import sqlite3

def connect():

conn = sqlite3.connect("hostel.db")

return conn

def create\_tables():

conn = connect()

cursor = conn.cursor()

# Students table

cursor.execute("""

CREATE TABLE IF NOT EXISTS students (

id TEXT PRIMARY KEY,

name TEXT,

age INTEGER,

room TEXT

)

""")

# Rooms table

cursor.execute("""

CREATE TABLE IF NOT EXISTS rooms (

room\_no TEXT PRIMARY KEY,

capacity INTEGER,

occupants INTEGER DEFAULT 0

)

""")

# Payments table

cursor.execute("""

CREATE TABLE IF NOT EXISTS payments (

payment\_id INTEGER PRIMARY KEY AUTOINCREMENT,

student\_id TEXT,

amount REAL,

date TEXT

)

""")

# Insert rooms 1-100 if not exists

for i in range(1, 101):

cursor.execute("INSERT OR IGNORE INTO rooms (room\_no, capacity, occupants) VALUES (?, 2, 0)", (str(i),))

conn.commit()

conn.close()

def assign\_room():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT room\_no FROM rooms WHERE occupants < capacity ORDER BY room\_no ASC LIMIT 1")

room = cursor.fetchone()

conn.close()

if room:

return room[0]

else:

return None

def add\_student(id, name, age):

conn = connect()

cursor = conn.cursor()

room = assign\_room()

if not room:

raise Exception("No available rooms!")

try:

cursor.execute("INSERT INTO students (id, name, age, room) VALUES (?, ?, ?, ?)", (id, name, age, room))

cursor.execute("UPDATE rooms SET occupants = occupants + 1 WHERE room\_no = ?", (room,))

conn.commit()

except sqlite3.IntegrityError:

raise Exception("Student ID already exists!")

conn.close()

def delete\_student(student\_id):

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT room FROM students WHERE id = ?", (student\_id,))

room = cursor.fetchone()

if room:

cursor.execute("DELETE FROM students WHERE id = ?", (student\_id,))

cursor.execute("UPDATE rooms SET occupants = occupants - 1 WHERE room\_no = ?", (room[0],))

conn.commit()

conn.close()

def view\_students():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students")

rows = cursor.fetchall()

conn.close()

return rows

def search\_student(keyword):

conn = connect()

cursor = conn.cursor()

cursor.execute("""

SELECT \* FROM students WHERE

id LIKE ? OR name LIKE ? OR age LIKE ? OR room LIKE ?

""", (f'%{keyword}%', f'%{keyword}%', f'%{keyword}%', f'%{keyword}%'))

rows = cursor.fetchall()

conn.close()

return rows

def add\_room(room\_no, capacity):

conn = connect()

cursor = conn.cursor()

cursor.execute("INSERT INTO rooms (room\_no, capacity, occupants) VALUES (?, ?, 0)", (room\_no, capacity))

conn.commit()

conn.close()

def delete\_room(room\_no):

conn = connect()

cursor = conn.cursor()

cursor.execute("DELETE FROM rooms WHERE room\_no = ?", (room\_no,))

conn.commit()

conn.close()

def view\_rooms():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM rooms")

rows = cursor.fetchall()

conn.close()

return rows

def search\_room(keyword):

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM rooms WHERE room\_no LIKE ?", (f'%{keyword}%',))

rows = cursor.fetchall()

conn.close()

return rows

def add\_payment(student\_id, amount, date):

conn = connect()

cursor = conn.cursor()

cursor.execute("INSERT INTO payments (student\_id, amount, date) VALUES (?, ?, ?)", (student\_id, amount, date))

conn.commit()

conn.close()

def delete\_payment(payment\_id):

conn = connect()

cursor = conn.cursor()

cursor.execute("DELETE FROM payments WHERE payment\_id = ?", (payment\_id,))

conn.commit()

conn.close()

def view\_payments():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM payments")

rows = cursor.fetchall()

conn.close()

return rows

def search\_payment(keyword):

conn = connect()

cursor = conn.cursor()

cursor.execute("""

SELECT \* FROM payments WHERE

student\_id LIKE ? OR amount LIKE ? OR date LIKE ?

""", (f'%{keyword}%', f'%{keyword}%', f'%{keyword}%'))

rows = cursor.fetchall()

conn.close()

return rows