NORMALIZATION IN DBMS

Developed by Swapnjeet S

ORGANIZING DATA FOR EFFICIENCY AND ACCURACY

What is Normalization?

📝 **Definition:**

A database design technique to minimize redundancy and dependency by organizing fields and tables.

🎯 **Goal:**

To ensure data integrity and reduce anomalies in data manipulation.

Functional Dependency

It is a relationship that exist between two sets of attributes of a relational table where one set of attributes can determine value of other set of attributes.

Denoted by X ➞ Y; where X is determinant and Y is dependent.

Partial dependency: A non-primary column depends on a single column that is part of a composite primary key.

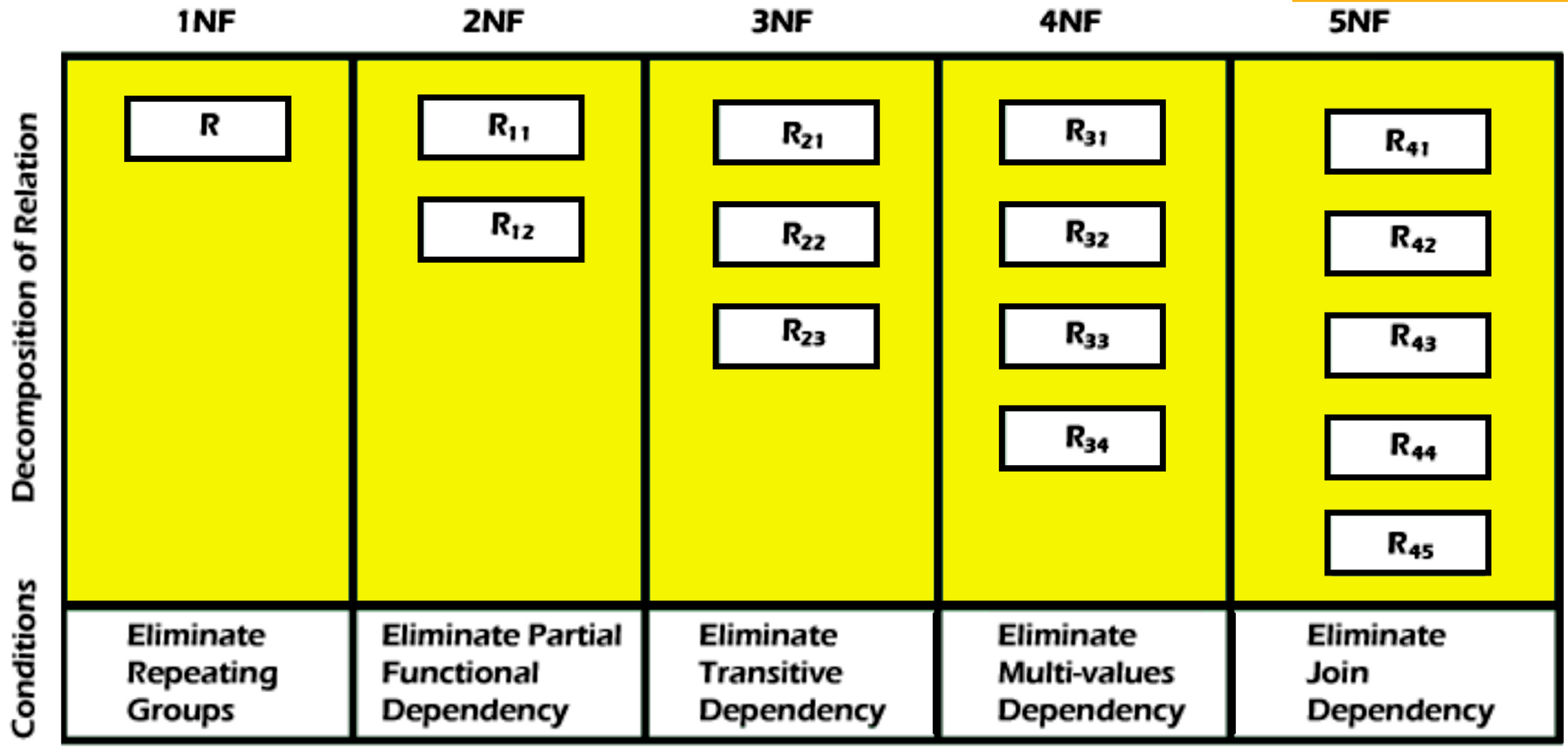
Transitive dependency: A non-primary column depends on another non-primary column, creating an indirect relationship between columns in the same table. Transitive dependencies only exist in tables with at least three columns.

Full functional dependency: A state of normalization that meets the requirements of 1NF and ensures that all non-key attributes are fully functionally dependent on the primary key.

Types of Normal Forms

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

Normalization

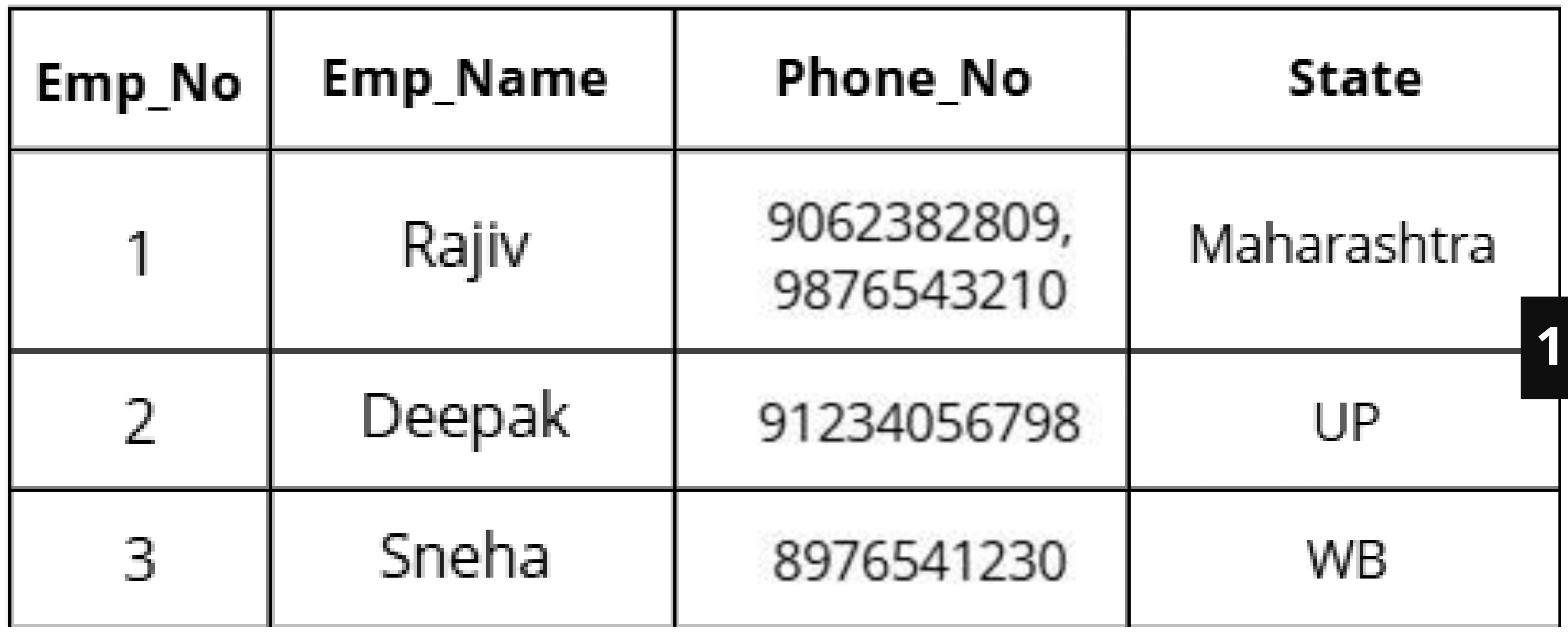

1NF
2NF
3NF
4NF
5NF
Decomposition of Relation
Conditions
R
R_{11}
R_{12}
R_{21}
R_{22}
R_{23}
R_{31}
R_{32}
R_{33}
R_{34}
R_{41}
R_{42}
R_{43}
R_{44}
R_{45}
Eliminate Repeating Groups
Eliminate Partial Functional Dependency
Eliminate Transitive Dependency
Eliminate Multi-values Dependency
Eliminate Join Dependency

1NF

First Normal Form

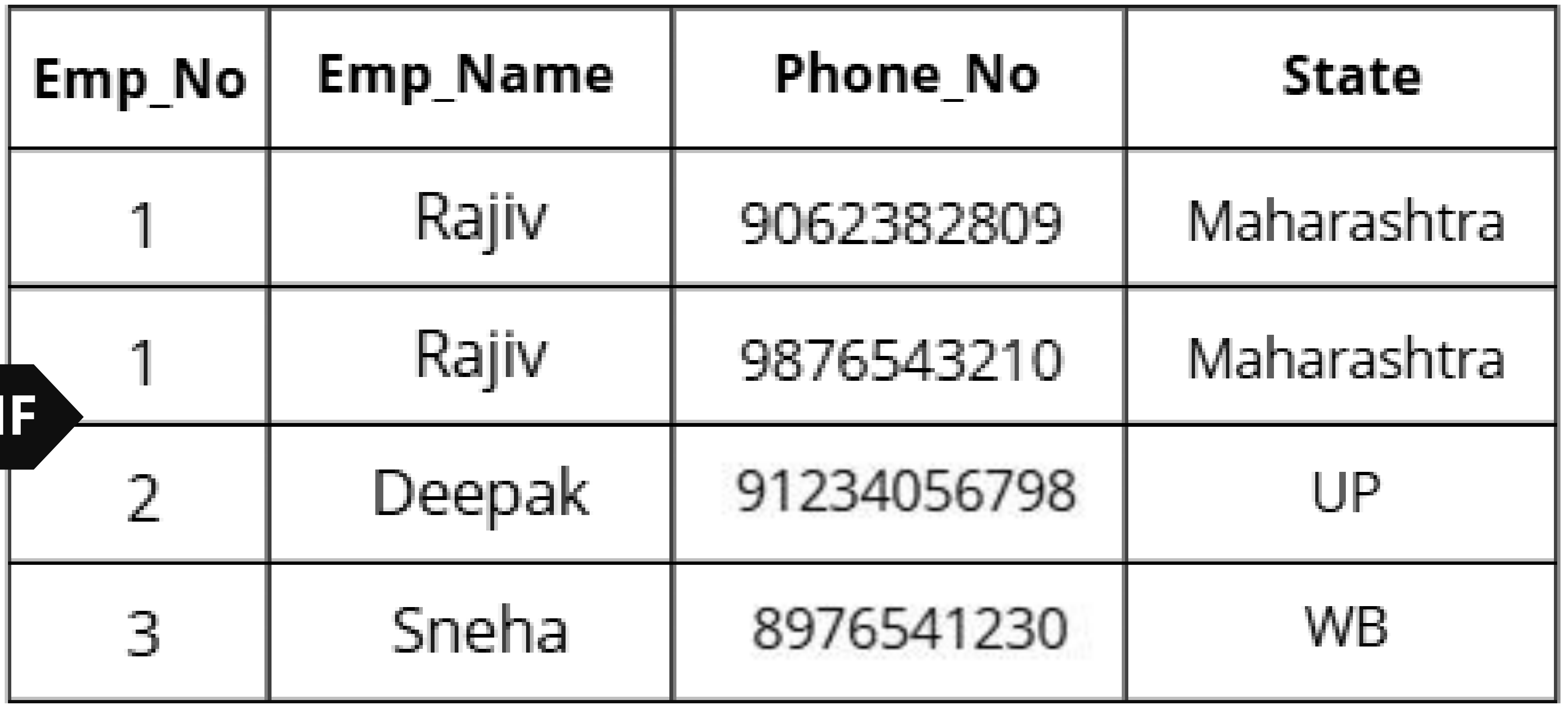
In 1NF, each table cell should contain only a single value, and each column should have a unique name.

E.g: let’s take this **Employee** Table



Emp_No	Emp_Name	Phone_No	State
1	Rajiv	9062382809, 9876543210	Maharashtra
2	Deepak	91234056798	UP
3	Sneha	8976541230	WB

1NF



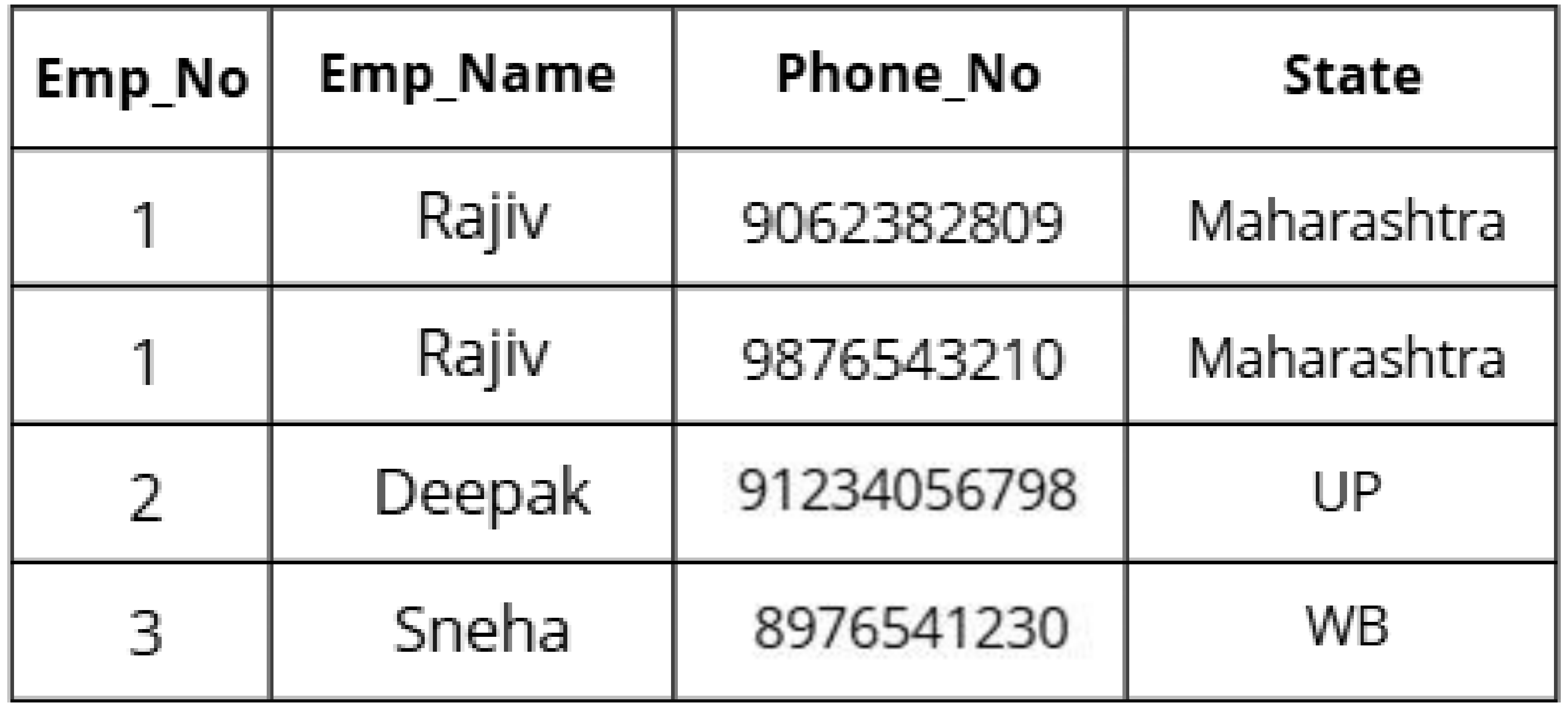
Emp_No	Emp_Name	Phone_No	State
1	Rajiv	9062382809	Maharashtra
1	Rajiv	9876543210	Maharashtra
2	Deepak	91234056798	UP
3	Sneha	8976541230	WB

2NF

Second Normal Form

2NF eliminates redundant data by requiring that each non-key attribute be dependent on the primary key. This means that each column should be directly related to the primary key, and not to other columns.

E.g: let's take this 1NF **Employee** Table

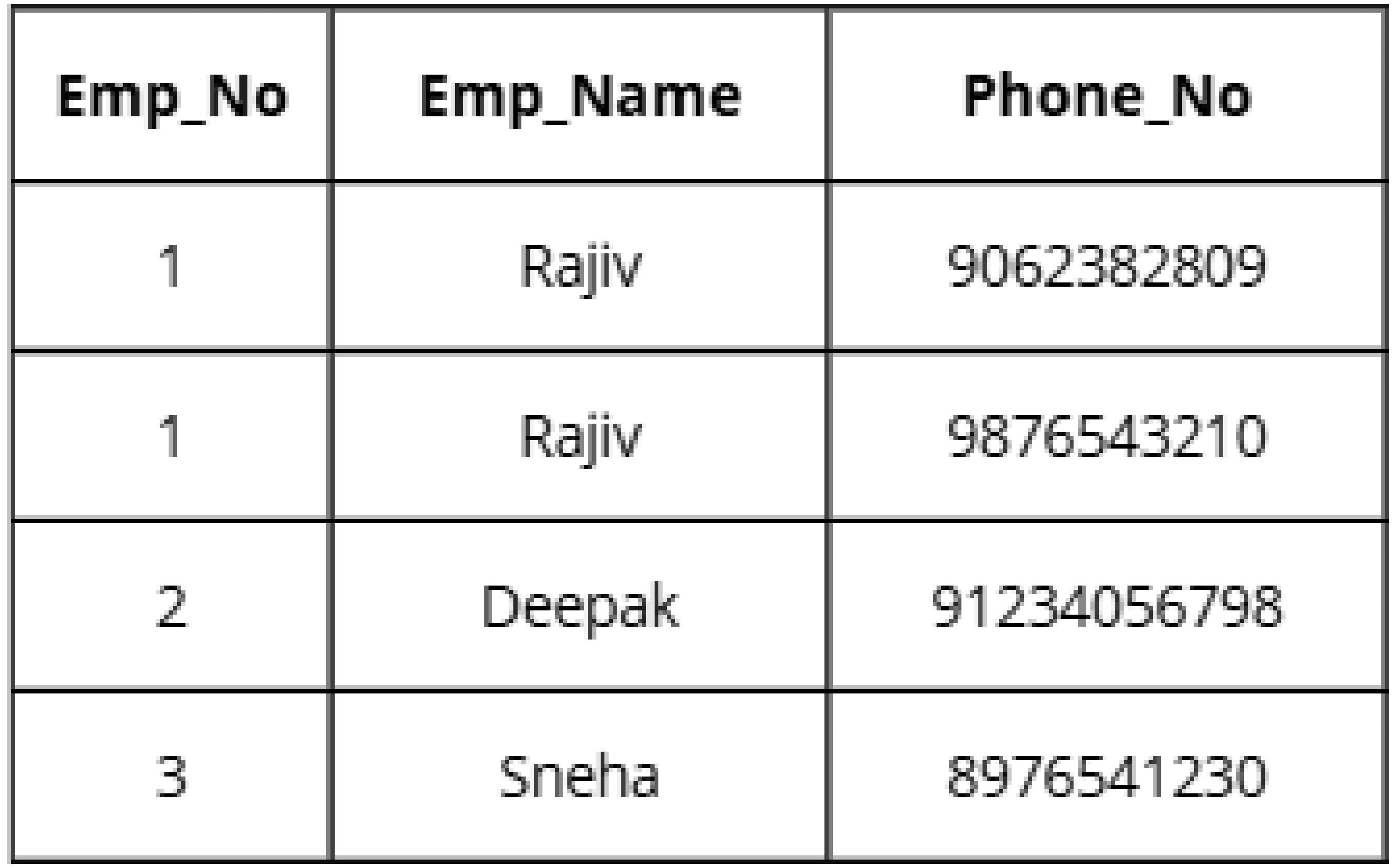


Emp_No	Emp_Name	Phone_No	State
1	Rajiv	9062382809	Maharashtra
1	Rajiv	9876543210	Maharashtra
2	Deepak	91234056798	UP
3	Sneha	8976541230	WB

Normalization

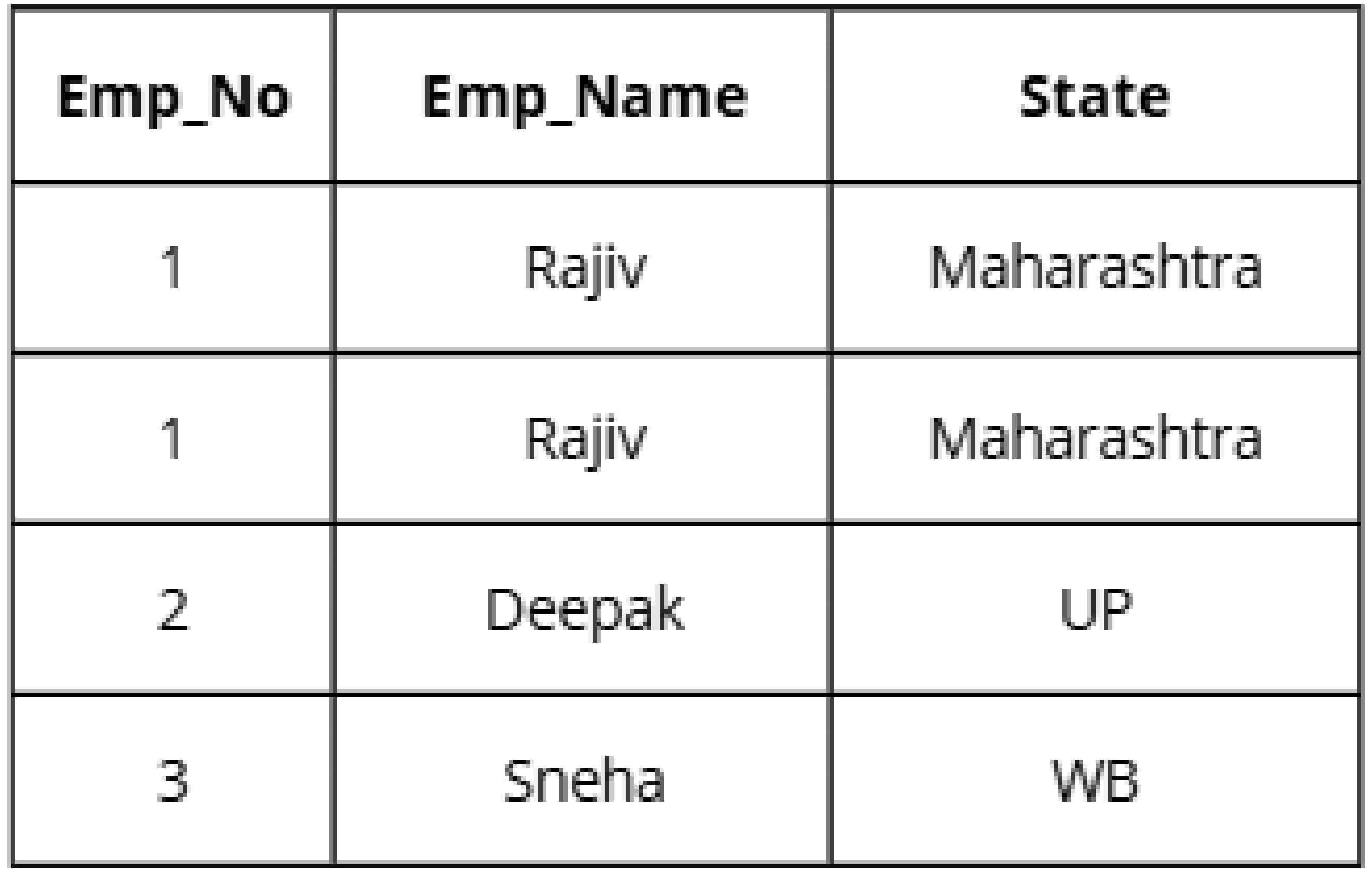
To convert the given table into 2NF, we decompose it into two tables:

Table 1: **EMP_PhoneNo**



Emp_No	Emp_Name	Phone_No
1	Rajiv	9062382809
1	Rajiv	9876543210
2	Deepak	91234056798
3	Sneha	8976541230

Table 2: **EMP_State_data**



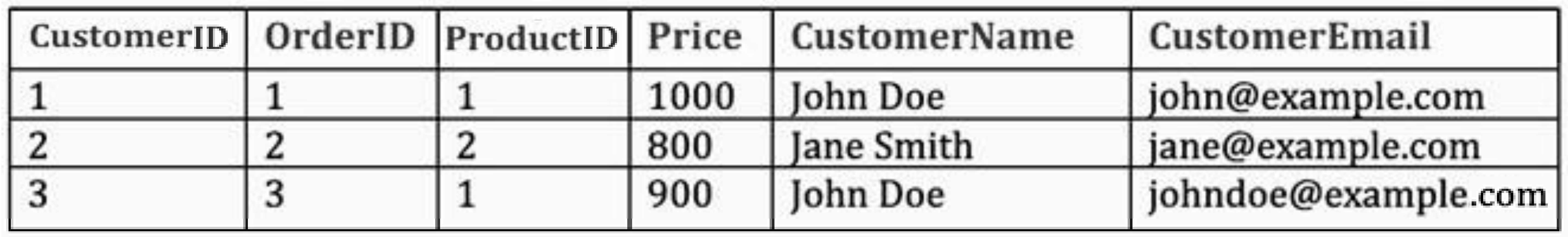
Emp_No	Emp_Name	State
1	Rajiv	Maharashtra
1	Rajiv	Maharashtra
2	Deepak	UP
3	Sneha	WB

3NF

Third Normal Form

This builds on 1NF and 2NF by removing any data that depends on non-key attributes (attributes that aren't part of the primary key).

Here's an example: suppose we have a table named **Sales_Data**



CustomerID	OrderID	ProductID	Price	CustomerName	CustomerEmail
1	1	1	1000	John Doe	john@example.com
2	2	2	800	Jane Smith	jane@example.com
3	3	1	900	John Doe	johndoe@example.com

Normalization

To convert the given table into 3NF, we decompose it into three tables:

Table1: **Customers**

```
+------------+--------------+---------------------+
| CustomerID | CustomerName | CustomerEmail       |
+------------+--------------+---------------------+
| 1          | John Doe     | john@example.com    |
| 2          | Jane Smith   | jane@example.com    |
| 3          | John Doe     | johndoe@example.com |
+------------+--------------+---------------------+
```

Table2: **Orders**

```
+---------+------------+-----------+-------+
| OrderID | CustomerID | ProductID | Price |
+---------+------------+-----------+-------+
| 1       | 1          | 1         | 1000  |
| 2       | 2          | 2         | 800   |
| 3       | 3          | 1         | 900   |
+---------+------------+-----------+-------+
```

Table3: **Products**

```
+-----------+-------------+
| ProductID | ProductName |
+-----------+-------------+
| 1         | Product A   |
| 2         | Product B   |
+-----------+-------------+
```

BCNF

Boyce-Codd Normal Form

BCNF ensures that each determinant (attribute which determine another attributes) is a unique identifier for that attribute or must be a super key. A table is in BCNF if every functional dependency **X➞Y**, X is the super key of the table.

Consider a table **Course_Enrollment**: **>>**

```
+-------------+-------------+--------------+
| StudentID   | CourseID    | Instructor   |
+-------------+-------------+--------------+
| 1           | 101         | Smith        |
| 2           | 102         | Johnson      |
| 1           | 102         | Johnson      |
| 3           | 101         | Smith        |
+-------------+-------------+--------------+
```

Functional Dependencies:

StudentID, CourseID ➞ Instructor
Instructor ➞ CourseID

(assuming an instructor only teaches one course)

Normalization

The first dependency (`StudentID, CourseID ➞ Instructor`) is fine because `StudentID, CourseID` together form a superkey. The second dependency (`Instructor ➞ CourseID`) is problematic because `Instructor` is not a superkey (it doesn’t uniquely identify all the rows).

To achieve BCNF, decompose the table into two tables:

Table1: **Course_Instructor**

```
+----------+------------+
| CourseID | Instructor |
+----------+------------+
| 101      | Smith      |
| 102      | Johnson    |
+----------+------------+
```

Table2: **Student_Course**

```
+-----------+----------+
| StudentID | CourseID |
+-----------+----------+
| 1         | 101      |
| 2         | 102      |
| 1         | 102      |
| 3         | 101      |
+-----------+----------+
```

4NF

Fourth Normal Form

A relation will be in 4NF if it is in BCNF and has no multi-valued dependency. For a dependency A ➞ B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

The given **STUDENT** table is in 3NF, but the Course and Hobby are two independent entity. Hence, there is no relationship between Course and Hobby.

StudentID 21 contains **two courses**, **Computer** & **Maths** and **two hobbies**, **Dancing** & **Singing**. So there is a Multi-valued dependency on Student ID, which leads to unnecessary repetition of data.

StudentID	Course	Hobby
21	Computer	Dancing
21	Maths	Singing
34	Chemistry	Cricket
59	Physics	Hockey

Normalization

So to make the STUDENT table into 4NF, we can decompose it into two tables:

Table1: **Student_Course**

StudentID	Course
21	Computer
21	Maths
34	Chemistry
59	Physics

Table2: **Student_Hobby**

StudentID	Hobby
21	Dancing
21	Singing
34	Cricket
59	Hockey

5NF

Fifth Normal Form

A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless. 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy. 5NF is also known as Project-Join Normal Form (PJNF).

Suppose, we've a table **SEM_LECTURE**. Here, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Subject	Lecturer	Semester
Computer	Joe	Semester1
Computer	John	Semester1
Maths	John	Semester1
Maths	Akash	Semester2

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1:

Semester	Subject
Semester1	Computer
Semester1	Maths
Semester2	Maths

P2:

Subject	Lecturer
Computer	Joe
Computer	John
Maths	John
Maths	Akash

P3:

Semester	Lecturer
Semester1	Joe
Semester1	John
Semester2	Akash

Advantages of
Normalization in DBMS

- **Reduced data redundancy:** helps to eliminate duplicate data in tables, reducing the amount of storage space needed and improving database efficiency.
- **Improved data consistency:** ensures that data is stored in a consistent and organized manner, reducing the risk of data inconsistencies and errors.
- **Simplified database design:** provides guidelines for organizing tables and data relationships, making it easier to design and maintain a database.
- **Improved query performance:** tables are typically easier to search and retrieve data from, resulting in faster query performance.
- **Easier database maintenance:** reduces the complexity of a database by breaking it down into smaller, more manageable tables.

Disadvantages of Normalization in DBMS

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e. 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to affect query performance and serious problems.