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

Github: https://github.com/illya-U/bd_lab_2022/tree/develop/lab_2

Виконав: студент III курсу

ФПМ групи КВ-04

Устименко І.В.

Перевірив:

Київ – 2022

Загальне завдання роботи полягає в такому:

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

Деталізоване завдання:

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

Приклад генерації 100 псевдовипадкових чисел:

```
select trunc(random()*1000)::int
from generate_series(1,100)
```

Data Output | Explain | Messages | Notific

	trunc integer
1	368
2	773
3	29
4	66
5	497
6	956

Приклад генерації 5 псевдовипадкових рядків:

```
select chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)
from generate_series(1,5)
```

Data Output | Explain | Messages | Notifications

	?column? text
1	NE
2	MQ
3	RN
4	DW
5	DA

Приклад генерації псевдовипадкової мітки часу з діапазону доступний за посиланням.

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

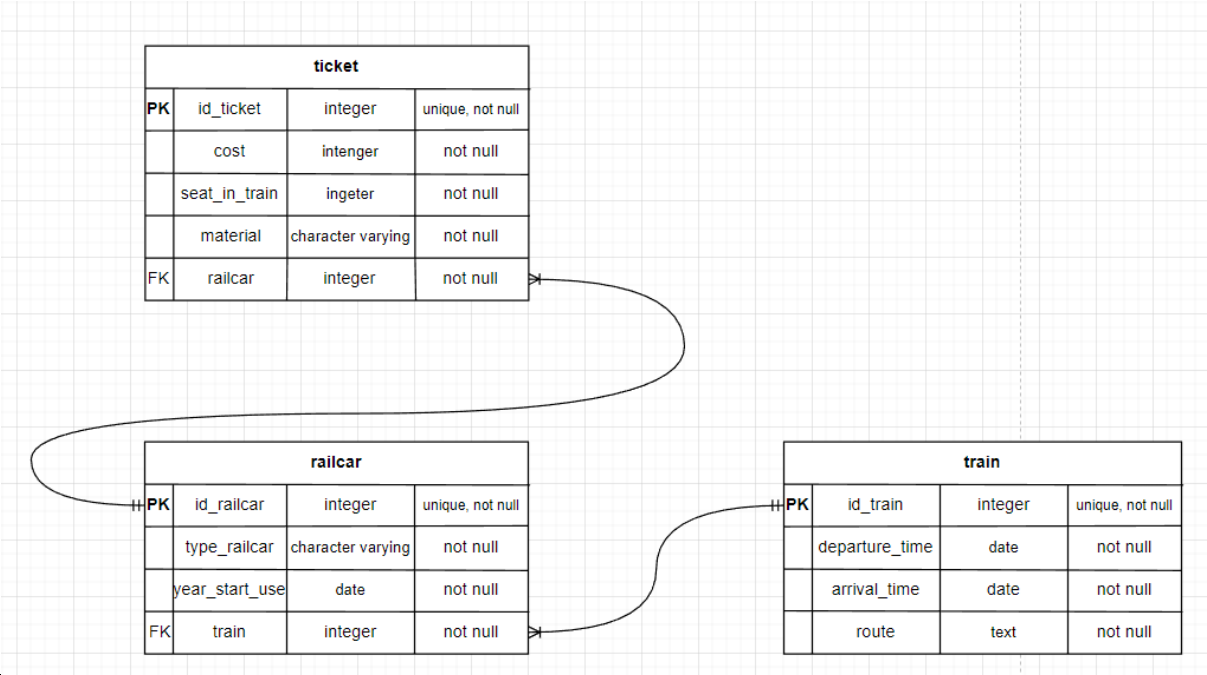
Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності foreign key).

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 уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після

виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

4. Програмний код організувати згідно шаблону Model-View-Controller MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** без ORM).

   Рекомендована бібліотека взаємодії з PostgreSQL Psycopg2: http://initd.org/psycopg/docs/usage.html)



Сутність	Атрибут	Опис Атрибуту	Тип	Обмеженя
ticket	id	unique identifier	integer	not null unique
	cost	ціна в ~~₴~~	integer	not null
	seat_in_the_train	Місце перебування пасажира протягом більшої	integer	not null

		частини шляху		
	material	Матеріал з якого виготовлено квиток	character varying	not null
	train	посилання на характеристику	integer	not null
raicar	id	unique identifier	integer	not null unique
	type_railcar	тип вагону	character varying	not null
	year_start_use	рік початку експлуатації вагону	date	not null
	train	посилання на характеристику	integer	not null
train	id	unique identifier	integer	not null unique
	departure_time	час відправлення потягу	date	not null
	arrival_time	час прибуття потягу	date	not null
	route	Шлях по якому прямує потяг	text	not null

Опис функціоналу меню:
Update – оновлення данних в певній колонці,
Add – додання нової колонки,
Delete – видалення нової колонки
Random - рандомна генерація нових значень для колонок на n раз
Search – пошук по базовим данним якоїсь колонки
Info about tables – повна інформація по бд

Відповідь на вимоги до пункту №1 деталізованого завдання:

Ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилучення даних:

```
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 42
You have to enter the command from 1 to 5 ERROR

Process finished with exit code 0
```

Ілюстрації валідації даних при уведенні користувачем:

```
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,lik
Enter command : 3
Your table_name name: train, ticket, railcar
Enter table_name name trin
Enter column atribute,based on which you want to change t
delete FROM trin WHERE
42601
WARNING:Error ОШИБКА:  ошибка синтаксиса в конце
LINE 1: delete FROM trin WHERE
                             ^


Process finished with exit code 0
```

Видалення данних
Данні потяга до видалення:

	id_train [PK] integer	departure_time date	arrival_time date	route text
1	1	2022-10-10	2022-10-10	kyiv-kharkiv
2	10	1993-09-30	1970-01-10	L
3	11	2019-12-13	1970-01-10	A
4	12	2027-07-22	1970-01-10	liva
5	13	2020-10-12	2020-10-14	fsf
6	14	2031-05-23	1970-01-10	X
7	15	1971-10-05	1970-01-10	F
8	16	2000-10-10	2001-10-10	roui
9	17	2000-10-10	2000-10-11	kyiv-poltava
10	18	2000-10-10	2000-10-11	kic

Данні вагона до видалення:

	id_railcar [PK] integer	type_railcar character varying	year_start_use_railcar date	train integer
1	23	bisnes	2020-10-10	1
2	24	D	1978-06-27	16
3	25	F	1984-07-20	11
4	26	H	1989-12-29	10
5	27	H	1987-09-16	11
6	28	B	1972-12-19	14
7	29	A	1972-05-21	14
8	30	A	1971-06-29	13
9	31	B	1972-07-21	15
10	32	E	1981-04-05	1
11	33	D	1978-08-24	12
12	34	bisnes_clas	1999-10-01	1

Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 3
Your table_name name: train, ticket, railcar
Enter table_name name train
Enter column atribute,based on which you want to change this fileld like 'Column_name = 'value" and press Enter,when you want to stop enter columns press '-' and press Enter:id_train = 1
Enter column atribute,based on which you want to change this fileld like 'Column_name = 'value" and press Enter,when you want to stop enter columns press '-' and press Enter:-

delete FROM train WHERE id_train = 1

Process finished with exit code 0

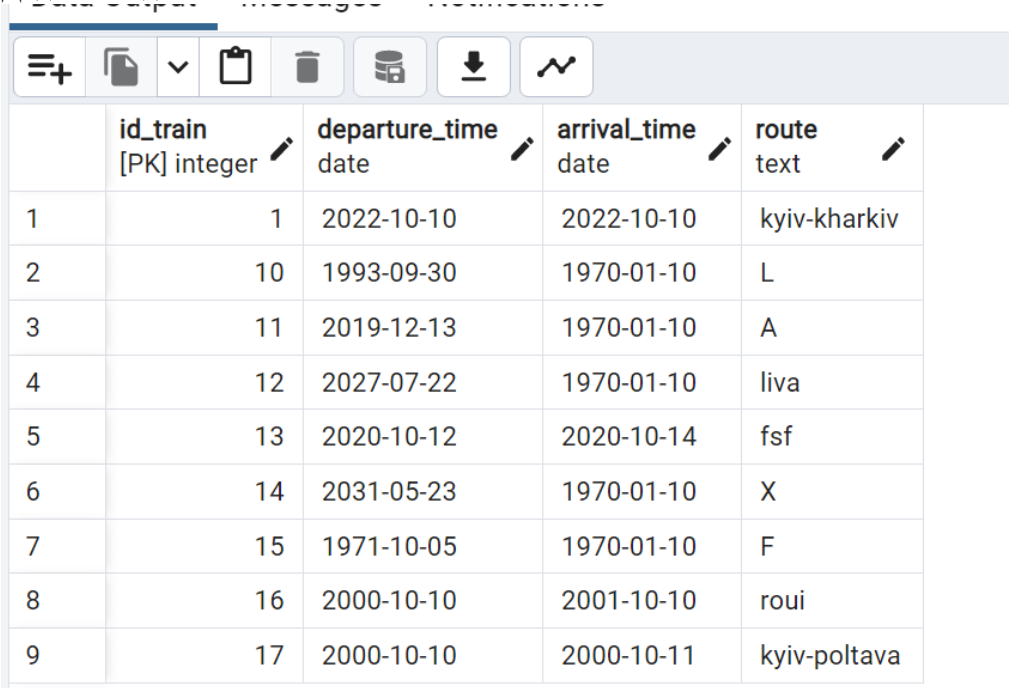
Потяг після видалення:

	id_railcar [PK] integer	type_railcar character varying	year_start_use_railcar date	train integer
1	24	D	1978-06-27	16
2	25	F	1984-07-20	11
3	26	H	1989-12-29	10
4	27	H	1987-09-16	11
5	28	B	1972-12-19	14
6	29	A	1972-05-21	14
7	30	A	1971-06-29	13
8	31	B	1972-07-21	15
9	33	D	1978-08-24	12

Вагон після видалення:

	id_railcar [PK] integer	type_railcar character varying	year_start_use_railcar date	train integer
1	24	D	1978-06-27	16
2	25	F	1984-07-20	11
3	26	H	1989-12-29	10
4	27	H	1987-09-16	11
5	28	B	1972-12-19	14
6	29	A	1972-05-21	14
7	30	A	1971-06-29	13
8	31	B	1972-07-21	15
9	33	D	1978-08-24	12

Додавання данних:



	id_train [PK] integer	departure_time date	arrival_time date	route text
1	1	2022-10-10	2022-10-10	kyiv-kharkiv
2	10	1993-09-30	1970-01-10	L
3	11	2019-12-13	1970-01-10	A
4	12	2027-07-22	1970-01-10	liva
5	13	2020-10-12	2020-10-14	fsf
6	14	2031-05-23	1970-01-10	X
7	15	1971-10-05	1970-01-10	F
8	16	2000-10-10	2001-10-10	roui
9	17	2000-10-10	2000-10-11	kyiv-poltava

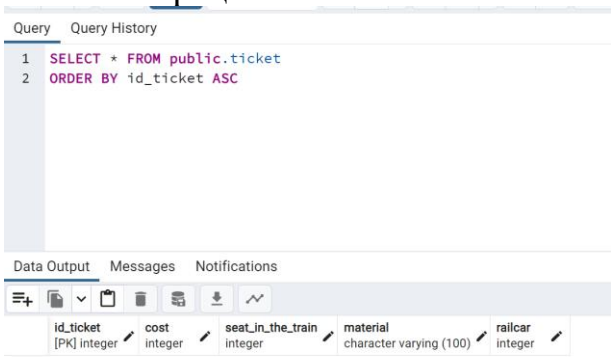
```
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 2
Your table_name name: train, ticket, railcar
Enter table_name name train
departure_time:10-10-2000
arrival_time:11-10-2000
route:kic
INSERT INTO train (departure_time,arrival_time,route) VALUES ('10-10-2000','11-10-2000','kic')

Process finished with exit code 0
```

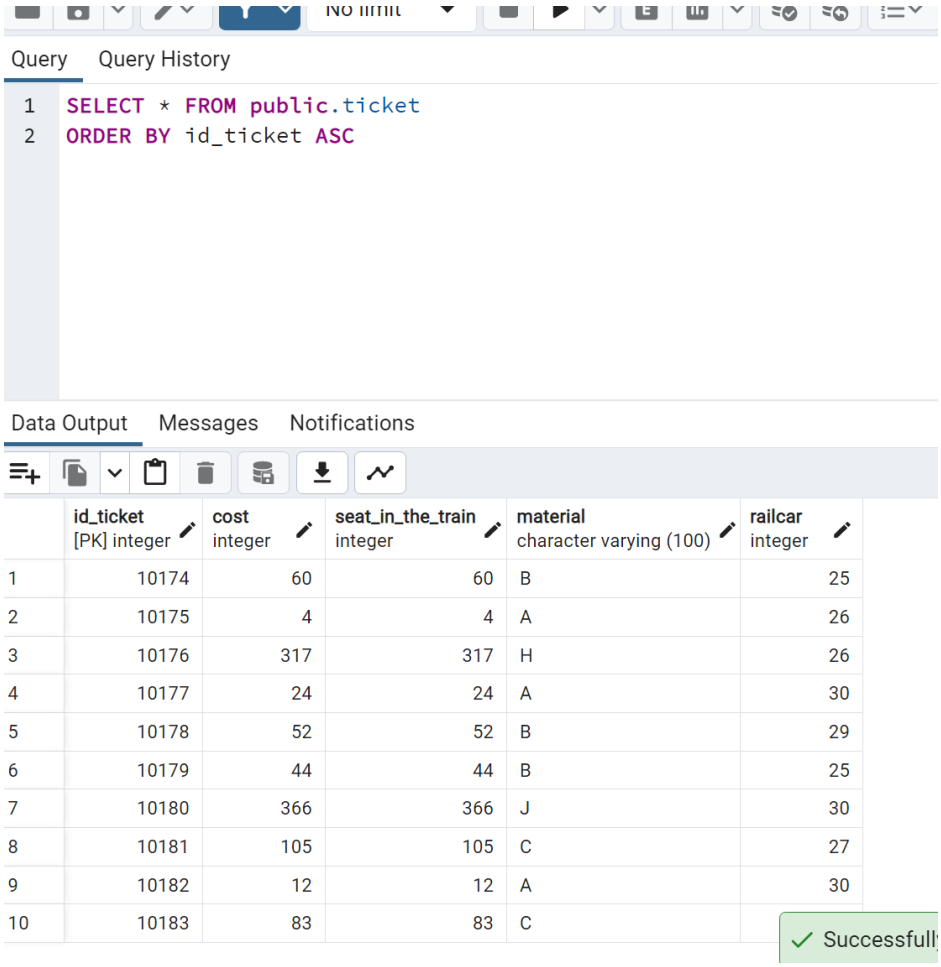
	id_train [PK] integer	departure_time date	arrival_time date	route text
1	1	2022-10-10	2022-10-10	kyiv-kharkiv
2	10	1993-09-30	1970-01-10	L
3	11	2019-12-13	1970-01-10	A
4	12	2027-07-22	1970-01-10	liva
5	13	2020-10-12	2020-10-14	fsf
6	14	2031-05-23	1970-01-10	X
7	15	1971-10-05	1970-01-10	F
8	16	2000-10-10	2001-10-10	roui
9	17	2000-10-10	2000-10-11	kyiv-poltava
10	18	2000-10-10	2000-10-11	kic

Вимоги до пункту №2 деталізованого завдання:

Меню генерації:




```
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 4
Your table_name name: train, ticket, railcar
Enter table_name name ticket
Enter value: 10
INSERT INTO ticket (cost,seat_in_the_train,material,railcar) SELECT trunc(r
```



	id_ticket [PK] integer	cost integer	seat_in_the_train integer	material character varying (100)	railcar integer
1	10174	60	60	B	25
2	10175	4	4	A	26
3	10176	317	317	H	26
4	10177	24	24	A	30
5	10178	52	52	B	29
6	10179	44	44	B	25
7	10180	366	366	J	30
8	10181	105	105	C	27
9	10182	12	12	A	30
10	10183	83	83	C	

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації уведення пошукового запиту та результатів виконання запитів:

	id_train [PK] integer	departure_time date	arrival_time date	route text
1	10	1993-09-30	1970-01-10	L
2	11	2019-12-13	1970-01-10	A
3	12	2027-07-22	1970-01-10	liva
4	13	2020-10-12	2020-10-14	fsf
5	14	2031-05-23	1970-01-10	X
6	15	1971-10-05	1970-01-10	F
7	16	2000-10-10	2001-10-10	roui
8	17	2000-10-10	2000-10-11	kyiv-polta...
9	18	2000-10-10	2000-10-11	kic

```
C:\Users\Admin\Documents\bd\lab2\new_interpritator_3_10\Scripts\python.exe C:\Users\Admin\Documents\bd\lab2\main.py
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 5
Choose presearch scenario which you want:1
Enter column where you want to search values separated by'|':route
please input scenario for route str'k%'
SELECT * FROM railcar,ticket,train WHERE route LIKE 'k%'
[(24, 'D', datetime.date(1978, 6, 27), 16, 10174, 60, 60, 'B', 25, 17, datetime.date(2000, 10, 10), datetime.date(2000, 10, 11), 'kyiv-poltava')]

Process finished with exit code 0
```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git:

main.py:

```python
import psycopg2
from psycopg2 import errors
from model import *
from view import *
import sys


def request():
    input_command = comand_identification()

    if (input_command == '1'):
        table = table_name()
        str_of_columns = input_colums_str_upd()
        str_of_updating_column= take_inf_about_param(str_of_columns)
        str_of_based_column = take_inf_based()
        updt(table,str_of_updating_column,str_of_based_column)
    elif (input_command == '2'):
        table = table_name()
        inf = take_inf_for_adding(table)
        add_inf(table,inf)
    elif (input_command == '3'):
        table = table_name()
        str_of_based_column = take_inf_based()
        delete(table,str_of_based_column)
    elif (input_command == '4'):
        table = table_name()
        data = Data()
        random(table, data)
    elif (input_command == '5'):
        scenario = choose_scenario_search()
        str_of_columns = input_colums_str_search()
        dict_of_searching_var = take_searching_rows(str_of_columns)
        list_of_searching = Search(scenario,dict_of_searching_var)
        print_searching_values(list_of_searching)
    elif (input_command == '6'):
        print_info(info())
    elif (input_command is not str(range(6))):
        comndErr()
        sys.exit()
```

```
menu()

request()
```

view.py(функції виводу данних і їх вводу)

```
import sys
import pprint
import additional_inf

def comndErr():
   print("You have to enter the command from 1 to 5 ERROR")


def comand_identification():
   return input('Enter command : ')

def table_name():
   print('Your table_name name: train, ticket, railcar ')
   return input('Enter table_name name ')

def print_info(my_dict):
    pprint.pprint(my_dict)

def input_colums_str_upd():
    return input("Enter column which you want to update separated by'|':")

def input_colums_str_search():
    return input("Enter column where you want to search values separated
by'|':")

def input_inf_for_column(column_name):
    return input(column_name + ':')

def finding_fields_which_you_want_change():
    return input("Enter column atribute,based on which you want to change
this fileld like 'Column_name = 'value'' and press Enter,when you want to
stop enter columns press '-' and press Enter:")

def Data():
    return input('Enter value: ')

def row():
    return int(input('Enter value: '))

def tablevalid():
    print('The table_name name is wrong ERROR')
    sys.exit()

def choose_scenario_search():
    scenario = input("Choose presearch scenario which you want:")
    return (int(scenario) - 1)

def menu():
    print("Update press 1")
    print("Add press 2")
    print("Delete press 3")
    print("Random press 4")
    print("Search press 5")
    print("Info about tables press 6")
    print("Please print all not numeric types like int and float,like this
'value'")


def take_inf_based():
```

```
    list_of_based_column = []
    while (1):
        temp_str = finding_fields_which_you_want_change()
        if(temp_str == "-"):
            break
        list_of_based_column.append(temp_str)
    str_of_based_column = " AND ".join(list_of_based_column)
    return str_of_based_column

def take_inf_for_adding(table_name):
    mass = []
    for param in additional_inf.parametrs[table_name][1:]:
        mass.append(input_inf_for_column(param[0]))
    return mass

def take_inf_about_param(str_of_columns):
    columns = str_of_columns.split("|")
    str_of_updating_column = ""
    for column in columns:
        str_of_updating_column += column + " = " +
input_inf_for_column(column) + " , "
    str_of_updating_column = str_of_updating_column.rstrip(", ")
    return str_of_updating_column

def take_searching_rows(str_of_columns):
    columns = str_of_columns.split("|")
    dict_of_tacking = {}
    for column in columns:
        type_ = search_type_of_column(column)
        match type_:
                case "int":
                    dict_of_tacking[column] = input(f"please input range int
numbers for {column} separated by '|'").split("|")
                case "str":
                    dict_of_tacking[column] = input(f"please input scenario
for {column} str")
                case "time":
                    dict_of_tacking[column] = input(f"please input range
between two dates for {column} separated by '|'").split("|")
    return dict_of_tacking


def search_type_of_column(column_name):
    for tables in additional_inf.parametrs:
        for parametr in additional_inf.parametrs[tables]:
            if(column_name == parametr[0]):
                return parametr[1]
    raise("this column is not exist")

def print_searching_values(list_of_searching):
    print(list_of_searching)
```

model.py(логіка програми)

```
import psycopg2
import additional_inf


def setup_conf(func):
    def wrapper(*args, **kwargs):
        con = psycopg2.connect(
            database="railway ticket sales service",
            user="postgres",
            password="1111",
            host="localhost",
```

```
            port="5432"
        )
        con.set_session(autocommit=True)
        curso_r = con.cursor()
        smth = func(con, curso_r, *args, **kwargs)
        curso_r.close()
        con.close()
        return smth
    return wrapper


@setup_conf
def info(con, cursor_r):
    dict_of_all_tables = {}
    try:
        for table in additional_inf.parametrs:
            dict_of_all_tables[table] = []
            flag_of_take_memory = 1
            for param in additional_inf.parametrs[table]:
                param = param[0]

cursor_r.execute(additional_inf.set_command["inf"].replace("table",
table).replace("param", param))
                list_of_all_parametrs = cursor_r.fetchall()
                if (flag_of_take_memory):
                    for memory_take in range(len(list_of_all_parametrs)):
                        dict_of_all_tables[table].append({})
                    flag_of_take_memory = 0
                for i in range(len(list_of_all_parametrs)):
                    dict_of_all_tables[table][i][param] =
list_of_all_parametrs[i][0]
            flag_of_take_memory = 1
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')
    else:
        return dict_of_all_tables

@setup_conf
def random(con, cursor_r, table_name, n):
    param = ",".join([*map(lambda
x:additional_inf.parametrs[table_name][x][0], range(1,
len(additional_inf.parametrs[table_name])))])
    try:
        for i in range(int(n)):
            take_sql_str_to_two_stream(cursor_r,
additional_inf.set_command["random"][table_name].replace("parametrs", param))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')

@setup_conf
def delete(con, cursor_r, table_name, str_of_based_column):
    try:
        take_sql_str_to_two_stream(cursor_r,
additional_inf.set_command["delete"].replace("str_of_based_column",
str_of_based_column).replace("table_name", table_name))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')

@setup_conf
def add_inf(con, cursor_r, table_name, mass):
    param = ",".join([*map(lambda x:
```

```
additional_inf.parametrs[table_name][x][0], range(1,
len(additional_inf.parametrs[table_name]))])
    mass = [*map(lambda x: "'{0}'".format(x), mass)]
    values = ",".join(mass)
    try:
        take_sql_str_to_two_stream(cursor_r,
additional_inf.set_command["add_inf"].replace("table_name",
table_name).replace("parametrs", param).replace("values", values))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')


@setup_conf
def updt(con, cursor_r, table_name, str_of_updating_column,
str_of_based_column):
    try:
        take_sql_str_to_two_stream(cursor_r,
additional_inf.set_command["upd_inf"].replace("table_name",
table_name).replace("str_of_updating_column",
str_of_updating_column).replace("str_of_based_column", str_of_based_column))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')


@setup_conf
def Search(con, cursor_r, scenario, dict_of_searching_var):
    list_of_commands = []
    for key, value in dict_of_searching_var.items():
        if(isinstance(value, list)):
            list_of_commands.append(key + " > " + value[0] + " AND " + key +
" < " + value[1])
        else:
            list_of_commands.append(key + " LIKE " + value)
    commands = ' AND '.join(list_of_commands)
    try:
        take_sql_str_to_two_stream(cursor_r,
additional_inf.set_command["presearch"][scenario] + commands)
        searching_values = cursor_r.fetchall()
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')
    else:
        return searching_values

def take_sql_str_to_two_stream(cursor_r, sql_stream):
    print(sql_stream)
    cursor_r.execute(sql_stream)
```

set_command(словник для інформації(використовуються як json))

```
set_command = \
    {
        "random":
            {
              "train":"INSERT INTO train (parametrs) SELECT timestamp '1970-
01-10 20:00:00' + random()* (timestamp '2033-01-20 20:00:00' - timestamp
'1970-01-10 10:00:00'),timestamp '1970-01-10 20:00:00'+ random() * (timestamp
'1970-01-20 20:00:00' - timestamp '1970-01-20
20:00:00'),chr(trunc(65+random()*25)::int)",
              "railcar":f"INSERT INTO railcar (parametrs) SELECT
chr(trunc(65+random()*25)::int), timestamp '1970-01-10 20:00:00'+ random() *
(timestamp '2033-01-20 20:00:00' - timestamp '1970-01-10 10:00:00'),id_train
From train order by random() limit 1",
```

```
            "ticket":f"INSERT INTO ticket (parametrs) SELECT
trunc(random()*1000)::int,trunc(random()*1000)::int,chr(trunc(65+random()*25)
::int),id_railcar FROM railcar order by random() limit 1"
            },
        "delete":"delete FROM table_name WHERE str_of_based_column",
        "add_inf":"INSERT INTO table_name (parametrs) VALUES (values)",
        "upd_inf":f"UPDATE table_name SET str_of_updating_column WHERE
str_of_based_column;",
        "inf":"SELECT param FROM table",
        "presearch":
            [
                "SELECT * FROM train, ticket WHERE ",
                "SELECT id_railcar,id_ticket,material,cost FROM railcar,
ticket WHERE ",
                "SELECT * FROM railcar,ticket,train WHERE "
            ]
    }

parametrs = \
            {

"train":[("id_train","int"),("departure_time","time"),("arrival_time","time")
,("route","str")],

"railcar":[("id_railcar","int"),("type_railcar","str"),("year_start_use_railc
ar","time"),("train","int")],

"ticket":[("id_ticket","int"),("cost","int"),("seat_in_the_train","int"),("ma
terial","str"),("railcar","int")]
            }
```

requirements.txt(використанні ліби)

```
attrs==22.1.0
colorama==0.4.6
exceptiongroup==1.0.1
iniconfig==1.1.1
packaging==21.3
pluggy==1.0.0
psycopg2==2.9.5
pyparsing==3.0.9
tomli==2.0.1
```