**## MySQL Database Design**

Here are **6 primary tables** to cover essential features:

1. patients
2. doctors
3. appointments
4. admins
5. payments
6. doctor\_availability

We will also **design with foreign keys, constraints**, and leave business rules like email/phone format validation to the application layer.

**Relational Design Considerations**

* **Cascading deletes** should be used with care. For example, deleting a patient could **soft-delete** instead of removing records, preserving appointment and payment history.
* **Appointment overlap** for doctors must be prevented via logic or database constraints.
* Doctors should **predefine availability** so patients can book only valid time slots.
* **Prescriptions and notes** will be stored in **MongoDB**, since they are unstructured/flexible. We'll just link via appointment\_id.

**Table: patients**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| full\_name | VARCHAR(100) | NOT NULL |
| email | VARCHAR(150) | NOT NULL, UNIQUE |
| password\_hash | VARCHAR(255) | NOT NULL |
| phone\_number | VARCHAR(20) |  |
| date\_of\_birth | DATE |  |
| gender | ENUM('M','F','O') |  |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Table: doctors**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| full\_name | VARCHAR(100) | NOT NULL |
| email | VARCHAR(150) | NOT NULL, UNIQUE |
| password\_hash | VARCHAR(255) | NOT NULL |
| specialization | VARCHAR(100) | NOT NULL |
| phone\_number | VARCHAR(20) |  |
| bio | TEXT |  |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Table: appointments**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| patient\_id | INT | NOT NULL, FOREIGN KEY → patients(id) |
| doctor\_id | INT | NOT NULL, FOREIGN KEY → doctors(id) |
| appointment\_time | DATETIME | NOT NULL |
| duration\_minutes | INT | DEFAULT 60 |
| status | ENUM('BOOKED', 'CANCELLED', 'COMPLETED') | DEFAULT 'BOOKED' |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Constraints:**

* Add **composite unique constraint** on (doctor\_id, appointment\_time) to **prevent overlapping bookings**.
* Alternatively: store availability and enforce bookings only through that logic.

**Table: admins**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| full\_name | VARCHAR(100) | NOT NULL |
| email | VARCHAR(150) | NOT NULL, UNIQUE |
| password\_hash | VARCHAR(255) | NOT NULL |
| role | VARCHAR(50) | DEFAULT 'ADMIN' |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Table: payments**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| appointment\_id | INT | FOREIGN KEY → appointments(id) |
| amount | DECIMAL(10,2) | NOT NULL |
| payment\_method | VARCHAR(50) | NOT NULL |
| payment\_status | ENUM('PAID','PENDING','FAILED') | DEFAULT 'PENDING' |
| transaction\_id | VARCHAR(100) | UNIQUE |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Table: doctor\_availability**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| doctor\_id | INT | FOREIGN KEY → doctors(id) |
| day\_of\_week | ENUM('MON','TUE','WED','THU','FRI','SAT','SUN') | NOT NULL |
| start\_time | TIME | NOT NULL |
| end\_time | TIME | NOT NULL |

**Notes:**

* Populate weekly recurring availability for doctors.
* Bookings can only be scheduled within these time windows.

**Optional Table: clinic\_locations (if multiple branches exist)**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(100) | NOT NULL |
| address | TEXT | NOT NULL |
| phone\_number | VARCHAR(20) |  |
| operating\_hours | VARCHAR(100) | e.g. 'Mon-Fri 8am-6pm' |

**Data Retention Strategy**

* **Appointments:** Keep historical data forever for legal and medical purposes.
* **Patients:** Prefer *soft deletes* (e.g., is\_active flag) to preserve links with history.
* **Prescriptions, notes, uploads:** Stored in MongoDB, linked by appointment\_id.

**## MongoDB Collection Design**

Given the use cases, I’ll go with a **prescriptions** collection — an ideal fit for MongoDB because:

* Prescription structures vary between doctors/specialties
* Notes, dosage instructions, and metadata can be deeply nested
* Schema evolution is easy if we need to later add tags, attachments, etc.

**MongoDB Collection: prescriptions**

**Relation to MySQL:**

Each document in this collection links to a specific **appointment\_id**, which is stored in MySQL. This keeps relational data (appointments, patients, doctors) in MySQL, and flexible medical records here.

**Example Document (prescriptions)**

{

"\_id": "66572b3eaabd1f0024d2e0f3",

"appointment\_id": 1025,

"doctor\_id": 53,

"patient\_id": 78,

"date\_issued": "2025-06-01T09:45:00Z",

"medications": [

{

"name": "Amoxicillin",

"dosage": "500mg",

"frequency": "3 times a day",

"duration\_days": 7,

"notes": "Take after meals"

},

{

"name": "Ibuprofen",

"dosage": "200mg",

"frequency": "As needed",

"duration\_days": 5,

"notes": "Do not exceed 1200mg/day"

}

],

"notes": {

"chief\_complaint": "Persistent sore throat and mild fever.",

"diagnosis": "Suspected bacterial pharyngitis.",

"recommendations": "Increase fluid intake and rest. Monitor temperature daily."

},

"tags": ["infection", "antibiotics", "follow-up-needed"],

"attachments": [

{

"type": "pdf",

"file\_name": "throat\_scan\_results.pdf",

"url": "https://s3.smartclinic.ai/prescriptions/scan\_1025.pdf"

}

],

"created\_at": "2025-06-01T09:50:00Z",

"last\_updated": "2025-06-01T09:50:00Z",

"version": 1

}

**Why this design works well:**

* **Flexibility**: Doctors can add multiple medications with varied notes.
* **Scalability**: If the format evolves (e.g. adding lab\_results, teleconsult flag), it won’t break the rest of the schema.
* **Tagging**: Supports analytics later — e.g., most commonly used medications.
* **Attachments**: Can store links to lab results, scans, or other files.

**Design Considerations**

| **Question** | **Design Approach** |
| --- | --- |
| **Full patient object or just ID?** | Just store the patient\_id and doctor\_id. Pull full details from MySQL when needed. Keeps documents lean. |
| **Schema evolution?** | New fields can be added freely over time. For example, teleconsultation: true or prescribed\_by\_nurse: false. |
| **Nested arrays?** | Used for medications and attachments to reduce join-like overhead. |
| **Auditing versions?** | You can increment a version field and optionally store a history collection. |