Assignment 05

Last updated by | Hashim Javed | Aug 8, 2025 at 4:39 PM GMT+5

SQL Database Design Extension Assignment

1. Overview & Objectives

In this extension, you will translate the Patient Visit Manager console app into a fully normalized SQL database. You will:

- Design a relational schema covering all core entities
- Normalize tables through 1NF, 2NF, and 3NF
- Implement the schema via DDL and populate it with DML
- Write stored procedures for every CRUD operation
- Produce an ERD documenting tables and relationships

By the end, you'll have hands-on experience with end-to-end database design, implementation, and documentation.

2. Assignment Requirements

Part 1: Database Schema Design

1. Entity Identification
   - Review the original console app requirements and identify every distinct entity (e.g., Patient, Visit, Doctor, VisitType, UserRole, ActivityLog, FeeSchedule).
2. Table Schemas
   - For each entity, define:
     - A primary key
     - All attributes with appropriate SQL data types and constraints (NOT NULL, UNIQUE, etc.)
     - Any lookup tables (e.g., VisitType, Roles, FeeSchedule)

3. Relationships

- Establish foreign-key relationships (e.g., Patient → Visit, Doctor → Visit etc.).

4. Normalization

- Demonstrate the design in 1NF, 2NF, and 3NF:
  - 1NF: Eliminate repeating groups and ensure atomicity
  - 2NF: Remove partial dependencies on composite keys
  - 3NF: Remove transitive dependencies

Part 2: SQL Implementation

1. DDL Scripts (Project-1-DDL.sql)

- CREATE TABLE statements for each table, including keys and constraints.
- CREATE SCHEMA or USE statements if applicable.

2. DML Scripts (Project-1-DML.sql)

- INSERT statements to populate lookup tables (e.g., VisitType, Role, FeeSchedule).
- Sample data for Patients, Doctors, Visits, and ActivityLog (Just like you did in project 1).

3. Stored Procedures (Project-1-SP.sql)

- One SP per CRUD operation (e.g., stp_AddPatient etc.).
- You store procedure naming convention should be stp_[Name of SP], like in above and SP name should be meaningful.
- Ensure parameter validation and error handling.

4. ERD Generation

- Use a tool (e.g., SQL Server Management Studio, draw.io) to produce a clear ERD showing all tables, PKs, FKs, and relationships.

Part 3: File & Branch Organization

- Git Branch: project-1-db-design
- Files:
    1. project-01-DDL.sql
    2. project-01-DML.sql
    3. project-01-SP.sql

Include a top-of-file comment in each SQL file with your name, date, and a brief description.

3. Step-by-Step Guidelines

1. Kickoff & Planning: Read the original project spec and list entities and attributes. Sketch an initial ERD.
2. Schema Drafting: Translate your sketch into CREATE TABLE statements. Annotate each table with its normalization level.
3. Normalization Review: Verify 1NF, 2NF, and 3NF compliance; note any changes.
4. Implementation: Write and test your DDL; populate lookup tables via DML; generate sample data.
5. Stored Procedures: Implement SPs covering Create, Read, Update, Delete; test edge cases.
6. ERD Finalization: Reverse-engineer your implemented schema to produce a polished ERD diagram with cardinalities.
7. Documentation: Explain design decisions, normalization steps, and how to run scripts and SPs.

4. Submission Guidelines

Deadline: Monday, August 11th (deliver when you reach office before 11:00 AM)

- **Deliverables:**
  • Git branch project-1-db-design with three SQL files

  • ERD diagram (PDF or image)
  • README (Explain design decisions, normalization steps, and how to run scripts and SPs.)

- **Quality:**
  • All relationships enforced via FK constraints
  • Scripts execute without errors on a fresh database
  • ERD is legible and complete

5. Grading Criteria

- Criteria Weight
- Schema Design & Normalization 30%
- SQL Implementation 30%
- Stored Procedures 20%
- ERD & Documentation 10%
- Organization & Submission 10%