Exercise 6 (Date: Oct 05, 2025)

Exercise 1: Basic CRUD with Users Table

Requirement: Create SQLite database people.db, create table Users(name, email). Add some rows, update one row, delete one row and print the remaining results in ascending email order.

Exercise 2: Normalize & JOIN (Music model: Artist–Album–Track–Genre)

Requirement: Create 4 tables (Artist, Album, Genre, Track) with integer primary key and foreign key, add some records and query for song name, artist, album, genre.

Exercise 3: Repeated data entry: "INSERT or ignore" + foreign key lookup

Requirement: Given a list (artist, album, track, genre), write a program that does not duplicate artist/album/genre; the track record points to the correct foreign key.

Exercise 4: N-N Relationship (User–Course via Member table)

Requirement: Build 3 tables User(id,name,email), Course(id,title) and Member(user_id,course_id,role) (dual primary key). Add data and query the list of students by each subject, sort by role DESC (1 = lecturer, 0 = student) then by name.

Exercise 5: Practice Exercise: Simple Statistics & Indexes

Requirement: With music.db database (**Exercise 2**),
a) Count the number of tracks by artist.
b) Get the Top 3 tracks with the highest count.
c) Create an index on Track(title) and compare the LIKE query time.

Exercise 6: Normalize "flat playlist" → 4 tables (Artist/Album/Genre/Track) + transaction

Requirement:

- Import the tuple (artist, album, title, genre, len, rating, count) playlist into the normalized database.
- Do not duplicate logical records; use transaction for safe/fast batch insertion.

Exercise 7: JOIN multiple tables + conditional sorting (instructor first)

Requirement:

- N–N model: User–Member–Course.
- Query the list of classes by subject: lecturer (role=1) comes first, then students in alphabetical order.

Exercise 8: **Optimizing queries using Index + measuring time**

Requirement:

- Create 50k random records in Track(title) and measure the query time for LIKE '%Rock%' before/after creating index on title.
- Report the number of ms.

Exercise 9: **Data Constraints & UPSERT: Ensure Uniqueness by Pair**

Requirement:

- For the Album(title, artist_id) table, set UNIQUE(title, artist_id).
- Write an “upsert” function to add/update albums by (title, artist) to avoid duplicates and update when there are changes.

Exercise 10: **Foreign Keys + CASCADE + Transaction rollback**

Requirement:

- Turn on PRAGMA foreign_keys=ON.
- Define FK with ON DELETE CASCADE.
- Insert 1 artist, 1 album, 2 tracks → delete artist and prove track/album is deleted.
- Then try in transaction and ROLLBACK to restore.