Entity Explanation:

User
The User entity represents an individual who interacts with the system. It stores personal information such as the Username and Password, and remains separate from educational content. By keeping the user entity distinct from other entities, such as courses or problems, it becomes easier to manage user-specific data independently of the learning material.

Enrollments
The Enrollments entity acts as a bridge between the User and Courses entities, representing the many-to-many relationship between them. Since a user can enroll in multiple courses, and a course can have many users, the enrollments table captures each unique pairing of a user with a course. Additionally, this entity stores details specific to each enrollment, such as the user's Score in a course, which provides more detailed tracking of the relationship between users and courses.

Courses
The Courses entity tracks all the courses offered within the system, including their unique Topic. It ensures that courses can be categorized and managed effectively, allowing the system to handle various topics, durations, and learning objectives. This separation helps maintain clarity and makes it easy to organize and modify educational content.

Tutor
The AI tutor entity represents the different tutors within the system, with each tutor specializing in a specific topic. The AI tutors are designed to be a separate entity rather than an attribute of another entity in order to allow for a more tailored learning experience, which may involve certain learning materials and teaching styles that are specific to each subject.

Problems
The problems entity represents the different questions that the tutor can ask and answer for a specific subject. Since we have unique questions and explanations for each question that go under a specific course topic, we have designed the problems to be a separate entity instead of an attribute of another entity.

Relation Explanation:

User -- Enrollments:

A user might be enrolled in many classes so there is a one to many relationship between them, but each enrollment has only one user. This means that for every record in the Enrollments table, there is only one corresponding User.

Enrollments -- Courses:

There are multiple enrollments for every course so this represents a many to one relationship. Since every class can have more than one student, there are more than one enrollment records corresponding to each course.

User -- Tutor:
A single user can be enrolled in several different courses that cover different topics, so a user might require assistance from multiple AI tutors. AI tutors can also help multiple users since there can be multiple users who need help with the same topic. Since a user can be assigned multiple tutors depending on what courses they are taking and an AI tutors can be assigned to multiple students who need help with the same topic, there is a many to many relationship between User and Tutor.

Courses -- Problems:
Every course has a set of multiple associated problems and also a question can belong to more than one course which indicates a many to many relationship between Courses and Problems. The problem is a separate entity from the course as we have additional attributes when it comes to Problems such as difficulty and answers.

Problems -- Tutor:

A single tutor can explain multiple problems and a problem can be explained by multiple tutors which means it is a many to many relationship. A tutor can specialize in explaining specific courses/topics hence they are associated with multiple problems. A Problem may also have different explanations as each tutor can cover different courses/topics.

Normalizing Database and Relational Schema:

All attributes are atomic as there are no repeating groups or arrays. Each field contains only indivisible values. Thus it meets 1NF

In order to achieve 2NF, Non-key attributes are fully functionally dependent on the primary key, Problems entity is split into two entities. One representing the **Topic** with topicID and one representing the **Problems** with ProblemID. Courses and Topic have a many to 1 relationship since many courses could have the same topic. Topic to problems is 1 to many relationship, as one topic can correspond to many problems.

Updated Entities:
Courses:
CourseId int
TopicId int

Topic:
TopicId int
QuestionId int

Problems:
QuestionId int
QuestionContent varchar(250)
Answers varchar(25)
Difficulty int

Tutor:
TutorId int
TopicId int
Release_dt date

User.UserId (attributes depend only on primary key)
Enrollments.UserId + Enrollments.CourseId (Composite Key)
Courses.CourseId (attributes depend only on primary key)
Topic.TopicId (attributes depend only on primary key)
Problems.QuestionId (attributes depend only on primary key)
Tutor.TutorId (attributes depend only on primary key)

Since all non-key attributes are dependent on the primary key it is 2NF

For 3NF, we already proved that non-key attributes for each entity depend solely on the primary key for all the entities. Thus, this is in Third Normal Form. The final entities will look like the following

User: UserId(int), username(varchar(50)), Password(varchar(5))
Enrollments: UserId(int), CourseId(int), Score(int) (Score is dependent on both the UserId and CourseId)
Courses: CourseId(int), TopicId(int)
Topic: TopicId(int), QuestionId(int),
Problems: QuestionId(int), QuestionContent varchar(250), Answers varchar(25), Difficulty (int)
Tutor: TutorId(int), TopicId (int) Release_dt (date)