Design a table for an Employee Management System that includes EmployeeID, Name, Address, Department, and Manager. Normalize it up to BCNF.

(An employee belongs to **one department**, and each department has **one manager**)

Step 1: Identify Required Attributes & Thier data types

click this link for see -> https://dbdiagram.io/d/employee_management_system-67e3af1275d75cc844805985

Step 2: create Initial Table

```
mysql> create table employee_management_system(EmployeeID int primary key, name varchar(20) not null
, address text, department varchar(20), manager int not null,created_at timestamp);
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> desc employee_management_system;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| employeeID | int         | NO   | PRI | NULL    |       |
| name       | varchar(20) | NO   |     | NULL    |       |
| address    | text        | YES  |     | NULL    |       |
| department | varchar(20) | YES  |     | NULL    |       |
| manager    | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Step 3: Insert Values Into Table

```
mysql> insert into employee_management_system(employeeID,name,address,department,manager) values(1,'
Pankaj','Paonta Sahib','IT','Vikas');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from employee_management_system;
+------------+--------+--------------+------------+---------+
| employeeID | name   | address      | department | manager |
+------------+--------+--------------+------------+---------+
|          1 | Pankaj | Paonta Sahib | IT         | Vikas   |
+------------+--------+--------------+------------+---------+
1 row in set (0.00 sec)
```

```
mysql> select * from employee_management_system;
+------------+--------+--------------+------------+---------+
| employeeID | name   | address      | department | manager |
+------------+--------+--------------+------------+---------+
|          1 | Pankaj | Paonta Sahib | IT         | Vikas   |
|          2 | Mehul  | Chandigarh   | IT         | Vikas   |
|          3 | Anuj   | Chandigarh   | ECE        | Dinesh  |
+------------+--------+--------------+------------+---------+
3 rows in set (0.00 sec)
```

Step 4: Convert to 1NF

1NF Rules:

1. Ensure all attributes have atomic values (no multiple values in a single column).
2. Each column must have a unique name.

3 Ensure the table has a primary key.

the table follows 1NF since all attributes contain atomic values.

Step 5: Convert to 2NF

2NF Rules:

1 Must be in 1NF.

2 No Partial Dependency – All non-key attributes should depend entirely on the primary key.

Identifying Dependencies

- The Department and Manager are not dependent on EmployeeID but on the department itself.
- Solution: Split into two tables to separate employee details and department details.

```
mysql> create table employee(id int primary key, name varchar(20) not null, address text, department
ID int not null);
Query OK, 0 rows affected (0.17 sec)
```

```
mysql> select * from employee;
+----+--------+--------------+--------------+
| id | name   | address      | departmentID |
+----+--------+--------------+--------------+
|  1 | Pankaj | Paonta Sahib |          101 |
|  2 | Mehul  | Chandigarh   |          101 |
|  3 | Anuj   | Chandigarh   |          102 |
+----+--------+--------------+--------------+
3 rows in set (0.00 sec)
```

```
mysql> create table department(id int primary key, name varchar(20) not null, managerID int not null
);
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from department;
+-----+------+-----------+
| id  | name | managerID |
+-----+------+-----------+
| 101 | IT   |         1 |
| 102 | ECE  |         2 |
+-----+------+-----------+
2 rows in set (0.00 sec)
```

The tables follow 2NF since there are no partial dependencies.

Step 6: Convert to 3NF

3NF Rules:

1. Must be in 2NF.

2 No Transitive Dependency – Non-key attributes should not depend on other non-key attributes.

Identifying Transitive Dependencies

- The Manager is dependent on Department, not DepartmentID.
- Solution: We already separated department details, so the table is in **3NF**.

Revised Tables (Now in 3NF)

```
mysql> select * from employee;
+----+--------+--------------+--------------+
| id | name   | address      | departmentID |
+----+--------+--------------+--------------+
|  1 | Pankaj | Paonta Sahib |          101 |
|  2 | Mehul  | Chandigarh   |          101 |
|  3 | Anuj   | Chandigarh   |          102 |
+----+--------+--------------+--------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from department;
+-----+------+-----------+
| id  | name | managerID |
+-----+------+-----------+
| 101 | IT   |         1 |
| 102 | ECE  |         2 |
+-----+------+-----------+
2 rows in set (0.00 sec)
```

```
mysql> select * from manager;
+----+--------+
| id | name   |
+----+--------+
|  1 | Vikas  |
|  2 | Dinesh |
+----+--------+
2 rows in set (0.00 sec)
```

The tables follow 3NF as there are no transitive dependencies.

Step 7: Convert to BCNF

BCNF Rules:

1. Must be in 3NF.
2. If there are multiple candidate keys, only superkeys should determine non-key attributes.

Checking for BCNF Violations

- The Department Table already follows BCNF because:
    - DepartmentID is the only candidate key.
    - Manager is functionally dependent on DepartmentID.

The tables are now in BCNF