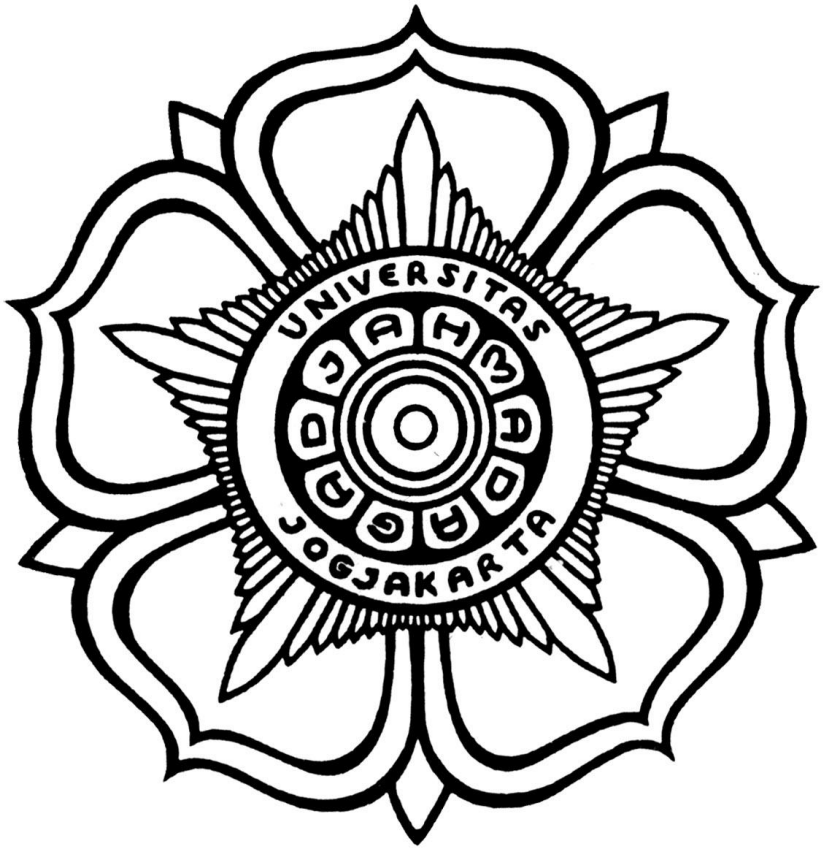
Week 3 - Basic CRUD Operations

Clinic Appointment System



Kelompok 3

Chelsea Natasja Jesslyne Sembiring (24/543571/PA/23097)

Miera Ardiyanti Farica (24/534436/PA/22665)

KOM A

1. SQL Schema

```
CREATE TABLE doctor
(
  doctorID INT NOT NULL,
  firstName VARCHAR(100) NOT NULL,
  lastName VARCHAR(100) NOT NULL,
  specialization VARCHAR(100) NOT NULL,
  PRIMARY KEY (doctorID)
);

CREATE TABLE patient
(
  patientID INT NOT NULL,
  firstName VARCHAR(100) NOT NULL,
  lastName VARCHAR(100) NOT NULL,
  emailAddress VARCHAR(255) NOT NULL,
  PRIMARY KEY (patientID)
);

CREATE TABLE clinic
(
  clinicID INT NOT NULL,
  clinicName VARCHAR(255) NOT NULL,
  city VARCHAR(100) NOT NULL,
  address VARCHAR(255) NOT NULL,
  PRIMARY KEY (clinicID)
);

CREATE TABLE room
(
  roomID INT NOT NULL,
  roomName VARCHAR(100) NOT NULL,
  capacity INT NOT NULL,
  clinicID INT NOT NULL,
  PRIMARY KEY (roomID),
```

```
  FOREIGN KEY (clinicID) REFERENCES clinic(clinicID)
);

CREATE TABLE doctor_clinic
(
  doctorID INT NOT NULL,
  clinicID INT NOT NULL,
  PRIMARY KEY (doctorID, clinicID),
  FOREIGN KEY (doctorID) REFERENCES doctor(doctorID),
  FOREIGN KEY (clinicID) REFERENCES clinic(clinicID)
);

CREATE TABLE doctor_phoneNumber
(
  phoneNumber VARCHAR(50) NOT NULL,
  doctorID INT NOT NULL,
  PRIMARY KEY (phoneNumber, doctorID),
  FOREIGN KEY (doctorID) REFERENCES doctor(doctorID)
);

CREATE TABLE patient_phoneNumber
(
  phoneNumber VARCHAR(50) NOT NULL,
  patientID INT NOT NULL,
  PRIMARY KEY (phoneNumber, patientID),
  FOREIGN KEY (patientID) REFERENCES patient(patientID)
);

CREATE TABLE appointment
(
  appointmentID INT NOT NULL,
  date DATE NOT NULL,
  time TIME NOT NULL,
  status VARCHAR(50) NOT NULL,
  patientID INT NOT NULL,
  doctorID INT NOT NULL,
```

```
  roomID INT NOT NULL,
  PRIMARY KEY (appointmentID),
  FOREIGN KEY (patientID) REFERENCES patient(patientID),
  FOREIGN KEY (doctorID) REFERENCES doctor(doctorID),
  FOREIGN KEY (roomID) REFERENCES room(roomID)
);
```

2. Basic CRUD Operations

- **Doctor Entity**
  - Create (Add a new doctor)

```
INSERT INTO doctor (doctorID, firstName, lastName,
specialization)
VALUES
    (1, 'Chelsea', 'Natasja', 'Surgeon');
```

  - Read (View doctor details)

```
// View a specific doctor by their ID
SELECT * FROM doctor WHERE doctorID = 1;

// View all doctors
SELECT * FROM doctor;
```

  - Update (Change a doctor's specialization)

```
UPDATE
    doctor
SET
    specialization = 'Pediatrics'
WHERE
    doctorID = 1;
```

  - Delete (Remove a doctor)

```
DELETE FROM
    doctor
WHERE
```

```
    doctorID = 1;
```

- **Patient Entity**
  - Create (Add a new patient)

```
INSERT INTO patient (patientID, firstName, lastName,
emailAddress)
VALUES
    (101, 'Siti', 'Aminah', 'siti.a@example.com');
```

  - Read (View patient details)

```
// View a specific patient by their ID
SELECT * FROM patient WHERE patientID = 101;


// View all patients
SELECT * FROM patient;
```

  - Update (Change a patient's email)

```
UPDATE
    patient
SET
    emailAddress = 'siti.aminah@newdomain.com'
WHERE
    patientID = 101;
```

  - Delete (Remove a patient)

```
DELETE FROM
    patient
WHERE
    patientID = 101;
```

- **Appointment Entity**
  - Create (Schedule a new appointment)

```
INSERT INTO appointment (appointmentID, date, time, status,
patientID, doctorID, roomID)
VALUES
     (1001, '2025-11-20', '10:00:00', 'Scheduled', 101, 1, 12);
```

- Read (View appointment details)

```
// View a specific appointment
SELECT * FROM appointment WHERE appointmentID = 1001;

// View all appointments for a specific patient
SELECT * FROM appointment WHERE patientID = 101;

// View all appointments for a specific doctor
SELECT * FROM appointment WHERE doctorID = 1;
```

- Update (Change appointment status or time)

```
// Example: Mark as 'Completed'
UPDATE appointment
SET status = 'Completed'
WHERE appointmentID = 1001;

// Example: Reschedule
UPDATE appointment
SET date = '2025-11-21', time = '11:00:00'
WHERE appointmentID = 1001;
```

- Delete (Cancel an appointment)

```
DELETE FROM
    appointment
WHERE
    appointmentID = 1001;
```

- **Clinic & Room Entities**
  - Create (Add a new clinic)

```
INSERT INTO clinic (clinicID, clinicName, city, address)
VALUES
    (1, 'Klinik Sehat Utama', 'Yogyakarta', 'Jl. Malioboro 1');
```

- Create (Add a room to that clinic)

```
INSERT INTO room (roomID, roomName, capacity, clinicID)
VALUES
    (12, 'Ruang Periksa 1', 1, 1);
```

- Read (View all clinic informations in Yogyakarta)

```
SELECT *
FROM
    clinic
WHERE
    city = 'Yogyakarta';
```

- Update (Change room capacity)

```
UPDATE
    room
SET
    capacity = 2
WHERE
    roomID = 12;
```

- Delete (Remove a room)

```
DELETE FROM
    room
WHERE
    roomID = 12;
```

- **Multivalued and Relationship Attributes**
  - doctor_clinic: Create (Assign a doctor to a clinic)

```
INSERT INTO doctor_clinic (doctorID, clinicID)
VALUES
    (1, 1);
```

  - patient_phoneNumber: Create (Add a phone number to a patient)

```
INSERT INTO patient_phoneNumber (patientID, phoneNumber)
VALUES
    (101, '08123456789');
```

  - doctor_clinic: Delete (Remove a doctor from a clinic)

```
DELETE FROM
    doctor_clinic
WHERE
    doctorID = 1
    AND clinicID = 1;
```

  - patient_phoneNumber: Delete (Remove a patient's phone number)

```
DELETE FROM
    patient_phoneNumber
WHERE
    patientID = 101
    AND phoneNumber = '08123456789';
```

Implementasi Kode CRUD dengan Python

Untuk memenuhi tugas Week 3 z'deliverable CRUD code", kami merancang query SQL dan juga membuat skrip aplikasi sederhana untuk mengeksekusi query tersebut. Ini berfungsi sebagai "backend" dasar operasi CRUD kami.

Tools yang Digunakan:

- **Bahasa:** Python 3
- **Database:** SQLite 3 (menggunakan library sqlite3 bawaan Python)
- **Web Framework:** Flask

1. **Inisiasi Database**

   Membuat skrip 'init_db.py' untuk membuat file database (clinic.db) dari skema SQL (clinic.sql) yang telah kami rancang.

```
import sqlite3

DATABASE_FILE = 'clinic.db'
SCHEMA_FILE = 'clinic.sql'

conn = sqlite3.connect(DATABASE_FILE)

print(f"Membaca skema dari {SCHEMA_FILE}...")

with open(SCHEMA_FILE) as f:
    schema_sql = f.read()

conn.executescript(schema_sql)

print(f"Database dan semua tabel berhasil dibuat di '{DATABASE_FILE}'")

conn.commit()
conn.close()
```

2. Pembuatan Skrip app.py

Membuat satu file utama bernama app.py yang berisi semua logika untuk operasi CRUD. Struktur file ini adalah sebagai berikut:

a) **Koneksi Database:** Dibuat sebuah fungsi *helper* get_db_conn() untuk terhubung ke file clinic.db dan mengatur row_factory agar hasil query lebih mudah dibaca.

b) **Fungsi CRUD per Entitas:** Menerjemahkan setiap query SQL dari rancangan ke dalam sebuah fungsi Python.

c) **Demonstrasi (Main Function):** Di bagian akhir skrip, digunakan blok if __name__ == "__main__": untuk memanggil fungsi-fungsi CRUD secara berurutan (Create, Read, Update, Delete) dan mencetak hasilnya ke terminal.

```
import sqlite3
import time

DATABASE_FILE = 'clinic.db'

def get_db_conn():
    """Membuka koneksi baru ke database SQLite."""
    conn = sqlite3.connect(DATABASE_FILE)
    conn.row_factory = sqlite3.Row
    return conn

# --- FUNGSI PRINT ---

def print_all_doctors():
    print("--- Isi Tabel Doctor ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM doctor")
    doctors = cursor.fetchall()
    if not doctors:
        print("  [Kosong]")
    for doc in doctors:
            print(f"   ID:  {doc['doctorID']},  Nama:  {doc['firstName']}
{doc['lastName']}, Spesialis: {doc['specialization']}")
    conn.close()
```

```
    print("---------------------------------")

def print_all_patients():
    print("--- Isi Tabel Patient ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM patient")
    patients = cursor.fetchall()
    if not patients:
        print("  [Kosong]")
    for p in patients:
              print(f"   ID:  {p['patientID']},  Nama:  {p['firstName']}
{p['lastName']}, Email: {p['emailAddress']}")
    conn.close()
    print("---------------------------------")

def print_all_appointments():
    print("--- Isi Tabel Appointment ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM appointment")
    apps = cursor.fetchall()
    if not apps:
        print("  [Kosong]")
    for app in apps:
                  print(f"    ID:   {app['appointmentID']},   PasienID:
{app['patientID']},      DokterID:      {app['doctorID']},      Status:
{app['status']}")
    conn.close()
    print("-------------------------------------")

def print_all_clinics():
    print("--- Isi Tabel Clinic ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM clinic")
    items = cursor.fetchall()
    if not items:
        print("  [Kosong]")
```

```
    for item in items:
          print(f"  ID: {item['clinicID']}, Nama: {item['clinicName']},
Kota: {item['city']}, Alamat: {item['address']}")
    conn.close()
    print("----------------------------------")

def print_all_rooms():
    print("--- Isi Tabel Room ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM room")
    items = cursor.fetchall()
    if not items:
        print("  [Kosong]")
    for item in items:
           print(f"   ID:  {item['roomID']},  Nama:  {item['roomName']},
Kapasitas: {item['capacity']}, ClinicID: {item['clinicID']}")
    conn.close()
    print("----------------------------------")

def print_all_doctor_clinics():
    print("--- Isi Tabel doctor_clinic (Relasi) ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM doctor_clinic")
    items = cursor.fetchall()
    if not items:
        print("  [Kosong]")
    for item in items:
                print(f"    DokterID:   {item['doctorID']},   ClinicID:
{item['clinicID']}")
    conn.close()
    print("----------------------------------")

def print_all_doctor_phones():
    print("--- Isi Tabel doctor_phoneNumber ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM doctor_phoneNumber")
```

```
    items = cursor.fetchall()
    if not items:
        print("  [Kosong]")
    for item in items:
                print(f"    DokterID:   {item['doctorID']},   No.Telp:
{item['phoneNumber']}")
    conn.close()
    print("-----------------------------------")

def print_all_patient_phones():
    print("--- Isi Tabel patient_phoneNumber ---")
    conn = get_db_conn()
    cursor = conn.execute("SELECT * FROM patient_phoneNumber")
    items = cursor.fetchall()
    if not items:
        print("  [Kosong]")
    for item in items:
                print(f"    PasienID:   {item['patientID']},   No.Telp:
{item['phoneNumber']}")
    conn.close()
    print("-----------------------------------")

# --- FUNGSI CRUD: DOCTOR ENTITY ---

def add_doctor(doctor_id, first_name, last_name, specialization):
    print(f"[C] Menambahkan dokter: {first_name} {last_name}")
    conn = get_db_conn()
    conn.execute(
              "INSERT   INTO   doctor   (doctorID,   firstName,   lastName,
specialization) VALUES (?, ?, ?, ?)",
        (doctor_id, first_name, last_name, specialization)
    )
    conn.commit()
    conn.close()

def get_doctor_by_id(doctor_id):
```

```
    print(f"[R] Mencari dokter ID: {doctor_id}")
    conn = get_db_conn()
     doc = conn.execute("SELECT * FROM doctor WHERE doctorID = ?",
(doctor_id,)).fetchone()
    conn.close()
    return doc

def update_doctor_specialization(doctor_id, new_specialization):
     print(f"[U] Mengupdate spesialisasi dokter ID {doctor_id} menjadi
'{new_specialization}'")
    conn = get_db_conn()
    conn.execute(
        "UPDATE doctor SET specialization = ? WHERE doctorID = ?",
        (new_specialization, doctor_id)
    )
    conn.commit()
    conn.close()

def delete_doctor(doctor_id):
    # print(f"[D] Menghapus dokter ID: {doctor_id}")
    conn = get_db_conn()
    conn.execute("DELETE FROM doctor WHERE doctorID = ?", (doctor_id,))
    conn.commit()
    conn.close()

# --- FUNGSI CRUD: PATIENT ENTITY ---

def add_patient(patient_id, first_name, last_name, email):
    print(f"[C] Menambahkan pasien: {first_name} {last_name}")
    conn = get_db_conn()
    conn.execute(
             "INSERT INTO patient (patientID, firstName, lastName,
emailAddress) VALUES (?, ?, ?, ?)",
        (patient_id, first_name, last_name, email)
    )
    conn.commit()
```

```
    conn.close()

def get_patient_by_id(patient_id):
    print(f"[R] Mencari pasien ID: {patient_id}")
    conn = get_db_conn()
    patient = conn.execute("SELECT * FROM patient WHERE patientID = ?",
(patient_id,)).fetchone()
    conn.close()
    return patient

def update_patient_email(patient_id, new_email):
      print(f"[U] Mengupdate email pasien ID {patient_id} menjadi
'{new_email}'")
    conn = get_db_conn()
    conn.execute(
        "UPDATE patient SET emailAddress = ? WHERE patientID = ?",
        (new_email, patient_id)
    )
    conn.commit()
    conn.close()

def delete_patient(patient_id):
    # print(f"[D] Menghapus pasien ID: {patient_id}")
    conn = get_db_conn()
    conn.execute("DELETE FROM patient WHERE patientID = ?", (patient_id,))
    conn.commit()
    conn.close()

# --- FUNGSI CRUD: APPOINTMENT ENTITY ---

def add_appointment(app_id, date, time, status, patient_id, doctor_id,
room_id):
    print(f"[C] Menjadwalkan appointment ID: {app_id}")
    conn = get_db_conn()
    conn.execute(
```

```
          "INSERT INTO appointment (appointmentID, date, time, status,
patientID, doctorID, roomID) VALUES (?, ?, ?, ?, ?, ?, ?)",
        (app_id, date, time, status, patient_id, doctor_id, room_id)
    )
    conn.commit()
    conn.close()

def update_appointment_status(app_id, new_status):
      print(f"[U] Mengupdate status appointment ID {app_id} menjadi
'{new_status}'")
    conn = get_db_conn()
    conn.execute(
        "UPDATE appointment SET status = ? WHERE appointmentID = ?",
        (new_status, app_id)
    )
    conn.commit()
    conn.close()

def delete_appointment(app_id):
    # print(f"[D] Membatalkan (menghapus) appointment ID: {app_id}")
    conn = get_db_conn()
     conn.execute("DELETE FROM appointment WHERE appointmentID = ?",
(app_id,))
    conn.commit()
    conn.close()

# --- FUNGSI CRUD: CLINIC & ROOM ENTITIES ---

def add_clinic(clinic_id, name, city, address):
    print(f"[C] Menambahkan klinik: {name}")
    conn = get_db_conn()
    conn.execute(
        "INSERT INTO clinic (clinicID, clinicName, city, address) VALUES
(?, ?, ?, ?)",
        (clinic_id, name, city, address)
    )
```

```
    conn.commit()
    conn.close()

def add_room(room_id, name, capacity, clinic_id):
    print(f"[C] Menambahkan ruang: {name} ke klinik ID {clinic_id}")
    conn = get_db_conn()
    conn.execute(
        "INSERT INTO room (roomID, roomName, capacity, clinicID) VALUES
(?, ?, ?, ?)",
        (room_id, name, capacity, clinic_id)
    )
    conn.commit()
    conn.close()

def update_room_capacity(room_id, new_capacity):
      print(f"[U]  Mengupdate  kapasitas  ruang  ID  {room_id}  menjadi
{new_capacity}")
    conn = get_db_conn()
    conn.execute(
        "UPDATE room SET capacity = ? WHERE roomID = ?",
        (new_capacity, room_id)
    )
    conn.commit()
    conn.close()

def delete_room(room_id):
    # print(f"[D] Menghapus ruang ID: {room_id}")
    conn = get_db_conn()
    conn.execute("DELETE FROM room WHERE roomID = ?", (room_id,))
    conn.commit()
    conn.close()

# --- FUNGSI CRUD: RELATIONSHIP & MULTIVALUED ---

def add_doctor_to_clinic(doctor_id, clinic_id):
```

```
      print(f"[C] Menghubungkan dokter ID {doctor_id} ke klinik ID {clinic_id}")
    conn = get_db_conn()
    conn.execute(
        "INSERT INTO doctor_clinic (doctorID, clinicID) VALUES (?, ?)",
        (doctor_id, clinic_id)
    )
    conn.commit()
    conn.close()

def add_phone_to_patient(patient_id, phone_number):
    print(f"[C] Menambahkan nomor telepon ke pasien ID {patient_id}")
    conn = get_db_conn()
    conn.execute(
        "INSERT INTO patient_phoneNumber (patientID, phoneNumber) VALUES (?, ?)",
        (patient_id, phone_number)
    )
    conn.commit()
    conn.close()

def remove_doctor_from_clinic(doctor_id, clinic_id):
    # print(f"[D] Menghapus hubungan dokter ID {doctor_id} dari klinik ID {clinic_id}")
    conn = get_db_conn()
    conn.execute(
        "DELETE FROM doctor_clinic WHERE doctorID = ? AND clinicID = ?",
        (doctor_id, clinic_id)
    )
    conn.commit()
    conn.close()

def remove_phone_from_patient(patient_id, phone_number):
    # print(f"[D] Menghapus nomor telepon pasien ID {patient_id}")
    conn = get_db_conn()
    conn.execute(
```

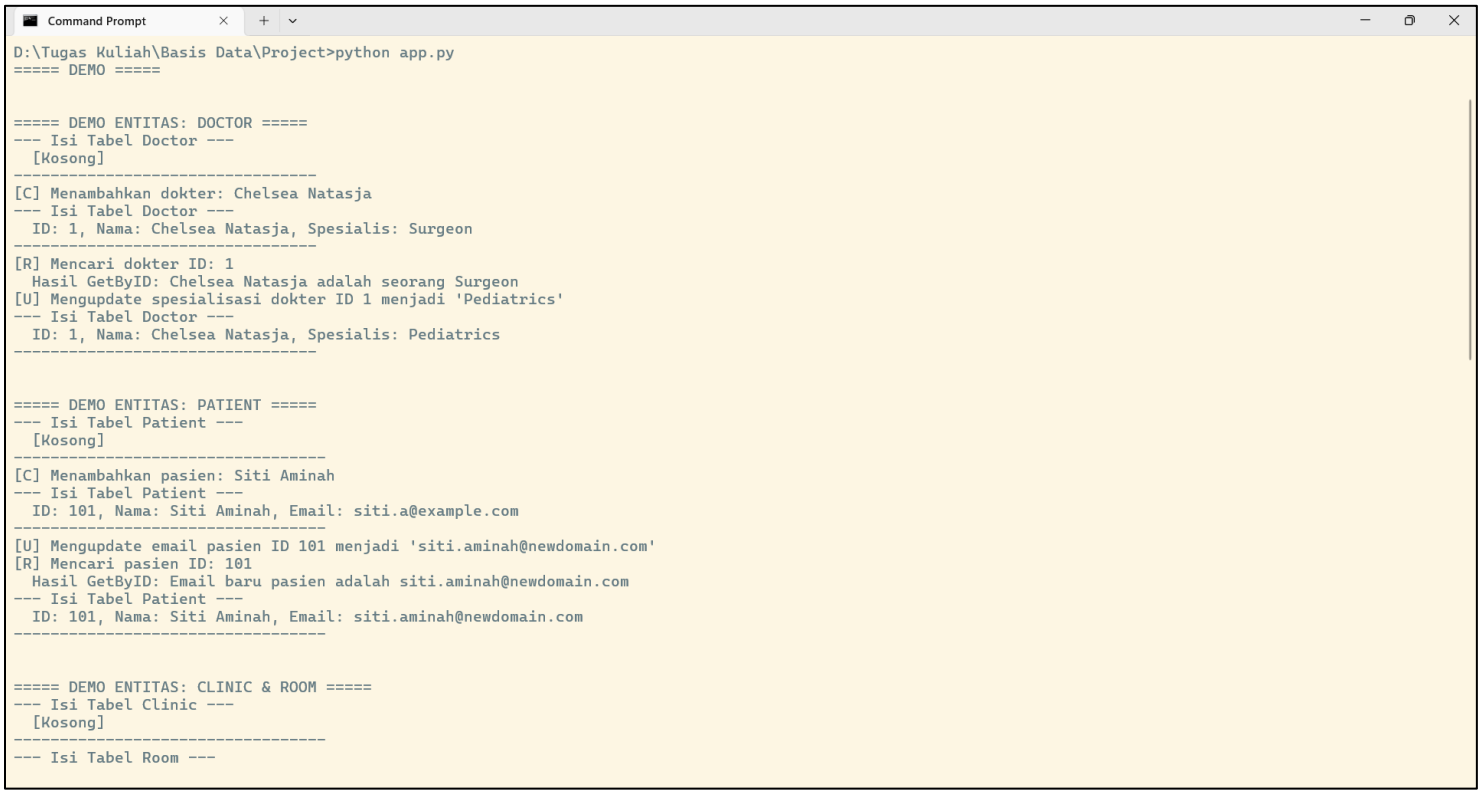
```
            "DELETE FROM patient_phoneNumber WHERE patientID = ? AND
phoneNumber = ?",
            (patient_id, phone_number)
        )
        conn.commit()
        conn.close()

# --- BAGIAN UTAMA ---

if __name__ == "__main__":

    print("===== DEMO =====")
    time.sleep(2)

    # --- DEMO DOCTOR ---
    print("\n\n===== DEMO ENTITAS: DOCTOR =====")
    print_all_doctors() # Tampilkan awal (kosong)
    add_doctor(1, 'Chelsea', 'Natasja', 'Surgeon')
    print_all_doctors() # Tampilkan setelah C
    doc = get_doctor_by_id(1)
    print(f"  Hasil GetByID: {doc['firstName']} {doc['lastName']} adalah
seorang {doc['specialization']}")
    update_doctor_specialization(1, 'Pediatrics')
    print_all_doctors() # Tampilkan setelah U

    # --- DEMO PATIENT ---
    print("\n\n===== DEMO ENTITAS: PATIENT =====")
    print_all_patients() # Tampilkan awal
    add_patient(101, 'Siti', 'Aminah', 'siti.a@example.com')
    print_all_patients() # Tampilkan setelah C
    update_patient_email(101, 'siti.aminah@newdomain.com')
    pat = get_patient_by_id(101)
        print(f"    Hasil   GetByID:   Email   baru   pasien   adalah
{pat['emailAddress']}")
    print_all_patients() # Tampilkan setelah U
```

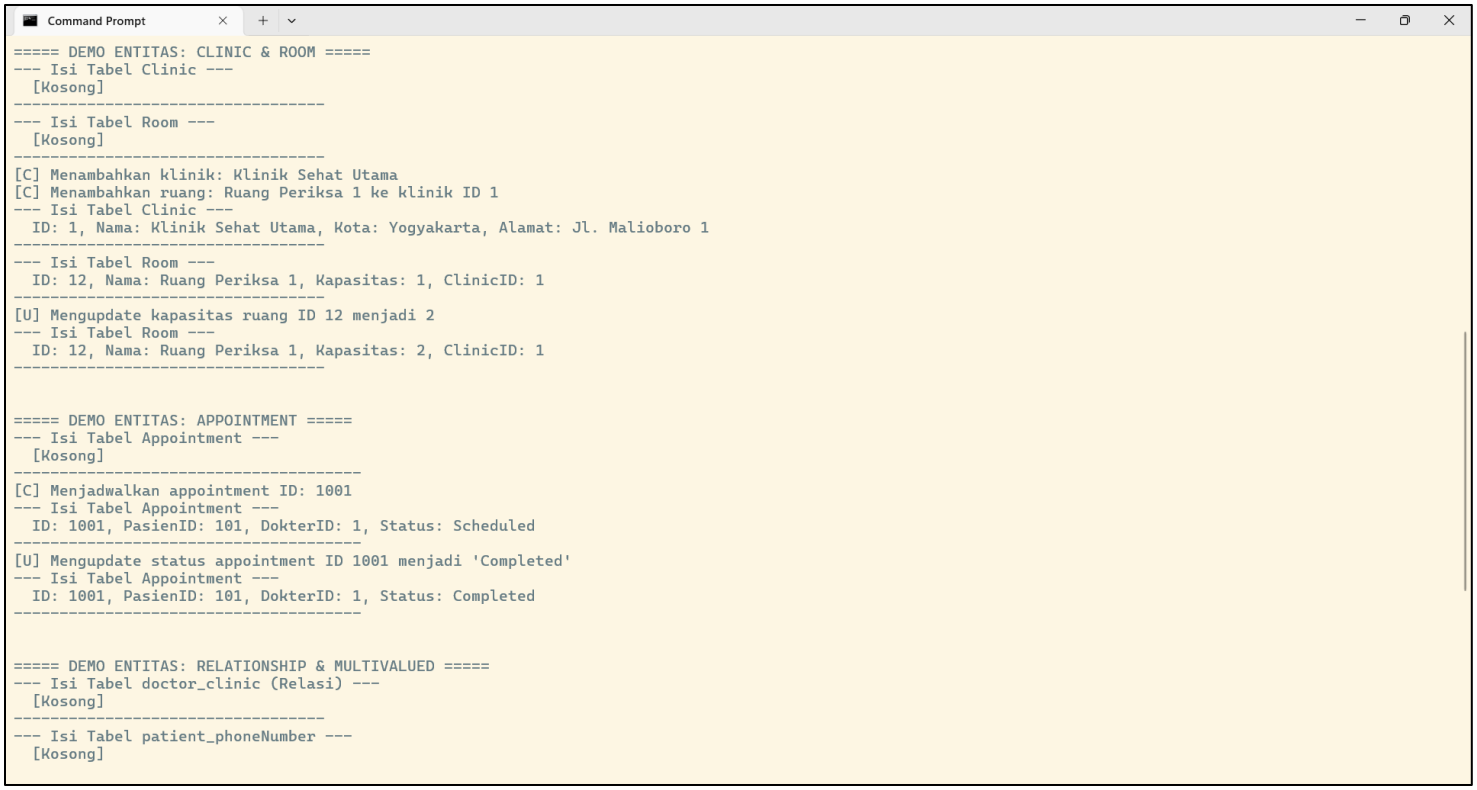
```
    # --- DEMO CLINIC & ROOM ---
    print("\n\n===== DEMO ENTITAS: CLINIC & ROOM =====")
    print_all_clinics() # Tampilkan awal
    print_all_rooms() # Tampilkan awal
    add_clinic(1, 'Klinik Sehat Utama', 'Yogyakarta', 'Jl. Malioboro 1')
    add_room(12, 'Ruang Periksa 1', 1, 1)
    print_all_clinics() # Tampilkan setelah C
    print_all_rooms() # Tampilkan setelah C
    update_room_capacity(12, 2)
    print_all_rooms() # Tampilkan setelah U

    # --- DEMO APPOINTMENT ---
    print("\n\n===== DEMO ENTITAS: APPOINTMENT =====")
    print_all_appointments() # Tampilkan awal
    add_appointment(1001, '2025-11-20', '10:00:00', 'Scheduled', 101, 1,
12)
    print_all_appointments() # Tampilkan setelah C
    update_appointment_status(1001, 'Completed')
    print_all_appointments() # Tampilkan setelah U

    # --- DEMO RELATIONSHIP & MULTIVALUED ---
    print("\n\n===== DEMO ENTITAS: RELATIONSHIP & MULTIVALUED =====")
    print_all_doctor_clinics() # Tampilkan awal
    print_all_patient_phones() # Tampilkan awal
    add_doctor_to_clinic(1, 1)
    add_phone_to_patient(101, '08123456789')
    print("  [C] Data relasi & multivalued berhasil ditambahkan.")
    print_all_doctor_clinics() # Tampilkan setelah C
    print_all_patient_phones() # Tampilkan setelah C

    # --- CLEANUP ---
    print("\n\n===== CLEANUP DATA DEMO =====")

    remove_phone_from_patient(101, '08123456789')
    remove_doctor_from_clinic(1, 1)
    delete_appointment(1001)
```

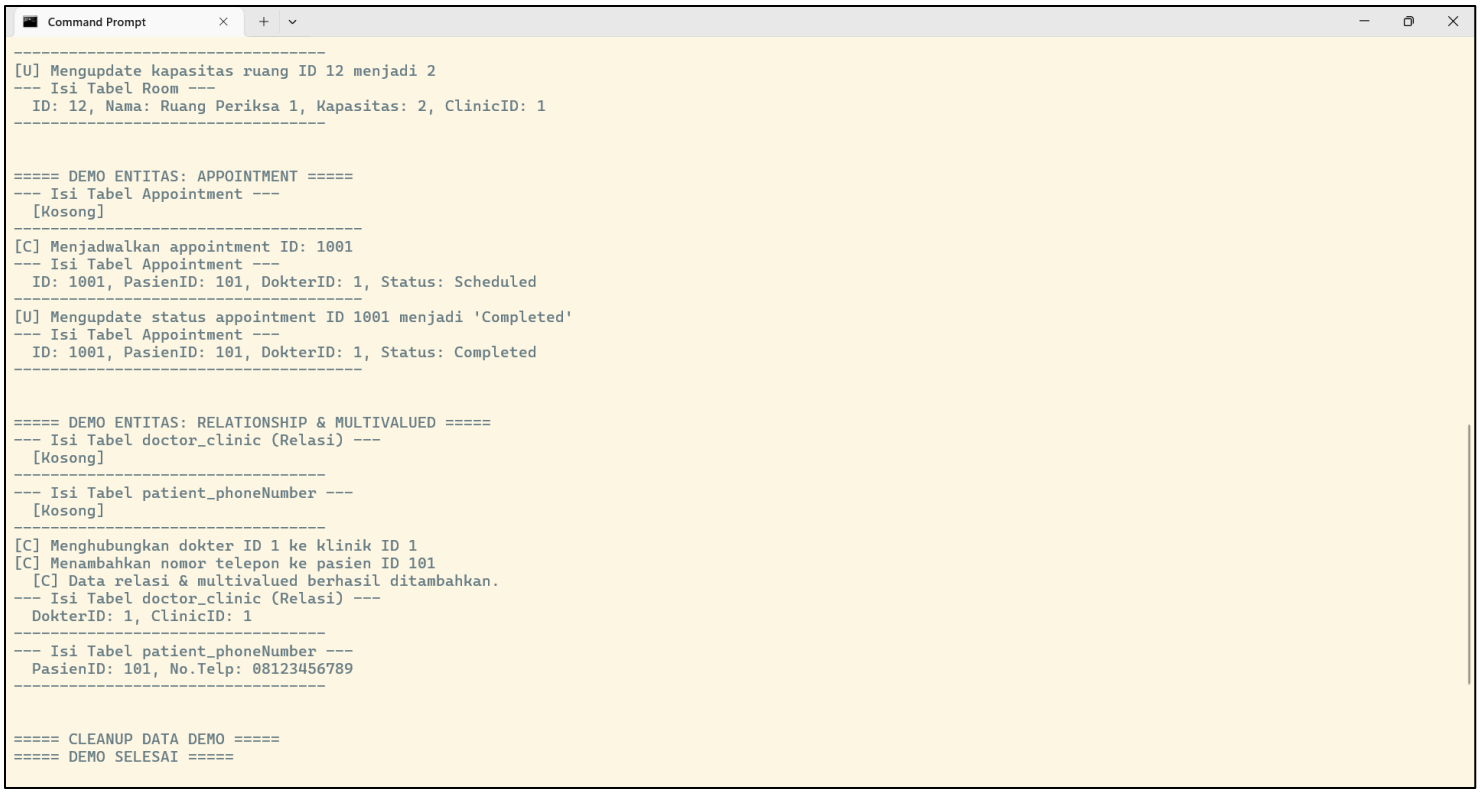
```
    delete_patient(101)
    delete_doctor(1)
    delete_room(12)

    conn = get_db_conn()
    conn.execute("DELETE FROM clinic WHERE clinicID = 1")
    conn.commit()
    conn.close()

    print("===== DEMO SELESAI =====")
```

3. Hasil dan Pengujian

Skrip app.py dijalankan melalui terminal menggunakan perintah python app.py. Output terminal membuktikan bahwa semua operasi CRUD (Create, Read, Update, Delete) berhasil dijalankan pada database.




Command Prompt
===== DEMO ENTITAS: CLINIC & ROOM =====
--- Isi Tabel Clinic ---
[Kosong]
--- Isi Tabel Room ---
[Kosong]
[C] Menambahkan klinik: Klinik Sehat Utama
[C] Menambahkan ruang: Ruang Periksa 1 ke klinik ID 1
--- Isi Tabel Clinic ---
ID: 1, Nama: Klinik Sehat Utama, Kota: Yogyakarta, Alamat: Jl. Malioboro 1
--- Isi Tabel Room ---
ID: 12, Nama: Ruang Periksa 1, Kapasitas: 1, ClinicID: 1
[U] Mengupdate kapasitas ruang ID 12 menjadi 2
--- Isi Tabel Room ---
ID: 12, Nama: Ruang Periksa 1, Kapasitas: 2, ClinicID: 1
===== DEMO ENTITAS: APPOINTMENT =====
--- Isi Tabel Appointment ---
[Kosong]
[C] Menjadwalkan appointment ID: 1001
--- Isi Tabel Appointment ---
ID: 1001, PasienID: 101, DokterID: 1, Status: Scheduled
[U] Mengupdate status appointment ID 1001 menjadi 'Completed'
--- Isi Tabel Appointment ---
ID: 1001, PasienID: 101, DokterID: 1, Status: Completed
===== DEMO ENTITAS: RELATIONSHIP & MULTIVALUED =====
--- Isi Tabel doctor_clinic (Relasi) ---
[Kosong]
--- Isi Tabel patient_phoneNumber ---
[Kosong]


Command Prompt
[U] Mengupdate kapasitas ruang ID 12 menjadi 2
--- Isi Tabel Room ---
ID: 12, Nama: Ruang Periksa 1, Kapasitas: 2, ClinicID: 1
===== DEMO ENTITAS: APPOINTMENT =====
--- Isi Tabel Appointment ---
[Kosong]
[C] Menjadwalkan appointment ID: 1001
--- Isi Tabel Appointment ---
ID: 1001, PasienID: 101, DokterID: 1, Status: Scheduled
[U] Mengupdate status appointment ID 1001 menjadi 'Completed'
--- Isi Tabel Appointment ---
ID: 1001, PasienID: 101, DokterID: 1, Status: Completed
===== DEMO ENTITAS: RELATIONSHIP & MULTIVALUED =====
--- Isi Tabel doctor_clinic (Relasi) ---
[Kosong]
--- Isi Tabel patient_phoneNumber ---
[Kosong]
[C] Menghubungkan dokter ID 1 ke klinik ID 1
[C] Menambahkan nomor telepon ke pasien ID 101
[C] Data relasi & multivalued berhasil ditambahkan.
--- Isi Tabel doctor_clinic (Relasi) ---
DokterID: 1, ClinicID: 1
--- Isi Tabel patient_phoneNumber ---
PasienID: 101, No.Telp: 08123456789
===== CLEANUP DATA DEMO =====
===== DEMO SELESAI =====