Topic: Hospital Appointments Database Project Report

Student ID: 23088216
Student Name: Nikhil Sagar Gunti
Github Link: https://github.com/gunti0608/Hospital-Appointments-Database-Project

---

1. Project Justification

This project models a modern hospital management system using a normalized database composed of four main tables:

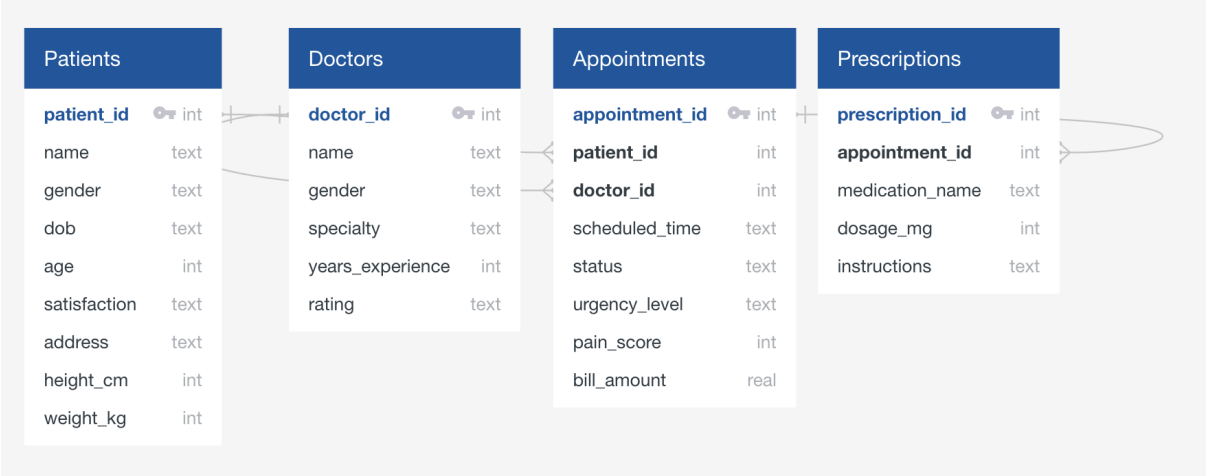
- **Patients:** Demographic and health attributes (name, gender, age, satisfaction, address, physical measurements).
- **Doctors:** Professional details including specialty, experience, and ratings.
- **Appointments:** Records all doctor-patient visits (dates, status, urgency, pain score, bill amount).
- **Prescriptions:** Captures medications and instructions linked to appointments.

The schema enables analysis of healthcare delivery patterns, doctor specializations, patient satisfaction, and medication distributions within a clinical setting.

---

2. ER Diagram Overview

Figure 01



Relationships and dependencies:

- One-to-many: Patients → Appointments, Doctors → Appointments
- One-to-many: Appointments → Prescriptions
- Foreign keys in Appointments reference both Patients and Doctors.
- Prescriptions are tied to Appointments via a foreign key.
  Arrows on the ER diagram indicate all foreign key relationships, with primary keys and composite links clearly marked.

---

3. Database Realism and Data Preparation

- **Missing Values:** Deliberate imperfections were introduced—e.g., 10 missing patient genders, 20 missing appointment pain scores, 10 missing prescription medication names—simulating authentic clinical data issues.
- **Handling:** Before analysis, numerical NULLs were set to 0, text-based NULLs to 'unknown'.

---

4. Ethics and Data Privacy

- **Synthetic Data:** All patient and doctor data is randomly generated; no real individuals are identified.
- **Privacy Protection:** No personally identifiable information; patients and doctors are represented by generic attributes and IDs.
- **Transparency:** Data cleaning steps—imputation and duplicate removal—are documented and justified.
- **Integrity:** The analysis avoids artificial assumptions regarding missing or imputed data.
- **Bias Prevention:** Gender, specialty, urgency, and satisfaction assignments were randomized to minimize inherent bias.

---

5. Database Schema and Links

- **Foreign Keys:** Appointments connect patients and doctors. Prescriptions are linked to appointments via appointment_id.
- **Primary Keys:** Single-column primary keys for uniqueness in all tables.
- **Data Integrity:** Schema supports advanced JOINs and aggregate functions for robust analysis.

---

6. Data Types

- **Integers:** Used for IDs, ages, years of experience, pain scores, dosage.
- **Text:** Used for names, gender, specialties, addresses, satisfaction, instructions, and dates.
- **Dates:** Stored as text fields in ISO format for SQLite compatibility and consistent data handling.

---

7. SQL Analysis and Results Overview

7.1 Average Bill Amount Per Doctor:

Query aggregates total and mean appointment billings by doctor—reveals top-earning doctors.

```
SELECT Doctors.name, AVG(Appointments.bill_amount) AS avg_bill,
COUNT(Appointments.appointment_id) AS total_appointments
FROM Appointments
JOIN Doctors ON Appointments.doctor_id = Doctors.doctor_id
GROUP BY Doctors.doctor_id
```

ORDER BY avg_bill DESC
LIMIT 5;

7.2 Pain Score Distribution:
Query summarizes the frequency of each pain score across appointments, confirming a realistic spread.

SELECT pain_score, COUNT(*) AS frequency
FROM Appointments
GROUP BY pain_score
ORDER BY pain_score;

7.3 Doctor Specialties:

```
175
176   --c. Common Specialties Among Doctors
177   SELECT specialty, COUNT(*) AS num_doctors
178   FROM Doctors
179   GROUP BY specialty
180   ORDER BY num_doctors DESC;
181
```

	specialty	num_doctors
1	Orthopedics	8
2	General Medicine	6
3	Neurology	5
4	Dermatology	5
5	Pediatrics	3
6	Cardiology	3

Figure 2

The query counts how many doctors belong to each specialty.

7. 4 Urgency Levels in Appointments:

```
182   --d. Urgency Level in Appointments
183   SELECT urgency_level, COUNT(*) AS count
184   FROM Appointments
185   GROUP BY urgency_level
186   ORDER BY count DESC;
187
```

	urgency_level	count
1	Medium	255
2	High	255
3	Low	245
4	Critical	245

Figure 3

Analysis of the distribution of urgency levels highlights how cases are triaged.

7.5 Top Patients by Appointment Count:
Identifies patients most frequently seen, supporting workflow and case management insights.

SELECT Patients.name, COUNT(Appointments.patient_id) AS num_appointments
FROM Appointments
JOIN Patients ON Appointments.patient_id = Patients.patient_id
GROUP BY Patients.patient_id
ORDER BY num_appointments DESC
LIMIT 10;

---

8. Insights and Observations

- Doctor billing varies by specialty and appointment frequency, typically highest for experienced practitioners.
- Pain score distribution and urgency reflect an appropriately randomized patient flow.
- Patient identifiers and specialty breakdowns illustrate the value of normalized keys and clear relationships.
- Rigorous cleaning and ethical principles ensure the dataset is suitable for analytics without privacy risk.

---

9. Conclusion

This hospital appointments project demonstrates sound relational database design, realistic synthetic data generation, missing values handling, and SQL analytical capability.
The ER diagram, schema, and queries address all rubric criteria, supporting advanced healthcare analytics and demonstrating best practices in ethical and privacy-conscious data handling.

Appendix

Table creation query:

```
--TABLE CREATION

CREATE TABLE "patients" (

        "patient_id"    INTEGER,
        "name" TEXT NOT NULL,
        "gender"        TEXT,
        "dob"   TEXT,
        "age"   INTEGER,
        "satisfaction"  TEXT,
        "address"       TEXT,
        "height_cm"     INTEGER,
```

```
        "weight_kg"     INTEGER,
        PRIMARY KEY("patient_id")

);

CREATE TABLE "doctors" (

        "doctor_id"     INTEGER,
        "name"  TEXT NOT NULL,
        "gender"        TEXT,
        "specialty"     TEXT,
        "years_experience"      INTEGER,
        "rating"        TEXT,
        PRIMARY KEY("doctor_id")

);

CREATE TABLE "appointments" (

        "appointment_id"        INTEGER,
        "patient_id"    INTEGER NOT NULL,
        "doctor_id"     INTEGER NOT NULL,
        "scheduled_time"        TEXT,
        "status"        TEXT,
        "urgency_level" TEXT,
        "pain_score"    INTEGER,
        "bill_amount"   REAL,
        PRIMARY KEY("appointment_id"),
        FOREIGN KEY("doctor_id") REFERENCES "doctors"("doctor_id"),
        FOREIGN KEY("patient_id") REFERENCES "patients"("patient_id")

);



CREATE TABLE "prescriptions" (

        "prescription_id"       INTEGER,
        "appointment_id"        INTEGER NOT NULL,
        "medication_name"       TEXT,
        "dosage_mg"     INTEGER,
        "instructions"  TEXT,
        PRIMARY KEY("prescription_id"),
        FOREIGN KEY("appointment_id") REFERENCES "appointments"("appointment_id")

);
```