Certainly! The terms 1NF (First Normal Form), 2NF (Second Normal Form), and 3NF (Third Normal Form) are stages of normalization in database design. Let's define each one with an example:

**1NF (First Normal Form):**

**Definition:**   
In simple terms, 1NF ensures that each value in a table is a single, indivisible piece of information, but it doesn't necessarily dictate that each entire record must be unique. Uniqueness at the record level is generally ensured by the primary key constraint in the database design.

**Example:** Consider a denormalized table named **StudentCourses**:

| **StudentID** | **Courses** |
| --- | --- |
| 1 | Math, English |
| 2 | Science, Math |
| 3 | English, History, Science |

This table is not in 1NF because the "Courses" column contains multiple values. To bring it into 1NF, we would create a separate table for courses:

**Table: Students**

| **StudentID** |
| --- |
| 1 |
| 2 |
| 3 |

**Table: Courses**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Math |
| 1 | English |
| 2 | Science |
| 2 | Math |
| 3 | English |
| 3 | History |
| 3 | Science |
|  |  |

**2NF (Second Normal Form):**

**Definition:**   
Second Normal Form (2NF) in simple terms all non primary key attributes are dependent on the entire primary key.

**Example:** Consider a table named **Orders**:

| **OrderID** | **Product** | **Category** | **Price** |
| --- | --- | --- | --- |
| 1 | Laptop | Electronics | 800 |
| 2 | Tablet | Electronics | 300 |
| 3 | Book | Books | 20 |

The "Category" column is partially dependent on the primary key (OrderID, Product). To bring it into 2NF, we separate it into a new table:

**Table: Orders**

| **OrderID** | **Product** | **Price** |
| --- | --- | --- |
| 1 | Laptop | 800 |
| 2 | Tablet | 300 |
| 3 | Book | 20 |

**Table: ProductCategories**

| **Product** | **Category** |
| --- | --- |
| Laptop | Electronics |
| Tablet | Electronics |
| Book | Books |

**3NF (Third Normal Form):**

**Definition:** Third Normal Form (3NF) in simple terms means that in addition to meeting the requirements of First Normal Form (1NF) and Second Normal Form (2NF), there should be no transitive dependencies.

**Example:** Consider a table named **Employees**:

| **EmployeeID** | **Department** | **Location** | **ManagerID** |
| --- | --- | --- | --- |
| 1 | IT | Building A | 3 |
| 2 | HR | Building B | 3 |
| 3 | Operations | Building A | NULL |

In this case, "Location" depends on "Department" (transitive dependency). To bring it into 3NF, we create a new table:

**Table: Employees**

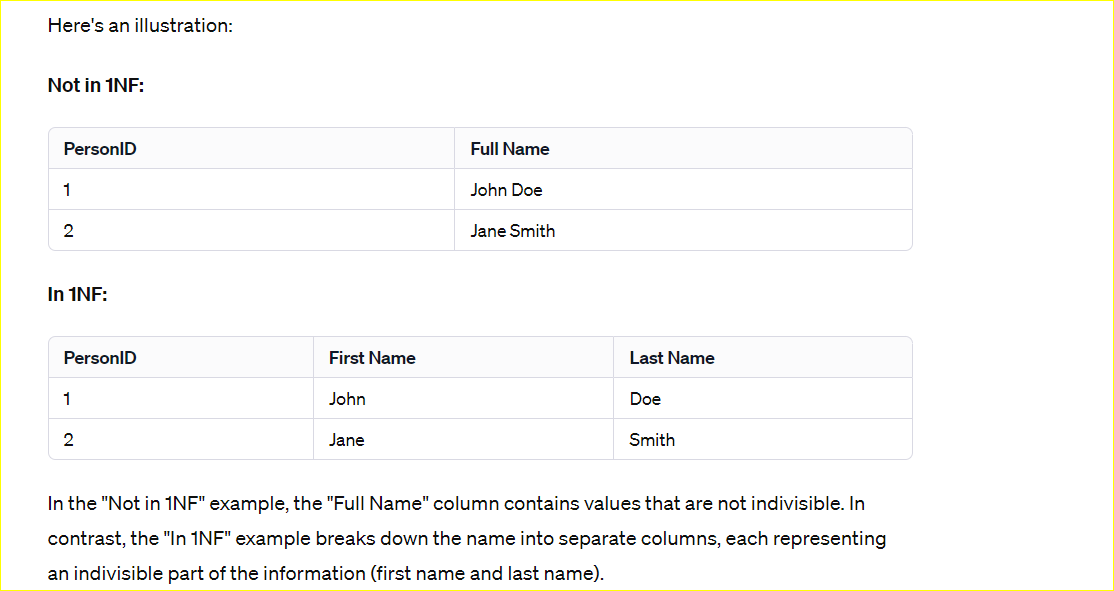
| **EmployeeID** | **Department** | **ManagerID** |
| --- | --- | --- |
| 1 | IT | 3 |
| 2 | HR | 3 |
| 3 | Operations | NULL |

**Table: Departments**

| **Department** | **Location** |
| --- | --- |
| IT | Building A |
| HR | Building B |
| Operations | Building A |

In this example, we removed the transitive dependency by creating a separate table for departments.

**ILLUSTRATION OF 1NF**



**ILLUSTRATION OF 3NF**

Certainly! Let's consider a hypothetical scenario of tracking orders and products:

**Example: Order Information**

Consider a denormalized table named **Orders**:

| **OrderID** | **Product** | **Category** | **Price** |
| --- | --- | --- | --- |
| 1 | Laptop | Electronics | 800 |
| 2 | Tablet | Electronics | 300 |
| 3 | Book | Books | 20 |

This table is in 1NF and 2NF because each column holds atomic values, and non-prime attributes (attributes not part of the primary key) are fully functionally dependent on the primary key.

Now, let's analyze it for 3NF:

* **Primary Key:** OrderID
* **Non-Prime Attributes:** Product, Category
* **Transitive Dependency:** Category depends on Product.

To achieve 3NF, we create a new table for Products:

**Table: Orders**

| **OrderID** | **Product** | **Price** |
| --- | --- | --- |
| 1 | Laptop | 800 |
| 2 | Tablet | 300 |
| 3 | Book | 20 |

**Table: ProductCategories**

| **Product** | **Category** |
| --- | --- |
| Laptop | Electronics |
| Tablet | Electronics |
| Book | Books |

Now, the "Category" information is no longer transitively dependent on the primary key (OrderID), and the tables are in 3NF. Each table represents a distinct concept (order information and product details), and relationships between them are maintained through the common attribute (Product).

Top of Form

## **CONVERTING A TABLE FROM 1NF TO 3NF**

# Let's revisit the example to ensure it is in 1NF before moving on to 2NF.

**Example: Student Course Enrollment**

Consider a table named **StudentCourses**:

| **StudentID** | **Courses** |
| --- | --- |
| 1 | Math, English |
| 2 | Science, Math |
| 3 | English, History, Science |

This table is not in 1NF because the "Courses" column contains multiple values in some cells.

To bring it into 1NF, we create a separate row for each course:

**Table: StudentCourses**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Math |
| 1 | English |
| 2 | Science |
| 2 | Math |
| 3 | English |
| 3 | History |
| 3 | Science |

Now, each cell contains a single, indivisible value, and the table is in 1NF.

User

# Certainly! Let's continue with the 2NF conversion for the corrected 1NF table:

**Table: StudentCourses**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Math |
| 1 | English |
| 2 | Science |
| 2 | Math |
| 3 | English |
| 3 | History |
| 3 | Science |

Now, we need to identify the primary key. In this case, it seems that a combination of StudentID and Course can serve as a composite primary key.

* **Primary Key:** (StudentID, Course)

The issue with the current 2NF table is that there is no additional information about the courses, such as the instructor or schedule. To address this, we can create a new table for CourseDetails:

**Table: StudentCourses**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Math |
| 1 | English |
| 2 | Science |
| 2 | Math |
| 3 | English |
| 3 | History |
| 3 | Science |

**Table: CourseDetails**

| **Course** | **Instructor** | **Schedule** |
| --- | --- | --- |
| Math | Prof. X | MWF 10:00 AM |
| English | Prof. Y | TTh 1:30 PM |
| Science | Prof. Z | MWF 2:45 PM |
| History | Prof. W | TTh 9:00 AM |

Now, the information about each course is stored in a separate table, and the tables are in 2NF. The relationship between them is maintained through the common attribute (Course).

User

**ChatGPT**

Certainly! Let's evaluate the 2NF tables and make sure they satisfy the requirements for Third Normal Form (3NF). In 3NF, we want to eliminate transitive dependencies.

**Table: StudentCourses**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Math |
| 1 | English |
| 2 | Science |
| 2 | Math |
| 3 | English |
| 3 | History |
| 3 | Science |

**Table: CourseDetails**

| **Course** | **Instructor** | **Schedule** |
| --- | --- | --- |
| Math | Prof. X | MWF 10:00 AM |
| English | Prof. Y | TTh 1:30 PM |
| Science | Prof. Z | MWF 2:45 PM |
| History | Prof. W | TTh 9:00 AM |

In this scenario, we can observe that the "Instructor" and "Schedule" attributes in the **CourseDetails** table depend only on the "Course" column and not on the entire primary key (StudentID, Course) of the **StudentCourses** table.

To achieve 3NF, we need to create a new table for CourseInstructors:

**Table: StudentCourses**

| **StudentID** | **Course** |
| --- | --- |
| 1 | Math |
| 1 | English |
| 2 | Science |
| 2 | Math |
| 3 | English |
| 3 | History |
| 3 | Science |

**Table: CourseDetails**

| **Course** | **Schedule** |
| --- | --- |
| Math | MWF 10:00 AM |
| English | TTh 1:30 PM |
| Science | MWF 2:45 PM |
| History | TTh 9:00 AM |

**Table: CourseInstructors**

| **Course** | **Instructor** |
| --- | --- |
| Math | Prof. X |
| English | Prof. Y |
| Science | Prof. Z |
| History | Prof. W |

Now, each table is in 3NF. The information about each course, its schedule, and instructor is stored in separate tables, and relationships between them are maintained through the common attribute (Course).

Top of Form

 As a result, 2nf is a table with no partial dependencies, whereas 3nf is a table with no transitive dependencies.