Assignment: Normalization Practice

Imagine you have been given a table named "StudentCourses" that contains information about students and the courses they are enrolled in. Analyze the table, identify normalization issues, and then proceed to normalize it.

Original "**StudentCourses**" Table:

Student_ID	Student_Name	Course1	Course2	Course3
101	Alice	Math	History	Physics
102	Bob	Physics	Null	Null
103	Carol	Chemistry	Biology	Null

Assignment Tasks:

1. Identify Normalization Issues:

   List the normalization issues present in the "StudentCourses" table. Explain why these issues violate normalization rules.

2. Normalize the Table:

   Create one or more normalized tables that represent the same data without the identified normalization issues.

Explain how you have transformed the original table to achieve normalization.

Answer:

The “StudentCourses” table provided exhibits several normalization issues:

1. Repeating Groups:

   The columns “Course1”, “Course2”, “Course3” imply a repeating group structure. It suggests that a student can enroll in multiple courses, but the number of courses is limited to three due to the fixed columns, violating the first normal form (1NF).

2. Data Redundancy:

   Null values in columns like “Course2” and “Course3” indicate potential redundancy in the table structure. This can lead to wasted storage space and difficulty in querying the data efficiently.

To normalize the “StudentCourses” table and address these normalization issues, we can create three separate tables: Students, Courses, and a junction table to represent the many-to-many relationship between students and courses.

Here’s an example of the normalized structure:

❖ Students Table:

Student_ID	Student_Name
101	Alice
102	Bob
103	Carol

<u>Columns:</u> Student_ID (Primary Key), Student_Name. This table holds unique student information.

❖ Courses Table:

Course_ID	Course_Name
1	Math
2	Physics
3	Chemistry
4	Biology
5	History

<u>Columns:</u> Course_ID (Primary Key), Course_Name. This table contains unique course details.

- StudentCourseID Junction Table:

StudentCourseID	Student_ID	Course_ID
1	101	1
2	101	2
3	101	5
4	102	2
5	103	3
6	103	4

Columns: StudentCourseID (Primary Key), Student_ID (Foreign Key referencing Students table), Course_ID (Foreign Key referencing Courses table).

This table establishes the many-to-many relationship between students and courses.

After normalization:

- The Students table stores individual student details without redundancy.
- The Courses table contains distinct course information without repetition.
- The StudentCourseID junction table links students to their enrolled courses, allowing flexibility for multiple enrollments and avoiding the fixed-column issue.