```
--  Hospital Management System – PostgreSQL Design,Database & Schema Setup

-- Create the database
CREATE DATABASE hospital_db;

-- departments Table
CREATE TABLE departments (
   dept_id SERIAL PRIMARY KEY,
   name VARCHAR(100) NOT NULL UNIQUE,
   head_doctor_id INT,
   floor INT NOT NULL,
   contact_phone VARCHAR(20),
   description TEXT
);

-- doctors Table
CREATE TABLE doctors (
   doctor_id SERIAL PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   specialization VARCHAR(100),
   dept_id INT REFERENCES departments(dept_id) ON DELETE SET NULL,
   phone VARCHAR(15),
   email VARCHAR(100) UNIQUE,
   experience_years INT CHECK (experience_years >= 0),
   shift_start TIME,
   shift_end TIME,
   is_active BOOLEAN DEFAULT TRUE
);
```

```
-- Updating records
Update departments.head_doctor_id to reference doctors.doctor_id.
ALTER TABLE departments
ADD CONSTRAINT fk_head_doctor
FOREIGN KEY (head_doctor_id) REFERENCES doctors(doctor_id) ON DELETE SET NULL;

-- patients Table
CREATE TABLE patients (
    patient_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    dob DATE NOT NULL,
    gender VARCHAR(10) CHECK (gender IN ('Male', 'Female', 'Other')),
    phone VARCHAR(15),
    email VARCHAR(100) UNIQUE,
    address TEXT,
    emergency_contact_name VARCHAR(100),
    emergency_contact_phone VARCHAR(15),
    created_at TIMESTAMP DEFAULT NOW()
);

-- staff Table (Nurses, Admins, etc.)
CREATE TABLE staff (
    staff_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    role VARCHAR(50) NOT NULL, -- Nurse, Admin, Technician
    dept_id INT REFERENCES departments(dept_id) ON DELETE SET NULL,
    phone VARCHAR(15),
    email VARCHAR(100) UNIQUE,
```

```
  shift VARCHAR(20), -- Day, Night, Rotational
  hire_date DATE DEFAULT CURRENT_DATE
);

-- rooms Table
CREATE TABLE rooms (
  room_id SERIAL PRIMARY KEY,
  room_number VARCHAR(10) NOT NULL,
  room_type VARCHAR(20) NOT NULL CHECK (room_type IN ('General', 'ICU', 'Operation', 'Maternity', 'Pediatric')),
  dept_id INT REFERENCES departments(dept_id) ON DELETE SET NULL,
  floor INT NOT NULL,
  status VARCHAR(20) DEFAULT 'Available' CHECK (status IN ('Available', 'Occupied', 'Maintenance')),
  UNIQUE(room_number, floor)
);

-- appointments Table
CREATE TABLE appointments (
  appointment_id SERIAL PRIMARY KEY,
  patient_id INT NOT NULL REFERENCES patients(patient_id) ON DELETE CASCADE,
  doctor_id INT NOT NULL REFERENCES doctors(doctor_id) ON DELETE CASCADE,
  dept_id INT REFERENCES departments(dept_id) ON DELETE SET NULL,
  appointment_date TIMESTAMP NOT NULL,
  status VARCHAR(20) DEFAULT 'Scheduled' CHECK (status IN ('Scheduled', 'Completed', 'Cancelled', 'No-show')),
  reason TEXT,
  notes TEXT,
  created_at TIMESTAMP DEFAULT NOW(),
```

```
    -- Prevent double booking (same doctor at same time)
    CONSTRAINT unique_doctor_time UNIQUE (doctor_id, appointment_date)
);
-- medical_records Table
CREATE TABLE medical_records (
    record_id SERIAL PRIMARY KEY,
    patient_id INT NOT NULL REFERENCES patients(patient_id) ON DELETE CASCADE,
    doctor_id INT NOT NULL REFERENCES doctors(doctor_id) ON DELETE RESTRICT,
    visit_date TIMESTAMP DEFAULT NOW(),
    diagnosis TEXT,
    treatment TEXT,
    prescriptions TEXT, -- Or use a separate table for multiple prescriptions
    lab_tests JSONB, -- Store as JSON: e.g., '[{"test":"CBC","result":"Normal"}]'
    next_followup DATE,
    created_at TIMESTAMP DEFAULT NOW()
);

-- bills Table
CREATE TABLE bills (
    bill_id SERIAL PRIMARY KEY,
    patient_id INT NOT NULL REFERENCES patients(patient_id) ON DELETE CASCADE,
    appointment_id INT REFERENCES appointments(appointment_id) ON DELETE SET NULL,
    total_amount DECIMAL(10,2) NOT NULL CHECK (total_amount >= 0),
    payment_status VARCHAR(20) DEFAULT 'Unpaid' CHECK (payment_status IN ('Paid', 'Unpaid',
'Partial')),
    payment_method VARCHAR(50) CHECK (payment_method IN ('Cash', 'Credit Card', 'Insurance',
'UPI', 'Bank Transfer')),
    issued_date TIMESTAMP DEFAULT NOW(),
    paid_date TIMESTAMP,
```

```
    notes TEXT
);

-- Insering Sample Data
-- Insert Department
INSERT INTO departments (name, floor, contact_phone, description)
VALUES ('Cardiology', 3, '+2334567890 ext. 101', 'Heart and cardiovascular care');

-- Insert Doctor
INSERT INTO doctors (name, specialization, dept_id, phone, email, experience_years, shift_start, shift_end)
VALUES ('Dr. Alice Smith', 'Cardiologist', 1, '+1987654321', 'alice.smith@hospital.com', 10, '09:00', '17:00');

-- Update department head
UPDATE departments SET head_doctor_id = 1 WHERE dept_id = 1;

-- Insert Patient
INSERT INTO patients (first_name, last_name, dob, gender, phone, email, address, emergency_contact_name, emergency_contact_phone)
VALUES ('John', 'Doe', '1985-04-15', 'Male', '+1234567890', 'john.doe@email.com', '123 Main St, New York, NY 10001', 'Jane Doe', '+1234567891');

-- Insert Appointment
INSERT INTO appointments (patient_id, doctor_id, dept_id, appointment_date, reason, status)
VALUES (1, 1, 1, '2025-04-05 10:00:00', 'Routine Checkup', 'Scheduled');

-- Insert Room
INSERT INTO rooms (room_number, room_type, dept_id, floor, status)
VALUES ('101', 'ICU', 1, 3, 'Available');
```

```
-- Insert Medical Record

INSERT INTO medical_records (patient_id, doctor_id, diagnosis, treatment, prescriptions,
lab_tests, next_followup)

VALUES (1, 1, 'Hypertension', 'Lifestyle changes and medication', 'Lisinopril 10mg daily',

    '[{"test": "Blood Pressure", "result": "150/95 mmHg"}, {"test": "Cholesterol", "result":
"High"}]'::JSONB,

    '2025-05-05');


-- Insert Bill

INSERT INTO bills (patient_id, appointment_id, total_amount, payment_status, payment_method,
paid_date)

VALUES (1, 1, 250.00, 'Paid', 'Credit Card', NOW());


--  Useful Queries

-- List all appointments with patient and doctor names

SELECT

  a.appointment_id,

  p.first_name || ' ' || p.last_name AS patient_name,

  d.name AS doctor_name,

  dept.name AS department,

  a.appointment_date,

  a.status

FROM appointments a

JOIN patients p ON a.patient_id = p.patient_id

JOIN doctors d ON a.doctor_id = d.doctor_id

JOIN departments dept ON a.dept_id = dept.dept_id

ORDER BY a.appointment_date;


-- Find unpaid bills with patient info
```

```
SELECT
  b.bill_id,
  p.first_name || ' ' || p.last_name AS patient_name,
  b.total_amount,
  b.issued_date
FROM bills b
JOIN patients p ON b.patient_id = p.patient_id
WHERE b.payment_status = 'Unpaid';

--  Count available rooms by type
SELECT room_type, COUNT(*) AS available_count
FROM rooms
WHERE status = 'Available'
GROUP BY room_type;

-- Get doctors by department

SELECT
  d.name,
  d.specialization,
  dept.name AS department
FROM doctors d
JOIN departments dept ON d.dept_id = dept.dept_id
ORDER BY dept.name, d.name;

-- Indexes for Performance
CREATE INDEX idx_patients_email ON patients(email);
CREATE INDEX idx_patients_phone ON patients(phone);
```

```
CREATE INDEX idx_doctors_specialization ON doctors(specialization);
CREATE INDEX idx_doctors_dept ON doctors(dept_id);

CREATE INDEX idx_appointments_date ON appointments(appointment_date);
CREATE INDEX idx_appointments_patient ON appointments(patient_id);
CREATE INDEX idx_appointments_doctor ON appointments(doctor_id);

CREATE INDEX idx_bills_status ON bills(payment_status);
CREATE INDEX idx_bills_patient ON bills(patient_id);

CREATE INDEX idx_medical_records_patient ON medical_records(patient_id);
CREATE INDEX idx_medical_records_date ON medical_records(visit_date);
```