Normalization in MS SQL (Simple Explanation)

Normalization is the process of organizing data in a database to reduce redundancy (duplicate data) and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them.

Why Normalize?

- **Eliminate redundant data:** Reduce duplicate storage.
- **Ensure data integrity:** Keep data consistent.
- **Make database easier to maintain and update.**

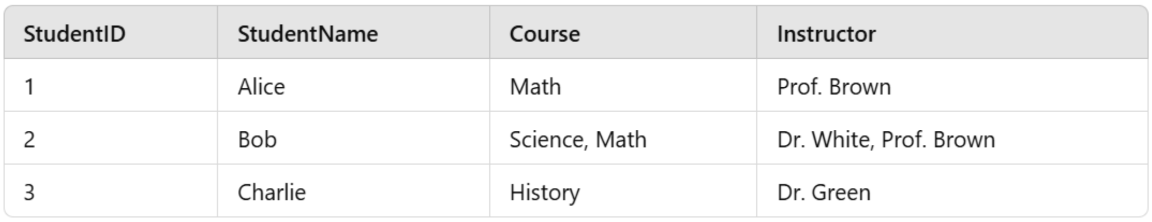
Normalization Process

Normalization is done in stages, called **Normal Forms (NF)**. Each stage builds on the previous one.

Example: Student Database

Step 1: Unnormalized Table (UNF)

Suppose we have a single table:



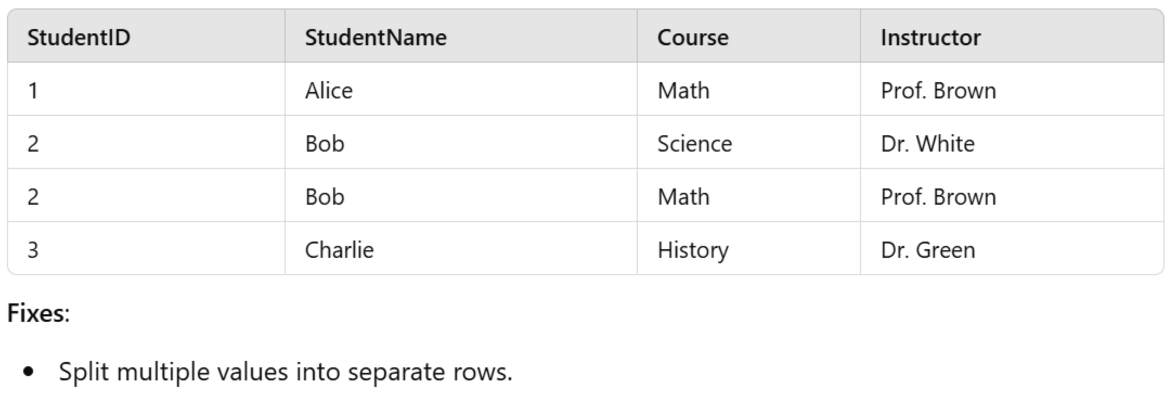
StudentID	StudentName	Course	Instructor
1	Alice	Math	Prof. Brown
2	Bob	Science, Math	Dr. White, Prof. Brown
3	Charlie	History	Dr. Green

Problems:

1. **Data redundancy** (Instructor "Prof. Brown" appears multiple times).
2. **Difficult to query** (Courses and instructors stored in the same field).

Step 2: First Normal Form (1NF)

Rule: Data must be atomic (no multiple values in a single cell).



StudentID	StudentName	Course	Instructor
1	Alice	Math	Prof. Brown
2	Bob	Science	Dr. White
2	Bob	Math	Prof. Brown
3	Charlie	History	Dr. Green

Fixes:

- Split multiple values into separate rows.

Step 3: Second Normal Form (2NF)

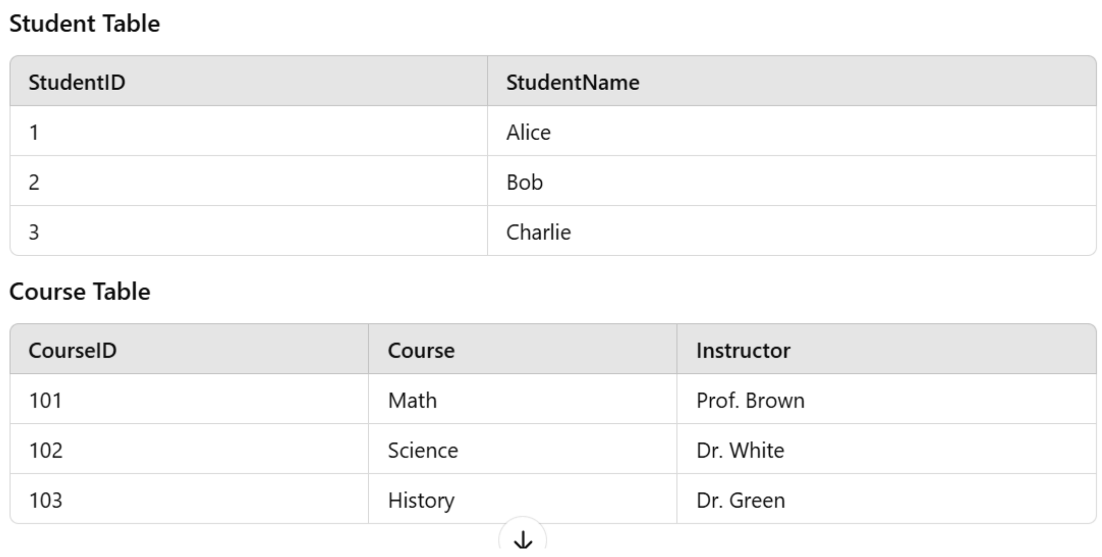
Rule: Eliminate partial dependency (non-key columns must depend on the entire primary key).

In the 1NF table:

- `Course` and `Instructor` depend only on `StudentID`, but this causes duplication for `Course` and `Instructor`.

Solution: Create separate tables:

Student Table

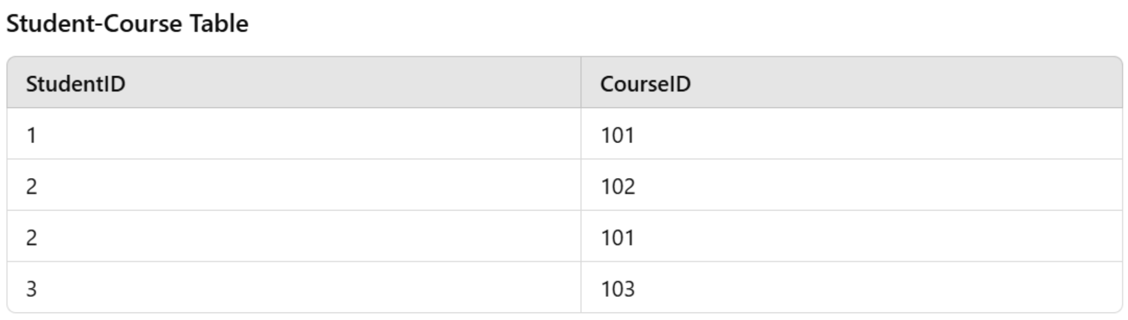


Student Table

StudentID	StudentName
1	Alice
2	Bob
3	Charlie

Course Table

CourseID	Course	Instructor
101	Math	Prof. Brown
102	Science	Dr. White
103	History	Dr. Green



Student-Course Table

StudentID	CourseID
1	101
2	102
2	101
3	103

Step 4: Third Normal Form (3NF)

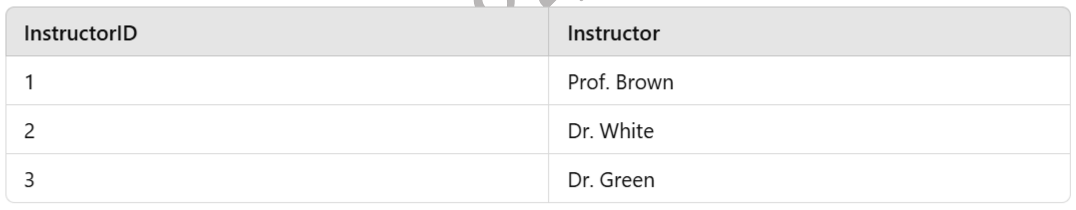
Rule: Eliminate transitive dependency (non-key columns must depend only on the primary key).

In the 2NF tables:

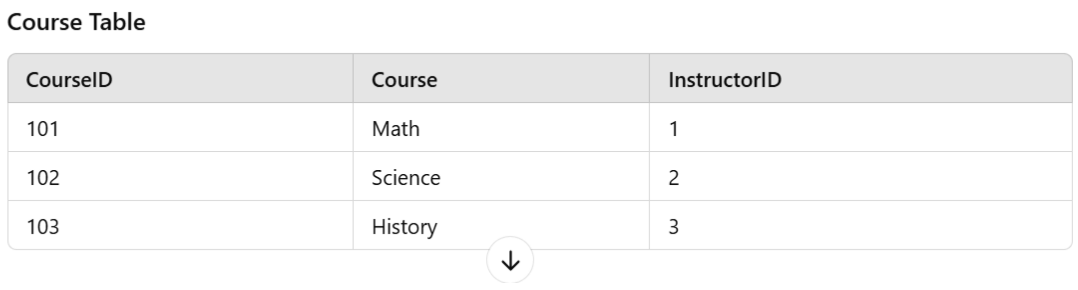
- `Instructor` in the `Course` table depends on `Course`, not directly on `CourseID`.

Solution: Create an additional table for instructors:

Instructor Table



InstructorID	Instructor
1	Prof. Brown
2	Dr. White
3	Dr. Green



Course Table

CourseID	Course	InstructorID
101	Math	1
102	Science	2
103	History	3

Benefits of Normalization

1. **Reduced Redundancy:** Instructor and courses are stored only once.
2. **Improved Data Integrity:** If an instructor's name changes, update only in the `Instructor` table.
3. **Simpler Queries:** Organized relationships make queries easier to write and maintain