- **Lossless Join**:

A decomposition of the Relation is said to be lossless when it is decomposed into two new relations whose FD satisfies the following condition:

1. Union of attributes of two relations is equal to that of the original relation.
2. Intersection of Attributes of two decomposed relations must not be NULL.
3. Common attributes between the relations must be a key for at least one of them.

- **Dependency Preserving:**

Dependencies in DBMS are a relation between two or more attributes. On decomposing a relation into two, all dependencies of the original relation either must be a part of both the decomposed relation or must be derivable from the combination of dependencies of both the relation.

Let's continue with the types of Normalisation:

3. <u>Third Normal Form (3 NF):</u> For a table to be in the third normal form:

- It should already be in the 2NF.
- It should not have Transitive Dependency.

Let's take a look at an Example:

std_id	sub_id	marks	exam_name	total_marks
3160	9	85	Maths - II	100
7067	10	98	Ortho	120
3160	7	70	CS271	90
5276	9	78	Maths - II	100
3523	4	66	CE201	100

Above is the Score table, with attributes stating Student marks in the examination. The primary key for Score table is a composite key, i.e. { std_id + sub_id }
Let's analyse the dependencies of attributes on the primary key,

Attribute marks are dependent on both student and subject as it tells us, whose (student) marks are these and in which subject these marks are scored.

Marks can be determined using a combination of std_id and sub_id, because if we are asked to get the marks of a student with id 3160, we won’t be able to fetch anything as we don’t know for which subject. Similarly, when asked about marks for subjects with ID 9, we won’t get anything as we don’t know for which student.

So we can say that marks are dependent on both student_id and subject_id.

Attribute exam_name depends on both student and subject.
For illustration, a Chemical engineering student (std_id=3523), will have a Lab exam but all the engineering mathematics students won’t have it. Also, some subjects might have practical exams and some might not.
So we notice that exam_name is dependent on both student_id and subject_id.

Now last but not the least attribute total_marks, this one depends on exam_name because see different exam types different weightage.
For example, different markings for practical and theoretical exams.
But, exam_name is not a primary key or a part of the primary key and total_marks depends on it.
Hence, exam_name⟶ Total_marks (i.e. nonprime attribute ⟶ nonprime attribute), and { std_id + sub_id } ⟶ exam_name.
Therefore, by working out the transitive rule of FD's, { std_id + sub_id } ⟶ Total_marks.
This is Known as the Transitive dependency, Which is a violation of Third Normal form.

How to fix this ?
Create another table named Exams with attributes exam_name and total_marks along with exam_id use it when required for reference.

Table Score:

std_id	sub_id	marks	exam_id
3160	9	85	3
7067	10	98	1
3160	7	70	2
5276	9	78	3
3523	4	66	3

It's in Third Normal Form.

Table Exams:

exam_id	exam_name	total
1	Practicals	120
2	Labs	90

3	Theoreticals	100

4. <u>Boyce-Codd Normal Form (BCNF):</u> It is an extension of 3NF and is also known as 3.5 Normal Form.
For a table to be in the Boyce-Codd Normal Form, it should satisfy 2 rules:

- It should be in the Third Normal Form.
- A prime attribute shouldn't be dependent on a non-prime attribute. (i.e. if M⟶ N, then M is a super key)

Let's take a look at an Example:

ninja_id	Actor	Movie
1	Robert Downey Jr	Iron Man 2
5	Chris Evans	Avengers: Endgame
5	Elizabeth Oleson	Avengers: Infinity War
7	Chris Evans	Captain America: Civil War
4	Chris Hemsworth	Thor: Ragnarok

Now the above table is formulated by surveying every ninja's favorite Marvel actor along with the movie in which they liked that actor the most.

We can observe that a single ninja might have multiple favorite Marvel Actors and there is also a case present that certain ninjas like the same Actor but in different movies like ninja 5 likes Chris Evans in movie Avengers: Endgame, whereas ninja 7 likes Chris Evans in the movie Captain America: Civil War.

In This table ninja_id and Actor combined can be used as Primary key to determine all the other table attributes.

Now if we only talk about this table, we can also observe that the ‘Actor’ attribute is being determined from the ‘Movie’ attribute.
So the FD’s will be like,
{ninja_id, Actor} ⟶ Movie
Movie ⟶ Actor

But Actor is a prime attribute, as it is part of the candidate key, whereas movie is a non-prime attribute.
Hence, in this table above we have a prime attribute being dependent on a non-prime attribute. Hence the table isn’t satisfying BCNF.

How to fix this?

To make the table satisfy the BCNF conditions we need to split the above table into two.
One table containing the ninja_id and the movie_id.
Another table containing the movie_id, the movie and Actor.

Note: The above table satisfied the Third normal form, then only we evaluated for the second rule of BCNF.