

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №3**

з дисципліни **Бази даних і засоби управління**

*на тему: “ Засоби оптимізації роботи СУБД PostgreSQL”*

Виконав:

студент ІII курсу

групи КВ-02

Войтович Гліб Олександрович

Перевірив: Павловський Володимир Ілліч

Київ – 2022

*Метою роботи* є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

*Завдання* роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).

2. Створити та проаналізувати різні типи індексів у PostgreSQL.

3. Розробити тригер бази даних PostgreSQL.

4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

**Виконання роботи**

Логічна модель предметної області «Комп’ютерний клуб»:

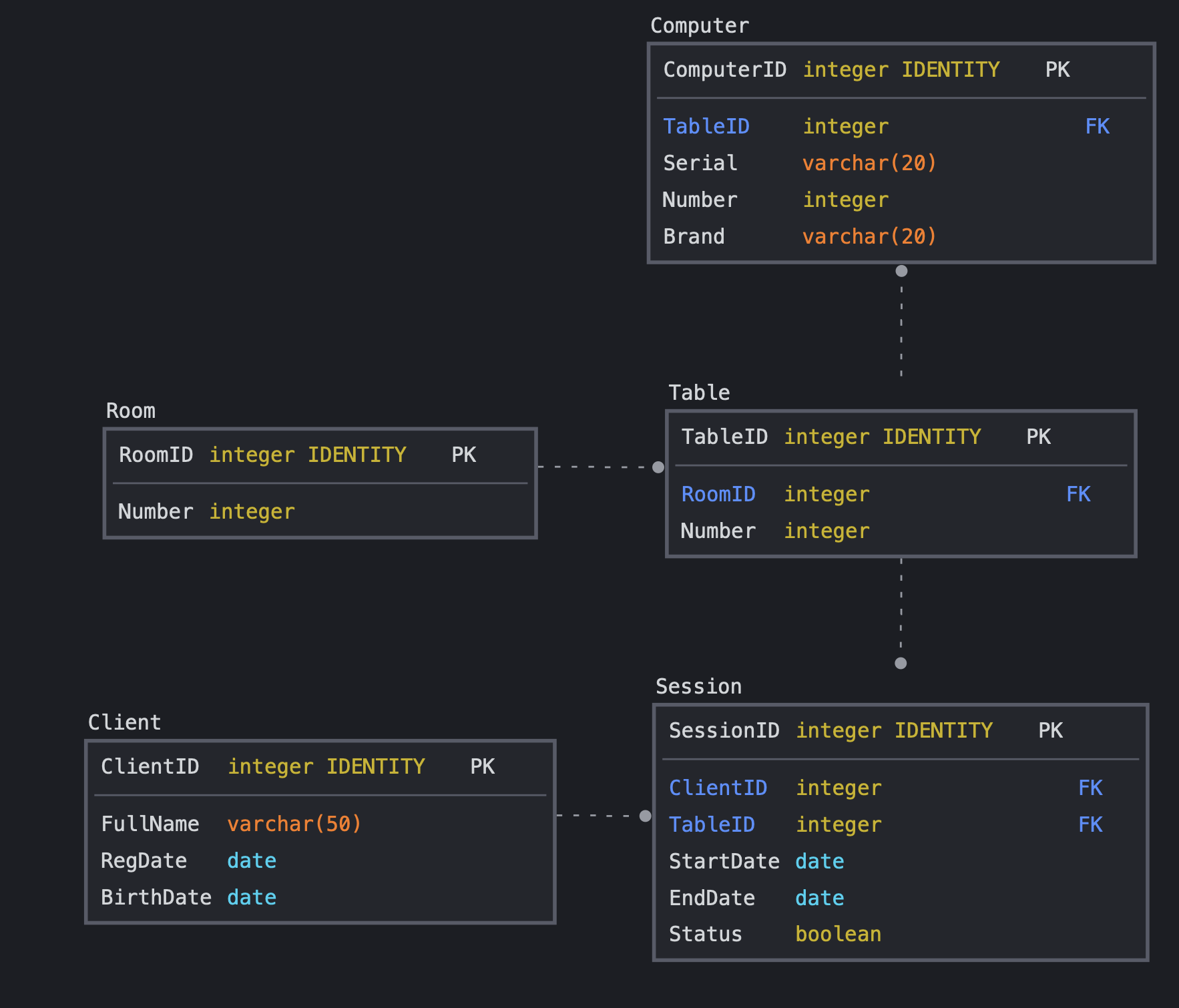


Рисунок 1. Схема бази даних (інструмент: sqldbm.com)

**Середовище та компоненти розробки**

Для розробки використовувалась мова програмування Python, середовище розробки PyCharm та стороння бібліотека SQLAlchemy.

**Структура програми та її опис**

Програма складається з трьох файлів-модулів: Controller, View та Model.

* Controller відповідає за обробку запитів користувача.
* Model за з’єднання та роботу з БД.
* View за оформлення та вивід інформації.

**Модулі моделі ORM**

*dbModel.py*

import dbView as View  
import dbModel as Model  
import time  
  
  
def operationInsert(userChoice):  
 print("Enter values for " + userChoice + " table (using ,)")  
 val = View.getSplitInput()  
  
 conn = Model.makerSession()  
 if userChoice == "Room":  
 res = Model.Room(Number=val[0])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Table":  
 res = Model.Table(TableID=val[0], Number=val[1])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Computer":  
 res = Model.Computer(TableID=val[0], Serial=val[1], Number=val[2], Brand=val[3])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Client":  
 res = Model.Client(FullName=val[0], RegDate=val[1], BirthDate=val[2])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Session":  
 res = Model.Session(ClientID=val[0], TableID=val[1], StartDate=val[2], EndDate=val[3], Status=int(val[4]))  
 conn.add(res)  
 conn.commit()  
 conn.close()  
  
  
def operationDelete(userChoice):  
 print("Enter ID in " + userChoice + " table")  
  
 val = View.getInput(None)  
  
 conn = Model.makerSession()  
 if userChoice == "Room":  
 res = conn.query(Model.Room).filter(Model.Room.RoomID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Table":  
 res = conn.query(Model.Table).filter(Model.Table.TableID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Computer":  
 res = conn.query(Model.Computer).filter(Model.Computer.ComputerID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Client":  
 res = conn.query(Model.Client).filter(Model.Client.ClientID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Session":  
 res = conn.query(Model.Session).filter(Model.Session.SessionID == val).first()  
 conn.delete(res)  
 conn.commit()  
 conn.close()  
  
  
def operationEdit(userChoice):  
 print("Enter ID in " + userChoice + " table")  
 valID = View.getInput(None)  
 print("Enter values for " + userChoice + " table (using ,)")  
 val = View.getSplitInput()  
  
 conn = Model.makerSession()  
 if userChoice == "Room":  
 res = conn.query(Model.Room).filter(Model.Room.RoomID == valID).first()  
 res.Number = val[0]  
 conn.commit()  
 elif userChoice == "Table":  
 res = conn.query(Model.Table).filter(Model.Table.TableID == valID).first()  
 res.RoomID = val[0]  
 res.Number = val[1]  
 conn.commit()  
 elif userChoice == "Computer":  
 res = conn.query(Model.Computer).filter(Model.Computer.ComputerID == valID).first()  
 res.TableID = val[0]  
 res.Serial = val[1]  
 res.Number = val[2]  
 res.Brand = val[3]  
 conn.commit()  
 elif userChoice == "Client":  
 res = conn.query(Model.Client).filter(Model.Client.ClientID == valID).first()  
 res.FullName = val[0]  
 res.FullName = val[1]  
 res.BirthDate = val[2]  
 conn.commit()  
 elif userChoice == "Session":  
 res = conn.query(Model.Session).filter(Model.Session.SessionID == valID).first()  
 res.ClientID = val[0]  
 res.TableID = val[1]  
 res.StartDate = val[2]  
 res.EndDate = val[3]  
 res.Status = int(val[4])  
 conn.commit()  
  
 conn.close()  
  
  
def mainMenu():  
 View.showMainMenu()  
 choiceMenu = View.getInput(None)  
 choiceTable = View.getInput(" (?) Write table name: ")  
  
 if choiceMenu == '1':  
 operationInsert(choiceTable)  
 elif choiceMenu == '2':  
 operationDelete(choiceTable)  
 elif choiceMenu == '3':  
 operationEdit(choiceTable)

*dbController.py*

import dbModel as Model  
import dbView as View  
  
  
def operationInsert(userChoice):  
 print("Enter values for " + userChoice + " table (using ,)")  
 val = View.getSplitInput()  
  
 conn = Model.makerSession()  
 if userChoice == "Room":  
 res = Model.Room(Number=val[0])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Table":  
 res = Model.Table(TableID=val[0], Number=val[1])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Computer":  
 res = Model.Computer(TableID=val[0], Serial=val[1], Number=val[2], Brand=val[3])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Client":  
 res = Model.Client(FullName=val[0], RegDate=val[1], BirthDate=val[2])  
 conn.add(res)  
 conn.commit()  
 elif userChoice == "Session":  
 res = Model.Session(ClientID=val[0], TableID=val[1], StartDate=val[2], EndDate=val[3], Status=int(val[4]))  
 conn.add(res)  
 conn.commit()  
 conn.close()  
  
  
def operationDelete(userChoice):  
 print("Enter ID in " + userChoice + " table")  
  
 val = View.getInput(None)  
  
 conn = Model.makerSession()  
 if userChoice == "Room":  
 res = conn.query(Model.Room).filter(Model.Room.RoomID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Table":  
 res = conn.query(Model.Table).filter(Model.Table.TableID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Computer":  
 res = conn.query(Model.Computer).filter(Model.Computer.ComputerID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Client":  
 res = conn.query(Model.Client).filter(Model.Client.ClientID == val).first()  
 conn.delete(res)  
 conn.commit()  
 elif userChoice == "Session":  
 res = conn.query(Model.Session).filter(Model.Session.SessionID == val).first()  
 conn.delete(res)  
 conn.commit()  
 conn.close()  
  
  
def operationEdit(userChoice):  
 print("Enter ID in " + userChoice + " table")  
 valID = View.getInput(None)  
 print("Enter values for " + userChoice + " table (using ,)")  
 val = View.getSplitInput()  
  
 conn = Model.makerSession()  
 if userChoice == "Room":  
 res = conn.query(Model.Room).filter(Model.Room.RoomID == valID).first()  
 res.Number = val[0]  
 conn.commit()  
 elif userChoice == "Table":  
 res = conn.query(Model.Table).filter(Model.Table.TableID == valID).first()  
 res.RoomID = val[0]  
 res.Number = val[1]  
 conn.commit()  
 elif userChoice == "Computer":  
 res = conn.query(Model.Computer).filter(Model.Computer.ComputerID == valID).first()  
 res.TableID = val[0]  
 res.Serial = val[1]  
 res.Number = val[2]  
 res.Brand = val[3]  
 conn.commit()  
 elif userChoice == "Client":  
 res = conn.query(Model.Client).filter(Model.Client.ClientID == valID).first()  
 res.FullName = val[0]  
 res.RegDate = val[1]  
 res.BirthDate = val[2]  
 conn.commit()  
 elif userChoice == "Session":  
 res = conn.query(Model.Session).filter(Model.Session.SessionID == valID).first()  
 res.ClientID = val[0]  
 res.TableID = val[1]  
 res.StartDate = val[2]  
 res.EndDate = val[3]  
 res.Status = int(val[4])  
 conn.commit()  
  
 conn.close()  
  
  
def mainMenu():  
 View.showMainMenu()  
 choiceMenu = View.getInput(None)  
 choiceTable = View.getInput(" (?) Write table name: ")  
  
 if choiceMenu == '1':  
 operationInsert(choiceTable)  
 elif choiceMenu == '2':  
 operationDelete(choiceTable)  
 elif choiceMenu == '3':  
 operationEdit(choiceTable)

*dbView.py*

def showMainMenu():  
 print(" KV-02 Voitovych Hlib. Lab Work 3")  
 print(" [------MAIN MENU------]")  
 print(" [1] Insert data in table")  
 print(" [2] Remove data in table")  
 print(" [3] Edit data in table ")  
 print(" Please choose operation ")  
  
  
def getSplitInput():  
 inp = input().split(', ')  
 return inp  
  
  
def getInput(string):  
 if string is None:  
 inp = input(" > ")  
 else:  
 inp = input(string)  
 return inp

*main.py*

import dbController as Controller  
  
Controller.mainMenu()

**Головне меню програми**

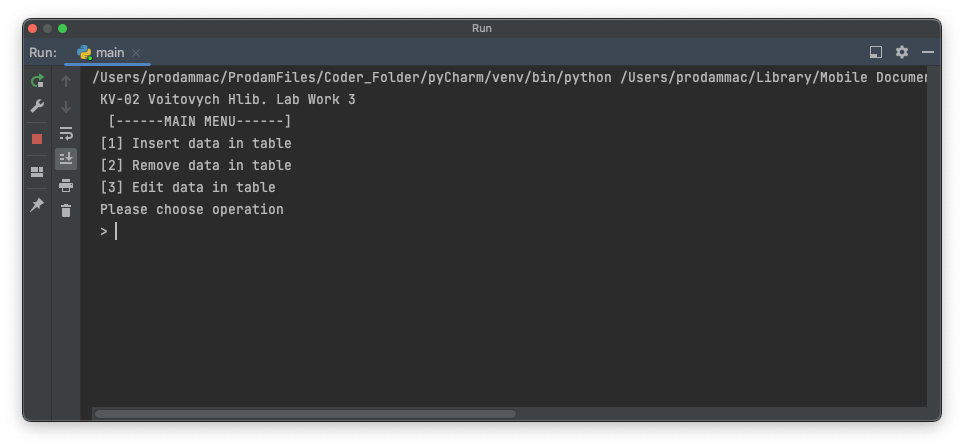
****

Рисунок 2. Початкове меню програми

Меню для користувача складається із трьох пунктів (Рис. 3):

* Перший пункт меню (Рис. 4) пропонує розпочати операцію внесення нових даних до таблиці, яку вибере користувач.
* Другий пункт меню (Рис. 5) пропонує приступити до операції видалення даних із таблиці, яку вибере користувач.
* Третій пункт (Рис. 6) пропонує редагувати дані у таблиці, яку вибере користувач.

**Результати роботи програми**

**Внесення даних**

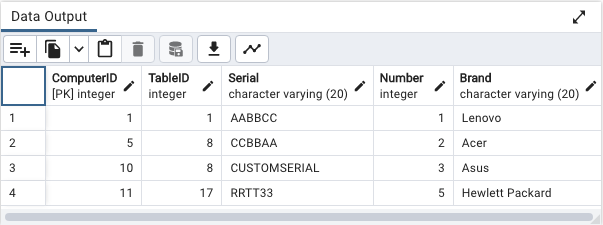


Рисунок 3. Стан таблиці Computer перед внесенням нових даних

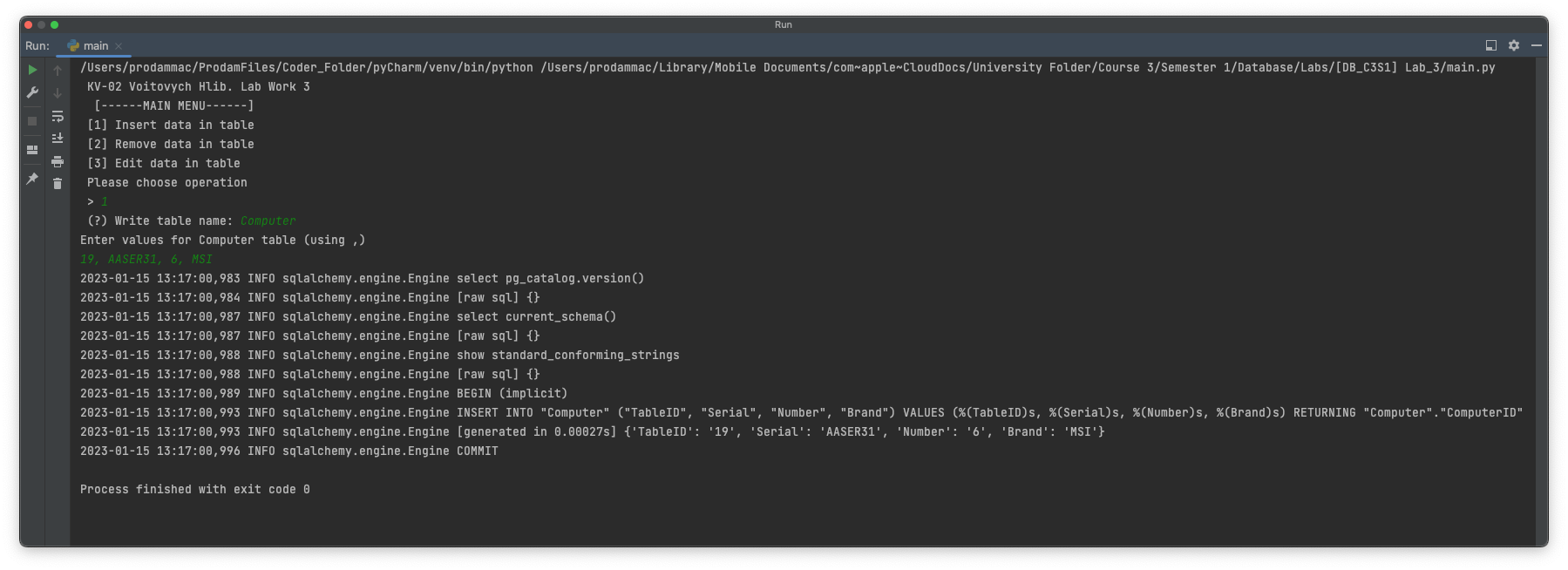


Рисунок 4. Внесення даних до обраної таблиці

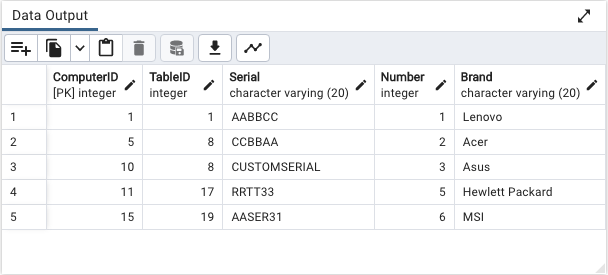


Рисунок 5. Стан таблиці Computer після внесення нових даних

**Видалення даних**

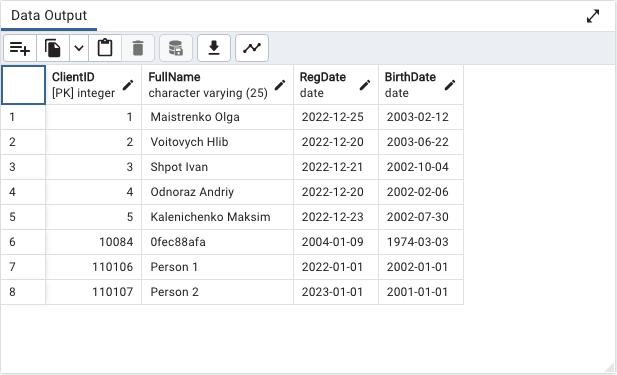


Рисунок 6. Стан таблиці Client перед видаленням даних

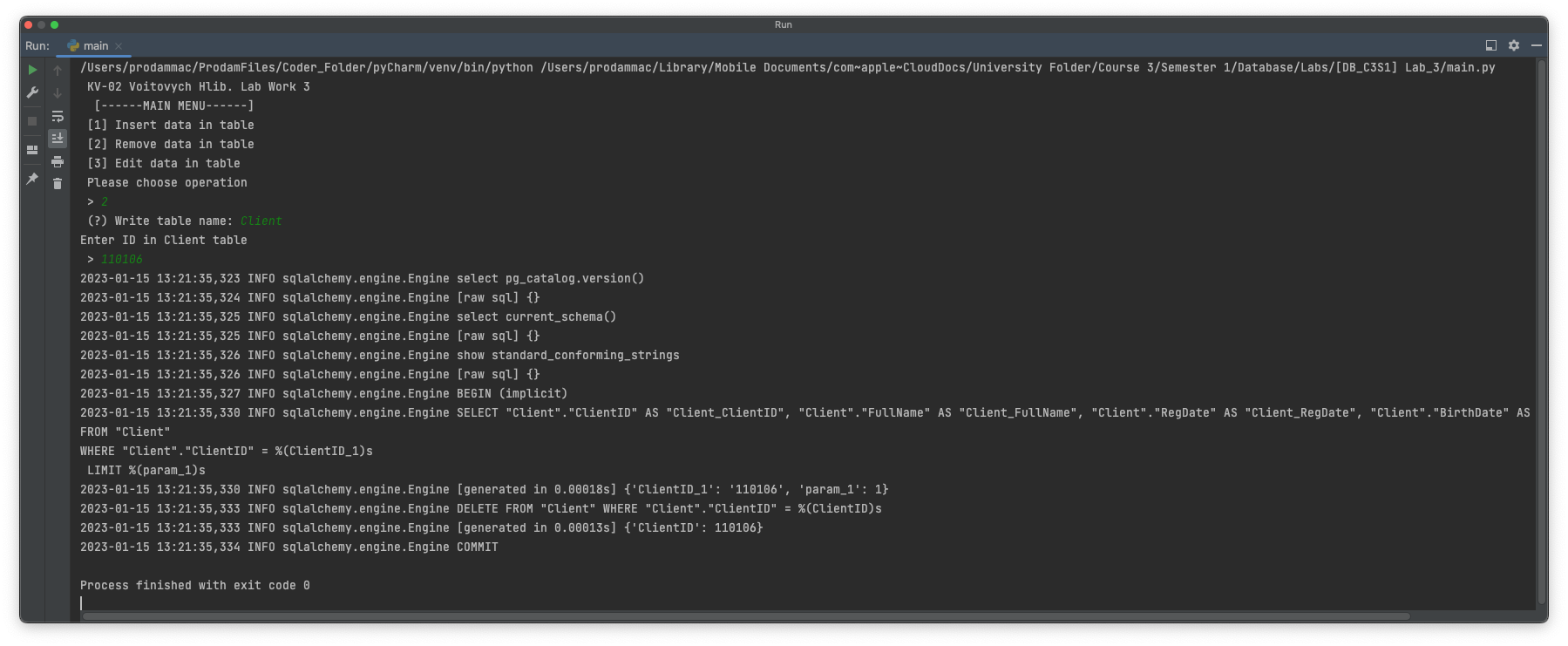


Рисунок 7. Видалення даних з обраної таблиці

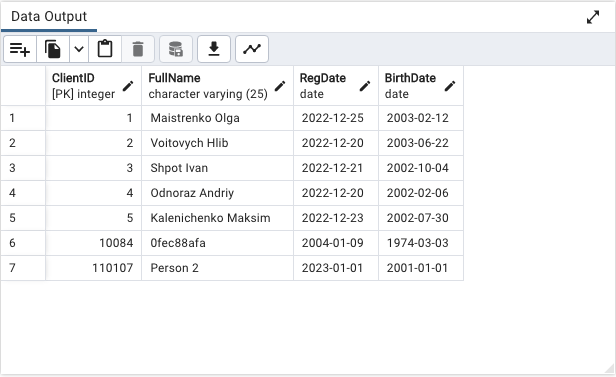


Рисунок 8. Стан таблиці Client після видалення даних

**Редагування даних**

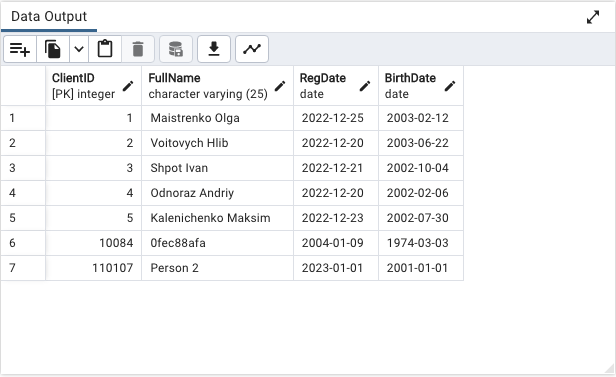


Рисунок 9. Стан таблиці Client перед редагуванням даних

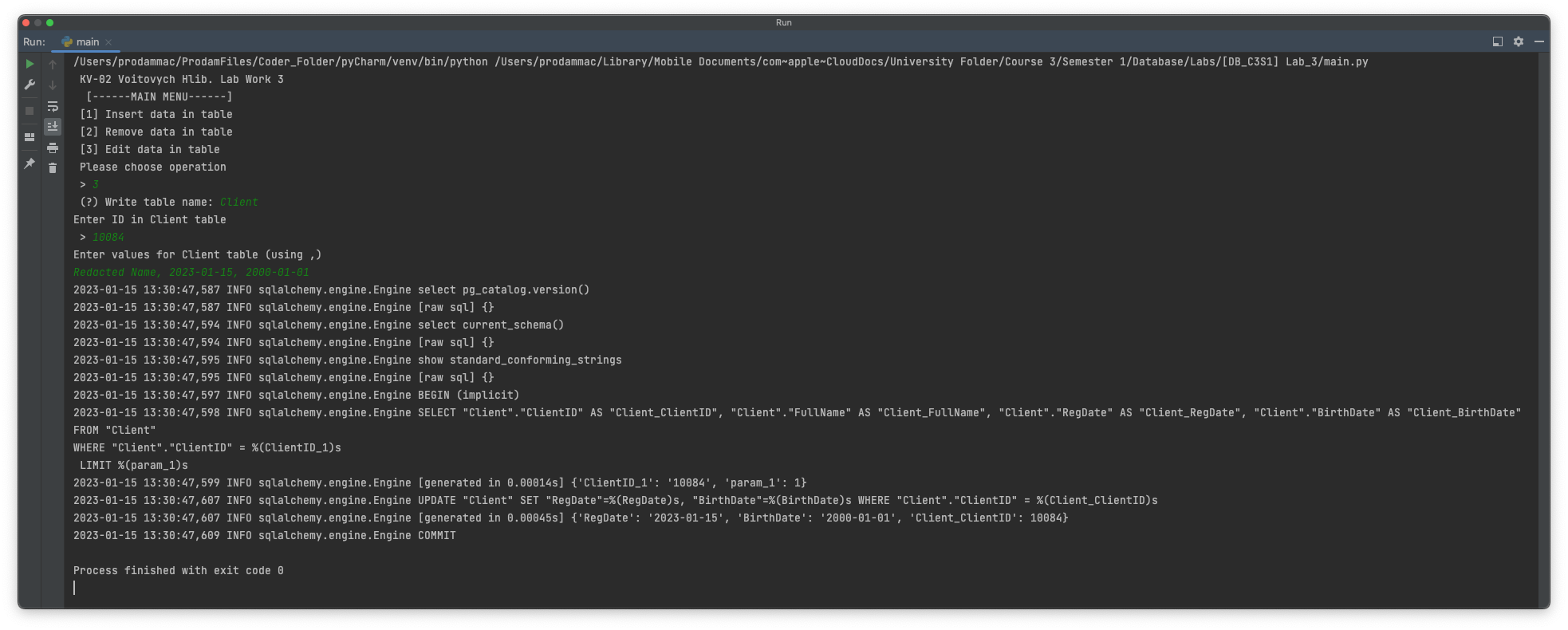


Рисунок 10. Редагування даних в обраній таблиці

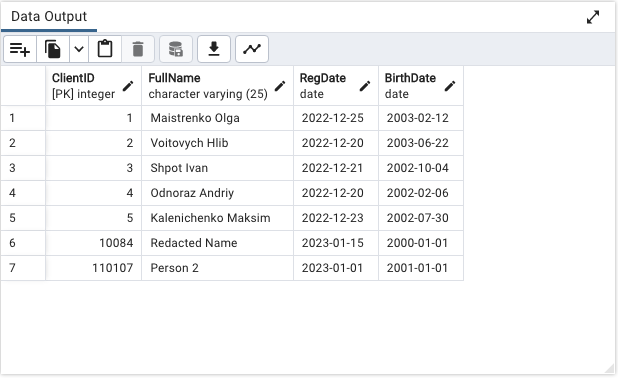


Рисунок 11. Стан таблиці Client після редагуванням даних

**Тригер бази даних PostgreSQL. Умова для тригера – after delete, insert**

Відповідно до мого завдання, я розробив два тригери, які будуть реагувати на видалення та вставку в таблиці Session. Після видалення або вставки в цій таблиці буде додано новий запис у спеціально створеній для демонстрації роботи тригера таблиці LogSession.

*Код функції SesLogInsert():*

**BEGIN**

**INSERT** **INTO** public."LogSession" ( "SessionID", "User", "TimeStamp", "Message")

VALUES(**NEW**."SessionID",**CURRENT\_USER**, **CURRENT\_TIMESTAMP**, 'Session entry added');

**RETURN** **NEW**;

**END**;

*Тригер SesLogInsertTrigger:*

**CREATE** **TRIGGER** SesLogInsertTrigger

**AFTER** **INSERT**

**ON** public."Session"

**FOR** **EACH** **ROW**

**EXECUTE** **PROCEDURE** SesLogInsert();

**Демонстрація роботи тригера SesLogInsertTrigger**

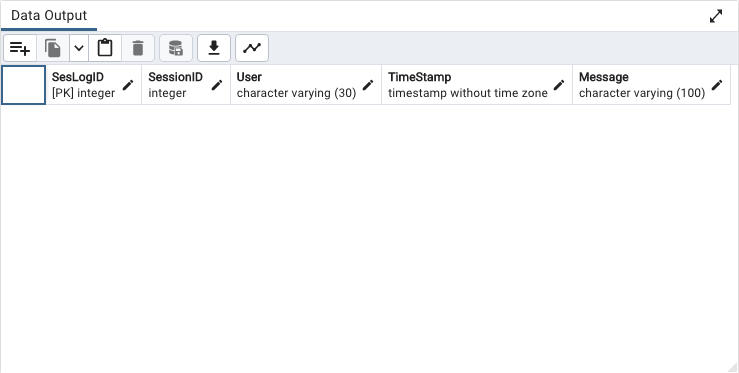


Рисунок 12. Стан таблиці LogSession до роботи тригера

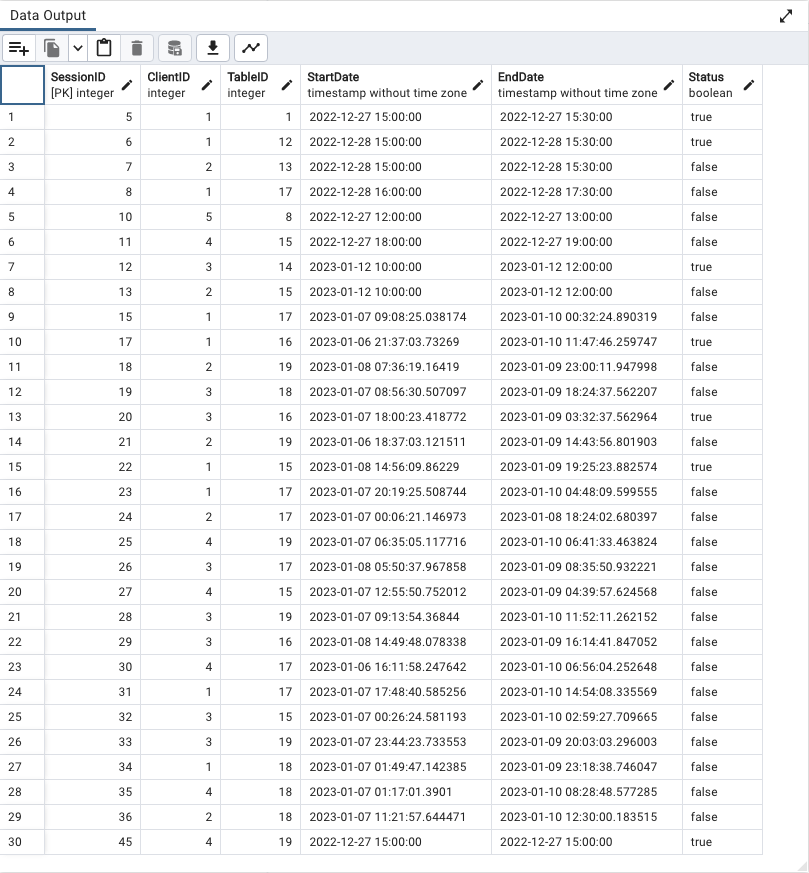


Рисунок 13. Стан таблиці Session перед внесенням нових даних



Рисунок 14. Внесення даних до таблиці Session

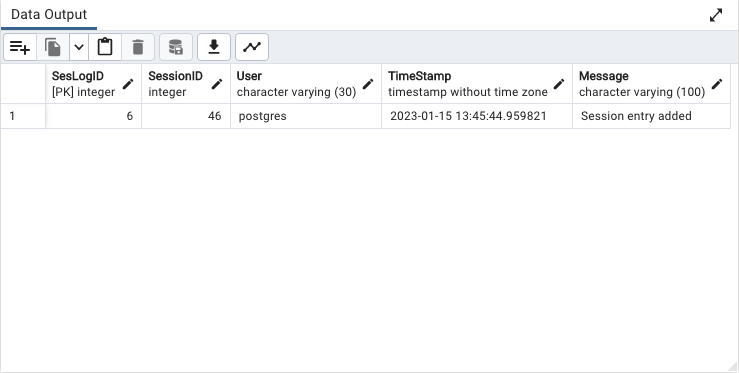


Рисунок 15. Стан таблиці LogSession після роботи тригера

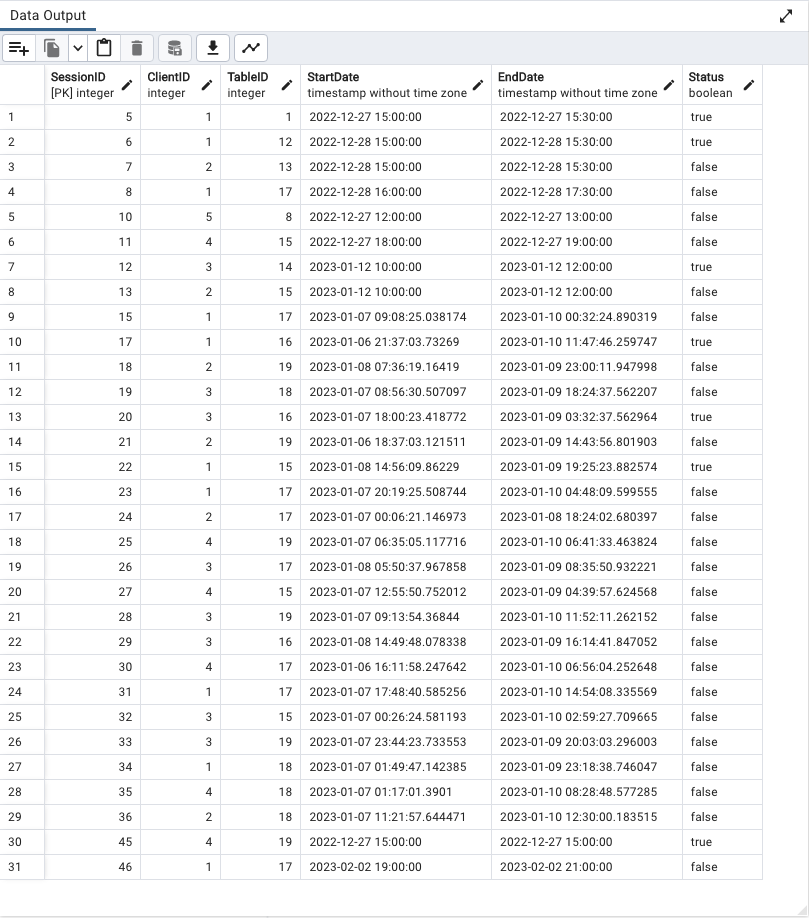


Рисунок 16. Стан таблиці Session після внесення нових даних

*Код функції SesLogDelete():*

**BEGIN**

**INSERT** **INTO** public."LogSession" ( "SessionID", "User", "TimeStamp", "Message")

VALUES(**OLD**."SessionID",**CURRENT\_USER**, **CURRENT\_TIMESTAMP**, 'Session entry deleted');

**RETURN** **NEW**;

**END**;

*Тригер SesLogDeleteTrigger:*

**CREATE** **TRIGGER** SesLogDeleteTrigger

**AFTER** **DELETE**

**ON** public."Session"

**FOR** **EACH** **ROW**

**EXECUTE** **PROCEDURE** SesLogDelete();

**Демонстрація роботи тригера SesLogDeleteTrigger**

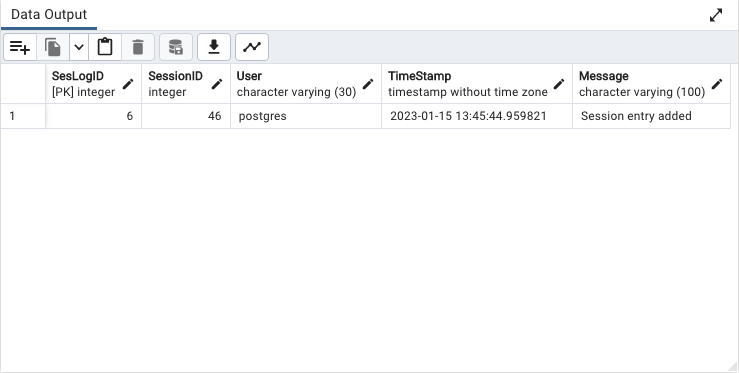


Рисунок 17. Стан таблиці LogSession до роботи тригера

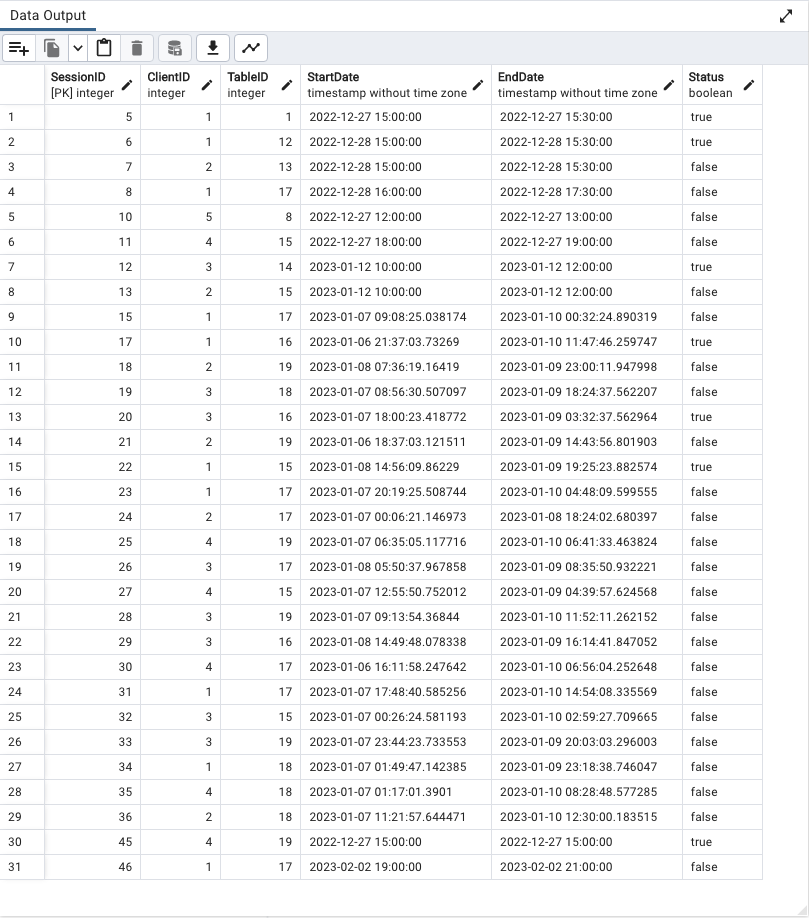


Рисунок 18. Стан таблиці Session перед видаленням даних

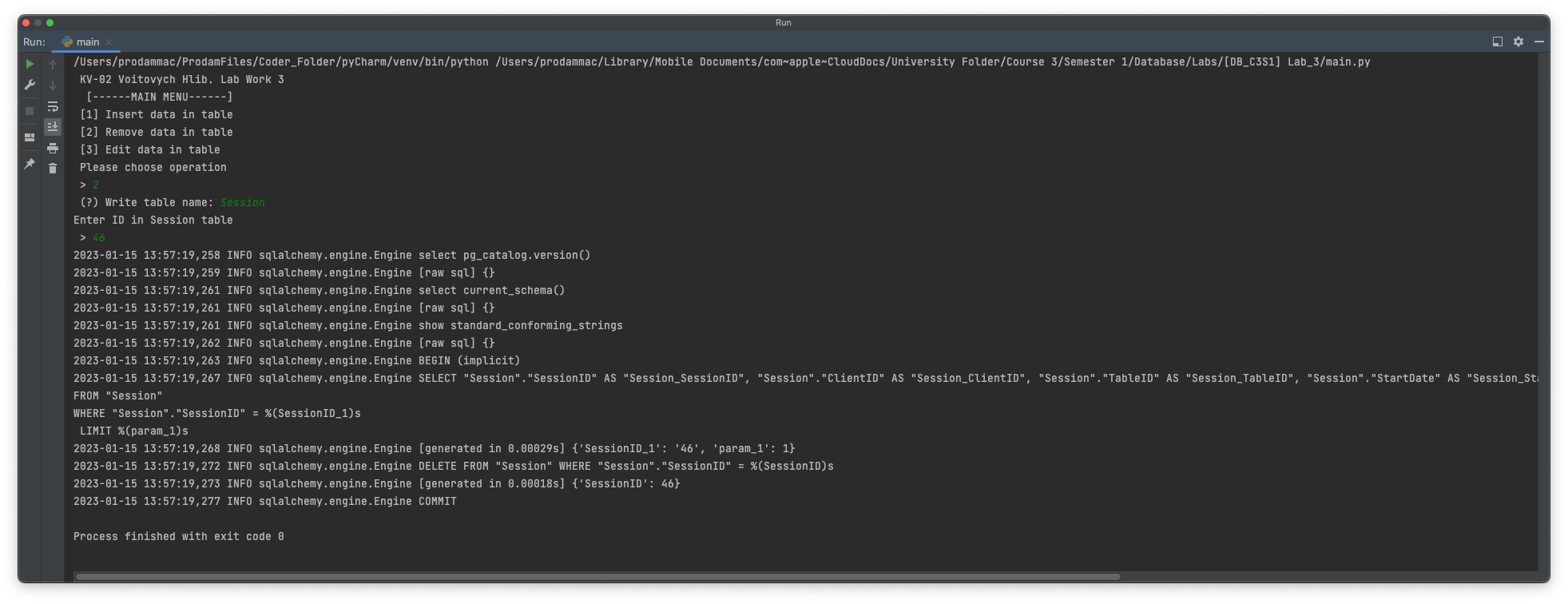


Рисунок 19. Видалення даних з таблиці Session

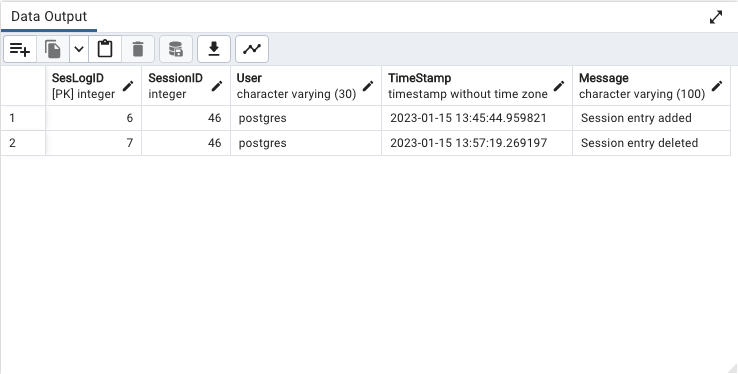


Рисунок 20. Стан таблиці LogSession після роботи тригера

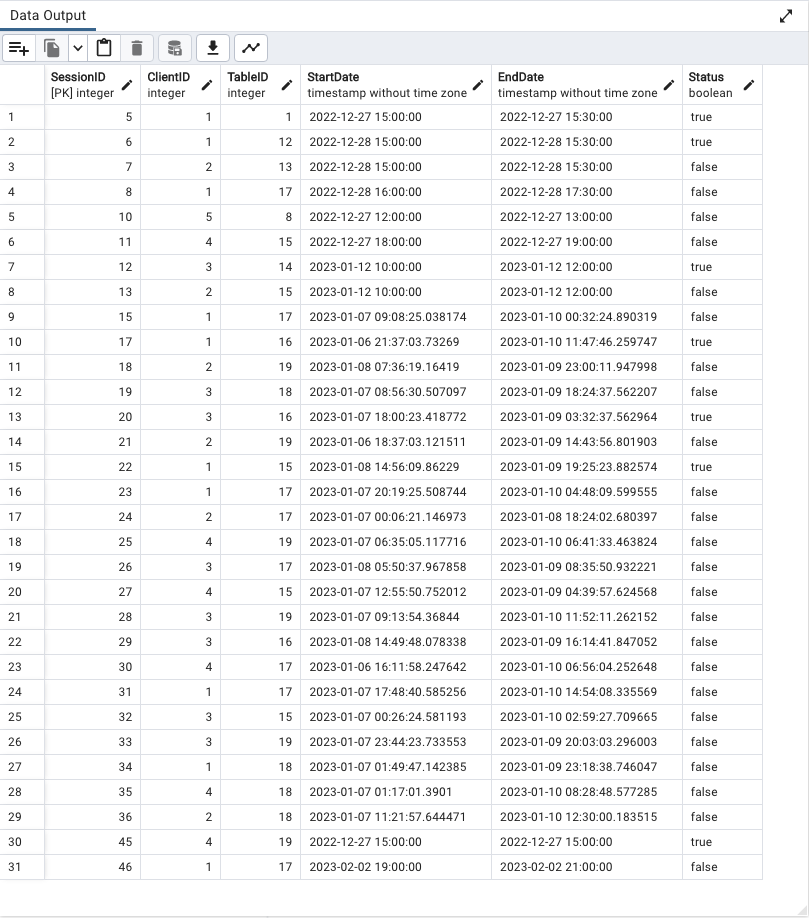


Рисунок 21. Стан таблиці Session після видалення даних