

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**

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

Виконав: студент ІII курсу групи КВ-92 Пургін В.А.

Перевірив: Петрашенко А.В.

Київ – 2021

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

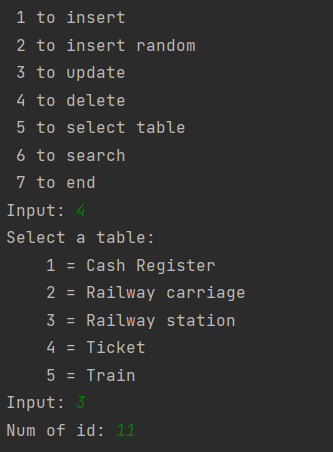
*Загальне завдання* роботи полягає у наступному:

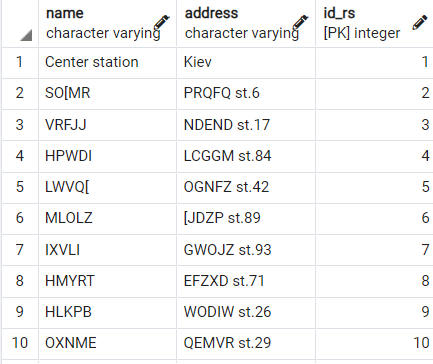
1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами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

URL репозиторію з вихідним кодом: https://github.com/VladislavPurgin/BD\_lab

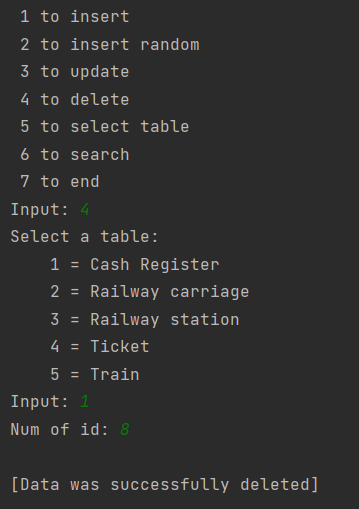
**Результати виконання операції вилучення запису батьківської таблиці**:

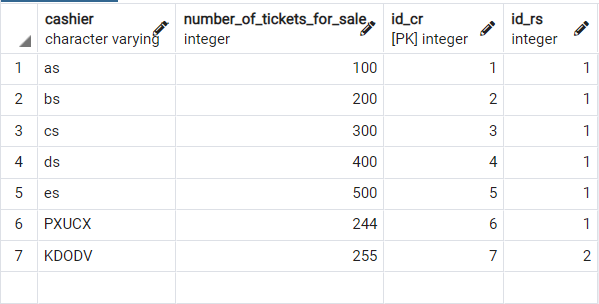
З таблиці Railway station:



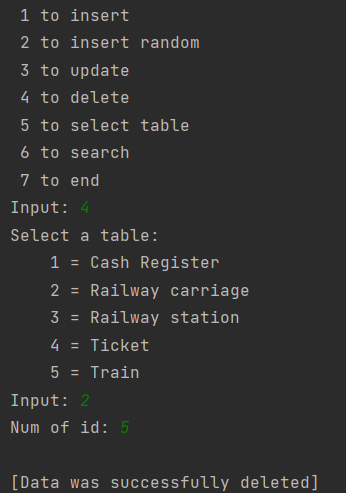


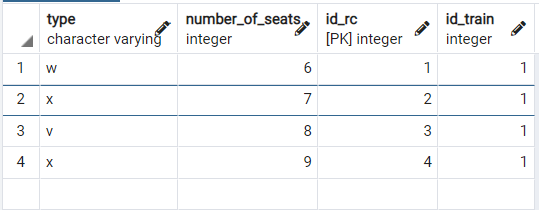
З таблиці Cash Register:





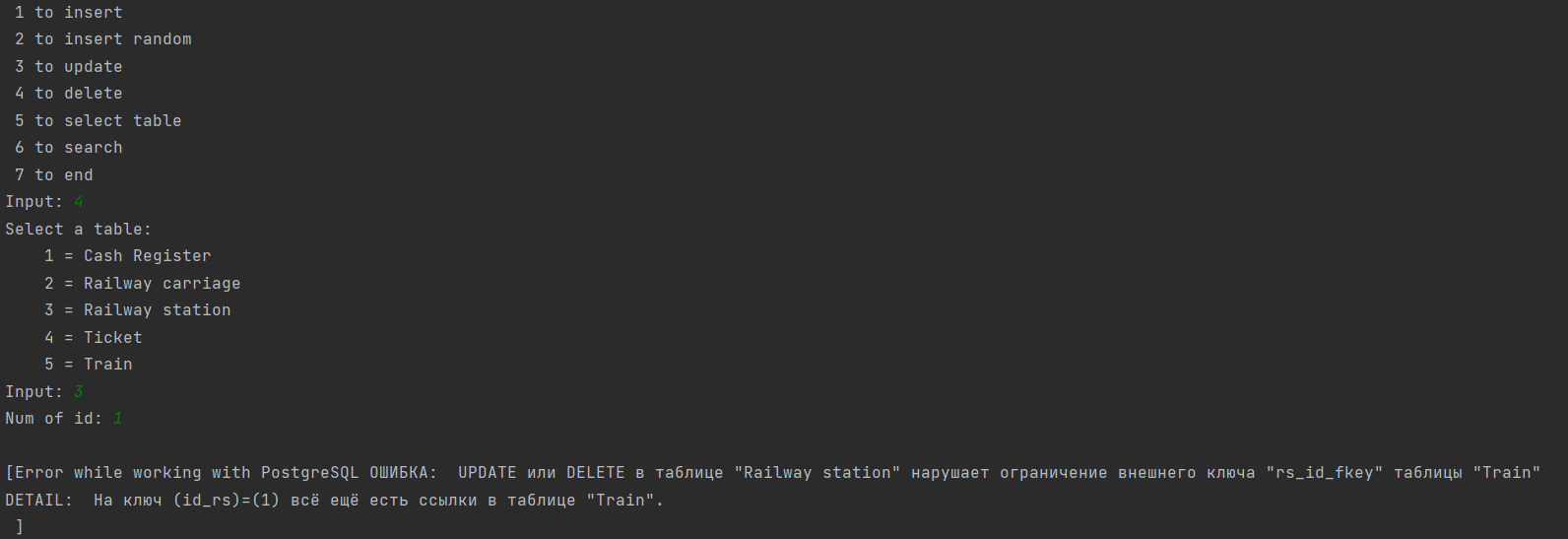
З таблиці Railway carriage:





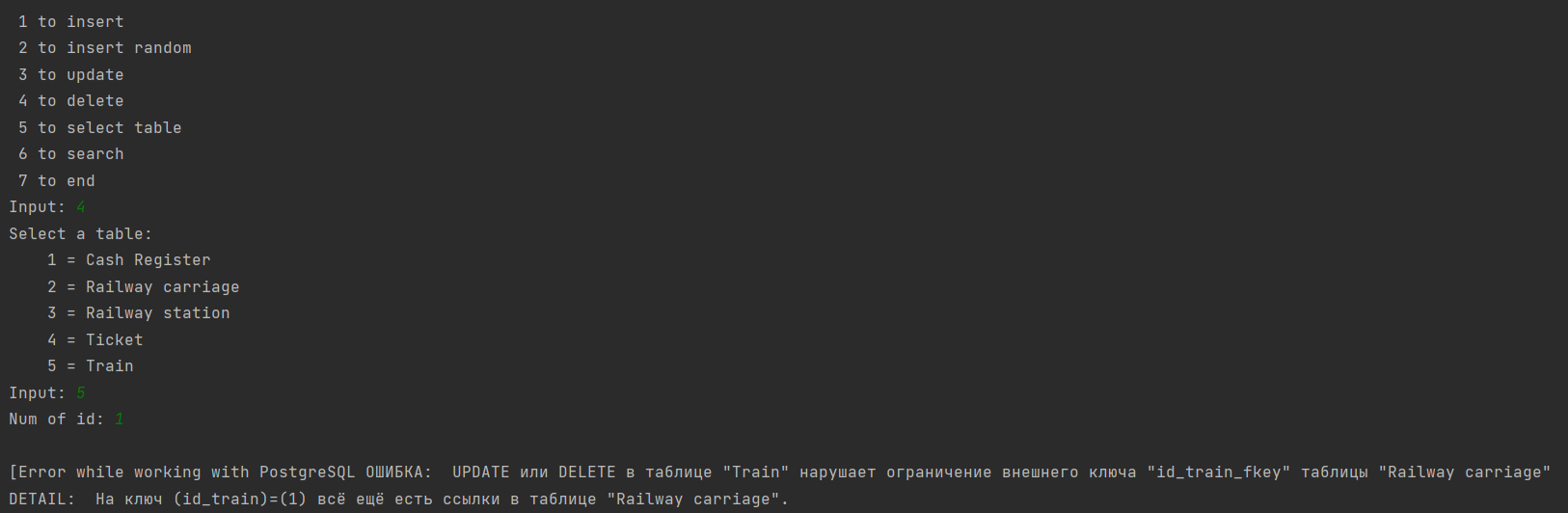
Результат, коли вилучення неможливе

З таблиці Railway station:



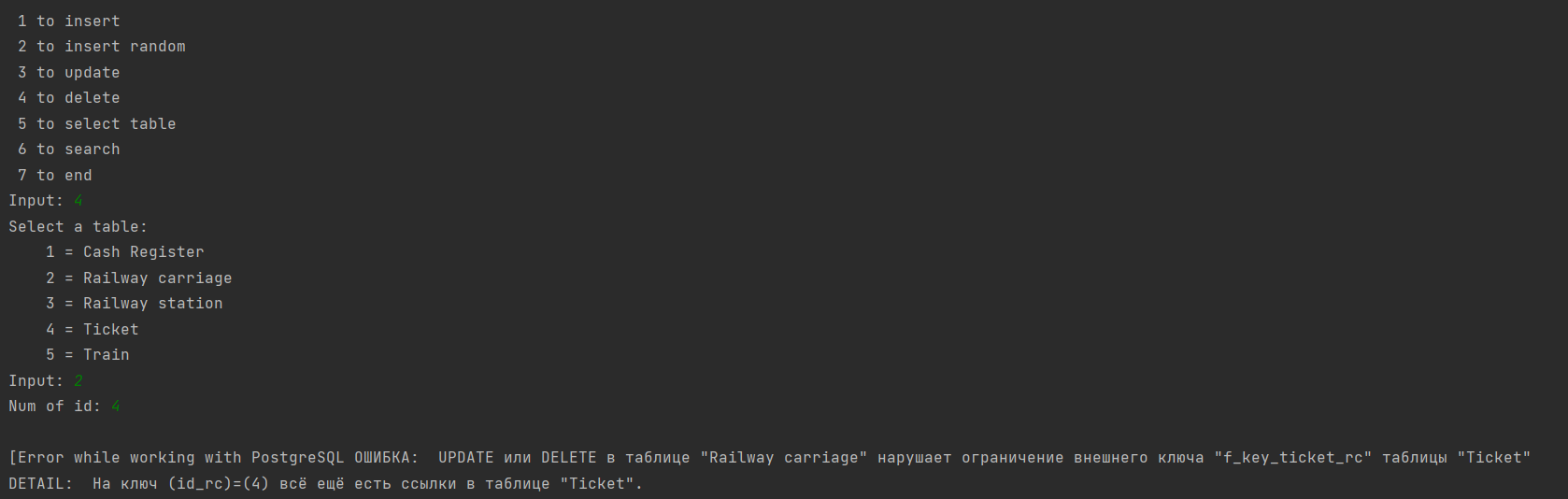
Результат, коли вилучення неможливе

З таблиці Train:



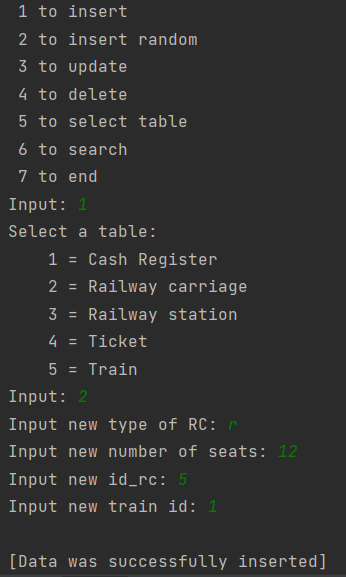
Результат, коли вилучення неможливе

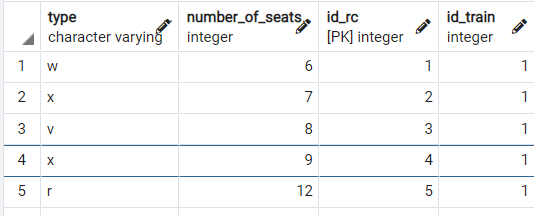
З таблиці Railway carriage:



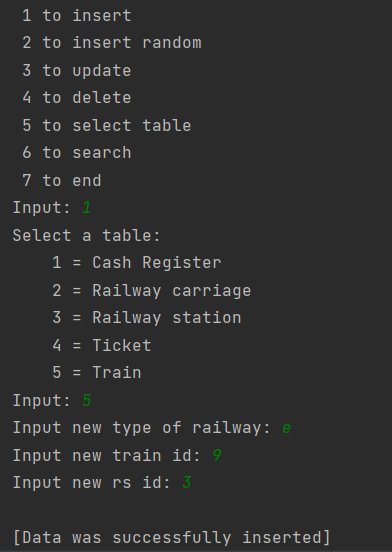
# Результати виконання операції вставки запису в дочірню таблицю:

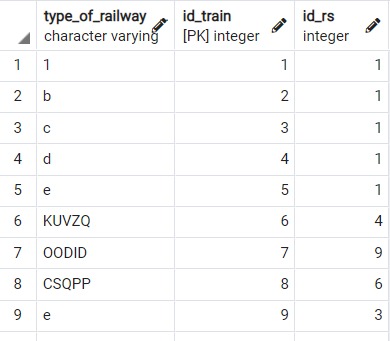
В таблицю Railway carriage:





В таблицю Train:





Неможливість запису(в батьківській таблиці немає відповідного запису):

B Railway carriage:



В таблицю Cash Register:



# Ілюстрації фрагментів згенерованих таблиць з відповідними SQL- запитами:

# В Railway station:

# 

# 

# В Train:

# 

# 

# Ілюстрації результатів пошукових запитів з відповідними SQL- запитами:

ВRailway station:

Пошук за id\_rs:

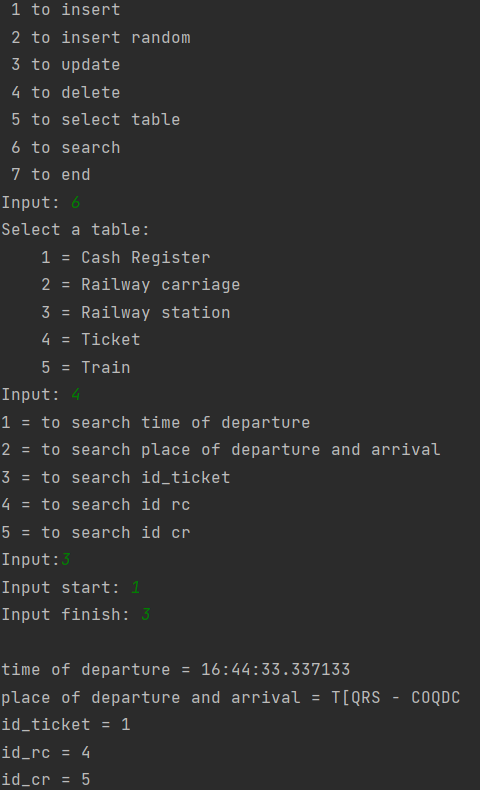
# 

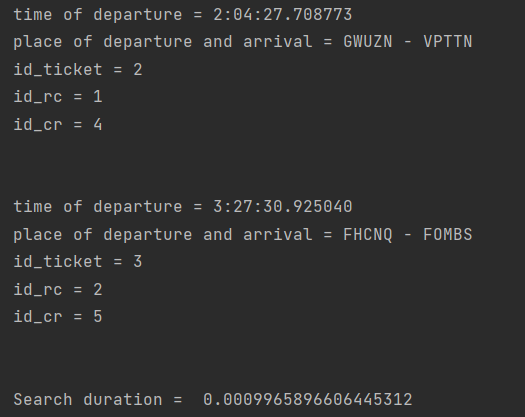
# 

# 

ВTicket:

Пошук за id\_ticket:

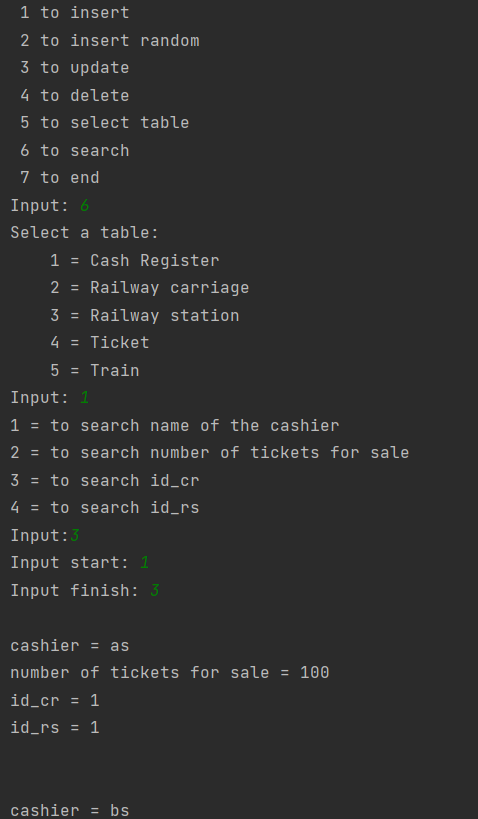


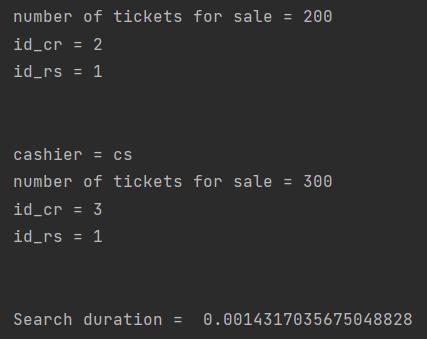




ВCash Register:

Пошук за id\_cr:

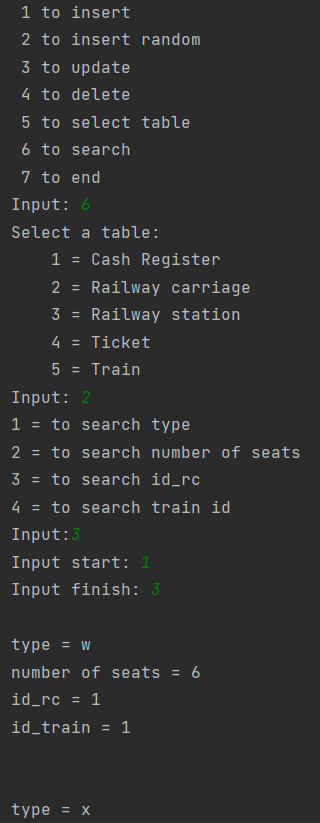


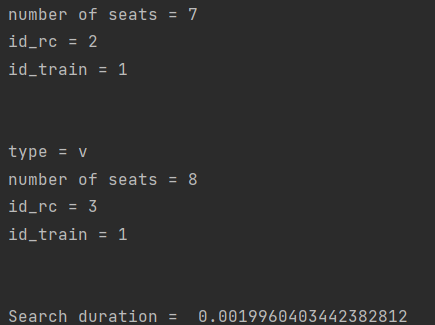




ВRailway carriage:

Пошук за id\_rc:

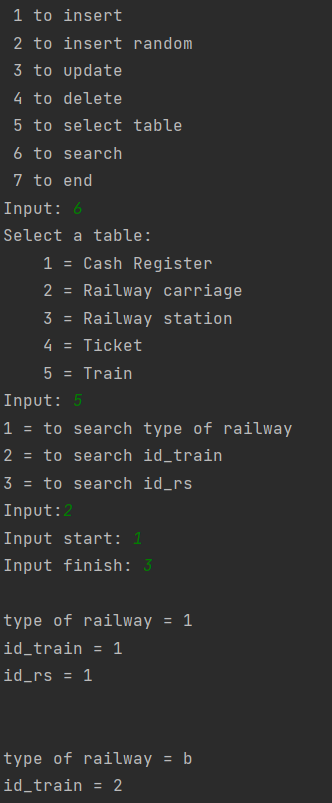


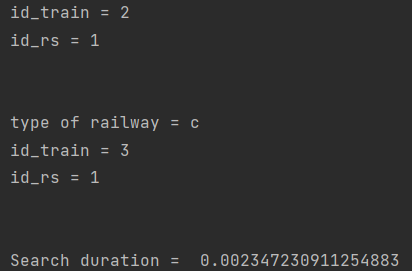




ВTrain:

Пошук за id\_train:







**Ілюстрації програмного коду модуля “Model”:**

**Функція insert:**

Забезпечує можливість введення даних до таблиць баз даних.

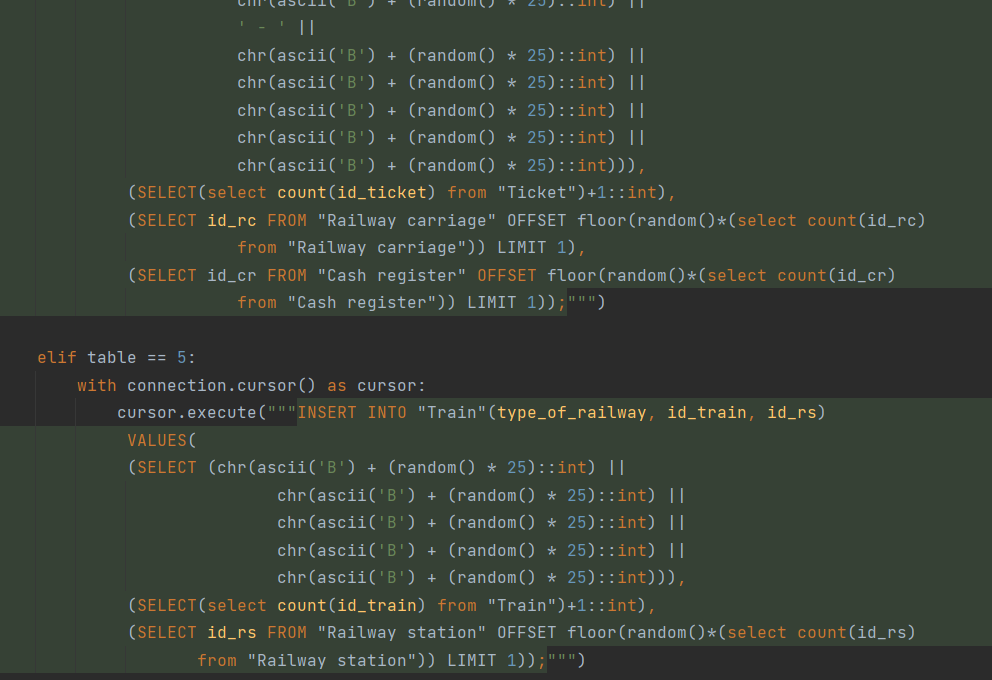
****

**Функція insert\_rand:**

Забезпечує можливість заповнення таблиць баз даних випадковими даними.



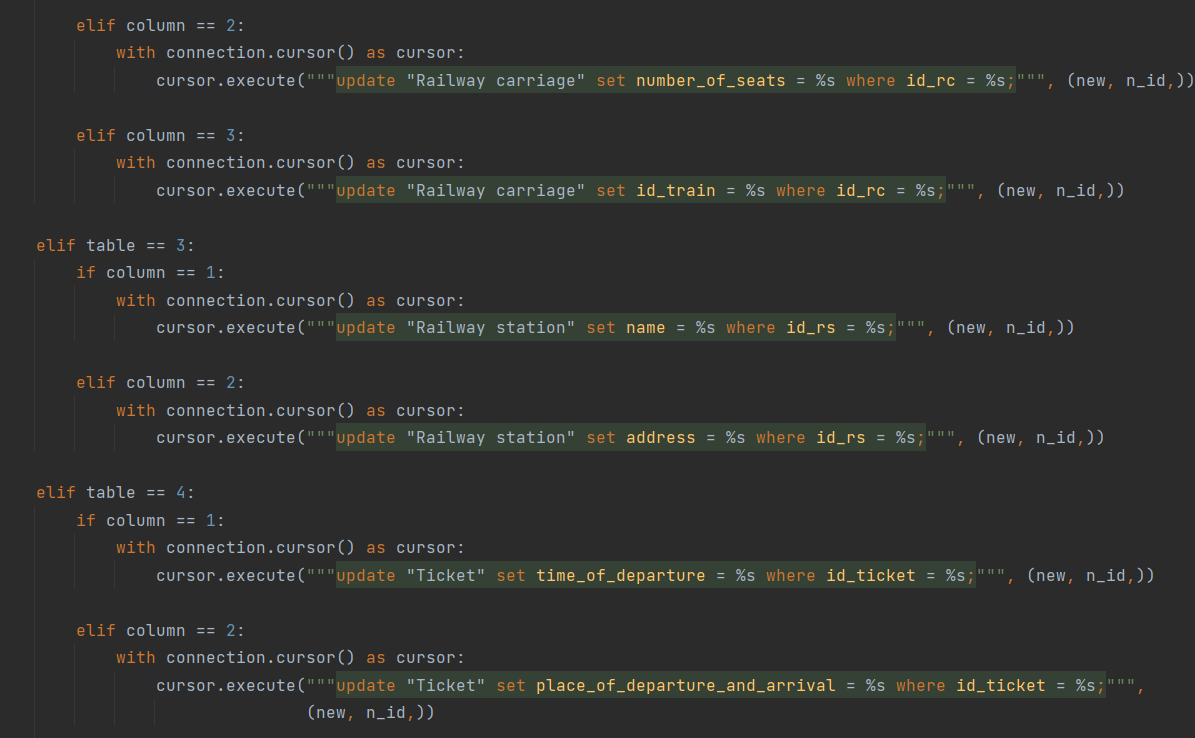


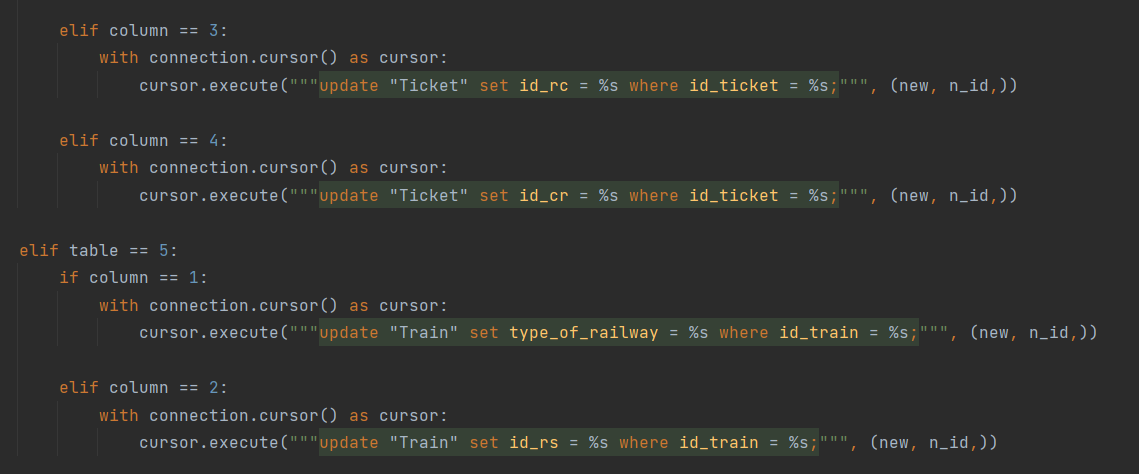


**Функція update:**

Забезпечує можливість оновлення потрібних даних в таблиці баз даних.

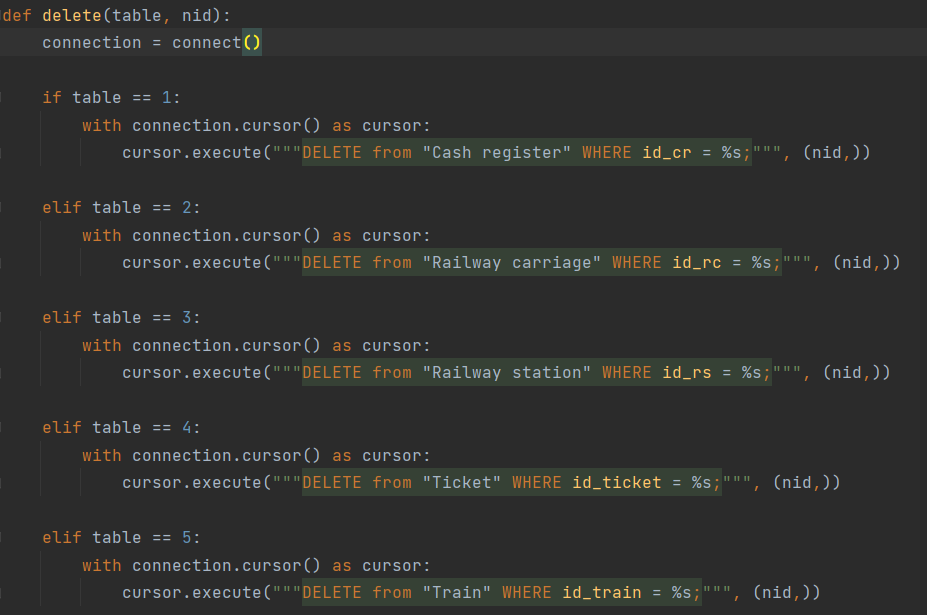






**Функція delete:**

Забезпечує можливість видалення потрібних таблиць в таблиці баз даних.

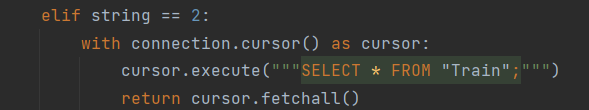
****

**Функція select\_table:**

Забезпечує можливість виведення потрібних таблиць баз даних, де ми можемо вивести, як всі таблиці, так і одну.

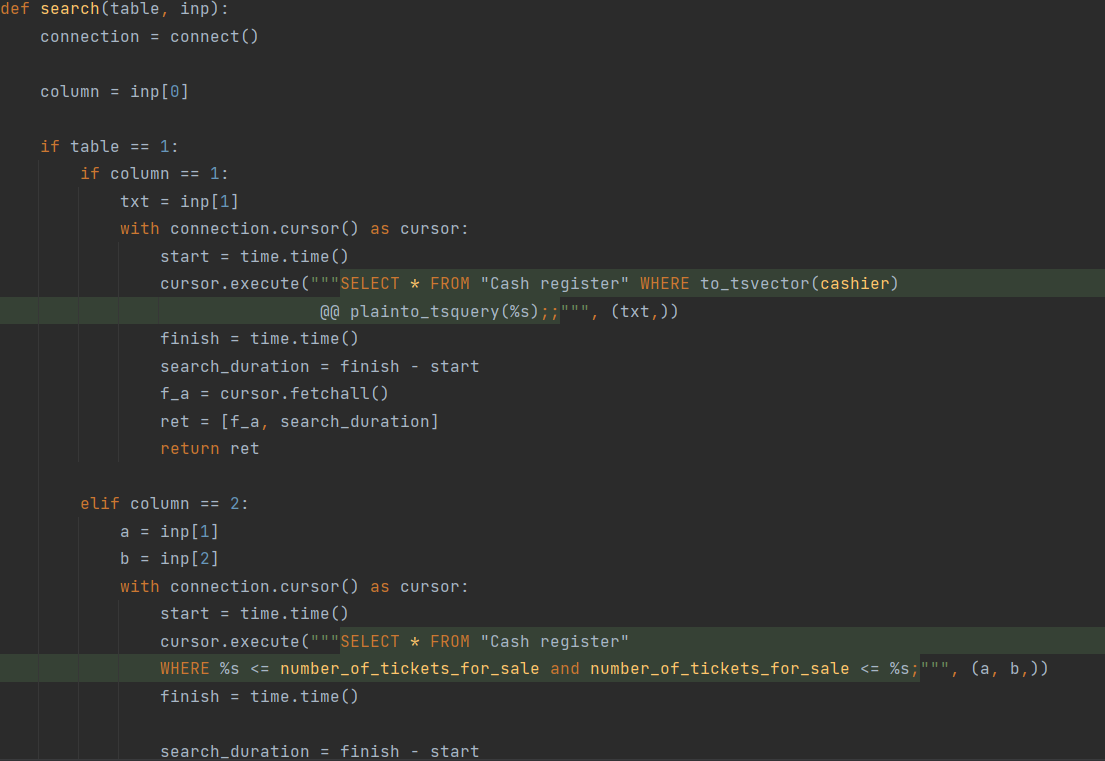


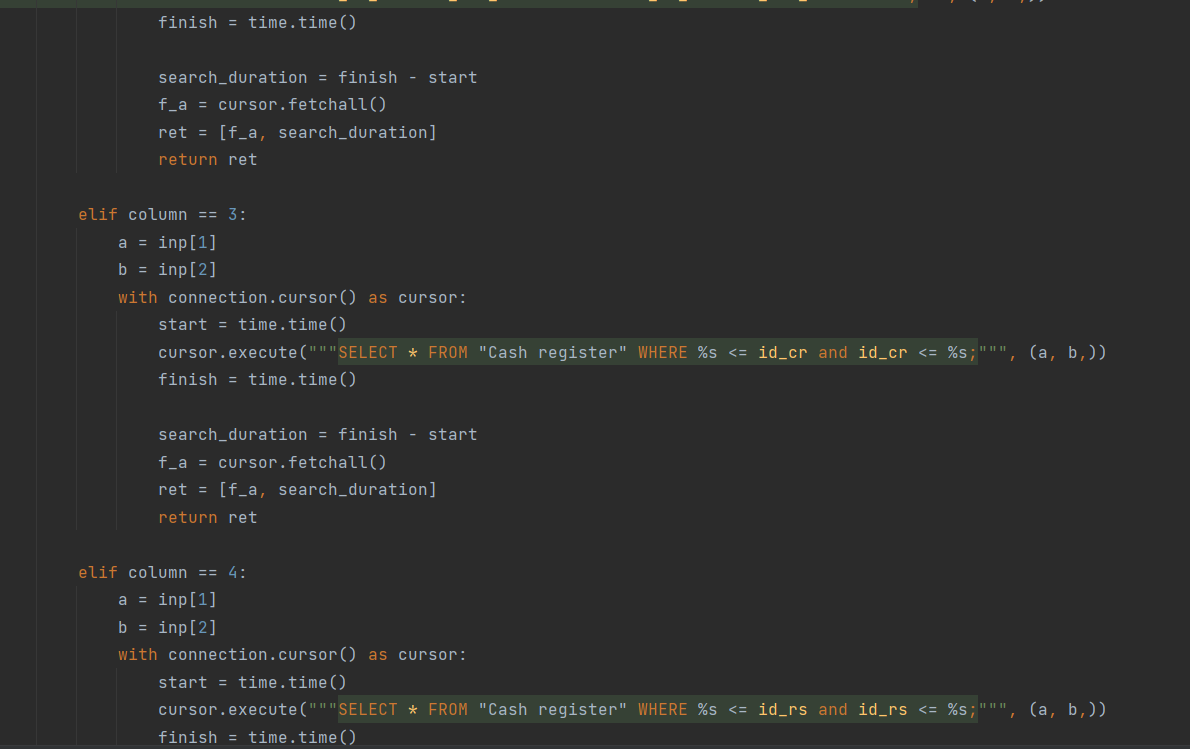


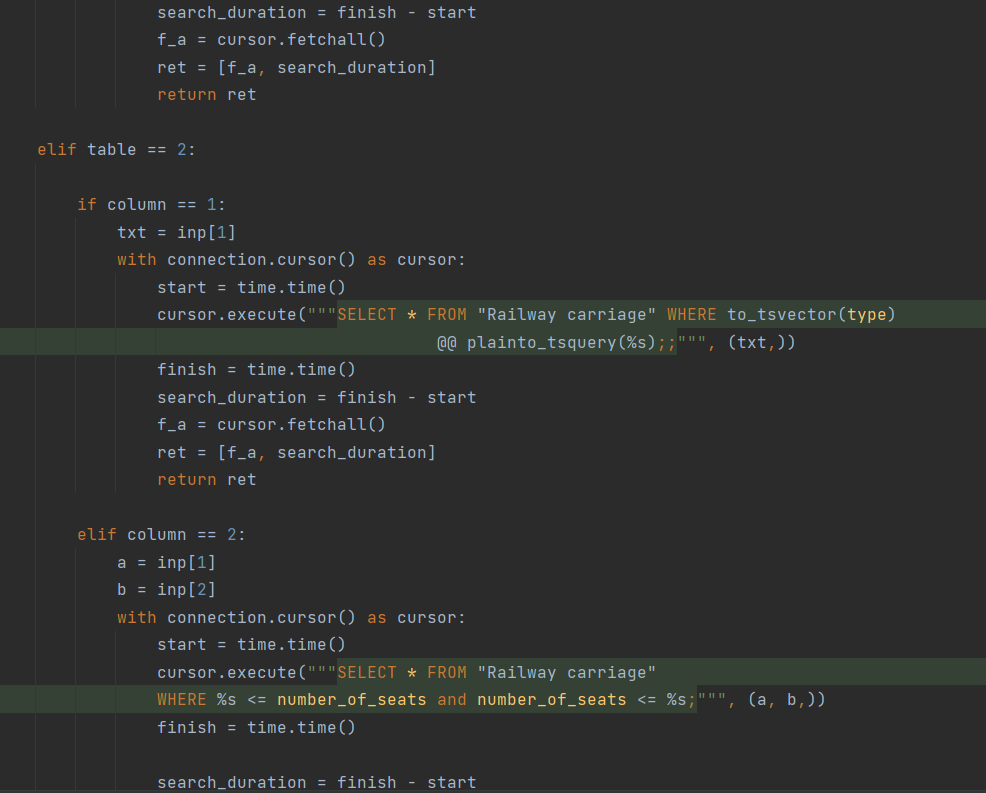


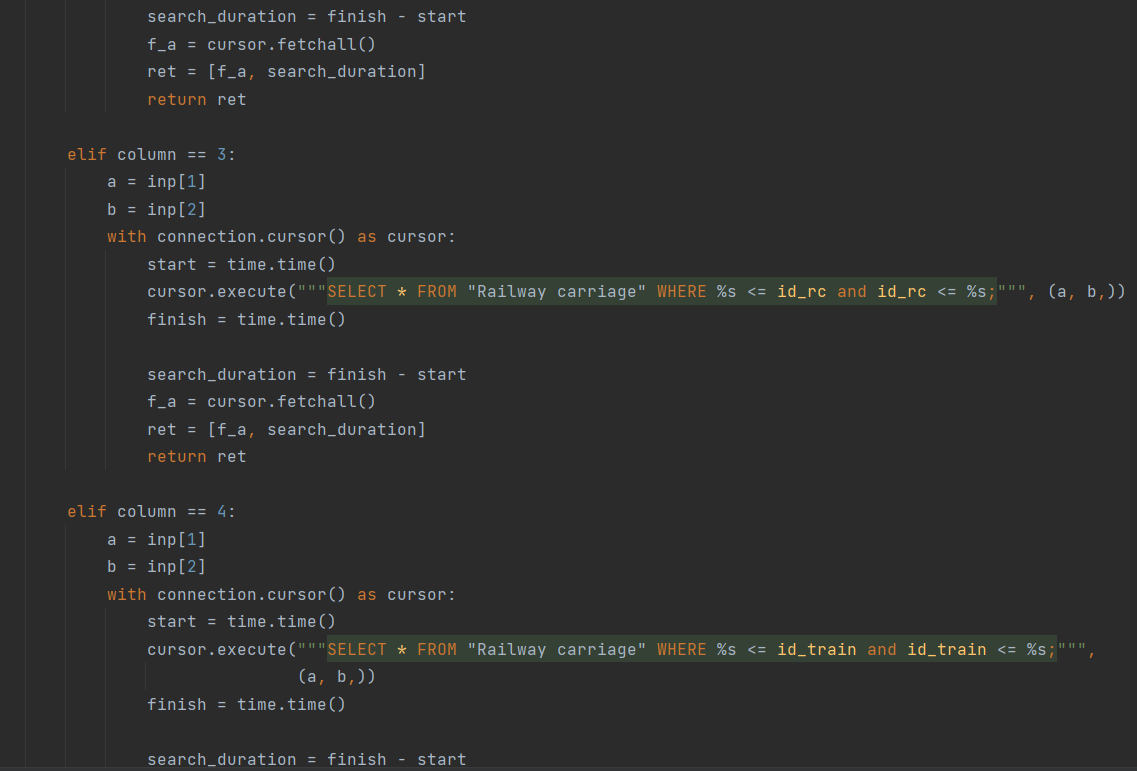
**Функція search:**

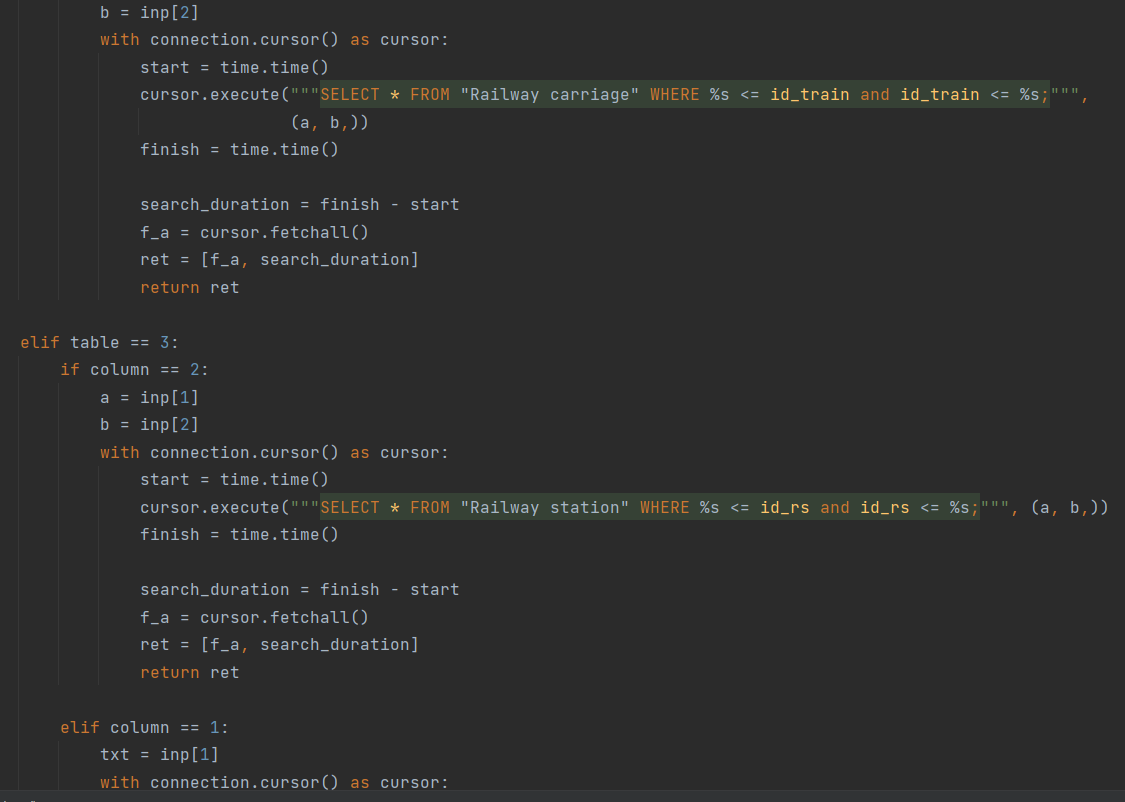
Забезпечує можливість пошуку потрібних даних по всі базі даних.

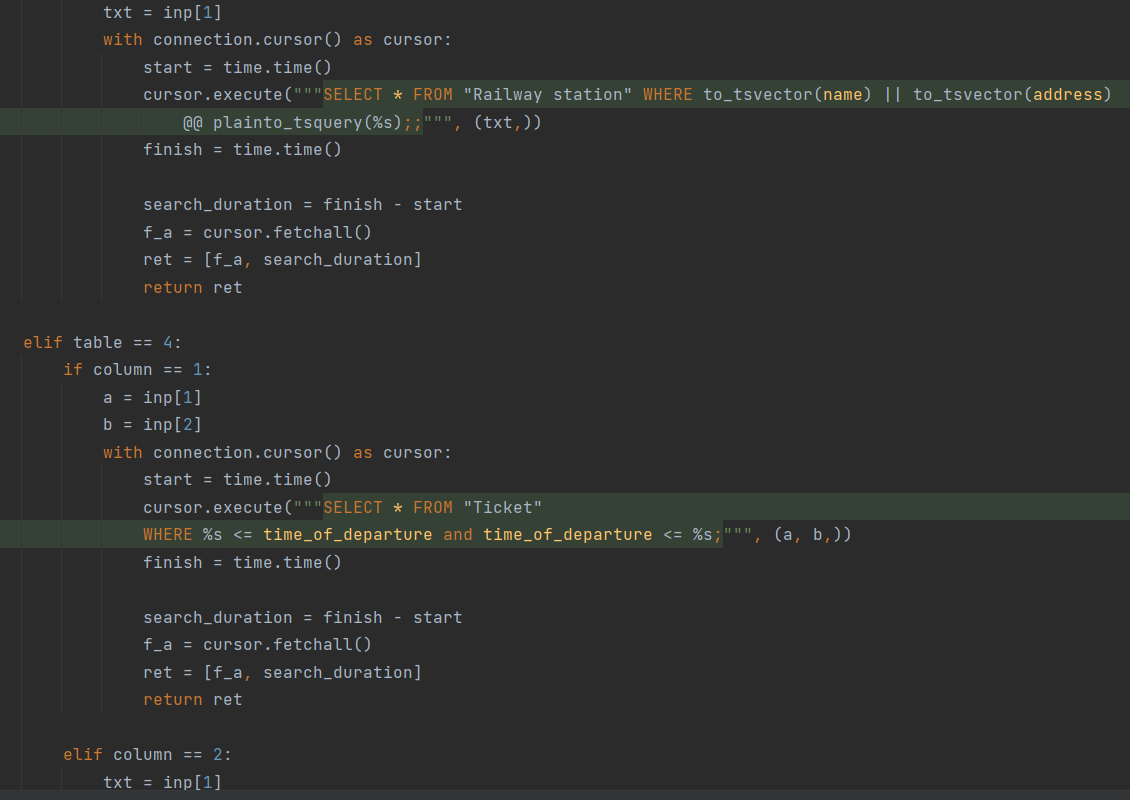
****

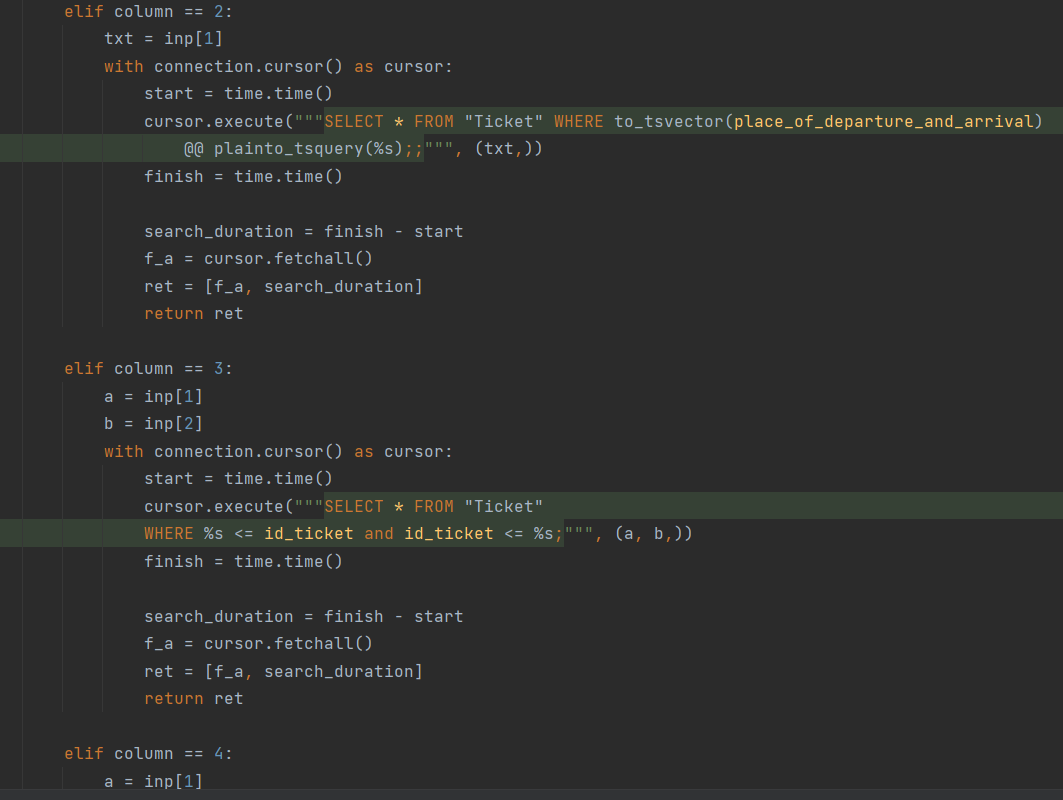
****

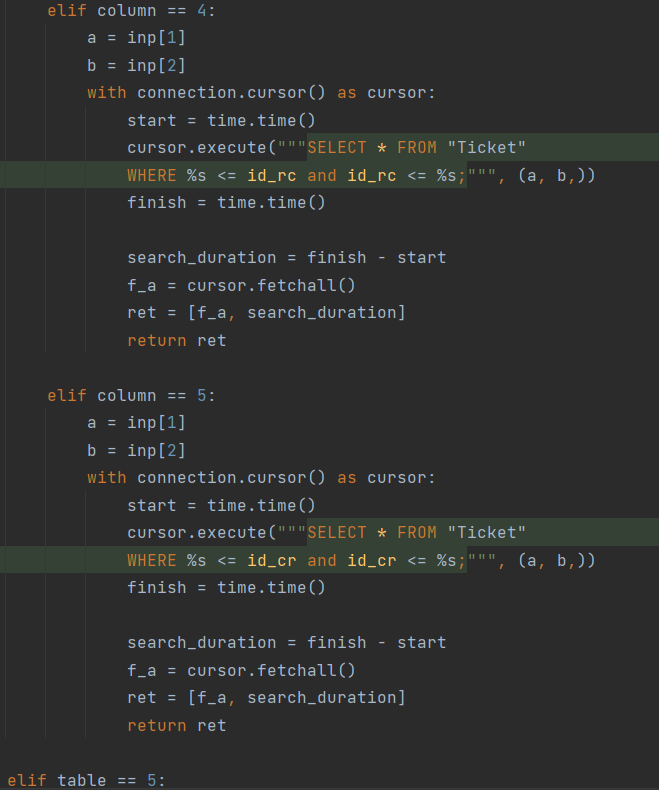
****

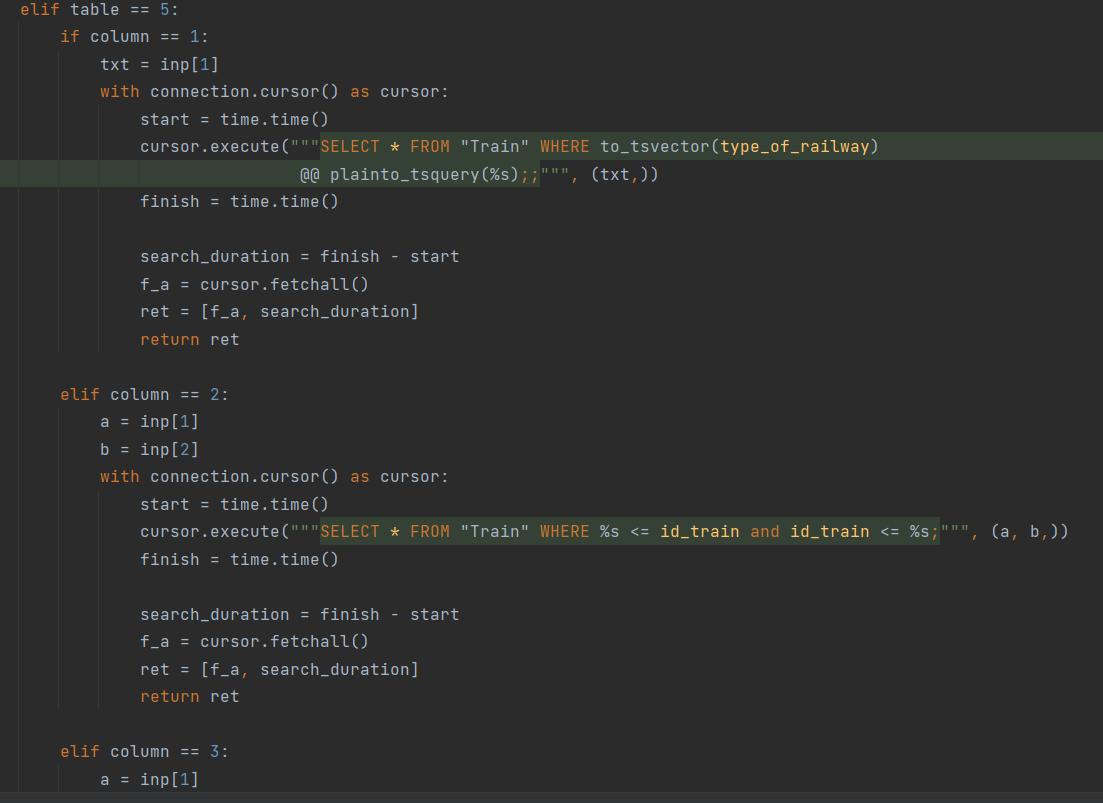
****

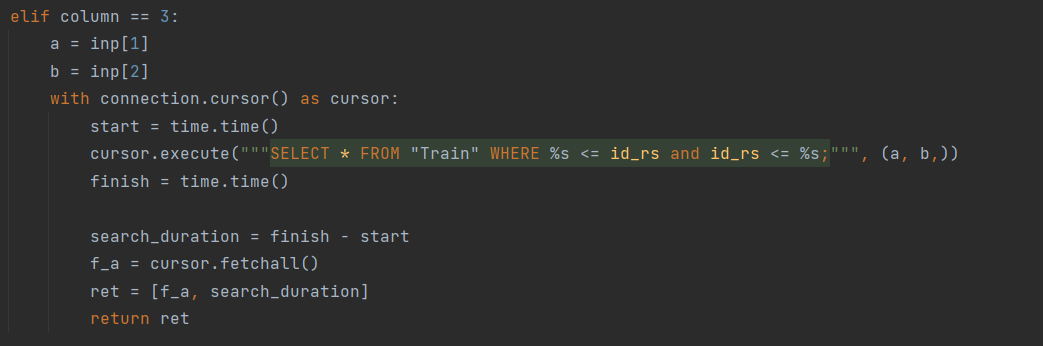
****

****

****

****

****

****

# Модель «сутність-зв’язок»:

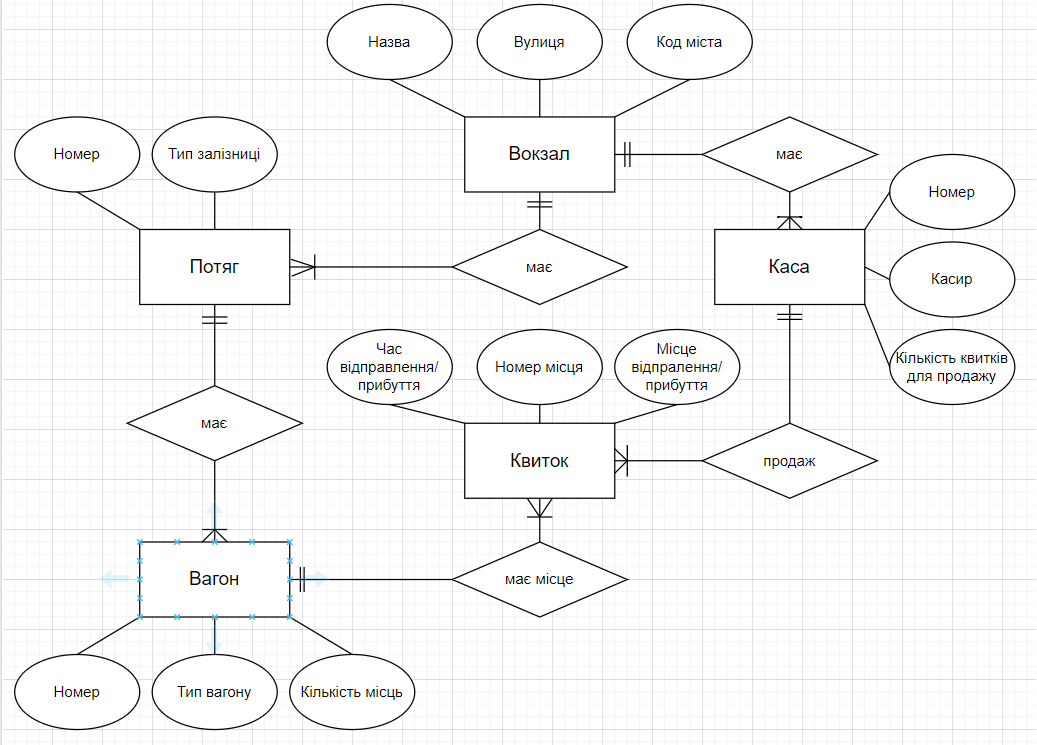
****

Рис1. ER-діаграма за нотацією Чена

**Схеми бази даних:**

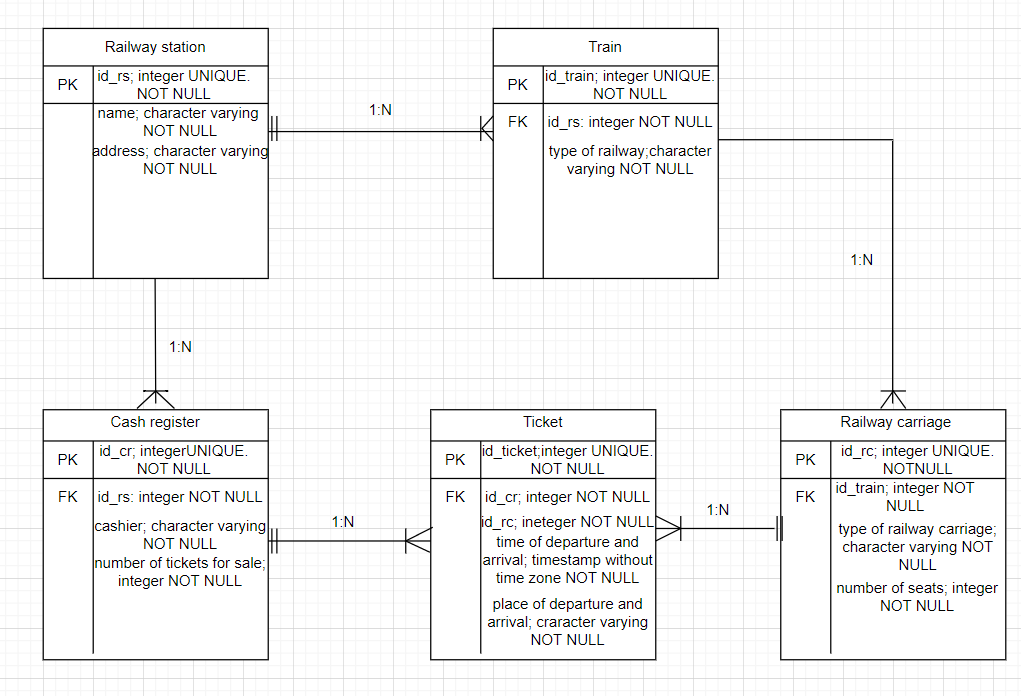


Рис2. Схема бази даних

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| **Railway station** – містить інформацію про вокзал | **id\_rs** – унікальний ідентифікатор вокзалу  **name** – назва вокзалу  **address** – адреса вокзалу | **integer**  **character varying**  **character varying** |
| **Cash register** – містить інформацію про касу | **id\_cr** – унікальний ідентифікатор каси  **id\_rs** – ідентифікатор вокзалу в якому знаходить каса  **cashier** – касир, який знаходиться на касі  **number of tickets for sale** – кількість квитків для продажу | **integer**  **integer**  **character varying**  **integer** |
| **Train** – містить у собі дані потягів | **id\_train** – унікальний ідентифікатор потягу  **id\_rs** – ідентифікатор вокзалу в якому знаходиться цей потяг  **type of railway** – тип залізниці, у формуванні якої знаходиться цей потяг | **integer**  **integer**  **character varying** |
| **Railway carriage** – містить дані про сеанс на фільм | **id\_rc** – унікальний ідентифікатор вагона  **id\_train** – ідентифікатор потяга, у составі якого знаходиться цей вагон  **type of railway carriage** – тип вагона | **integer**  **integer**  **character varying** |
| **Ticket** – містить дані про ваше місце в потягу | **id\_ticket** – унікальний ідентифікатор квитка  **id\_rc** – ідентифікатор вагона, у якому знаходиться ваше місце згідно з купленим квитком  **id\_cr** – ідентифікатор каси, у якій було куплено цей квиток  **time of departure and arrival** – час відбуття і прибуття  **place of departure and arrival** – місце відбуття і прибуття | **integer**  **integer**  **integer**  **timestamp without time zone**  **character varying** |

**Опис зв’язків:**

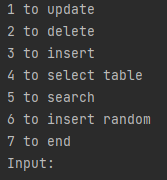
У Вокзала є багато Кас та Потягів, тому між сутностями Вокзал і Каса та сутностями Вокзал і Потяг зв’язок 1:N.

Потяг має декілька вагонів, тому між сутностями Потяг і Вагон зв’язок 1:N.

У касі людина має можливість купити один, декілька або будь-яку кількість квитків, яка менша або дорівнює кількості доступних до продажу квитків, тому сутності Квиток і Каса мають зв’язок 1:N.

У вагон людина має можливість купити один, декілька або будь-яку кількість квитків, яка менша або дорівнює кількості доступних до продажу квитків у цьому вагоні, тому сутності Квиток і Вагон мають зв’язок 1:N.

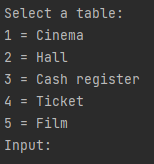
# Опис меню:

Головне меню:

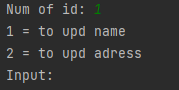
1 – оновлення даних в вибраній таблиці 2 – видалення даних в вибраній таблиці 3 – введення даних в вибрану таблицю 4 – вивід вибраної таблиці

5 – пошук по таблицями бази даних 6 – ввід випадкових даних у таблицю 7 – вихід з меню

Після вибору одного з пунктів 1 – 6, ми побачимо наступне меню для вибору таблиці бази даних:



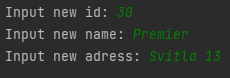
Якщо, в першому меню ми вибрали пункт 1 та в другому вибрали таблицю, то побачимо:

 меню для вибору даних для оновлення.

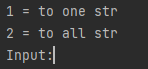
Якщо, в першому меню ми вибрали пункт 2 та в другому вибрали таблицю, то побачимо:

 меню де вводимо id для видалення рядка.

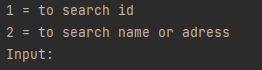
Якщо, в першому меню ми вибрали пункт 3 та в другому вибрали таблицю, то побачимо:

меню для вводу нових даних

Якщо, в першому меню ми вибрали пункт 4 та в другому вибрали таблицю, то побачимо:

меню для вибору вивести один рядок чи всі,  якщо, вибираємо 1 то вводимо id рядка для виводу.

Якщо, в першому меню ми вибрали пункт 5 та в другому вибрали таблицю, то побачимо:



меню для вибору атрибути, та пошуку

конкретних даних

Якщо, в першому меню ми вибрали пункт 6 та в другому вибрали таблицю, то побачимо:

* для таблиці 1 та 5



меню для вводу кількості даних для

генерації

* для інших таблиць буде проведена одна генерація

**Код програми**

**main.py**

from Controller import menu  
  
menu()

**Controler.py**

from Model import \*  
from View import \*  
  
  
def menu():  
 task = start\_menu()  
  
 if task == 1:  
 table = tables()  
 inp = input\_insert(table)  
 try:  
 insert(table, inp)  
 except Exception as \_ex:  
 err\_except(\_ex)  
 data(task)  
  
 elif task == 2:  
 table = tables()  
 if table == 3:  
 count = count\_rand()  
 try:  
 insert\_rand(table, count)  
 except Exception as \_ex:  
 err\_except(\_ex)  
 else:  
 try:  
 insert\_rand(table, None)  
 except Exception as \_ex:  
 err\_except(\_ex)  
 data(task)  
  
 elif task == 3:  
 table = tables()  
 inp = input\_update(table)  
 try:  
 update(table, inp)  
 except Exception as \_ex:  
 err\_except(\_ex)  
 data(task)  
  
 elif task == 4:  
 table = tables()  
 n\_id = del\_id()  
 try:  
 delete(table, n\_id)  
 except Exception as \_ex:  
 err\_except(\_ex)  
 data(task)  
  
 elif task == 5:  
 table = tables()  
 inp = input\_select()  
  
 try:  
 f\_t = select\_table(table, inp)  
 fetch(table, f\_t)  
 except Exception as \_ex:  
 err\_except(\_ex)  
  
 elif task == 6:  
 table = tables()  
 inp = input\_search(table)  
 try:  
 f\_t = search(table, inp)  
 fetch(table, f\_t[0])  
 search\_time(f\_t[1])  
 except Exception as \_ex:  
 err\_except(\_ex)  
  
 elif task == 7:  
 close()  
 print\_close()  
 return None  
  
 menu()

**Model.py**

import time  
import psycopg2  
  
  
def connect():  
 connection = psycopg2.connect(  
 database="Vlad",  
 user="postgres",  
 password="v2l7a0d9",  
 host="127.0.0.1")  
 connection.autocommit = True  
 return connection  
  
  
def close():  
 connection = connect()  
 connection.close()  
  
  
def insert(table, inp):  
 connection = connect()  
 if table == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Cash register"(cashier, number\_of\_tickets\_for\_sale, id\_cr, id\_rs)   
 VALUES(%s, %s, %s, %s);""", (inp[0], inp[1], inp[2], inp[3],))  
  
 elif table == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Railway carriage"(type, number\_of\_seats, id\_rc, id\_train)   
 VALUES(%s, %s, %s, %s);""", (inp[0], inp[1], inp[2], inp[3],))  
  
 elif table == 3:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Railway station"(name, address, id\_rs)   
 VALUES(%s, %s, %s);""", (inp[0], inp[1], inp[2],))  
  
 elif table == 4:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Ticket"(time\_of\_departure, place\_of\_departure\_and\_arrival,  
 id\_ticket, id\_rc, id\_cr) VALUES(%s, %s, %s, %s, %s);""", (inp[0], inp[1], inp[2], inp[3], inp[4],))  
  
 elif table == 5:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Train"(type\_of\_railway, id\_train, id\_rs)   
 VALUES(%s, %s, %s);""", (inp[0], inp[1], inp[2],))  
  
  
def insert\_rand(table, count):  
 connection = connect()  
 if table == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Cash register"(cashier, number\_of\_tickets\_for\_sale, id\_cr, id\_rs)  
 VALUES( (SELECT (chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int))),  
 trunc(random()\*100+200)::int,  
 (SELECT(select count(id\_cr) from "Cash register")+1::int),  
 (SELECT id\_rs FROM "Railway station" OFFSET floor(random()\*(select count(id\_rs)   
 from "Railway station")) LIMIT 1));""")  
  
 elif table == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Railway carriage"(type, number\_of\_seats, id\_rc, id\_train)  
 VALUES(   
 (SELECT (chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int))),  
 trunc(random()\*100+200)::int,  
 (SELECT(select count(id\_rc) from "Railway carriage")+1::int),  
 (SELECT id\_train FROM "Train" OFFSET floor(random()\*(select count(id\_train)   
 from "Train")) LIMIT 1));""")  
  
 elif table == 3:  
 with connection.cursor() as cursor:  
 for i in range(0, count):  
 cursor.execute("""INSERT INTO "Railway station"(name, address, id\_rs)   
 VALUES(  
 (SELECT (chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int))),  
 (SELECT (chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 ' st.' || (trunc(random()\*100)::int))),  
 (SELECT(select count(id\_rs) from "Railway station")+1::int));""")  
  
 elif table == 4:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Ticket"(time\_of\_departure, place\_of\_departure\_and\_arrival,  
 id\_ticket, id\_rc, id\_cr) VALUES(   
 (select (random() \* (interval '21 hour')) + '2 hour'),  
 (SELECT (chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 ' - ' ||   
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int))),  
 (SELECT(select count(id\_ticket) from "Ticket")+1::int),  
 (SELECT id\_rc FROM "Railway carriage" OFFSET floor(random()\*(select count(id\_rc)   
 from "Railway carriage")) LIMIT 1),  
 (SELECT id\_cr FROM "Cash register" OFFSET floor(random()\*(select count(id\_cr)   
 from "Cash register")) LIMIT 1));""")  
  
 elif table == 5:  
 with connection.cursor() as cursor:  
 cursor.execute("""INSERT INTO "Train"(type\_of\_railway, id\_train, id\_rs)   
 VALUES(   
 (SELECT (chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int) ||  
 chr(ascii('B') + (random() \* 25)::int))),  
 (SELECT(select count(id\_train) from "Train")+1::int),  
 (SELECT id\_rs FROM "Railway station" OFFSET floor(random()\*(select count(id\_rs)   
 from "Railway station")) LIMIT 1));""")  
  
  
def update(table, inp):  
 connection = connect()  
  
 n\_id = inp[0]  
 column = inp[1]  
 new = inp[2]  
  
 if table == 1:  
 if column == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Cash register" set cashier = %s where id\_cr = %s;""", (new, n\_id,))  
  
 elif column == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Cash register" set number\_of\_tickets\_for\_sale = %s where id\_cr = %s;""",  
 (new, n\_id,))  
  
 elif column == 3:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Cash register" set id\_rs = %s where id\_cr = %s;""", (new, n\_id,))  
  
 elif table == 2:  
 if column == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Railway carriage" set type = %s where id\_rc = %s;""", (new, n\_id,))  
  
 elif column == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Railway carriage" set number\_of\_seats = %s where id\_rc = %s;""", (new, n\_id,))  
  
 elif column == 3:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Railway carriage" set id\_train = %s where id\_rc = %s;""", (new, n\_id,))  
  
 elif table == 3:  
 if column == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Railway station" set name = %s where id\_rs = %s;""", (new, n\_id,))  
  
 elif column == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Railway station" set address = %s where id\_rs = %s;""", (new, n\_id,))  
  
 elif table == 4:  
 if column == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Ticket" set time\_of\_departure = %s where id\_ticket = %s;""", (new, n\_id,))  
  
 elif column == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Ticket" set place\_of\_departure\_and\_arrival = %s where id\_ticket = %s;""",  
 (new, n\_id,))  
  
 elif column == 3:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Ticket" set id\_rc = %s where id\_ticket = %s;""", (new, n\_id,))  
  
 elif column == 4:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Ticket" set id\_cr = %s where id\_ticket = %s;""", (new, n\_id,))  
  
 elif table == 5:  
 if column == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Train" set type\_of\_railway = %s where id\_train = %s;""", (new, n\_id,))  
  
 elif column == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""update "Train" set id\_rs = %s where id\_train = %s;""", (new, n\_id,))  
  
  
def delete(table, nid):  
 connection = connect()  
  
 if table == 1:  
 with connection.cursor() as cursor:  
 cursor.execute("""DELETE from "Cash register" WHERE id\_cr = %s;""", (nid,))  
  
 elif table == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""DELETE from "Railway carriage" WHERE id\_rc = %s;""", (nid,))  
  
 elif table == 3:  
 with connection.cursor() as cursor:  
 cursor.execute("""DELETE from "Railway station" WHERE id\_rs = %s;""", (nid,))  
  
 elif table == 4:  
 with connection.cursor() as cursor:  
 cursor.execute("""DELETE from "Ticket" WHERE id\_ticket = %s;""", (nid,))  
  
 elif table == 5:  
 with connection.cursor() as cursor:  
 cursor.execute("""DELETE from "Train" WHERE id\_train = %s;""", (nid,))  
  
  
def select\_table(table, inp):  
 connection = connect()  
  
 string = inp[0]  
  
 if table == 1:  
 if string == 1:  
 n\_id = inp[1]  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Cash register" where id\_cr = %s;""", (n\_id,))  
 return cursor.fetchmany(1)  
  
 elif string == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Cash register";""")  
 return cursor.fetchall()  
  
 elif table == 2:  
 if string == 1:  
 n\_id = inp[1]  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Railway carriage" where id\_rc = %s;""", (n\_id,))  
 return cursor.fetchmany(1)  
  
 elif string == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Railway carriage";""")  
 return cursor.fetchall()  
  
 elif table == 3:  
 if string == 1:  
 n\_id = inp[1]  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Railway station" where id\_rs = %s;""", (n\_id,))  
 return cursor.fetchmany(1)  
  
 elif string == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Railway station";""")  
 return cursor.fetchall()  
  
 elif table == 4:  
 if string == 1:  
 n\_id = inp[1]  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Ticket" where id\_ticket = %s;""", (n\_id,))  
 return cursor.fetchmany(1)  
  
 elif string == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Ticket";""")  
 return cursor.fetchall()  
  
 elif table == 5:  
 if string == 1:  
 n\_id = inp[1]  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Train" where id\_train = %s;""", (n\_id,))  
 return cursor.fetchmany(1)  
  
 elif string == 2:  
 with connection.cursor() as cursor:  
 cursor.execute("""SELECT \* FROM "Train";""")  
 return cursor.fetchall()  
  
  
def search(table, inp):  
 connection = connect()  
  
 column = inp[0]  
  
 if table == 1:  
 if column == 1:  
 txt = inp[1]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Cash register" WHERE to\_tsvector(cashier)  
 @@ plainto\_tsquery(%s);;""", (txt,))  
 finish = time.time()  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 2:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Cash register"   
 WHERE %s <= number\_of\_tickets\_for\_sale and number\_of\_tickets\_for\_sale <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 3:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Cash register" WHERE %s <= id\_cr and id\_cr <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 4:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Cash register" WHERE %s <= id\_rs and id\_rs <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif table == 2:  
  
 if column == 1:  
 txt = inp[1]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Railway carriage" WHERE to\_tsvector(type)  
 @@ plainto\_tsquery(%s);;""", (txt,))  
 finish = time.time()  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 2:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Railway carriage"   
 WHERE %s <= number\_of\_seats and number\_of\_seats <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 3:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Railway carriage" WHERE %s <= id\_rc and id\_rc <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 4:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Railway carriage" WHERE %s <= id\_train and id\_train <= %s;""",  
 (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif table == 3:  
 if column == 2:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Railway station" WHERE %s <= id\_rs and id\_rs <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 1:  
 txt = inp[1]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Railway station" WHERE to\_tsvector(name) || to\_tsvector(address)  
 @@ plainto\_tsquery(%s);;""", (txt,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif table == 4:  
 if column == 1:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Ticket"   
 WHERE %s <= time\_of\_departure and time\_of\_departure <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 2:  
 txt = inp[1]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Ticket" WHERE to\_tsvector(place\_of\_departure\_and\_arrival)  
 @@ plainto\_tsquery(%s);;""", (txt,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 3:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Ticket"   
 WHERE %s <= id\_ticket and id\_ticket <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 4:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Ticket"   
 WHERE %s <= id\_rc and id\_rc <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 5:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Ticket"   
 WHERE %s <= id\_cr and id\_cr <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif table == 5:  
 if column == 1:  
 txt = inp[1]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Train" WHERE to\_tsvector(type\_of\_railway)  
 @@ plainto\_tsquery(%s);;""", (txt,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 2:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Train" WHERE %s <= id\_train and id\_train <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret  
  
 elif column == 3:  
 a = inp[1]  
 b = inp[2]  
 with connection.cursor() as cursor:  
 start = time.time()  
 cursor.execute("""SELECT \* FROM "Train" WHERE %s <= id\_rs and id\_rs <= %s;""", (a, b,))  
 finish = time.time()  
  
 search\_duration = finish - start  
 f\_a = cursor.fetchall()  
 ret = [f\_a, search\_duration]  
 return ret

**View.py**

import datetime  
  
  
def start\_menu():  
 sm = int(input(" 1 to insert \n"  
 " 2 to insert random \n"  
 " 3 to update \n"  
 " 4 to delete \n"  
 " 5 to select table \n"  
 " 6 to search \n"  
 " 7 to end \n"  
 "Input: "))  
 if sm < 1 or sm > 7:  
 err()  
 start\_menu()  
 else:  
  
 return sm  
  
  
def tables():  
 choice\_table = int(input("Select a table:\n"  
 " 1 = Cash Register\n"  
 " 2 = Railway carriage\n"  
 " 3 = Railway station\n"  
 " 4 = Ticket\n"  
 " 5 = Train\n"  
 "Input: "))  
 if choice\_table < 1 or choice\_table > 5:  
 err()  
 tables()  
 else:  
 return choice\_table  
  
  
def input\_insert(table):  
 if table == 1:  
 name = str(input("Input new name of the cashier: "))  
 tickets = int(input("Input new number of tickets for sale: "))  
 id\_cr = int(input("Input new id\_cr: "))  
 id\_rs = int(input("Input new id\_rs: "))  
  
 inp = [name, tickets, id\_cr, id\_rs]  
 return inp  
  
 elif table == 2:  
 type\_rc = str(input("Input new type of RC: "))  
 num = int(input("Input new number of seats: "))  
 id\_rc = int(input("Input new id\_rc: "))  
 id\_train = int(input("Input new train id: "))  
  
 inp = [type\_rc, num, id\_rc, id\_train]  
 return inp  
  
 elif table == 3:  
  
 name = str(input("Input new name: "))  
 address = str(input("Input new address: "))  
 id\_rs = int(input("Input new rs id: "))  
  
 inp = [name, address, id\_rs]  
 return inp  
  
 elif table == 4:  
 time\_entry = input("Input a time of departure in HH:MM format: ")  
 hour, minute = map(int, time\_entry.split(':'))  
 dep = datetime.time(hour, minute)  
 place = str(input("Input new place of departure and arrival: "))  
 id\_tick = int(input("Input new ticket id: "))  
 id\_rc = int(input("Input new rc id: "))  
 id\_cr = int(input("Input new cr id: "))  
  
 inp = [dep, place, id\_tick, id\_rc, id\_cr]  
 return inp  
  
 elif table == 5:  
 type\_r = str(input("Input new type of railway: "))  
 id\_train = int(input("Input new train id: "))  
 id\_rs = int(input("Input new rs id: "))  
  
 inp = [type\_r, id\_train, id\_rs]  
 return inp  
  
  
def count\_rand():  
 count = int(input("Input amount of data to generate: "))  
 return count  
  
  
def input\_update(table):  
 new = None  
 if table == 1:  
 id\_cr = int(input("Num of cr id: "))  
 column = int(input("1 = to upd name of the cashier\n"  
 "2 = to upd number of tickets for sale\n"  
 "3 = to upd id\_rs\n"  
 "Input:"))  
  
 if column == 1:  
 new = str(input("New: "))  
 elif column == 3 or column == 2:  
 new = int(input("New: "))  
 else:  
 err()  
 input\_update(table)  
  
 inp = [id\_cr, column, new]  
 return inp  
  
 elif table == 2:  
 id\_rc = int(input("Num of rc id: "))  
 column = int(input("1 = to upd type\n"  
 "2 = to upd number of seats\n"  
 "3 = to upd train id\n"  
 "Input:"))  
  
 if column == 1:  
 new = str(input("New: "))  
 elif column == 3 or column == 2:  
 new = int(input("New: "))  
 else:  
 err()  
 input\_update(table)  
  
 inp = [id\_rc, column, new]  
 return inp  
  
 elif table == 3:  
 id\_rs = int(input("Num of rs id: "))  
 column = int(input("1 = to upd name\n"  
 "2 = to upd address\n"  
 "Input:"))  
  
 if column == 1 or column == 2:  
 new = str(input("New: "))  
 else:  
 err()  
 input\_update(table)  
  
 inp = [id\_rs, column, new]  
 return inp  
  
 elif table == 4:  
 id\_t = int(input("Num of ticket id: "))  
 column = int(input("1 = to upd time of departure\n"  
 "2 = to upd place of departure and arrival\n"  
 "3 = to upd id rc\n"  
 "4 = to upd id cr\n"  
 "Input:"))  
  
 if column == 1:  
 time\_entry = input("New time in HH:MM format: ")  
 hour, minute = map(int, time\_entry.split(':'))  
 new = datetime.time(hour, minute)  
 elif column == 2:  
 new = str(input("New: "))  
 elif column == 3 or column == 4:  
 new = int(input("New: "))  
 else:  
 err()  
 input\_update(table)  
  
 inp = [id\_t, column, new]  
 return inp  
  
 elif table == 5:  
 id\_cr = int(input("Num of train id: "))  
 column = int(input("1 = to upd type of railway\n"  
 "2 = to upd id\_rs\n"  
 "Input:"))  
  
 if column == 1:  
 new = str(input("New: "))  
 elif column == 2:  
 new = int(input("New: "))  
 else:  
 err()  
 input\_update(table)  
  
 inp = [id\_cr, column, new]  
 return inp  
  
  
def del\_id():  
 nid = int(input("Num of id: "))  
 return nid  
  
  
def input\_select():  
 string = int(input("1 = to one str\n"  
 "2 = to all str\n"  
 "Input:"))  
 if string == 1:  
 nid = int(input("Num of id: "))  
 inp = [string, nid]  
 return inp  
  
 elif string == 2:  
 inp = [string]  
 return inp  
  
 else:  
 err()  
 input\_select()  
  
  
def input\_search(table):  
 if table == 1:  
 column = int(input("1 = to search name of the cashier\n"  
 "2 = to search number of tickets for sale\n"  
 "3 = to search id\_cr\n"  
 "4 = to search id\_rs\n"  
 "Input:"))  
 if column == 2 or column == 3 or column == 4:  
 a = int(input("Input start: "))  
 b = int(input("Input finish: "))  
 inp = [column, a, b]  
 return inp  
  
 elif column == 1:  
 txt = str(input("Input text: "))  
 inp = [column, txt]  
 return inp  
  
 else:  
 err()  
 input\_search(table)  
  
 elif table == 2:  
 column = int(input("1 = to search type\n"  
 "2 = to search number of seats\n"  
 "3 = to search id\_rc\n"  
 "4 = to search train id\n"  
 "Input:"))  
 if column == 2 or column == 3 or column == 4:  
 a = int(input("Input start: "))  
 b = int(input("Input finish: "))  
 inp = [column, a, b]  
 return inp  
  
 elif column == 1:  
 txt = str(input("Input text: "))  
 inp = [column, txt]  
 return inp  
  
 else:  
 err()  
 input\_search(table)  
  
 elif table == 3:  
 column = int(input("1 = to search name or address\n"  
 "2 = to search id\_rs\n"  
 "Input:"))  
  
 if column == 1:  
 txt = str(input("Input text: "))  
 inp = [column, txt]  
 return inp  
 elif column == 2:  
 a = int(input("Input start: "))  
 b = int(input("Input finish: "))  
 inp = [column, a, b]  
 return inp  
 else:  
 err()  
 input\_search(table)  
  
 elif table == 4:  
 column = int(input("1 = to search time of departure\n"  
 "2 = to search place of departure and arrival\n"  
 "3 = to search id\_ticket\n"  
 "4 = to search id rc\n"  
 "5 = to search id cr\n"  
  
 "Input:"))  
 if column == 1:  
 time\_entry\_a = input("New start time in HH:MM format: ")  
 hour, minute = map(int, time\_entry\_a.split(':'))  
 a = datetime.time(hour, minute)  
  
 time\_entry\_b = input("New finish time in HH:MM format: ")  
 hour, minute = map(int, time\_entry\_b.split(':'))  
 b = datetime.time(hour, minute)  
  
 inp = [column, a, b]  
 return inp  
  
 elif column == 3 or column == 4 or column == 5:  
 a = int(input("Input start: "))  
 b = int(input("Input finish: "))  
 inp = [column, a, b]  
 return inp  
  
 elif column == 2:  
 txt = str(input("Input text: "))  
 inp = [column, txt]  
 return inp  
  
 else:  
 err()  
 input\_search(table)  
  
 elif table == 5:  
 column = int(input("1 = to search type of railway\n"  
 "2 = to search id\_train\n"  
 "3 = to search id\_rs\n"  
  
 "Input:"))  
 if column == 2 or column == 3:  
 a = int(input("Input start: "))  
 b = int(input("Input finish: "))  
 inp = [column, a, b]  
 return inp  
  
 elif column == 1:  
 txt = str(input("Input text: "))  
 inp = [column, txt]  
 return inp  
  
 else:  
 err()  
 input\_search(table)  
  
  
def fetch(table, f\_table):  
 if table == 1:  
 for i in f\_table:  
 print("\ncashier =", i[0])  
 print("number of tickets for sale =", i[1])  
 print("id\_cr =", i[2])  
 print("id\_rs =", i[3], "\n")  
  
 elif table == 2:  
 for i in f\_table:  
 print("\n""type =", i[0])  
 print("number of seats =", i[1])  
 print("id\_rc =", i[2])  
 print("id\_train =", i[3], "\n")  
  
 elif table == 3:  
 for i in f\_table:  
 print("\n""name =", i[0])  
 print("address =", i[1])  
 print("id\_rs =", i[2], "\n")  
  
 elif table == 4:  
 for i in f\_table:  
 print("\n""time of departure =", i[0])  
 print("place of departure and arrival =", i[1])  
 print("id\_ticket =", i[2])  
 print("id\_rc =", i[3])  
 print("id\_cr =", i[4], "\n")  
  
 elif table == 5:  
 for i in f\_table:  
 print("\n""type of railway =", i[0])  
 print("id\_train =", i[1])  
 print("id\_rs =", i[2], "\n")  
  
  
def data(task):  
 if task == 1:  
 print("\n[Data was successfully inserted]\n")  
 elif task == 2:  
 print("\n[Random Data was successfully inserted]\n")  
 elif task == 3:  
 print("\n[Data was successfully updated]\n")  
 elif task == 4:  
 print("\n[Data was successfully deleted]\n")  
  
  
def search\_time(t):  
 print("\nSearch duration = ", t, "\n")  
  
  
def err():  
 print("\n[Error input, try again!]\n")  
  
  
def print\_close():  
 print("\n[PostgreSQL connection closed]")  
  
  
def err\_except(\_ex):  
 print("\n[Error while working with PostgreSQL", \_ex, "]\n")

Мова програмування – Python 3.8

Середовище розробки програмного забезпечення – PyCharm Community Edition.

Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.

Використані бібліотеки:

psycopg – для роботи з PostgresSQL

time – для роботи з часом, а саму визначення швидкодії пошуку по базі

random – для роботи з рандомізацією, а саме вибір випадкової вулиці з масиву.