Assignment 2 Solutions

1. First, Second, and Third Normal Forms (1NF, 2NF, 3NF)

To achieve the normalized forms (1NF, 2NF, and 3NF), we start by analyzing the attributes in the original table:
- Lec_Name, Stu_No, Stu_Name, Age, Course, Project_Name, Project_Desc, Lec_Contact.

Assumptions: We assume that a Lecturer can have multiple students, each student can have only one project, and each project belongs to a specific course.
Using these assumptions, we can perform the following normalizations:

1. **First Normal Form (1NF)**: In 1NF, we eliminate any repeating groups by ensuring each attribute contains only atomic values.
2. **Second Normal Form (2NF)**: In 2NF, we remove partial dependencies by separating attributes into tables based on their dependency.
3. **Third Normal Form (3NF)**: In 3NF, we ensure transitive dependencies are removed.

Following these steps, we achieve three distinct tables.

For example, the 1NF table might look like this:

Stu_No	Stu_Name	Age	Course	Project_Name	Lec_Name	Lec_Contact
001	John Doe	20	Math	Calculus	Dr. Lee	123-456-789
...						

In 2NF and 3NF, these tables will be further divided to eliminate partial and transitive dependencies.

2. ER Diagram of Normalized Entities

The ER Diagram for the normalized tables shows the following relationships:
1. **Lecturer** entity connected to **Student** with a one-to-many relationship.
2. **Student** entity linked to **Project** with a one-to-one relationship.
3. **Course** linked to **Project** in a many-to-one relationship.

The ER Diagram illustrates these relationships in a clear, relational format.

3. SQL Code for Normalized Tables

Based on the normalized tables, here is an SQL code that defines primary keys, foreign keys, and constraints.

```sql
CREATE TABLE Lecturer (
   Lec_ID INT PRIMARY KEY,
   Lec_Name VARCHAR(50),
   Lec_Contact VARCHAR(15)
);

CREATE TABLE Student (
   Stu_No INT PRIMARY KEY,
   Stu_Name VARCHAR(50),
   Age INT,
   Course VARCHAR(50),
   Lec_ID INT,
   FOREIGN KEY (Lec_ID) REFERENCES Lecturer(Lec_ID)
);

CREATE TABLE Project (
   Project_ID INT PRIMARY KEY,
   Project_Name VARCHAR(50),
   Project_Desc TEXT,
   Stu_No INT UNIQUE,
   FOREIGN KEY (Stu_No) REFERENCES Student(Stu_No)
);
```
These tables reflect the structure from 3NF, with each table containing only necessary dependencies.

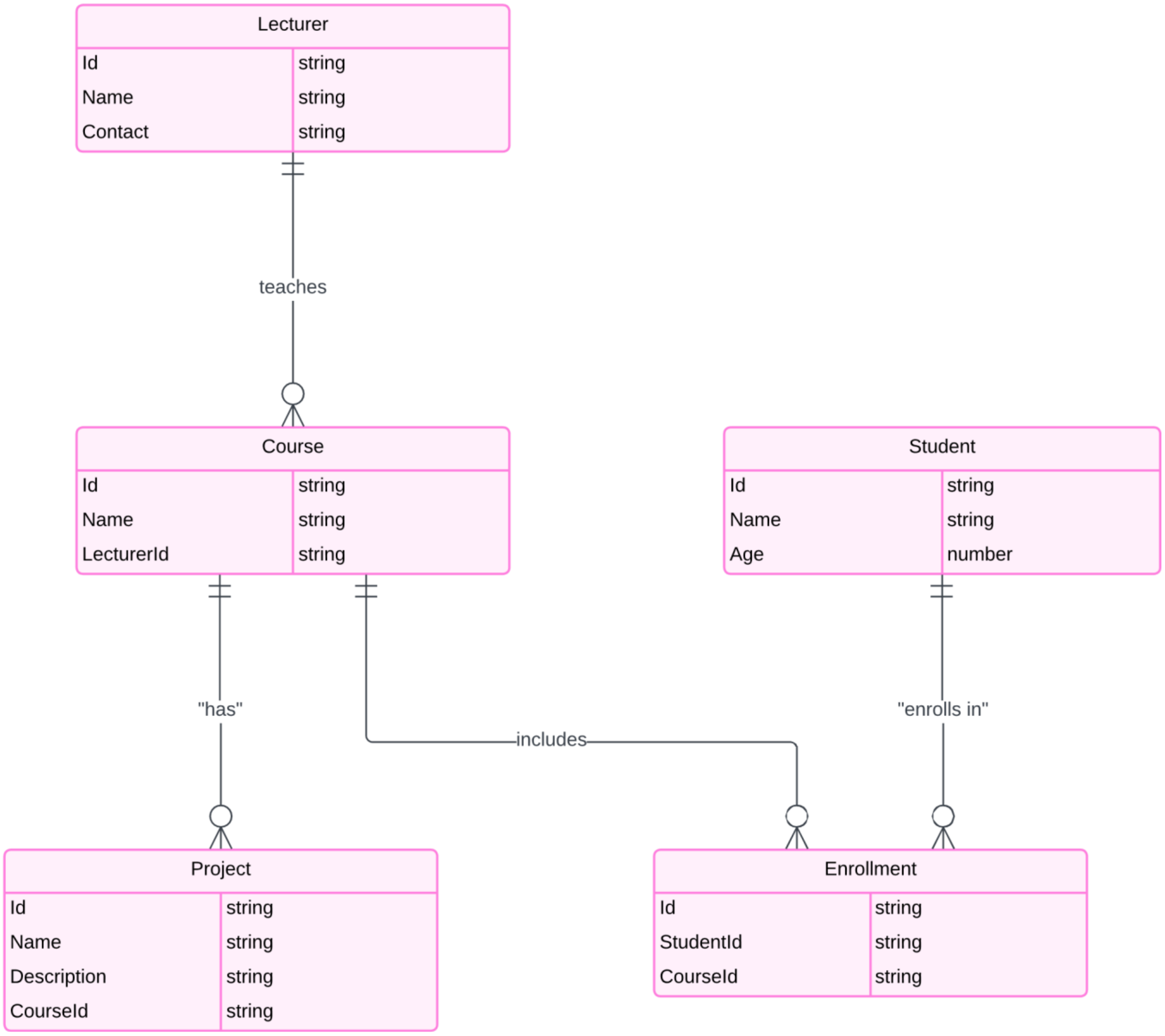
4. Mapping Cardinalities and Diagram

Mapping cardinalities define the types of relationships between entities:

1. **Lecturer to Student**: One-to-many (a lecturer can have many students, but each student has only one lecturer).

2. **Student to Project**: One-to-one (each student has a unique project).
3. **Project to Course**: Many-to-one (each project belongs to a course, but a course can contain multiple projects).

Diagrammatically, this is represented with connecting lines and cardinality notations.


Lecturer
Id string
Name string
Contact string
teaches
Course
Id string
Name string
LecturerId string
Student
Id string
Name string
Age number
"has"
includes
"enrolls in"
Project
Id string
Name string
Description string
CourseId string
Enrollment
Id string
StudentId string
CourseId string