speciality + "%"}\''

**if** subject\_hours\_min:

**query** += f' AND "Subject"."Hours" >= {subject\_hours\_min}'

**if** subject\_hours\_max:

**query** += f' AND "Subject"."Hours" <= {subject\_hours\_max}'

**if** subject\_name:

**query** += f' AND "Subject"."Name" LIKE \'{"%" + subject\_name + "%"}\''

**if** teacher\_experience\_min:

**if** teacher\_experience\_max:

**query** += f' AND "Teacher"."Experience" >= {teacher\_experience\_min} AND "Teacher"."Experience" <= {teacher\_experience\_max}'

**else**:

**query** += f' AND "Teacher"."Experience" >= {teacher\_experience\_min}'

**elif** teacher\_experience\_max:

**query** += f' AND "Teacher"."Experience" <= {teacher\_experience\_max}'

**if** teacher\_qualification:

**query** += f' AND "Teacher"."Qualification category" LIKE \'{"%" + teacher\_qualification + "%"}\''

**if** student\_grades\_min:

**if** student\_grades\_max:

**query** += f' AND "Education seeker"."Grades" >= {student\_grades\_min} AND "Education seeker"."Grades" <= {student\_grades\_max}'

**else**:

**query** += f' AND "Education seeker"."Grades" >= {student\_grades\_min}'

**elif** student\_grades\_max:

**query** += f' AND "Education seeker"."Grades" <= {student\_grades\_max}'

**c** = self.**conn**.**cursor**()

start\_time = **time**.**time**()

**c**.**execute**(**query**)

end\_time = **time**.**time**()

**return** (**c**.fetchall(), end\_time - start\_time)

**def** find\_teaching\_classes(self, class\_name, subject\_name, teacher\_name):

**c** = self.**conn**.**cursor**()

**c**.**execute**('''

SELECT l."Subject Name", c."Class Name", t."Teacher Name"

FROM "Lay out" l

JOIN "Class" c ON l."Class ID" = c."Class ID"

JOIN "Teacher" t ON l."Teacher ID" = t."Teacher ID"

WHERE l."Subject Name" = %s AND c."Class Name" = %s AND t."Teacher Name" = %s

GROUP BY l."Subject Name", c."Class Name", t."Teacher Name";

''', (subject\_name, class\_name, teacher\_name))

**rows** = **c**.fetchall()

**for** **row** **in** **rows**:

**print**(**row**)

Опис функцій

\_\_init\_\_ – функція ініціалізації об’єкту, в ній відбувається підключення до бази даних та створюються таблиці

create\_tables – функція створення таблиць бази даних, якщо вони не існують

add\_<table\_name> – додати запис до відповідної таблиці

get\_all\_<table\_name> – отримати всі записи з таблиці

update\_<table\_name> – змінити запис таблиці

delete\_<table\_name> – видалити запис з таблиці

generate\_<table\_name> – згенерувати задану кількість записів таблиці

find – пошук записів за параметрами