**Step-by-Step Guide to Setting Up the SQLite Database**

**Step 1: Install SQLite (if needed)**

* SQLite is embedded in Python, so you don’t need to install anything extra. If you do not have it installed, make sure you have Python 3.x which comes with SQLite support by default.

**Step 2: Import SQLite Library**

* You start by importing the SQLite library, which provides the functions needed to interact with the SQLite database. This is done with the following import statement:

import sqlite3

**Step 3: Create a Database Connection**

* To interact with the SQLite database, you'll need to create a connection. If the database does not exist, SQLite will create it for you. In this case, the database is named healthcare.db.

# connect to sqlite database

conn = sqlite3.connect('healthcare.db')

**Step 4: Create a Cursor Object**

* Once connected to the database, you need to create a cursor. The cursor is used to execute SQL queries.

cursor = conn.cursor()

**Step 5: Create the Users Table**

* You are creating a table for storing user information, such as username, password, and role. The role will define the type of user (e.g., doctor, admin, nurse). The user\_id is the primary key, and it auto-increments whenever a new user is added.

cursor.execute("""

CREATE TABLE IF NOT EXISTS users (

user\_id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL UNIQUE,

password TEXT NOT NULL,

role TEXT NOT NULL

);

""")

* IF NOT EXISTS ensures that if the table already exists, it won't be recreated.
* user\_id is an auto-incrementing primary key to uniquely identify each user.
* username and password are required fields and cannot be NULL.
* role is also required and indicates the type of user (e.g., doctor, admin).

**Step 6: Create the Patients Table**

* This table will store information about patients, including name, age, gender, contact, and medical\_history. The id is the primary key.

cursor.execute("""

CREATE TABLE IF NOT EXISTS patients (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

age INTEGER,

gender TEXT,

contact TEXT,

medical\_history TEXT

);

""")

* id is the primary key that uniquely identifies a patient.
* name is a required field, while age, gender, contact, and medical\_history are optional but helpful for storing patient data.

**Step 7: Create the Appointments Table**

* This table will track appointments. It will link a patient\_id to a user\_id (which refers to a doctor or user handling the appointment), along with the appointment\_time.

cursor.execute("""

CREATE TABLE IF NOT EXISTS appointments (

id INTEGER PRIMARY KEY AUTOINCREMENT,

patient\_id INTEGER NOT NULL,

user\_id INTEGER,

appointment\_time TEXT,

FOREIGN KEY (patient\_id) REFERENCES patients (id)

);

""")

* id is the primary key for each appointment.
* patient\_id is a foreign key that links to the patients table.
* user\_id refers to the user (e.g., a doctor or nurse) handling the appointment.
* appointment\_time stores the scheduled time for the appointment.

**Step 8: Commit Changes and Close Connection**

* Once all the tables are created, you should commit the changes to ensure they are saved. After committing, you can close the connection to the database.

conn.commit()

conn.close()

This ensures all the changes are saved and the connection is properly closed.

**Step 9: Print Confirmation Message**

* Finally, you can print a success message to confirm that the SQLite database and tables were created successfully.

print("SQLite database and tables created successfully")

After running the code:

1. The healthcare.db database file will be created (or opened if it already exists).
2. The tables (users, patients, appointments) will be created if they do not already exist.
3. You will have a structured SQLite database ready for storing data on users, patients, and appointments.

This setup gives you a starting point for your healthcare system, allowing you to store and manage relevant data like users, patient records, and appointments.