Department of Software Engineering
Mehran University of Engineering and Technology, Jamshoro

Course: SW222 – Database Management & Administration			
Instructor	Ms Shafiya Qadeer	**Practical/Lab No.**	03
Date	27/28-01-2021	**CLOs**	3
Signature		**Assessment Score**	2 Marks

Topic	**To become familiar with constraints enforcement**
Objectives	- **DATABASE NORMALIZATION**

Lab Discussion: Theoretical concepts and Procedural steps

DATABASE ANOMALIES

- Database anomalies are the problems or errors that occur due to redundancy in the relations/tables.
- There are three types of anomalies that occur when the database is not normalized.

1. Insertion Anomaly
2. Update Anomaly
3. Deletion anomaly

S_id	**S_Name**	**S_Address**	**Subject_opted**
401	Ali	Hyderabad	C++
402	haris	karachi	Java
403	erum	Lahore	Java
404	Ali	Hyderbad	PHP

Updation Anamoly: To update address of a student who occurs twice or more than twice in a table, we will have to update S_Address column in all the rows.

Insertion Anamoly: Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert NULL there, leading to Insertion Anamoly.

Deletion Anamoly: If (S_id) 402 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

NORMALIZATION

- Database Normalization is a technique of organizing the data in the database
- It is a method of decomposing tables to eliminate data redundancy/duplication. (for example, storing the same data in more than one table).

PURPOSE OF NORMALIZATION:

- It is used for mainly two purpose:

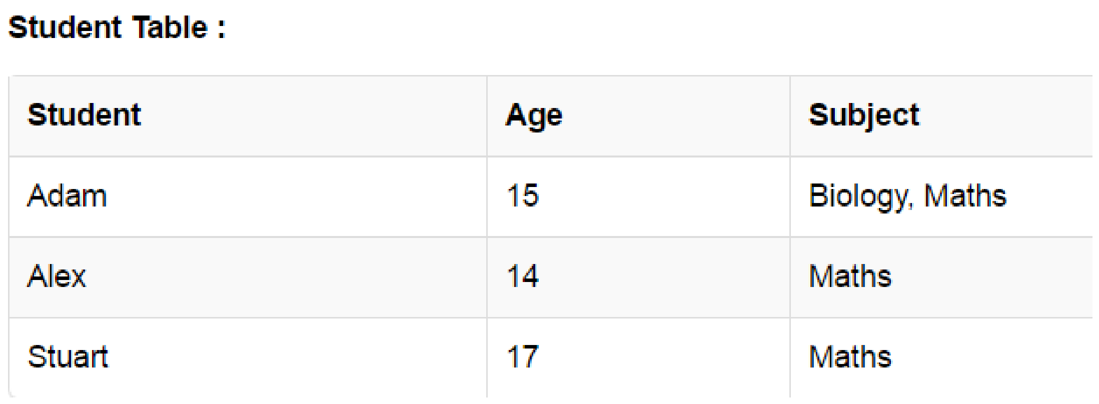
1. Removing duplicated(reduntant) data from the relation tables.
2. Ensuring data dependencies make sense i.e data is logically stored.

- Normalization rule are divided into following normal forms.
- (1NF) First Normal Form
- (2NF) Second Normal Form
- (3NF) Third Normal Form
- BCNF

1NF (First Normal Form) Rules

- Each table cell should contain single value.
- This rule defines that all the attributes in a relation must have atomic/single values. OR
- an attribute (columns) of a table cannot hold multiple values. It should hold only atomic/single values.

EXAMPLE1

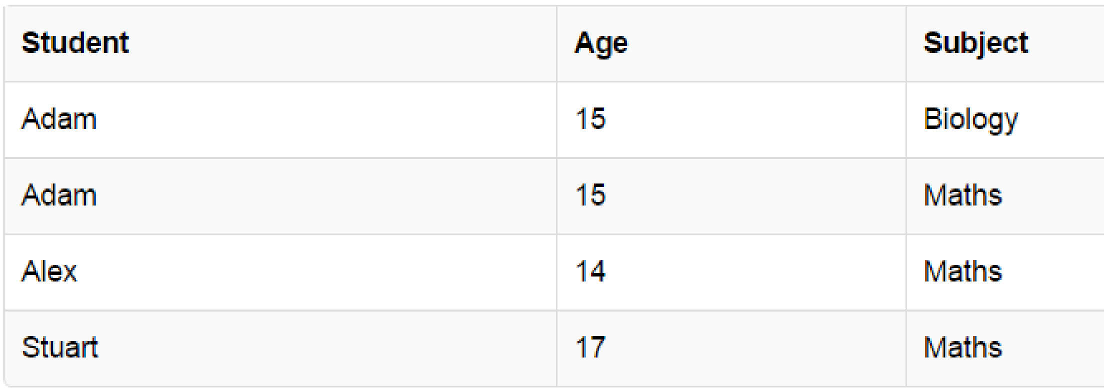


Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

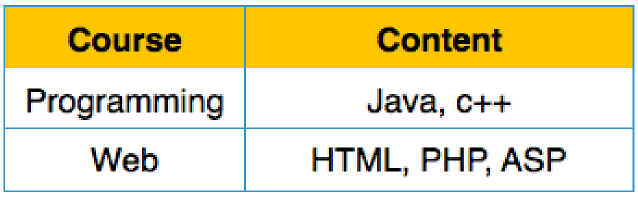
In 1NF, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :



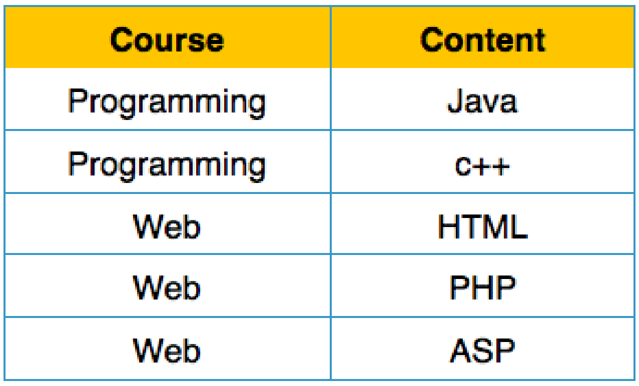
Student	Age	Subject
Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

EXAMPLE of 1NF:



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.



Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

CANDIDATE KEY:

A candidate key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more candidate keys, but one candidate key is unique, and it is called the primary key.

- **PRIME ATTRIBUTES:**

  The attributes which are the part of candidate key are known as prime attributes.

- **NON- PRIME ATTRIBUTES:**

  An attribute that does not occur in candidate key is called a non-prime attribute.

- **FUNCTIONAL DEPENDENCY:**

Functional dependency is a relationship that exists between attributes. If R is a relation with attributes X and Y, a functional dependency between the attributes are represented as X->Y, which specifies Y is functionally dependent on X.

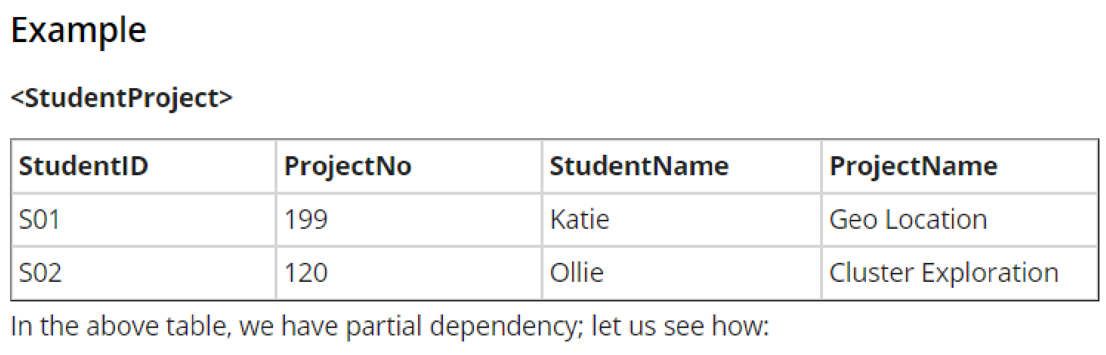
- **full functional dependency:**

If removal of any attribute A from X means that the dependency does not hold any more; that is, for any attribute A ε X, (X − {A}) does not functionally determine Y.

Partial dependency: means that a nonprime attribute is functionally dependent on part of a candidate key.

full functional dependency Example:

Roll no	Name	C.G.P.A
123	Ali	3.0
456	Rabeea	3.2
789	Bushra	3.6
145	Hina	2.4
201	Ali	2.5

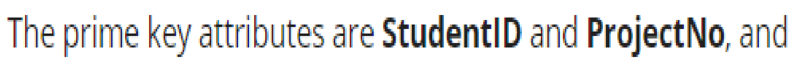


Example

<StudentProject>

StudentID	ProjectNo	StudentName	ProjectName
S01	199	Katie	Geo Location
S02	120	Ollie	Cluster Exploration

In the above table, we have partial dependency; let us see how:


The prime key attributes are **StudentID** and **ProjectNo**, and

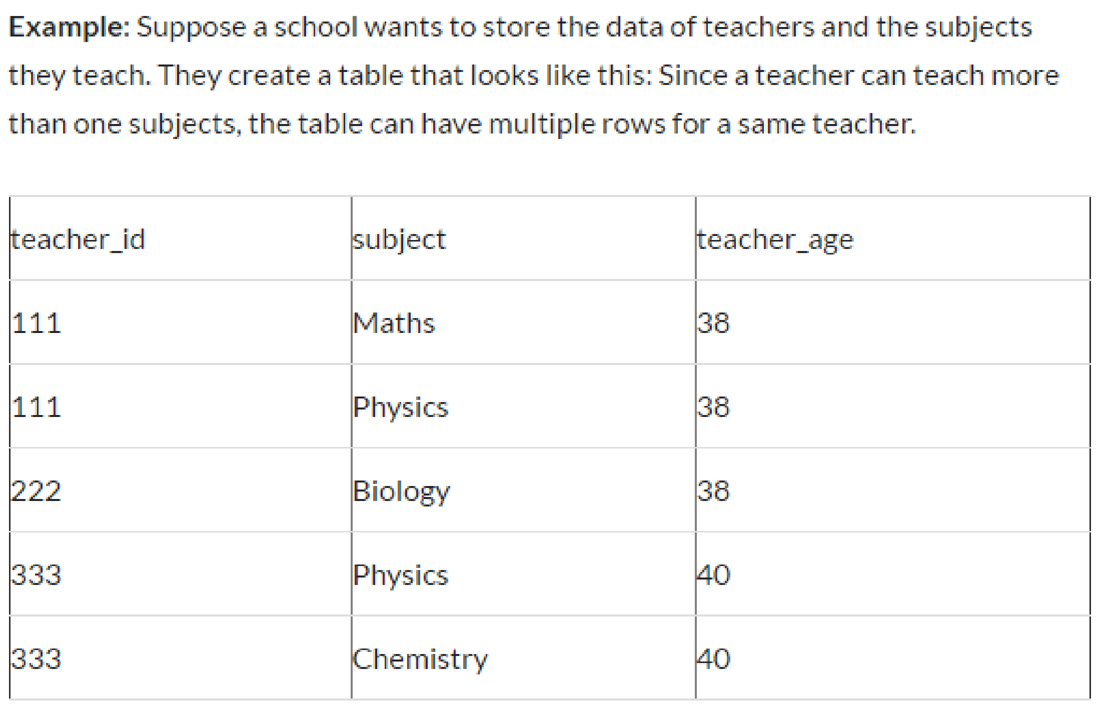
Second Normal Form (2NF)

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

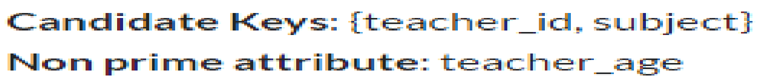
1. Table is in 1NF (First normal form)

2. Every non-prime attribute of the table is fully functional dependent on the whole of a candidate key.

- ❖ no partial dependency is allowed in 2NF.

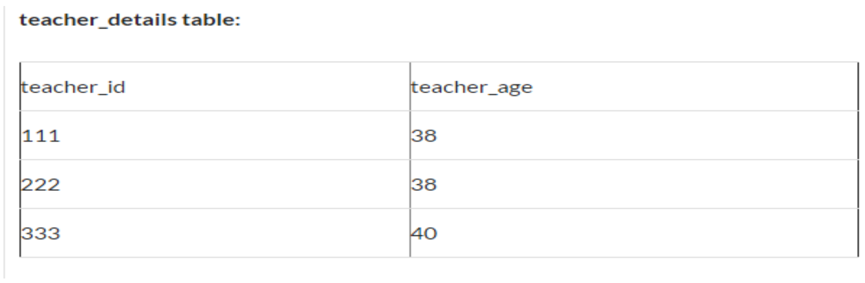

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

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

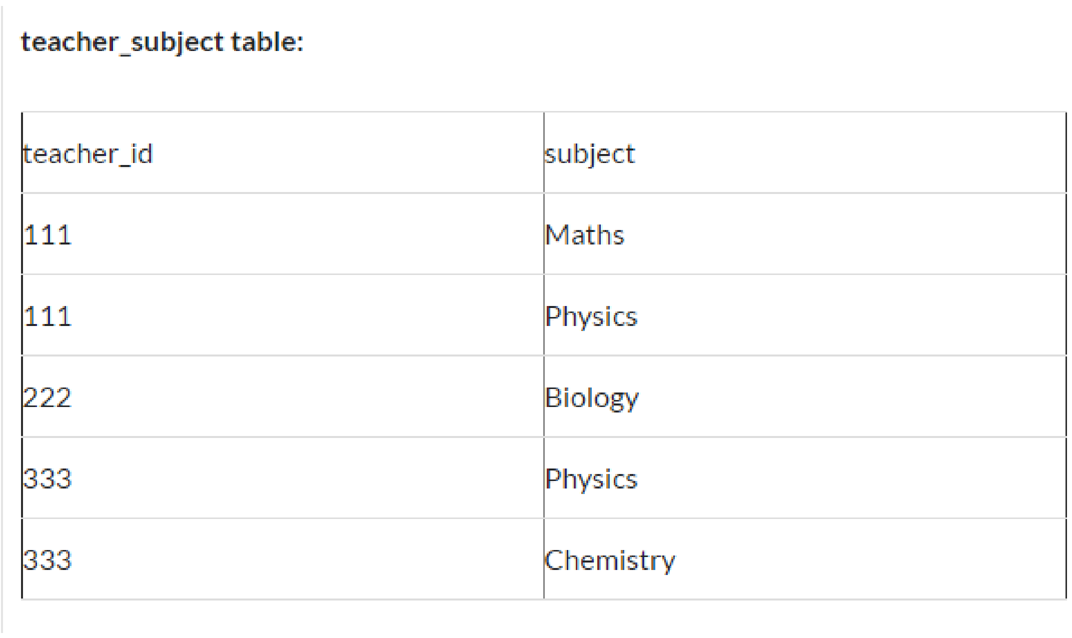

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age

If a table is not in second normal form:

- ➢ Move that data item i.e teacher_age and the primary key on which it is functionally dependent to a new table.


teacher_details table:

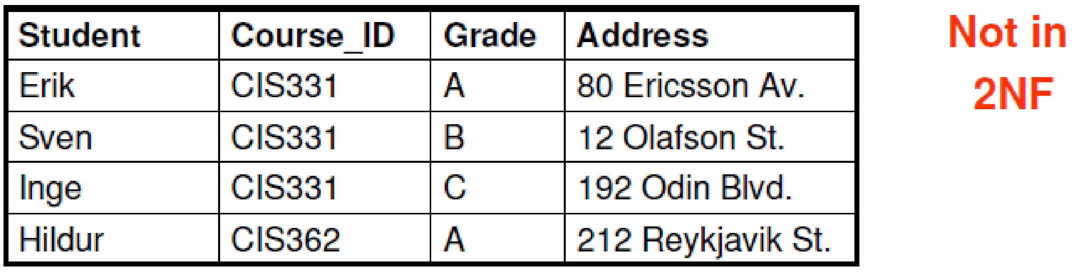
teacher_id	teacher_age
111	38
222	38
333	40



teacher_subject table:

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

EXAMPLE of 2NF:

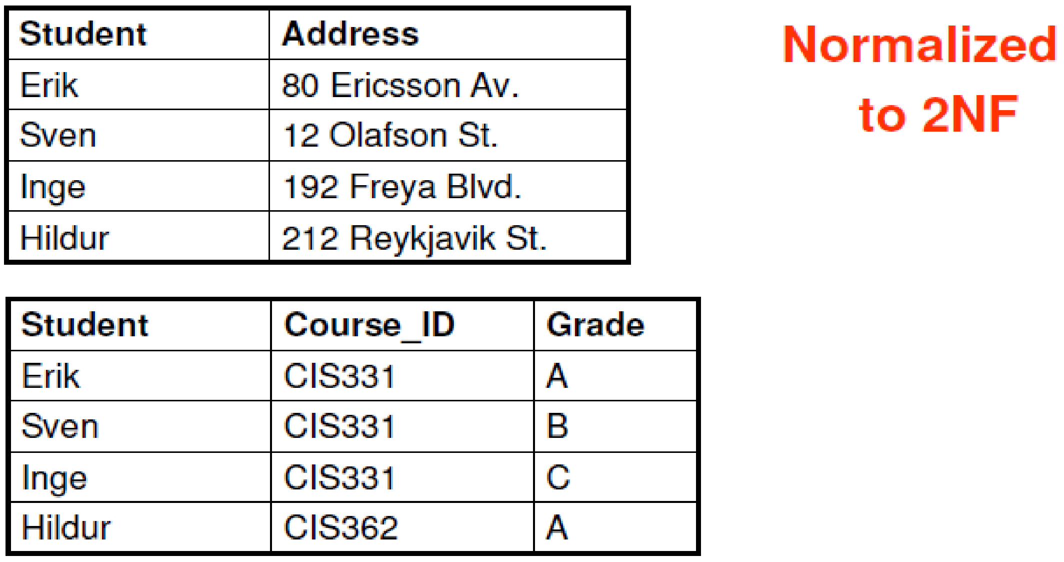


Student	Course_ID	Grade	Address
Erik	CIS331	A	80 Ericsson Av.
Sven	CIS331	B	12 Olafson St.
Inge	CIS331	C	192 Odin Blvd.
Hildur	CIS362	A	212 Reykjavik St.

Not in 2NF

Candidate key: student, course_id

Non prime attributed: address, grade



Student	Address
Erik	80 Ericsson Av.
Sven	12 Olafson St.
Inge	192 Freya Blvd.
Hildur	212 Reykjavik St.

Student	Course_ID	Grade
Erik	CIS331	A
Sven	CIS331	B
Inge	CIS331	C
Hildur	CIS362	A

Normalized to 2NF

TRANSITIVE DEPENDENCY:

A condition where A, B and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on A via B.

Third Normal form (3NF)

- The relation R (table) must be in 1NF and second normal form (2NF).
- No non-primary-key column is transitively dependent on the primary key.
- There should not be the case that a non-prime attribute is determined by another non-prime attribute. OR
- No non-prime attribute is dependent on another non-prime attribute.

Student_Detail Table :

Student_id	Student_name	DOB	Street	city	State	Zip

In this table Student_id is Primary key, but street, city and state(non-prime attributes) depends upon Zip(another non-prime attribute). The dependency between zip and street, city and state is called transitive dependency. Hence to apply 3NF, we need to move the street, city and state to new table, with Zip as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	Zip

Address Table :

Zip	Street	city	state

EXAMPLE of 3NF:

Student	Course_ID	Grade	Grade_value
Erik	CIS331	A	4.00
Sven	CIS331	B	3.00
Inge	CIS331	C	2.00
Hildur	CIS362	A	4.00

Not in 3NF

Grade_value and grade both are non-prime attribute and dependent on each other. and transitively dependent on student. (not allowed in 3NF)

Student	Course_ID	Grade
Erik	CIS331	A
Sven	CIS331	B
Inge	CIS331	C
Hildur	CIS362	A

Grade	Grade_value
A	4.00
B	3.00
C	2.00

Normalized to 3NF

Boyce and Codd Normal Form (BCNF)

- It is an advance version of 3NF that's why it is also referred as 3.5NF.
- 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.

X->Y

Y is functionally dependent on X.

and X is a determinant of Y. (means Y is determined by X).

- **Boyce-Codd Normal Form (BCNF):**

"Everything should depend on the key, the whole key, and nothing but the key".

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

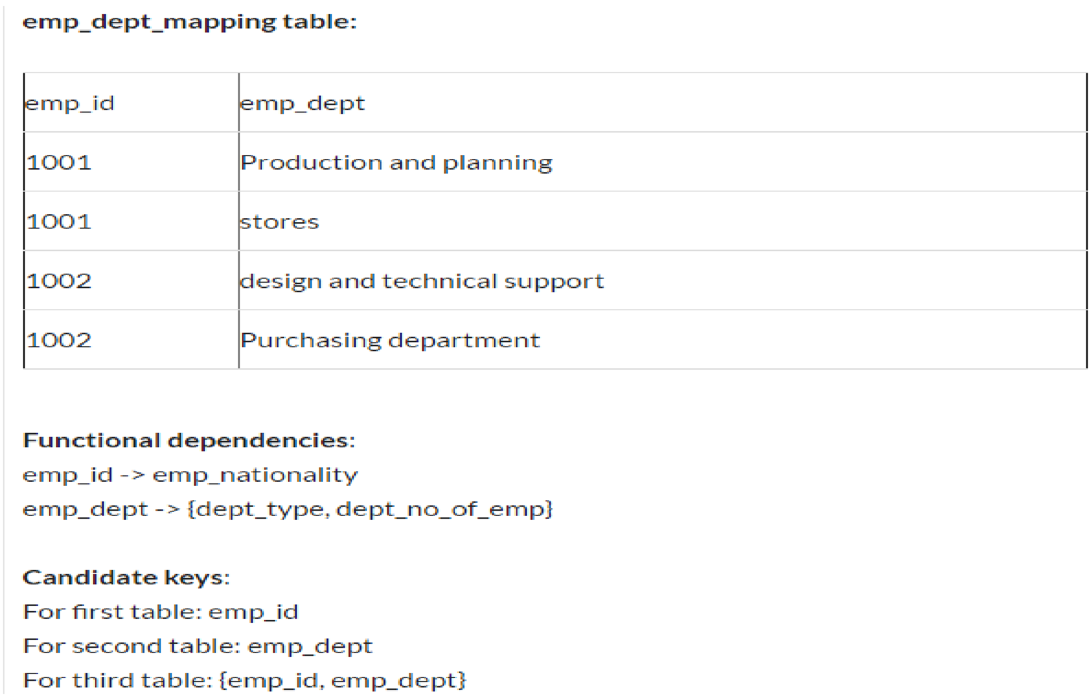
The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:
emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600



emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}

TYPES OF KEYS:

1. SUPER KEY:

A superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple. A candidate key is a minimal set of attributes necessary to identify a tuple; this is also called a minimal superkey.

Every CK is a super key but Every Superkey cannot be a candidate key.

2. PRIMARY KEY:

- A primary key, also called a primary keyword, is a key in a relational database that is unique for each record. It is a unique identifier, such as a driver license number, telephone number (including area code), or vehicle identification number (VIN). A relational database must always have one and only one primary key and it doesn't contain null value.

3. FOREIGN KEY:

It is defined in a second table, but it refers to the primary key in its own table.

Lab Tasks

TASK1:

ITEM	**COLORS**	**PRICE($)**	**TAX**
T-shirt	Red, Blue	12.00	0.60
Polo	Red, Yellow	12.00	0.60
Sweat Shirt	Blue, Black	25.00	1.25

Convert the above table in: 1NF, 2NF, 3NF

TASK2:

SUPPLIER	**NAME**	**CITY_ID**	**CITY**	**PART**	**QUANTITY**
S1	Ali	10	Paris	P3	257
S1	Ali	10	Paris	P1	500
S1	Ali	10	Paris	P4	125
S2	James	12	London	P3	200
S7	Robert	10	Paris	P4	342

Convert the above table in: 1NF, 2NF, 3NF