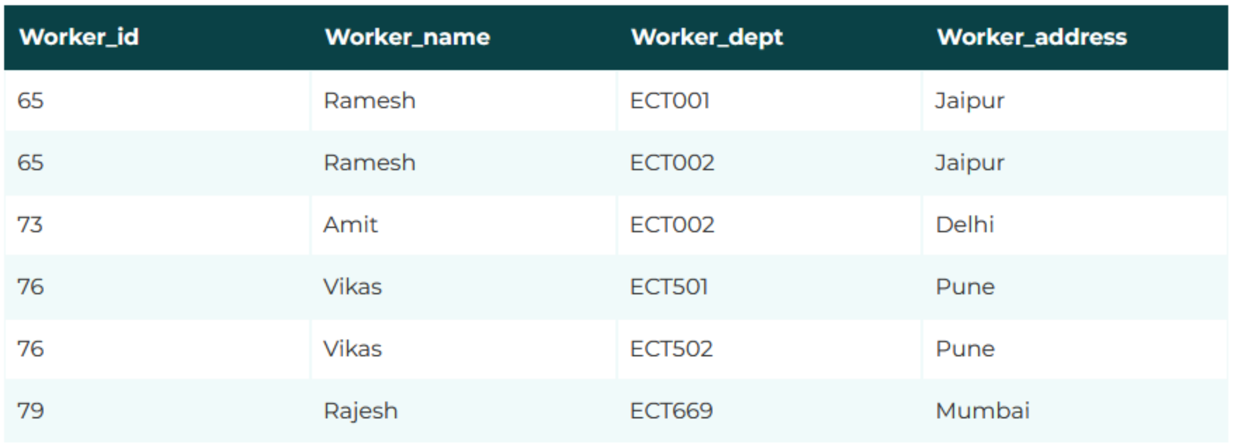
Anomalies:

1. Insertion Anomaly
2. Deletion Anomaly
3. Modification/Update Anomaly

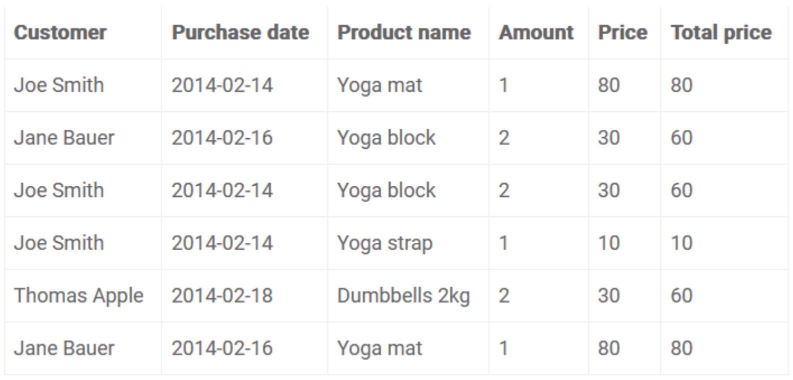


Worker_id	Worker_name	Worker_dept	Worker_address
65	Ramesh	ECT001	Jaipur
65	Ramesh	ECT002	Jaipur
73	Amit	ECT002	Delhi
76	Vikas	ECT501	Pune
76	Vikas	ECT502	Pune
79	Rajesh	ECT669	Mumbai

Insertion Anomaly: Let's say this table is created in such a way, it can not take NULL values. Now, we want to insert a new worker "80", who is not assigned to any "Worker_dept" yet. So, unless we assign a dept. For him, we can not insert his/her value in the table as we are not allowed to put NULL. This is an insertion anomaly.

Deletion Anomaly: Let's say we want to delete a value "Worker_id" = 76. There are no other tables, just this one table. As a result, "Worker_dept" = ECT501 and its address "Pune" will be lost. Because, we did not save the data separately in another table. This unintentional data loss is a deletion anomaly.

Update Anomaly:



Customer	Purchase date	Product name	Amount	Price	Total price
Joe Smith	2014-02-14	Yoga mat	1	80	80
Jane Bauer	2014-02-16	Yoga block	2	30	60
Joe Smith	2014-02-14	Yoga block	2	30	60
Joe Smith	2014-02-14	Yoga strap	1	10	10
Thomas Apple	2014-02-18	Dumbbells 2kg	2	30	60
Jane Bauer	2014-02-16	Yoga mat	1	80	80

Here, if we want to change the name of the product; for example: ”Yoga Mat”, we have to change in multiple places or rows. There might be scenario, it was not changed in several places. This scenario is called Update Anomaly.

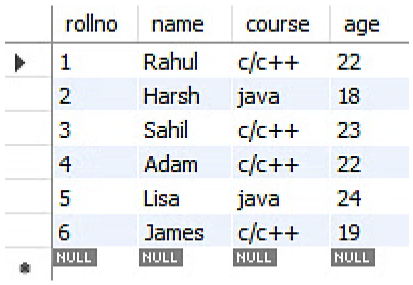
1st Normalization:

A table is said to be in 1NF if it meets the following criteria:

- A single cell must not hold more than one value (atomicity)
- There must be a primary key for identification
- No duplicated rows or columns
- Each column must have only one value for each row in the table

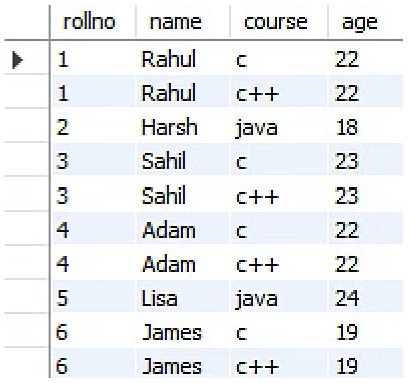
Example:

Here, in the following table, the “course” column has two values. Thus it does not follow the First Normal Form (atomicity).



rollno	name	course	age
1	Rahul	c/c++	22
2	Harsh	java	18
3	Sahil	c/c++	23
4	Adam	c/c++	22
5	Lisa	java	24
6	James	c/c++	19
NULL	NULL	NULL	NULL

After applying 1NF:



rollno	name	course	age
1	Rahul	c	22
1	Rahul	c++	22
2	Harsh	java	18
3	Sahil	c	23
3	Sahil	c++	23
4	Adam	c	22
4	Adam	c++	22
5	Lisa	java	24
6	James	c	19
6	James	c++	19

2nd Normalization:

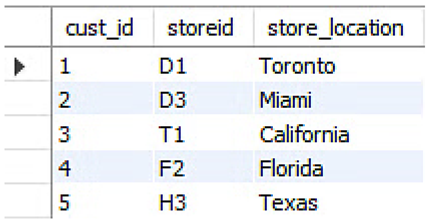
A table is said to be in 2NF if it meets the following criteria:

- It's already in 1NF
- It has no partial dependency. That is, all non-key attributes are fully dependent on a primary key.

Partial dependency: Partial dependency is a situation in which a non-key attribute of a table depends on only a part of the primary key meaning the primary key is a composite key.

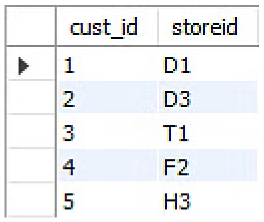
Example:

The cust_id+storeid = primary key; meaning a composite key. Here, store location depends on storeid only. It does not depend on cust_id, which is also a part of the primary key. So, this scenario is partial dependency.

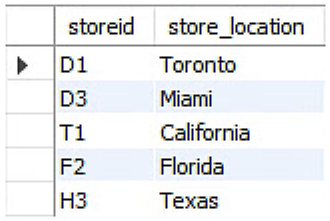


cust_id	storeid	store_location
1	D1	Toronto
2	D3	Miami
3	T1	California
4	F2	Florida
5	H3	Texas

After applying 2NF:



cust_id	storeid
1	D1
2	D3
3	T1
4	F2
5	H3



storeid	store_location
D1	Toronto
D3	Miami
T1	California
F2	Florida
H3	Texas

To remove this partial dependency in this scenario, what we can do is split the table into two parts. The new table will contain the store_id (part of the primary key on which the attribute was dependent) and store_location (the attribute which was dependent on part of the primary key which is store_id).

3rd Normalization:

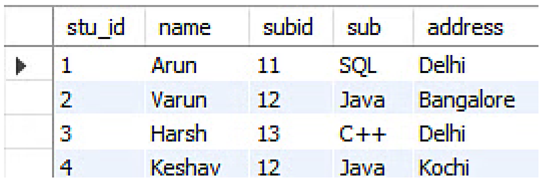
A table is said to be in 3NF if it meets the following criteria:

- It needs to be in 2NF
- It has no transitive partial dependency.

Transitive Dependency: If the value of a non-primary attribute can be defined using another non-primary attribute then it is called a transitive dependency.

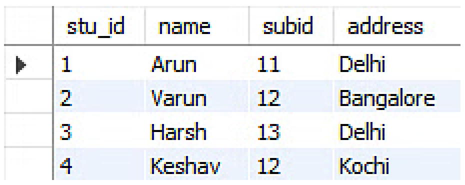
Example:

Below, the student table has “stu_id” as the primary key. We can get “subid” from “stu_id” and “sub” from “sub_id”. So, “stu_id” -> “subid” and, “subid” -> “sub”. Meaning, “stu_id” -> “sub”. This scenario is transitive dependency. Because, “subid” is not a key and still “sub” depends on “sub_id”.

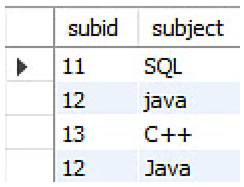


stu_id	name	subid	sub	address
1	Arun	11	SQL	Delhi
2	Varun	12	Java	Bangalore
3	Harsh	13	C++	Delhi
4	Keshav	12	Java	Kochi

After applying 3NF:



stu_id	name	subid	address
1	Arun	11	Delhi
2	Varun	12	Bangalore
3	Harsh	13	Delhi
4	Keshav	12	Kochi



subid	subject
11	SQL
12	java
13	C++
12	Java

Now, in order to solve the transitive dependency, we can divide the table and make the “sub_id” **(which is not the primary key in prev. table)** primary key attribute in the new table and add the dependent attributes.