

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

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

Лабораторна робота №2

з дисципліни «Бази даних і засоби управління»

Тема: «Створення додатку бази даних, орієнтованого на взаємодію з
СУБД PostgreSQL»

Виконав: студент III курсу

ФПМ групи КВ-81

Поляков Є.А.

Викладач: Петрашенко А. В.

Київ 2020

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

Загальне завдання роботи полягає у наступному:

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

Деталізоване завдання:

1. Забезпечити можливість введення/редагування/видалення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв'язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглий таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом!**

Кількість даних для генерування має вводити користувач з клавіатури. Для тесту взяти 100 000 записів для однієї-двох таблиць.

Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

3. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість введення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
4. Програмний код організувати згідно шаблону Model-View-Controller(MVC. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Посилання на репозиторій:

https://github.com/Gregor-an/Database_lab.git

Пункт №1:

Можливість перегляду назв таблиць і колонок які містяться в них:

```
Choose category:
```

```
1.Tables
2.Insert
3.Update
4.Delete
5.Select
6.Random data
7.Readme
8.Exit
```

```
Tables:
```

```
Write table to see columns
```

```
#####
```

```
Ratings
```

```
Comments
```

```
Users
```

```
Posts
```

```
#####
```

```
Exit
```

```
Write 'Exit' to roll back...
```

```
Columns of 'Users' table:
```

```
#####
```

```
UserID
```

```
Name
```

```
E-mail
```

```
Numbers of subscribers
```

```
#####
```

```
Write any button to roll back...
```

Insert:

```
Insert:(Write table to insert or 'Exit'):  
Users
```

```
Insert:(Write columns to insert into):  
UserID  
Name  
E-mail  
Numbers of subscribers
```

```
Insert:  
Write values to insert 1:  
13  
Name_2  
Name_email  
225
```

```
Insert:  
Write values to insert 3:('string')  
  
Error ПОМИЛКА: стовпець "UserID" відношення "Users" не існує  
LINE 1: INSERT INTO "Users" ("UserID", "Name", "E-mail", "Numbers o...  
      ^  
  
Can't insert in 'Users' table in ['UserID', 'Name', 'E-mail', 'Numbers of subscribers']  
columns this ['13', "'Name_2'", "'Name_2email'", '225', '14', "'Name_4'", "'Name_4email'  
, '333'] values
```

Замість колонки UserID було введено UsersID, що спричинило появу помилки, при введенні таблиці якої не існує чи значень неправильних типів виведеться відповідні повідомлення про помилки.

Після цього користувача поверне на початковий екран.

При введенні коректних значень користувачу виведеться повідомлення:

```
Insert in 'Users' table in ['UserID', 'Name', 'E-mail', 'Numbers of subscribers']  
columns this ['14', "'Emma'", "'Emma@email'", '125'] values
```

Update:

```
Update:(Write table to update or 'Exit'):  
Users
```

```
Update:(Write columns and values):  
Name  
Samuel_
```

```
Update:(Write condition):("table_name",'string')  
  
Error ПОМИЛКА: повторювані значення ключа порушують обмеження унікальності "Users_Name_key"  
DETAIL: Ключ ("Name")=(Samuel) вже існує.  
  
Can't update in 'Users' table and set "Name"='Samuel' by 't' condition
```

Користувач може не ввести умову оновлення значень і відповідна команда спробує оновити всі значення в колонці в даному випадку 'Name', але в таблиці 'Users' є обмеження на унікальність поля 'Name', тому виведено помилку.

```
Update:(Write condition):("table_name",'string')  
"UserID" = 14  
Update in 'Users' table and set "Name"='Samuel' by "UserID" = 14 condition
```

При вдалому запиті виведеться повідомлення.

```
Update:(Write condition):("table_name",'string')  
"UserID" = 14  
Update in 'Users' table and set "Name"='Samuel' by "UserID" = 14 condition
```

Delete:

```
Delete:(Write table to delete from or 'Exit'):  
User_
```

```
Delete:  
Write condition or press 't' to delete all  
Press 'Enter' to return back:  
("table_name",'string')  
"UserID" = 12_
```

```
Delete:  
Write condition or press 't' to delete all  
Press 'Enter' to return back:  
("table_name",'string')  
"UserID" = 14_
```

```
Delete:  
Write condition or press 't' to delete all  
Press 'Enter' to return back:  
("table_name",'string')  
"UserID" = 14  
Can't delete in item(s) '"User"' table by "UserID" = 14 condition
```

“Delete” виводить помилку тільки у випадку неправильного вводу назви таблиці, або неіснуючої колонки.

“Delete” дозволяє користувачу видалити поле із таблиці “Users” навіть коли на нього вказують інші таблиці. “Delete” видаляє всі значення в усіх таблицях які були зв’язані з тим що потрібно видалити.

Відповідно якщо користувач захоче видалити всі поля із таблиці “Users”, то “Delete” видалить всю інформацію із усіх таблиць.

При введенні користувачем назв таблиць, колонок та простих значень відбувається валідація цих даних.

У випадку колонок і таблиць до них додаються “” та розділювальні знаки:

```
for x in range(len(columns)):
    if x==len(columns)-1:
        col=col+"\\" + columns[x] + "\\"
    else:
        col=col+"\\" + columns[x] + "\\"+", "

for x in range(len(tables)):
    if x==len(tables)-1:
        tab=tab + "\\"+ tables[x]+ "\\"
    else:
        tab=tab + "\\"+ tables[x]+ "\\" +", "
```

У випадку змінних розрізняються чисельні і рядкові константи.

Якщо введений рядок то до нього додаються “’”:

```
for x in range(len(values)):
    if isinstance(values[x], str):
        values[x]="\'" + values[x] + "\'
```

Після цього оброблені дані відправляються далі, де вже об’єднуються у виконувану команду.

Пункт №2:

Генерація даних автоматично реалізована для таблиць “Users” і “Posts”.

При введенні користувачем іншої таблиці буде виведено повідомлення про не можливість виконати дану команду.

```
Can't add in 'User' table 5 rows of random data
```

```
Random Data:(Write table to add data or 'Exit'):  
Users_
```

```
Random Data:  
How much rows to add?  
100000  
  
Please wait...  
Add in 'Users' table 100000 rows of random data
```

	UserID [PK] integer	Name name	E-mail character varying	Numbers of subscribers integer
1	100017	0f38a	da9a7	94
2	100016	33377	fd380	99
3	100015	e079b	ed097	36
4	100014	35991	99a26	75
5	100013	02d27	028e0	82
6	100012	6de3e	4dd55	84
7	100011	1f035	252d9	11
8	100010	76ecd	16aef	32
9	100009	3e4f0	5c5f6	36
10	100008	fc6ce	3f9ad	19
11	100007	a6419	bc7cc	48
12	100006	623af	67c7a	93
13	100005	790d7	8db28	97
14	100004	07e63	d7e32	5
15	100003	fd516	06bc9	65
16	100002	30275	d43c8	65
17	100001	b4c0a	8b5d3	8

Дана SQL команда генерує дані для таблиці “Users”.

```
insert into "Users" ("Name","E-mail","Numbers of subscribers")
values (
(select substr(md5(random()::varchar), 0, 6)),
(select substr(md5(random()::varchar), 0, 6)),
(random()*100));""
```

У випадку “Posts” значення для “UserIDFK” беруться рандомно із вже існуючих значень “UserID” в таблиці “Users”.

```
insert into "Posts" ("Topic of the post","Post","Time of creating","UserIDFK")
values (
(select substr(md5(random()::varchar), 0, 6)),
(select substr(md5(random()::text), 0, 10)),
(select NOW() -
random() * (NOW() -
timestamp '2014-01-10 10:00:00')),
(SELECT "UserID" FROM "Users"
OFFSET floor(random()*(select count("UserID") from "Users")) LIMIT 1));""
```

Пункт №3:

Select:

```
Select:
Write table(s) to select from:
Users
```

```
Select:
Write columns to select:(* to all)
*
```

```
Select:(Write condition or press 'Enter' to skip):
"UserID" < 20_
```

```
Select ['UserID', 'Name', 'E-mail', 'Numbers of subscribers'] column(s) in "Users" table(s) is done
```

UserID	Name	E-mail	Numbers of subscribers
1	Name_1	Name_1.email.com	223
2	bf59c	091ff	17
3	71890	4c9ec	4
4	536ca	a5b15	99
5	f4e3e	6eeb8	72
6	16ed3	ad51a	50
7	7f4e3	0688a	54
8	b4d78	8efeb	77
9	11720	f11fa	24
10	fdf1c	197ae	96
11	a	Ea	23
12	B	EB	232
13	ef5d7	4aea5	79
14	5b0fd	785d0	46
15	3706d	ea6f0	55
16	97865	063a9	86
17	b6f04	ad0fe	11
18	733c9	a0b88	52
19	6fd04	891e4	87






```
Time of execution: 0.0026171207427978516 sec
```

В даному випадку умова використана для зручності так як в таблиці “Users” 100017 записів і їх вивід займе деякий час.

Ця ж команда в PgAdmin:

```
1 select * from "Users" where "UserID" < 20
```

Data Output

	 UserID [PK] integer 	Name name 	E-mail character varying 	Numbers of subscribers integer 
1	1	Name_1	Name_1.email.com	223
2	2	bf59c	091ff	17
3	3	71890	4c9ec	4
4	4	536ca	a5b15	99
5	5	f4e3e	6eeb8	72
6	6	16ed3	ad51a	50
7	7	7f4e3	0688a	54
8	8	b4d78	8efeb	77
9	9	11720	f11fa	24
10	10	fdf1c	197ae	96
11	11	a	Ea	23
12	12	B	EB	232
13	13	ef5d7	4aea5	79
14	14	5b0fd	785d0	46
15	15	3706d	ea6f0	55
16	16	97865	063a9	86
17	17	b6f04	ad0fe	11
18	18	733c9	a0b88	52
19	19	6fd04	891e4	87

Ми маємо можливість задати більше таблиць в пошук.

```
Select:
Write table(s) to select from:
Posts
Comments_
```

А також декілька конкретних колонок.

```
Select:
Write columns to select:(* to all)
PostID
CommentID
Post
_
```

В таблиці “Posts” також багато полів, тому додамо умову, щоб отримати тільки ті дописи до яких поставили коментар.

```
Select:(Write condition or press 'Enter' to skip):
"PostID" = "PostIDFK"
```

```
Select ['PostID', 'CommentID', 'Post'] column(s) in "Posts","Comments" table(s) is done

PostID          CommentID      Post
20026           1              77f2faec1
20027           2              a85984e00
20028           3              af601dd1c
20029           4              a77c1bbe1
20030           5              fd0dbf3eb
20031           6              f5e6a2bed

Time of execution: 0.003988504409790039 sec
```

Ми отримали вибірку із всіх постів до яких написані коментарі.

Ця ж команда в PgAdmin:

Query Editor

```
1 select "PostID","CommentID","Post"
2 from "Posts","Comments"
3 where "PostID" = "PostIDFK"
```

Data Output

	PostID integer	CommentID integer	Post text
1	20026	1	77f2faec1
2	20027	2	a85984e00
3	20028	3	af601dd1c
4	20029	4	a77c1bbe1
5	20030	5	fd0dbf3eb
6	20031	6	f5e6a2bed

```
Select:
Write table(s) to select from:
Comments_
```

```
Select:
Write columns to select:(* to all)
*_
```

```
Select ['CommentID', 'Comment', 'Time of sending', 'UserIDFK', 'PostIDFK'] column(s) in "Comments" table(s) is done
```

CommentID	Comment	Time of sending	UserIDFK	PostIDFK
1	Com_1	2020-01-01	1	20026
2	Com_2	2020-01-02	2	20027
3	Com_3	2020-01-03	2	20028
4	Com_4	2020-01-04	3	20029
5	Com_5	2020-01-05	1	20030
6	Com_6	2020-02-05	3	20031

```
Time of execution: 0.0 sec
```

Повна таблиця “Comments”

Ця ж команда в PgAdmin:

Query Editor

1 `select * from "Comments"`

Data Output

	CommentID [PK] integer	Comment text	Time of sending date	UserIDFK integer	PostIDFK integer
1	1	Com_1	2020-01-01	1	20026
2	2	Com_2	2020-01-02	2	20027
3	3	Com_3	2020-01-03	2	20028
4	4	Com_4	2020-01-04	3	20029
5	5	Com_5	2020-01-05	1	20030
6	6	Com_6	2020-02-05	3	20031

Ілюстрація програмного коду з Git:

Controller.py

```

1 import backend_sql as bs
2 import sys, os
3 import time
4
5 class Controller(object):
6
7     def __init__(self, model, view):
8         self.model = model
9         self.view = view
10
11     def tables(self):
12         self.model.cursor=self.model.table()
13         cur=self.model.cursor
14         self.view.show_tables(cur)
15
16     def columns_in_tab(self,table):
17         tab = "\"" + table + "\""
18         self.model.cursor=self.model.columns_in_tab(tab)
19         cur=self.model.cursor
20         self.view.show_table_columns(cur,table)
21
22     def insert(self,table,columns,values,count):
23         try:
24             tab = "\"" + table + "\""
25             col=""
26             val=""
27             for x in range(len(columns)):
28                 if x==len(columns)-1:
29                     col=col+"\"" + columns[x] + "\""
30                 else:
31                     col=col+"\"" + columns[x] + "\"", "
32             for x in range(len(values)):
33                 if isinstance(values[x], str):
34                     values[x]="\'" + values[x] + \'
35             tmp=0
36             ind=int(len(values)/count)
37             for i in range(ind):
38                 val=val+'('
39                 for x in range(count):
40                     if x==count-1:
41                         val=val + values[tmp+x]
42                     else:
43                         val=val + values[tmp+x] + ", "
44                 if i ==ind-1:
45                     val=val+')'
46             else:

```



```

46         else:
47             val=val+',')
48             tmp=tmp+count
49             self.view.show_insert(table,columns,values,self.model.insert(tab,col,val))
50     except:
51         return
52
53
54
55     def update(self,table,columns,values,condition):
56         if len(condition)==0:
57             condition="\t\t\t"
58             tab = "\"" + table + "\""
59             set=""
60             for x in range(len(values)):
61                 if isinstance(values[x], str):
62                     values[x]="'" + values[x] + "'"
63             for x in range(len(columns)):
64                 if x==len(columns)-1:
65                     set=set+"\"" + columns[x] + "\""+"="+ values[x]
66                 else:
67                     set=set+"\"" + columns[x] + "\""+"="+ values[x] + ", "
68
69
70             self.view.show_update(table,set,condition,self.model.update(tab,set,condition))
71
72     def delete(self,table,condition):
73         table="\""+table+"\"
74         if table == "\"Users\"":
75             flag=self.delete_user(table, condition)
76         elif table == "\"Posts\"":
77             flag=self.delete_post(table, condition)
78         elif table == "\"Comments\"":
79             flag=self.delete_comment(table, condition)
80         elif table == "\"Ratings\"":
81             flag=self.delete_rating(table, condition)
82         else:
83             flag=False
84
85         self.view.display_delete(table, condition,flag)
86
87     def delete_user(self,table, condition):
88         try:
89             f1=self.model.delete("\"Ratings\"", "\"UserIDFK\" in (select \"UserID\" from \"Users\" where "+condition+")"
90                     + "or \"PostIDFK\" in (select \"PostID\" from \"Posts\" where "
91                     + "\"UserIDFK\" in (select \"UserID\" from \"Users\" where "+condition+"))")

```

```

91         + "\"UserIDFK\" in (select \"UserID\" from \"Users\" where "+condition+"))")
92     f2=self.model.delete("\"Comments\"", "\"UserIDFK\" in (select \"UserID\" from \"Users\" where "+condition+")"
93         + "or \"PostIDFK\" in (select \"PostID\" from \"Posts\" where "
94         + "\"UserIDFK\" in (select \"UserID\" from \"Users\" where "+condition+"))")
95     f3=self.model.delete("\"Posts\"", "\"UserIDFK\" in (select \"UserID\" from \"Users\" where "+condition+"))")
96     if f1 and f2 and f3:
97         return self.model.delete("\"Users\"", condition)
98     else:
99         return False
100 except:
101     return False;
102
103 def delete_post(self, table, condition):
104     try:
105         f1=self.model.delete("\"Ratings\"", "\"PostIDFK\" in (select \"PostID\" from \"Posts\" where "+condition+"))")
106         f2=self.model.delete("\"Comments\"", "\"PostIDFK\" in (select \"PostID\" from \"Posts\" where "+condition+"))")
107         if f1 and f2 :
108             return self.model.delete("\"Posts\"", condition)
109         else:
110             return False
111     except:
112         return False;
113
114 def delete_comment(self, table, condition):
115     try:
116         return self.model.delete("\"Comments\"", condition)
117     except:
118         return False;
119
120 def delete_rating(self, table, condition):
121     try:
122         return self.model.delete("\"Ratings\"", condition)
123     except:
124         return False;
125
126 def rand_data(self, table, n_rows):
127     if table == "Users":
128         f1=self.model.rand_data_users(n_rows)
129     elif table == "Posts":
130         f1=self.model.rand_data_posts(n_rows)
131     else:
132         f1=False
133     self.view.rand_data(table, n_rows, f1)
134

```

```

def select(self, columns , tables, condition):
    col=""
    tab=""
    for x in range(len(columns)):
        if x==len(columns)-1:
            if columns[0]=="*":
                col=columns[0]
            else:
                col=col+"\" + columns[x] + "\""
        else:
            col=col+"\" + columns[x] + "\"+", "
    for x in range(len(tables)):
        if x==len(tables)-1:
            tab=tab + "\"+ tables[x]+ \""
        else:
            tab=tab + "\"+ tables[x]+ "\" +", "
    if len(condition)== 0:
        condition="'t\'"

    col_view=[]
    if columns[0]=="*":
        for x in tables:
            t = "\" + x +\""
            for y in self.model.columns_in_tab(t):
                col_view.append(y[0])
    else:
        col_view=columns

    start_time = time.time()
    self.view.select(col_view , tab,self.model.select(col,tab, condition),time.time() - start_time)

```

model.py

```
1  import backend_sql as bs
2
3  class Model(object):
4
5      def __init__(self, cursor,conn):
6          self._cursor = cursor
7          self._conn = conn
8
9      @property
10     def cursor(self):
11         return self._cursor
12
13     @cursor.setter
14     def cursor(self, new_cursor):
15         self._cursor = new_cursor
16
17     @property
18     def conn(self):
19         return self._conn
20
21     @conn.setter
22     def conn(self, new_conn):
23         self._conn = new_conn
24
25     def insert(self,table, columns , values):
26         if bs.insert_into(self._cursor,table, columns,values):
27             self._conn.commit()
28             return True
29         else:
30             self._conn.rollback()
31             return False
32
33
34     def update(self,table,set,where_cond):
35         if bs.update(self._cursor,table,set,where_cond):
36             self._conn.commit()
37             return True
38         else:
39             self._conn.rollback()
40             return False
```

```

def delete(self,table,condition):
    if bs.delete(self._cursor,table,condition):
        self._conn.commit()
        return True
    else:
        self._conn.rollback()
        return False

def rand_t(self,table,count):
    if table == "\"Users\"":
        bs.rand_data_users(self._cursor,self.conn,count)
    elif table == "\"Posts\"":
        bs.rand_data_posts(self._cursor,self.conn,count)
    self._conn.commit()

def select(self, columns , tables, condition):
    return bs.select(self._cursor,self._conn,columns,tables,condition)

def table(self):
    return bs.tables(self._cursor)

def columns_in_tab(self,table):
    return bs.columns_in_tab(self._cursor,table)

def rand_data_users(self,n_rows):
    return bs.rand_data_users(self._cursor,self._conn,n_rows)

def rand_data_posts(self,n_rows):
    return bs.rand_data_posts(self._cursor,self._conn,n_rows)

```

view.py

```
1  import sys, os
2  from psycpg2 import sql
3
4  class View(object):
5
6      @staticmethod
7      def show_tables(cursor):
8          print('#'*20)
9          for tab in cursor:
10             for t in tab:
11                 print(t,end='')
12             print()
13             print('#'*20)
14
15      @staticmethod
16      def show_table_columns(cursor,table):
17          cur=cursor.fetchone()
18          if cur!=None:
19              os.system('cls')
20              print("Columns of '{}' table:\n".format(table))
21              print('#'*20)
22              for c in cur:
23                  print(c,end='')
24              print()
25              for tab in cursor:
26                  for t in tab:
27                      print(t,end='')
28                  print()
29                  print('#'*20)
30          else:
31              os.system('cls')
32              print("Can't write columns of '{}' table".format(table))
33
34      @staticmethod
35      def show_insert(table,columns,values,flag):
36          if flag:
37              print("Insert in '{}' table in '{}' columns this {} values".format(table,columns,values))
38          else:
39              print("Can't insert in '{}' table in {} columns this {} values".format(table,columns,values))
40
```

```

41     @staticmethod
42     def show_update(table,set,cond,flag):
43         if flag:
44             print("Update in '{}' table and set {} by {} condition".format(table,set,cond))
45         else:
46             print("Can\'t update in '{}' table and set {} by {} condition".format(table,set,cond))
47
48     @staticmethod
49     def display_delete(table, condition,flag):
50         if flag:
51             print("Delete in item(s) '{}' table by {} condition".format(table,condition))
52         else:
53             print("Can\'t delete in item(s) '{}' table by {} condition".format(table,condition))
54
55     @staticmethod
56     def rand_data(table,n_rows,flag):
57         if flag:
58             print("Add in '{}' table {} rows of random data".format(table,n_rows))
59         else:
60             print("Can't add in '{}' table {} rows of random data".format(table,n_rows))
61
62     @staticmethod
63     def select(columns , tables,cursor,time):
64         if cursor!=None:
65             print("Select {} column(s) in {} table(s) is done\n".format(columns,tables))
66             for x in columns:
67                 print("%-25s" % x,end='')
68             print()
69             for cur in cursor:
70                 for c in cur:
71                     print("%-25s" % c,end='')
72                 print()
73             print("\nTime of execution: {} sec".format(time))
74         else:
75             print("Can't select {} column(s) in {} table(s)".format(columns,tables))

```

backend_sql.py

```
1  from psycopg2 import sql
2
3  #####INSERT#####
4  def insert_into(cursor,table, columns , values):
5      try:
6          str= "INSERT INTO "
7          str=str + table + ' ('
8          str=str+columns+') '
9          str=str+' VALUES ' + values
10
11         cursor.execute(sql.SQL(str))
12         return True
13     except Exception as err:
14         print("Error {} ".format(err))
15         return False
16
17  #####UPDATE#####
18  def update(cursor,table,set,where_cond):
19      try:
20          str="UPDATE "
21          str= str + table
22          str= str + ' set ' + set
23          str= str + ' where ' + where_cond
24          cursor.execute(sql.SQL(str))
25          return True
26     except Exception as err:
27         print("Error {} ".format(err))
28         return False
29
30
31  #####DELETE#####
```



```

31 #####DELETE#####
32 def delete(cursor, table,condition):
33     try:
34         str='DELETE FROM ' + table + ' where ' + condition
35         cursor.execute(sql.SQL(str))
36         return True
37     except Exception as err:
38         print("Error {} ".format(err))
39         return False
40
41 #####
42
43 def rand_data_users(cursor,conn,n_rows):
44     x=0
45     while x < n_rows:
46         x= r_d_u(cursor,x,conn)
47         x=x+1
48     return True
49 #####
50 def r_d_u(cursor,x,conn):
51     try:
52         str_ind=""" select setval('\Users_UserID_seq',(select max("\UserID") from "\Users\"));"""
53         cursor.execute(sql.SQL(str_ind))
54         conn.commit()
55     except:
56         conn.rollback()
57         return x-1
58     try:
59         str_rand="""
60         insert into "Users" ("Name","E-mail","Numbers of subscribers")
61         values (
62             (select substr(md5(random()::varchar), 0, 6)),
63             (select substr(md5(random()::varchar), 0, 6)),
64             (random()*100));"""
65         cursor.execute(sql.SQL(str_rand))
66     except Exception as err:
67         conn.rollback()
68         return x-1
69     else:
70         conn.commit()
71     return x

```

```

73 def rand_data_posts(cursor,conn,n_rows):
74     x=0
75     while x < n_rows:
76         x= r_d_p(cursor,x,conn)
77         x=x+1
78     return True
79
80 def r_d_p(cursor,x,conn):
81     try:
82         str_ind=""" select setval('\Posts_PostID_seq\',(select max(\PostID\) from \Posts\));"""
83         cursor.execute(sql.SQL(str_ind))
84         conn.commit()
85     except:
86         conn.rollback()
87         return x-1
88     try:
89         str_rand="""
90         insert into "Posts" ("Topic of the post","Post","Time of creating","UserIDFK")
91         values (
92             (select substr(md5(random)::varchar), 0, 6)),
93             (select substr(md5(random)::text), 0, 10)),
94             (select NOW() -
95             random() * (NOW() -
96                 timestamp '2014-01-10 10:00:00')),
97             (SELECT "UserID" FROM "Users"
98                 OFFSET floor(random()*(select count("UserID") from "Users")) LIMIT 1));"""
99         cursor.execute(sql.SQL(str_rand))
100     except Exception as err:
101         conn.rollback()
102         return x-1
103     else:
104         conn.commit()
105     return x
106 #####

```

```

108 def select(cursor,conn, columns , tables, condition):
109     try:
110         str= "SELECT " + columns
111         str = str + " FROM " + tables
112         str = str + " WHERE " +condition
113         cursor.execute(sql.SQL(str))
114         return cursor
115     except Exception as err:
116         print("Error {} ".format(err))
117         conn.rollback()
118         return []
119
120 def tables(cursor):
121     try:
122         str= """
123             SELECT DISTINCT TABLE_NAME
124             FROM INFORMATION_SCHEMA.COLUMNS
125             WHERE TABLE_CATALOG = 'Blog' AND TABLE_SCHEMA = 'public'
126             """
127         cursor.execute(sql.SQL(str))
128         return cursor
129     except Exception as err:
130         print("Error {} ".format(err))
131
132 def columns_in_tab(cursor,table):
133     try:
134         str= """
135             SELECT column_name
136             FROM INFORMATION_SCHEMA.COLUMNS
137             WHERE TABLE_NAME = """ + table
138
139         cursor.execute(sql.SQL(str))
140         return cursor
141     except Exception as err:
142         print("Error {} ".format(err))
143

```

menu.py

```
1  import Controller as cont
2  import model as mod
3  import view
4  import psycpg2 as pc
5  import sys, os
6
7
8  def layer_1():
9      print("Choose category:\n")
10     print("1.Tables")
11     print("2.Insert")
12     print("3.Update")
13     print("4.Delete")
14     print("5.Select")
15     print("6.Random data")
16     print("7.Readme")
17     print("8.Exit")
18
19  def layer_2_tables(c):
20     print("Tables:\nWrite table to see columns\n")
21     c.tables()
22     print("\nExit")
23
24  def layer_2_insert(c):
25     print("Insert:(Write table to insert or 'Exit'):")
26     ch=input();
27     if ch=="Exit":
28         return
29     else:
30         os.system('cls')
31         tab=ch
32         values=[]
33         columns=[]
34         print("Insert:(Write columns to insert into):")
35         while True:
36             ch=input()
37             if len(ch)==0:
38                 break
39             columns.append(ch)
40         tmp=0
41         count=1
42         print("Insert:\nWrite values to insert {}:(\'string\')".format(count))
43         while True:
44             if tmp ==len(columns):
45                 os.system('cls')
```

```

46         count=count+1
47         print("Insert:\nWrite values to insert {}:({}\string\')".format(count))
48         tmp=0
49         ch=input()
50         tmp=tmp+1
51         if len(ch)==0:
52             break
53         values.append(ch)
54         c.insert(tab,columns,values,len(columns))
55         input()
56
57     def layer_2_update(c):
58         print("Update:(Write table to update or 'Exit'):")
59         ch=input();
60         if ch=="Exit":
61             return
62         else:
63             os.system('cls')
64             tab=ch
65             values=[]
66             columns=[]
67             print("Update:(Write columns and values):")
68             while True:
69                 ch=input()
70                 if len(ch)==0:
71                     break
72                 v=input()
73                 columns.append(ch)
74                 values.append(v)
75             os.system('cls')
76             print("Update:(Write condition):({}\table_name\','\string\') ")
77             cond=input()
78             c.update(tab,columns,values,cond)
79             input()
80
81     def layer_2_delete(c):
82         print("Delete:(Write table to delete from or 'Exit'):")
83         ch=input();
84         if ch=="Exit":
85             return
86         else:
87             os.system('cls')
88             tab=ch
89             print("Delete:\nWrite condition or press 't' to delete all\nPress 'Enter' to return back:\n({}\table_name\','\string\')")
90             . . .

```

```

89         print("Delete:\nWrite condition or press 't' to delete all\nPress
90         ch=input()
91         if len(ch)==0:
92             return
93         cond=ch
94         c.delete(tab,cond)
95         input()
96
97     def layer_2_r_d(c):
98         print("Random Data:(Write table to add data or 'Exit'):")
99         ch=input();
100         if ch=="Exit":
101             return
102         else:
103             os.system('cls')
104             tab=ch
105             try:
106                 print("Random Data:\nHow much rows to add?")
107                 rows=int(input())
108                 print("\nPlease wait...")
109             except:
110                 return
111             c.rand_data(tab,rows)
112             input()
113
114     def layer_2_select(c):
115         os.system('cls')
116         condition=""
117         tables=[]
118         columns=[]
119         print("Select:\nWrite table(s) to select from:")
120         while True:
121             ch=input()
122             if len(ch)==0:
123                 break
124             tables.append(ch)
125         os.system('cls')
126         print("Select:\nWrite columns to select:(* to all)")
127         while True:
128             ch=input()
129             if len(ch)==0:
130                 break

```

```

130         break
131     columns.append(ch)
132     os.system('cls')
133     print("Select:(Write condition or press 'Enter' to skip):")
134     condition = input()
135     c.select(columns,tables,condition)
136     input()
137
138 def help():
139     os.system('cls')
140     print("Integer ---- just number ----- 3")
141     print("String ---- using ' ---- 'just_string'")
142     print("Date ---- using ' ---- 'year-month-day'")
143     print("Press anything to return")
144     input()
145
146 def menu():
147     conn=pc.connect(dbname='Blog', user='postgres', password='Egor2708', host='localhost')
148     cur=conn.cursor()
149     c = cont.Controller(mod.Model(cur,conn), view.View())
150     while True:
151         os.system('cls')
152         layer_1()
153         ch= input()
154         if ch=="1":
155             while True:
156                 os.system('cls')
157                 layer_2_tables(c)
158                 print("Write 'Exit' to roll back...")
159                 ch=input()
160                 if ch=="Exit":
161                     os.system('cls')
162                     break
163                 else:
164                     c.columns_in_tab(ch)
165                     print("Write any button to roll back...")
166                     input()
167
168             elif ch== "2":
169                 os.system('cls')
170                 layer_2_insert(c)
171
172             elif ch== "3":
173                 os.system('cls')
174                 layer_2_update(c)
175

```

```
176         elif ch=="4":
177             os.system('cls')
178             layer_2_delete(c)
179
180         elif ch=="5":
181             os.system('cls')
182             layer_2_select(c)
183
184         elif ch=="6":
185             os.system('cls')
186             layer_2_r_d(c)
187
188         elif ch=="7":
189             os.system('cls')
190             help()
191
192         elif ch=="8":
193             os.system('cls')
194             break
195         else:
196             os.system('cls')
197
198
199
200
201     cur.close()
202     conn.close()
203
204
205
206
207 def main():
208     menu()
209
210
211 if __name__ == '__main__':
212     main()
```