https://www.youtube.com/watch?v=UrYLYV7WSHM

A relation is in 1NF if it contains an atomic value. 2NF. A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. 3NF. A relation will be in 3NF if it is in 2NF and no transition dependency exists

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee Name | Phone Number | Salary |
| EDU001 | Alex | 8097043699  9870571480 | 5000 |
| EDU002 | Barry | 8378737383 | 6000 |
| EDU003 | Costa | 9595895859  9474848474 | 3000 |
|  |  |  |  |

1NF: In this Normal Form, we tackle the problem of atomicity. Here atomicity means values in the table should not be further divided. In simple terms, a single cell cannot hold multiple values. If a table contains a composite or multi-valued attribute, it violates the First Normal Form.

In the above table, we can clearly see that the Phone Number column has two values. Thus it violated the 1st NF. Now if we apply the 1st NF to the above table we get the below table as the result.

|  |  |  |  |
| --- | --- | --- | --- |
| Employee ID | Employee Name | Phone Number | Salary |
| EDU001 | Alex | 8097043699 | 5000 |
| EDU001 | Alex | 9870571480 | 5000 |
| EDU002 | Barry | 8378737383 | 6000 |
| EDU003 | Costa | 9595895859 | 3000 |
| EDU003 | Costa | 9474848474 | 3000 |

By this, we have achieved atomicity and also each and every column have unique values.

### 2NF: 2nd Normal Form (2NF)

The first condition in the 2nd NF is that the table has to be in 1st NF. The table also should not contain partial dependency. Here partial dependency means the proper subset of candidate key determines a non-prime attribute. To understand in a better way lets look at the below example.

|  |  |  |
| --- | --- | --- |
| Emp ID | Dept ID | Office Loc |
| EDU001 | ED-T1 | Pune |
| EDU002 | ED-S2 | Delhi |
| EDU003 | ED-T3 | Mumbai |

Consider the table

This table has a composite [primary key](https://www.edureka.co/blog/primary-key-in-sql/) **Emplyoee ID**, **Department ID**. The non-key attribute is **Office Location**. In this case, **Office Location** only depends on **Department ID**, which is only part of the primary key. Therefore, this table does not satisfy the second Normal Form.

To bring this table to Second Normal Form, we need to break the table into two parts. Which will give us the below tables:

|  |  |
| --- | --- |
| Dept ID | Office Loc |
| ED-T1 | Pune |
| ED-S2 | Delhi |
| ED-T3 | Mumbai |

As you can see we have removed the partial functional dependency that we initially had. Now, in the table, the column **Office Location** is fully dependent on the primary key of that table, which is **Department ID**.

3NF:

### ****3rd Normal Form (3NF)****

The same rule applies as before i.e, the table has to be in 2NF before proceeding to 3NF. The other condition is there should be no transitive dependency for non-prime attributes. That means non-prime attributes (which doesn’t form a candidate key) should not be dependent on other non-prime attributes in a given table. So a transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and Y → Z (where it is not the case that Y → X)

Let’s understand this more clearly with the help of an example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudentID | Student Name | Subject Id | Subject | Address |
| 101 | Alex | S1 | SQL | Goa |
| 102 | Barry | S2 | JAVA | Mumbai |
| 103 | Costa | S3 | PYTHON | Delhi |

In the above table, **Student ID** determines **Subject ID**, and **Subject ID** determines **Subject**. Therefore, **Student ID** determines **Subject** via **Subject ID.**This implies that we have a transitive functional dependency, and this structure does not satisfy the third normal form.