НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп’ютерних систем

Лабораторна робота 2

з дисципліни
«Бази даних і засоби управління»

Тема: «Проектування бази даних та ознайомлення збазовими операціями СУБД PostgreSQL»

Виконав: студент III курсу

ФПМ групи КВ-94

Чекмезов Г. В.

Перевірив: Петрашенко А.В.

Київ 2021

Загальне завдання роботи:

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).

GitHub: https://github.com/glebbovski/DataBase/tree/main/Lab2

Мова програмування: Python.

Використані бібліотеки: psycopg2, time

“Сутність-зв’язок”

Сутності

Електронна пошта

Користувачі

Повідомлення

Папки

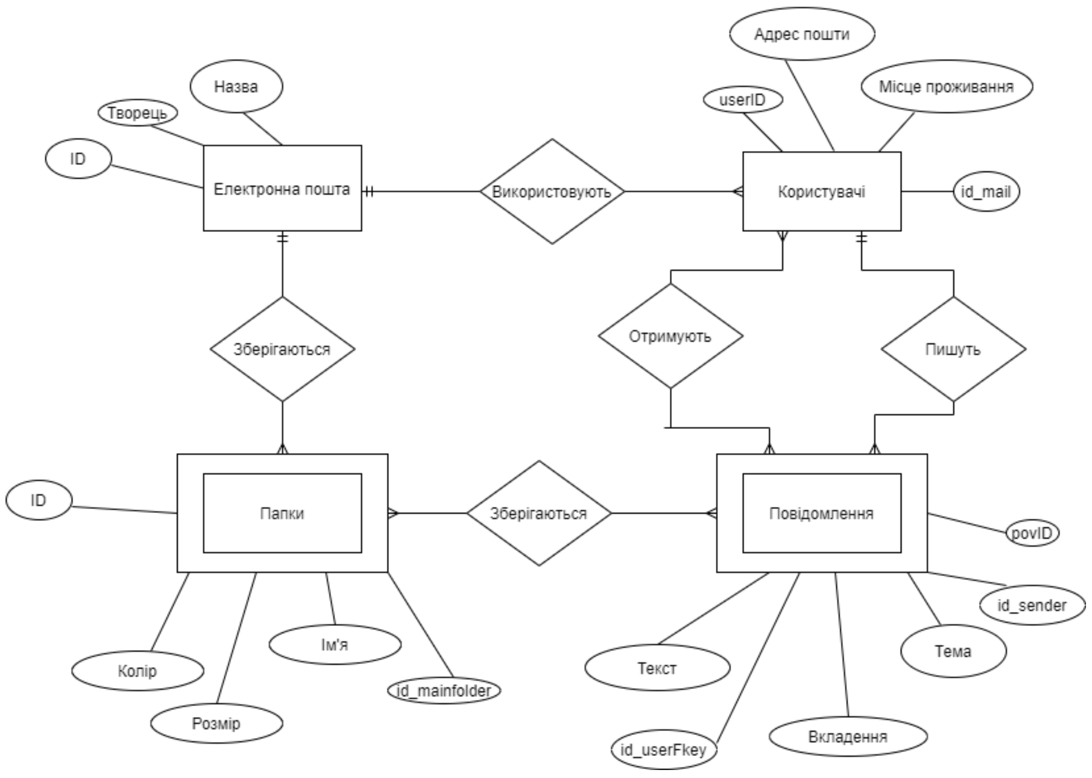
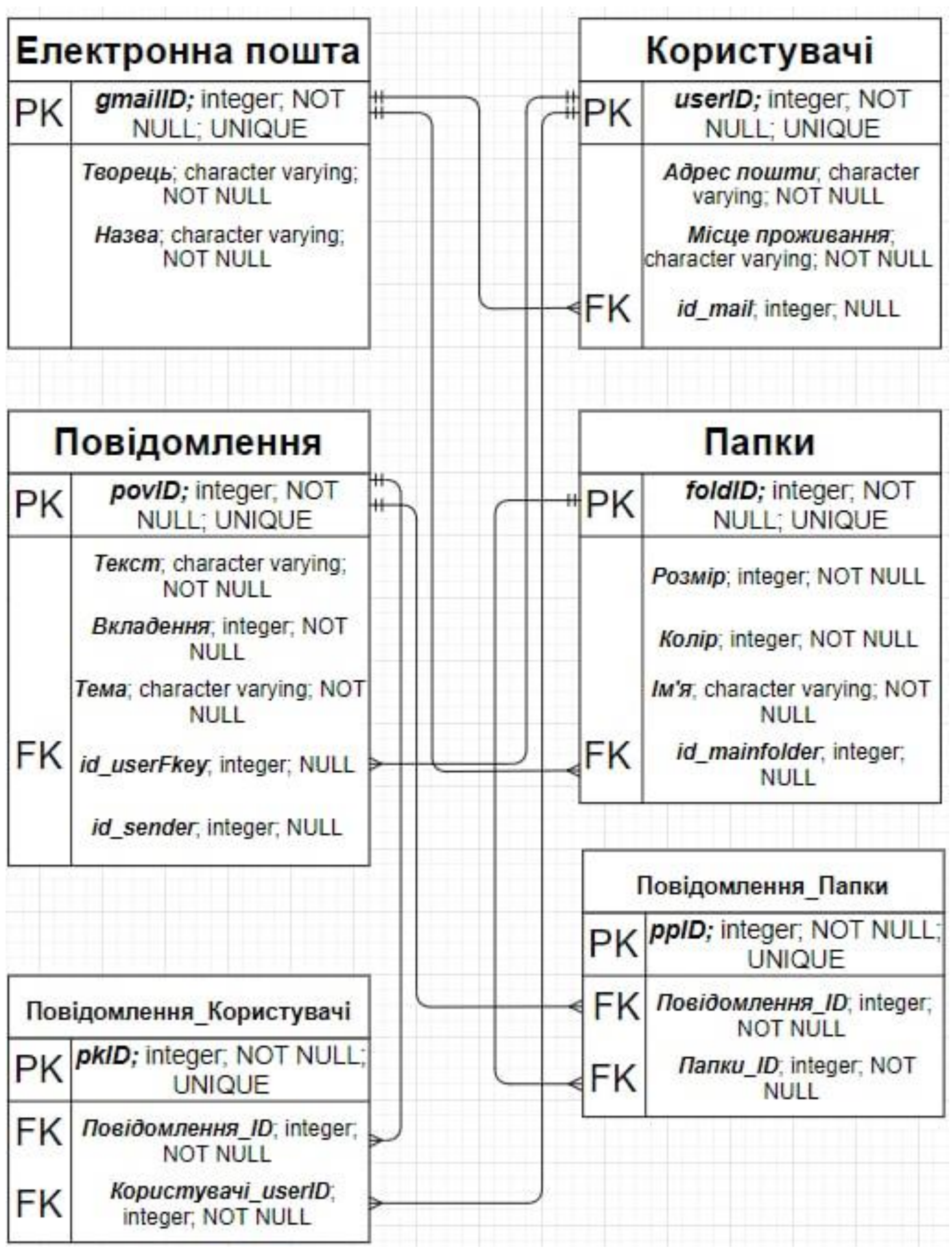


Схема бази даних у графічному вигляді:


Електронна пошта
PK gmailID; integer; NOT NULL; UNIQUE
Творець; character varying; NOT NULL
Назва; character varying; NOT NULL
Користувачі
PK userID; integer; NOT NULL; UNIQUE
Адрес пошти; character varying; NOT NULL
Місце проживання; character varying; NOT NULL
FK id_mail; integer; NULL
Повідомлення
PK povID; integer; NOT NULL; UNIQUE
Текст; character varying; NOT NULL
Вкладення; integer; NOT NULL
Тема; character varying; NOT NULL
FK id_userFkey; integer; NULL
id_sender; integer; NULL
Папки
PK foldID; integer; NOT NULL; UNIQUE
Розмір; integer; NOT NULL
Колір; integer; NOT NULL
Ім'я; character varying; NOT NULL
FK id_mainfolder; integer; NULL
Повідомлення_Папки
PK ppID; integer; NOT NULL; UNIQUE
FK Повідомлення_ID; integer; NOT NULL
FK Папки_ID; integer; NOT NULL
Повідомлення_Користувачі
PK pkID; integer; NOT NULL; UNIQUE
FK Повідомлення_ID; integer; NOT NULL
FK Користувачі_userID; integer; NOT NULL

Опис бази даних:

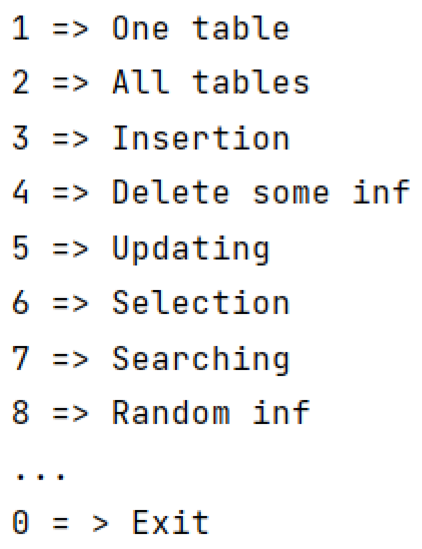
У даному випадку маємо 4 сутності: Електронна пошта, Користувачі, Повідомлення, Папки.

Перша сутність “Електронна пошта” потрібна для оброблення інформації, яку саме електронну пошту використовує користувач у даний момент часу (хто є автором даної пошти, її назва).

Друга сутність – “Користувачі”. Використовується для ведення обліку користувачів пошти шляхом ідентифікації. Також містить інформацію про унікальний поштовий адрес кожного користувача та про місце проживання.
Третя сутність називається “Повідомлення”. Використовується для ведення обліку усіх повідомлень, відправлених чи отриманих певним користувачем та визначення окремих особливостей пошти, таких як: ID, тему, вкладення та текст.
Четверта сутність – “Папки”. Необхідна для ведення обліку папок, які містять різні повідомлення, на певній електронній пошті. Має такі характерні риси як колір, ім’я, розмір та ID.

Опис меню програми:

Меню складається з 9 пунктів:



```
1 => One table
2 => All tables
3 => Insertion
4 => Delete some inf
5 => Updating
6 => Selection
7 => Searching
8 => Random inf
...
0 = > Exit
```

1) One table – вивід на екран однієї таблиці, яку обере користувач.
2) All tables – вивід на екран усіх таблиць.
3) Insertion – вставка у вибрану користувачем таблицю нового рядка.
4) Delete some inf – видалення одного або декількох рядків з обраної таблиці.
5) Updating – оновлення даних у будь-якому рядку, який обере користувач у конкретній таблиці.
6) Selection – формування запитів для фільтрації трьома способами.
7) Random inf – заповнення таблиць випадковими даними.

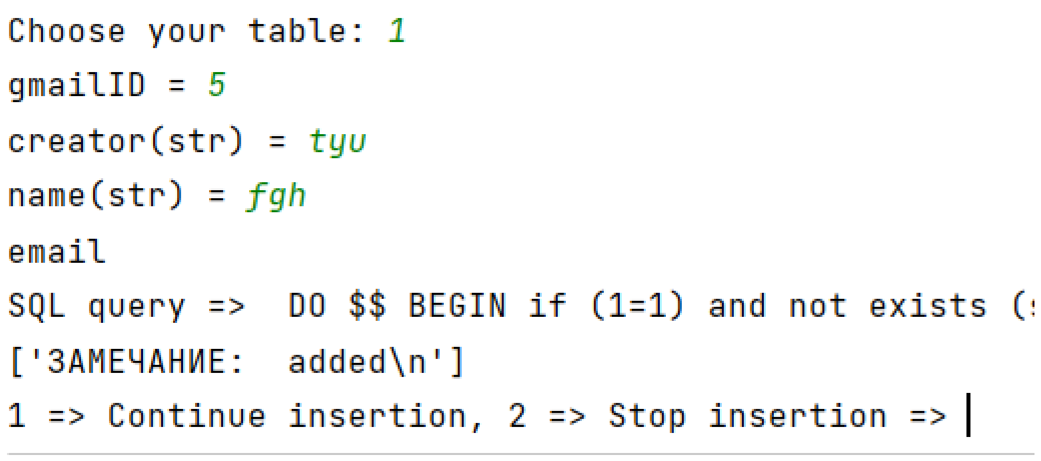
8) Exit – завершення роботи програми.

Завдання 1

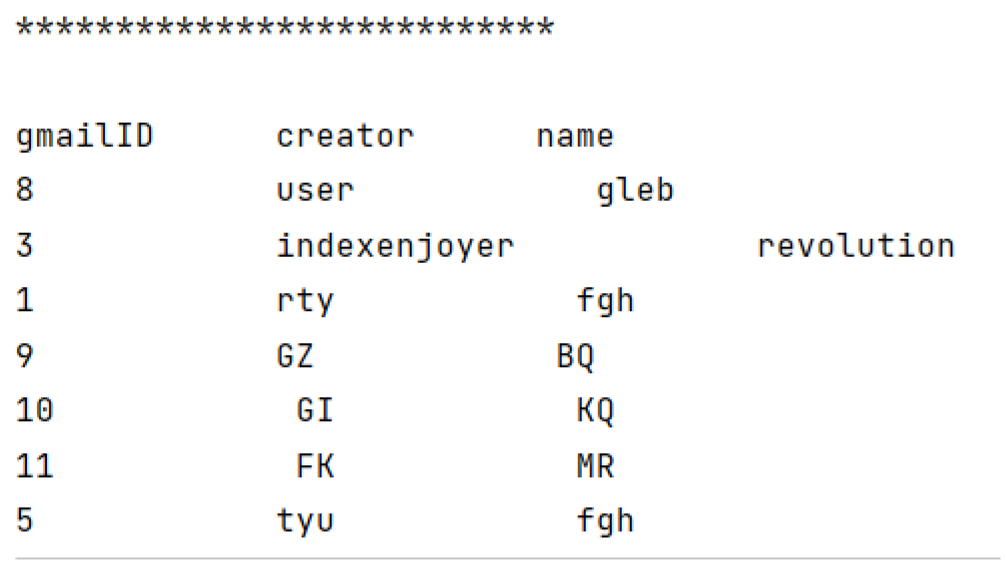
Insert

На прикладі батьківської таблиці Email та дочірньої Folders

Запис у Email:

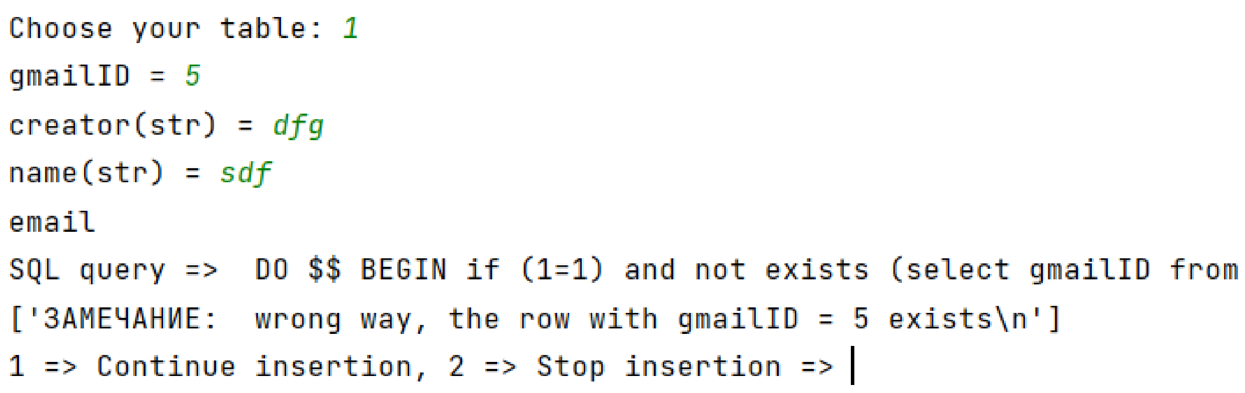


```
Choose your table: 1
gmailID = 5
creator(str) = tyu
name(str) = fgh
email
SQL query =>  DO $$ BEGIN if (1=1) and not exists (
['ЗАМЕЧАНИЕ:  added\n']
1 => Continue insertion, 2 => Stop insertion =>
```



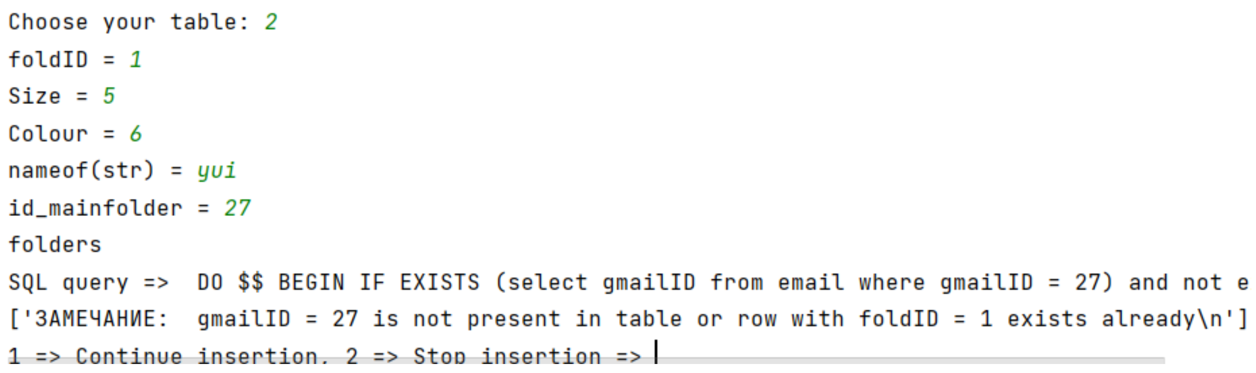
```
***************************

gmailID      creator      name
8            user            gleb
3            indexenjoyer            revolution
1            rty            fgh
9            GZ            BQ
10            GI            KQ
11            FK            MR
5            tyu            fgh
```

Запис рядка с первинним ключем, який вже знаходиться у таблиці:



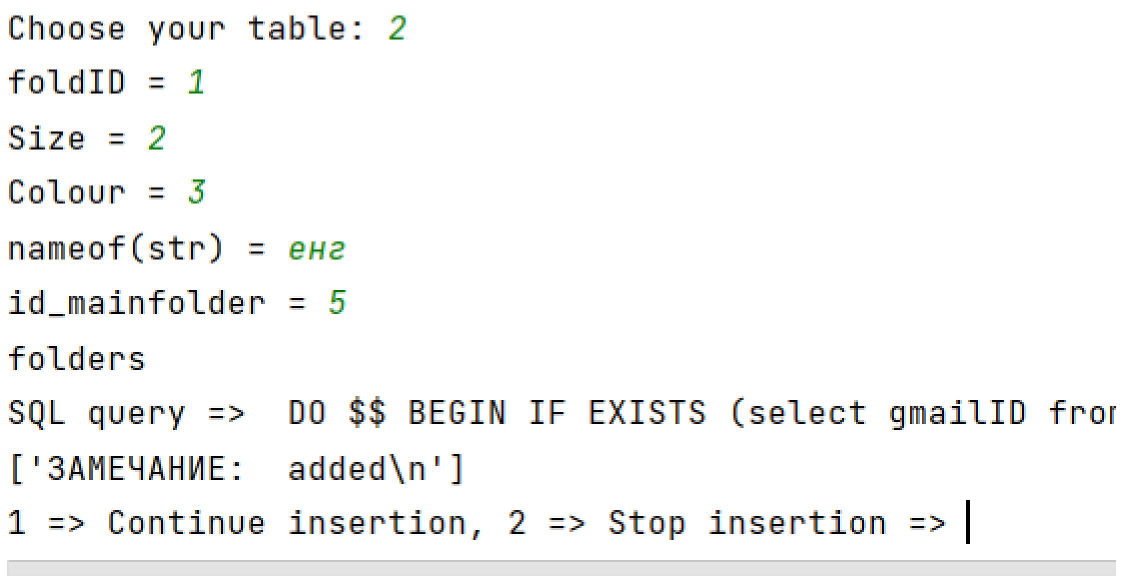
```
Choose your table: 1
gmailID = 5
creator(str) = dfg
name(str) = sdf
email
SQL query =>  DO $$ BEGIN if (1=1) and not exists (select gmailID from
['ЗАМЕЧАНИЕ:  wrong way, the row with gmailID = 5 exists\n']
1 => Continue insertion, 2 => Stop insertion =>
```

Спроба запису у дочірню таблицю з вторинним ключем, який не відповідає первинному батьківської:

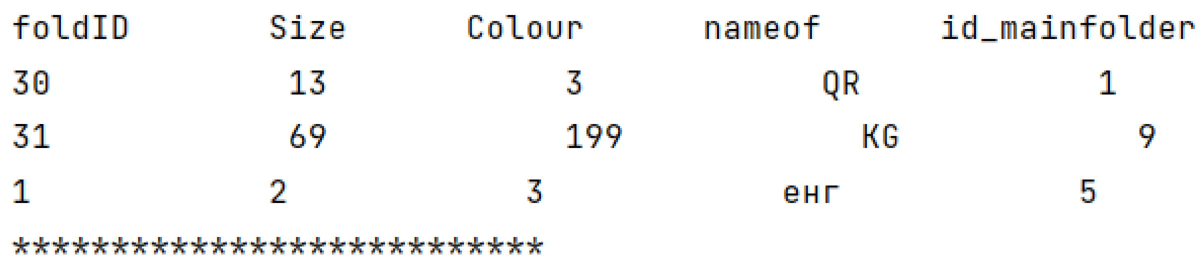


```
Choose your table: 2
foldID = 1
Size = 5
Colour = 6
nameof(str) = yui
id_mainfolder = 27
folders
SQL query =>  DO $$ BEGIN IF EXISTS (select gmailID from email where gmailID = 27) and not e
['ЗАМЕЧАНИЕ:  gmailID = 27 is not present in table or row with foldID = 1 exists already\n']
1 => Continue insertion, 2 => Stop insertion =>
```

Запис з ключем, який відповідає первинному:



```
Choose your table: 2
foldID = 1
Size = 2
Colour = 3
nameof(str) = енг
id_mainfolder = 5
folders
SQL query =>  DO $$ BEGIN IF EXISTS (select gmailID fror
['ЗАМЕЧАНИЕ:  added\n']
1 => Continue insertion, 2 => Stop insertion =>
```



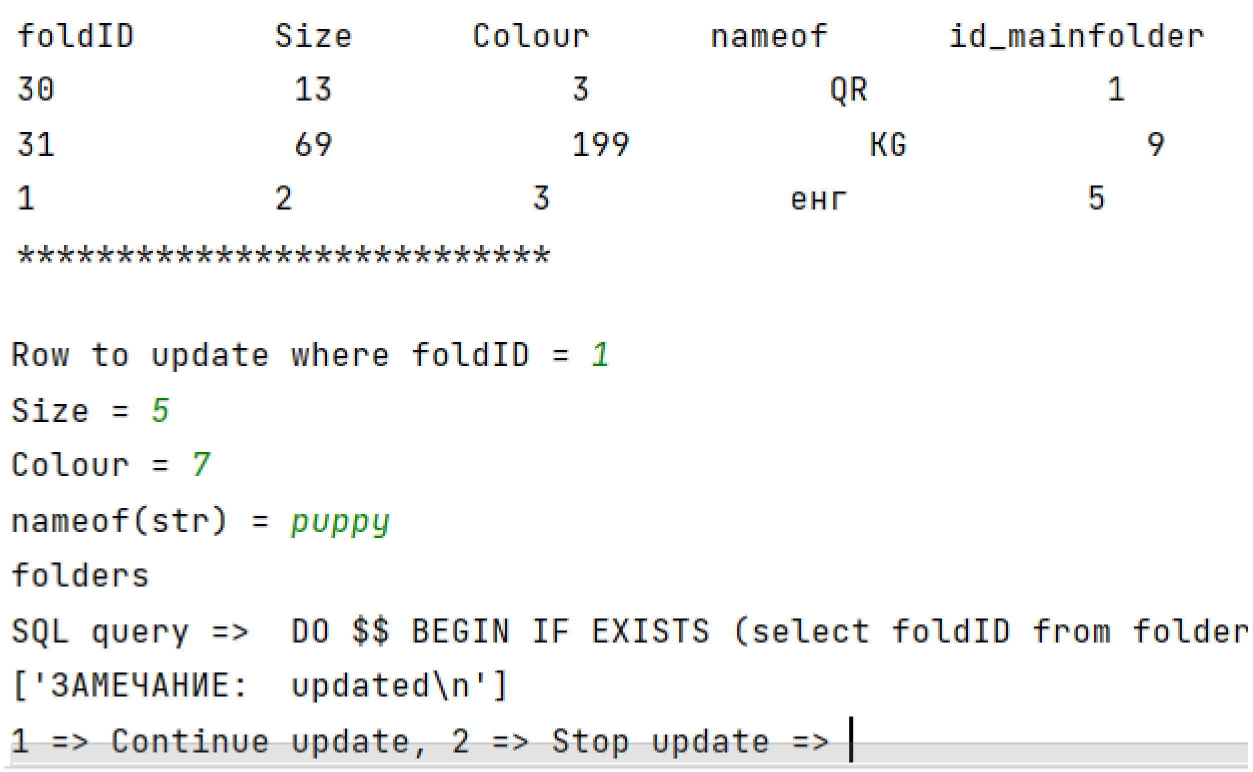
```
foldID         Size        Colour        nameof        id_mainfolder
30              13              3             QR              1
31              69            199               KG              9
1              2             3             енг             5
***************************
```

Лістинг для Insert:

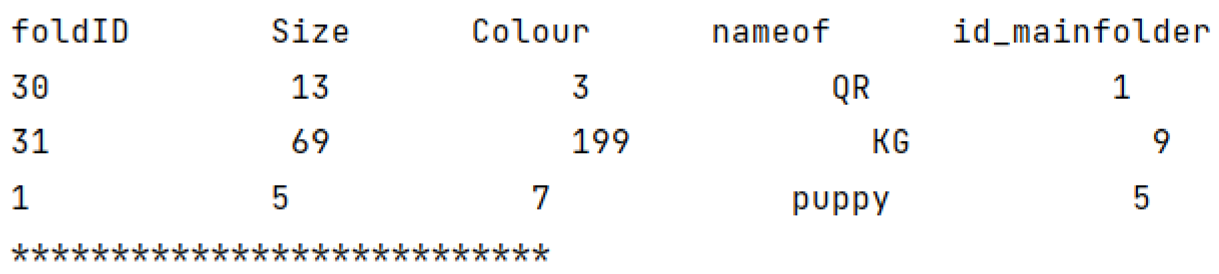
```
def insertbyuser():
    connect = connection.connection()
    cursor = connect.cursor()
    check = True
    while check:
        View.listof()
        table = Model.existingtable()

        if table == 1:
            f = input('gmailID = ')
            s = input('creator(str) = ')
            t = input('name(str) = ')
            added = 'added'
            added = "'" + added + "'"
            notice = 'wrong way, the row with gmailID = {} exists'.format(f)
            notice = "'" + notice + "'"
            if str(f).isdigit() and s.isalnum() and t.isalnum():
                s = "'" + s + "'"
                t = "'" + t + "'"
                insert = 'DO $$ BEGIN if (1=1) and not exists (select gmailID
from email where gmailID = {}) then INSERT INTO email(gmailID, creator, name)
VALUES ({},{},{}); ' \
                     'raise notice {}; else raise notice {}; ' \
                     'end if; end $$;'.format(f, f, s, t, added, notice)
                check = False
            else:
                print('The values are wrong')


        elif table == 2:
            f = input('foldID = ')
            s = input('Size = ')
            t = input('Colour = ')
            fouth = input('nameof(str) = ')
            fifth = input('id_mainfolder = ')
            notice = 'gmailID = {} is not present in table or row with foldID
= {} exists already'.format(fifth,f)
            notice = "'" + notice + "'"
            added = 'added'
            added = "'" + added + "'"

           # insert = 'INSERT INTO folders(foldID, Size, Colour, nameof,
id_mainfolder) VALUES ({},{},{},{},{})'.format(f, s, t,fouth,fifth)
            if str(f).isdigit() and str(s).isdigit() and str(t).isdigit() and
fouth.isalnum() and str(fifth).isdigit():
                fouth = "'" + fouth + "'"
                insert = 'DO $$    BEGIN IF EXISTS (select gmailID from email
where gmailID = {}) and not exists (select foldId from folders where foldId =
{}) THEN ' \
                     'INSERT INTO folders(foldID, Size, Colour, nameof,
id_mainfolder) values ({}, {}, {}, {}, {}); ' \
                     'RAISE NOTICE {};' \
                     ' ELSE RAISE NOTICE {};' \
                     'END IF; ' \
                        'END $$;'.format(fifth,f, f, s, t, fouth, fifth,
added, notice)
                check = False
            else:
                print('The values are wrong')

        elif table == 3:
            f = input('userID = ')
            s = input('adress(str) = ')
            t = input('place(str) = ')

            fouth = input('id_mail = ')
            notice = 'gmailID = {} is not present in table or userID = {}
exists already'.format(fouth, f)
            notice = "'" + notice + "'"
            added = 'added'
            added = "'" + added + "'"

            if str(f).isdigit() and s.isalnum() and t.isalnum() and
str(fouth).isdigit():
                s = "'" + s + "'"
                t = "'" + t + "'"
                insert = 'DO $$ BEGIN IF EXISTS (select gmailID from email
where gmailID = {}) and not exists (select userID from users where userID =
{}) THEN ' \
                     'INSERT INTO users(userID, adress, place, id_mail)
values ({}, {}, {}, {}); ' \
                     'RAISE NOTICE {};' \
                     ' ELSE RAISE NOTICE {};' \
                     'END IF; ' \
                    'END $$;'.format(fouth,f, f, s, t, fouth,added, notice)
                check = False
            else:
                print('The values are wrong')

        elif table == 4:
            f = input('povID = ')
            s = input('text(str) = ')
            t = input('addfiles = ')
            fouth = input('title(str) = ')
            fifth = input('id_userfkey = ')
            sixth = input('id_sender = ')
            notice = 'userID = {} is not present in table or povID = {}
exists already'.format(fifth, f)
            notice = "'" + notice + "'"
            added = 'added'
            added = "'" + added + "'"

          #  insert = 'INSERT INTO notifications(povID, text, addfiles,
title, id_userfkey, id_sender) VALUES ({},{},{},{}, {}, {})'.format(
           #     f, s, t, fouth, fifth, sixth)
            if str(f).isdigit() and s.isalnum() and str(t).isdigit() and
fouth.isalnum() and str(fifth).isdigit() and str(sixth).isdigit():
                s = "'" + s + "'"
                fouth = "'" + fouth + "'"

                insert = 'DO $$    BEGIN IF EXISTS (select userID from users
where userID = {}) and not exists (select povID from notifications where
povID = {}) THEN ' \
                     'INSERT INTO notifications(povID, text, addfiles, title,
id_userfkey, id_sender) values ({}, {}, {}, {}, {}, {}); ' \
                     'RAISE NOTICE {};' \
                     ' ELSE RAISE NOTICE {};' \
                     'END IF; ' \
                    'END $$;'.format(fifth, f, f, s, t, fouth, fifth, sixth,
added, notice)
                check = False
            else:
                print('The values are wrong')


        elif table == 5:
            f = input('ppID = ')
            s = input('notifications_ID = ')
            t = input('folders_ID = ')
            notice = 'foldID = {} or povID = {} or both is not present in
tables. Or ppId = {} exists already'.format(t, s, f)
            notice = "'" + notice + "'"
            added = 'added'
            added = "'" + added + "'"

           # insert = 'INSERT INTO folders_notifications(ppID,
notifications_ID, folders_ID) VALUES ({},{},{})'.format(
           #    f, s, t)
            if str(f).isdigit() and str(s).isdigit() and str(t).isdigit():
                insert = 'DO $$ BEGIN IF EXISTS (select foldID from folders
where foldID = {}) and ' \
                     'EXISTS (select povID from notifications where povID =
{}) and ' \
                         'not exists (select ppID from folders_notifications
where ppID = {}) THEN ' \
                     'INSERT INTO folders_notifications(ppID,
notifications_ID, folders_ID) VALUES ({},{},{}); ' \
                     'RAISE NOTICE {};' \
                     ' ELSE RAISE NOTICE {};' \
                     'END IF; ' \
                     'END $$;'.format(t,s, f, f,s,t, added, notice)
                check = False
            else:
                print('The values are wrong')

        elif table == 6:
            f = input('pkID = ')
            s = input('notifications_ID = ')
            t = input('users_ID = ')
            added = 'added'
            added = "'" + added + "'"
            notice = 'userID = {} or povID = {} is not present in tables. Or
both.'.format(t, s)

            notice = "'" + notice + "'"

            #insert = 'INSERT INTO notifications_users(pkID,
notifications_ID, users_ID) VALUES ({},{},{})'.format(
             #   f, s, t)

            if str(f).isdigit() and str(s).isdigit() and str(t).isdigit():
                insert = 'DO $$ BEGIN ' \
                     'IF EXISTS (select userID from users where userID = {})
and EXISTS (select povID from notifications where povID = {})' \
                         'and not exists (select pkID from
notifications_users where pkID = {}) THEN ' \
                     'INSERT INTO notifications_users(pkID, notifications_ID,
users_ID) VALUES ({},{},{}); ' \
                    'RAISE NOTICE {};' \
                     ' ELSE RAISE NOTICE {};' \
                     'END IF; ' \
                     'END $$;'.format(t,s, f, f,s,t, added, notice)
                check = False
            else:
                print('The values are wrong')
                check = False


        else:
            print('Try again.')
    print(Tables[table])
    print('SQL query => ', insert)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)
```

Update

У нашому випадку редагування ключів є неможливим

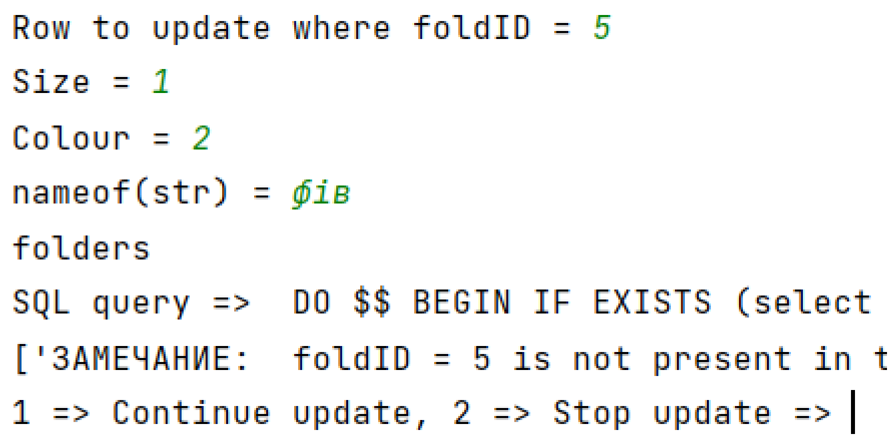


```
foldID        Size       Colour       nameof       id_mainfolder
30             13            3             QR             1
31             69            199             KG             9
1             2             3             енг             5
***************************

Row to update where foldID = 1
Size = 5
Colour = 7
nameof(str) = puppy
folders
SQL query =>  DO $$ BEGIN IF EXISTS (select foldID from folder
['ЗАМЕЧАНИЕ:  updated\n']
1 => Continue update, 2 => Stop update =>
```



```
foldID        Size       Colour       nameof       id_mainfolder
30             13            3             QR             1
31             69            199             KG             9
1             5             7             puppy             5
***************************
```

При спробі редагувати рядок, якого не існує:



```
Row to update where foldID = 5
Size = 1
Colour = 2
nameof(str) = фів
folders
SQL query =>  DO $$ BEGIN IF EXISTS (select
['ЗАМЕЧАНИЕ:  foldID = 5 is not present in t
1 => Continue update, 2 => Stop update =>
```

Лістинг для Update:

```
def updatebyuserallrow():
    connect = connection.connection()
```

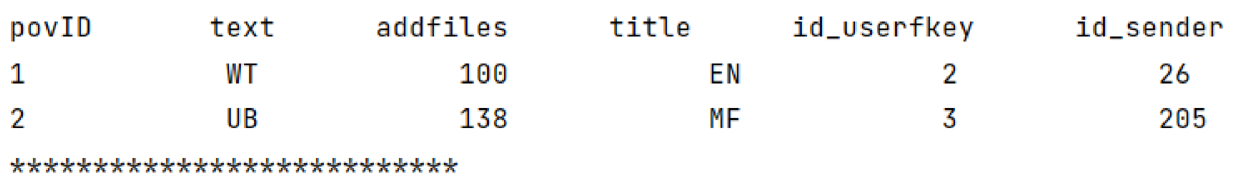
```
        cursor = connect.cursor()
        check = True
        updated = 'updated'
        updated = "'" + updated + "'"
        while check:
            View.listof()
            table = Model.existingtable()
            table = int(table)
            View.listofdatatoupdate(table)
            if table == 1:
                idk = input('Row to update where gmailID = ')
                notice = 'gmailID = {} is not present in table.'.format(idk)
                notice = "'" + notice + "'"
                set2 = input('creator(str) = ')

                set3 = input('name(str) = ')

                if set2.isalnum() and set3.isalnum() and str(idk).isdigit():
                    set2 = "'" + set2 + "'"
                    set3 = "'" + set3 + "'"

                    update = 'DO $$ BEGIN IF EXISTS (select gmailID from email
    where gmailID = {}) THEN ' \
                             'update email set creator = {}, name = {} where
    gmailID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format(idk, set2, set3, idk, updated,
    notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')
            elif table == 2:
                idk = input('Row to update where foldID = ')
                notice = 'foldID = {} is not present in table.'.format( idk)
                notice = "'" + notice + "'"
                set1 = input('Size = ')
                set2 = input('Colour = ')
                set3 = input('nameof(str) = ')

                if str(idk).isdigit() and str(set1).isdigit() and
    str(set2).isdigit() and set3.isalnum():
                    set3 = "'" + set3 + "'"
                    update = 'DO $$ BEGIN IF EXISTS (select foldID from folders
    where foldID = {})' \
                             ' THEN ' \
                             'update folders set Size = {}, Colour = {}, nameof =
    {} where foldID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format( idk, set1, set2, set3, idk,
    updated, notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')

            elif table == 3:
                idk = input('Row to update where userID = ')
                notice = 'userID = {} is not present in table.'.format( idk)
                notice = "'" + notice + "'"
                adress = input('adress(str) = ')

                place = input('place(str) = ')

                if str(idk).isdigit() and adress.isalnum() and place.isalnum():
                    adress = "'" + adress + "'"

                    place = "'" + place + "'"

                    update = 'DO $$ BEGIN IF EXISTS (select userID from users 
    where userID = {}) ' \
                             ' THEN ' \
                             'update users set adress = {}, place = {} where 
    userID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format(idk, adress, place, idk, updated, 
    notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')
            elif table == 4:
                idk = input('Row to update where povID = ')
                notice = 'povID = {} is not present in table.'.format(idk, idk)
                notice = "'" + notice + "'"
                text = input('text(str) = ')
                addfiles = input('addfiles = ')
                title = input('title(str) = ')
                sender = input('id_sender = ')

                if str(idk).isdigit() and text.isalnum() and 
    str(addfiles).isdigit() and title.isalnum() and str(sender).isdigit():
                    title = "'" + title + "'"
                    text = "'" + text + "'"

                    update = 'DO $$ BEGIN IF EXISTS (select povID from 
    notifications where povID = {}) ' \
                             ' THEN ' \
                             'update notifications set text = {}, addfiles = {}, 
    title = {}, id_sender = {} where povID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format( idk, text, addfiles, title, 
    sender, idk, updated, notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')
            else:
                print('Try again')
        print(Tables[table])
        print("SQL query => ", update)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)
        pass
```

Delete

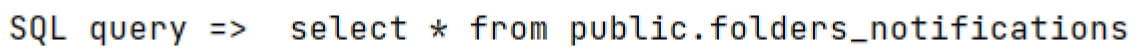
На прикладі таблиці Notifications та її дочірніх таблиць notifications_users та folders_notifications

Таблиці до видалення інфомарції:

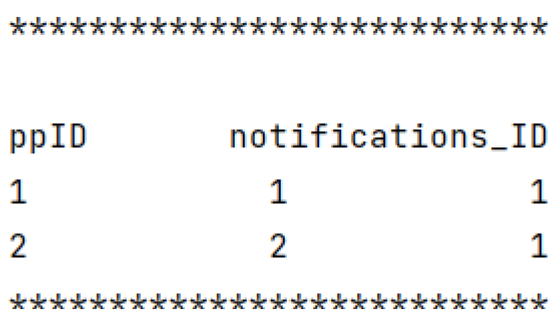
Notifications:



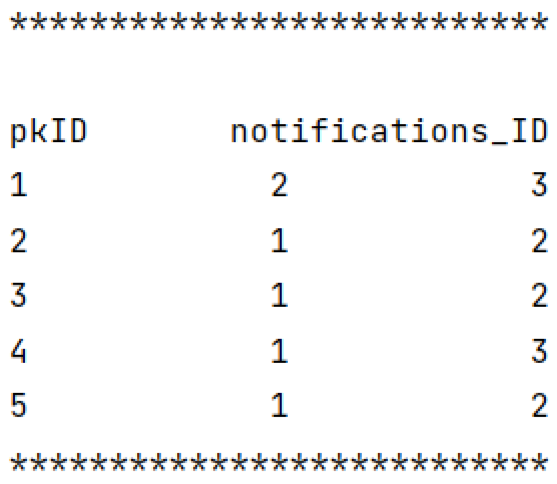
```
povID        text      addfiles       title      id_userfkey        id_sender
1             WT           100           EN              2              26
2             UB           138           MF              3              205
***************************
```



```
SQL query =>  select * from public.folders_notifications
```

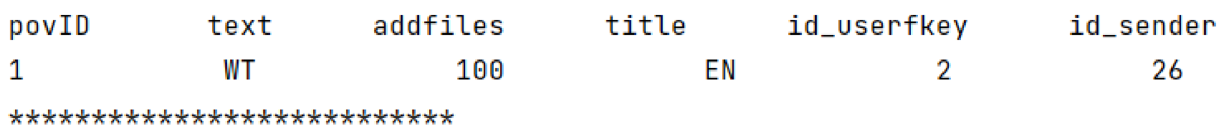


```
***************************

ppID       notifications_ID    folders_ID
1            1            1
2            2            1
***************************
```

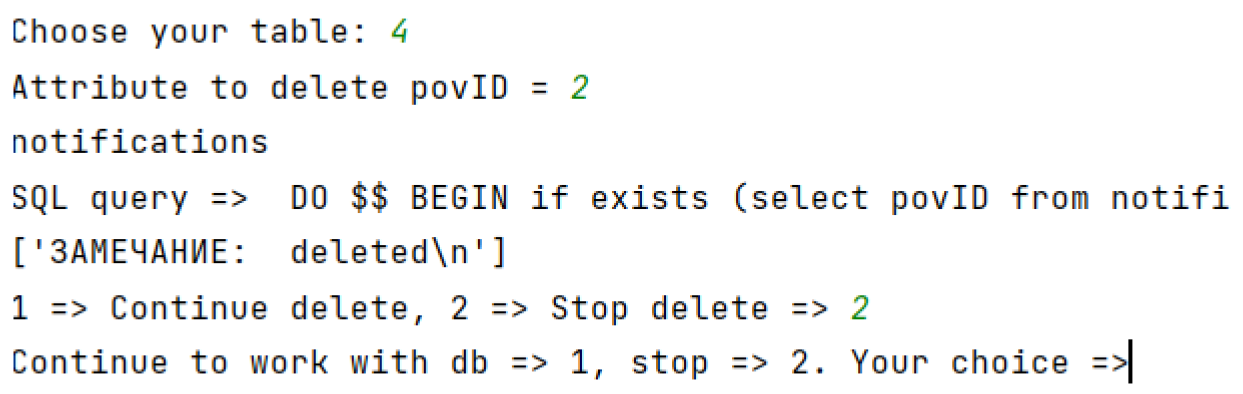


```
***************************

pkID       notifications_ID    users_ID
1            2            3
2            1            2
3            1            2
4            1            3
5            1            2
***************************
```

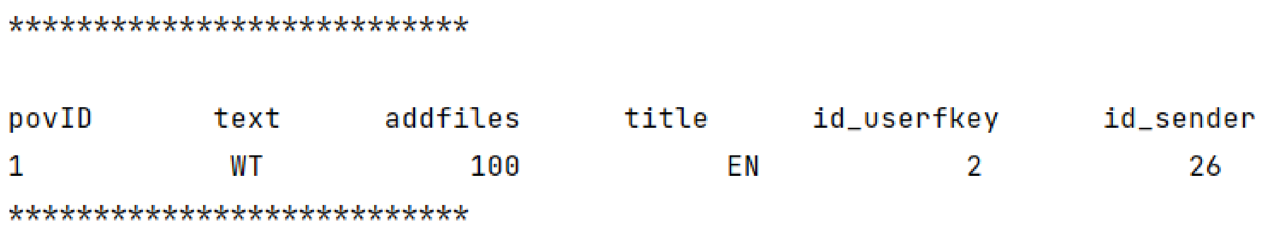
Видалення:



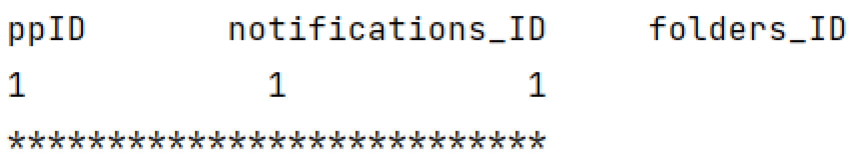
```
povID        text      addfiles       title      id_userfkey        id_sender
1             WT           100           EN              2              26
***************************
```



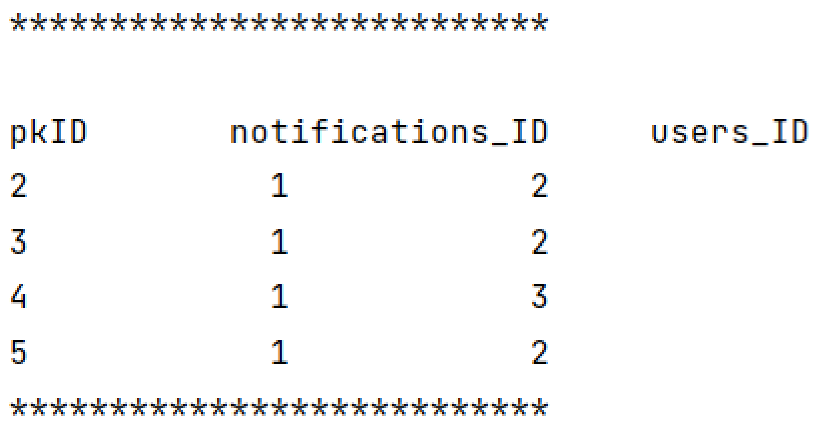
```
Choose your table: 4
Attribute to delete povID = 2
notifications
SQL query =>  DO $$ BEGIN if exists (select povID from notifi
['ЗАМЕЧАНИЕ:  deleted\n']
1 => Continue delete, 2 => Stop delete => 2
Continue to work with db => 1, stop => 2. Your choice =>
```



```
***************************

povID       text      addfiles      title      id_userfkey      id_sender
1            WT           100           EN             2             26
***************************
```

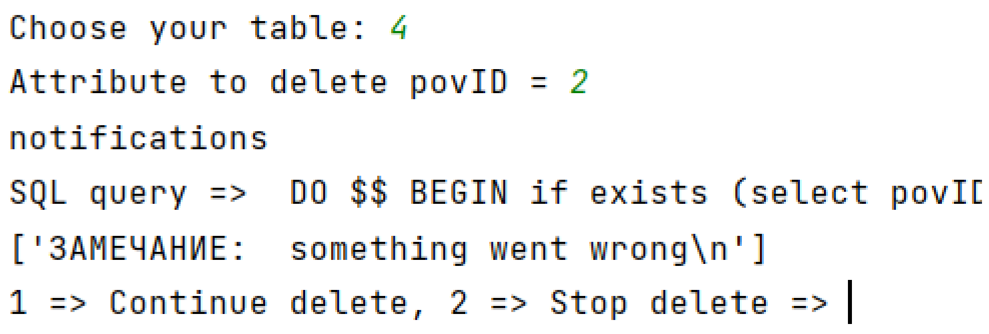


```
ppID       notifications_ID     folders_ID
1             1             1
***************************
```



```
***************************

pkID       notifications_ID     users_ID
2             1             2
3             1             2
4             1             3
5             1             2
***************************
```

При спробі видалення неіснуючого рядка:



```
Choose your table: 4
Attribute to delete povID = 2
notifications
SQL query =>  DO $$ BEGIN if exists (select povI
['ЗАМЕЧАНИЕ:  something went wrong\n']
1 => Continue delete, 2 => Stop delete =>
```

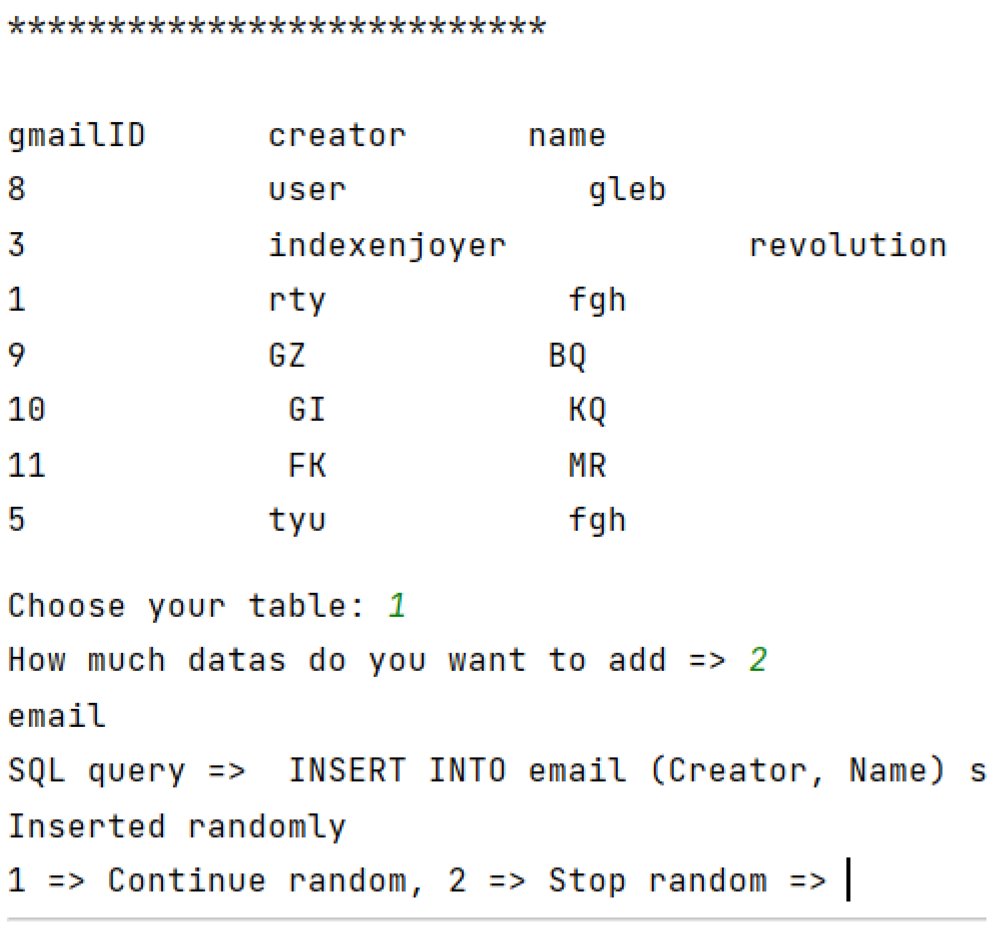
Лістинг Delete:

```
@staticmethod
def deletebyuser():
    connect = connection.connection()
    cursor = connect.cursor()
    check = True
    delete = 'deleted'
    delete = "'" + delete + "'"
    notice = 'something went wrong'
    notice = "'" + notice + "'"
    while check:
        View.listof()
        table = Model.existingtable()

        if table == 1:
            idk = input('Attribute to delete gmailID = ')
            idk = int(idk)

            #    'delete from notifications_users where notifications_ID =
(select povID from notifications where id_userfkey = (select userID from
users where id_mail = {}));' \
            #  'delete from folders_notifications where notifications_ID =
(select povID from notifications where id_userfkey = (select userID from
users where id_mail = {}));' \

            delete = 'DO $$ BEGIN IF EXISTS (select gmailID from email where
gmailID = {}) then ' \
                     'delete from folders_notifications where folders_ID in
(select foldID from folders where id_mainfolder = {});' \
                     'delete from folders_notifications where
notifications_ID in (select povID from notifications where id_userfkey in
(select userID from users where id_mail = {}));' \
                     'delete from notifications_users where notifications_ID
in (select povID from notifications where id_userfkey in (select userID from
users where id_mail = {}));' \
                     'delete from notifications_users where users_ID in
(select userID from users where id_mail = {});' \
                     'delete from notifications where id_userfkey in (select
userID from users where id_mail = {});' \
                     'delete from users where id_mail = {};' \
                     'delete from folders where id_mainfolder = {};' \
                     'delete from email where gmailID= {};' \
                     'raise notice {};' \
                     'else raise notice {};' \
                     'end if;' \
                     'end $$;'.format(idk, idk, idk, idk, idk, idk, idk, idk,
idk, delete, notice)


            check = False
        elif table == 2:
            idk = input('Attribute to delete foldID = ')
            ddelete = 'DO $$ BEGIN if ' \
                      'exists (select foldID from folders where foldID =
{}) then ' \
                      ' delete from folders_notifications where folders_ID
in (select foldID from folders where id_mainfolder = {});' \
                      'delete from folders where foldID= {};' \
                      'raise notice {};' \
                      'else raise notice {};' \
                      'end if;' \
                      'end $$;'.format(idk, idk, idk, delete, notice)
            check = False
        elif table == 3:
            idk = input('Attribute to delete userID = ')
            delete = 'DO $$ BEGIN if ' \
                     'exists (select userID from users where userID = {})
then ' \
                     'delete from notifications_users where notifications_ID
in (select povID from notifications ' \
                     'where id_userfkey in (select userID from users where
userID = {}));' \
                     'delete from notifications_users where users_ID in
(select povID from notifications ' \
                     'where id_userfkey = {});' \
                     'delete from notifications where id_userfkey = {};' \
                     'delete from users where userID = {};' \
                     'raise notice {};' \
                     'else raise notice {};' \
                     'end if;' \
                     'end $$;'.format(idk, idk, idk, idk, idk, delete,
notice)
            check = False
        elif table == 4:
            idk = input('Attribute to delete povID = ')
            delete = 'DO $$ BEGIN if exists (select povID from notifications
where povID = {}) then ' \
                     'delete from notifications_users where notifications_ID
= {};' \
                     'delete from folders_notifications where
notifications_ID = {};' \
                     'delete from notifications where povID= {};' \
                     'raise notice {};' \
                     'else raise notice {};' \
                     'end if;' \
                     'end $$;'.format(idk, idk, idk, idk, delete, notice)
            check = False
        elif table == 5:
            idk = input('Attribute to delete ppID = ')
            delete = 'DO $$ begin if exists (select ppID from
folders_notifications where ppID = {}) then ' \
                     ' delete from folders_notifications where ppID= {};' \
                     'raise notice {};' \
                     'else raise notice {};' \
                     'end if;' \
                     'end $$;'.format(idk, idk, delete, notice)
            check = False
        elif table == 6:
            idk = input('Attribute to delete pkID = ')
            delete = 'do $$ begin if exists (select pkID from
notifications_users where pkID = {}) then ' \
                     'delete from notifications_users where pkID= {};' \
                     'raise notice {};' \
                     'else raise notice {};' \
                     'end if;' \
                     'end $$;'.format(idk, idk, delete, notice)
            check = False
        else:
            print('Try again.')

    print(Tables[table])
    print("SQL query => ", delete)
    cursor.execute(delete)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)
```

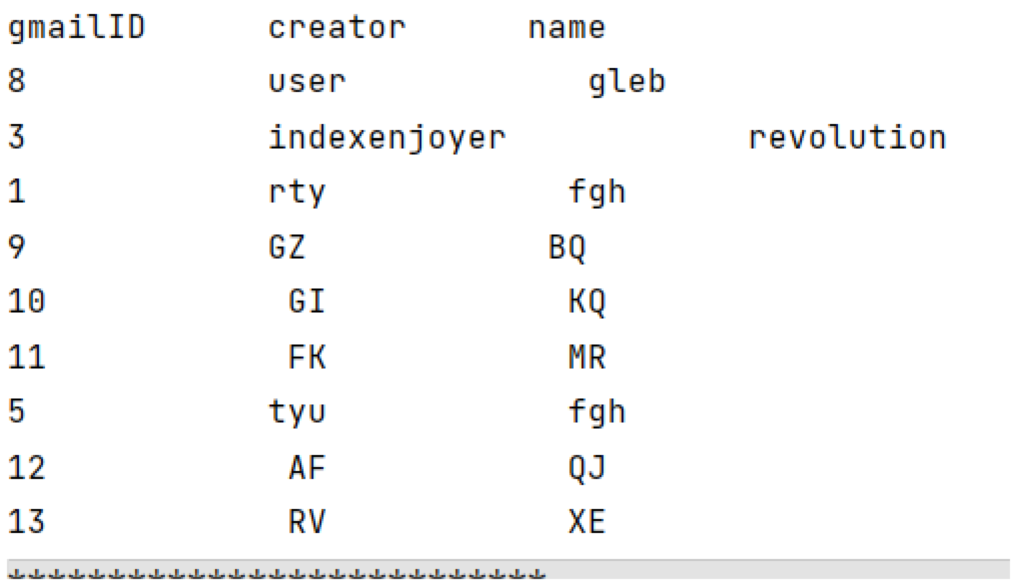
Завдання №2

Передбачити автоматичне пакетне генерування “рандомізованих” даних:

На прикладі таблиці Email:



```
****************************

gmailID      creator      name
8            user            gleb
3            indexenjoyer            revolution
1            rty            fgh
9            GZ            BQ
10            GI            KQ
11            FK            MR
5            tyu            fgh

Choose your table: 1
How much datas do you want to add => 2
email
SQL query =>  INSERT INTO email (Creator, Name) s
Inserted randomly
1 => Continue random, 2 => Stop random => 
```



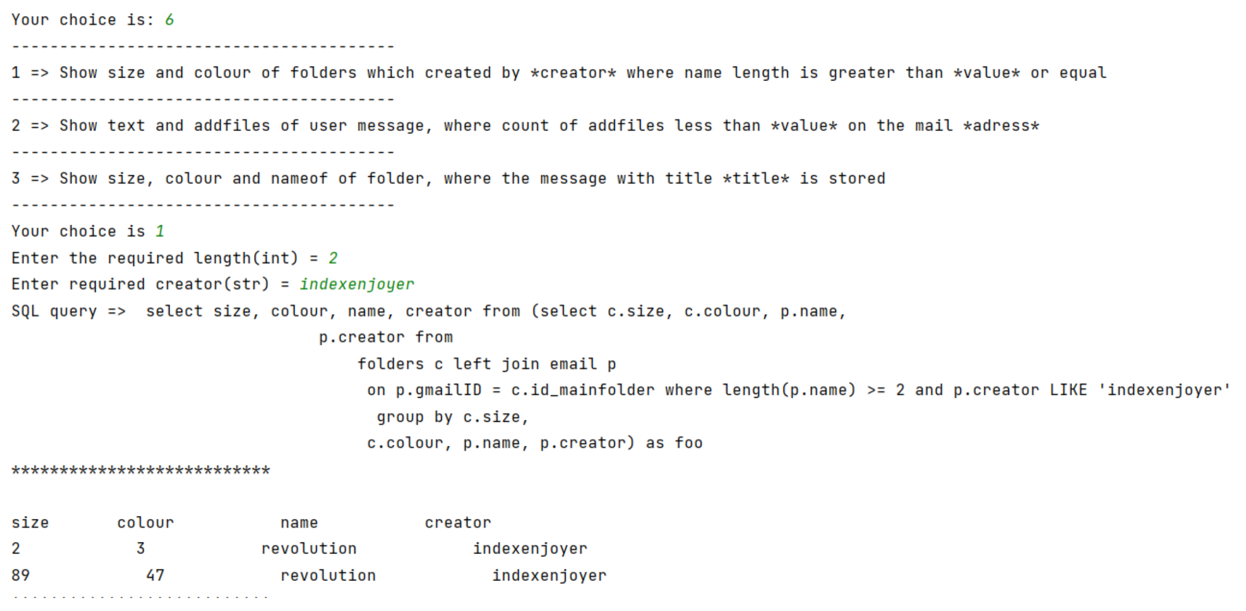
```
gmailID      creator      name
8            user            gleb
3            indexenjoyer            revolution
1            rty            fgh
9            GZ            BQ
10            GI            KQ
11            FK            MR
5            tyu            fgh
12            AF            QJ
13            RV            XE
```

Лістинг:

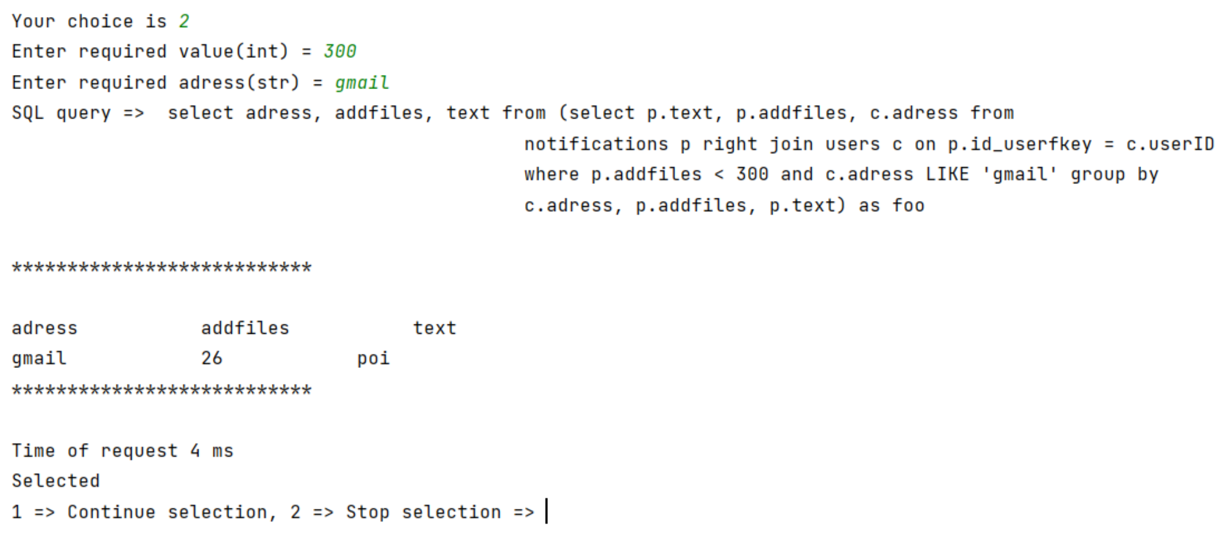
```
@staticmethod
def randomik():
        connect = connection.connection()
        cursor = connect.cursor()
        check = True
        while check:
            View.listof()
            table = Model.existingtable()
            kolvo = input('How much datas do you want to add => ')
            kolvo = int(kolvo)

            if table == 1:
                res = 0
                insert = "INSERT INTO email (Creator, Name) select 
chr(trunc(65 + random()*26)::int)||chr(trunc(65 + r" \
                         "andom()*26)::int), " \
                         "chr(trunc(65 + random()*26)::int)||chr(trunc(65 +
random()*26)::int) " \
                         "from generate_series(1,{})".format(kolvo)
                cursor.execute(insert)
                check = False
            elif table == 2:
                res = 0
                while (True):
                    insert = "INSERT INTO folders(Size, Colour, Nameof,
id_mainfolder) select random() * 256," \
                         "random() * 256," \
                         "chr(trunc(65 + random()*26)::int)||chr(trunc(65 +
random()*26)::int)," \
                         "(select gmailID from email order by random() limit
1)"\
                         "from generate_series(1,1)"
                    cursor.execute(insert)
                    res = res + 1
                    if(res == kolvo):
                        break
                check = False
            elif table == 3:
                res = 0
                while (res != kolvo):
                    insert = "INSERT INTO users (adress, place, id_mail)
select " \
                             "chr(trunc(65 + random()*25)::int)||chr(trunc(65
+ " \
                             "random()*25)::int), " \
                             "chr(trunc(65 + random()*25)::int)||chr(trunc(65
+ random()*25)::int)," \
                             "(select gmailID from email order by random()
limit 1) " \
                             "from generate_series(1,1)"
                    cursor.execute(insert)
                    res = res + 1

                check = False
            elif table == 4:
                res = 0
                while (res != kolvo):
                    insert = "INSERT INTO notifications (text, addfiles,
title, id_userfkey, id_sender) select " \
                         "chr(trunc(65 + random()*26)::int)||chr(trunc(65 +
r" \
                         "andom()*26)::int), random() * 256," \
                         "chr(trunc(65 + random()*26)::int)||chr(trunc(65 +
random()*26)::int)," \
                         "(select userID from users order by random() limit
1)," \
                         "random() * 256 " \
                         "from generate_series(1,1)"
                    cursor.execute(insert)
                    res = res + 1
                check = False
            elif table == 5:
                res = 0
                while (res!=kolvo):

                     insert = "INSERT INTO folders_notifications 
(notifications_id, folders_id) select " \
                         "(select povID from notifications order by random()
limit 1)," \
                         "(select foldID from folders order by random() limit
1) " \
                         "from generate_series(1,1)"
                     cursor.execute(insert)
                     res = res + 1
                check = False
            elif table == 6:
                res = 0
                while (res!=kolvo):
                        insert = "INSERT INTO notifications_users 
(notifications_id, users_id) select " \
                         "(select povID from notifications order by random()
limit 1)," \
                         "(select userID from users order by random() limit
1) " \
                         "from generate_series(1,1)"
                        cursor.execute(insert)
                        res = res + 1
                check = False
            else:
                print("Try again")
        print(Tables[table])
        print("SQL query => ", insert)
        connect.commit()
        print('Inserted randomly')
        cursor.close()
        connection.connectionlost(connect)
```

Завдання №3

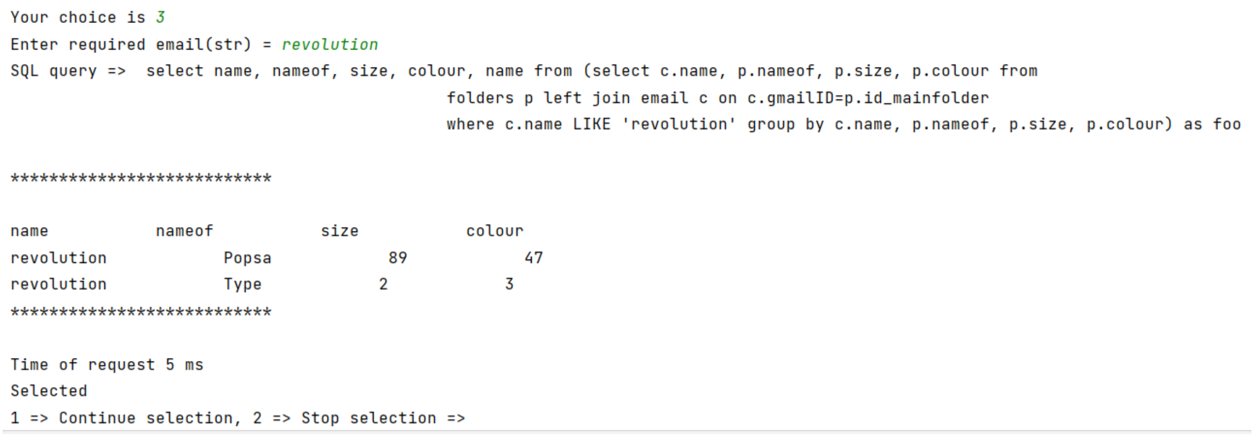
Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно.



```
Your choice is: 6
----------------------------------------
1 => Show size and colour of folders which created by *creator* where name length is greater than *value* or equal
----------------------------------------
2 => Show text and addfiles of user message, where count of addfiles less than *value* on the mail *adress*
----------------------------------------
3 => Show size, colour and nameof of folder, where the message with title *title* is stored
----------------------------------------
Your choice is 1
Enter the required length(int) = 2
Enter required creator(str) = indexenjoyer
SQL query =>  select size, colour, name, creator from (select c.size, c.colour, p.name,
                                p.creator from
                                    folders c left join email p
                                     on p.gmailID = c.id_mainfolder where length(p.name) >= 2 and p.creator LIKE 'indexenjoyer'
                                      group by c.size,
                                     c.colour, p.name, p.creator) as foo
***************************

size       colour          name           creator
2            3           revolution            indexenjoyer
89            47           revolution            indexenjoyer
...........................
```



```
Your choice is 2
Enter required value(int) = 300
Enter required adress(str) = gmail
SQL query =>  select adress, addfiles, text from (select p.text, p.addfiles, c.adress from
                                             notifications p right join users c on p.id_userfkey = c.userID
                                             where p.addfiles < 300 and c.adress LIKE 'gmail' group by
                                             c.adress, p.addfiles, p.text) as foo

***************************

adress           addfiles           text
gmail            26             poi
***************************

Time of request 4 ms
Selected
1 => Continue selection, 2 => Stop selection => |
```



```
Your choice is 3
Enter required email(str) = revolution
SQL query =>  select name, nameof, size, colour, name from (select c.name, p.nameof, p.size, p.colour from
                                             folders p left join email c on c.gmailID=p.id_mainfolder
                                             where c.name LIKE 'revolution' group by c.name, p.nameof, p.size, p.colour) as foo

***************************

name           nameof            size           colour
revolution            Popsa            89            47
revolution            Type           2            3
***************************

Time of request 5 ms
Selected
1 => Continue selection, 2 => Stop selection =>
```

Код програмного модулю “model.py”:

```
import random
import connection
from view import View
import time

Tables = {
    1: 'email',
    2: 'folders',
    3: 'users',
    4: 'notifications',
    5: 'folders_notifications',
    6: 'notifications_users'
}

class Model:
    @staticmethod
    def existingtable():
        while True:
            table = input('Choose your table: ')
            table = int(table)
            if table == 1 or table == 2 or table == 3 or table == 4 or table
== 5 or table == 6:
             return table
            else:
                print('Try again.')

    @staticmethod
    def outputonetable():
        View.listof()
        connect = connection.connection()
        cursor = connect.cursor()
        table = Model.existingtable()

        show = 'select * from public.{}'.format(Tables[table])

        print("SQL query => ", show)
        print('')
        cursor.execute(show)
        datas = cursor.fetchall()
        obj = View(table, datas)
        obj.output()
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def outputalltables():
        connect = connection.connection()
        cursor = connect.cursor()
        for table in range(1, 7):
            show = 'select * from public.{}'.format(Tables[table])

            print("SQL query => ", show)
            print('')
            cursor.execute(show)
            datas = cursor.fetchall()
            obj = View(table, datas)
            obj.output()
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def insertbyuser():
        connect = connection.connection()
        cursor = connect.cursor()
        check = True
        while check:
            View.listof()
            table = Model.existingtable()

            if table == 1:
                f = input('gmailID = ')
                s = input('creator(str) = ')

                t = input('name(str) = ')
                added = 'added'
                added = "'" + added + "'"
                notice = 'wrong way, the row with gmailID = {}
exists'.format(f)
                notice = "'" + notice + "'"
                if str(f).isdigit() and s.isalnum() and t.isalnum():
                    s = "'" + s + "'"
                    t = "'" + t + "'"
                    insert = 'DO $$ BEGIN if (1=1) and not exists (select
gmailID from email where gmailID = {}) then INSERT INTO email(gmailID,
creator, name) VALUES ({},{},{}); ' \
                             'raise notice {}; else raise notice {}; ' \
                             'end if; end $$;'.format(f, f, s, t, added, notice)
                    check = False
                else:
                    print('The values are wrong')


            elif table == 2:
                f = input('foldID = ')
                s = input('Size = ')
                t = input('Colour = ')
                fouth = input('nameof(str) = ')
                fifth = input('id_mainfolder = ')
                notice = 'gmailID = {} is not present in table or row with
foldID = {} exists already'.format(fifth,f)
                notice = "'" + notice + "'"
                added = 'added'
                added = "'" + added + "'"

               # insert = 'INSERT INTO folders(foldID, Size, Colour, nameof,
id_mainfolder) VALUES ({},{},{},{},{})'.format(f, s, t,fouth,fifth)
                if str(f).isdigit() and str(s).isdigit() and str(t).isdigit()
and fouth.isalnum() and str(fifth).isdigit():
                    fouth = "'" + fouth + "'"
                    insert = 'DO $$    BEGIN IF EXISTS (select gmailID from
email where gmailID = {}) and not exists (select foldId from folders where
foldId = {}) THEN ' \
                             'INSERT INTO folders(foldID, Size, Colour, nameof,
id_mainfolder) values ({}, {}, {}, {}, {}); ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                                'END $$;'.format(fifth,f, f, s, t, fouth, fifth,
added, notice)
                    check = False
                else:
                    print('The values are wrong')

            elif table == 3:
                f = input('userID = ')
                s = input('adress(str) = ')
                t = input('place(str) = ')

                fouth = input('id_mail = ')
                notice = 'gmailID = {} is not present in table or userID = {}
exists already'.format(fouth, f)
                notice = "'" + notice + "'"
                added = 'added'
                added = "'" + added + "'"

                if str(f).isdigit() and s.isalnum() and t.isalnum() and
str(fouth).isdigit():
                    s = "'" + s + "'"
                    t = "'" + t + "'"
                    insert = 'DO $$ BEGIN IF EXISTS (select gmailID from
email where gmailID = {}) and not exists (select userID from users where
userID = {}) THEN ' \
                         'INSERT INTO users(userID, adress, place, id_mail)
values ({}, {}, {}, {}); ' \
                         'RAISE NOTICE {};' \
                         ' ELSE RAISE NOTICE {};' \
                         'END IF; ' \
                        'END $$;'.format(fouth,f, f, s, t, fouth,added,
notice)
                    check = False
                else:
                    print('The values are wrong')

            elif table == 4:
                f = input('povID = ')
                s = input('text(str) = ')
                t = input('addfiles = ')
                fouth = input('title(str) = ')
                fifth = input('id_userfkey = ')
                sixth = input('id_sender = ')
                notice = 'userID = {} is not present in table or povID = {}
exists already'.format(fifth, f)
                notice = "'" + notice + "'"
                added = 'added'
                added = "'" + added + "'"

              #  insert = 'INSERT INTO notifications(povID, text, addfiles,
title, id_userfkey, id_sender) VALUES ({},{},{},{}, {}, {})'.format(
               #      f, s, t, fouth, fifth, sixth)
                if str(f).isdigit() and s.isalnum() and str(t).isdigit() and
fouth.isalnum() and str(fifth).isdigit() and str(sixth).isdigit():
                    s = "'" + s + "'"
                    fouth = "'" + fouth + "'"

                    insert = 'DO $$    BEGIN IF EXISTS (select userID from
users where userID = {}) and not exists (select povID from notifications
where povID = {}) THEN ' \
                         'INSERT INTO notifications(povID, text, addfiles,
title, id_userfkey, id_sender) values ({}, {}, {}, {}, {}, {}); ' \
                         'RAISE NOTICE {};' \
                         ' ELSE RAISE NOTICE {};' \
                         'END IF; ' \
                        'END $$;'.format(fifth, f, f, s, t, fouth, fifth,
sixth, added, notice)
                    check = False
                else:
                    print('The values are wrong')


            elif table == 5:
                f = input('ppID = ')
                s = input('notifications_ID = ')
                t = input('folders_ID = ')
                notice = 'foldID = {} or povID = {} or both is not present in
tables. Or ppId = {} exists already'.format(t, s, f)
                notice = "'" + notice + "'"
                added = 'added'
                added = "'" + added + "'"

               # insert = 'INSERT INTO folders_notifications(ppID,
notifications_ID, folders_ID) VALUES ({},{},{})'.format(
                #    f, s, t)
                if str(f).isdigit() and str(s).isdigit() and
str(t).isdigit():
                    insert = 'DO $$ BEGIN IF EXISTS (select foldID from
folders where foldID = {}) and ' \
                         'EXISTS (select povID from notifications where povID
= {}) and ' \
                             'not exists (select ppID from
folders_notifications where ppID = {}) THEN ' \
                         'INSERT INTO folders_notifications(ppID,
notifications_ID, folders_ID) VALUES ({},{},{}); ' \
                         'RAISE NOTICE {};' \
                         ' ELSE RAISE NOTICE {};' \
                         'END IF; ' \
                         'END $$;'.format(t,s, f, f,s,t, added, notice)
                    check = False
                else:
                    print('The values are wrong')

            elif table == 6:
                f = input('pkID = ')
                s = input('notifications_ID = ')
                t = input('users_ID = ')
                added = 'added'
                added = "'" + added + "'"
                notice = 'userID = {} or povID = {} is not present in tables.
Or both.'.format(t, s)

                notice = "'" + notice + "'"

                #insert = 'INSERT INTO notifications_users(pkID,
notifications_ID, users_ID) VALUES ({},{},{})'.format(
                 #   f, s, t)

                if str(f).isdigit() and str(s).isdigit() and
str(t).isdigit():
                    insert = 'DO $$ BEGIN ' \
                         'IF EXISTS (select userID from users where userID =
{}) and EXISTS (select povID from notifications where povID = {})' \
                             'and not exists (select pkID from
notifications_users where pkID = {}) THEN ' \
                         'INSERT INTO notifications_users(pkID,
notifications_ID, users_ID) VALUES ({},{},{}); ' \
                        'RAISE NOTICE {};' \
                         ' ELSE RAISE NOTICE {};' \
                         'END IF; ' \
                         'END $$;'.format(t,s, f, f,s,t, added, notice)
                    check = False
                else:
                    print('The values are wrong')
            else:
                print('Try again.')


        print(Tables[table])
        print('SQL query => ', insert)
        cursor.execute(insert)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def deletebyuser():
        connect = connection.connection()
        cursor = connect.cursor()
        check = True
        delete = 'deleted'
        delete = "'" + delete + "'"
        notice = 'something went wrong'
        notice = "'" + notice + "'"
        while check:
            View.listof()
            table = Model.existingtable()

            if table == 1:
                idk = input('Attribute to delete gmailID = ')
                idk = int(idk)

                #    'delete from notifications_users where notifications_ID
= (select povID from notifications where id_userfkey = (select userID from
users where id_mail = {}));' \
                #   'delete from folders_notifications where notifications_ID
= (select povID from notifications where id_userfkey = (select userID from
users where id_mail = {}));' \

                delete = 'DO $$ BEGIN IF EXISTS (select gmailID from email
where gmailID = {}) then ' \
                         'delete from folders_notifications where folders_ID
in (select foldID from folders where id_mainfolder = {});' \
                         'delete from folders_notifications where
notifications_ID in (select povID from notifications where id_userfkey in
(select userID from users where id_mail = {}));' \
                         'delete from notifications_users where
notifications_ID in (select povID from notifications where id_userfkey in
(select userID from users where id_mail = {}));' \
                         'delete from notifications_users where users_ID in
(select userID from users where id_mail = {});' \
                         'delete from notifications where id_userfkey in
(select userID from users where id_mail = {});' \
                         'delete from users where id_mail = {};' \
                         'delete from folders where id_mainfolder = {};' \
                         'delete from email where gmailID= {};' \
                         'raise notice {};' \
                         'else raise notice {};' \
                         'end if;' \
                         'end $$;'.format(idk, idk, idk, idk, idk, idk, idk,
idk, idk, delete, notice)


                check = False
            elif table == 2:
                idk = input('Attribute to delete foldID = ')
                ddelete = 'DO $$ BEGIN if ' \
                              'exists (select foldID from folders where foldID
= {}) then ' \
                              ' delete from folders_notifications where
folders_ID in (select foldID from folders where id_mainfolder = {});' \
                              'delete from folders where foldID= {};' \
                              'raise notice {};' \
                              'else raise notice {};' \
                              'end if;' \
                              'end $$;'.format(idk, idk, idk, delete, notice)
                check = False
            elif table == 3:
                idk = input('Attribute to delete userID = ')
                delete = 'DO $$ BEGIN if ' \
                         'exists (select userID from users where userID = {})
then ' \
                         'delete from notifications_users where
notifications_ID in (select povID from notifications ' \
                         'where id_userfkey in (select userID from users
where userID = {}));' \
                         'delete from notifications_users where users_ID in
(select povID from notifications ' \
                         'where id_userfkey = {});' \
                         'delete from notifications where id_userfkey = {};'
\
                         'delete from users where userID = {};' \
                         'raise notice {};' \
                         'else raise notice {};' \
                         'end if;' \
                         'end $$;'.format(idk, idk, idk, idk, idk, delete,
notice)
                check = False
            elif table == 4:
                idk = input('Attribute to delete povID = ')
                delete = 'DO $$ BEGIN if exists (select povID from
notifications where povID = {}) then ' \
                         'delete from notifications_users where
notifications_ID = {};' \
                         'delete from folders_notifications where
notifications_ID = {};' \
                         'delete from notifications where povID= {};' \
                         'raise notice {};' \
                         'else raise notice {};' \
                         'end if;' \
                         'end $$;'.format(idk, idk, idk, idk, delete, notice)
                check = False
            elif table == 5:
                idk = input('Attribute to delete ppID = ')
                delete = 'DO $$ begin if exists (select ppID from
folders_notifications where ppID = {}) then ' \
                         ' delete from folders_notifications where ppID= {};'
\
                         'raise notice {};' \
                         'else raise notice {};' \
                         'end if;' \
                         'end $$;'.format(idk, idk, delete, notice)
                check = False
            elif table == 6:
                idk = input('Attribute to delete pkID = ')
                delete = 'do $$ begin if exists (select pkID from
notifications_users where pkID = {}) then ' \
                         'delete from notifications_users where pkID= {};' \
                         'raise notice {};' \
                         'else raise notice {};' \
                         'end if;' \
                         'end $$;'.format(idk, idk, delete, notice)
                check = False
            else:
                print('Try again.')

        print(Tables[table])
        print("SQL query => ", delete)
        cursor.execute(delete)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def deletealot():
        connect = connection.connection()
        cursor = connect.cursor()
        delete = 'deleted'
        delete = "'" + delete + "'"
        notice = 'Something went wrong'
        notice = "'" + notice + "'"
        check = True
        while check:
            View.listof()
            table = Model.existingtable()
            start = input('From which number do you want to start? =>')
            stop = input('Number to stop? =>')
            start = int(start)
            stop = int(stop)
            if stop <= start:
                return 'Try again'
            if table == 1:
                while start != stop:
                    #                              'delete from
folders_notifications where notifications_ID = (select povID from
notifications where id_userfkey = (select userID from users where id_mail =
{}));' \
                    #                             'delete from
notifications_users where notifications_ID = (select povID from notifications
where id_userfkey = (select userID from users where id_mail = {}));' \
                    #  'exists (select id_mainfolder from folders where
id_mainfolder = {}) and ' \
                    #  'exists (select id_mail from users where id_mail = {})
and ' \
                    #  'exists (select id_userfkey from notifications where
id_userfkey in (select userID from users where id_mail = {})) and ' \
                    #  'exists (select users_ID from notifications_users
where users_ID in (select userID from users where id_mail = {})) and' \
                    #  'exists (select notifications_ID from
notifications_users where notifications_ID in (select povID from
notifications where id_userfkey in (select userID from users where id_mail =
{}))) and ' \
                    #  'exists (select notifications_ID from
folders_notifications where notifications_ID in (select povID from
notifications where id_userfkey in (select userID from users where id_mail =
{}))) and ' \
                    #   'exists (select folders_ID from folders_notifications
where folders_ID in (select foldID from folders where id_mainfolder = {}))
THEN ' \

                    delete = 'DO $$ BEGIN IF EXISTS (select gmailID from
email where gmailID = {}) then ' \
                             'delete from folders_notifications where
folders_ID in (select foldID from folders where id_mainfolder = {});' \
                             'delete from folders_notifications where
notifications_ID in (select povID from notifications where id_userfkey in
(select userID from users where id_mail = {}));' \
                             'delete from notifications_users where
notifications_ID in (select povID from notifications where id_userfkey in
(select userID from users where id_mail = {}));' \
                             'delete from notifications_users where users_ID
in (select userID from users where id_mail = {});' \
                             'delete from notifications where id_userfkey in
(select userID from users where id_mail = {});' \
                             'delete from users where id_mail = {};' \
                             'delete from folders where id_mainfolder = {};'
\
                             'delete from email where gmailID= {};' \
                             'raise notice {};' \
                             'else raise notice {};' \
                             'end if;' \
                             'end $$;'.format(
                        start, start, start, start, start, start, start,
start, start, delete, notice)
                    check = False
                    cursor.execute(delete)
                    start = start + 1
            elif table == 2:
                while start != stop:
                    delete = 'DO $$ BEGIN if ' \
                             'exists (select foldID from folders where foldID
= {}) then ' \
                             ' delete from folders_notifications where
folders_ID in (select foldID from folders where id_mainfolder = {});' \
                             'delete from folders where foldID= {};' \
                             'raise notice {};' \
                             'else raise notice {};' \
                             'end if;' \
                             'end $$;'.format(start, start, start, delete,
notice)
                    check = False
                    cursor.execute(delete)
                    start = start + 1
            elif table == 3:
                while start != stop:
                    delete = 'DO $$ BEGIN if ' \
                             'exists (select userID from users where userID =
{}) then ' \
                             'delete from notifications_users where
notifications_ID in (select povID from notifications ' \
                             'where id_userfkey in (select userID from users
where userID = {}));' \
                             'delete from notifications_users where users_ID
in (select povID from notifications ' \
                             'where id_userfkey = {});' \
                             'delete from notifications where id_userfkey =
{};' \
                             'delete from users where userID = {};' \
                             'raise notice {};' \
                             'else raise notice {};' \
                             'end if;' \
                             'end $$;'.format(start, start, start, start,
start, delete, notice)
                    check = False
                    cursor.execute(delete)
                    start = start + 1
            elif table == 4:
                while start != stop:
                    delete = 'DO $$ BEGIN if exists (select povID from
notifications where povID = {}) then ' \
                             'delete from notifications_users where
notifications_ID = {};' \
                             'delete from folders_notifications where
notifications_ID = {};' \
                             'delete from notifications where povID= {};' \
                             'raise notice {};' \
                             'else raise notice {};' \
                             'end if;' \
                             'end $$;'.format(start, start, start, start,
delete, notice)
                    check = False
                    cursor.execute(delete)
                    start = start + 1
            elif table == 5:
                while start != stop:
                    delete = 'DO $$ begin if exists (select ppID from
folders_notifications where ppID = {}) then ' \
                             ' delete from folders_notifications where ppID=
{};' \
                             'raise notice {};' \
                             'else raise notice {};' \
                             'end if;' \
                             'end $$;'.format(start, start, delete, notice)
                    check = False
                    cursor.execute(delete)
                    start = start + 1
            elif table == 6:
                while start != stop:
                    delete = 'do $$ begin if exists (select pkID from
notifications_users where pkID = {}) then ' \
                             'delete from notifications_users where pkID=
{};' \
                             'raise notice {};' \
                             'else raise notice {};' \
                             'end if;' \
                             'end $$;'.format(start, start, delete, notice)
                    check = False
                    cursor.execute(delete)
                    start = start + 1
            else:
                print('Try again.')
        print(Tables[table])
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def updatebyuser():
        connect = connection.connection()
        cursor = connect.cursor()
        check = True

        updated = 'updated'
        updated = "'" + updated + "'"
        while check:
            View.listof()
            table = Model.existingtable()
            if table == 1:
                idk = input('Attribute to update(where) gmailID = ')
                notice = 'gmailID = {} is not present in table.'.format(idk)
                notice = "'" + notice + "'"
                View.listofdatatoupdate(1)
                checkin = True
                while checkin:
                    atnum = input('Number of attribute => ')
                    if(int(atnum) < 2 or int(atnum) > 3):
                        print('Try again')
                        continue
                    nval = input('New value = ')
                   # if atnum == '1':
                    #    set = 'gmailID = {}'.format(nval)
                     #   checkin = False
                    if atnum == '2':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'creator = {}'.format(nval)

                            update = 'DO $$ BEGIN IF EXISTS (select gmailID
from email where gmailID = {})' \
                                     ' THEN ' \
                            'update email set {} where gmailID = {}; ' \
                            'RAISE NOTICE {};' \
                            ' ELSE RAISE NOTICE {};' \
                            'END IF; ' \
                            'END $$;'.format(idk, set, idk, updated, notice)

                            check = False
                            pass
                        else:
                            print('Column should contain only chars or
numbers')
                    elif atnum == '3':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'name = {}'.format(nval)
                            update = 'DO $$ BEGIN IF EXISTS (select gmailID
from email where gmailID = {}) THEN ' \
                                     'update email set {} where gmailID = {};
' \

                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format(idk, set, idk, updated,
notice)
                            check = False
                            pass
                        else:
                            print('Column should contain only chars or
numbers')
                    else:
                        print('Try again')
            elif table == 2:
                idk = input('Attribute to update(where) foldID = ')
                notice = 'foldID = {} is not present in table.'.format(idk)
                notice = "'" + notice + "'"
                View.listofdata(2)
                checkin = True
                while checkin:
                    atnum = input('Number of attribute => ')
                    if (int(atnum) < 2 or int(atnum) > 4):
                        print('Try again')
                        continue
                    nval = input('New value = ')
                   # if atnum == '1':
                    #     set = 'foldID = {}'.format(nval)
                    #     checkin = False
                    if atnum == '2':
                        set = 'Size = {}'.format(nval)
                        checkin = False
                        if (str(nval).isdigit() and str(idk).isdigit()):
                            update = 'DO $$ BEGIN IF EXISTS (select foldID
from folders where foldID = {})' \
                                     ' THEN ' \
                                     'update folders set {} where foldID =
{}; ' \
                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format(idk, set, idk, updated,
notice)
                            check = False

                        else:
                            print('Column should contain only numbers')
                    elif atnum == '3':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'colour = {}'.format(nval)
                            update = 'DO $$ BEGIN IF EXISTS (select foldID
from folders where foldID = {}) ' \
                                     ' THEN ' \
                                     'update folders set {} where foldID =
{}; ' \
                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format(idk, set, idk, updated,
notice)
                            check = False

                        else:
                            print('Column should contain only chars or
numbers')
                    elif atnum == '4':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'nameof = {}'.format(nval)
                            update = 'DO $$ BEGIN IF EXISTS (select foldID
from folders where foldID = {}) ' \
                                     ' THEN ' \
                                     'update folders set {} where foldID =
{}; ' \
                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format(idk, set, idk, updated,
notice)
                            check = False

                        else:
                            print('Column should contain only chars or
numbers')
                   # elif atnum == '5':
                    #     set = 'id_mainfolder = {}'.format(nval)
                     #   checkin = False
                    else:
                        print('Try again')
                pass
            elif table == 3:
                idk = input('Attribute to update(where) userID = ')
                notice = 'userID = {} is not present in table.'.format(idk)
                notice = "'" + notice + "'"
                View.listofdata(3)
                checkin = True
                while checkin:
                    atnum = input('Number of attribute => ')
                    if (int(atnum) < 2 or int(atnum) > 3):
                        print('Try again')
                        continue
                    nval = input('New value = ')
                    #if atnum == '1':
                     #   set = 'userID = {}'.format(nval)
                    #    checkin = False
                    if atnum == '2':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'adress = {}'.format(nval)

                            update = 'DO $$ BEGIN IF EXISTS (select userID
from users where userID = {}) ' \
                                     ' THEN ' \
                                     'update users set {} where userID = {};
' \

                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format(idk, set, idk, updated,
notice)
                            check = False
                        else:
                            print('Column should contain only chars or
numbers')
                    elif atnum == '3':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'place = {}'.format(nval)
                            update = 'DO $$ BEGIN IF EXISTS (select userID
from users where userID = {})' \
                                     '  THEN ' \
                                     'update users set {} where userID = {};
' \

                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format( idk, set, idk,
updated, notice)
                            check = False
                        else:
                            print('Column should contain only chars or
numbers')
                   # elif atnum == '4':
                    #    set = 'id_mail = {}'.format(nval)
                     #   checkin = False
                    else:
                        print('Try again')
                pass
            elif table == 4:
                idk = input('Attribute to update(where) povID = ')
                notice = 'povID = {} is not present in table.'.format( idk)
                notice = "'" + notice + "'"
                View.listofdata(4)
                checkin = True
                while checkin:
                    atnum = input('Number of attribute => ')
                    if (int(atnum) < 2 or int(atnum) > 6 or int(atnum) == 5):
                        print('Try again')
                        continue
                    nval = input('New value = ')
                    #if atnum == '1':
                     #    set = 'povID = {}'.format(nval)
                    #     checkin = False
                    if atnum == '2':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'text = {}'.format(nval)

                            update = 'DO $$ BEGIN IF EXISTS (select povID
from notifications where povID = {}) ' \
                                     ' THEN ' \
                                     'update notifications set {} where povID
= {}; ' \
                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format( idk, set, idk,
updated, notice)
                            check = False
                        else:
                            print('Column should contain only chars or
numbers')
                    elif atnum == '3':
                        set = 'addfiles = {}'.format(nval)
                        checkin = False
                        if (str(nval).isdigit() and str(idk).isdigit()):
                            update = 'DO $$ BEGIN IF EXISTS (select povID
from notifications where povID = {}) ' \
                                     ' THEN ' \
                                     'update notifications set {} where povID
= {}; ' \
                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format( idk, set, idk,
updated, notice)
                            check = False
                        else:
                            print('Column should contain only numbers')
                    elif atnum == '4':
                        checkin = False
                        if (nval.isalnum() and str(idk).isdigit()):
                            nval = "'" + nval + "'"
                            set = 'title = {}'.format(nval)
                            update = 'DO $$ BEGIN IF EXISTS (select povID
from notifications where povID = {}) ' \
                                     ' THEN ' \
                                     'update notifications set {} where povID
= {}; ' \
                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format( idk, set, idk,
updated, notice)
                            check = False
                        else:
                            print('Column should contain only chars or
numbers')
                    #elif atnum == '5':
                      #  set = 'id_userfkey = {}'.format(nval)
                     #   checkin = False
                    elif atnum == '6':

                        checkin = False
                        if (str(nval).isdigit() and str(idk).isdigit()):
                            set = 'id_sender = {}'.format(nval)
                            update = 'DO $$ BEGIN IF EXISTS (select povID
from notifications where povID = {})' \
                                     ' THEN ' \
                                     'update notifications set {} where povID
= {}; ' \

                                     'RAISE NOTICE {};' \
                                     ' ELSE RAISE NOTICE {};' \
                                     'END IF; ' \
                                     'END $$;'.format( idk, set, idk,
updated, notice)
                            check = False
                        else:
                            print('Column should contain only chars or
numbers')
                    else:
                        print('Try again')
                pass

          #  elif table == 5:
           #     idk = input('Attribute to update(where) ppID = ')
            #    View.listofdata(5)
             #   checkin = True
              #  while checkin:
               #     atnum = input('Number of attribute => ')
                #    nval = input('New value = ')
                 #   if atnum == '1':
                  #      set = 'ppID = {}'.format(nval)
                   #     checkin = False
                   # elif atnum == '2':
                    #    set = 'notifications_ID = {}'.format(nval)
                     #   checkin = False
                   # elif atnum == '3':
                    #    set = 'folders_ID = {}'.format(nval)
                     #   checkin = False

                   # else:
                   #     print('Try again')
                # update = 'update folders_notifications set {} where ppID =
{}'.format(set, idk)
                # check = False
                # pass
            # elif table == 6:
             #   idk = input('Attribute to update(where) pkID = ')
              #  View.listofdata(6)
              #  checkin = True
              #  while checkin:
              #      atnum = input('Number of attribute => ')
              #      nval = input('New value = ')
              #      if atnum == '1':
              #           set = 'pkID = {}'.format(nval)
              #           checkin = False
              #       elif atnum == '2':
               #          set = 'notifications_ID = {}'.format(nval)
               #          checkin = False
                #    elif atnum == '3':
                #        set = 'users_ID = {}'.format(nval)
                #        checkin = False

                #    else:
                 #       print('Try again')
                #update = 'update notifications_users set {} where pkID =
{}'.format(set, idk)
                #check = False
                #pass
            else:
                print('Try again.')
        print(Tables[table])
        print("SQL query => ", update)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)
        pass

    @staticmethod
    def updatebyuserallrow():
        connect = connection.connection()
        cursor = connect.cursor()
        check = True
        updated = 'updated'
        updated = "'" + updated + "'"
        while check:
            View.listof()
            table = Model.existingtable()
            table = int(table)
            View.listofdatatoupdate(table)
            if table == 1:
                idk = input('Row to update where gmailID = ')
                notice = 'gmailID = {} is not present in table.'.format(idk)
                notice = "'" + notice + "'"
                set2 = input('creator(str) = ')

                set3 = input('name(str) = ')

                if set2.isalnum() and set3.isalnum() and str(idk).isdigit():
                    set2 = "'" + set2 + "'"
                    set3 = "'" + set3 + "'"

                    update = 'DO $$ BEGIN IF EXISTS (select gmailID from
email where gmailID = {}) THEN ' \
                             'update email set creator = {}, name = {} where
gmailID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format(idk, set2, set3, idk, updated,
notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')
            elif table == 2:
                idk = input('Row to update where foldID = ')
                notice = 'foldID = {} is not present in table.'.format( idk)
                notice = "'" + notice + "'"
                set1 = input('Size = ')
                set2 = input('Colour = ')
                set3 = input('nameof(str) = ')

                if str(idk).isdigit() and str(set1).isdigit() and
str(set2).isdigit() and set3.isalnum():
                    set3 = "'" + set3 + "'"
                    update = 'DO $$ BEGIN IF EXISTS (select foldID from
folders where foldID = {})' \
                             ' THEN ' \
                             'update folders set Size = {}, Colour = {},
nameof = {} where foldID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format( idk, set1, set2, set3, idk,
updated, notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')

            elif table == 3:
                idk = input('Row to update where userID = ')
                notice = 'userID = {} is not present in table.'.format( idk)
                notice = "'" + notice + "'"
                adress = input('adress(str) = ')

                place = input('place(str) = ')

                if str(idk).isdigit() and adress.isalnum() and
place.isalnum():
                    adress = "'" + adress + "'"

                    place = "'" + place + "'"

                    update = 'DO $$ BEGIN IF EXISTS (select userID from users
where userID = {}) ' \
                             ' THEN ' \
                             'update users set adress = {}, place = {} where
userID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format(idk, adress, place, idk,
updated, notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')
            elif table == 4:
                idk = input('Row to update where povID = ')
                notice = 'povID = {} is not present in table.'.format(idk,
idk)
                notice = "'" + notice + "'"
                text = input('text(str) = ')
                addfiles = input('addfiles = ')
                title = input('title(str) = ')
                sender = input('id_sender = ')

                if str(idk).isdigit() and text.isalnum() and
```

```
str(addfiles).isdigit() and title.isalnum() and str(sender).isdigit():
                    title = "'" + title + "'"
                    text = "'" + text + "'"

                    update = 'DO $$ BEGIN IF EXISTS (select povID from
notifications where povID = {}) ' \
                             ' THEN ' \
                             'update notifications set text = {}, addfiles =
{}, title = {}, id_sender = {} where povID = {}; ' \
                             'RAISE NOTICE {};' \
                             ' ELSE RAISE NOTICE {};' \
                             'END IF; ' \
                             'END $$;'.format( idk, text, addfiles, title,
sender, idk, updated, notice)
                    check = False
                    pass
                else:
                    print('The values are wrong')
            else:
                print('Try again')
        print(Tables[table])
        print("SQL query => ", update)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)
        pass


    @staticmethod
    def selection():
        connect = connection.connection()
        cursor = connect.cursor()
        check = True
        while check:
            print('-----------------------------------------')
            print('1 => Show size and colour of folders which created by
*creator* '
                  'where name length is greater than *value* or equal')
            print('-----------------------------------------')
            print('2 => Show text and addfiles of user message, where count
of addfiles less than *value* on the mail *adress*')
            print('-----------------------------------------')
            print('3 => Show size, colour and nameof of folders, which are
stored on the *email*')
            print('-----------------------------------------')
            choice = input('Your choice is ')
            choice = int(choice)

            if choice == 1:
                len = input('Enter the required length(int) = ')
                creator = input('Enter required creator(str) = ')
                select = """select size, colour, name, creator from (select
c.size, c.colour, p.name,
                             p.creator from
                                 folders c left join email p
                                  on p.gmailID = c.id_mainfolder where
length(p.name) >= {} and p.creator LIKE '{}'
                                   group by c.size,
                                  c.colour, p.name, p.creator) as
foo""".format(len, creator)
                check = False
            elif choice == 2:
                value = input('Enter required value(int) = ')
                adress = input('Enter required adress(str) = ')
                select = """select adress, addfiles, text from (select
p.text, p.addfiles, c.adress from
                                              notifications p right join
users c on p.id_userfkey = c.userID
                                              where p.addfiles < {} and
c.adress LIKE '{}' group by
                                              c.adress, p.addfiles, p.text)
as foo
                """.format(value, adress)
                check = False
            elif choice == 3:
                title = input('Enter required email(str) = ')
                select = """select name, nameof, size, colour, name from
(select c.name, p.nameof, p.size, p.colour from
                                             folders p left join email c on
c.gmailID=p.id_mainfolder
                                             where c.name LIKE '{}' group by
c.name, p.nameof, p.size, p.colour) as foo
                """.format(title)
                check = False
            else:
                print('Try again')
        print("SQL query => ", select)
        beg = int(time.time() * 1000)
        cursor.execute(select)
        end = int(time.time() * 1000) - beg
        datas = cursor.fetchall()
        obj = View(choice, datas)
        obj.output_spec()
        print('Time of request {} ms'.format(end))
        print('Selected')
        cursor.close()
        connection.connectionlost(connect)



    @staticmethod
    def randomik():
            connect = connection.connection()
            cursor = connect.cursor()
            check = True
            while check:
                View.listof()
                table = Model.existingtable()
                kolvo = input('How much datas do you want to add => ')
                kolvo = int(kolvo)

                if table == 1:
                    res = 0
                    insert = "INSERT INTO email (Creator, Name) select
chr(trunc(65 + random()*26)::int)||chr(trunc(65 + r" \
                             "andom()*26)::int), " \
                             "chr(trunc(65 + random()*26)::int)||chr(trunc(65
+ random()*26)::int) " \
                             "from generate_series(1,{})".format(kolvo)
                    cursor.execute(insert)
                    check = False
                elif table == 2:
                    res = 0
                    while (True):
                        insert = "INSERT INTO folders(Size, Colour, Nameof,
id_mainfolder) select random() * 256," \
                             "random() * 256," \
                             "chr(trunc(65 + random()*26)::int)||chr(trunc(65
+ random()*26)::int)," \
                             "(select gmailID from email order by random()
limit 1)"\
                             "from generate_series(1,1)"
                        cursor.execute(insert)
                        res = res + 1
                        if(res == kolvo):
                            break
                    check = False
                elif table == 3:
                    res = 0
                    while (res != kolvo):
                        insert = "INSERT INTO users (adress, place, id_mail)
select " \
                                 "chr(trunc(65 +
random()*25)::int)||chr(trunc(65 + " \
                                 "random()*25)::int), " \
                                 "chr(trunc(65 +
random()*25)::int)||chr(trunc(65 + random()*25)::int)," \
                                 "(select gmailID from email order by
random() limit 1) " \
                                 "from generate_series(1,1)"
                        cursor.execute(insert)
                        res = res + 1

                    check = False
                elif table == 4:
                    res = 0
                    while (res != kolvo):
                        insert = "INSERT INTO notifications (text, addfiles,
title, id_userfkey, id_sender) select " \
                             "chr(trunc(65 + random()*26)::int)||chr(trunc(65
+ r" \
                             "andom()*26)::int), random() * 256," \
                             "chr(trunc(65 + random()*26)::int)||chr(trunc(65
+ random()*26)::int)," \
                             "(select userID from users order by random()
limit 1)," \
                             "random() * 256 " \
                             "from generate_series(1,1)"
                        cursor.execute(insert)
                        res = res + 1
                    check = False
                elif table == 5:
                    res = 0
                    while (res!=kolvo):

                         insert = "INSERT INTO folders_notifications
(notifications_id, folders_id) select " \
                             "(select povID from notifications order by
random() limit 1)," \
                             "(select foldID from folders order by random()
limit 1) " \
                             "from generate_series(1,1)"
                         cursor.execute(insert)
                         res = res + 1
                    check = False
                elif table == 6:
                    res = 0
                    while (res!=kolvo):
                            insert = "INSERT INTO notifications_users
(notifications_id, users_id) select " \
                             "(select povID from notifications order by
random() limit 1)," \
                             "(select userID from users order by random()
limit 1) " \
                             "from generate_series(1,1)"
                            cursor.execute(insert)
                            res = res + 1
                    check = False
                else:
                    print("Try again")
            print(Tables[table])
            print("SQL query => ", insert)
            connect.commit()
            print('Inserted randomly')
            cursor.close()
            connection.connectionlost(connect)
```

existingtable() – перевірка на існування таблиці.

Outputonetable() – вивід вибраної таблиці.

Outpualltables() – вивід усіх таблиць даної БД.

Insertbyuser() – додавання рядків.

Deletebyuser() – видалення рядка.

Deletealot() – видалення декількох рядків.

Updatebyuser() – редагування одного з столбців рядка.

Updatebyuserallrow() – редагування усього рядка.

Selection() – пошуковий запит.

Randomik() – заповнення таблиць випадковими даними.