Продолжаем работу над проектом "Сайт фитнес клуба «BODY\_FIT»". В первой части задания вам нужно будет оформить полное создание базы данных через SQL: таблицы, ограничения, заполнение тестовыми данными и добавление индексов. Всё должно быть обернуто в транзакцию с отключением и последующим включением проверки внешних ключей. Во второй части вам предстоит перейти к использованию библиотеки `sqlite3` в Python: написать функции для создания новой записи и поиска существующих, работая с базой через код.'

hw\_theme:

  - sqlite

  - CREATE

  - INSERT

  - UPDATE

  - sqlite3

  - python

  - транзакции

  - индексы

**### Технологии: 🛠️**

- SQL (DDL, транзакции, индексы)

- SQLite

- Python

- Библиотека `sqlite3`

**### Часть 1: Оформление структуры базы данных**

Создайте SQL-скрипт, который позволит восстановить всю базу данных. В начале файла обязательно выключите проверку внешних ключей командой `PRAGMA foreign\_keys = OFF;`, затем откройте транзакцию через `BEGIN TRANSACTION;`. Внутри опишите создание всех таблиц, внешних ключей и вставку тестовых данных.

После этого добавьте индексы: не менее двух обычных и двух составных. Рядом с каждой командой создания индекса оставьте комментарий, где объясните, почему выбранные поля и как это ускорит работу базы. В завершение закройте транзакцию через `COMMIT;` и снова включите проверку внешних ключей командой `PRAGMA foreign\_keys = ON;`.

Все запросы в файле должны быть корректно завершены точкой с запятой `;`, без исключений.

**### Часть 2: Работа с базой данных через Python (`sqlite3`)**

Создайте Python-скрипт. В начале файла импортируйте библиотеку `sqlite3` и укажите две константы: путь к файлу базы данных и путь к SQL-файлу со структурой.

Далее реализуйте следующие функции:

- **\*\*`read\_sql\_file(filepath: str) -> str`\*\*** — читает текст SQL-скрипта из файла и возвращает его содержимое.

- **\*\*`execute\_script(conn, script: str) -> None`\*\*** — принимает соединение и текст скрипта, создаёт курсор, выполняет скрипт через метод `executescript`, сохраняет изменения.

- **\*\*`find\_appointment\_by\_phone(conn, phone: str) -> list[tuple]`\*\*** — принимает соединение и номер телефона, выполняет параметризованный SELECT-запрос на точное совпадение номера телефона, возвращает список найденных записей. В записях человекочитаемые имена мастеров и названия услуг.

- **\*\*`find\_appointment\_by\_comment(conn, comment\_part: str) -> list[tuple]`\*\*** — принимает соединение и часть комментария, ищет записи, где комментарий содержит переданную строку, используя оператор `LIKE`, возвращает список найденных записей. В записях человекочитаемые имена мастеров и названия услуг.

- **\*\*`create\_appointment(conn, client\_name: str, client\_phone: str, trainer\_name: str, workouts\_list: list[str], comment: str = None) -> int`\*\*** — создаёт новую запись в таблице клиентов, принимает имя клиента, телефон, имя тренера и список тренировок. Ищет тренера и тренировки по именам, вставляет запись в базу, связывает её с тренировками. Возвращает ID созданной записи.

**### Минимальные требования к сдаче**

- В архиве должны быть два файла: `.sql` и `.py`. После распаковки архив должен позволять создать базу данных и выполнить все основные функции без ошибок.

- Все SQL-запросы внутри функций должны быть **\*\*параметризованными\*\***.

- В конце файла добавьте тестовые вызовы всех функций с выводом результатов через `print`, чтобы можно было увидеть их работу.

> **#### Критерии проверки 👍**

>

> 1. Структура базы данных оформлена корректно, все ограничения работают.

> 2. Индексы осмысленно добавлены и объяснены в комментариях.

> 3. Python-скрипт корректно работает с базой данных.

> 4. Все запросы в Python параметризованы.

> 5. Функции имеют аннотации типов.

> 6. Код соответствует стандартам PEP-8.