SQL Database Project: Hospital Management System

🎯 Project Objective

Design and implement a SQL database for a hospital system that supports managing patients, doctors, appointments, departments, admissions, billing, and staff, to apply **all SQL categories** (DDL, DML, DQL, DCL, TCL), **normalization**, and **advanced features** like **views**, **functions**, **stored procedures**, and **triggers**.

📂 Required Database Objects

1. Tables (DDL)

Each table must have appropriate **constraints**:

- PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK, DEFAULT

Table	Description
Patients	Patient details: name, DOB, gender, contact info
Doctors	Doctor details: specialization, contact info
Departments	Departments: cardiology, dermatology, etc.
Appointments	Links patients with doctors and a time
Admissions	For in-patient stays: room number, date in/out
Rooms	Room number, type (ICU, general), availability
MedicalRecords	Diagnosis, treatment plans, date, notes
Billing	Total cost, patient ID, services, date
Staff	Nurses, admin: role, shift, assigned dept
Users	Login credentials: username, password, role

📦 Database Design Requirements

- Use **varied data types**: INT, VARCHAR, DATE, DECIMAL, BIT, etc.
- **Normalize** to **3NF**. Submit **ERD + relational schema**.
- Use **naming conventions** and **documentation (comments)**.

🔍 Queries to Test (DQL)

Require students to write SQL queries for:

- List all patients who visited a certain doctor.
- Count of appointments per department.
- Retrieve doctors who have more than 5 appointments in a month.
- Use JOINs across 3–4 tables.
- Use GROUP BY, HAVING, and aggregate functions.
- Use SUBQUERIES and EXISTS.

🧮 Functions & Stored Procedures

Require:

- Scalar function to calculate patient age from DOB.
- Stored procedure to admit a patient (insert to Admissions, update Room availability).
- Procedure to generate invoice (insert into Billing based on treatments).
- Procedure to assign doctor to department and shift.

🔁 Triggers

Implement:

- After insert on Appointments → auto log in MedicalRecords.
- Before delete on Patients → prevent deletion if pending bills exist.
- After update on Rooms → ensure no two patients occupy same room.

🔒 Security (DCL)

- Create at least **two user roles**: DoctorUser, AdminUser.
- GRANT SELECT for DoctorUser on Patients and Appointments only.
- GRANT INSERT, UPDATE for AdminUser on all tables.
- REVOKE DELETE for Doctors.

🔄 Transactions (TCL)

- Simulate a transaction: admit a patient → insert record, update room, create billing → commit.
- Add rollback logic in case of failure.

🔍 Views

- vw_DoctorSchedule: Upcoming appointments per doctor.
- vw_PatientSummary: Patient info with their latest visit.
- vw_DepartmentStats: Number of doctors and patients per department.

📄 Deliverables

1. **Database script** (.sql) containing:
    - Table creation
    - Sample data insertion (20+ records per table)
    - Views, triggers, procedures, functions
2. **ERD + Normalization Explanation** (PDF or image)
3. **List of SQL Queries** (DQL, DML, TCL)
4. **Security script** for user roles and permissions
5. **Documentation** (README explaining all steps and logic)

Additional Requirement: SQL Server Job (SQL Agent Job)

🎯 Objective

Create and schedule an **SQL Job** using **SQL Server Agent** that performs automatic, recurring tasks related to the Hospital System database.

✅ SQL Job Requirements

Students must create **at least one SQL Job** that does the following:

📌 Option 1: Daily Backup Job

- Job Name: Daily_HospitalDB_Backup
- Schedule: Every day at 2:00 AM
- Action: Database backup

📌 Option 2: Doctor Schedule Report

- Job Name: Doctor_Daily_Schedule_Report
- Schedule: Every morning at 7:00 AM
- Action:
    - A stored procedure that extracts the daily doctor schedule from Appointments and inserts it into a report table DoctorDailyScheduleLog.

📄 Deliverables for SQL Job Task

1. **Script to Create SQL Job** using sp_add_job, sp_add_jobstep, and sp_add_schedule
   (or alternatively, step-by-step GUI screenshots if done via SSMS)
2. **Description of the job's purpose** in the documentation (README)
3. **Execution Log** (manually run and capture output/log results, or explain how to verify it works)

📚 Bonus Challenge (Optional)

Set up a SQL job that:

- Sends an **email alert** if any doctor has more than 10 appointments per day.
- **Exports** the billing summary into a .CSV file weekly using BCP or PowerShell.

✅ Evaluation Criteria

Criteria	Points
Schema Design (3NF)	15
Proper Use of Constraints	10
DDL, DML, DQL Queries	20
Joins and Aggregate Queries	10
Views and Subqueries	10
Functions & Stored Procedures	10
Sql Jobs	5
Triggers	5
Transactions & TCL	5
Security (GRANT/REVOKE)	5
Documentation & Submission	5
Total	**100**

Criteria	Points