МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №2

з дисципліни “Бази даних”

тема “**Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL**”

Виконав	Перевірив
студент II курсу	“--” “вересня” 2020р.
групи КП-93	викладач
Варіант 9	Петрашенко Андрій Васильович
Звєрєв Костянтин Васильович	

Київ 2020

Мета роботи

Здобутти вмінь програмування прикладних додатків баз даних PostgreSQL.

Постановка завдання

1. Реалізувати функції 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).

Варіант: База даних Університет.

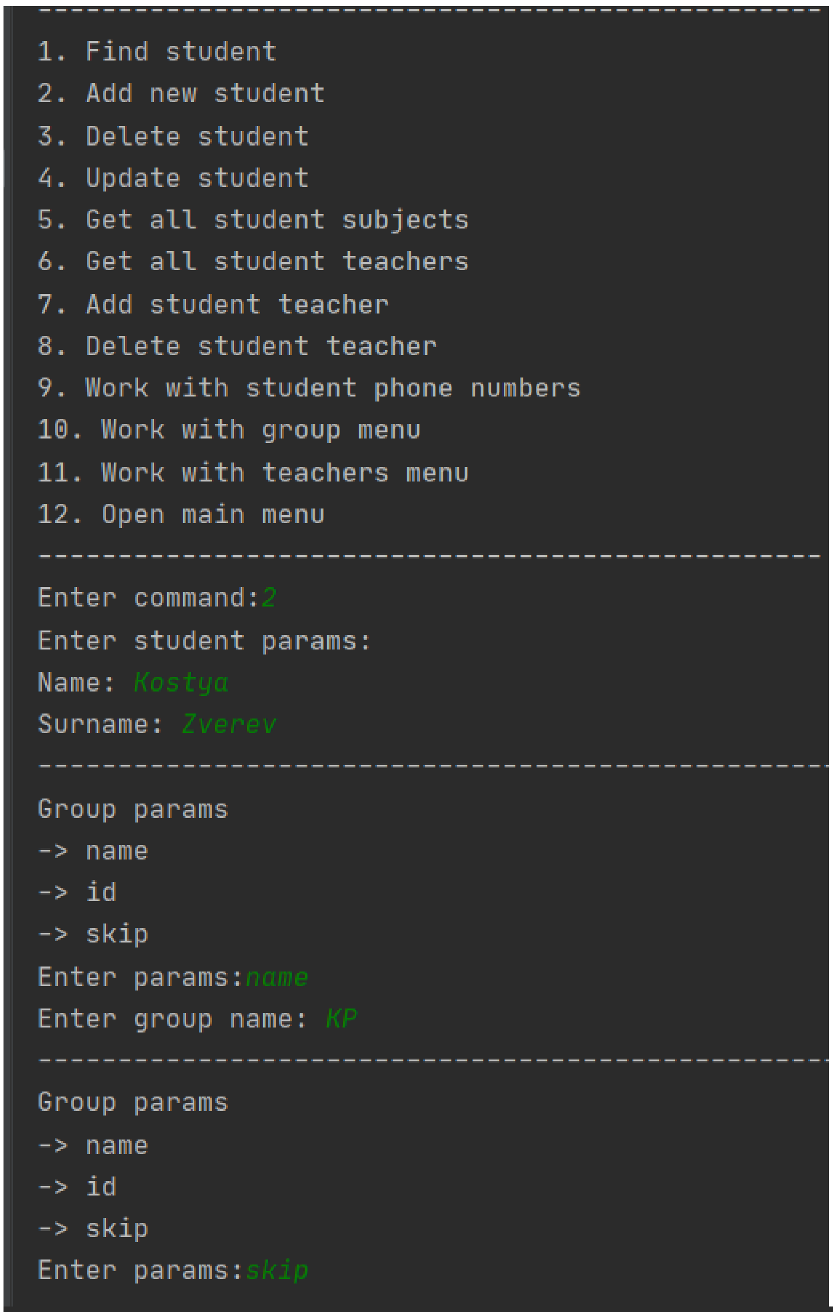
Посилання на Git репозиторій: https://github.com/zver-came/lab2

Посилання на bitbucket репозиторій:

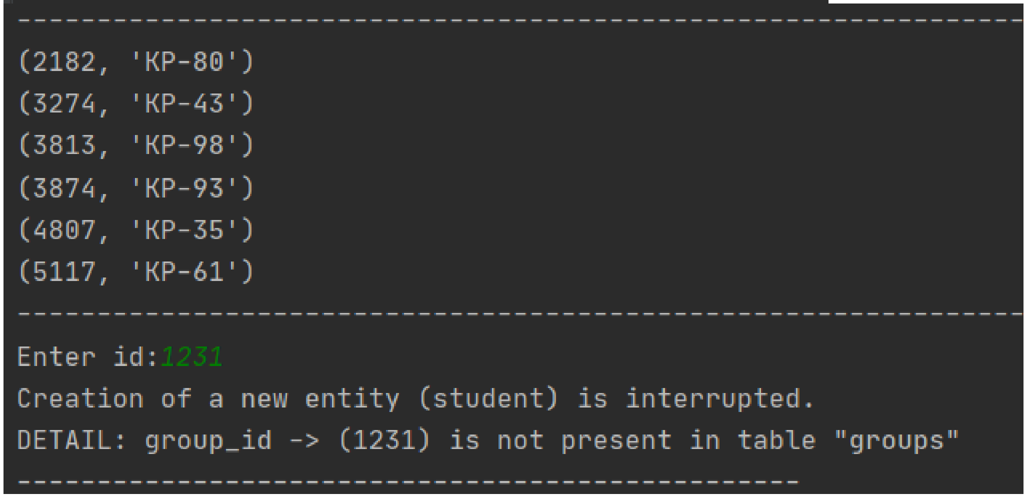
https://bitbucket.org/zverevfpm/databases/src/master/labs/lab2/

Завдання 1

Приклад обробки помилок при уведенні даних:



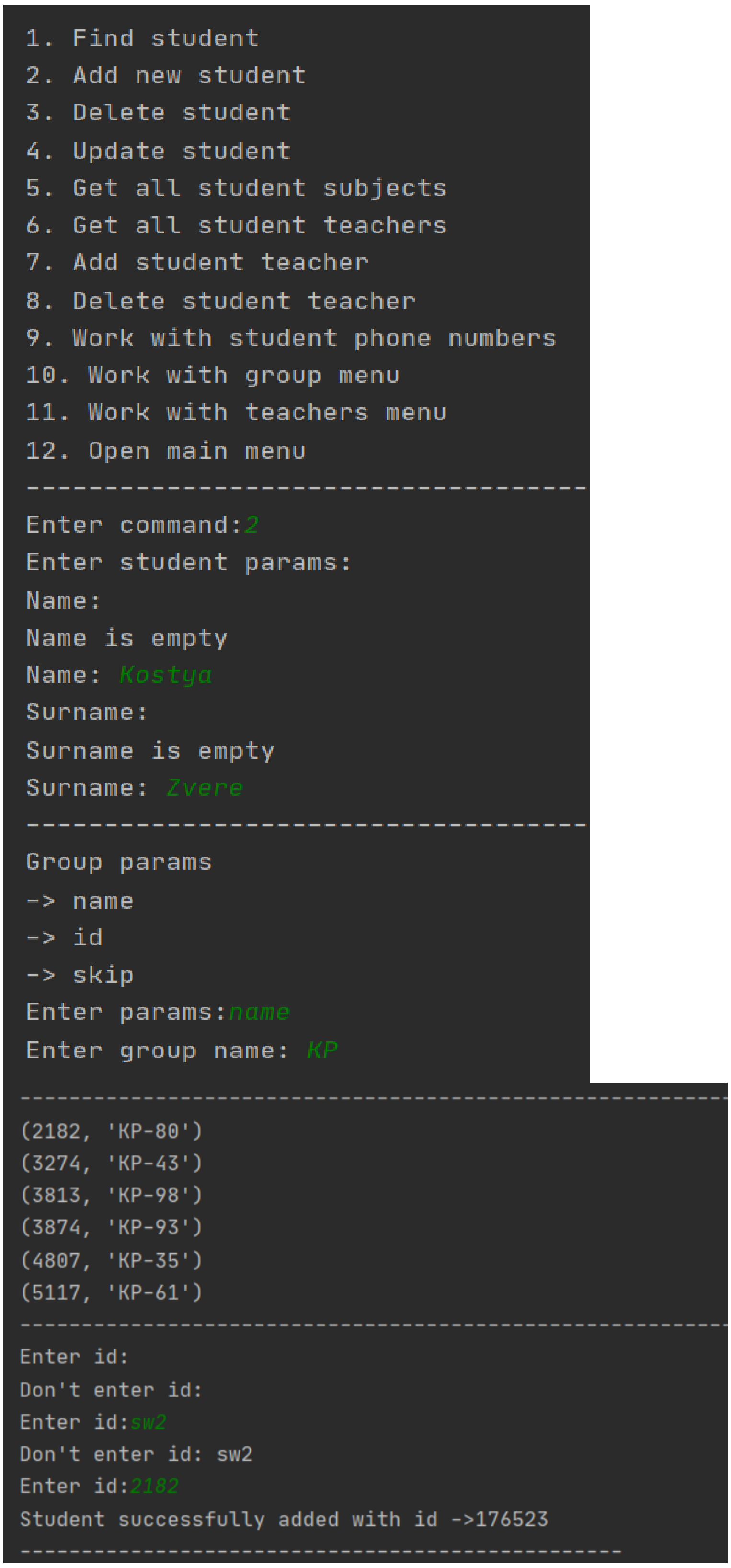
```
-----------------------------------------------
1. Find student
2. Add new student
3. Delete student
4. Update student
5. Get all student subjects
6. Get all student teachers
7. Add student teacher
8. Delete student teacher
9. Work with student phone numbers
10. Work with group menu
11. Work with teachers menu
12. Open main menu
-----------------------------------------------
Enter command:2
Enter student params:
Name: Kostya
Surname: Zverev
-----------------------------------------------
Group params
-> name
-> id
-> skip
Enter params:name
Enter group name: KP
-----------------------------------------------
Group params
-> name
-> id
-> skip
Enter params:skip
```



```
-------------------------------------------------------------
(2182, 'KP-80')
(3274, 'KP-43')
(3813, 'KP-98')
(3874, 'KP-93')
(4807, 'KP-35')
(5117, 'KP-61')
-------------------------------------------------------------
Enter id:1231
Creation of a new entity (student) is interrupted.
DETAIL: group_id -> (1231) is not present in table "groups"
-----------------------------------------------
```

При видаленні даних з бази даних (відповідних таблиць) помилок не було виявлено, оскільки при видаленні сутності за неіснуючим ідентифікатором помилка не виникає. Крім того, програма контролює зовнішні зв’язки (батьківські та підлеглі таблиці) тому при видаленні сутності з батьківської таблиці, видаляються записи і з підлеглих таблиць з відповідними зовнішніми ключами.

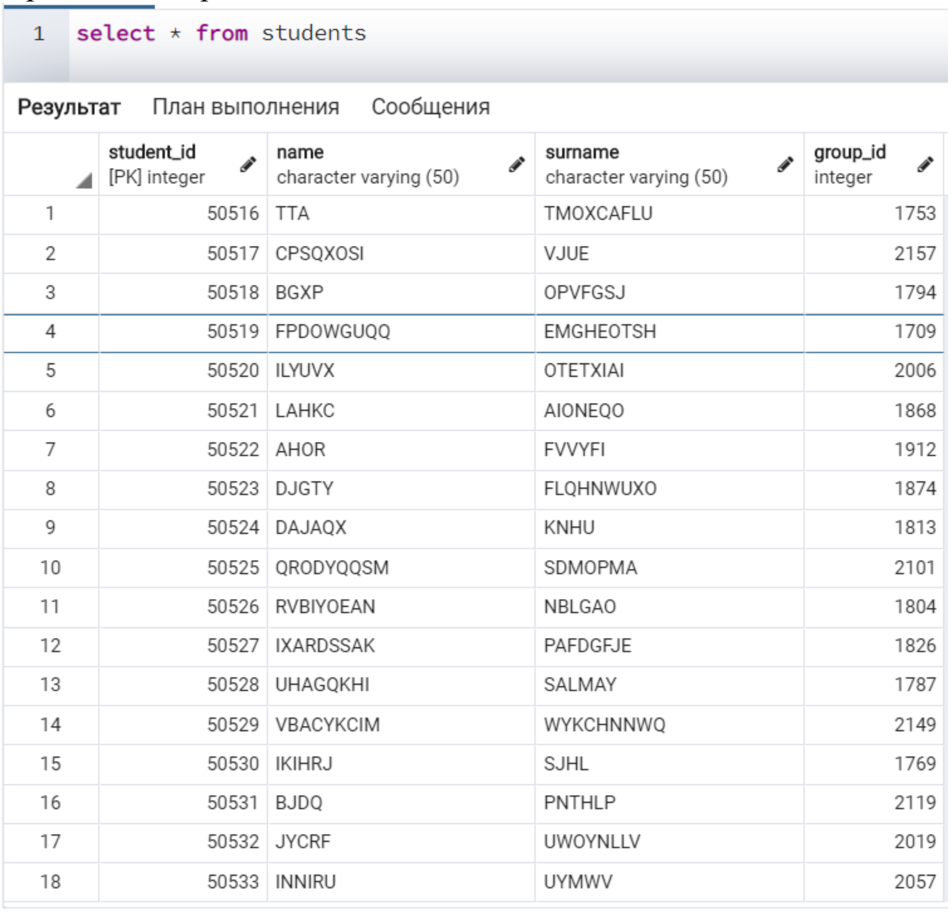
Приклад валідації даних при введенні:



```
1. Find student
2. Add new student
3. Delete student
4. Update student
5. Get all student subjects
6. Get all student teachers
7. Add student teacher
8. Delete student teacher
9. Work with student phone numbers
10. Work with group menu
11. Work with teachers menu
12. Open main menu
------------------------------------
Enter command:2
Enter student params:
Name:
Name is empty
Name: Kostya
Surname:
Surname is empty
Surname: Zvere
------------------------------------
Group params
-> name
-> id
-> skip
Enter params:name
Enter group name: KP
-----------------------------------------------------------
(2182, 'KP-80')
(3274, 'KP-43')
(3813, 'KP-98')
(3874, 'KP-93')
(4807, 'KP-35')
(5117, 'KP-61')
-----------------------------------------------------------
Enter id:
Don't enter id:
Enter id:sw2
Don't enter id: sw2
Enter id:2182
Student successfully added with id ->176523
--------------------------------------------------
```

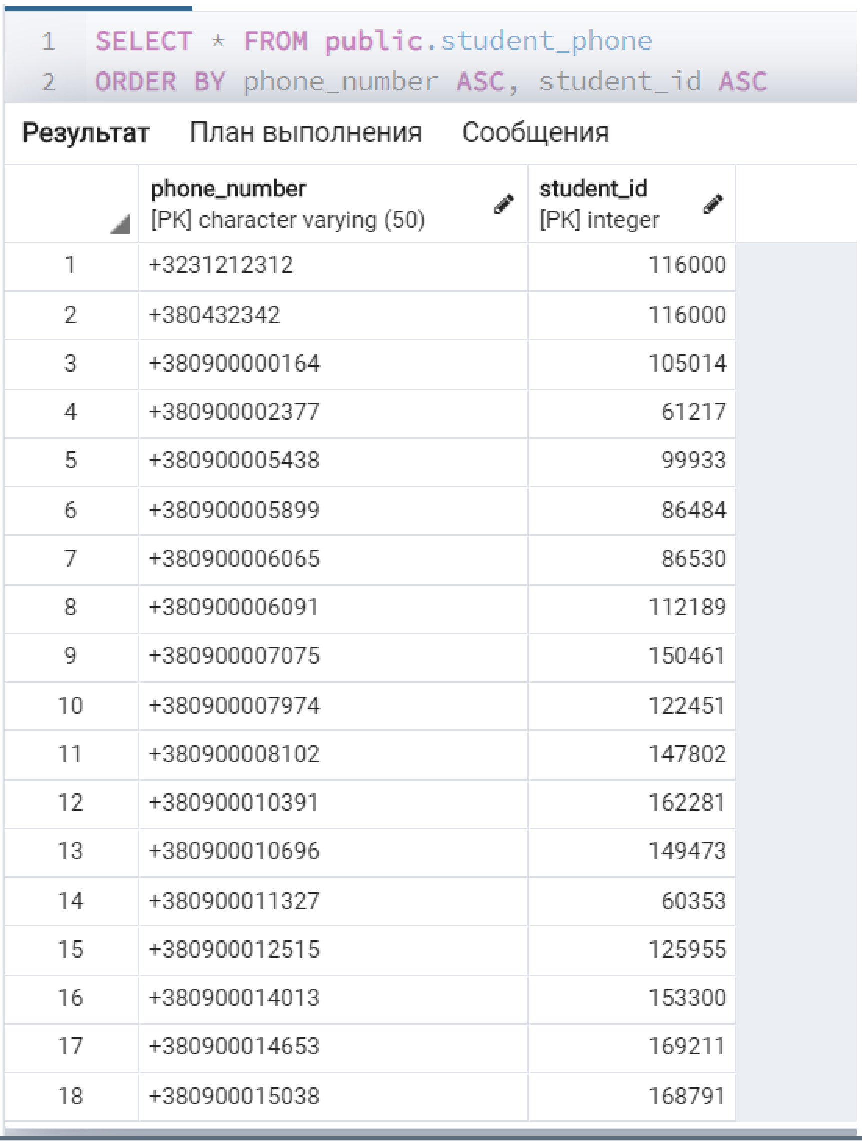
Завдання 2

Ілюстрації зі згенерованими даними таблиць та відповідними SQL запитами:



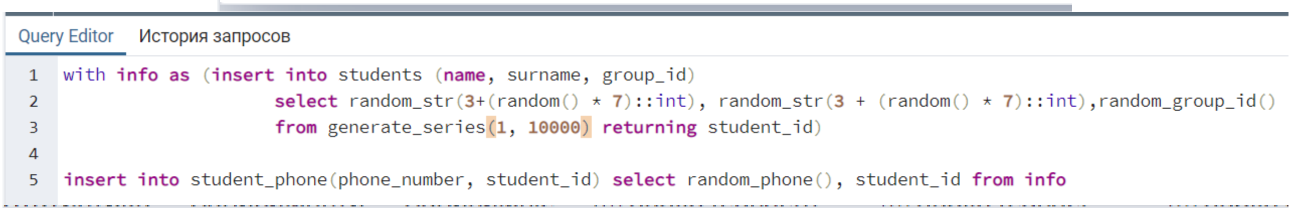
```
select * from students
```

	student_id [PK] integer	name character varying (50)	surname character varying (50)	group_id integer
1	50516	TTA	TMOXCAFLU	1753
2	50517	CPSQXOSI	VJUE	2157
3	50518	BGXP	OPVFGSJ	1794
4	50519	FPDOWGUQQ	EMGHEOTSH	1709
5	50520	ILYUVX	OTETXIAI	2006
6	50521	LAHKC	AIONEQO	1868
7	50522	AHOR	FVVYFI	1912
8	50523	DJGTY	FLQHNWUXO	1874
9	50524	DAJAQX	KNHU	1813
10	50525	QRODYQQSM	SDMOPMA	2101
11	50526	RVBIYOEAN	NBLGAO	1804
12	50527	IXARDSSAK	PAFDGFJE	1826
13	50528	UHAGQKHI	SALMAY	1787
14	50529	VBACYKCIM	WYKCHNNWQ	2149
15	50530	IKIHRJ	SJHL	1769
16	50531	BJDQ	PNTHLP	2119
17	50532	JYCRF	UWOYNLLV	2019
18	50533	INNIRU	UYMWV	2057

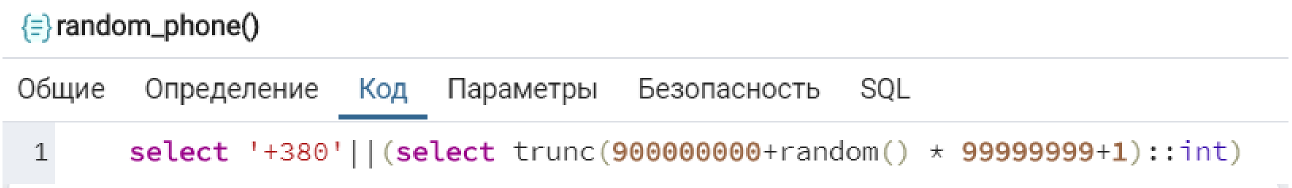


```sql
SELECT * FROM public.student_phone
ORDER BY phone_number ASC, student_id ASC
```

	phone_number [PK] character varying (50)	student_id [PK] integer
1	+3231212312	116000
2	+380432342	116000
3	+380900000164	105014
4	+380900002377	61217
5	+380900005438	99933
6	+380900005899	86484
7	+380900006065	86530
8	+380900006091	112189
9	+380900007075	150461
10	+380900007974	122451
11	+380900008102	147802
12	+380900010391	162281
13	+380900010696	149473
14	+380900011327	60353
15	+380900012515	125955
16	+380900014013	153300
17	+380900014653	169211
18	+380900015038	168791

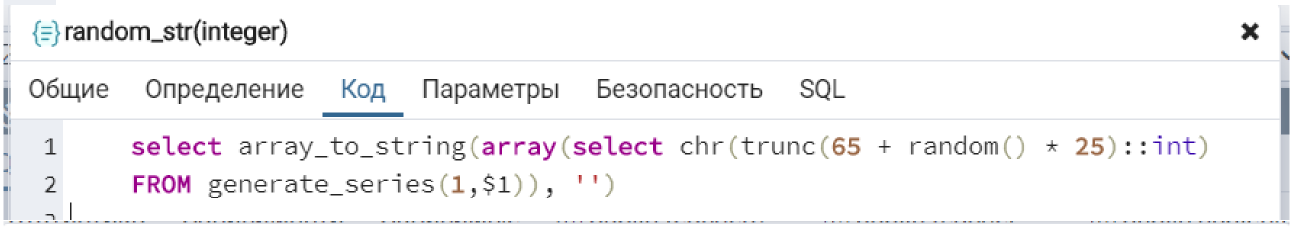


```sql
with info as (insert into students (name, surname, group_id)
              select random_str(3+(random() * 7)::int), random_str(3 + (random() * 7)::int),random_group_id()
              from generate_series(1, 10000) returning student_id)

insert into student_phone(phone_number, student_id) select random_phone(), student_id from info
```



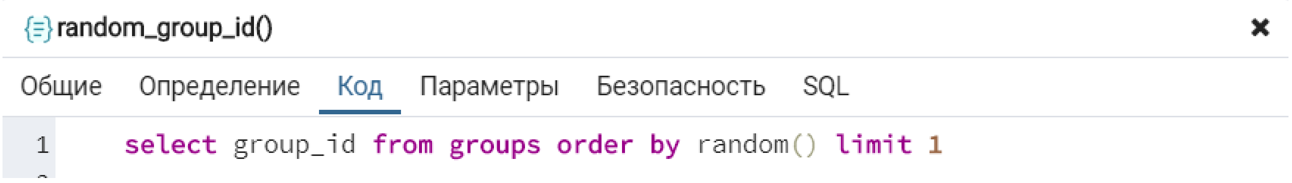
random_phone()

```sql
select '+380'||(select trunc(900000000+random() * 99999999+1)::int)
```



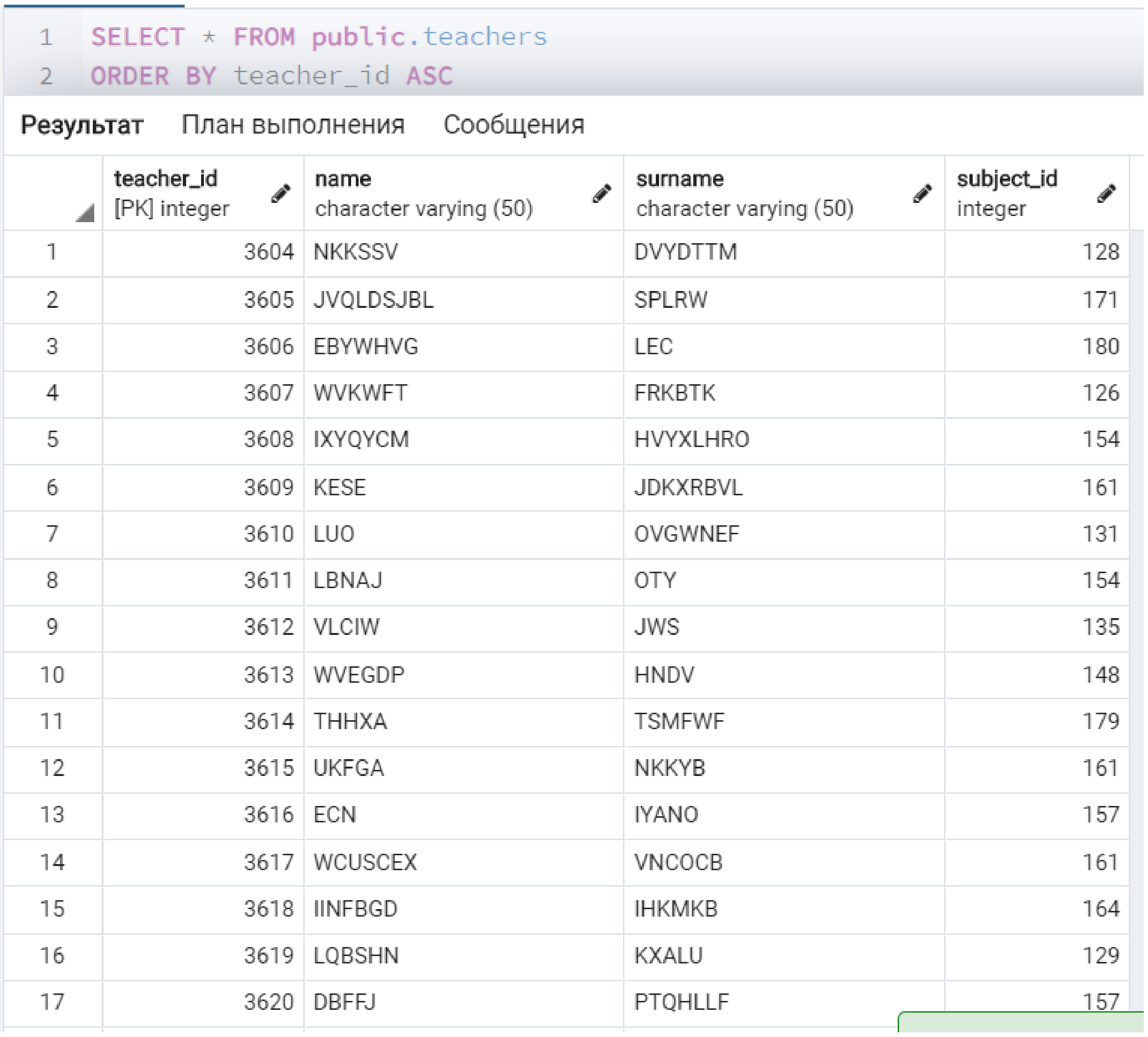
random_str(integer)

```sql
select array_to_string(array(select chr(trunc(65 + random() * 25)::int)
FROM generate_series(1,$1)), '')
```



random_group_id()

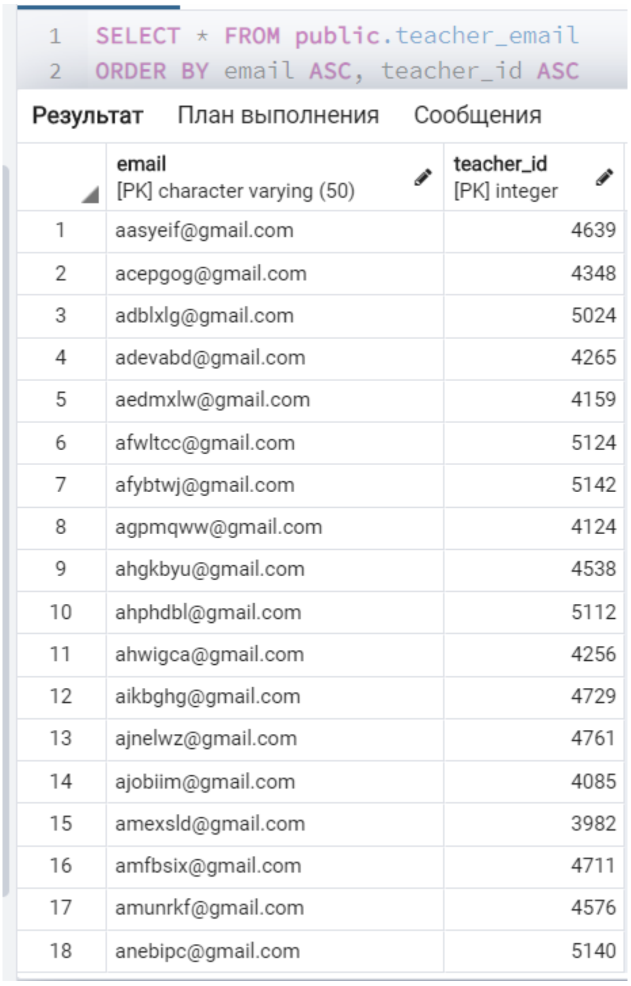
```sql
select group_id from groups order by random() limit 1
```



```
SELECT * FROM public.teachers
ORDER BY teacher_id ASC
```

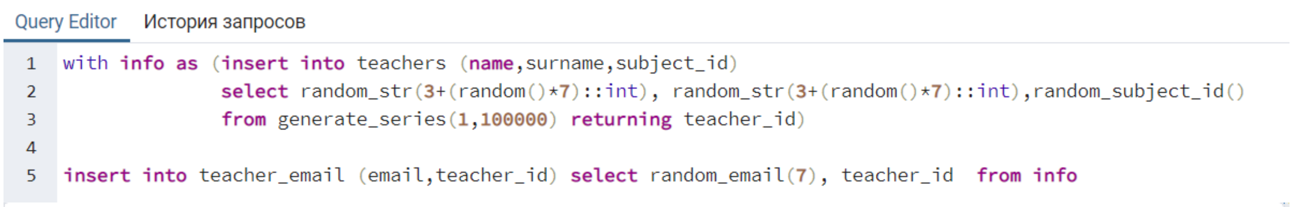
Результат   План выполнения   Сообщения

	teacher_id [PK] integer	name character varying (50)	surname character varying (50)	subject_id integer
1	3604	NKKSSV	DVYDTTM	128
2	3605	JVQLDSJBL	SPLRW	171
3	3606	EBYWHVG	LEC	180
4	3607	WVKWFT	FRKBTK	126
5	3608	IXYQYCM	HVYXLHRO	154
6	3609	KESE	JDKXRBVL	161
7	3610	LUO	OVGWNEF	131
8	3611	LBNAJ	OTY	154
9	3612	VLCIW	JWS	135
10	3613	WVEGDP	HNDV	148
11	3614	THHXA	TSMFWF	179
12	3615	UKFGA	NKKYB	161
13	3616	ECN	IYANO	157
14	3617	WCUSCEX	VNCOCB	161
15	3618	IINFBGD	IHKMKB	164
16	3619	LQBSHN	KXALU	129
17	3620	DBFFJ	PTQHLLF	157

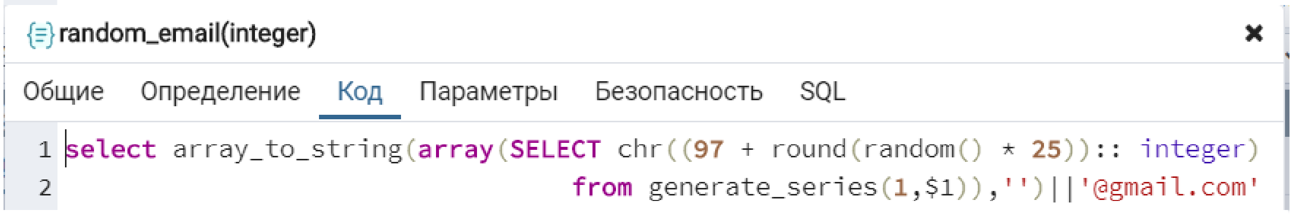


```
SELECT * FROM public.teacher_email
ORDER BY email ASC, teacher_id ASC
```

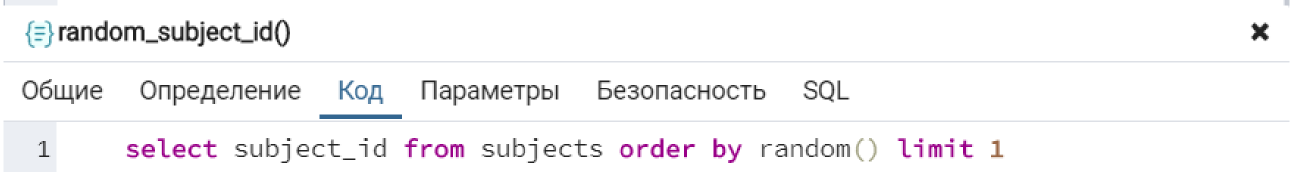
	email [PK] character varying (50)	teacher_id [PK] integer
1	aasyeif@gmail.com	4639
2	acepgog@gmail.com	4348
3	adblxlg@gmail.com	5024
4	adevabd@gmail.com	4265
5	aedmxlw@gmail.com	4159
6	afwltcc@gmail.com	5124
7	afybtwj@gmail.com	5142
8	agpmqww@gmail.com	4124
9	ahgkbyu@gmail.com	4538
10	ahphdbl@gmail.com	5112
11	ahwigca@gmail.com	4256
12	aikbghg@gmail.com	4729
13	ajnelwz@gmail.com	4761
14	ajobiim@gmail.com	4085
15	amexsld@gmail.com	3982
16	amfbsix@gmail.com	4711
17	amunrkf@gmail.com	4576
18	anebipc@gmail.com	5140



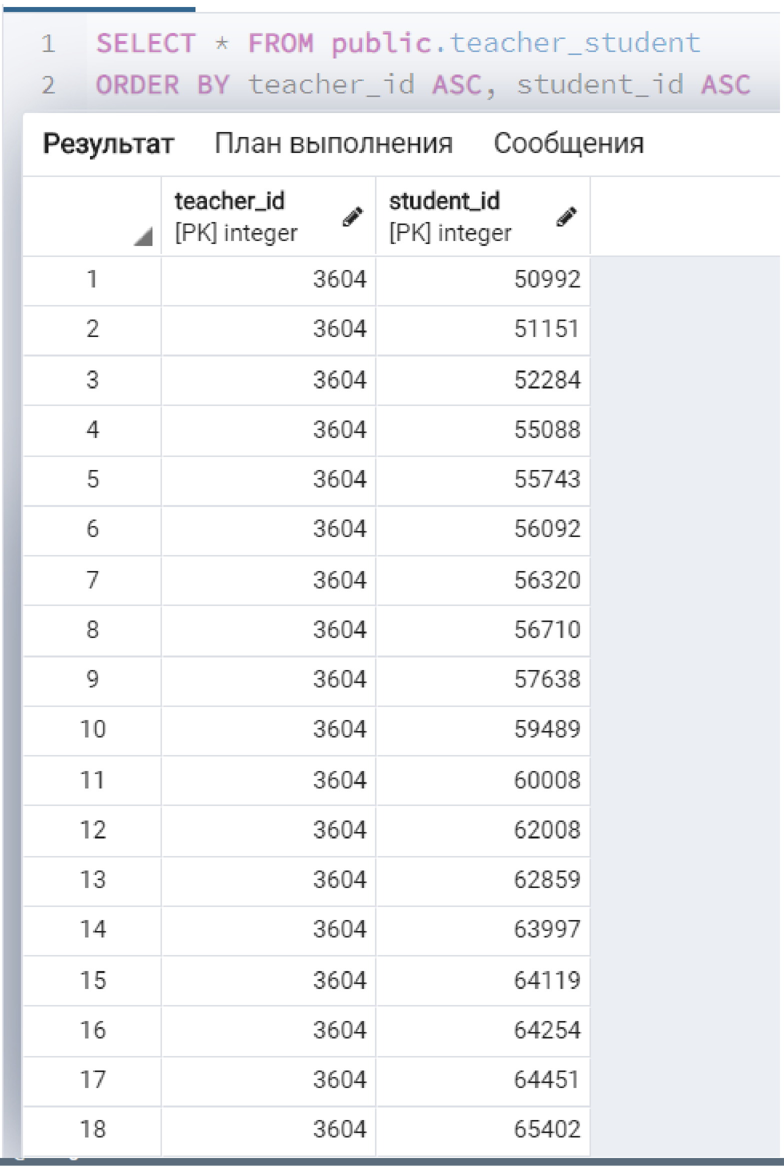
```
with info as (insert into teachers (name,surname,subject_id)
              select random_str(3+(random()*7)::int), random_str(3+(random()*7)::int),random_subject_id()
              from generate_series(1,100000) returning teacher_id)

insert into teacher_email (email,teacher_id) select random_email(7), teacher_id  from info
```



```
select array_to_string(array(SELECT chr((97 + round(random() * 25)):: integer)
                                          from generate_series(1,$1)),'')||'@gmail.com'
```

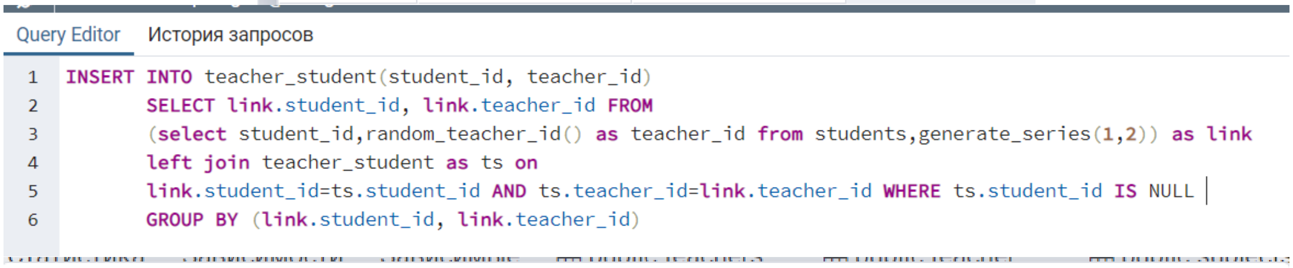


```
select subject_id from subjects order by random() limit 1
```

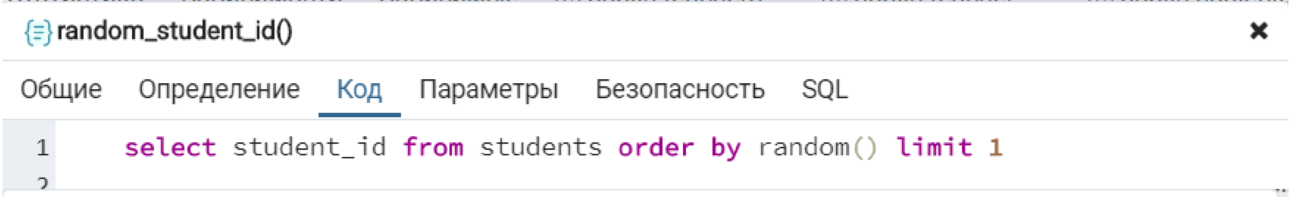


```sql
SELECT * FROM public.teacher_student
ORDER BY teacher_id ASC, student_id ASC
```

	teacher_id [PK] integer	student_id [PK] integer
1	3604	50992
2	3604	51151
3	3604	52284
4	3604	55088
5	3604	55743
6	3604	56092
7	3604	56320
8	3604	56710
9	3604	57638
10	3604	59489
11	3604	60008
12	3604	62008
13	3604	62859
14	3604	63997
15	3604	64119
16	3604	64254
17	3604	64451
18	3604	65402

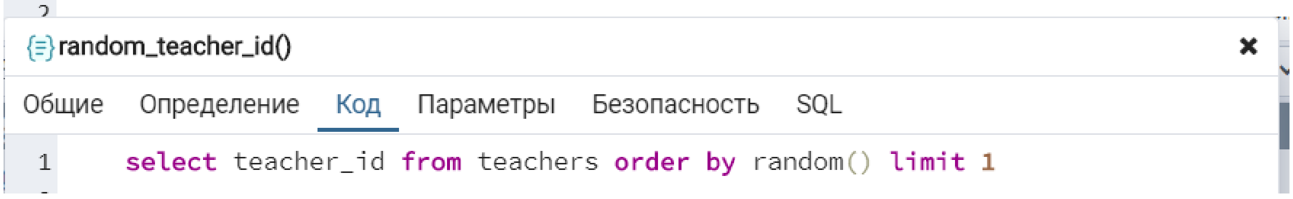


```sql
INSERT INTO teacher_student(student_id, teacher_id)
       SELECT link.student_id, link.teacher_id FROM
       (select student_id,random_teacher_id() as teacher_id from students,generate_series(1,2)) as link
       left join teacher_student as ts on
       link.student_id=ts.student_id AND ts.teacher_id=link.teacher_id WHERE ts.student_id IS NULL
       GROUP BY (link.student_id, link.teacher_id)
```



random_student_id()

```sql
select student_id from students order by random() limit 1
```

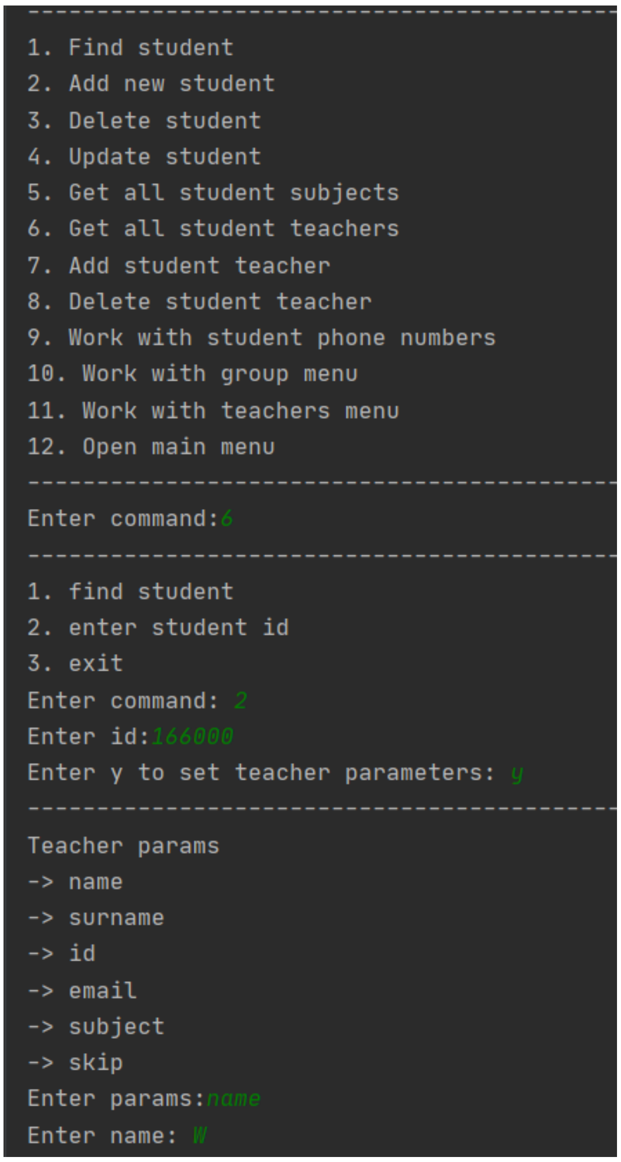


random_teacher_id()

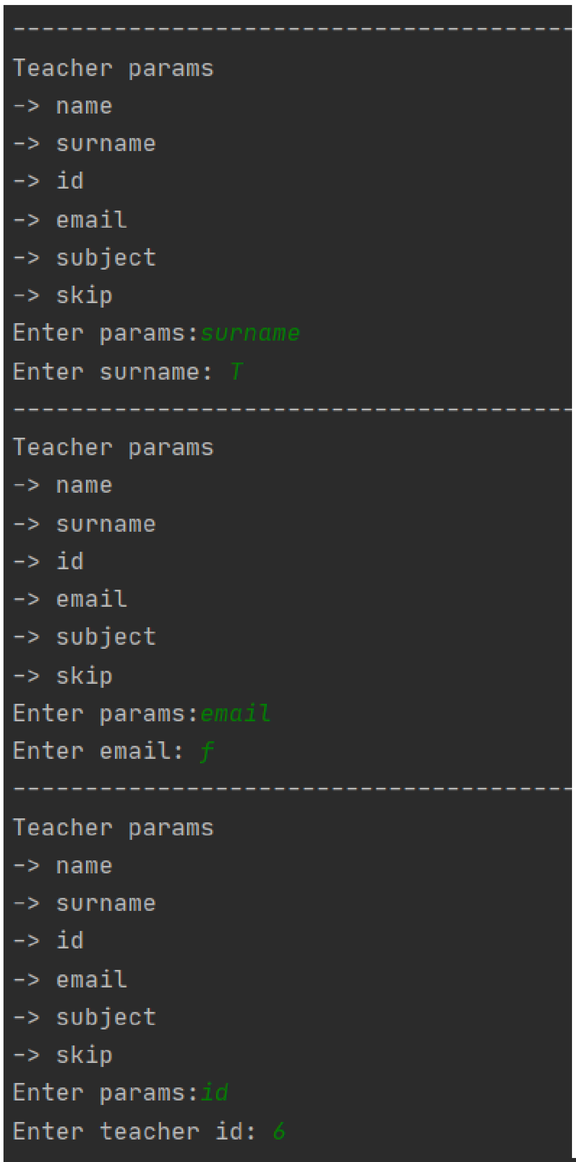
```sql
select teacher_id from teachers order by random() limit 1
```

Завдання 3

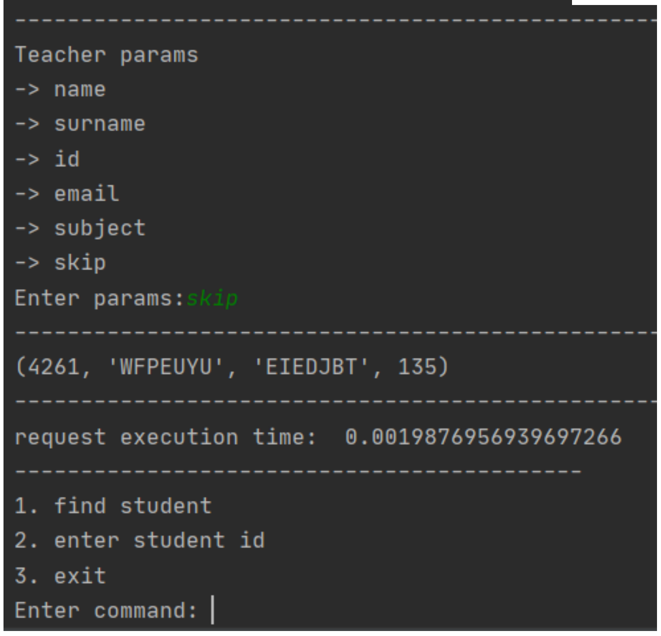
Знайти всіх вчителів студента з певними параметрами:
(name ='%W%', surname='%T%', Email='%f%', id='%6%')



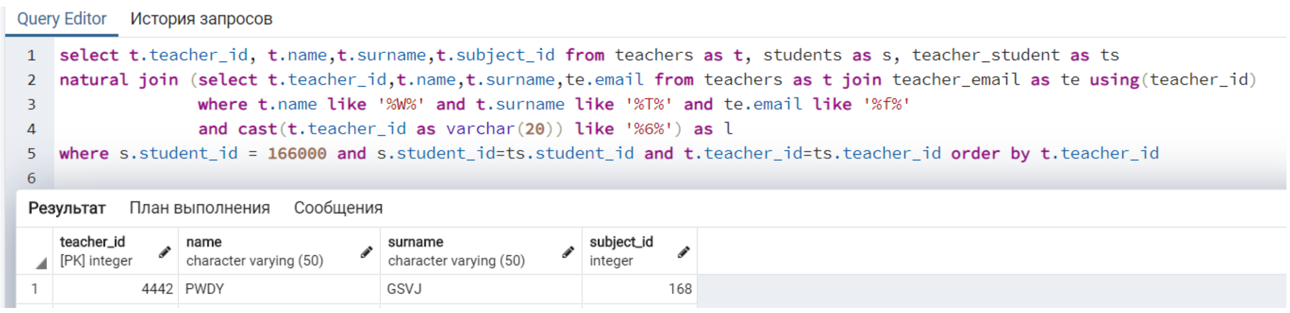
```
------------------------------------------
1. Find student
2. Add new student
3. Delete student
4. Update student
5. Get all student subjects
6. Get all student teachers
7. Add student teacher
8. Delete student teacher
9. Work with student phone numbers
10. Work with group menu
11. Work with teachers menu
12. Open main menu
------------------------------------------
Enter command:6
------------------------------------------
1. find student
2. enter student id
3. exit
Enter command: 2
Enter id:166000
Enter y to set teacher parameters: y
------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:name
Enter name: W
```



```
------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:surname
Enter surname: T
------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:email
Enter email: f
------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:id
Enter teacher id: 6
```



```
------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:skip
------------------------------------------
(4261, 'WFPEUYU', 'EIEDJBT', 135)
------------------------------------------
request execution time:  0.0019876956939697266
------------------------------------------
1. find student
2. enter student id
3. exit
Enter command: |
```



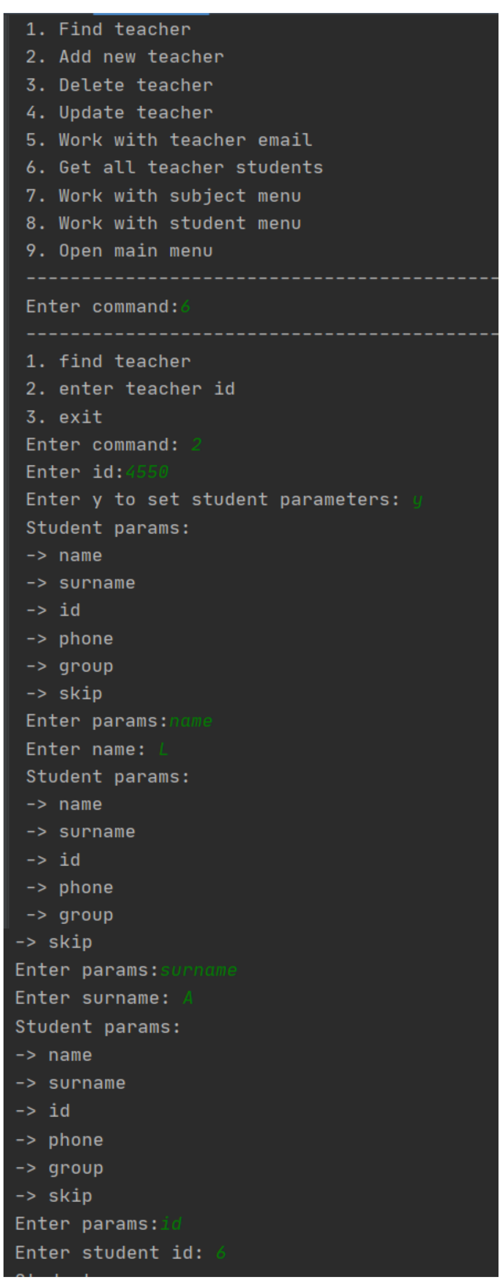
Query Editor История запросов

```sql
select t.teacher_id, t.name,t.surname,t.subject_id from teachers as t, students as s, teacher_student as ts
natural join (select t.teacher_id,t.name,t.surname,te.email from teachers as t join teacher_email as te using(teacher_id)
              where t.name like '%W%' and t.surname like '%T%' and te.email like '%f%'
              and cast(t.teacher_id as varchar(20)) like '%6%') as l
where s.student_id = 166000 and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by t.teacher_id

```

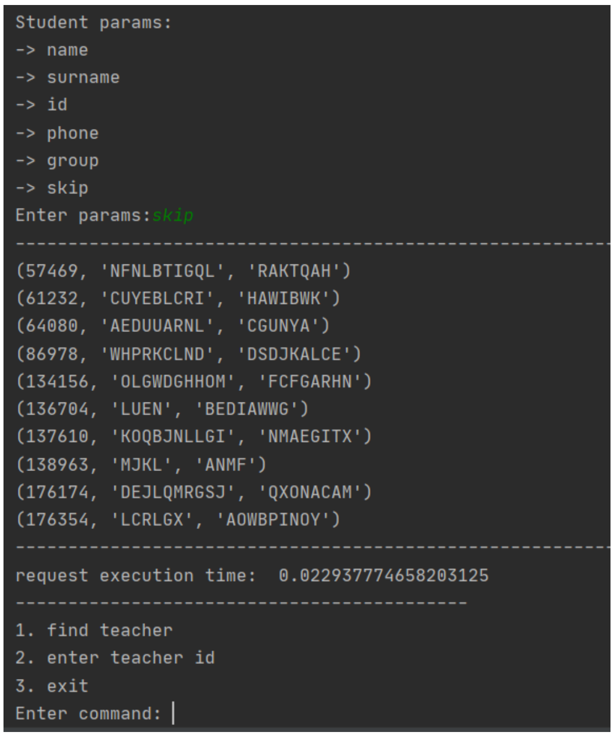
Результат План выполнения Сообщения

	teacher_id [PK] integer	name character varying (50)	surname character varying (50)	subject_id integer
1	4442	PWDY	GSVJ	168

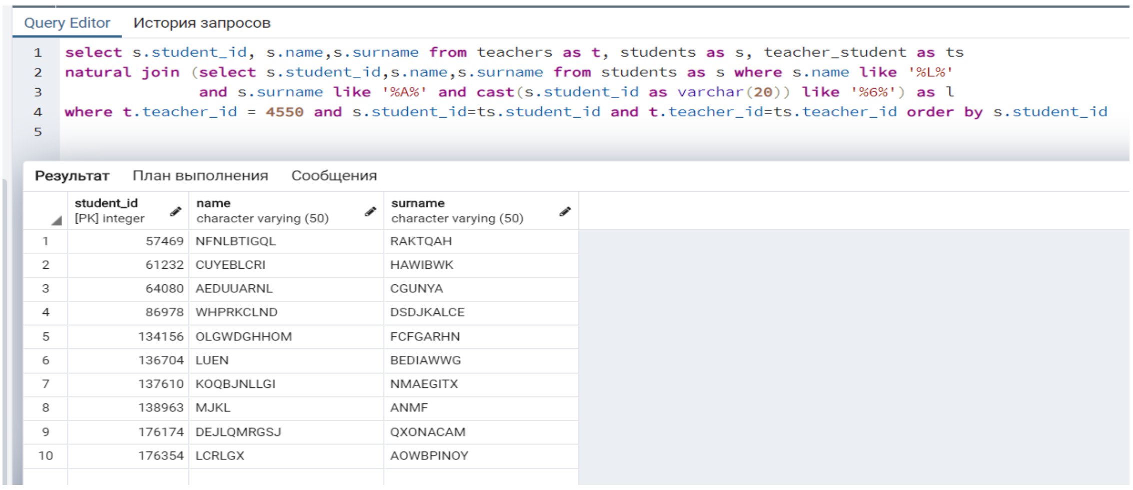
Знайти всіх студентів вчителя з певними параметрами:
(name =’%L%’, surname=’%A%’, id=’%6%’)



```
 1. Find teacher
 2. Add new teacher
 3. Delete teacher
 4. Update teacher
 5. Work with teacher email
 6. Get all teacher students
 7. Work with subject menu
 8. Work with student menu
 9. Open main menu
 -------------------------------------------
 Enter command:6
 -------------------------------------------
 1. find teacher
 2. enter teacher id
 3. exit
 Enter command: 2
 Enter id:4550
 Enter y to set student parameters: y
 Student params:
 -> name
 -> surname
 -> id
 -> phone
 -> group
 -> skip
 Enter params:name
 Enter name: L
 Student params:
 -> name
 -> surname
 -> id
 -> phone
 -> group
-> skip
Enter params:surname
Enter surname: A
Student params:
-> name
-> surname
-> id
-> phone
-> group
-> skip
Enter params:id
Enter student id: 6
```



```
Student params:
-> name
-> surname
-> id
-> phone
-> group
-> skip
Enter params:skip
---------------------------------------------------
(57469, 'NFNLBTIGQL', 'RAKTQAH')
(61232, 'CUYEBLCRI', 'HAWIBWK')
(64080, 'AEDUUARNL', 'CGUNYA')
(86978, 'WHPRKCLND', 'DSDJKALCE')
(134156, 'OLGWDGHHOM', 'FCFGARHN')
(136704, 'LUEN', 'BEDIAWWG')
(137610, 'KOQBJNLLGI', 'NMAEGITX')
(138963, 'MJKL', 'ANMF')
(176174, 'DEJLQMRGSJ', 'QXONACAM')
(176354, 'LCRLGX', 'AOWBPINOY')
---------------------------------------------------
request execution time:  0.022937774658203125
-------------------------------------------
1. find teacher
2. enter teacher id
3. exit
Enter command: 
```

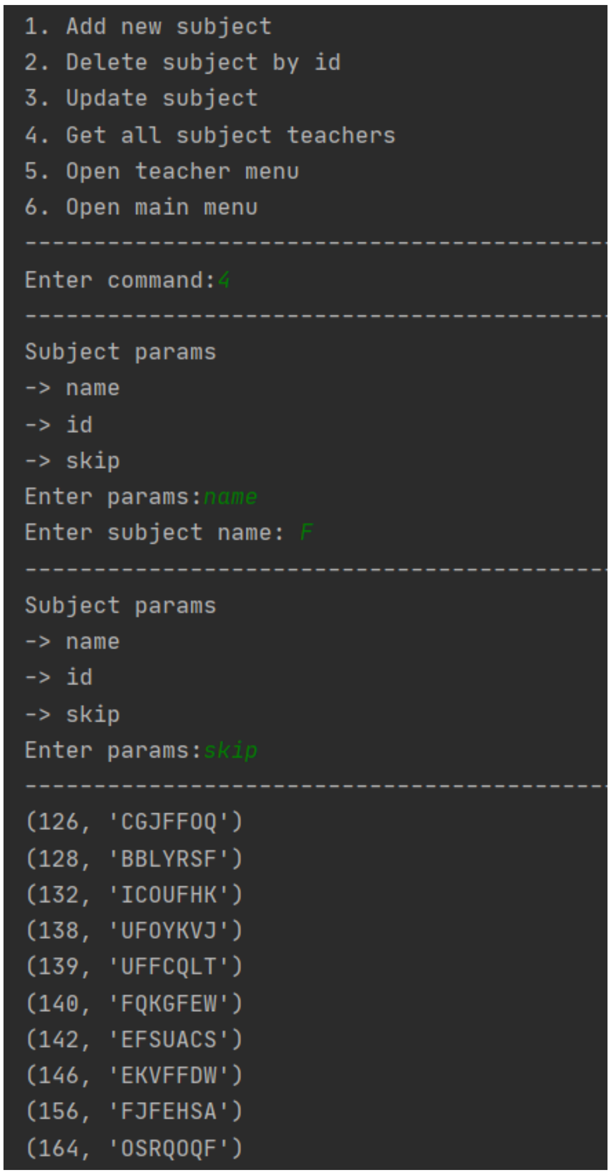

Query Editor История запросов

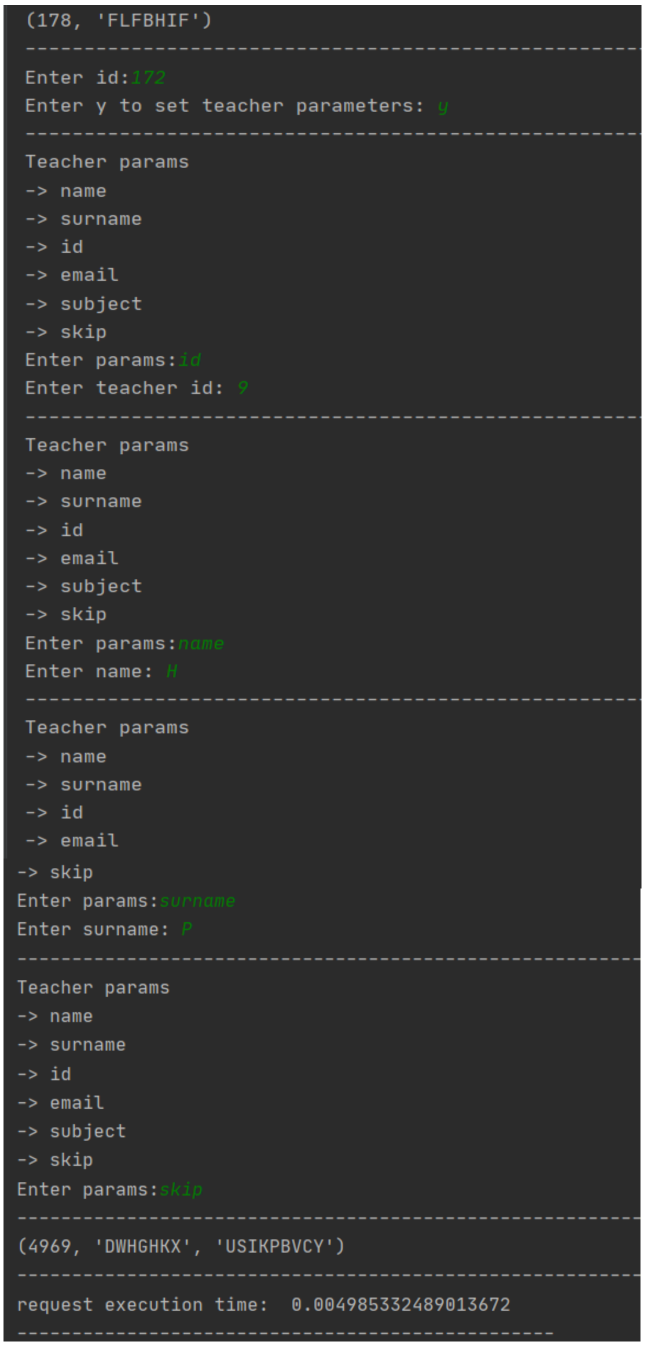
```sql
select s.student_id, s.name,s.surname from teachers as t, students as s, teacher_student as ts
natural join (select s.student_id,s.name,s.surname from students as s where s.name like '%L%'
              and s.surname like '%A%' and cast(s.student_id as varchar(20)) like '%6%') as l
where t.teacher_id = 4550 and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by s.student_id
```

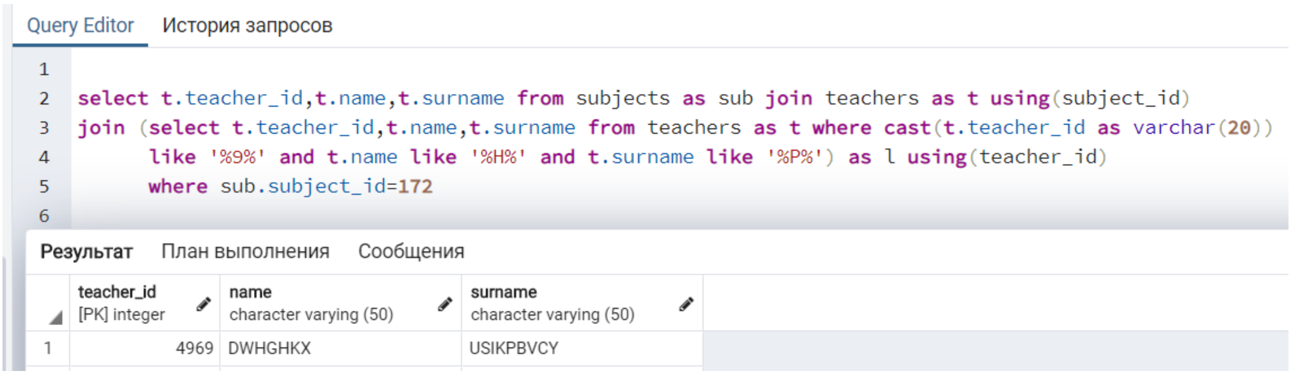
Результат План выполнения Сообщения

	student_id [PK] integer	name character varying (50)	surname character varying (50)
1	57469	NFNLBTIGQL	RAKTQAH
2	61232	CUYEBLCRI	HAWIBWK
3	64080	AEDUUARNL	CGUNYA
4	86978	WHPRKCLND	DSDJKALCE
5	134156	OLGWDGHHOM	FCFGARHN
6	136704	LUEN	BEDIAWWG
7	137610	KOQBJNLLGI	NMAEGITX
8	138963	MJKL	ANMF
9	176174	DEJLQMRGSJ	QXONACAM
10	176354	LCRLGX	AOWBPINOY

Знайти всіх вчителів предмета з певними параметрами:
(name =’%H%’, surname=’%P%’, Email=’%f%’, id=’%9%’)


```
1. Add new subject
2. Delete subject by id
3. Update subject
4. Get all subject teachers
5. Open teacher menu
6. Open main menu
------------------------------------------
Enter command:4
------------------------------------------
Subject params
-> name
-> id
-> skip
Enter params:name
Enter subject name: F
------------------------------------------
Subject params
-> name
-> id
-> skip
Enter params:skip
------------------------------------------
(126, 'CGJFFOQ')
(128, 'BBLYRSF')
(132, 'ICOUFHK')
(138, 'UFOYKVJ')
(139, 'UFFCQLT')
(140, 'FQKGFEW')
(142, 'EFSUACS')
(146, 'EKVFFDW')
(156, 'FJFEHSA')
(164, 'OSRQOQF')
```


```
(178, 'FLFBHIF')
----------------------------------------------------
Enter id:172
Enter y to set teacher parameters: y
----------------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:id
Enter teacher id: 9
----------------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:name
Enter name: H
----------------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> skip
Enter params:surname
Enter surname: P
----------------------------------------------------
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:skip
----------------------------------------------------
(4969, 'DWHGHKX', 'USIKPBVCY')
----------------------------------------------------
request execution time:  0.004985332489013672
----------------------------------------------------
```


Query Editor   История запросов

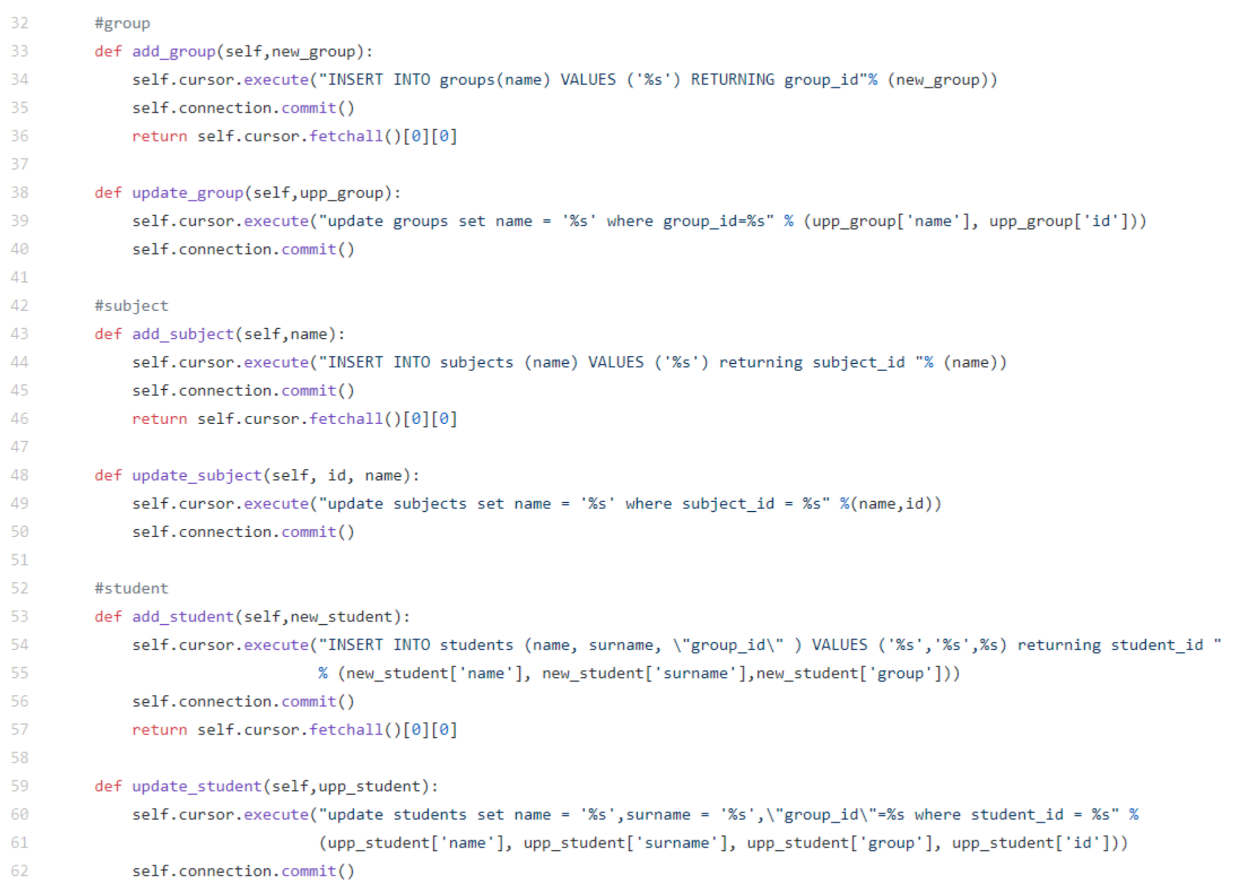
```sql

select t.teacher_id,t.name,t.surname from subjects as sub join teachers as t using(subject_id)
join (select t.teacher_id,t.name,t.surname from teachers as t where cast(t.teacher_id as varchar(20))
      like '%9%' and t.name like '%H%' and t.surname like '%P%') as l using(teacher_id)
      where sub.subject_id=172

```

Результат   План выполнения   Сообщения

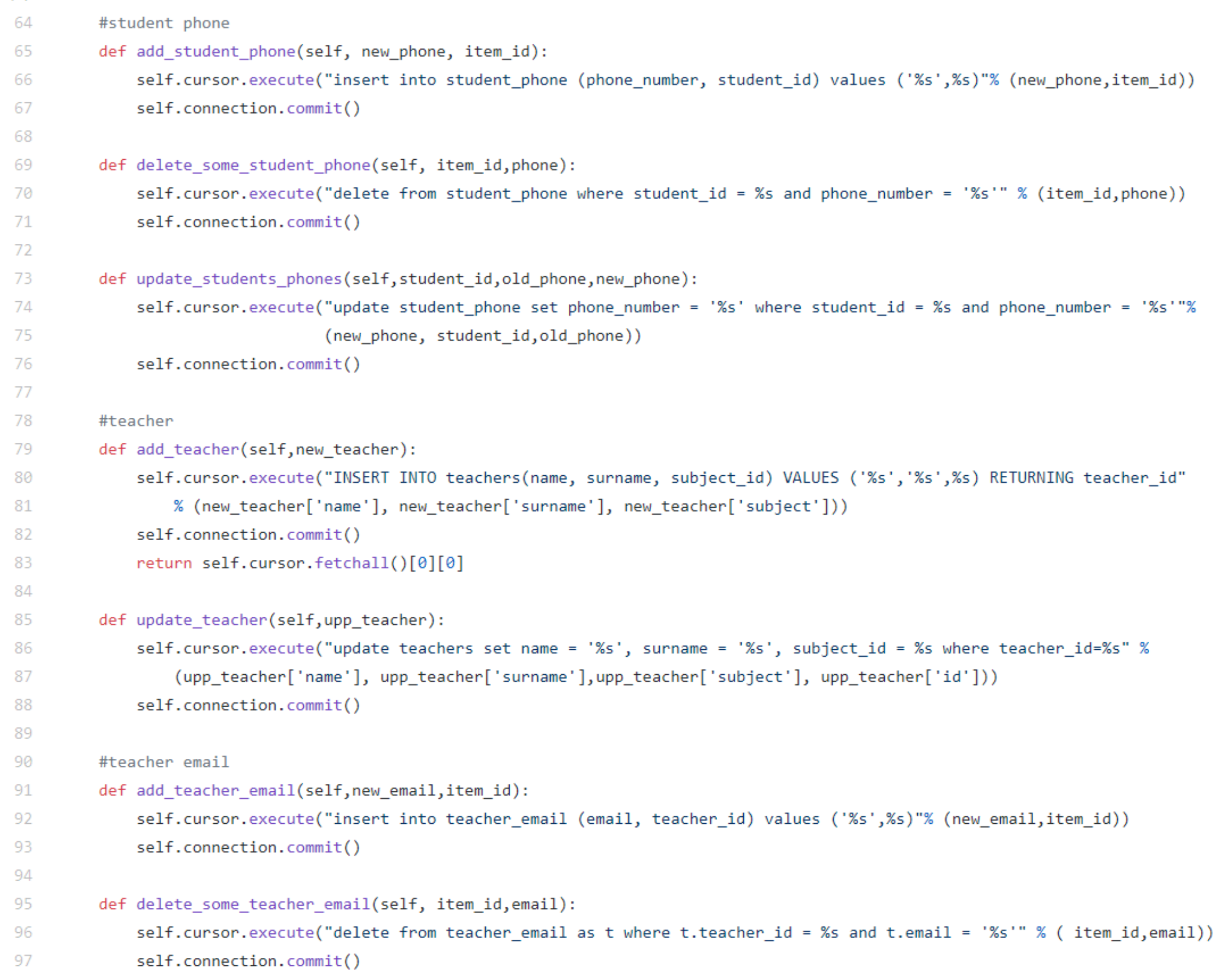
	teacher_id [PK] integer	name character varying (50)	surname character varying (50)
1	4969	DWHGHKX	USIKPBVCY

Завдання 4
Model:

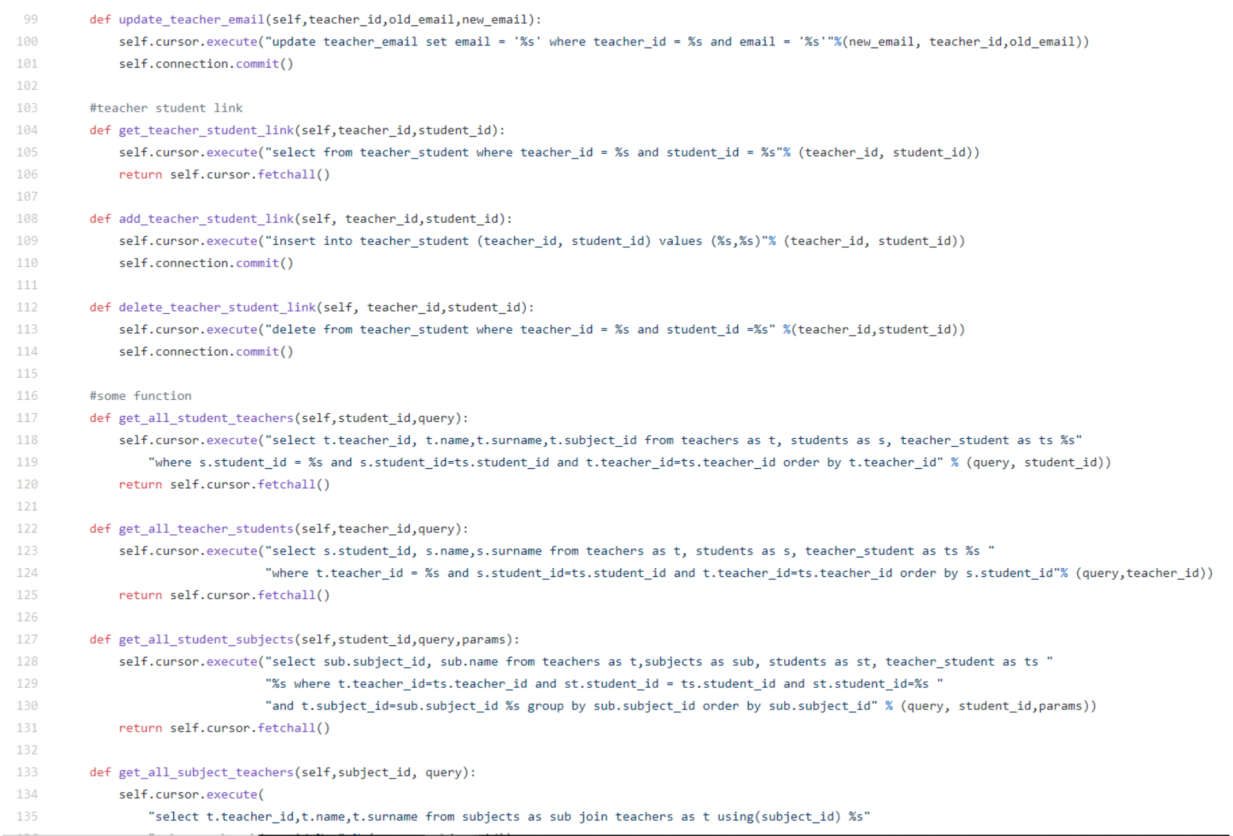
```python
import psycopg2

class Model:
    def __init__(self):
        self.cursor=None
        self.connection=None
        try:
            self.connection=psycopg2.connect(user="postgres",password="postgres",host="127.0.0.1",port="5432",database="database_lab1")
            self.cursor=self.connection.cursor()
        except(Exception, psycopg2.Error) as error:print("Error connection with PostgreSQL",error)

    def __del__(self):
        if self.connection:
            self.cursor.close()
            self.connection.close()
            print("Connection closed")

    def delete_item(self, item_id, table, item_point):
        self.cursor.execute("DELETE FROM %s WHERE %s= %s" % (table, item_point, item_id))
        self.connection.commit()

    def select_items(self, table):
        self.cursor.execute("select * FROM %s" % (table))
        self.connection.commit()
        return self.cursor.fetchall()

    def select_item_by_id(self, item_id, table, item_point):
        self.cursor.execute("select * FROM %s WHERE %s= %s" % (table, item_point, item_id))
        self.connection.commit()
        return self.cursor.fetchall()
```



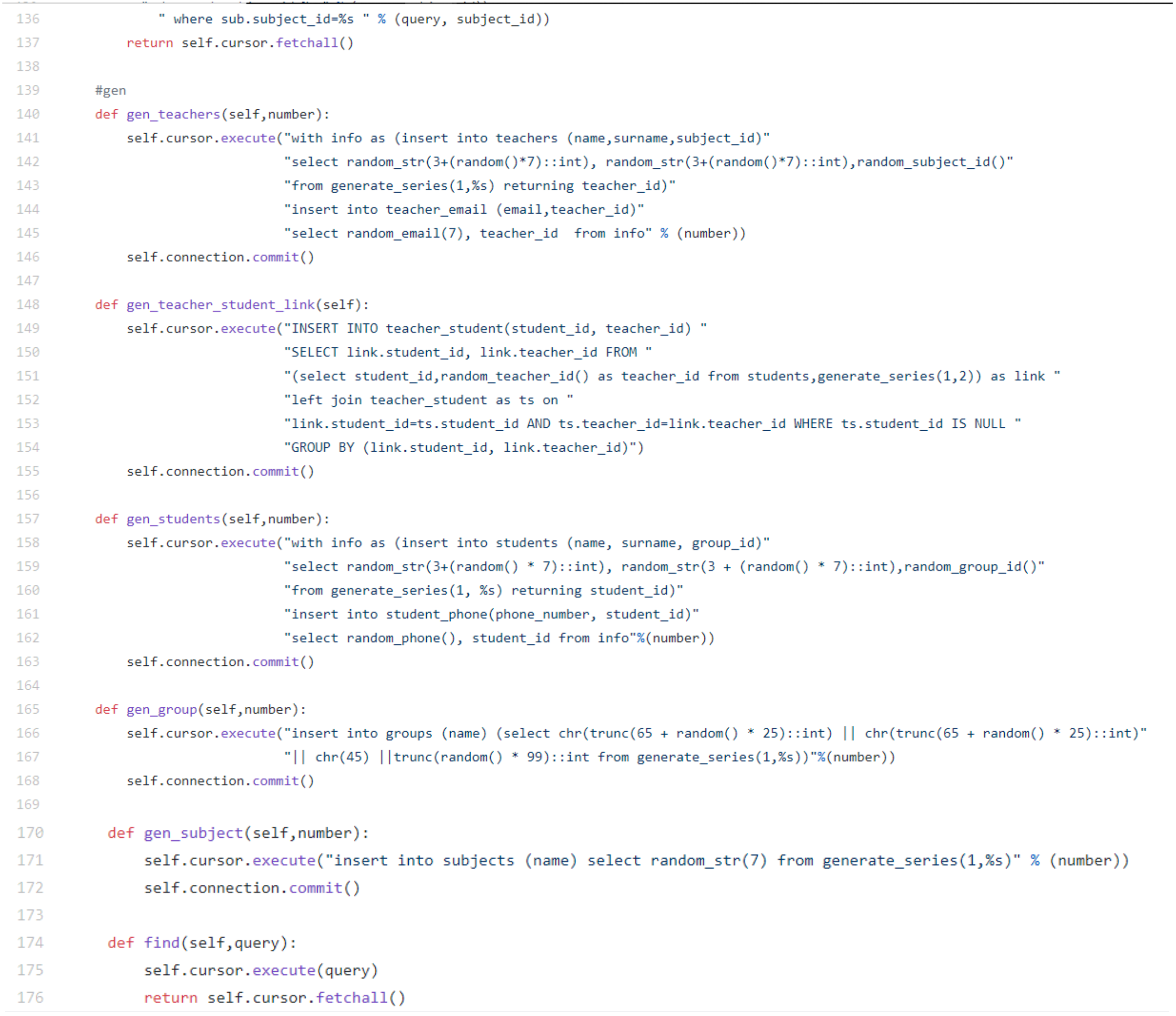
```python
    #group
    def add_group(self,new_group):
        self.cursor.execute("INSERT INTO groups(name) VALUES ('%s') RETURNING group_id"% (new_group))
        self.connection.commit()
        return self.cursor.fetchall()[0][0]

    def update_group(self,upp_group):
        self.cursor.execute("update groups set name = '%s' where group_id=%s" % (upp_group['name'], upp_group['id']))
        self.connection.commit()

    #subject
    def add_subject(self,name):
        self.cursor.execute("INSERT INTO subjects (name) VALUES ('%s') returning subject_id "% (name))
        self.connection.commit()
        return self.cursor.fetchall()[0][0]

    def update_subject(self, id, name):
        self.cursor.execute("update subjects set name = '%s' where subject_id = %s" %(name,id))
        self.connection.commit()

    #student
    def add_student(self,new_student):
        self.cursor.execute("INSERT INTO students (name, surname, \"group_id\" ) VALUES ('%s','%s',%s) returning student_id "
                            % (new_student['name'], new_student['surname'],new_student['group']))
        self.connection.commit()
        return self.cursor.fetchall()[0][0]

    def update_student(self,upp_student):
        self.cursor.execute("update students set name = '%s',surname = '%s',\"group_id\"=%s where student_id = %s" %
                            (upp_student['name'], upp_student['surname'], upp_student['group'], upp_student['id']))
        self.connection.commit()
```



```python
    #student phone
    def add_student_phone(self, new_phone, item_id):
        self.cursor.execute("insert into student_phone (phone_number, student_id) values ('%s',%s)"% (new_phone,item_id))
        self.connection.commit()

    def delete_some_student_phone(self, item_id,phone):
        self.cursor.execute("delete from student_phone where student_id = %s and phone_number = '%s'" % (item_id,phone))
        self.connection.commit()

    def update_students_phones(self,student_id,old_phone,new_phone):
        self.cursor.execute("update student_phone set phone_number = '%s' where student_id = %s and phone_number = '%s'"%
                            (new_phone, student_id,old_phone))
        self.connection.commit()

    #teacher
    def add_teacher(self,new_teacher):
        self.cursor.execute("INSERT INTO teachers(name, surname, subject_id) VALUES ('%s','%s',%s) RETURNING teacher_id"
            % (new_teacher['name'], new_teacher['surname'], new_teacher['subject']))
        self.connection.commit()
        return self.cursor.fetchall()[0][0]

    def update_teacher(self,upp_teacher):
        self.cursor.execute("update teachers set name = '%s', surname = '%s', subject_id = %s where teacher_id=%s" %
            (upp_teacher['name'], upp_teacher['surname'],upp_teacher['subject'], upp_teacher['id']))
        self.connection.commit()

    #teacher email
    def add_teacher_email(self,new_email,item_id):
        self.cursor.execute("insert into teacher_email (email, teacher_id) values ('%s',%s)"% (new_email,item_id))
        self.connection.commit()

    def delete_some_teacher_email(self, item_id,email):
        self.cursor.execute("delete from teacher_email as t where t.teacher_id = %s and t.email = '%s'" % ( item_id,email))
        self.connection.commit()
```



```python
    def update_teacher_email(self,teacher_id,old_email,new_email):
        self.cursor.execute("update teacher_email set email = '%s' where teacher_id = %s and email = '%s'"%(new_email, teacher_id,old_email))
        self.connection.commit()

    #teacher student link
    def get_teacher_student_link(self,teacher_id,student_id):
        self.cursor.execute("select from teacher_student where teacher_id = %s and student_id = %s"% (teacher_id, student_id))
        return self.cursor.fetchall()

    def add_teacher_student_link(self, teacher_id,student_id):
        self.cursor.execute("insert into teacher_student (teacher_id, student_id) values (%s,%s)"% (teacher_id, student_id))
        self.connection.commit()

    def delete_teacher_student_link(self, teacher_id,student_id):
        self.cursor.execute("delete from teacher_student where teacher_id = %s and student_id =%s" %(teacher_id,student_id))
        self.connection.commit()

    #some function
    def get_all_student_teachers(self,student_id,query):
        self.cursor.execute("select t.teacher_id, t.name,t.surname,t.subject_id from teachers as t, students as s, teacher_student as ts %s"
            "where s.student_id = %s and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by t.teacher_id" % (query, student_id))
        return self.cursor.fetchall()

    def get_all_teacher_students(self,teacher_id,query):
        self.cursor.execute("select s.student_id, s.name,s.surname from teachers as t, students as s, teacher_student as ts %s "
                            "where t.teacher_id = %s and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by s.student_id"% (query,teacher_id))
        return self.cursor.fetchall()

    def get_all_student_subjects(self,student_id,query,params):
        self.cursor.execute("select sub.subject_id, sub.name from teachers as t,subjects as sub, students as st, teacher_student as ts "
                            "%s where t.teacher_id=ts.teacher_id and st.student_id = ts.student_id and st.student_id=%s "
                            "and t.subject_id=sub.subject_id %s group by sub.subject_id order by sub.subject_id" % (query, student_id,params))
        return self.cursor.fetchall()

    def get_all_subject_teachers(self,subject_id, query):
        self.cursor.execute(
            "select t.teacher_id,t.name,t.surname from subjects as sub join teachers as t using(subject_id) %s"
```

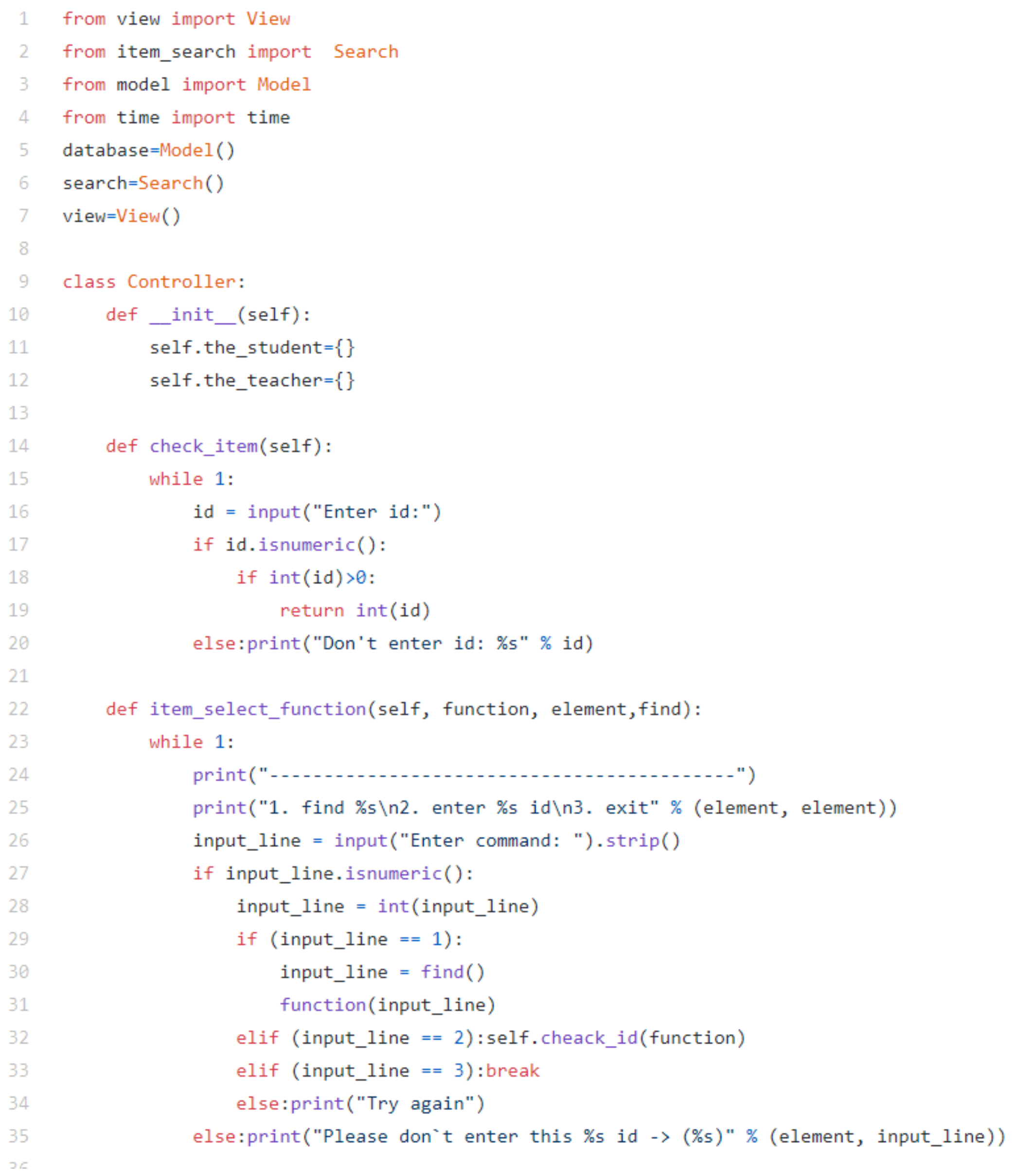


```python
            " where sub.subject_id=%s " % (query, subject_id))
        return self.cursor.fetchall()

    #gen
    def gen_teachers(self,number):
        self.cursor.execute("with info as (insert into teachers (name,surname,subject_id)"
                            "select random_str(3+(random()*7)::int), random_str(3+(random()*7)::int),random_subject_id()"
                            "from generate_series(1,%s) returning teacher_id)"
                            "insert into teacher_email (email,teacher_id)"
                            "select random_email(7), teacher_id  from info" % (number))
        self.connection.commit()

    def gen_teacher_student_link(self):
        self.cursor.execute("INSERT INTO teacher_student(student_id, teacher_id) "
                            "SELECT link.student_id, link.teacher_id FROM "
                            "(select student_id,random_teacher_id() as teacher_id from students,generate_series(1,2)) as link "
                            "left join teacher_student as ts on "
                            "link.student_id=ts.student_id AND ts.teacher_id=link.teacher_id WHERE ts.student_id IS NULL "
                            "GROUP BY (link.student_id, link.teacher_id)")
        self.connection.commit()

    def gen_students(self,number):
        self.cursor.execute("with info as (insert into students (name, surname, group_id)"
                            "select random_str(3+(random() * 7)::int), random_str(3 + (random() * 7)::int),random_group_id()"
                            "from generate_series(1, %s) returning student_id)"
                            "insert into student_phone(phone_number, student_id)"
                            "select random_phone(), student_id from info"%(number))
        self.connection.commit()

    def gen_group(self,number):
        self.cursor.execute("insert into groups (name) (select chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)"
                            "|| chr(45) ||trunc(random() * 99)::int from generate_series(1,%s))"%(number))
        self.connection.commit()

    def gen_subject(self,number):
        self.cursor.execute("insert into subjects (name) select random_str(7) from generate_series(1,%s)" % (number))
        self.connection.commit()

    def find(self,query):
        self.cursor.execute(query)
        return self.cursor.fetchall()
```

View:

```python
from model import Model
from time import time
database=Model()

class View:
    def print_item(self,items):
        print("----------------------------------------------------------------------")
        for item in items:
            print(item)
        print("----------------------------------------------------------------------")

    #student menu
    def print_student(self,student_id):
        student =database.select_item_by_id(student_id,"students","student_id")[0]
        print("ID: %s\t\tFullname: %s %s\tGroup: %s"%(student[0],student[1],student[2],database.select_item_by_id(student[3],"groups","group_id")[0][1]))
        self.print_item(database.select_item_by_id(student_id,"student_phone","student_id"))

    def print_teacher(self, teacher_id):
        student = database.select_item_by_id(teacher_id, "teachers", "teacher_id")[0]
        print("ID: %s\t\tFullname: %s %s\tSubject: %s" % (
        student[0], student[1], student[2], database.select_item_by_id(student[3], "subjects", "subject_id")[0][1]))
        self.print_item(database.select_item_by_id(teacher_id, "teacher_email", "teacher_id"))

    def add_new_student(self, new_student):print("Student successfully added with id ->%s"% database.add_student(new_student))

    def delete_student_by_id(self,student):
        database.delete_item(student, "student_phone", "student_id")
        database.delete_item(student, "teacher_student", "student_id")
        database.delete_item(student,"students","student_id")
        print("Student with id = (%s) successfully deleted"%student)

    def update_student(self, upp_student):
        database.update_student(upp_student)
        print("Student with id ->%s successfully updated" % upp_student['id'])

    def get_all_student_subject(self, student_id,query):
        start_time = time()
        if len(query) != 0:subjects = database.get_all_student_subjects(student_id, "natural join (" + query + ") as l "," and  l.subject_id=sub.subject_id ")
        else:subjects = database.get_all_student_subjects(student_id, query,"")
        final_time=time() - start_time
        self.print_item(subjects)
        print("request execution time: ",final_time)

    def get_all_student_teacher(self,student_id,query):
        start_time = time()
        if len(query) != 0:teachers = database.get_all_student_teachers(student_id,  "natural join (" + query + ") as l ")
        else:teachers = database.get_all_student_teachers(student_id,query)
        final_time=time() - start_time
        self.print_item(teachers)
        print("request execution time: ",final_time)

    #teacher menu
    def add_new_teacher(self, new_teacher):print("Teacher %s successfully added " % (database.add_teacher(new_teacher)))

    def delete_teacher_by_id(self,teacher_id):
        database.delete_item(teacher_id, "teacher_email", "teacher_id")
        database.delete_item(teacher_id, "teacher_student", "teacher_id")
        database.delete_item(teacher_id, "teachers", "teacher_id")
        print("Teacher %s successfully deleted " % (teacher_id))

    def update_tracher(self, upp_teacher):
        database.update_teacher(upp_teacher)
        print("Teacher %s successfully updated " % (upp_teacher['name']))
    def get_all_teacher_students(self,teacher_id,query):
        start_time = time()
        if len(query) != 0: students = database.get_all_teacher_students(teacher_id, "natural join (" + query + ") as l ")
        else: students = database.get_all_teacher_students(teacher_id, query)
        final_time = time() - start_time
        self.print_item(students)
        print("request execution time: ", final_time)

    # teacher email menu
    def add_new_email(self,teacher_id,email):
        database.add_teacher_email(email, teacher_id)
        print("Email: %s successfully added" % email)

    def delete_email(self,teacher_id,email):
        database.delete_some_teacher_email(teacher_id, email)
        print("Email: %s successfully deleted" % email)

    def update_email(self,teacher_id,old_email, new_email):
        database.update_teacher_email(teacher_id, old_email, new_email)
        print("Email successfully updated (%s)->(%s) " % (old_email, new_email))

    # teacher subject menu
    def add_new_subject(self,number):print("Subject successfully added with id-> %s"%database.add_subject(number))

    def delete_subject_by_id(self,subject_id):
        teachers = database.select_item_by_id(subject_id,"teachers","subject_id")
        for teacher in teachers:
            database.delete_item(teacher[0],"teacher_email","teacher_id")
            database.delete_item(teacher[0],"teacher_student","teacher_id")
            database.delete_item(teacher[0],"teachers","teacher_id")
        database.delete_item(subject_id,"subjects","subject_id")
        print("Subject with id: %s successfully deleted" % subject_id)

    def update_subject(self,subject_id, new_name):
        database.update_subject(subject_id,new_name)
        print("Subject successfully updated ->(%s) " % (new_name))
    # student phone menu
    def add_new_phone(self,phone,student_id):
        database.add_student_phone(phone,student_id)
        print("Phone successfully added")

    def delete_phone(self,student_id,phone):
        database.delete_some_student_phone(student_id,phone)
        print("Phone %s successfully deleted" % phone)

    def update_phone(self,student_id,old_number,new_number):
        database.update_students_phones(student_id,old_number,new_number)
        print("Phone successfully updated (%s)->(%s) " %(old_number,new_number))

    #group menu
    def add_new_group(self,new_group):print("Group successfully added with id -> %s" % database.add_group(new_group))

    def delete_group_by_id(self,group_id):
        students = database.select_item_by_id(group_id,"students","group_id")
        for student in students:
            database.delete_item(student[0],"student_phone","student_id")
            database.delete_item(student[0],"teacher_student","student_id")
            database.delete_item(student[0],"students","student_id")
        database.delete_item(group_id,"groups","group_id")
        print("Group with id: %s successfully deleted" % group_id)

    def update_group(self,group):
        database.update_group(group)
        print("Group (%s) successfully updated"%group['name'])

    #st link
    def add_new_student_teacher_link(self,teacher_id,student_id):
        try: database.add_teacher_student_link(teacher_id, student_id)
        except: print("Teacher (%s) Student (%s) link successfully added" % teacher_id, student_id)

    def delete_student_teacher_link(self,teacher_id,student_id):
        try: database.delete_teacher_student_link(teacher_id,student_id)
        except: print("Teacher (%s) Student (%s) link successfully deleted" % teacher_id, student_id)

    #gen menu
    def gen_teacher(self,number):
        start_time = time()
        database.gen_subject(int(number / 30) + 1)
        database.gen_teachers(number)
        final_time = time() - start_time
        print("request execution time: ", final_time,"\n%s teachers successfully added" % number)

    def gen_student(self,number):
        start_time = time()
        database.gen_group(int(number / 30) + 1)
        database.gen_students(number)
        final_time = time() - start_time
        print("request execution time: ", final_time, "\n%s student successfully added" % number)

    def gen_link(self):
        start_time = time()
        database.gen_teacher_student_link()
        final_time = time() - start_time
        print("request execution time: ", final_time, "\nteacher-student links successfully added")

    def get_all_subject_teachers(self,subject_id, query):
        start_time = time()
        if len(query) != 0:teachers = database.get_all_subject_teachers(subject_id, "join ("+query+") as l using(teacher_id)")
        else:teachers = database.get_all_subject_teachers(subject_id, query)
        final_time = time() - start_time
        self.print_item(teachers)
        print("request execution time: ", final_time)
```

Controller:



```python
from view import View
from item_search import  Search
from model import Model
from time import time
database=Model()
search=Search()
view=View()

class Controller:
    def __init__(self):
        self.the_student={}
        self.the_teacher={}

    def check_item(self):
        while 1:
            id = input("Enter id:")
            if id.isnumeric():
                if int(id)>0:
                    return int(id)
            else:print("Don't enter id: %s" % id)

    def item_select_function(self, function, element,find):
        while 1:
            print("-------------------------------------------")
            print("1. find %s\n2. enter %s id\n3. exit" % (element, element))
            input_line = input("Enter command: ").strip()
            if input_line.isnumeric():
                input_line = int(input_line)
                if (input_line == 1):
                    input_line = find()
                    function(input_line)
                elif (input_line == 2):self.cheack_id(function)
                elif (input_line == 3):break
                else:print("Try again")
            else:print("Please don`t enter this %s id -> (%s)" % (element, input_line))
    def cheack_id(self,function):
        while 1:
            id = input("Enter id:").strip()
            if id.isnumeric():
                function(int(id))
                break
            else:print("Don't enter id: %s" % id)

    def cheack_human(self,item,type):
        print("Enter %s params:"%type)
        while 1:
            item['name'] = input("Name: ").strip()
            if len(item['name']) == 0: print("Name is empty")
            else:break
        while 1:
            item['surname'] = input("Surname: ").strip()
            if len(item['surname']) == 0: print("Surname is empty")
            else:return item

    # student menu
    def add_new_student(self):
        if database.select_items("groups"):
            self.the_student=self.cheack_human(self.the_student,"student")
            search.find_group()
            groups = database.find(search.create_query())
            view.print_item(groups)
            self.the_student['group'] = False
            while not self.the_student['group']:
                id = self.check_item()
                for group in groups:
                    if (group[0] == id):
                        self.the_student['group'] = id
                        break
            view.add_new_student(self.the_student)
        else:print("Please add some group before adding student")

    def find_student(self):
        search.student_params()
        start_time = time()
        students = database.find(search.create_query())
        finish_time = time() - start_time
        view.print_item(students)
        print("request execution time: %s" % finish_time,
              "\n----------------------------------------------------------------------")
        id = self.check_item()
        for student in students:
            if (student[0] == id):
                return id
        print("Student with id = (%s) is not included in the list of found students" % id)
        return 0

    def print_student(self,student_id):
        if database.select_item_by_id(student_id, "students", "student_id"):view.print_student(student_id)
        else:print("Student with id: %s not found" % student_id)

    def print_student_by_id(self):self.item_select_function(self.print_student,"student",self.find_student)

    def delete_student(self,student_id):
        if database.select_item_by_id(student_id, "students", "student_id"):view.delete_student_by_id(student_id)
        else:print("Student with id: %s not found" % student_id)

    def delete_student_by_id(self):self.item_select_function(self.delete_student,"student",self.find_student)
    def update_student(self,student_id):
        if database.select_item_by_id(student_id,"students","student_id"):
            student = database.select_item_by_id(student_id,"students","student_id")
            self.the_student['id'] = student_id
            print("Enter params:")
            while 1:
                self.the_student['name'] = input("Name: ").strip()
                if len(self.the_student['name']) == 0: self.the_student['name'] = student[0][1]
                break
            while 1:
                self.the_student['surname'] = input("Surname: ").strip()
                if len(self.the_student['surname']) == 0: self.the_student['surname'] = student[0][2]
                break
            search.find_group()
            groups = database.find(search.create_query())
            view.print_item(groups)
            self.the_student['group'] = student[0][3]
            id = input("Enter id:").strip()
            if len(id) != 0 and id.isnumeric():
                for group in groups:
                    if (group[0] == int(id)):
                        self.the_student['group'] = int(id)
                        return
            view.update_student(self.the_student)
        else:print("Student with id: %s not found" % student_id)

    def update_student_by_id(self):self.item_select_function(self.update_student,"student",self.find_student)

    def get_all_student_teacher(self,student_id):
        if database.select_item_by_id(student_id, "students", "student_id"):
            query = ""
            input_line = input("Enter y to set teacher parameters: ")
            if (input_line == 'y'):
                search.teacher_params()
                query = search.create_query()
            view.get_all_student_teacher(student_id, query)
        else:print("Student with id: %s not found" % student_id)
    def get_all_student_subject(self, student_id):
        if database.select_item_by_id(student_id, "students", "student_id"):
            query = ""
            input_line = input("Enter y to set subject parameters: ")
            if (input_line == 'y'):
                search.find_subject()
                query = search.create_query()
            view.get_all_student_subject(student_id, query)
        else:print("Student with id: %s not found" % student_id)

    def get_all_student_by_id_subject(self):self.item_select_function(self.get_all_student_subject,"student",self.find_student)

    # teacher menu1
    def add_new_teacher(self):
        self.the_teacher = self.cheack_human(self.the_teacher, "teacher")
        search.find_subject()
        subjects=database.find(search.create_query())
        view.print_item(subjects)
        self.the_teacher['subject'] = False
        while not self.the_teacher['subject']:
            id = self.check_item()
            for subject in subjects:
                if (subject[0] == id):
                    self.the_teacher['subject'] = id
                    break
        view.add_new_teacher(self.the_teacher)

    def find_teacher(self):
        search.teacher_params()
        start_time=time()
        teachers = database.find(search.create_query())
        finish_time=time()-start_time
        view.print_item(teachers)
        print("request execution time: %s"%finish_time,"\n----------------------------------------------------------------------")
        id = self.check_item()
        for teacher in teachers:
            if (teacher[0] == id):
                return id
        print("Teacher with id = (%s) is not included in the list of found teachers" % id)
        return 0

    def print_teacher(self,teacher_id):
        if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):view.print_teacher(teacher_id)
        else:print("Teacher with id: %s not found" % teacher_id)

    def print_teacher_by_id(self):self.item_select_function(self.print_teacher, "teacher", self.find_teacher)

    def delete_teacher_by_id(self):self.item_select_function(view.delete_teacher_by_id,"teacher",self.find_teacher)

    def update_tracher(self, teacher_id):
        if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
            teacher = database.select_item_by_id(teacher_id,"teachers","teacher_id")
            self.the_teacher['id'] = teacher_id
            print("Enter teacher params:")
            while 1:
                self.the_teacher['name'] = input("Name: ").strip()
                if len(self.the_teacher['name']) == 0: self.the_teacher['name'] = teacher[0][1]
                break
        while 1:
            self.the_teacher['surname'] = input("Surname: ").strip()
            if len(self.the_teacher['surname']) == 0: self.the_teacher['surname'] = teacher[0][2]
            break
        search.find_subject()
        subjects = database.find(search.create_query())
        view.print_item(subjects)
        self.the_teacher['subject'] = teacher[0][3]
        id = input("Enter id:").strip()
        if len(id) != 0 and id.isnumeric():
            for subject in subjects:
                if (subject[0] == int(id)):
                    self.the_teacher['subject'] = int(id)
                    return
        view.update_tracher(self.the_teacher)
```

```
    def get_all_teacher_students(self,teacher_id):
        if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
            query=""
            input_line=input("Enter y to set student parameters: ")
            if(input_line=='y'):
                search.student_params()
                query=search.create_query()
            view.get_all_teacher_students(teacher_id,query)
        else:print("Teacher with id-> %s not found"%teacher_id)

    def get_all_subject_teachers(self):
        search.find_subject()
        view.print_item(database.find(search.create_query()))
        subject_id=self.check_item()
        if database.select_item_by_id(subject_id,"subjects","subject_id"):
            query = ""
            input_line = input("Enter y to set teacher parameters: ")
            if (input_line == 'y'):
                search.teacher_params()
                query = search.create_query()
            view.get_all_subject_teachers(subject_id, query)
        else:print("Subject with id-> %s not found"%subject_id)

    def get_all_teacher_by_id_students(self):self.item_select_function(self.get_all_teacher_students, "teacher", self.find_teacher)
    # email menu
    def add_new_email(self,teacher_id):
        if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
            while 1:
                number = input("Enter new email: ")
                if len(number) != 0:
                    view.add_new_email(teacher_id,number)
                    break
                else:print("Name is empty")
        else:print("Teacher with id: %s not found" % teacher_id)
```

```python
    def add_new_email_for_teacher(self):self.item_select_function(self.add_new_email, "teacher", self.find_teacher)

    def delete_email(self,teacher_id):
        if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
            try:
                number = input("Enter email adress: ").strip()
                view.delete_email(teacher_id, number)
            except:print("you enter bad email")
        else:print("Teacher with id: %s not found" % teacher_id)

    def delete_email_for_teacher(self):self.item_select_function(self.delete_email, "teacher", self.find_teacher)

    def update_email(self,teacher_id):
        if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
            try:
                number1 = input("Enter old email adress: ").strip()
                while len(number1) == 0:
                    print("Email is empty")
                    number1 = input("Enter old email adress: ").strip()
                number2 = input("Enter new email adress: ").strip()
                while len(number2) == 0:
                    print("Email is empty")
                    number2 = input("Enter new email adress: ").strip()
                view.update_email(teacher_id, number1, number2)
            except:print("Email %s isn`t exist"%number1)
        else:print("Teacher with id: %s not found" % teacher_id)

    def update_email_for_teacher(self): self.item_select_function(self.update_email, "teacher", self.find_teacher)

    # subject menu
    def add_new_subject(self):
        while 1:
            number = input("Enter subject name:")
            if len(number) != 0:
                view.add_new_subject(number)
                break
            else:print("Name is empty")

    def delete_subject_by_id(self):
        subject_id = self.check_item()
        if database.select_item_by_id(subject_id,"subjects","subject_id"):view.delete_subject_by_id(subject_id)
        else:print("Subject with id: %s not found" % subject_id)

    def update_subject(self):
        subject_id = self.check_item()
        if database.select_item_by_id(subject_id,"subjects","subject_id"):
            while 1:
                number = input("Enter new subject name:")
                if len(number) != 0:
                    view.update_subject(subject_id, number)
                    break
                else:
                    print("Name is empty")
        else:print("Subject with id: %s not found" % subject_id)

    # phone menu
    def add_new_phone(self,student_id):
        if database.select_item_by_id(student_id,"students","student_id"):
            while 1:
                number = input("Enter phone number: ")
                if len(number) != 0:
                    view.add_new_phone(number,student_id)
                    break
                else:print("Phone is empty")
        else:print("Student with id: %s not found" % student_id)

    def add_new_phone_for_student(self):self.item_select_function(self.add_new_phone,"student",self.find_student)

    def delete_phone(self,student_id):
        if database.select_item_by_id(student_id,"students","student_id"):
            number = input("Enter phone number: ").strip()
            try:
                number=int(number)
                view.delete_phone(student_id,number)
            except:print("Don`t enter this student phone -> %s" % number)
        else:print("Student with id: %s not found" % student_id)

    def delete_phone_for_student(self):self.item_select_function(self.delete_phone,"student",self.find_student)

    def update_phone(self,student_id):
        if database.select_item_by_id(student_id,"students","student_id"):
            try:
                number = input("Enter phone number: ").strip()
                while len(number) == 0:
                    print("phone is empty")
                    number = input("Enter phone number: ").strip()
                new_number = input("Enter new phone: ").strip()
                while len(new_number) == 0:
                    print("phone is empty")
                    new_number = input("Enter new phone: ").strip()
                view.update_phone(student_id, number, new_number)
            except:print("Phone %s isn`t exist"%number)
        else:print("Student with id: %s not found" % student_id)

    def update_phone_for_student(self):self.item_select_function(self.update_phone, "student", self.find_student)

    # group menu
    def add_new_group(self):
        while 1:
            number = input("Enter group name:")
            if len(number) != 0:
                view.add_new_group(number)
                break
            else:print("Name is empty")

    def delete_group_by_id(self):
        group_id = self.check_item()
        if database.select_item_by_id(group_id,"groups","group_id"):view.delete_group_by_id(group_id)
        else:print("Group with id-> (%s) don`t found"%group_id)
    def update_group(self):
        group_id = self.check_item()
        if database.select_item_by_id(group_id,"groups","group_id"):
            group = {};
            group['id'] = group_id
            number = input("Enter group name:").strip()
            while 1:
                if len(number) != 0:
                    group['name'] = number
                    view.update_group(group)
                    break
                else:print("Name is empty")
        else:print("Group with id-> (%s) don`t found" % group_id)

    #generation menu
    def generation_teacher(self):
        while 1:
            number = input("Enter count generation teacher:")
            if len(number) != 0 and number.isnumeric():
                view.gen_teacher(int(number))
                break
            else:print("enter another teacher count")

    def generation_student(self):
        while 1:
            number = input("Enter count generation student:")
            if len(number) != 0 and number.isnumeric():
                view.gen_student(int(number))
                break
            else:print("enter another teacher count")

    def generation_links(self): view.gen_link()
```

```python
    #link menu
    def add_new_student_teacher_link(self):
        student_id = self.find_student()
        if database.select_item_by_id(student_id, "students", "student_id"):
            teacher_id = self.find_teacher()
            if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):
                if (not database.get_teacher_student_link(teacher_id, student_id)): view.add_new_student_teacher_link(teacher_id, student_id)
                else:print("Link olready exist")
            else:print("Teacher with id-> (%s) not found" % teacher_id)
        else:print("Student with id-> (%s) not found"%student_id)

    def delete_student_teacher_link(self):
        student_id = self.find_student()
        if database.select_item_by_id(student_id, "students", "student_id"):
            teacher_id = self.find_teacher()
            if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):
                if (database.get_teacher_student_link(teacher_id, student_id)):
                    view.delete_student_teacher_link(teacher_id,student_id)
                else:print("Link don`t exist")
            else:print("Teacher with id-> (%s) not found" % teacher_id)
        else:print("Student with id-> (%s) not found"%student_id)
```