Please answer the following normalization questions:

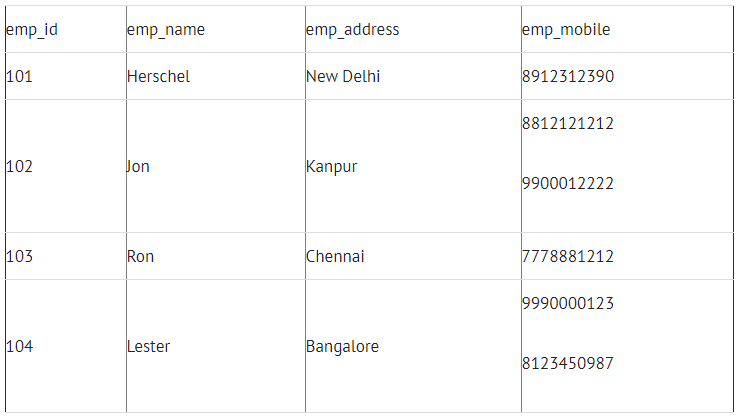
* Compare top-down and bottom-up approaches for database design. Which approach is better? Why?
* Any experience in distinguishing a bad or a good relation schema. If yes, what criteria or measures are used?
* Any experience in functional dependencies learned from the past in school or at work.
* Conduct a research and then provide 1NF, 2NF, 3NF, BCNF examples to share with classmates.
* You may also post any special functional dependencies or normalization (such as 1NF, 2NF, 3NF and BCNF) related questions or comments that you have after viewing the lectures.

Top-down methodology is generally more effective than botton-up methodology, since it captures the entity types and their associated attributes, as well as their relationships based on the requirements. Doing a bottom-up approach for database design can result in repeated attributes as well as making relationships more difficult to define.

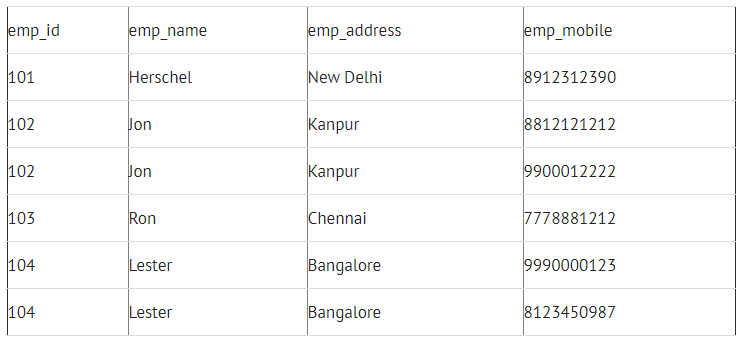
As I am a novice in database design, I do not have any experience in functional dependencies or relationship schemas from earlier schooling or work.

I was able to find some examples of 1NF, 2NF, 3NF online that I can share with the class. For reference, the entire article that I read was here: <https://beginnersbook.com/2015/05/normalization-in-dbms/>

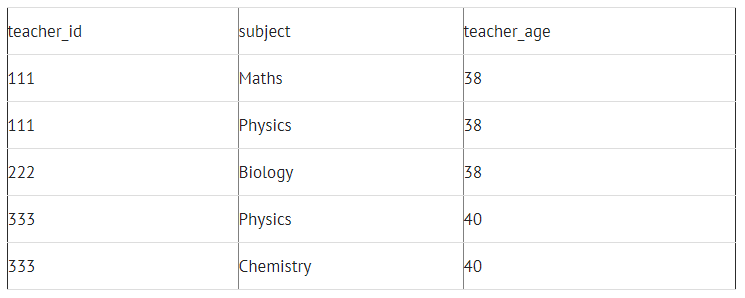
For 1NF, suppose a company wants to store employee contact information. One example of this is in the below table:



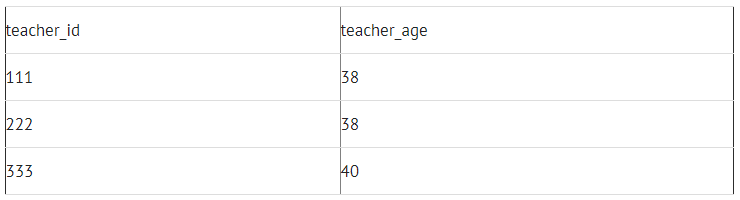
However, we note that some of the tuples have multiple values for emp\_mobile. This can be handled in the 1NF stage, and a resulting database design would look like:

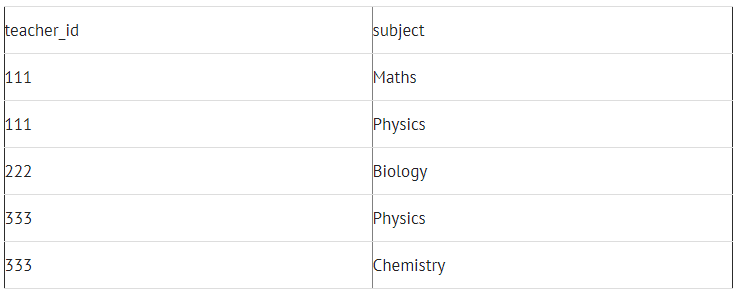


For 2NF, suppose a school wants to store information about their teachers. One example of a table to do this would be like this:



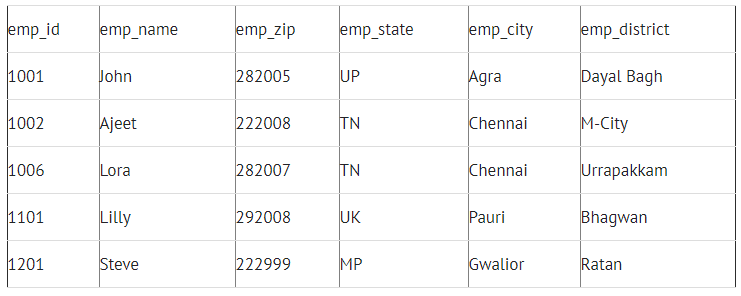
However, the teacher age does not depend on the subject taught. Thus, we can break this down into two different relations:



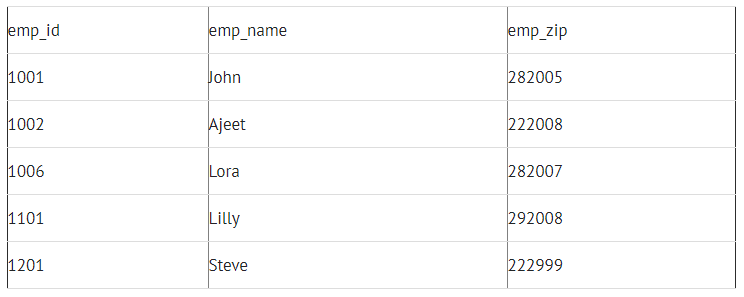


This eliminates some redundancy in the non-2NF table.

Finally, for 3NF, we note that some of the values in the table may not depend on the primary key for that entry. One example of this is in the following table which stores employee address information:

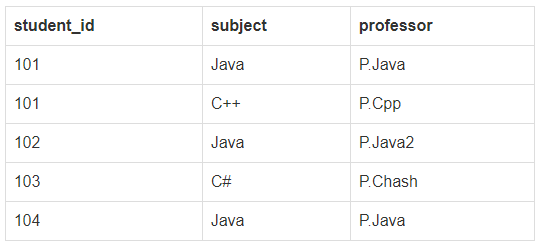


However, we note that the emp\_state, emp\_city, and emp\_district all depend on the emp\_zip. Thus, we can split this into two tables to reduce data redundancy. See the two tables below for how to do this:





BCNF is even stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y,X should be the super key of the table. One example of this would be the following table:



This table is not in BCNF since each professor is dependent on each subject. Thus we can break out the previous table in to the following two tables:





Thus these are examples of 1NF, 2NF, 3NF and BCNF.