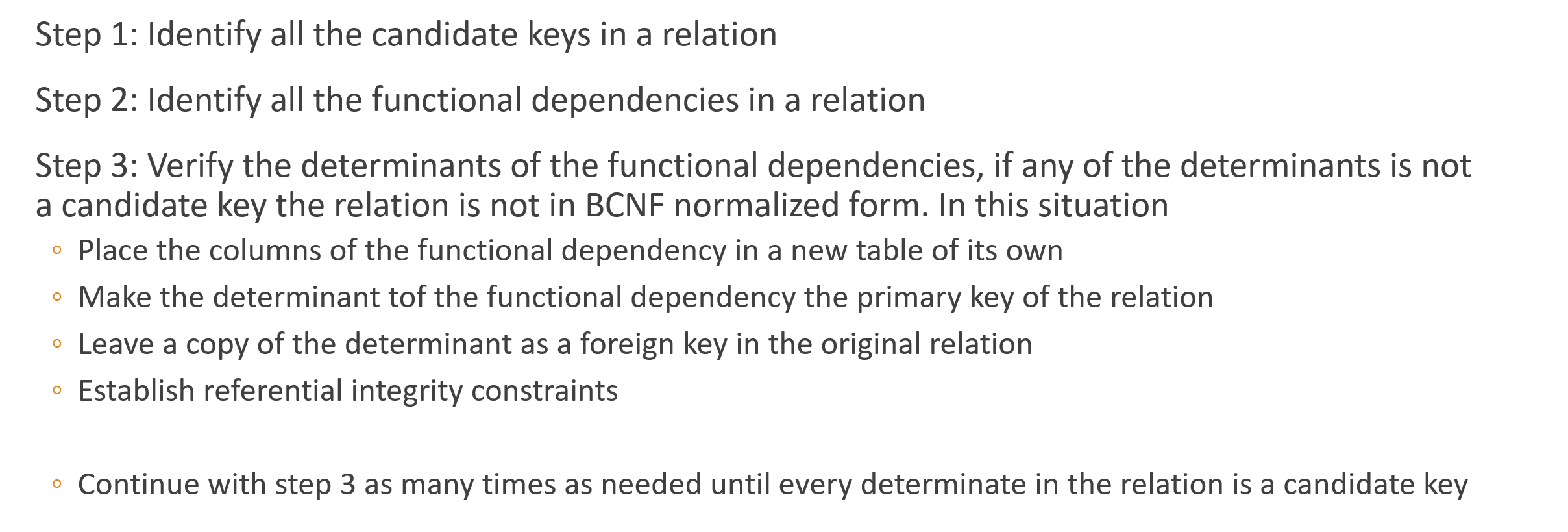
**Normalization Class Notes**



**EXAMPLE1 :Let us consider the following relation**

EMPLOYEE (EmployeeID, SSN, EmployeeFirstName,EmployeeLastName,EmployeePhone,DepartmentID, DepartmentName,DepartmentPhone,DepartmentAddress)

Business Rules: Employee works in one department. A department can have many employees.

Step1: Identify all candidate keys

Canindate keys:

1. EmployeeID
2. SSN

Are there any other canindate keys? No

Step2: Identify functional dependencies

1. EmployeeID-> SSN, EmployeeFirstName,EmployeeLastName,EmployeePhone,DepartmentID, DepartmentName,DepartmentPhone,DepartmentAddress

2. SSN-> EmployeeID, EmployeeFirstName,EmployeeLastName,EmployeePhone,DepartmentID, DepartmentName,DepartmentPhone,DepartmentAddress

3. DepartmentID functionally determines DepartmentName, DepartmentPhone and DepartmentAddress

DepartmentID🡪 DepartmentName, DeartmentPhone, DepartmentAddress

Step3: Is there a determinant that is not a canindate key?

Yes- DepartmentID is a determinant and not a candidate key

So we split the functional dependency to a relation of its own

DEPARTMENT (DepartmentID, DepartmentName, DepartmentPhone,DepartmentAddress)

We then leave a copy of DepartmentID in the original relation as a foreign key

We do not pick SSN as the primary key for security reasons.

**Final Normalized relations in BCNF**

EMPLOYEE(EmployeeID, SSN, EmployeeFirstName,EmployeeLastName,EmployeePhone,*DepartmentID*)

DEPARTMENT(DepartmentID, DepartmentName, DepartmentPhone, DepartmentAddress)

We also create the referential integrity rule

1. DepartmentID in EMPLOYEE must exist in DepartmentID in DEPARTMENT

Now every determinant of every relation is a canindate key so we stop.

**Note that Primary keys are underlined in a relation and the foreign keys are italized. This is improtant to denote in your final BCNF relations**

**EXAMPLE 2: Let us consider the following relation**

PATIENT(PatientID, PatientName, PatientEmail, PatientDOB, PrimaryDoctorLicenseNumber, PrimaryDoctorName, PrimaryDoctorTelephone, PracticeID, PracticeName, PracticeLocation, PracticePhone)

**BusinessRule**s: A patient can be seen at only one practice location. A patient has only one primary doctor. A doctor can practice in many locations. A doctor can see many patients. A practice has many patients

Step1: Identify all candidate keys

1. PatientID

PatientID-> PatientName, PatientDOB, PatientEmail, PrimaryDoctorLicenseNumber, PrimaryDoctorName, PrimaryDoctorTelephone, PracticeName, PracticeLocation, PracticePhone

Are there any other canindate keys? No

Step2: Identify functional dependencies

PatientID-> PatientName, PatientDOB, PatientEmail, PrimaryDoctorLicenseNumber, PrimaryDoctorName, PrimaryDoctorTelephone, PracticeName, PracticeLocation, PracticePhone

PrimaryDoctorLicenseNumber-> PrimaryDoctorName, PrimaryDoctorTelephone

PracticeID-> PracticeName, PracticeLocation, PracticePhone

Step3: Is there a functional determinant that is not a canindate key?

Yes-

1. PrimaryDoctorLicenseNumber is a determinant and not a candidate key
2. PracticeID is a determinant and not a candidate key

So we split the functional dependency to a relation of its own

So we split the functional dependencies to a relation of its own

DOCTOR(PrimaryDoctorLicenseNumber, PrimaryDoctorName, PrimaryDoctorTelephone)

PRACTICE (PracticeID, PracticeName, PracticeLocation, PracticePhone)

We then leave a copy of PrimaryDoctorLicenseNumber and PracticeID in the original relation as a foreign keys

**Final Normalized relations in BCNF**

**PATIENT**(PatientID, PatientName, PatientDOB, PatientEmail, *DoctorLicenseNumber*, *PracticeID*)

**DOCTOR**(PrimaryDoctorLicenseNumber, PrimaryDoctorName, PrimaryDoctorTelephone)

**PRACTICE** (PracticeID, PracticeName, PracticeLocation, PracticePhone)

We also create the referential integrity rule

1. DoctorLicenseNumber in PATIENT must exist in DoctorLicenseNumber in DOCTOR
2. PracticeID in PATIENT must exist in PracticceID in PRACTICE