Database name : matrix

Core Tables:

```
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role ENUM('patient', 'doctor') NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    last_login DATETIME,
    is_active BOOLEAN DEFAULT TRUE
);

CREATE TABLE PatientDetails (
    patient_id INT PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    date_of_birth DATE,
    gender ENUM('Male', 'Female', 'Other'),
    phone VARCHAR(20),
    address TEXT,
    FOREIGN KEY (patient_id) REFERENCES Users(user_id)
);

CREATE TABLE DoctorDetails (
```

```
    doctor_id INT PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    specialty_id INT NOT NULL,
    license_number VARCHAR(50) UNIQUE,
    years_experience INT,
    bio TEXT,
    consultation_fee DECIMAL(10,2),
    FOREIGN KEY (doctor_id) REFERENCES Users(user_id),
    FOREIGN KEY (specialty_id) REFERENCES Specialties(specialty_id)
);

CREATE TABLE Specialties (
    specialty_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
```

Appointment System:

```
CREATE TABLE Appointments (
    appointment_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_id INT NOT NULL,
    doctor_id INT NOT NULL,
    appointment_date DATETIME NOT NULL,
    status ENUM('Booked', 'Completed', 'Cancelled') DEFAULT 'Booked',
    notes TEXT,
```

```
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (patient_id) REFERENCES Users(user_id),
    FOREIGN KEY (doctor_id) REFERENCES Users(user_id)
);

CREATE TABLE DoctorSchedules (
    schedule_id INT PRIMARY KEY AUTO_INCREMENT,
    doctor_id INT NOT NULL,
    day_of_week ENUM('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'),
    start_time TIME,
    end_time TIME,
    is_available BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (doctor_id) REFERENCES Users(user_id)
);
```

Health Metrics & Calculators:

```
CREATE TABLE HealthMetrics (
    metric_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_id INT NOT NULL,
    weight DECIMAL(5,2),
    height DECIMAL(5,2),
    age INT,
blood_pressure VARCHAR(15),
heart_rate VARCHAR(15),
temperature VARCHAR(15),
```

```
 notes TEXT,

    recorded_date DATE DEFAULT CURRENT_DATE,
    FOREIGN KEY (patient_id) REFERENCES Users(user_id)
);
```

Payment & Prescriptions:

```
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    appointment_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    payment_method ENUM('Credit Card', 'Debit Card', 'UPI', 'PayPal'),
    transaction_id VARCHAR(255),
    status ENUM('Pending', 'Completed', 'Failed') DEFAULT 'Pending',
    payment_date DATETIME,
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id)
);

CREATE TABLE Prescriptions (
    prescription_id INT PRIMARY KEY AUTO_INCREMENT,
    appointment_id INT NOT NULL,
    diagnosis TEXT,
    prescription_date DATE DEFAULT CURRENT_DATE,
    instructions TEXT,
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id)
```

);

Medical Information:

```
CREATE TABLE Medicines (
    medicine_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    manufacturer VARCHAR(255),
    dosage_form VARCHAR(100),
    price DECIMAL(10,2),
    stock INT DEFAULT 0
);

CREATE TABLE Diseases (
    disease_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) UNIQUE NOT NULL,
    description TEXT,
    symptoms TEXT,
    causes TEXT,
    prevention TEXT
);
```

Key Relationships (ER Diagram)

- Users → Appointments (1:M both for patients and doctors)
- Appointments → Payments (1:1)
- Appointments → Prescriptions (1:1)
- DoctorDetails→ Specialties (M:1)

- Prescriptions → Medicines (M:M through junction table)

Indexing

```
CREATE INDEX idx_users_email ON Users(email);

CREATE INDEX idx_appointments_date ON Appointments(appointment_date);

CREATE INDEX idx_patient_metrics ON HealthMetrics(patient_id, recorded_date);

CREATE INDEX idx_doctor_specialty ON DoctorDetails(specialty_id);
```

Security Considerations

1. Always store passwords using bcrypt or Argon2 hashing
2. Use prepared statements for all database queries
3. Encrypt sensitive data like transaction IDs
4. Implement role-based access control
5. Regularly backup the database

Normalization

All tables are in 3NF with:

- No transitive dependencies
- All non-key attributes fully dependent on primary keys
- Separate junction tables for M:M relationships

Admin Table Schema

Copy

```
CREATE TABLE Admin (
    admin_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    user_id INT UNIQUE NOT NULL,
    full_name VARCHAR(255) NOT NULL,
    admin_role ENUM('super_admin', 'hospital_admin', 'records_admin', 'billing_admin') NOT NULL,
    department VARCHAR(100),
    permissions JSON,
    last_access DATETIME,
    is_active BOOLEAN DEFAULT TRUE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
```

Modifications to Existing Tables

```
ALTER TABLE Users
MODIFY COLUMN role ENUM('patient', 'doctor', 'admin') NOT NULL;
```

2. Create admin-specific relationships with other tables:

sql

Copy

```
-- Admin access logs
CREATE TABLE AdminAccessLogs (
    log_id INT PRIMARY KEY AUTO_INCREMENT,
    admin_id INT NOT NULL,
    action VARCHAR(255) NOT NULL,
```

```
    table_affected VARCHAR(100) NOT NULL,
    record_id INT,
    ip_address VARCHAR(45),
    user_agent TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (admin_id) REFERENCES Admin(admin_id)
);

-- Admin to Appointments (track who modifies appointments)
ALTER TABLE Appointments
ADD COLUMN last_modified_by INT,
ADD FOREIGN KEY (last_modified_by) REFERENCES Admin(admin_id);

-- Admin to Prescriptions (track who approves prescriptions)
ALTER TABLE Prescriptions
ADD COLUMN approved_by INT,
ADD FOREIGN KEY (approved_by) REFERENCES Admin(admin_id);

-- Admin to Payments (track who processes payments)
ALTER TABLE Payments
ADD COLUMN processed_by INT,
ADD FOREIGN KEY (processed_by) REFERENCES Admin(admin_id);
Indexes for Admin Tables
sql
Copy
CREATE INDEX idx_admin_user ON Admin(user_id);
```

```
CREATE INDEX idx_admin_role ON Admin(admin_role);

CREATE INDEX idx_admin_access ON AdminAccessLogs(admin_id, created_at);
```