Sitecore + .NET + SQL Developer Assignment

# Objective

Build a mini “Patient Management System” using **Sitecore,** **.NET 6/8 Web API** and **SQL Server** that supports CRUD operations, reporting, and optimized querying.

# .NET (Core API Development)

1. **API Design**

* Create RESTful APIs to:
  + Add a new patient
  + Update patient details
  + Delete patient
  + Fetch patient by ID
  + Search patients by filters (age range, gender, city, condition)

2. **Business Logic**

* Ensure **validation** (DOB can’t be in future, unique email/phone).
* Implement **dependency injection**.
* Use **Repository + Service pattern**.

3. **Advanced Requirements**

* Implement **Global Exception Handling** middleware.
* Use **Fluent Validation** or custom attributes for request models.
* Add **Unit Tests** (xUnit / NUnit + Moq).
* Secure APIs with **JWT Authentication**.

# SQL Tasks

1. **Database Design**

* Create tables:
  + Patients (Id, FirstName, LastName, DOB, Gender, City, Email, Phone)
  + Conditions (Id, Name, Description)
  + PatientConditions (PatientId, ConditionId, DiagnosedDate)
* Write **DDL scripts** with constraints:
  + Primary/foreign keys
  + Unique Email
  + Proper indexing (clustered + non-clustered)

2. **SQL Queries**

* Fetch all patients **diagnosed in the last 6 months**.
* Get the **top 3 cities** with maximum patients.
* Find patients who have **more than 2 conditions**.
* Write a stored procedure to **search patients dynamically** (filters: age range, condition, city).
* Write a query to get the **average age of patients grouped by condition**.

3. **Optimization**

* Provide **execution plan analysis** for at least 1 query.
* Suggest **indexing strategies**.

# Sitecore Task – Patient Management Page

Document the **steps** (If Sitecore license is not available) to create a **Centralized Patient Management Page** in Sitecore that:

* Displays **static content** (editor-managed).
* Create Renderings, Templates, Placeholders.
* Make sure the rendering supports dynamic placeholder.
* The rendering should accept multiple themes.
* There needs to be datasource restriction at rendering level.
* Provides **Admin users** the ability to **Create / Update / Delete Patients** using the backend API.

# Deliverables

**GitHub repo** with:

* .NET API code
* SQL scripts
* README with instructions

# Evaluation Criteria

✅Clean, modular .NET code (SOLID principles, proper layering)  
✅ Error handling & validations  
✅ Unit testing quality  
✅ SQL schema design (normalization, indexing, constraints)  
✅ Query efficiency & correctness  
✅ Code readability + Git commit hygiene