

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»

Виконав: студент 3 курсу

ФПМ групи КВ-82

Ященко Іван Васильович

Перевірив: Павловський В.І.

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 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 2-х та більше сутностей одночасно: для числових атрибутів – у рамок діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамок 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:
   1. контроль при введенні - валідація даних;
   2. 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. З боку батьківської таблиці необхідно контролювати **вилучення (ON DELETE)** рядків за умови наявності даних у підлеглій таблиці. З боку підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** до неї нових даних. Унеможливити виведення програмою на екрані системних помилок PostgreSQL шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

1. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не програмою**, а відповідним **SQL-запитом**! Кількість даних для генерування має вводити користувач з клавіатури.
2. Для реалізації багатокритеріального 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. Після виведення даних вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
3. Програмний код організувати згідно шаблону Model-ViewController (MVC). Приклад організації коду згідно шаблону доступний [за даним посиланням](https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/). Модель, подання (представлення) та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

**Модель бази даних**

На рис. 1 наведено логічну структуру бази даних яка використовується в даній роботі. Суттєвих змін, в порівнянні з 1 Лабораторною роботою не відбулося, єдине що змінювалося це режими ON DELETE, ON UPDATE для дослідження в останньому пункті

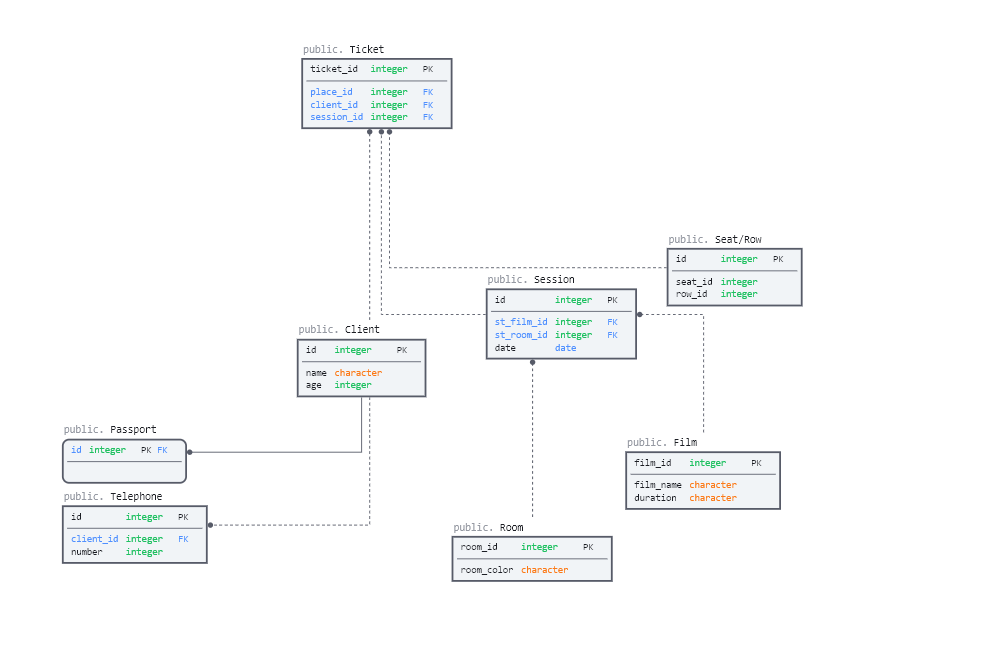


Рис. 1 – Логічна модель (Структура) БД “ Сервіс продажу квитків кіно ” (засобами SqlDMB)

**Середовище розробки та налаштування підключення до бази даних**

Для виконання лабораторної роботи використовувалась мова програмування C# та IDE Visual Studio 2019. Для підключення до серверу бази даних PostgreSQL використовувався пакет Npgsql.

**Підключення до БД**

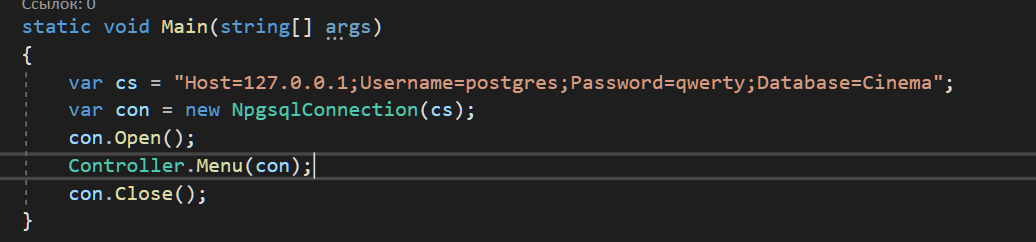


Рис 2. ‑ Фотографія екрану, де описано підключення до БД засобами Npgsql

**Структура програми**

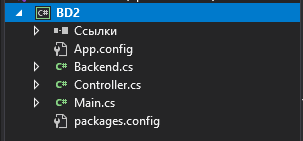


Рис 3. – Фотографія екрану, де описана структура програми

Опис структури програми. Програма містить 3 модуля, серед яких Backend – модуль, де описані усі взаємодії програми із базою даних (усі sql запити і її реалізація), Controller – модуль, де описана робота меню програми, Main – модуль, де розташована main функція, і де розпочинається робота програми.

**Структура меню програми**

Головне меню

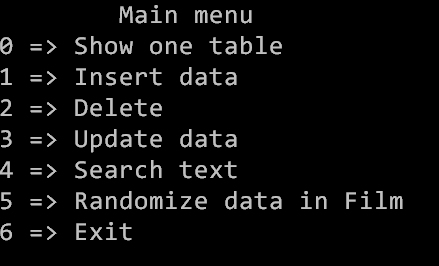


Рис 4. ­– Фотографія головного меню програми

Меню вибору таблиць

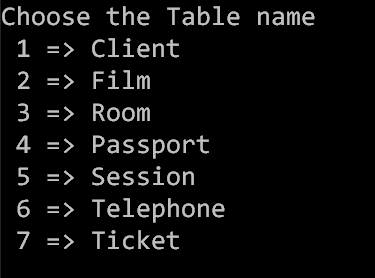


Рис 5. – Фотографія меню, яке з’являється якщо обрати пункти 0,1,2,3

Меню пошуку заготовлених динамічних запитів

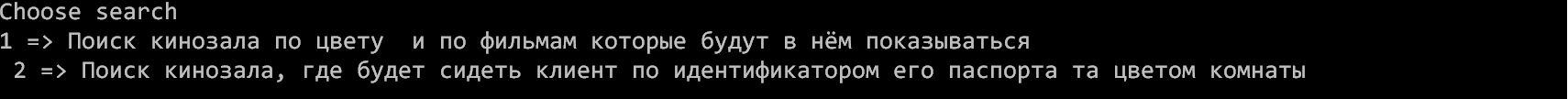


Рис 6. – Фотографія меню, яке з’являється якщо обрати пункт 4

Меню рандомізації даних

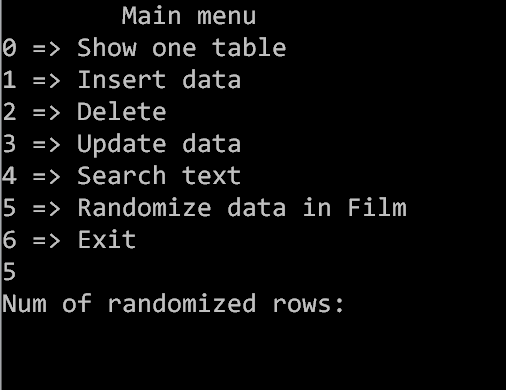
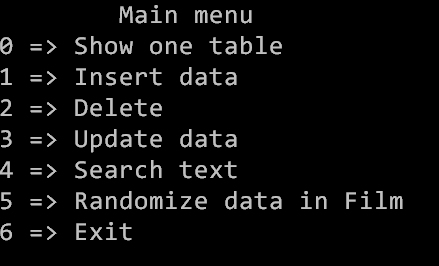
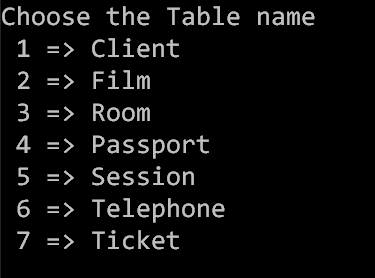
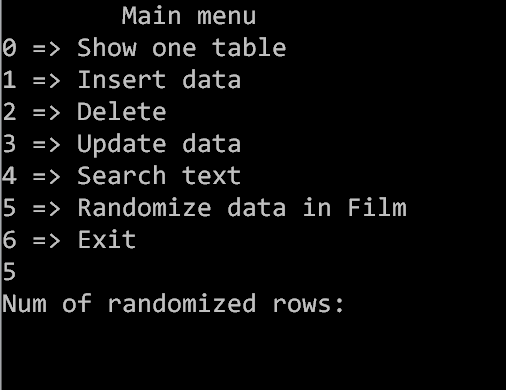


Рис 7. – Фотографія меню, яке з’являється якщо обрати пункт 5

Загальна структура меню



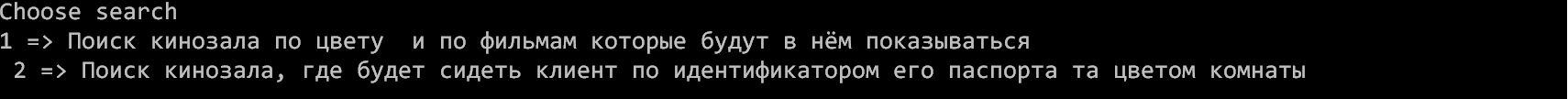


Рис 8. – Структура меню програми

[Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив 9](#_Toc55781009)

[Функції для внесення даних 9](#_Toc55781010)

[Функції для оновлення даних 17](#_Toc55781011)

[Функції для вилучення даних 22](#_Toc55781012)

[Лістинги програми з директивами внесення рандомізованих даних і виконання динамічних запитів у базі даних та результати виконання цих директив 28](#_Toc55781013)

[Рандомізоване внесення даних до таблиці «Film» 28](#_Toc55781014)

[Виконання динамічних запитів бази даних 31](#_Toc55781015)

[Обробка виняткових ситуацій (помилок) при введенні/вилученні та валідації даних 33](#_Toc55781016)

[Приклади перевірки на валідність вхідних даних 33](#_Toc55781018)

[Дослідження режимів обмеження ON DELETE 34](#_Toc55781019)

[Ілюстрації програмного коду на Github 37](#_Toc55781020)

**Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив**

## Функції внесення даних

Для кожної таблиці створений окремий метод, що приймає дані, що потрібні для внесення, і додає нові рядки у відповідну таблицю.

Функція внесення даних до таблиці Client та Passport

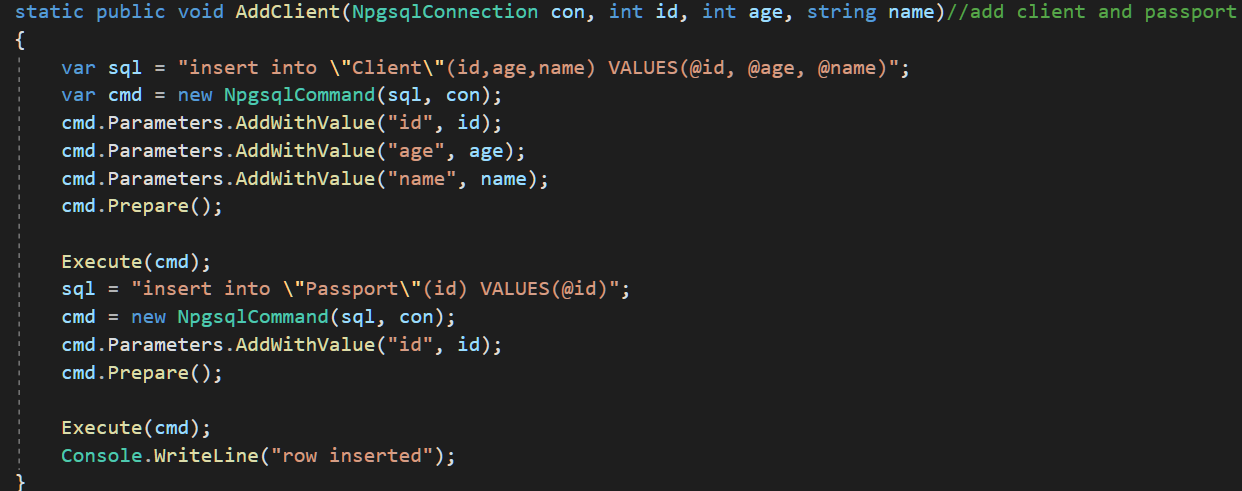
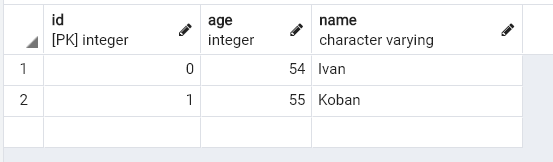
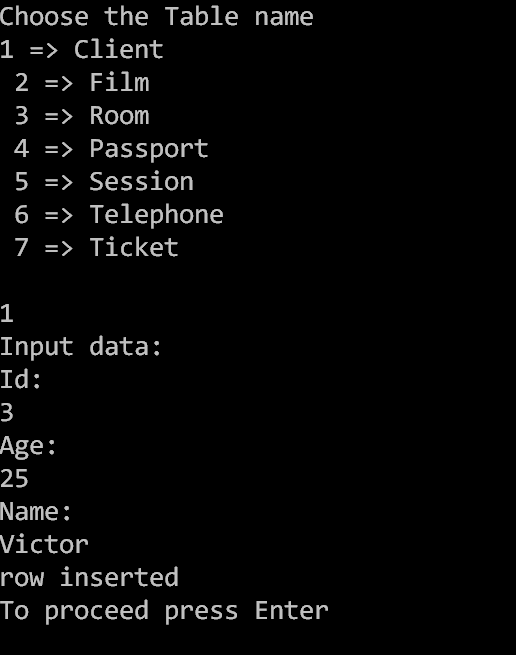


Рис 9. – Функція додавання даних до таблиці Client та Passport в файлі Backend.cs

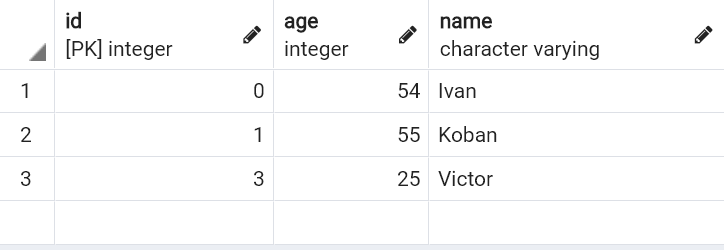
Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція внесення даних до таблиці Room

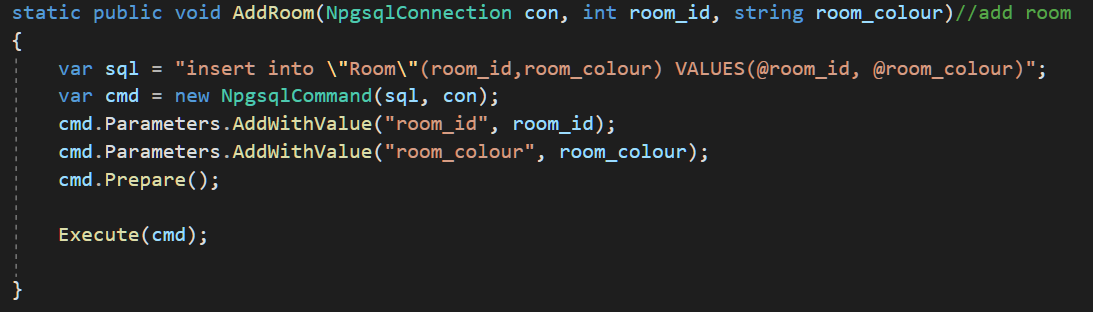
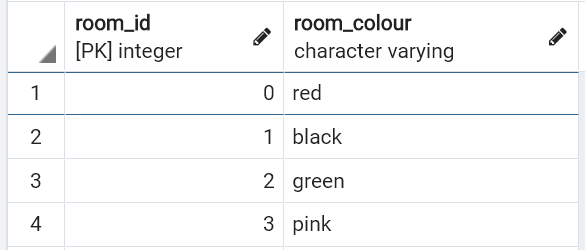
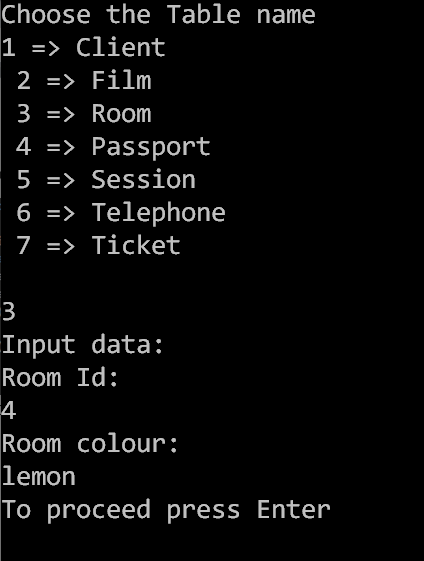


Рис 10. – Функція додавання даних до таблиці Room в файлі Backend.cs

Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція внесення даних до таблиці Film

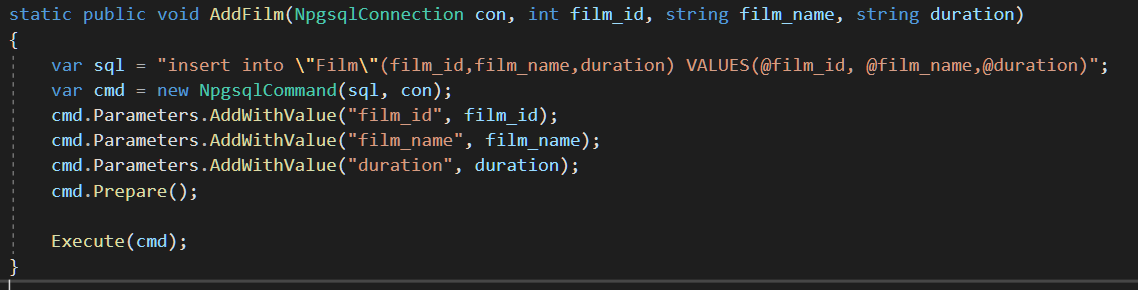
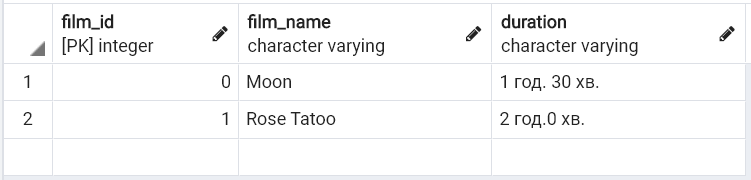
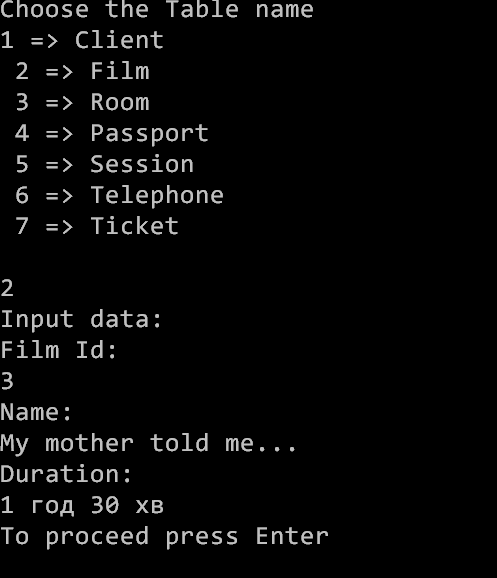


Рис 11. – Функція додавання даних до таблиці Room в файлі Backend.cs

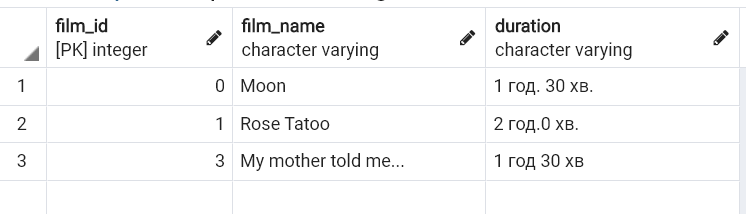
Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція внесення даних до таблиці Session

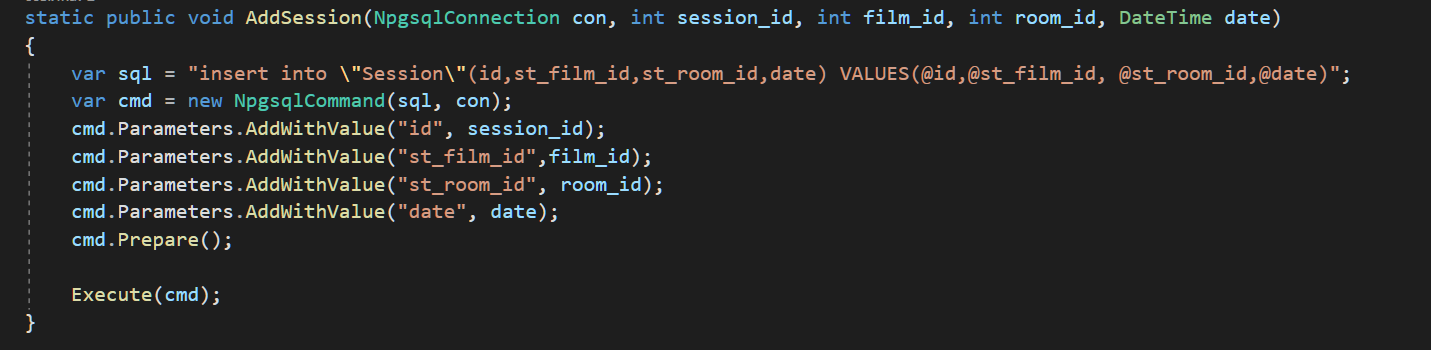
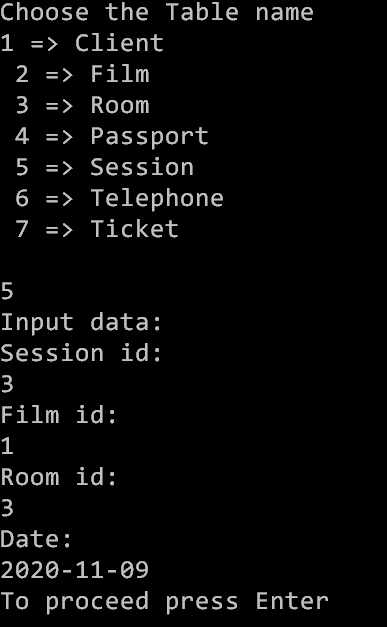


Рис 12. – Функція додавання даних до таблиці Session в файлі Backend.cs

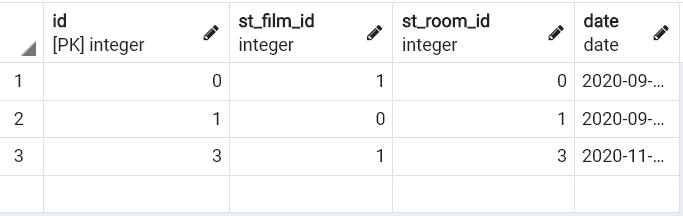
Початкова таблиця:



Запит у меню



Таблиця після запиту:



Аналогічно виглядають функції додавання для усіх інших таблиць

## Функції оновлення даних

Для кожної таблиці створений окремий метод, що приймає дані, що потрібні для внесення, і вносить зміни до бази даних.

Функція для оновлення даних до таблиці Client

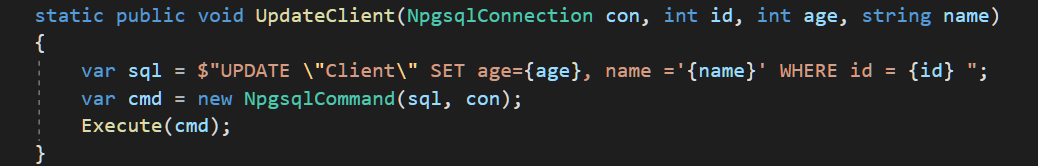
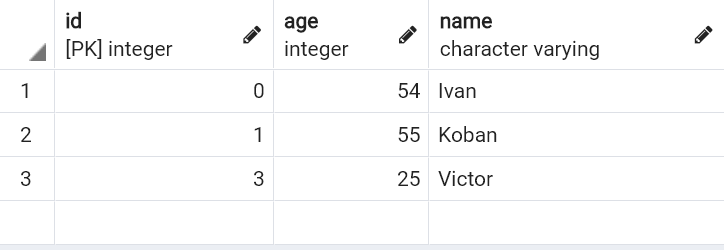
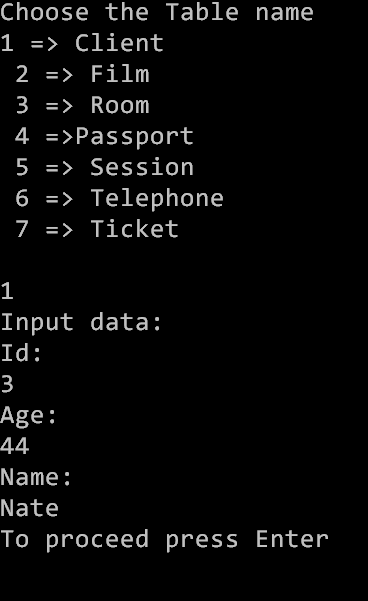


Рис 13. – Функція для оновлення даних до таблиці Client в файлі Backend.cs

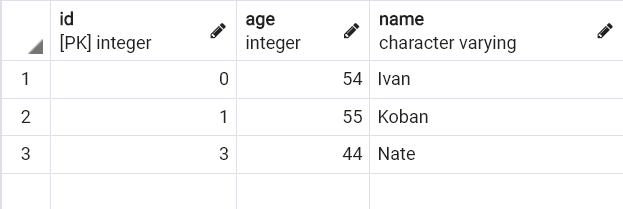
Початкова таблиця



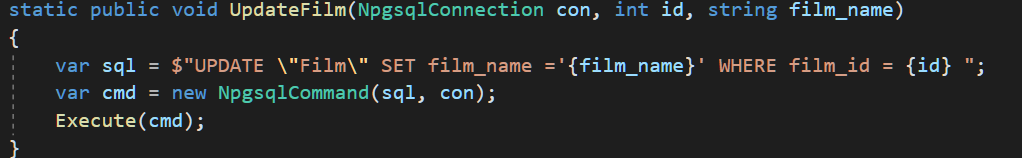
Запит у меню



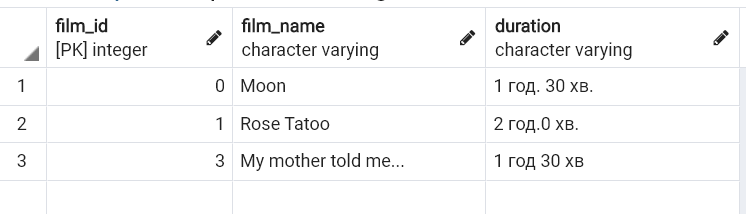
Таблиця після запиту:



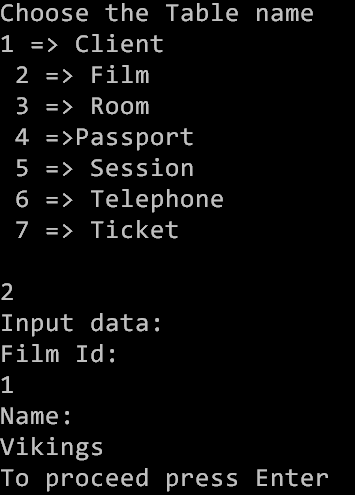
Функція оновлення даних до таблиці Film

Рис 14. – Функція оновлення даних до таблиці Film в файлі Backend.cs

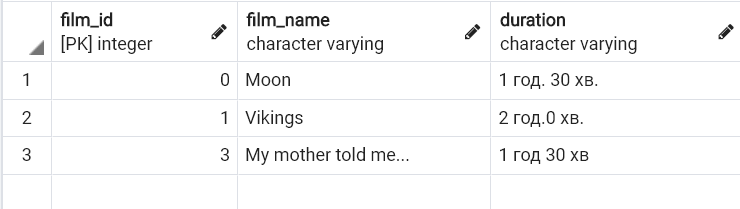
Початкова таблиця



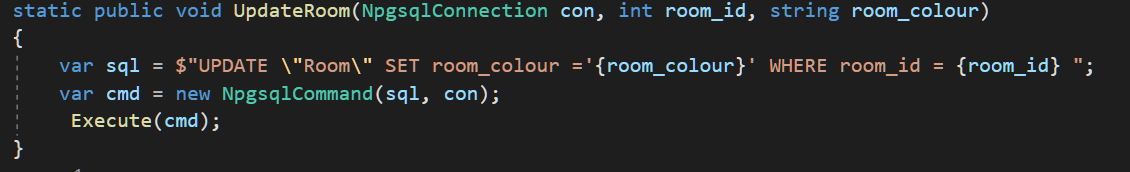
Запит у меню



Таблиця після запиту



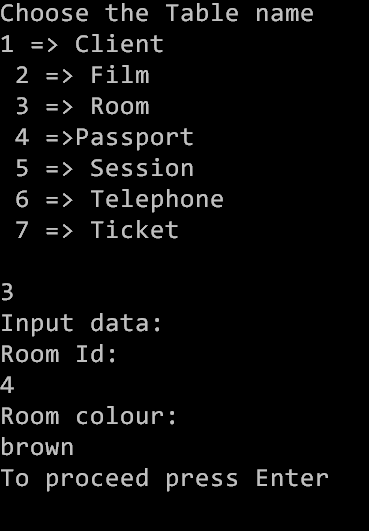
Функція оновлення даних до таблиці Room

Рис 15. – Функція оновлення даних до таблиці Room в файлі Backend.cs

Початкова таблиця



Запит у меню



Таблиця після запиту

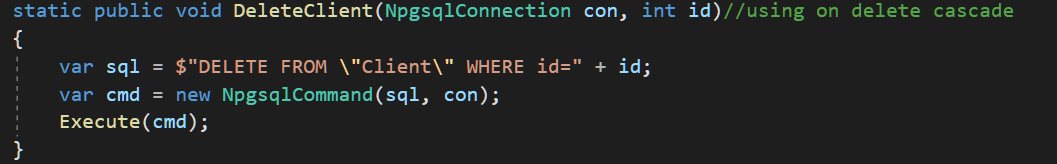


Аналогічно виглядають функції оновлення для усіх інших таблиць

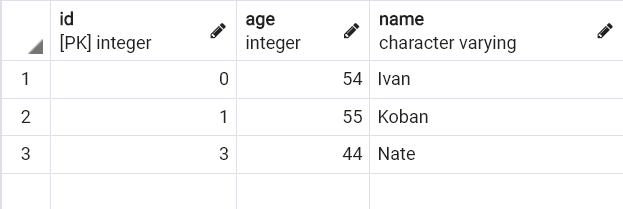
## Функції вилучення даних

Для кожної таблиці створений окремий метод, що приймає дані, що потрібні для видалення (id), і вносить зміни до бази даних.

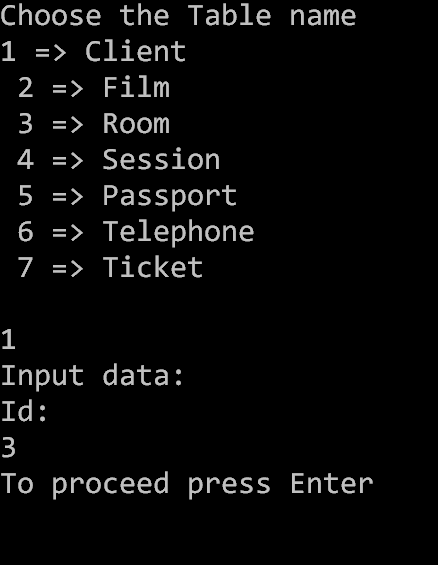
Функція для вилучення даних з таблиці Client

Рис 16. – Функція вилучення даних з таблиці Client в файлі Backend.cs

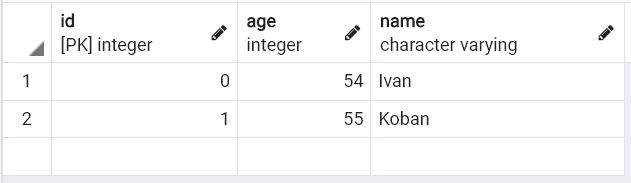
Початкова таблиця



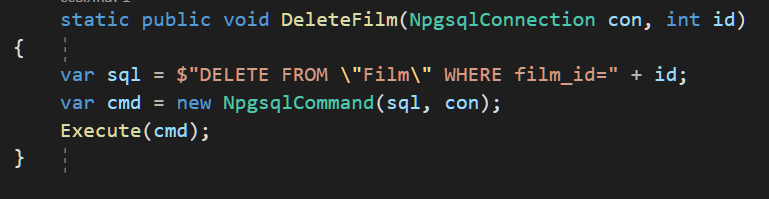
Запит у меню



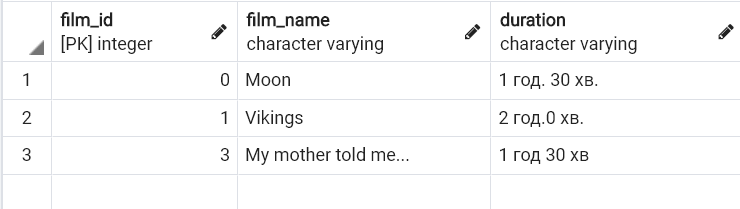
Таблиця після запиту



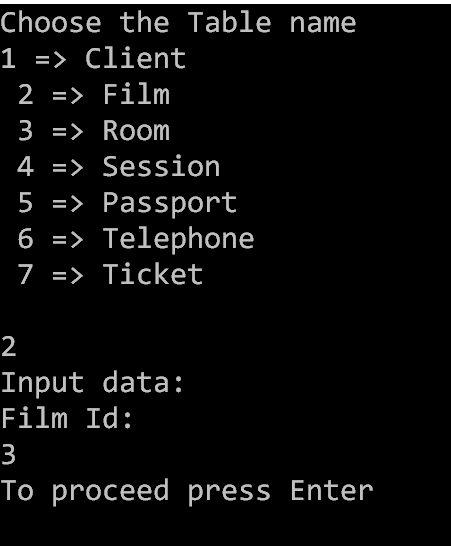
Функція вилучення даних з таблиці Film

Рис 17. – Функція вилучення даних з таблиці Film в файлі Backend.cs

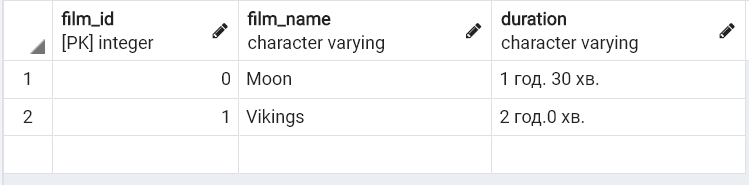
Початкова таблиця



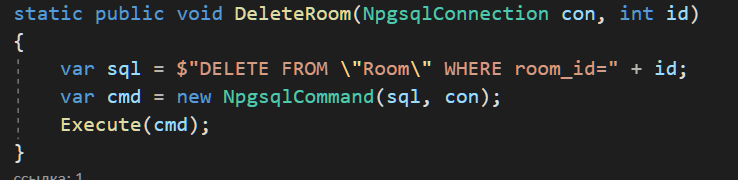
Запит у меню



Таблиця після запиту



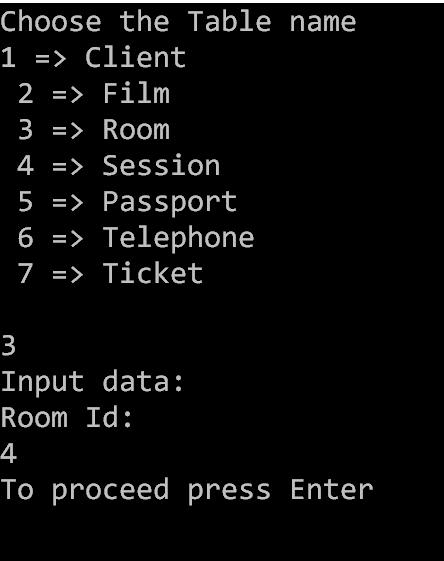
Функція вилучення даних з таблиці Room

Рис 18. – Функція вилучення даних з таблиці Room в файлі Backend.cs

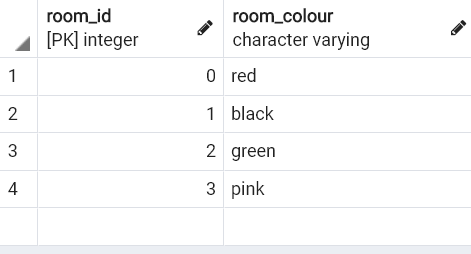
Початкова таблиця



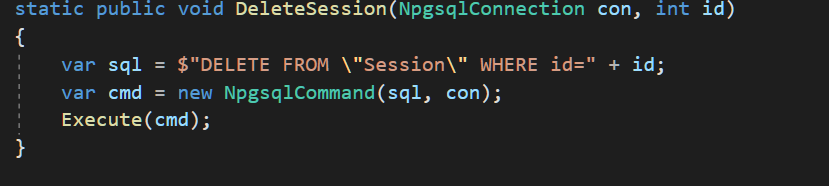
Запит у меню



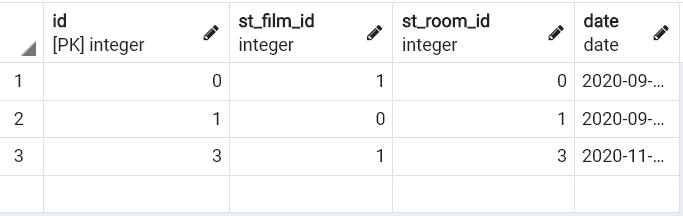
Таблиця після запиту



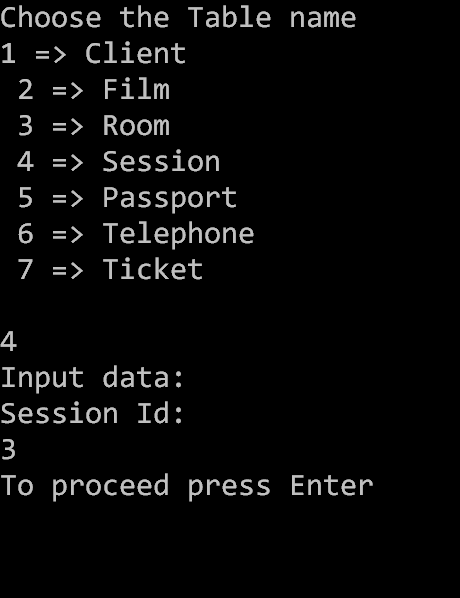
Функція вилучення даних з таблиці Session

Рис 19. – Функція вилучення даних з таблиці Session в файлі Backend.cs

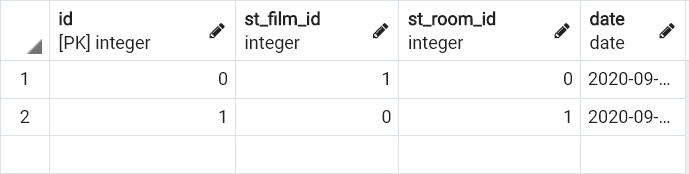
Початкова таблиця



Запит у меню



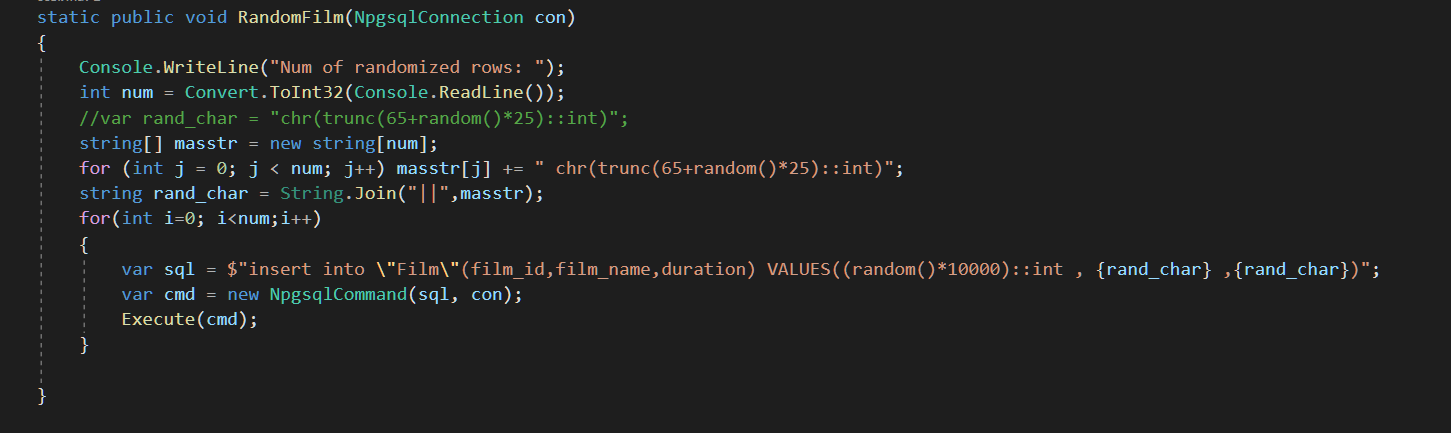
Таблиця після запиту



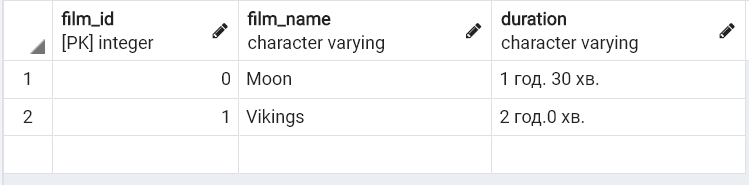
Аналогічно виглядають функції вилучення даних для усіх інших таблиць

# Лістинги програми з директивами внесення рандомізованих даних і виконання динамічних запитів у базі даних та результати виконання цих директив

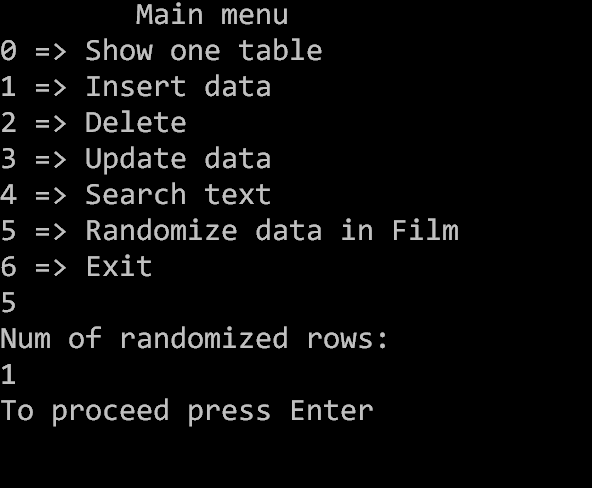
## Рандомізоване внесення даних до таблиці «Film»

Рис 20. – Фотографія коду програми для рандімізованого внесення данних в файлі Backend.cs

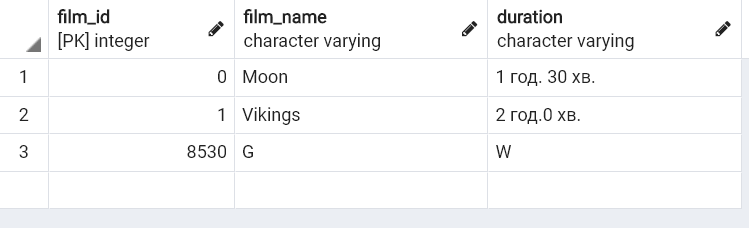
Початкова таблиця



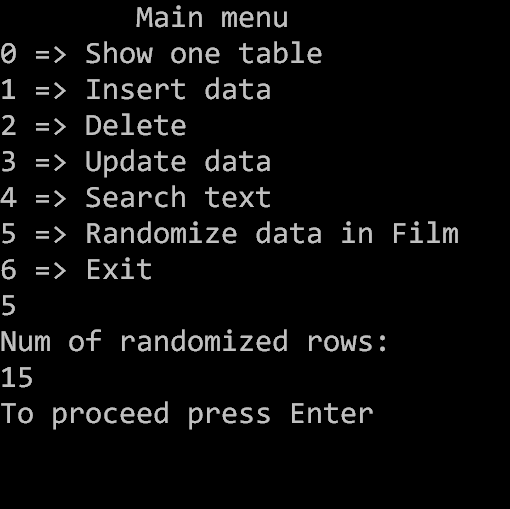
Запит у меню



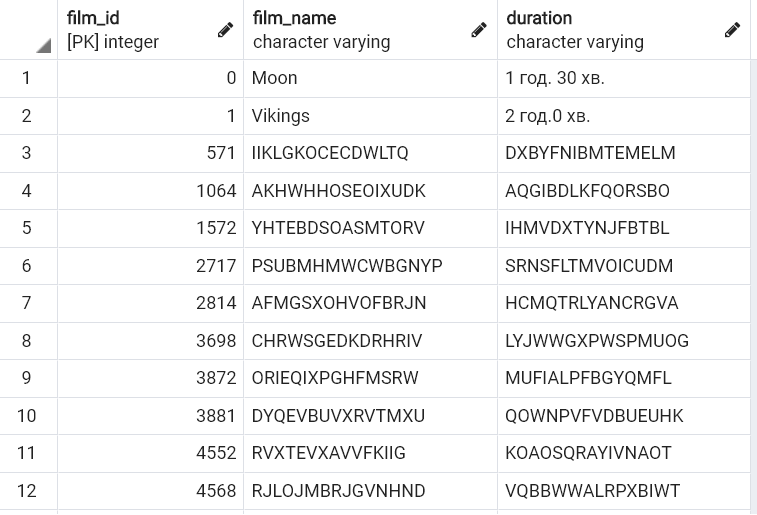
Таблиця після запиту



Початкова таблиця



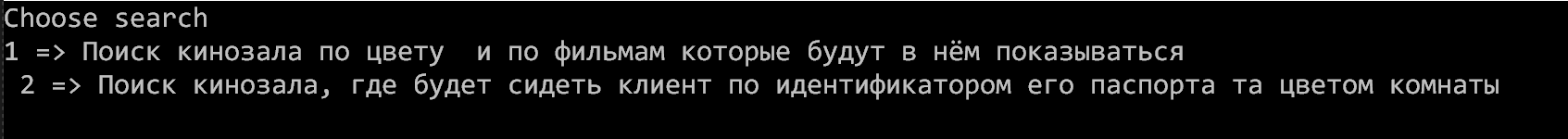
Таблиця після запиту

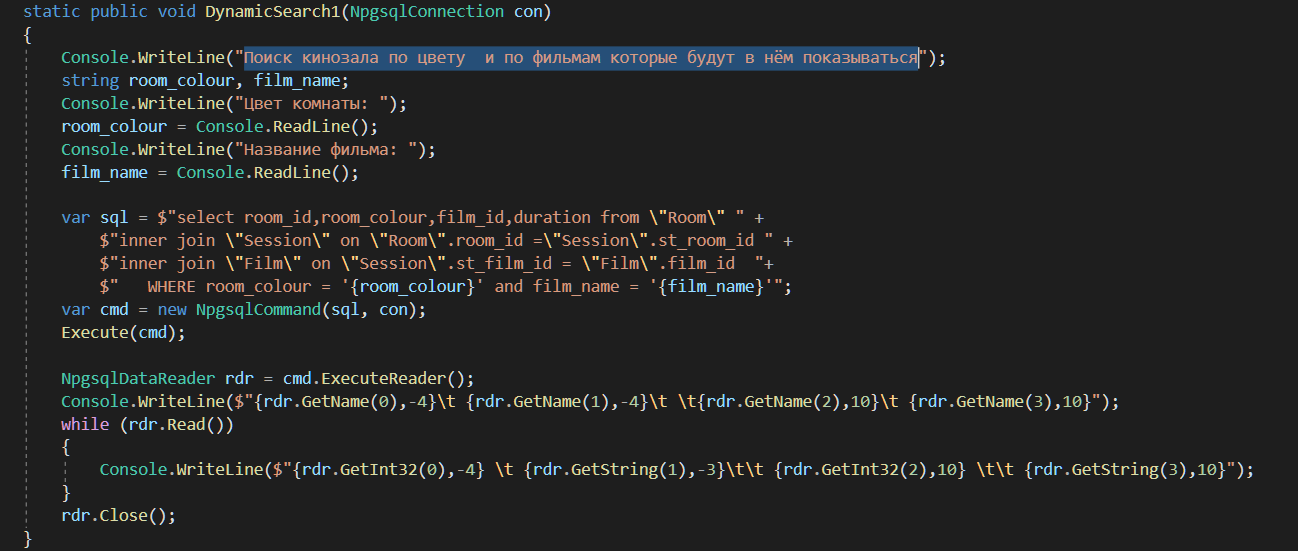




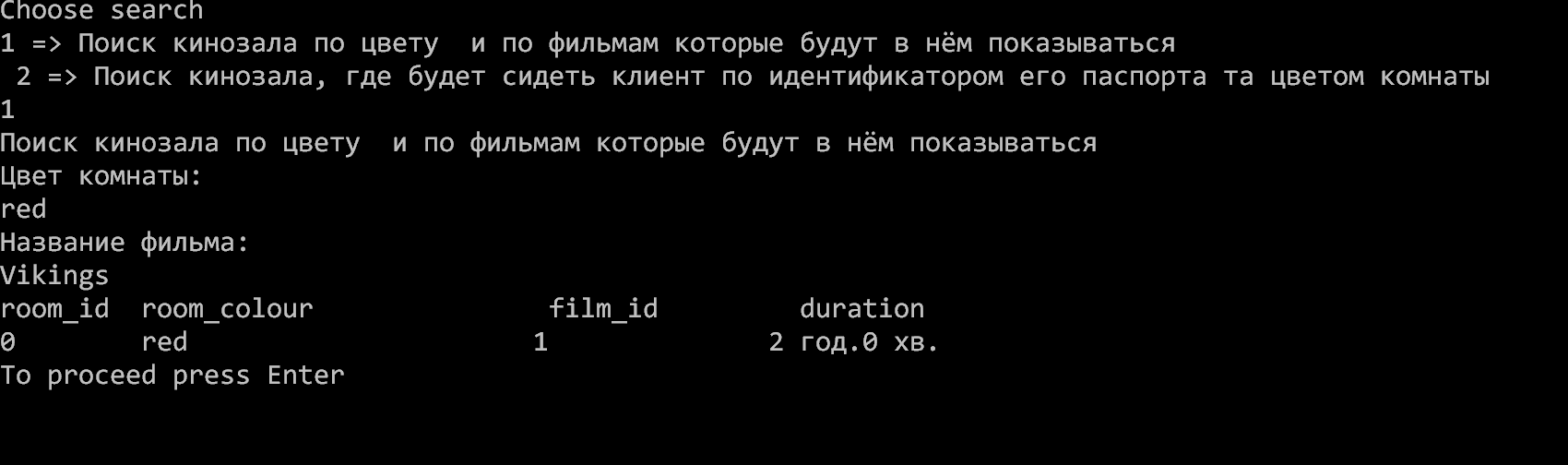
# Виконання динамічних запитів бази даних

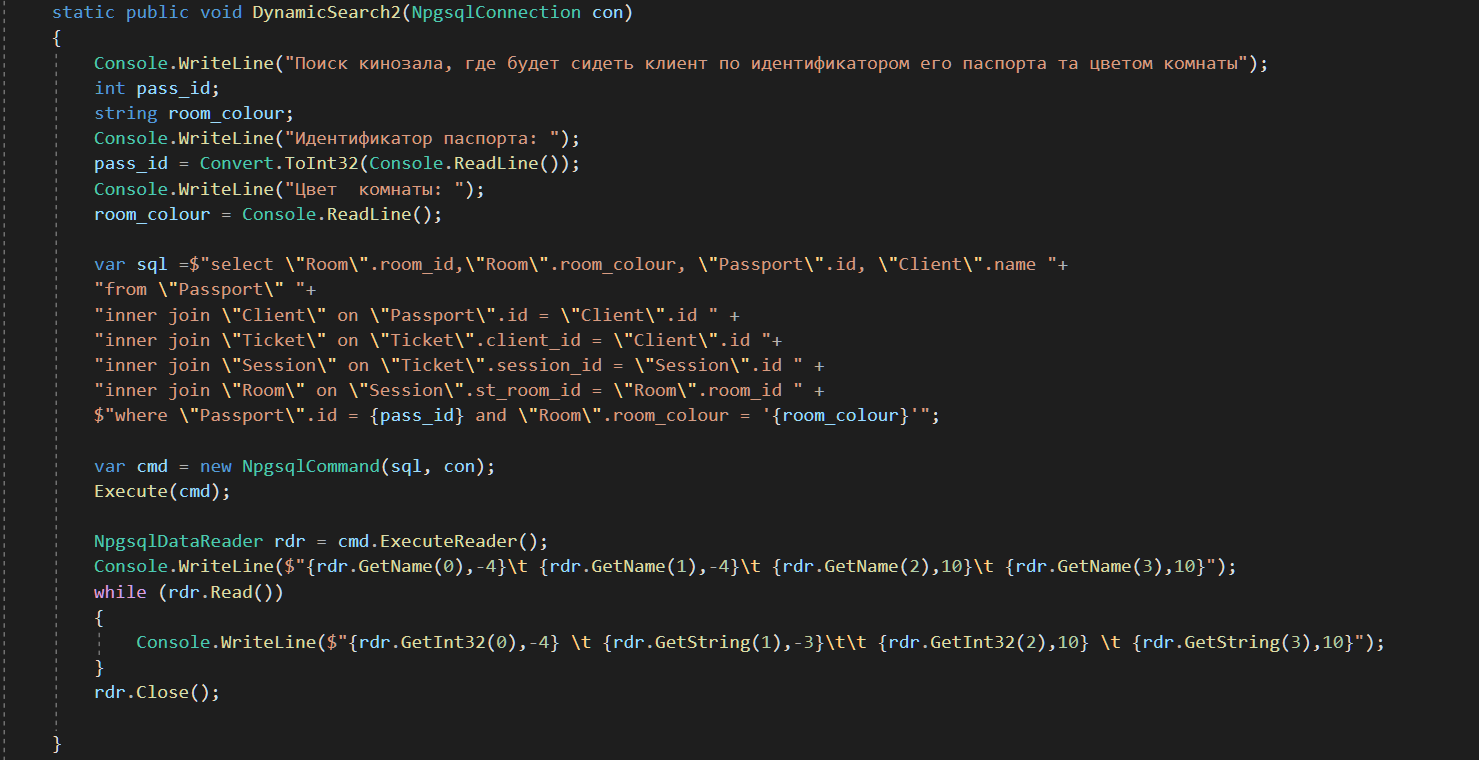
В даній програмі визначені типи динамічних запитів, які визначені в меню:

Рис 21. – Меню програми де ми можемо побачити типи доступних динамічних запитів

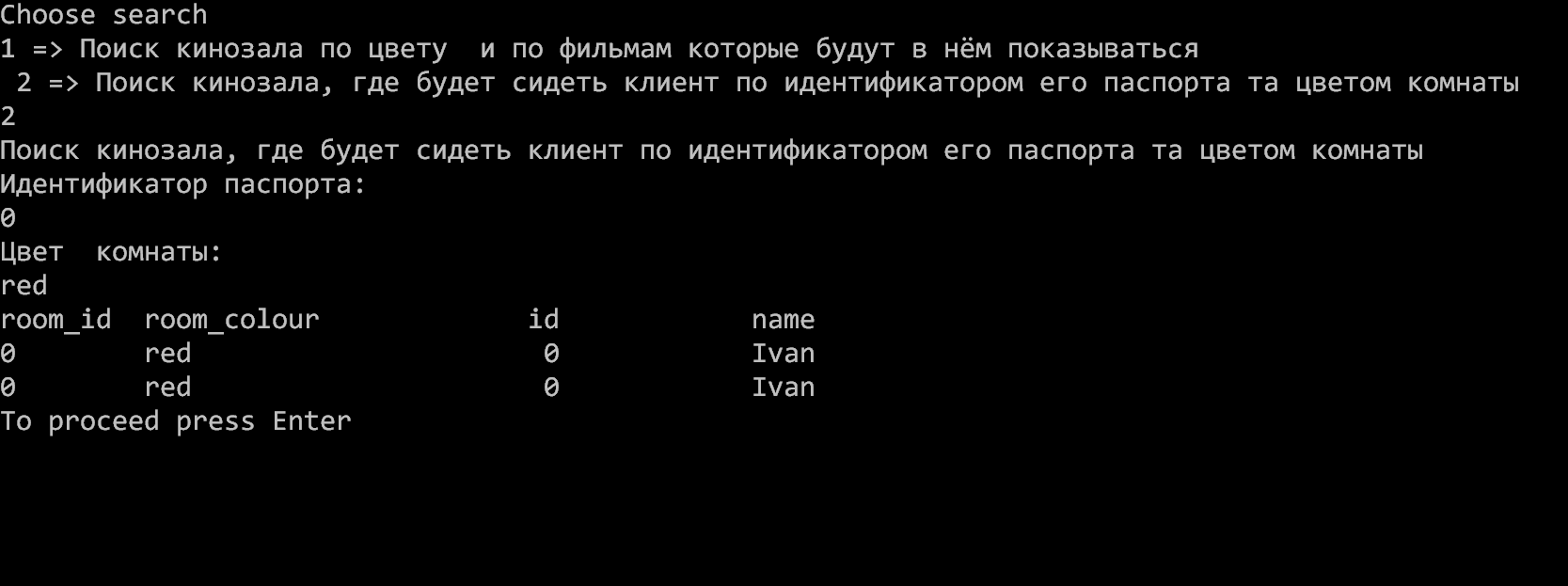
Рис 22. – Фотографія коду програми, 1-го динамічного запиту, що знаходиться в файлі Backend.cs

Запит в меню



Рис 23. – Фотографія коду програми, 2-го динамічного запиту, що знаходиться в файлі Backend.cs

Запит в меню



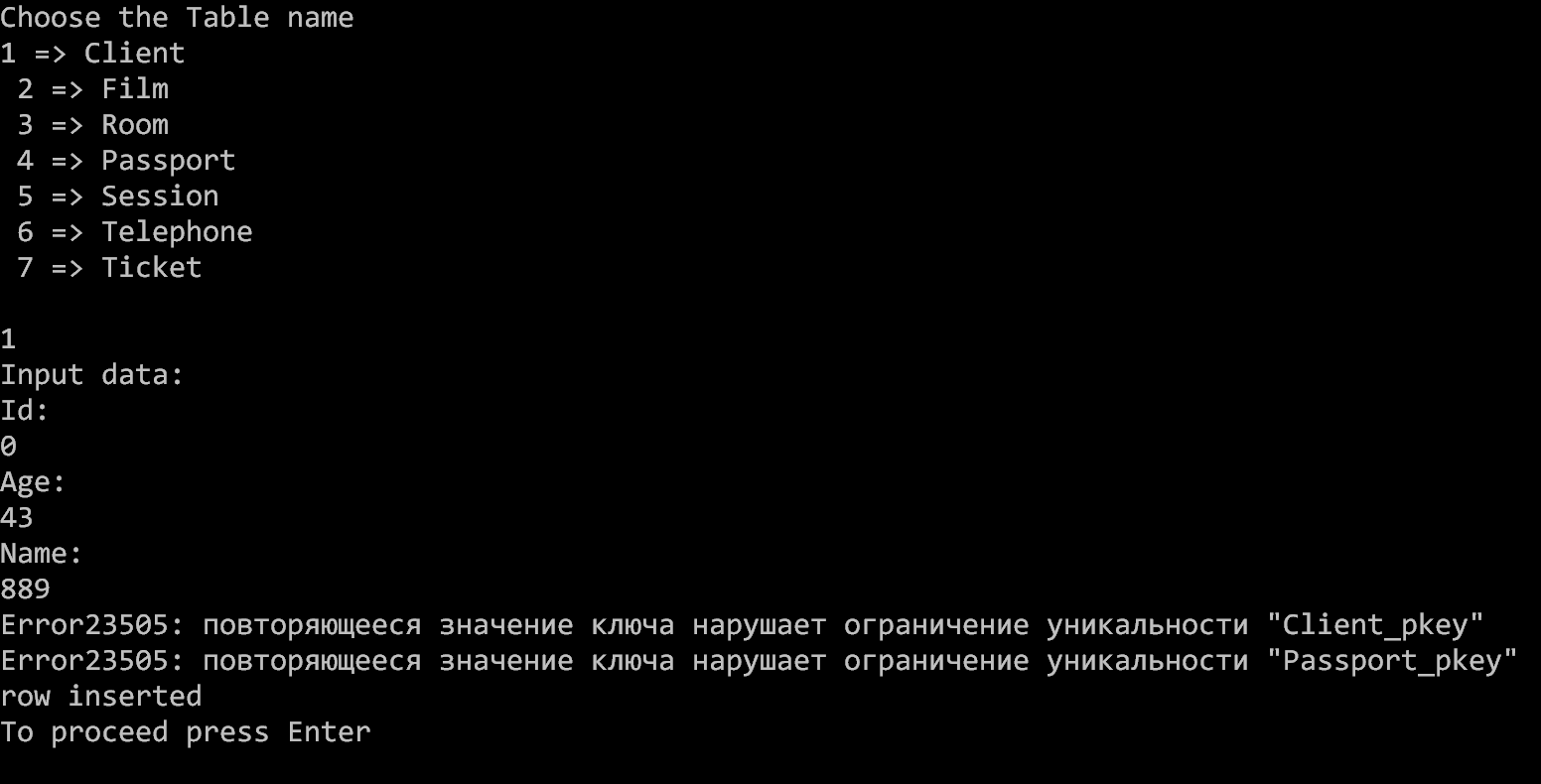
# Обробка виняткових ситуацій (помилок) при введенні/вилученні та валідації даних

Обробка виняткових при виконанні будь-якого запиту виконується за допомогою блоку try-catch у модулі Backend. При введенні помилкових даних помилка серверу SQL не зупинить роботу програми, а в меню користувача з’явиться повідомлення про помилку.

## 

Рис 24. – Фотографія коду програми, де зображений механізм try-catch помилок, що знаходиться у файлі Backend.cs

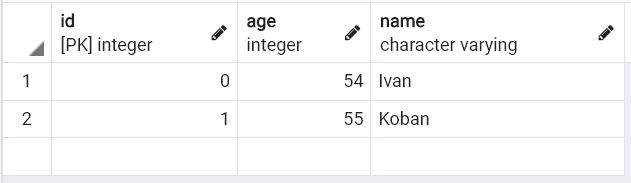
## Приклади перевірки на валідність вхідних даних

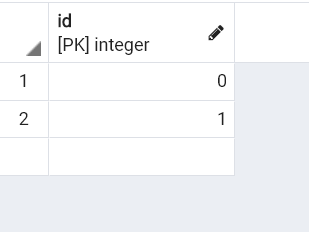


# Дослідження режимів обмеження ON DELETE

Режими обмеження ми будемо досліджувати на прикладі таблиць “Client” – батьківська таблиця та “Passport” - як дочірня.

Початкові таблиці

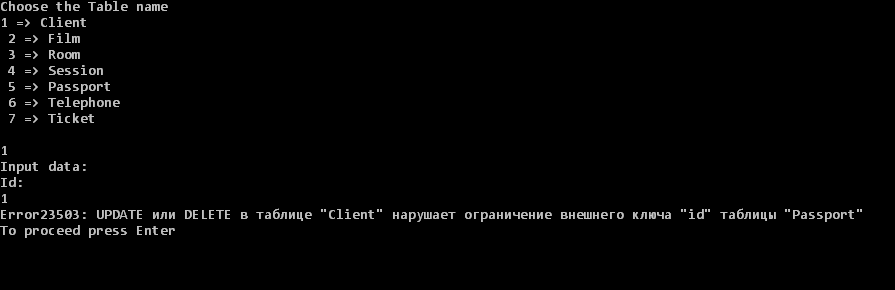




Режим **NO ACTION**, **SET NULL** (за умови що id має обмеження NOT NULL), **SET DEFAULT**, **RESTRICT**

NO ACTION

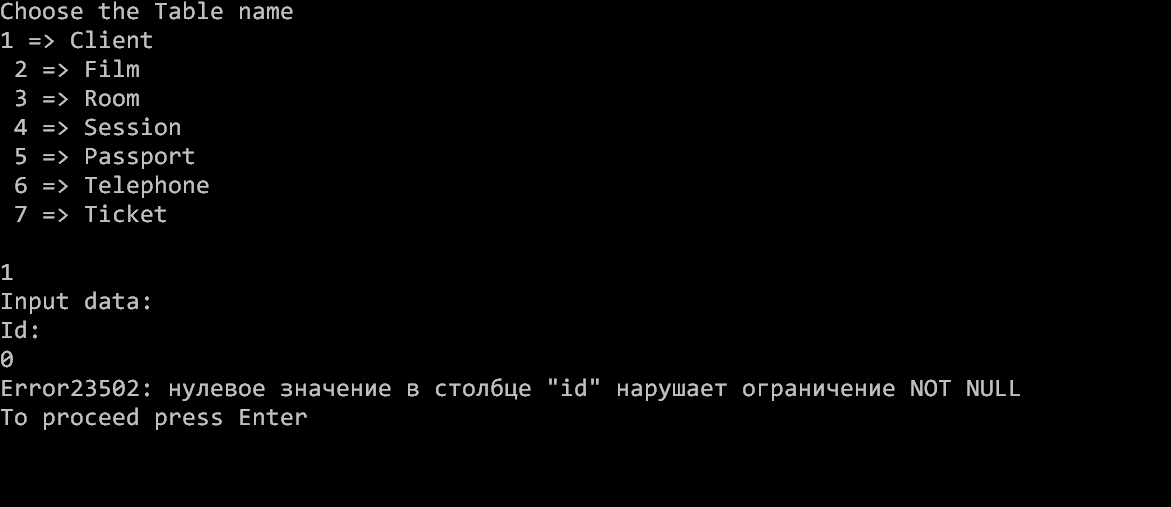
Запит меню



При видаленні запису з батьківської таблиці ,ми можемо побачити помилку щодо порушення зовнішнього ключа.

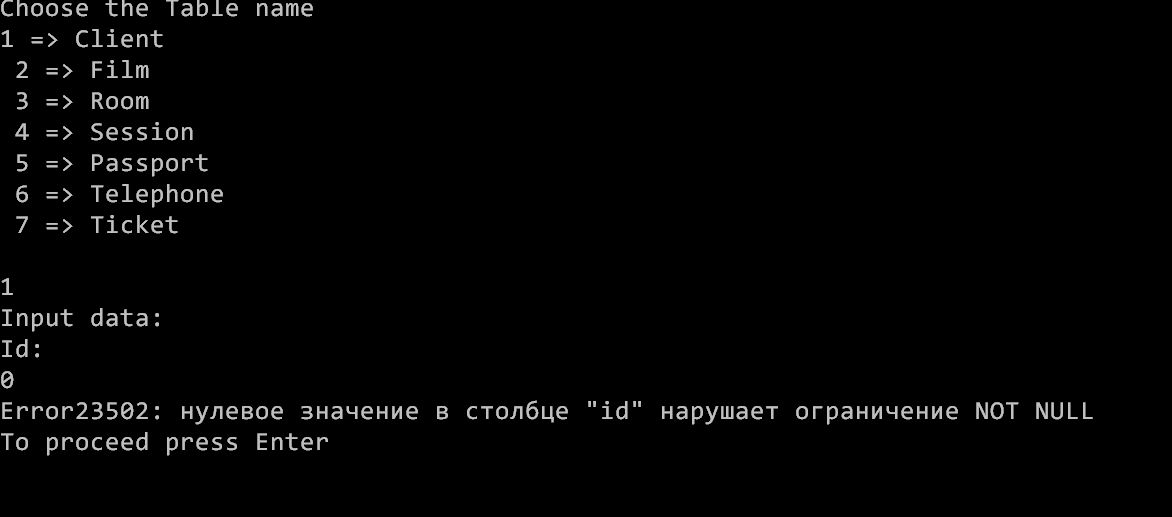
SET NULL

Запит меню



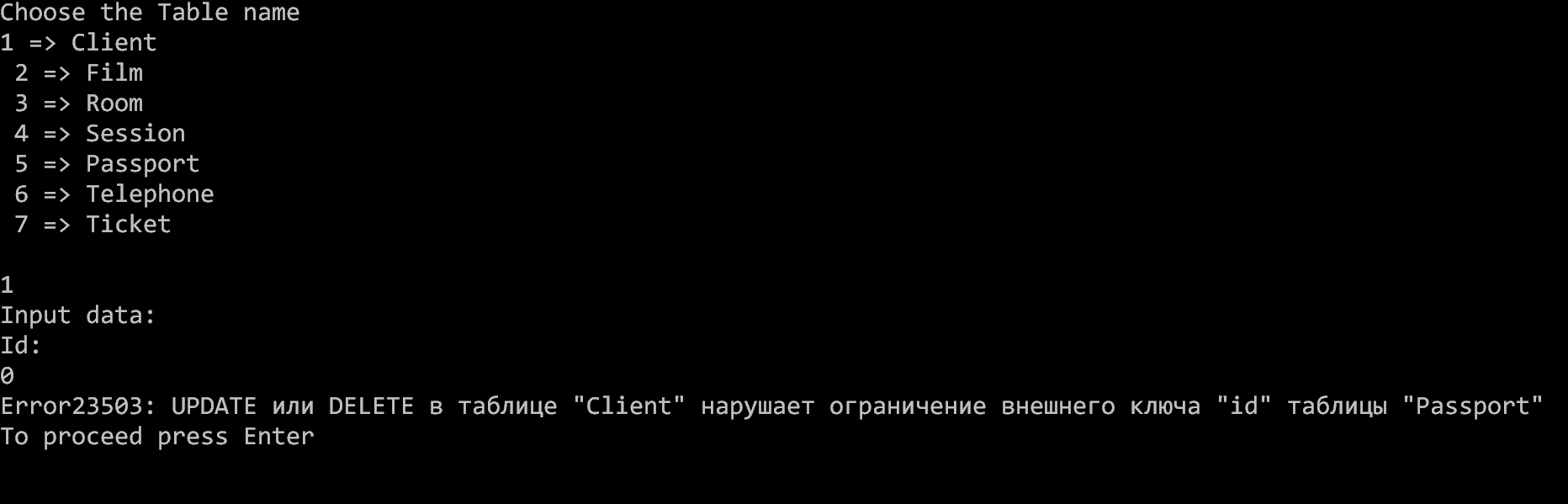
SET DEFAULT

Запит меню



RESTRICT

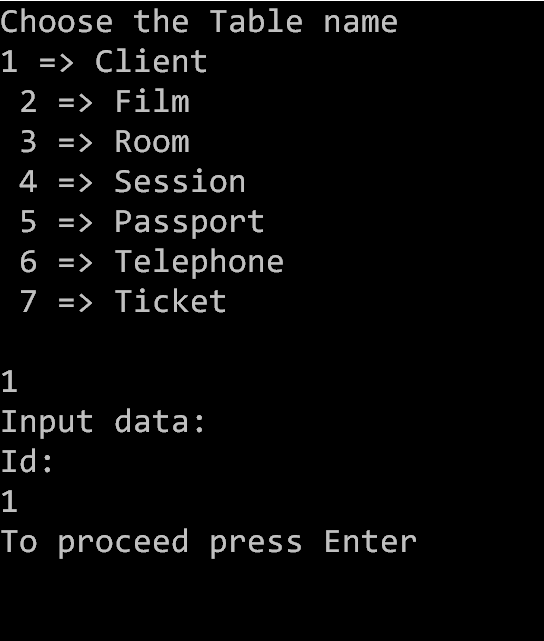
Запит меню



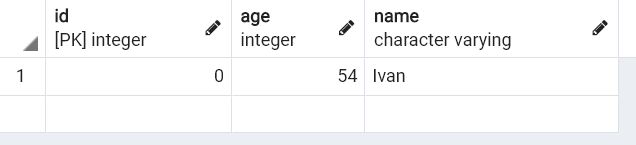
Кожний з вказаних режимів повертає одну й ту саму помилку, через особливості роботи кожного з них. Наприклад, **SET NULL** та **RESTRICT** ,не дозволяє видалити запис, через обмеження NOT NULL у полях запису, **NO ACTION** та **RESTRICT** просто забороняють видалення значень з батьківської таблиці, при наявності зовнішніх залежностей.

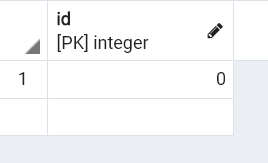
CASCADE

Запит меню



Таблиці після запиту





# Ілюстрації програмного коду на Github

