

# SQL DATABASE For Amina\_MedicalCare Hospital



AMINA IBRAHIM

Author

# INTRODUCTION

The purpose of this report is to design and implement a fully normalized hospital database system using SQL. The database is structured to manage essential hospital operations such as patient management, doctor scheduling, medical records tracking, and appointment booking. Through proper normalization techniques, the database aims to ensure data integrity, avoid redundancy, and support efficient query execution. The report will document the design process, SQL statements for table creation and population, and queries to retrieve critical hospital information.

- **Objective:**

The goal is to design and implement a fully normalized hospital database in **SQL Server**. This database will manage patients, doctors, appointments, and medical records while ensuring data consistency, integrity, and efficient retrieval. This report documents the database schema, queries, and justifications for design decisions.

- **Overview:**

The database covers the following functionalities:

- Managing patient records
- Scheduling and tracking appointments
- Storing medical records
- Department management

## Database Design and Normalization

- **Entity Identification:**

The main entities identified for the hospital database are: **(T-SQL)**

```
CREATE DATABASE Amina_MedicalCare
```

1. **PATIENTS:** Stores patient personal information and contact details.

**PatientID:** This column serves as the unique identifier for each patient.

**Data Type:** INT is chosen as it's an efficient and scalable way to store unique numbers.

**Constraint:** PRIMARY KEY ensures uniqueness, while NOT NULL guarantees that every patient must have an ID.

**FullName:** Stores the full name of the patient.

**Data Type:** VARCHAR (100) is used because names vary in length but typically don't exceed 100 characters.

**Address:** Holds the address of the patient.

**Data Type:** VARCHAR (100) is appropriate for most residential or postal addresses.

**DateOfBirth** (Captures the patient's date of birth.

**Data Type:** DATE is ideal for storing dates efficiently. (yyyy-mm-dd)

**Insurance** Stores the insurance provider information.

**Data Type:** VARCHAR (50) is used since insurance names or policies typically fit within this length.

**Username:** Holds the username for patient login or identification.

**Data Type:** VARCHAR (50) ensures flexibility for username lengths.

**Password:** Stores the patient's password for access to their records or hospital account.

**Data Type:** VARCHAR (50) is sufficient to handle encrypted or hashed password storage.

**Email:** Used to store the patient's email address.

**Data Type:** VARCHAR (100) is appropriate as email addresses generally don't exceed this length.

**PhoneNumber:** Captures the patient's contact number.

**Data Type:** VARCHAR (20) is chosen as phone numbers, including country codes, can be lengthy.

**DateLeft:** Stores the date the patient was discharged or left the hospital.

**Data Type:** DATE is appropriate for capturing this specific date information.

```
CREATE TABLE Patients (
    PatientID INT PRIMARY KEY IDENTITY (1,1) NOT NULL,
    FullName VARCHAR (100) NOT NULL,
    Address VARCHAR (100),
    DateOfBirth DATE NOT NULL,
    Insurance VARCHAR (50),
    Username VARCHAR (50) UNIQUE,
    Password VARCHAR (50) NOT NULL,
    Email VARCHAR (100) NOT NULL,
    PhoneNumber VARCHAR (20) NOT NULL,
    DateLeft DATE
);
```

```

CREATE DATABASE Amina_MedicalCare
GO
CREATE TABLE Patients (
    PatientID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    FullName VARCHAR(100) NOT NULL,
    Address VARCHAR(100),
    DateOfBirth DATE NOT NULL,
    Insurance VARCHAR(50),
    Username VARCHAR(50) UNIQUE,
    Password VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    PhoneNumber VARCHAR(20) NOT NULL,
    DateLeft DATE
);

```

PatientID	FullName	Address	DateOfBirth	Insurance	Username	Password	Email	PhoneNumber	DateLeft
1	John Doe	123 Main St	1980-06-01	Aetna	jdoe	password123	john.doe@example.com	+11234567890	NULL
2	Jane Smith	456 Oak St	1975-08-12	BlueCross	jsmith	securePass1	jane.smith@example.com	+19876543210	NULL
3	Emily Clark	789 Pine St	1990-02-22	Cigna	eclark	emily1234	emily.clark@example.com	+14045551234	NULL
4	Michael Brown	321 Cedar St	1985-05-17	Medicare	mbrown	michaelpass	michael.brown@example.com	+14445567890	NULL
5	Emma Johnson	654 Elm St	1992-09-30	Aetna	ejohnson	emmajohnson	emma.johnson@example.com	+13025551234	NULL
6	James Davis	888 Maple St	1970-03-12	UnitedHealth	jdavis	passjames	james.davis@example.com	+12015551234	NULL
7	Olivia Garcia	222 Birch St	1983-07-23	Cigna	ogarcia	olivepass	olivia.garcia@example.com	+19875551234	NULL
8	Lucas Wilson	111 Spruce St	1995-12-18	Aetna	lwilson	lucas1234	lucas.wilson@example.com	+15045551234	NULL
9	Sophia Martinez	333 Chestnut St	1987-08-25	BlueCross	smartinez	sophiapass	sophia.martinez@example.com	+14125551234	NULL
10	Mason Anderson	777 Aspen St	1973-11-04	UnitedHealth	manderson	masonpass	mason.anderson@example.com	+13015551234	NULL
11	Ava Thompson	999 Cypress St	1989-06-30	Cigna	athompson	avapass	ava.thompson@example.com	+16025551234	NULL
12	Ethan Lee	444 Redwood St	1982-05-15	BlueCross	elee	ethanlee	ethan.lee@example.com	+17025551234	NULL
13	Mia Harris	666 Oakwood St	1991-10-10	Aetna	mharris	miapass	mia.harris@example.com	+15015551234	NULL
14	Liam Clark	123 Sycamore St	1980-04-28	Medicare	lclark	liam123	liam.clark@example.com	+18015551234	NULL
15	Amelia Walker	345 Pinewood St	1993-11-11	UnitedHealth	awalker	amelia123	amelia.walker@example.com	+19025551234	NULL
16	Noah Perez	678 Oakwood St	1984-07-14	BlueCross	nperez	noah1234	noah.perez@example.com	+18125551234	NULL
17	Isabella Hall	890 Aspen St	1986-01-22	Cigna	ihall	isapass	isabella.hall@example.com	+15015551234	NULL
18	William Young	432 Cedarwood St	1978-03-20	Medicare	wyoung	willpass	william.young@example.com	+17015551234	NULL
19	Evelyn King	101 Redwood St	1994-02-02	UnitedHealth	eking	evelynking	evelyn.king@example.com	+14115551234	NULL
20	Benjamin Scott	654 Chestnut St	1977-12-05	Aetna	bscott	ben123	benjamin.scott@example.com	+19045551234	NULL

Figure 1: Patient Records Showing Patient Table

## 2. DOCTORS: Stores information on doctors and their specialties.

**DoctorID: (INT, PRIMARY KEY, NOT NULL)**: This column uniquely identifies each doctor.

**Data Type**: INT allows for an efficient primary key.

**Constraint**: PRIMARY KEY ensures each doctor has a unique identifier.

**FullName**: Stores the full name of the doctor.

**Data Type**: VARCHAR (100) allows for a wide range of name lengths.

**Specialty**: Stores the medical specialty of the doctor, such as "Cardiology" or "Pediatrics."

**Data Type**: VARCHAR (50) is sufficient for most medical specialty names.

CREATE TABLE Doctors (

DoctorID INT PRIMARY KEY IDENTITY (1,1) NOT NULL,

FullName VARCHAR (100) NOT NULL,

Specialty VARCHAR (50) NOT NULL

);

Figure 2: Doctors Records Showing Doctors Table

The screenshot shows the SSMS interface with the following details:

- Connections:** Welcome, localhost,1433, SQLQuery\_2 - (62) I...re (SA) 6.
- Servers:** localhost,1433, <default> (SA)
- Databases:** Amina\_MedicalCare
- SQL Query:**

```

85      INSERT INTO Doctors (FullName, Specialty)
86          VALUES
87      ('Dr. Gregory House', 'Gastroenterologists'),
88      ('Dr. Meredith Grey', 'Cardiologists'),
89      ('Dr. John Carter', 'Oncologists'),
90      ('Dr. Derek Shepherd', 'Neurosurgeons'),
91      ('Dr. Miranda Bailey', 'Pediatrics'),
92      ('Dr. Mark Sloan', 'Plastic Surgeons'),
93      ('Dr. Arizona Robbins', 'Orthopedic Surgeons'),
94      ('Dr. Alex Karev', 'Pediatric Surgeons'),
95      ('Dr. Owen Hunt', 'Trauma Surgeons'),
96      ('Dr. Richard Webber', 'Internal Medicine'),
97      ('Dr. April Kepner', 'Trauma Surgeons'),
98      ('Dr. Jackson Avery', 'Plastic Surgeons'),
99      ('Dr. Amelia Shepherd', 'Neurosurgeons'),
100     ('Dr. Teddy Altman', 'Cardiothoracic Surgeons'),
101     ('Dr. Callie Torres', 'Orthopedic Surgeons'),
102     ('Dr. Jo Wilson', 'General Surgeons'),
103     ('Dr. Maggie Pierce', 'Cardiologists'),
104     ('Dr. Andrew DeLuca', 'General Surgeons'),
105     ('Dr. Tom Koracick', 'Neurosurgeons'),
106     ('Dr. Cormac Hayes', 'Pediatrics');
    
```
- Results:** Shows the data from the 'Doctors' table in a grid format.
- Messages:** No messages displayed.
- Bottom Status Bar:** Ln 185, Col 1 (21 selected), Spaces: 4, UTF-8, LF, 20 rows, MSSQL, 00:00:00, localhost,1433 : Amina\_MedicalCare (62).

3. **APPOINTMENTS:** Tracks appointments between patients and doctors, including feedback.

**Appointment:** Unique identifier for each appointment.

**Data Type:** INT is an efficient choice for primary keys.

**Constraint:** PRIMARY KEY ensures uniqueness.

**PatientID:** Links the appointment to the patient.

**Data Type:** INT refers to the PatientID from the Patients table.

**Constraint:** NOT NULL and a FOREIGN KEY ensure it corresponds to an existing patient.

**DoctorID:** Links the appointment to the doctor.

**Data Type:** INT refers to the DoctorID from the Doctors table.

**Constraint:** NOT NULL and FOREIGN KEY ensure it references a valid doctor.

**Appointment Date:** Stores the date of the appointment.

**Data Type:** DATE stores this information efficiently.

**Constraint:** The CHECK constraint ensures that appointments cannot be scheduled in the past.

**Appointment Time:** Captures the time of the appointment.

**Data Type:** TIME is used to store the specific time of day.

**Department:** Indicates which department the appointment is for (e.g., Cardiology):

**Data Type:** is suitable for department names.

**Status:** Stores the status of the appointment.

**Data Type:** VARCHAR (20) to store the current appointment state.

**Constraint:** The CHECK constraint ensures only valid statuses are entered.

**Feedback:** Allows the patient or doctor to leave feedback on the appointment.

**Data Type:** VARCHAR (100) provides enough space for detailed comments.

```
CREATE TABLE Appointments (
```

```
    AppointmentID INT PRIMARY KEY IDENTITY (1,1) NOT NULL,
    PatientID INT NOT NULL,
    DoctorID INT NOT NULL,
    AppointmentTime TIME,
    AppointmentDate DATE,
    Department VARCHAR (50),
    Status VARCHAR (20) CHECK (Status IN ('Pending', 'Cancelled', 'Completed')),
    Feedback VARCHAR (100),
    FOREIGN KEY (PatientID) REFERENCES Patients (PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctors (DoctorID)
```

```
);
```

Figure 3: Appointment Records showing Appointment Table

The screenshot shows the SSMS interface with the 'Appointments' table data. The table has columns: AppointmentID, PatientID, DoctorID, AppointmentTime, AppointmentDate, Department, Status, and Feedback. The data consists of 133 rows, each representing an appointment record with details like patient ID, doctor ID, appointment time, date, department, status, and feedback.

	AppointmentID	PatientID	DoctorID	AppointmentTime	AppointmentDate	Department	Status	Feedback
1	1	1	1	09:00:00	2024-10-01	Gastroenterology	Completed	Excellent service
2	2	2	2	11:00:00	2024-10-02	Cardiology	Completed	Satisfied
3	3	3	3	14:00:00	2024-10-03	Oncology	Pending	NULL
4	4	4	4	13:30:00	2024-10-04	Neurosurgery	Completed	Well organized
5	5	5	5	10:00:00	2024-10-05	Pediatrics	Pending	NULL
6	6	6	6	16:00:00	2024-10-06	Plastic Surgery	Cancelled	Doctor not available
7	7	7	7	15:30:00	2024-10-07	Orthopedics	Completed	Smooth process
8	8	8	8	08:45:00	2024-10-08	Pediatric Surgery	Completed	Well done
9	9	9	9	09:15:00	2024-10-09	Trauma Surgery	Completed	Great
10	10	10	10	11:00:00	2024-10-10	Internal Medicine	Completed	Very good
11	11	11	11	14:30:00	2024-10-11	Trauma Surgery	Pending	NULL
12	12	12	12	12:00:00	2024-10-12	Plastic Surgery	Completed	Painless experience
13	13	13	13	13:15:00	2024-10-13	Neurosurgery	Pending	NULL
14	14	14	14	16:30:00	2024-10-14	Cardiothoracic Surgery	Completed	Excellent care
15	15	15	15	11:45:00	2024-10-15	Orthopedics	Completed	Efficient service
16	16	16	16	15:00:00	2024-10-16	General Surgery	Cancelled	Family emergency
17	17	17	17	10:30:00	2024-10-17	Cardiology	Completed	Well done
18	18	18	18	09:00:00	2024-10-18	General Surgery	Pending	NULL
19	19	19	19	14:00:00	2024-10-19	Neurosurgery	Completed	Good experience
20	20	20	20	12:30:00	2024-10-20	Pediatrics	Pending	NULL

4. **MEDICALRECORDS:** Stores a patient's medical history, diagnosis, and prescribed medicines.

**MedicalRecordID (INT, PRIMARY KEY, NOT NULL):** Unique identifier for each medical record.

**Data Type:** INT is used as the primary key for simplicity and efficiency.

**PatientID (INT, NOT NULL):** Links the medical record to a specific patient.

**Data Type:** INT refers to the PatientID in the Patients table.

**Constraint:** FOREIGN KEY ensures it corresponds to an existing patient.

**Past Appointments:** Lists past appointments for the patient.

**Data Type:** VARCHAR (100) provides enough space to store appointment summaries.

**Diagnoses** Stores diagnoses the patient has received.

**Data Type:** VARCHAR (100) allows for a detailed medical history.

**Medicines:** Lists medicines the patient has been prescribed.

**Data Type:** VARCHAR (100) provides sufficient space for multiple medications.

**MedicinePrescribedDate:** Captures the date when the medicine was prescribed.

**Data Type:** DATE is appropriate for storing this information.

**Allergies:** Stores known allergies for the patient.

**Data Type:** VARCHAR (100) provides enough space for detailed allergy descriptions.

```
CREATE TABLE Medical Records (
    MedicalRecordID INT PRIMARY KEY IDENTITY (1,1) NOT NULL,
    PatientID INT NOT NULL,
    PastAppointments VARCHAR (100),
    Diagnoses VARCHAR (100),
    Medicines VARCHAR (100),
    MedicinePrescribedDate DATE,
    Allergies VARCHAR (100),
    FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)
);
```

The screenshot shows the SSMS interface with the following details:

- Top Bar:** Welcome, localhost,1433, SQLQuery\_2 - (62) line (SA) 6, Search.
- Servers Tree:** SERVERS > localhost,1433, <default> (SA) > Databases > Amina\_MedicalCare.
- Query Editor:**
  - Text: An `INSERT INTO` statement for the `MedicalRecords` table, inserting 156 rows of medical records.
  - Results Grid:
 

	MedicalRecordID	PatientID	PastAppointments	Diagnoses	Medicines	MedicinePrescribedDate	Allergies
1	1	1	Gastroenterologist visit on 2023-10-15	Gastroenteritis	Omeprazole	2023-10-15	None
2	2	2	Cardiologist visit on 2024-01-12	Hypertension	Lisinopril	2024-01-12	Penicillin
3	3	3	Oncologist visit on 2024-03-05	Breast Cancer	Tamoxifen	2024-03-05	None
4	4	4	Neurosurgeon visit on 2024-04-22	Migraine	Sumatriptan	2024-04-22	None
5	5	5	Pediatrician visit on 2024-05-10	Asthma	Albuterol	2024-05-10	Peanuts
6	6	6	Plastic Surgeon visit on 2024-06-14	Skin Graft	None	2024-06-14	None
7	7	7	Orthopedic visit on 2024-07-19	Fracture	Ibuprofen	2024-07-19	None
8	8	8	Pediatric Surgery on 2024-08-01	Appendicitis	None	2024-08-01	None
9	9	9	Trauma Surgery on 2024-09-11	Concussion	Acetaminophen	2024-09-11	None
10	10	10	Internal Medicine visit on 2024-10-01	Diabetes	Metformin	2024-10-01	None
11	11	11	Trauma Surgery on 2024-11-12	Broken Rib	Oxycodone	2024-11-12	Shellfish
12	12	12	Plastic Surgery on 2024-12-22	Rhinoplasty	None	2024-12-22	None
13	13	13	Neurosurgery on 2025-01-15	Epilepsy	Levetiracetam	2025-01-15	None
14	14	14	Cardiothoracic Surgery on 2025-02-17	Aortic Aneurysm	Warfarin	2025-02-17	None
15	15	15	Orthopedics visit on 2025-03-22	ACL Tear	None	2025-03-22	None
16	16	16	General Surgery on 2025-04-08	Hernia Repair	None	2025-04-08	None
17	17	17	Cardiologist visit on 2025-05-14	Arrhythmia	Amiodarone	2025-05-14	None
18	18	18	General Surgery on 2025-06-25	Gallstones	None	2025-06-25	None
19	19	19	Neurosurgery on 2025-07-30	Brain Tumor	Temozolomide	2025-07-30	None
20	20	20	Pediatrics visit on 2025-08-12	Chickenpox	None	2025-08-12	None

Figure 4: MedicalRecords showing MedicalRecords Table

## 5. **DEPARTMENTS:** Lists the departments in the hospital.

**DepartmentID:** Unique identifier for each hospital department

**Data Type:** INT is efficient for a small, manageable list of departments.

**DepartmentName:** Stores the name of the department

**Data Type:** VARCHAR (50) is sufficient for typical department names.

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY IDENTITY (1,1) NOT NULL,

DepartmentName VARCHAR (50)

);

```

159    INSERT INTO Departments (DepartmentID, DepartmentName)
160        VALUES
161            (1, 'Cardiology'),
162            (2, 'Gastroenterology'),
163            (3, 'Oncology'),
164            (4, 'Pediatrics'),
165            (5, 'Neurology'),
166            (6, 'Orthopedics'),
167            (7, 'General Surgery'),
168            (8, 'Plastic Surgery'),
169            (9, 'Trauma Surgery'),
170            (10, 'Internal Medicine'),
171            (11, 'Endocrinology'),
172            (12, 'Dermatology'),
173            (13, 'Nephrology'),
174            (14, 'Pulmonology'),
175            (15, 'Psychiatry'),
176            (16, 'Urology'),
177            (17, 'Ophthalmology'),
178            (18, 'Rheumatology'),
179            (19, 'Obstetrics and Gynecology'),
180            (20, 'Otolaryngology');
181
182

```

	DepartmentID	DepartmentName
1	1	Cardiology
2	2	Gastroenterology
3	3	Oncology
4	4	Pediatrics
5	5	Neurology
6	6	Orthopedics
7	7	General Surgery
8	8	Plastic Surgery
9	9	Trauma Surgery
10	10	Internal Medicine
11	11	Endocrinology
12	12	Dermatology
13	13	Nephrology
14	14	Pulmonology
15	15	Psychiatry
16	16	Urology
17	17	Ophthalmology
18	18	Rheumatology
19	19	Obstetrics and Gynecology
20	20	Otolaryngology

Figure 5: Department Records showing Department Table

#### -- ADDING CONSTRAINT TO CHECK APPOINTMENT DATE NOT IN THE PAST

-- This constraint ensures appointments are only scheduled for the current or future dates.

#### ALTER TABLE Appointments

```
ADD CONSTRAINT chk_AppointmentDate CHECK (AppointmentDate >= CAST (GETDATE () AS DATE));
```

#### -- QUERY: PATIENTS OLDER THAN 40 WITH CANCER DIAGNOSIS

```
SELECT FullName, DateOfBirth, Diagnoses
```

```
FROM Patients P
```

```
JOIN MedicalRecords MR ON P.PatientID = MR.PatientID
```

```
WHERE Diagnoses LIKE '%Cancer%' AND DATEDIFF(YEAR, P.DateOfBirth, GETDATE()) > 40;
```

The screenshot shows the SSMS interface with a query window open. The query is:

```

193 -- Adding Constraint to Check Appointment Date Not in the Past
194 -- This constraint ensures appointments are only scheduled for the current or future dates.
195
196 ALTER TABLE Appointments
197 ADD CONSTRAINT chk_AppointmentDate CHECK (AppointmentDate >= CAST(GETDATE() AS DATE));
198
199 -- Query: Patients Older than 40 with Cancer Diagnosis
200
201 SELECT FullName, DateOfBirth, Diagnoses
202 FROM Patients P
203 JOIN MedicalRecords MR ON P.PatientID = MR.PatientID
204 WHERE Diagnoses LIKE '%Cancer%' AND DATEDIFF(YEAR, P.DateOfBirth, GETDATE()) > 40;
205
206 -- Query: Number of Completed Appointments by Gastroenterologists
207
208 SELECT COUNT(*) AS CompletedAppointments
209 FROM Appointments A
210 JOIN Doctors D ON A.DoctorID = D.DoctorID
211 WHERE D.Specialty = 'Gastroenterologists' AND A.Status = 'Completed';
212

```

The results pane is empty.

Figure 6: Patients Older than 40 with Cancer = o

#### -- QUERY: NUMBER OF COMPLETED APPOINTMENTS BY GASTROENTEROLOGISTS

**SELECT COUNT(\*) AS CompletedAppointments**

**FROM Appointments A**

**JOIN Doctors D ON A.DoctorID = D.DoctorID**

**WHERE D.Specialty = 'Gastroenterologists' AND A.Status = 'Completed';**

The screenshot shows the SSMS interface with the following details:

- Left Sidebar:** Shows connections, servers (localhost,1433), databases (Amina\_MedicalCare, animal\_care, SA\_Production, University\_New\_Student\_Rec..., Security, Server Objects), and system databases.
- Top Bar:** Welcome, localhost,1433, SQLQuery\_2 - (62) I...re (SA) 5, Run, Disconnect, Change, Database: Amina\_MedicalCare, Estimated Plan, Enable Actual Plan, Parse, Enable SQLCMD, To Notebook.
- Code Window:** Displays T-SQL code for creating a constraint and performing two queries. The first query adds a constraint to ensure appointment dates are not in the past. The second query selects patient information and diagnoses where patients are older than 40 and have cancer. The third query counts completed appointments by gastroenterologists.

```

193 -- Adding Constraint to Check Appointment Date Not in the Past
194 -- This constraint ensures appointments are only scheduled for the current or future dates.
195
196 ALTER TABLE Appointments
197 ADD CONSTRAINT chk_AppointmentDate CHECK (AppointmentDate >= CAST(GETDATE() AS DATE));
198
199 -- Query: Patients Older than 40 with Cancer Diagnosis
200
201 SELECT FullName, DateOfBirth, Diagnoses
202 FROM Patients P
203 JOIN MedicalRecords MR ON P.PatientID = MR.PatientID
204 WHERE Diagnoses LIKE '%Cancer%' AND DATEDIFF(YEAR, P.DateOfBirth, GETDATE()) > 40;
205
206 -- Query: Number of Completed Appointments by Gastroenterologists
207
208 SELECT COUNT(*) AS CompletedAppointments
209 FROM Appointments A
210 JOIN Doctors D ON A.DoctorID = D.DoctorID
211 WHERE D.Specialty = 'Gastroenterologists' AND A.Status = 'Completed';
212

```

- Results Grid:** Shows the result of the third query: "CompletedAppointments" with value "1".
- Bottom Status Bar:** Ln 208, Col 1 (172 selected), Spaces: 4, UTF-8, LF, 1 rows, MSSQL, 00:00:00, localhost,1433 : Amina\_MedicalCare (62).

Figure 7: No. of completed Appointments with Gastroenterologists

## NORMALIZATION

The database is designed in **Third Normal Form (3NF)** to eliminate redundancy, improve data integrity, and optimize queries. This involves:

- Removing repeating groups and ensuring atomicity in attributes (1NF).
- Eliminating partial dependencies by ensuring non-key attributes depend on the primary key (2NF).
- Removing transitive dependencies to ensure that all attributes are dependent solely on the primary key (3NF).

## ERD (ENTITY-RELATIONSHIP DIAGRAM)

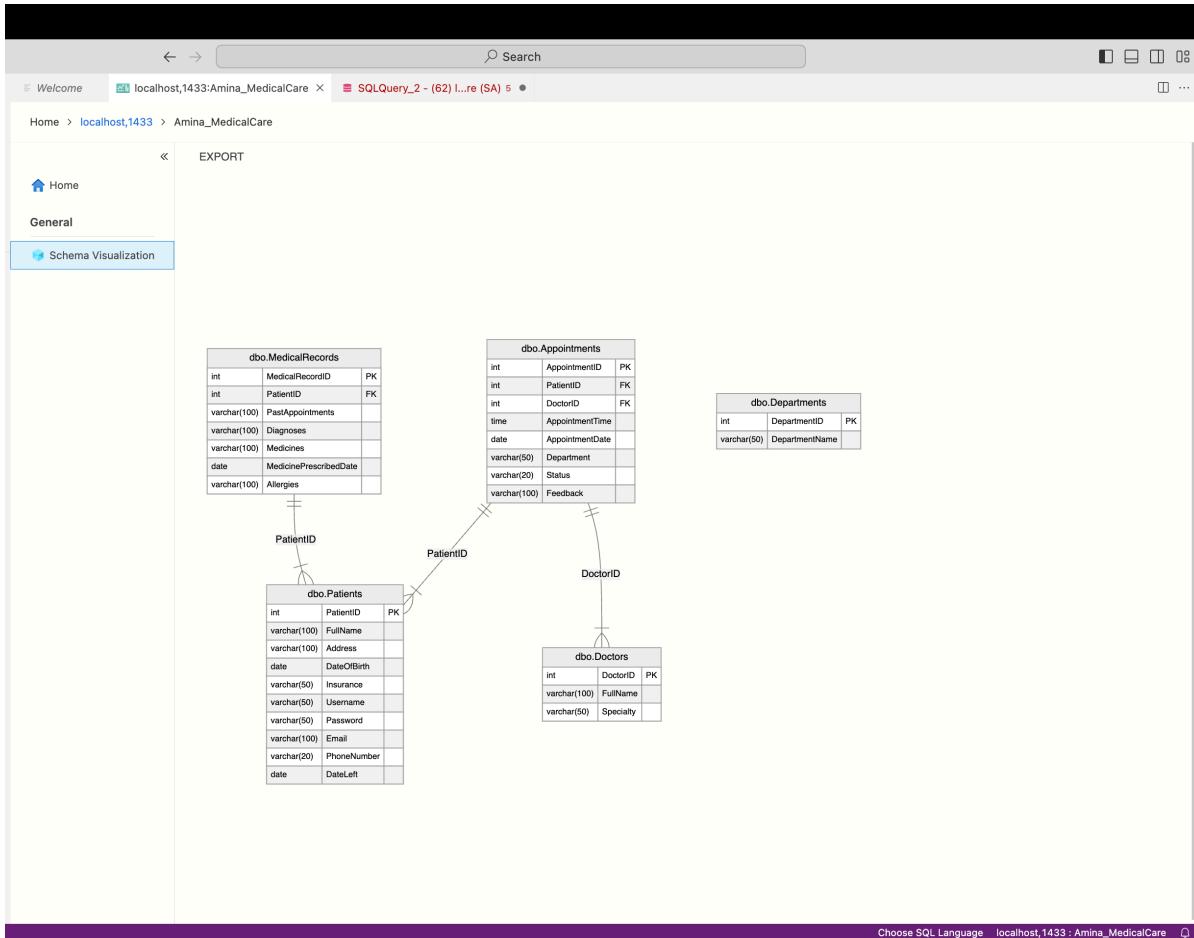


Figure 8: EDR visualization of all tables

## CONCLUSION

In summary, the process of designing, creating, and populating the hospital database involved key steps in establishing a structured, normalized schema to ensure data integrity and optimize query performance. By creating tables for patients, doctors, appointments, departments, and medical records, we were able to efficiently manage essential data for a hospital setting. The constraints applied, such as foreign keys and check constraints, ensure the database maintains valid relationships and restricts invalid data entries.

The queries developed allow for practical use of the database, such as ensuring no appointment is scheduled in the past, identifying patients with specific conditions, and tracking completed appointments by specific doctor specialties. Through this implementation, the database is set up to support accurate reporting and data analysis, contributing to improved hospital operations and patient care.

The design and execution of this database also emphasize the importance of proper data types and constraints in ensuring efficiency, accuracy, and scalability.