# Database Management System
# “Normalization”

Submitted by :

Richa Shukla

# Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The Normal form is used to reduce redundancy from the database table.

# Types of Normal form

- 1NF : A relation is in 1NF if it contains an atomic value.
- 2NF: A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
- 3NF: A relation will be in 3NF if it is in 2NF and no transition dependency exists.

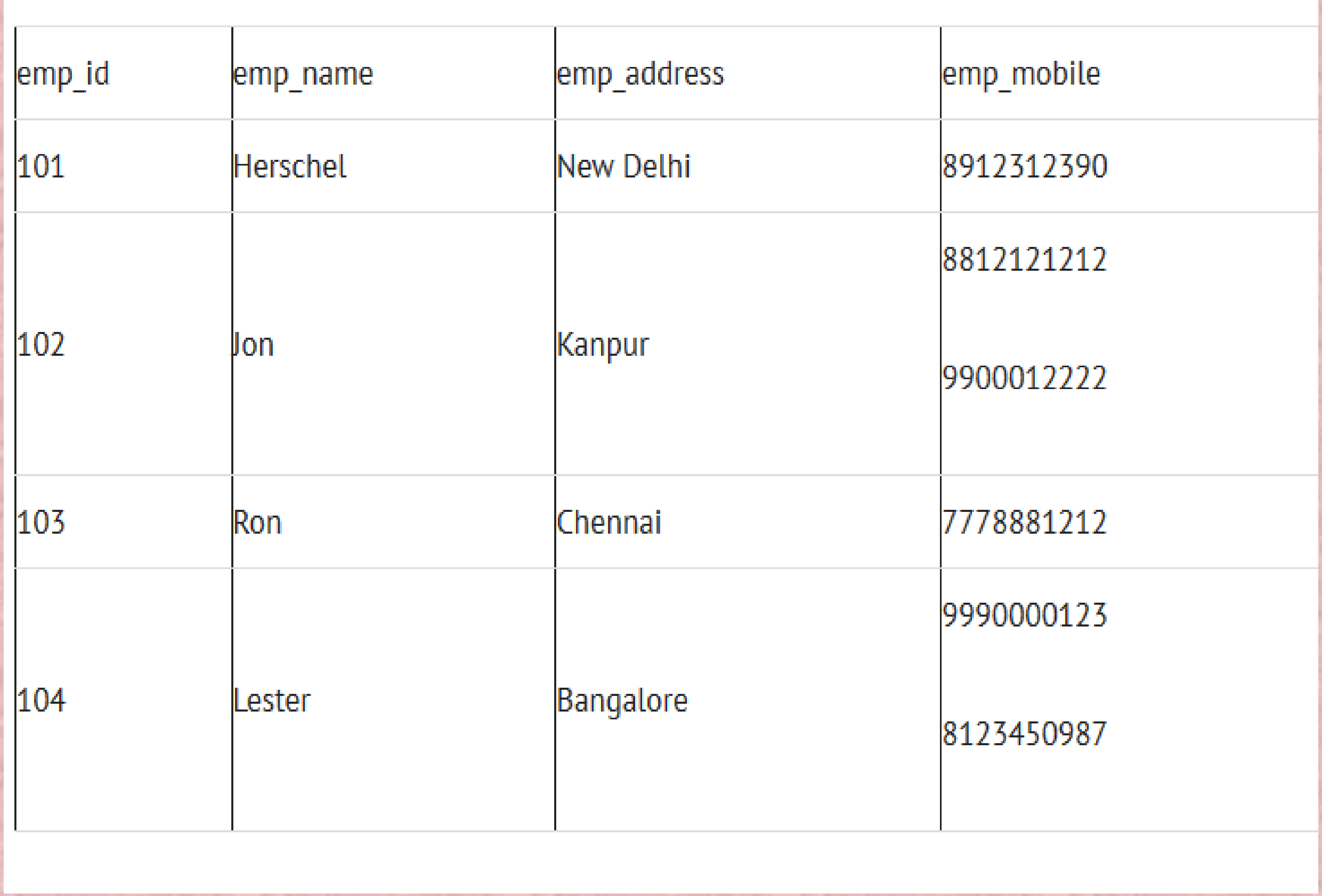
- BCNF : A relation will be in 3NF and for each functional dependency ( X → Y ), X should be a super Key.

- 4NF: A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.

- 5NF: A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless

# First Normal Form (1NF)

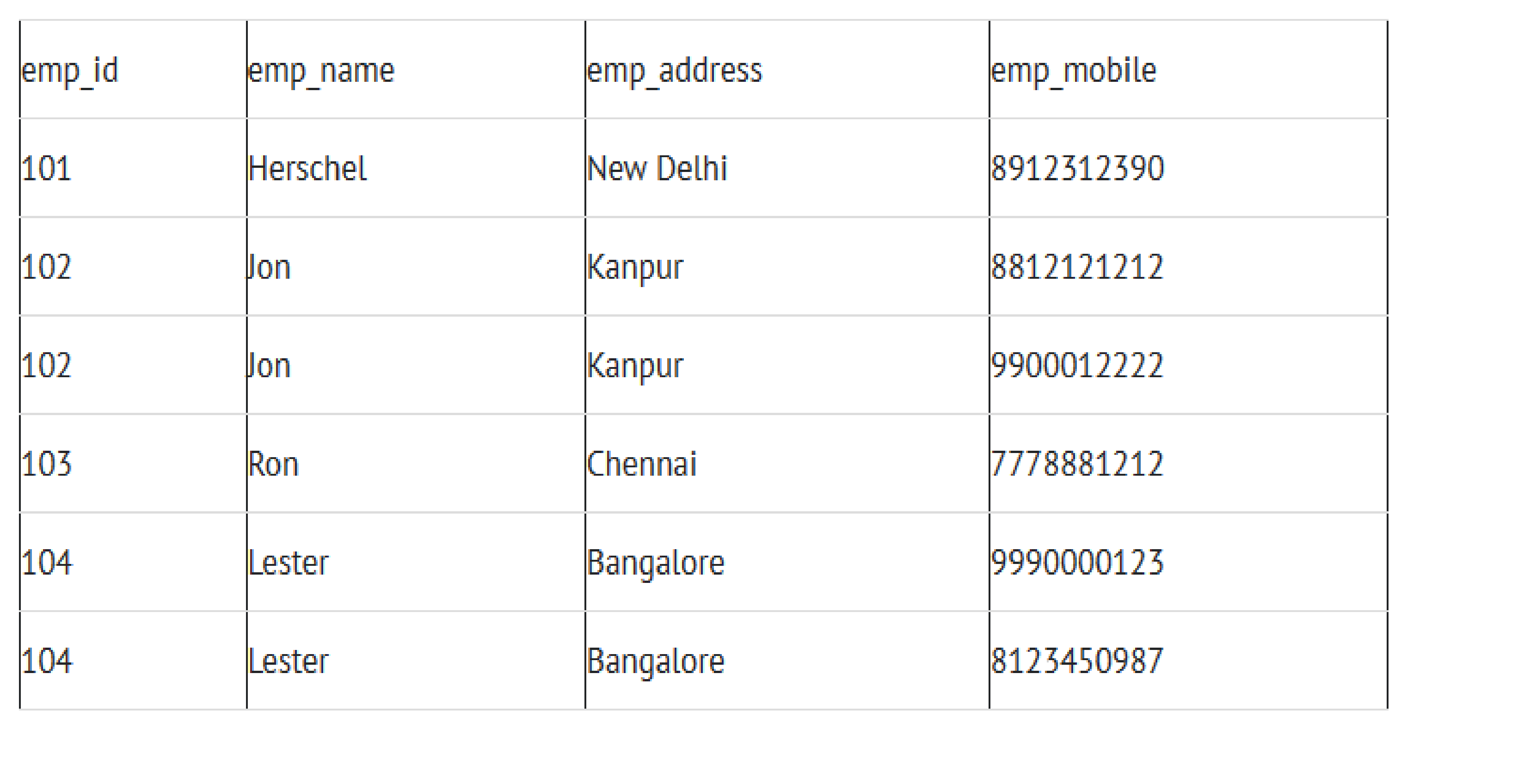
- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.



| emp_id | emp_name | emp_address | emp_mobile |
|---|---|---|---|
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212<br>9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123<br>8123450987 |

- Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.
- This table is **not in 1NF** as the rule says “each attribute of a table must have atomic (single) values”,

# To make the table complies with 1NF we should have the data like this:



| emp_id | emp_name | emp_address | emp_mobile |
|---|---|---|---|
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212 |
| 102 | Jon | Kanpur | 9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123 |
| 104 | Lester | Bangalore | 8123450987 |

# Second normal form (2NF)

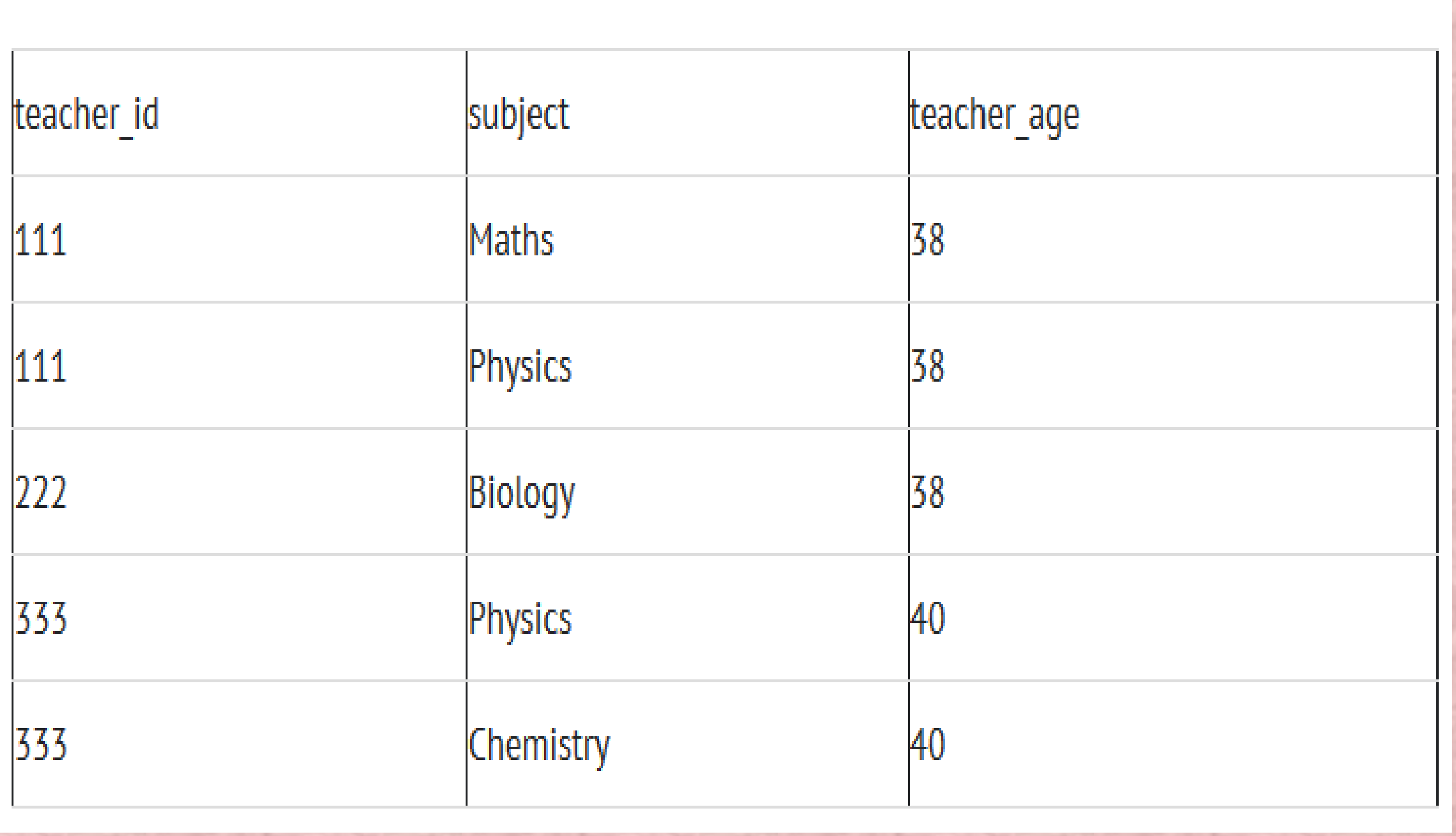
- A table is said to be in 2NF if both the following conditions hold:

❖ Table is in 1NF (First normal form)

❖ No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

# A teacher can teach more than one subjects, the table can have multiple rows for a same teacher.



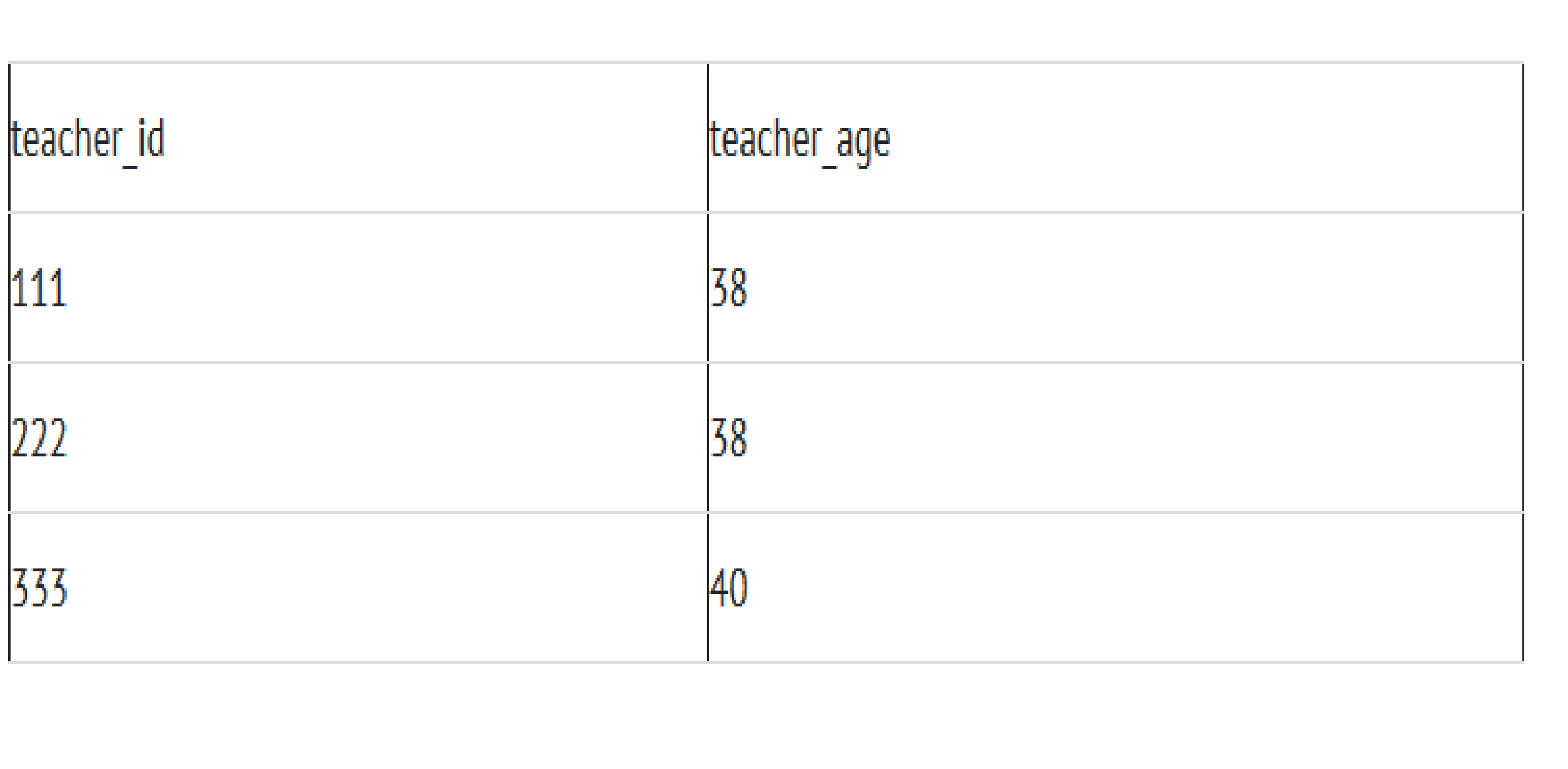
| teacher_id | subject | teacher_age |
|---|---|---|
| 111 | Maths | 38 |
| 111 | Physics | 38 |
| 222 | Biology | 38 |
| 333 | Physics | 40 |
| 333 | Chemistry | 40 |

- **Candidate Keys**: {teacher_id, subject}
  **Non prime attribute**: teacher_age

- The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key.
- This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

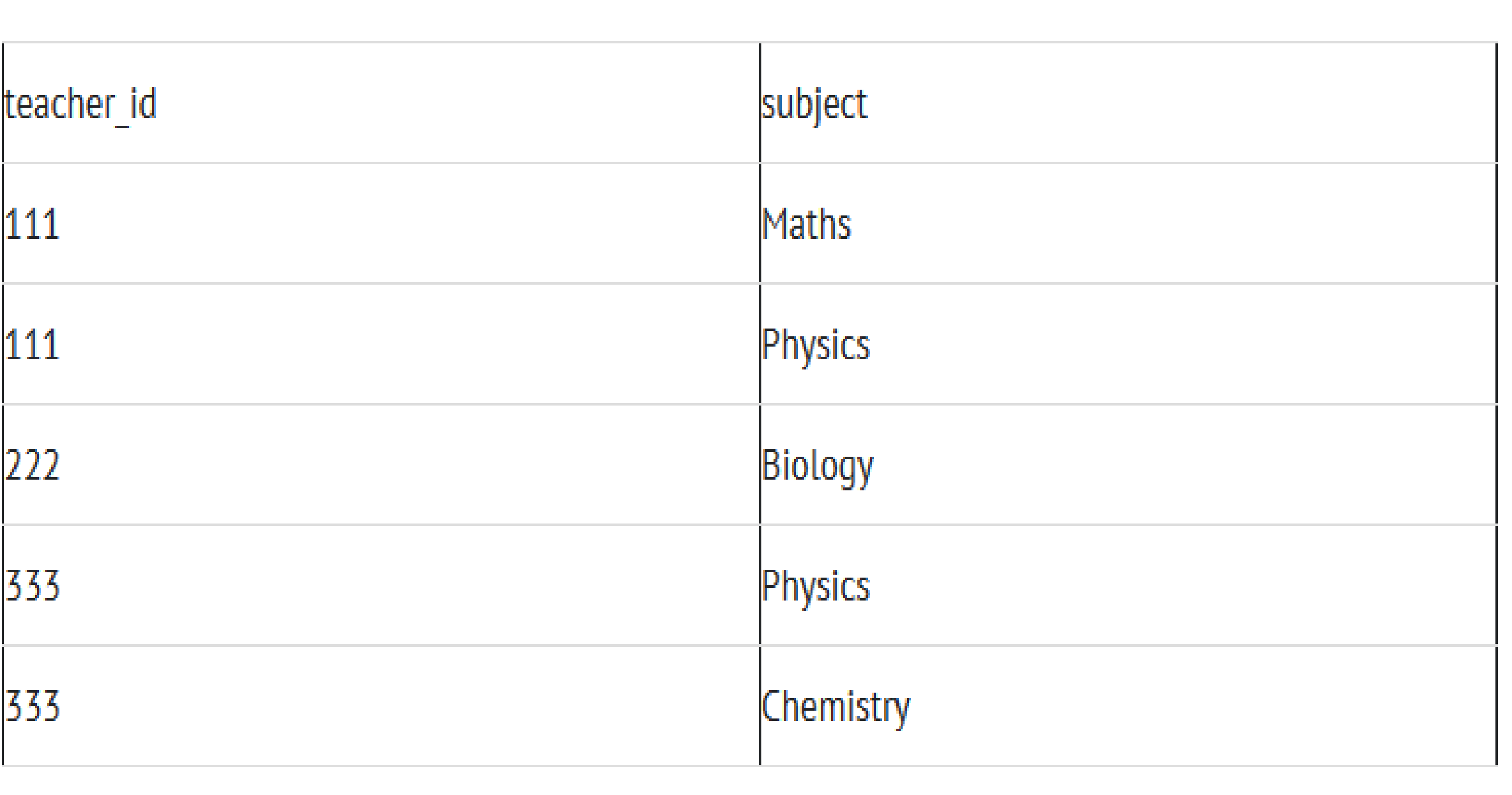
To make the table complies with 2NF we can break it in two tables.

## Teacher age details table:



| teacher_id | teacher_age |
|---|---|
| 111 | 38 |
| 222 | 38 |
| 333 | 40 |

# Teacher subject table:



| teacher_id | subject |
|---|---|
| 111 | Maths |
| 111 | Physics |
| 222 | Biology |
| 333 | Physics |
| 333 | Chemistry |

# Third Normal form (3NF)

- A table design is said to be in 3NF if both the following conditions hold:

❖ Table must be in 2NF

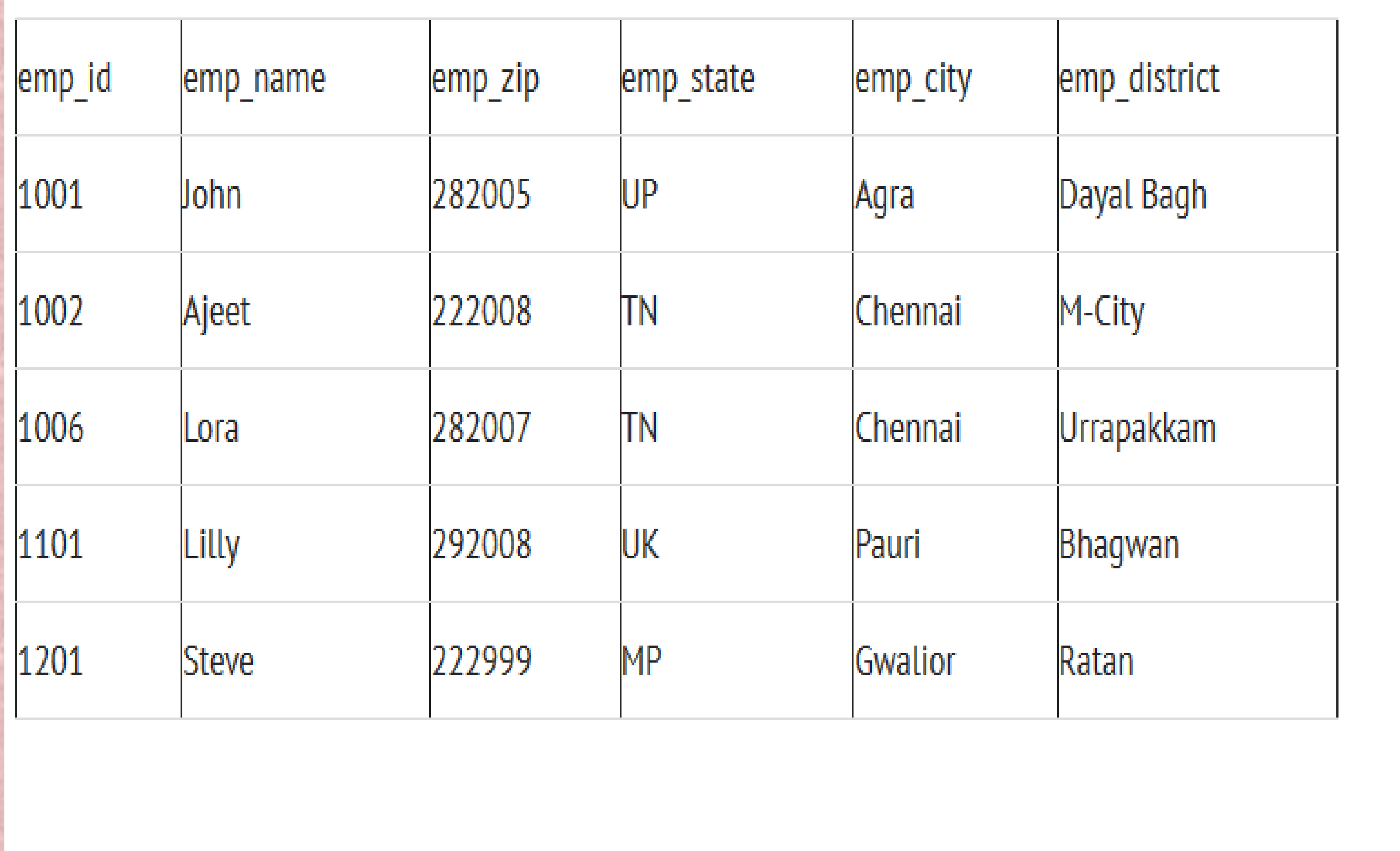
❖ Transitive functional dependency of non-prime attribute on any super key should be removed.

A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

# A company wants to store the complete address of each employee.

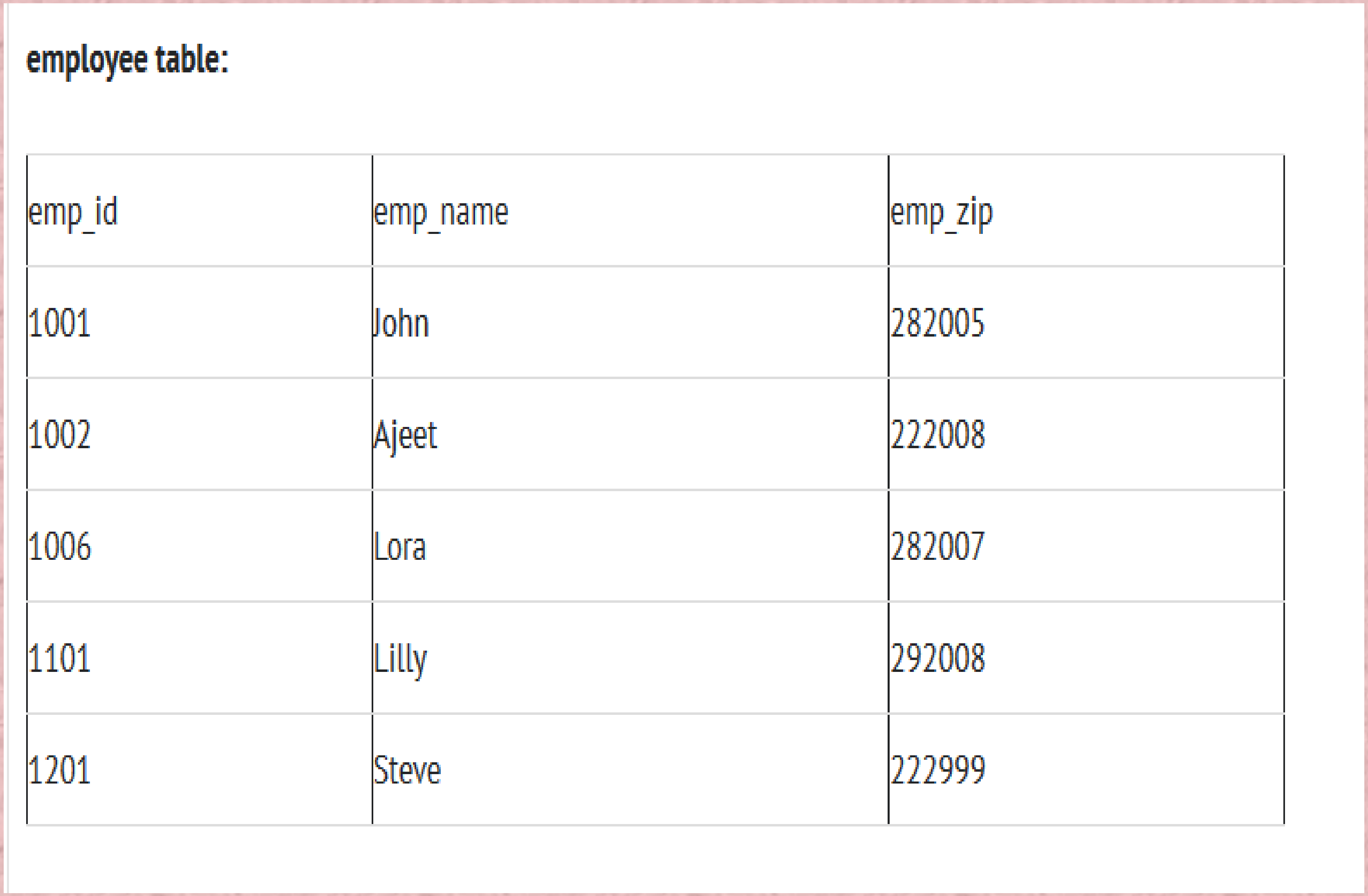


| emp_id | emp_name | emp_zip | emp_state | emp_city | emp_district |
|---|---|---|---|---|---|
| 1001 | John | 282005 | UP | Agra | Dayal Bagh |
| 1002 | Ajeet | 222008 | TN | Chennai | M-City |
| 1006 | Lora | 282007 | TN | Chennai | Urrapakkam |
| 1101 | Lilly | 292008 | UK | Pauri | Bhagwan |
| 1201 | Steve | 222999 | MP | Gwalior | Ratan |

- **Super keys**: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
  **Candidate Keys**: {emp_id}

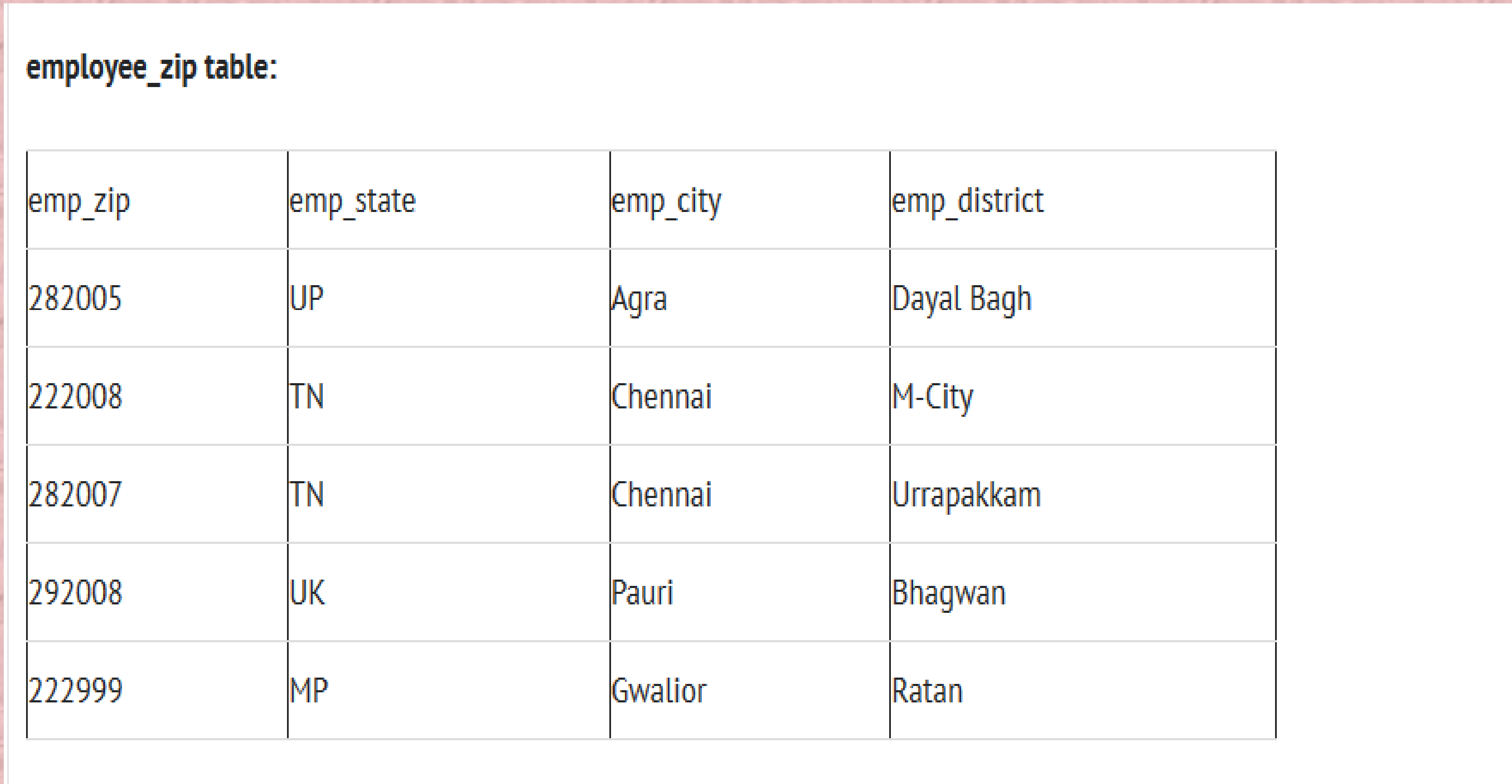
  **Non-prime attributes**: all attributes except emp_id are non-prime as they are not part of any candidate keys.

- Here, emp_state, emp_city & emp_district dependent on emp_zip.
- emp_zip is dependent on emp_id.
- That makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id).
- This violates the rule of 3NF.



**employee table:**

| emp_id | emp_name | emp_zip |
|---|---|---|
| 1001 | John | 282005 |
| 1002 | Ajeet | 222008 |
| 1006 | Lora | 282007 |
| 1101 | Lilly | 292008 |
| 1201 | Steve | 222999 |



**employee_zip table:**

| emp_zip | emp_state | emp_city | emp_district |
|---|---|---|---|
| 282005 | UP | Agra | Dayal Bagh |
| 222008 | TN | Chennai | M-City |
| 282007 | TN | Chennai | Urrapakkam |
| 292008 | UK | Pauri | Bhagwan |
| 222999 | MP | Gwalior | Ratan |

3NF is achieved, considered as the database is normalized.

Thank you