Лабораторная работа №5

Цель работы: разработать программу на языке Python, которая позволяет пользователю создавать таблицы в базе данных SQLite с произвольными полями и типами данных вводить данные в созданные таблицы, выводить введенные данные для проверки корректности работы программы. В ходе выполнения лабораторной работы изучаются основы работы с базами данных в Python с использованием библиотеки sqlite3, а также принципы динамического создания SQL-запросов на основе пользовательского ввода.

Ход работы:

1. **Подключение к базе данных:**
   - В программе используется библиотека sqlite3, которая предоставляет интерфейс для работы с базами данных SQLite. При запуске программы создается или подключается существующая база данных user_database.db.
2. **Создание таблицы на основе пользовательского ввода:**
   - Пользователь вводит название таблицы и затем последовательно вводит названия полей и типы данных для каждого поля (например, TEXT, INTEGER, и т.д.).
   - Программа завершает ввод полей при вводе пустой строки. После этого сгенерированный SQL-запрос создает таблицу в базе данных с указанными полями.
3. **Ввод данных в таблицу:**
   - После создания таблицы программа предлагает пользователю ввести данные для каждого из полей таблицы. Ввод продолжается до тех пор, пока пользователь не решит остановиться (ввод пустой строки).
   - Данные вставляются в таблицу с использованием параметризованных запросов для защиты от SQL-инъекций.
4. **Вывод данных для проверки:**
   - После завершения ввода данных программа выполняет SQL-запрос для выборки всех данных из таблицы и выводит их на экран. Это позволяет пользователю убедиться, что данные были успешно добавлены в таблицу.
5. **Завершение работы программы:**
   - Все изменения сохраняются в базу данных, соединение закрывается, и программа завершает свою работу.
   -

Скриншоты программы с работоспособностью программы продемонстрированы на рисунке 1.1

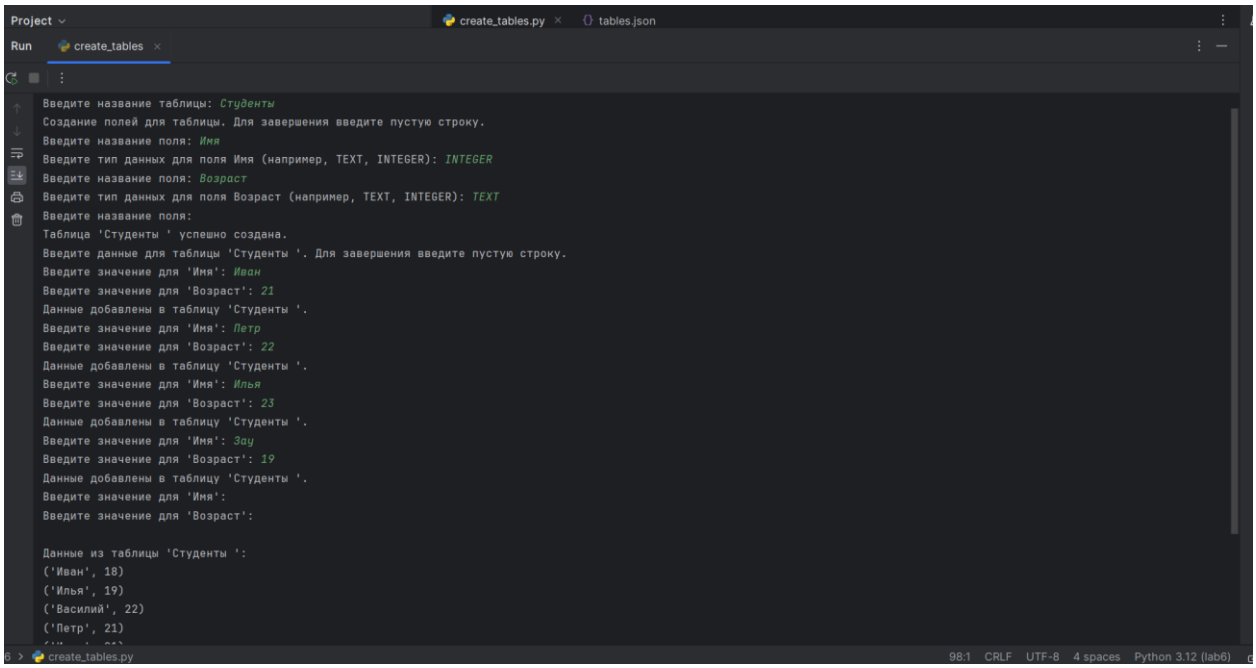

Рис.1.1

Вывод:

В результате выполнения работы была разработана программа, позволяющая пользователю создавать таблицы в базе данных SQLite с произвольными полями и типами данных, вводить данные в таблицу и выводить их для проверки. Программа демонстрирует основы работы с базами данных в Python, динамическое создание SQL-запросов на основе пользовательского ввода, а также безопасную вставку данных через параметризованные запросы. Знания, полученные в ходе работы, позволяют применять SQLite для хранения и обработки данных в различных проектах, где требуется работа с небольшими локальными базами данных.

Листинг программы:

```
import sqlite3

def create_table():
    # Подключение к базе данных SQLite
    conn = sqlite3.connect('user_database.db')
    cursor = conn.cursor()

    # Получение имени таблицы от пользователя
    table_name = input("Введите название таблицы: ")

    # Получение полей для таблицы
    columns = []
    print("Создание полей для таблицы. Для завершения введите пустую
```

```
строку.")
    while True:
        column_name = input("Введите название поля: ")
        if column_name == "":  # Пустая строка завершает ввод полей
            break
        column_type = input(f"Введите тип данных для поля {column_name}
(например, TEXT, INTEGER): ")
        columns.append(f"{column_name} {column_type}")

    # Формируем SQL-запрос для создания таблицы
    create_table_query = f"CREATE TABLE IF NOT EXISTS {table_name} ({',
'.join(columns)});"
    cursor.execute(create_table_query)
    print(f"Таблица '{table_name}' успешно создана.")

    # Сохранение изменений
    conn.commit()

    # Возвращаем курсор и подключение для дальнейшей работы
    return conn, cursor, table_name


def insert_data(cursor, table_name):
    # Получение данных для вставки
    print(f"Введите данные для таблицы '{table_name}'. Для завершения введите
пустую строку.")

    while True:
        # Получаем информацию о столбцах таблицы
        column_names_query = f"PRAGMA table_info({table_name});"
        cursor.execute(column_names_query)
        columns_info = cursor.fetchall()
        column_names = [column[1] for column in columns_info]  # Извлекаем
названия колонок

        values = []
        for column_name in column_names:
            value = input(f"Введите значение для '{column_name}': ")
            values.append(value)

        # Если все введенные значения пусты, прерываем ввод
        if all(value == "" for value in values):
            break

        # Формирование SQL-запроса для вставки данных
        placeholders = ', '.join(['?' for _ in values])  # Создаем
плейсхолдеры для параметров
        insert_query = f"INSERT INTO {table_name} ({', '.join(column_names)})
VALUES ({placeholders})"

        # Выполнение SQL-запроса с передачей значений для вставки
        cursor.execute(insert_query, values)

        print(f"Данные добавлены в таблицу '{table_name}'.")


def display_table_data(cursor, table_name):
    # Вывод всех данных из таблицы
    select_query = f"SELECT * FROM {table_name};"
    cursor.execute(select_query)
    rows = cursor.fetchall()

    if rows:
        print(f"\nДанные из таблицы '{table_name}':")
        for row in rows:
```

```
            print(row)
    else:
        print(f"\nТаблица '{table_name}' пуста.")


def main():
    # Создание таблицы
    conn, cursor, table_name = create_table()

    # Ввод данных в таблицу
    insert_data(cursor, table_name)

    # Сохранение и закрытие соединения
    conn.commit()

    # Вывод данных из таблицы
    display_table_data(cursor, table_name)

    conn.close()

    print("Программа завершена.")


if __name__ == "__main__":
    main()
```