Problem Statement

You are designing a backend system for a multi-tenant medical coding platform that processes patient encounters for healthcare providers.

Each encounter must support:

- Diagnosis codes (ICD-10)
- Procedure codes (CPT / HCPCS)
- Modifier codes that can be applied to procedure codes

The platform also manages:

- Organizations (tenants)
- Providers within each organization
- Patients associated with providers
- Encounters associated with patients

You must design:

Part A — Database Design

Design an ERD + schema that supports the following:

1. Multi-Tenant Structure

- Multiple organizations use the system.
- Each organization has multiple providers.
- Providers have multiple patients and encounters.

2. Medical Coding

Your schema must support:

- ICD-10 diagnosis codes
- CPT/HCPCS procedure codes
- Modifier codes
- Ability to attach multiple diagnoses to an encounter
- Ability to attach multiple procedures to an encounter
- Ability to attach multiple modifiers to each procedure on an encounter

3. Modifier Rules

Some modifiers are valid only for certain procedure codes.
Your DB must support storing these rules.

4. Versioning

All code sets (diagnosis, procedure, modifier) must support versioning
(e.g., effective dates for updates).

Deliverables

Provide:

- ERD diagram
- Table definitions (name, fields, PK/FK, constraints)

Part B — API Design

Design REST APIs that support:

1. Organization, Provider, Patient, and Encounter Management

Include CRUD or minimal lifecycle endpoints as appropriate.

2. Retrieval of Code Sets

Endpoints to query:

- Diagnosis codes
- Procedure codes
- Modifier codes

(Search/filtering optional)

3. Assigning Codes to Encounters

Design endpoints to:

- Add diagnoses to an encounter
- Add procedures to an encounter
- Add modifiers to a specific procedure within an encounter

4. Modifier Rule Lookup

Provide an endpoint that returns valid modifiers for a procedure code.

Deliverables

Provide:

- Endpoint list with URLs
- HTTP methods
- Request/response bodies

Part C — Short Explanation

Provide a short write-up explaining your design decisions, including:

- How multi-tenancy is handled
- How you represented many-to-many relationships
- Why your versioning approach works

Evaluation Criteria

- DB normalization & correctness
- Handling of modifier–procedure relationships
- Logical API structure
- Clear reasoning

****