Normalization:

We need to perform Normalization on our database to reduce Data Redundancy.
It minimizes redundancy using certain rules or sets of rules.

There are many types of normal forms, although we are going to focus on 1Nf, 2NF, 3NF and BCNF (also known as 3.5 NF).

Types of Normal Forms:

1. **First Normal Form (1NF):** This is the Step 1 of the Normalisation Process.
   For a Relation/table to justify 1NF it needs to satisfy 4 basic conditions:
   - Each attribute should contain atomic values. (i.e. No multivalued attributes)
   - Each Value stored in an attribute should be of the same type.
   - All the attributes in a table should have unique names.
   - The order of the data stored in the table doesn't matter.

   Let's take a look at an Example:

Chrac_id	Chrac_name	Actor
54342	Iron Man	Robert Downey Jr
33243	Spiderman	Tobey Margiue, Andrew Garfield, Tom Holland
24352	Loki	Tom Hiddleston
74564	Thor	Chris Hemsworth

Above is the table of Characters with the Actors name who played that certain character over the years for Marvel Cinematic Universe.

Now on observing the data above, if we apply the rules of normalisation known to us and check whether this data is in first normal form or not.

We observe that 3 out of 4 rules are followed, that is,

- Each attribute has a unique name,
- Each attribute contains data of the same type and

- the order of the data is the form we want it to be.

But the value of data in each attribute isn't atomic, for example, the 2nd instance of the above relation has multiple values in the Actor attribute as Spiderman was played by three different actors over the years.
This is a violation of 1NF.

Now, how to solve this problem, to bring the table to satisfy all conditions/rules of the first normal form we need to break the multiple values into atomic values.
Therefore, now the above table will look like:

Chrac_id	Chrac_name	Actor
54342	Iron Man	Robert Downey Jr
33243	Spiderman	Tobey Margiue
33243	Spiderman	Andrew Garfield
33243	Spiderman	Tom Holland
24352	Loki	Tom Hiddleston
74564	Thor	Chris Hemsworth

Now the above table is said to be in 1NF.

2. **Second Normal Form:** For a Relation/table to justify 2NF it needs to satisfy 2 rules:
    - It should be in First Normal Form.
    - It should not have any partial dependencies i.e. when a nonprime attribute is derivable from only a part of a candidate key.

Let's take a look at an Example:

charac_id	movie_name	Charac_rating	Director
54342	Iron Man 2	9	Jon Favreau
33243	Spiderman	10	Sam Raimi
33243	The Amazing Spider Man	7	Marc Webb

53463	Iron Man 2	9	Jon Favreau
74564	Thor: Ragnarok	9	Taika Waititi
54342	Iron Man 3	9	Shane Black

Above table lists famous movies with their main characters and the actors it was played by along with the movie director.

Now for the above Relation to be in 2NF, it should be in 1NF, which it is already in. Secondly, there should be no partial dependencies.

Although, here we can observe that, (charac_id + movie_name) acts as the primary key (which is a chosen candidate key).

Although, in the above relation from movie_name attribute Director attribute can be derived and has nothing to do with charc_id attribute.
Now an attribute is only dependent on part of the primary key but not on the whole set, hence it's called Partial Dependencies.

How to fix this?
One of the simplest solutions is to remove the Director Attribute from the table.

The Table becomes:

charac_id movie_nameCharac_rating

54342	Iron Man 2	9
33243	Spiderman	10
33243	The Amazing Spider Man	7
53463	Iron Man 2	9
74564	Thor: Ragnarok	9
54342	Iron Man 3	9

And now it is in Second normal form.

And the attribute Director gets adjusted to other relation in the database as below:

movie_name	Director
Iron Man 2	Jon Favreau
Spiderman	Sam Raimi
The Amazing Spider Man	Marc Webb
Thor: Ragnarok	Taika Waititi
Iron Man 3	Shane Black