Unit 3. Normalization

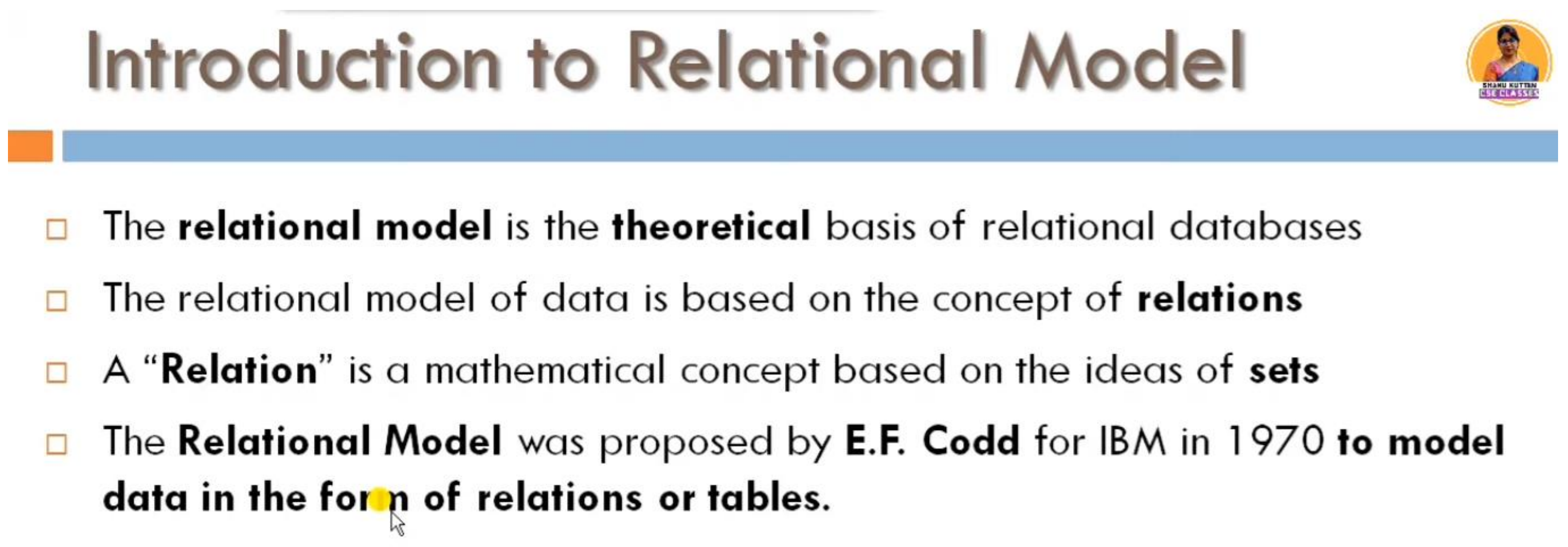
3.1. Importance of Normalization

3.2. Functional Dependencies-– definition, trivial and non-trivial FD, closure of FD set, closure of attributes

3.3. Integrity and Domain constraints

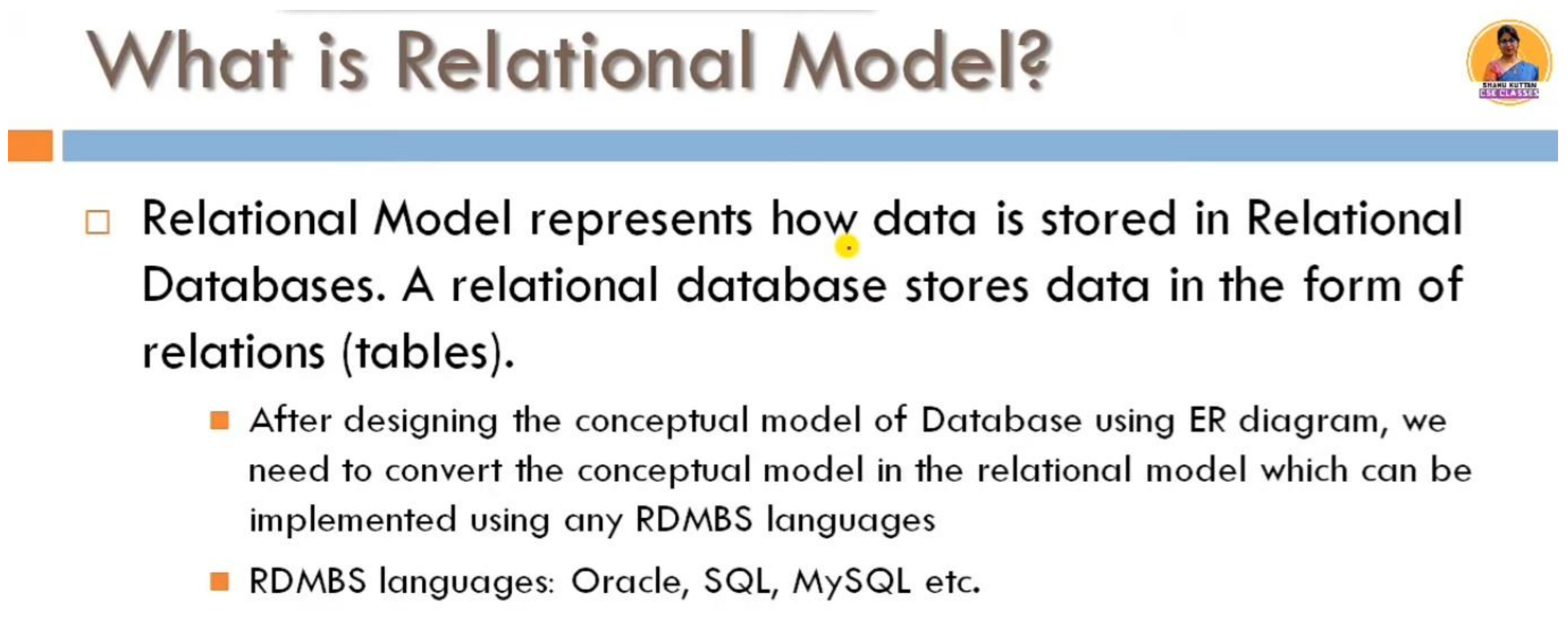
3.4. Normal forms (1NF, 2NF, 3NF, BCNF)

3.3. Integrity and Domain constraints



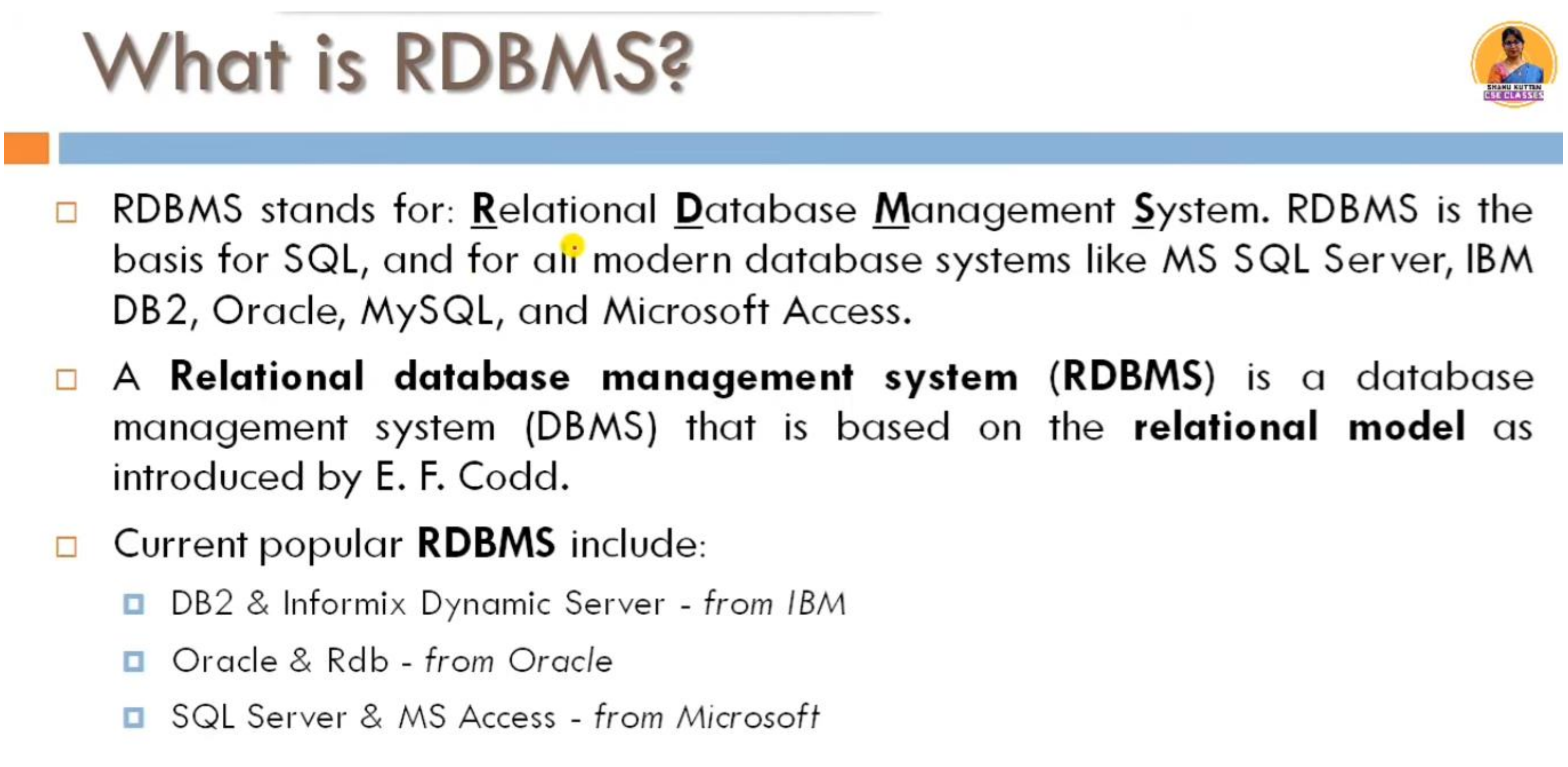
Introduction to Relational Model

- The **relational model** is the **theoretical** basis of relational databases
- The relational model of data is based on the concept of **relations**
- A "**Relation**" is a mathematical concept based on the ideas of **sets**
- The **Relational Model** was proposed by **E.F. Codd** for IBM in 1970 **to model data in the form of relations or tables.**



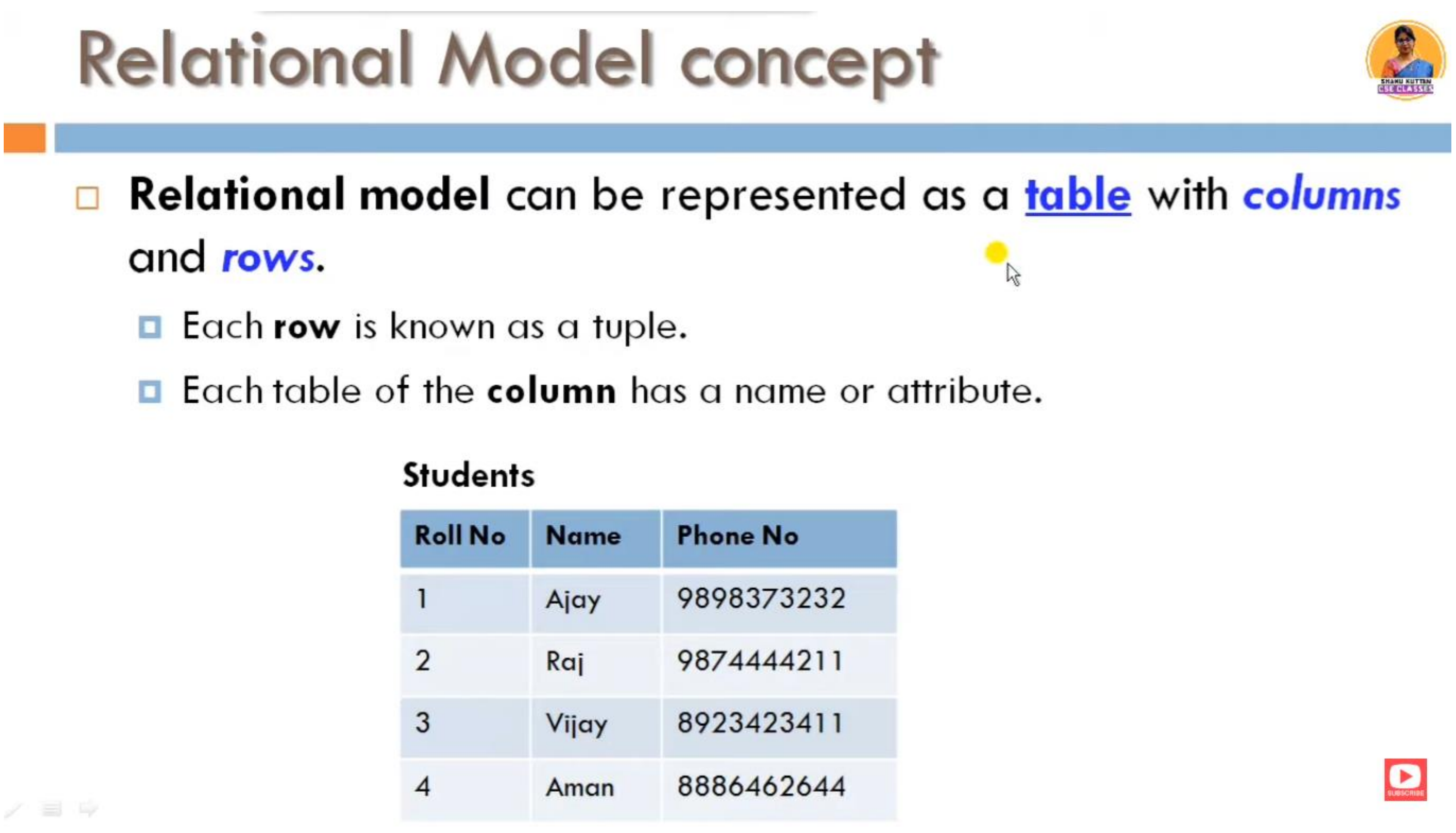
What is Relational Model?

- Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).
  - After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDMBS languages
  - RDMBS languages: Oracle, SQL, MySQL etc.



What is RDBMS?

- RDBMS stands for: Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- A **Relational database management system** (**RDBMS**) is a database management system (DBMS) that is based on the **relational model** as introduced by E. F. Codd.
- Current popular **RDBMS** include:
  - DB2 & Informix Dynamic Server - *from IBM*
  - Oracle & Rdb - *from Oracle*
  - SQL Server & MS Access - *from Microsoft*

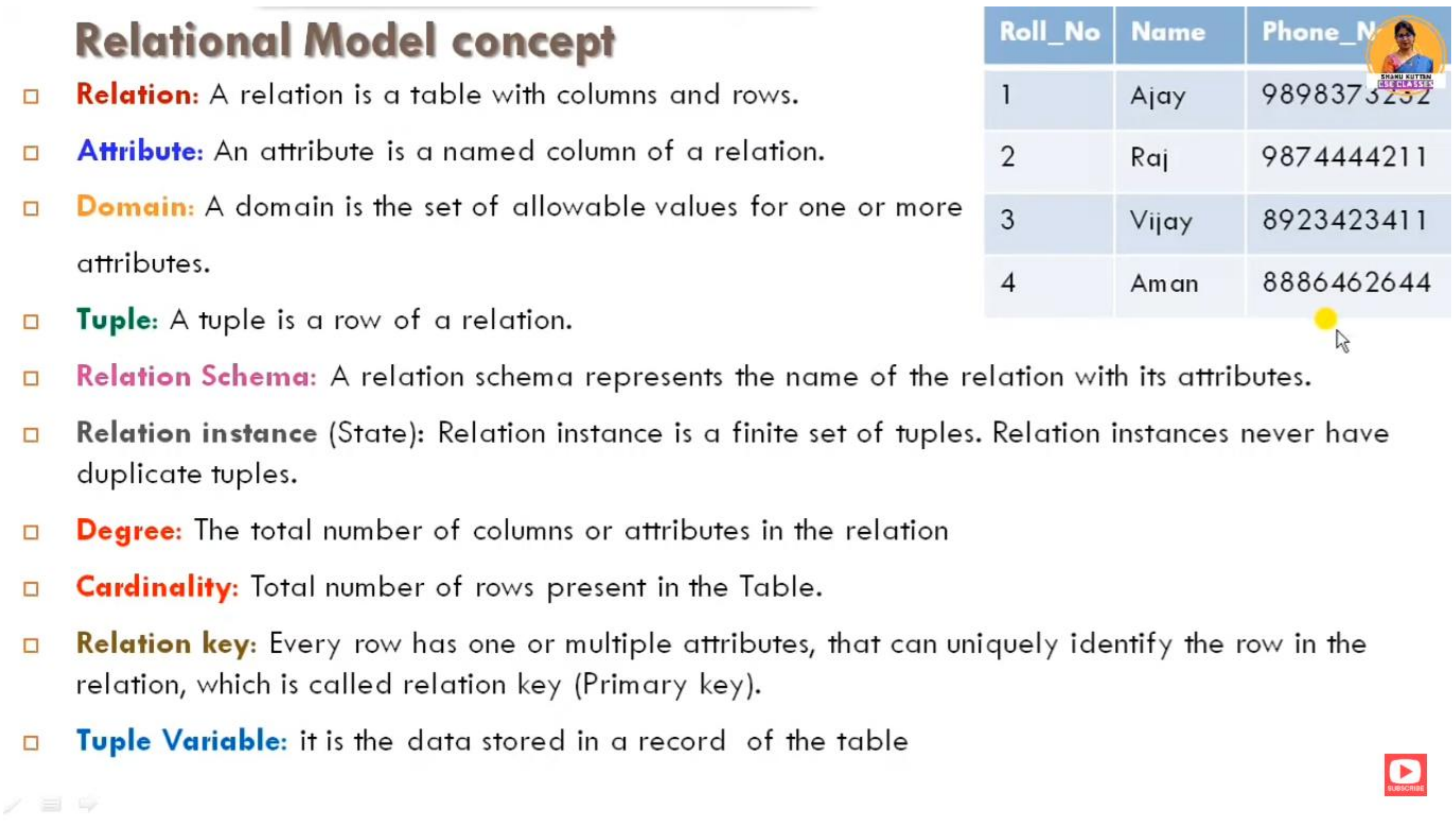


Relational Model concept

- **Relational model** can be represented as a table with ***columns*** and ***rows***.
  - Each **row** is known as a tuple.
  - Each table of the **column** has a name or attribute.

Students

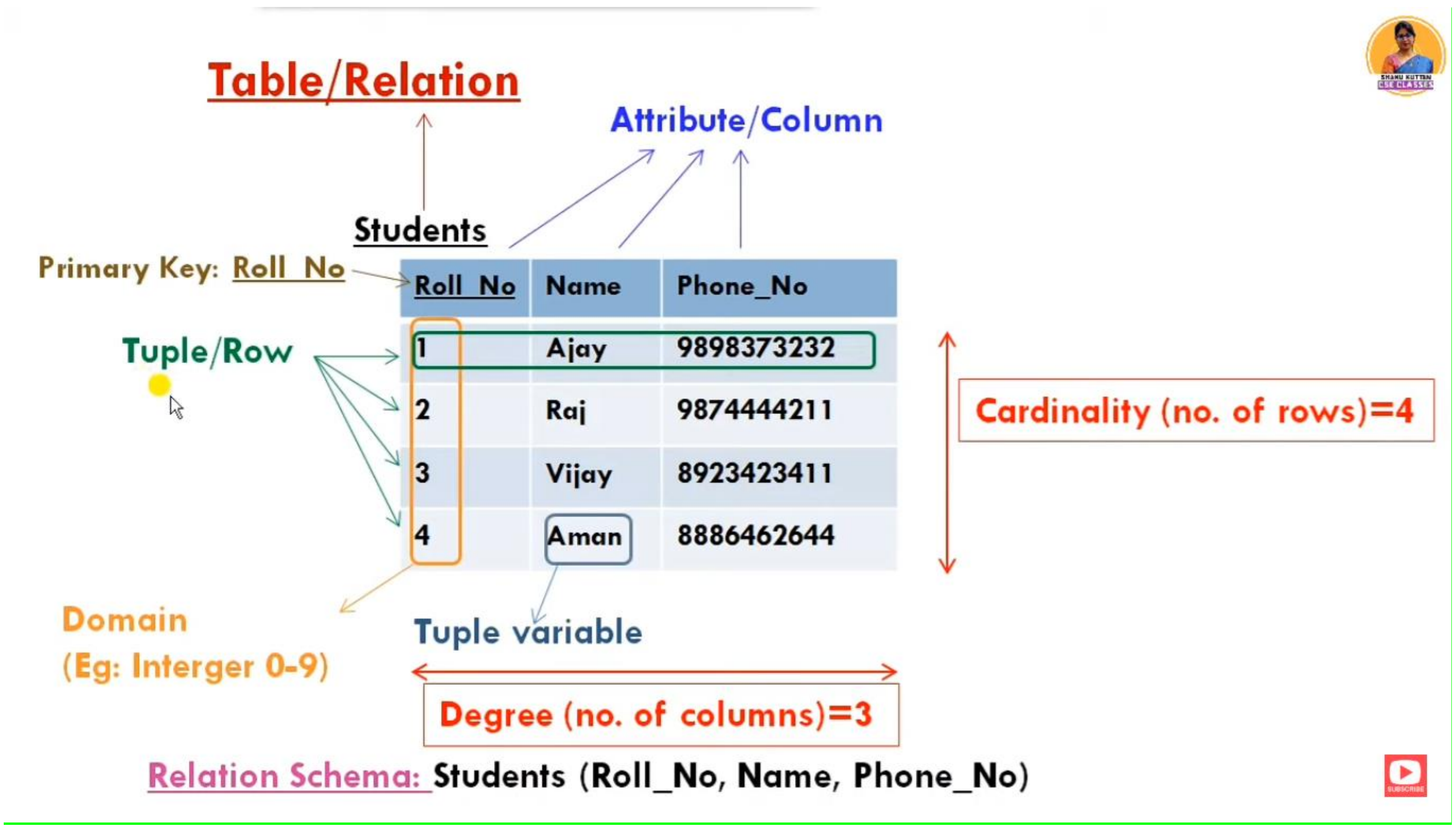
Roll No	Name	Phone No
1	Ajay	9898373232
2	Raj	9874444211
3	Vijay	8923423411
4	Aman	8886462644



Relational Model concept

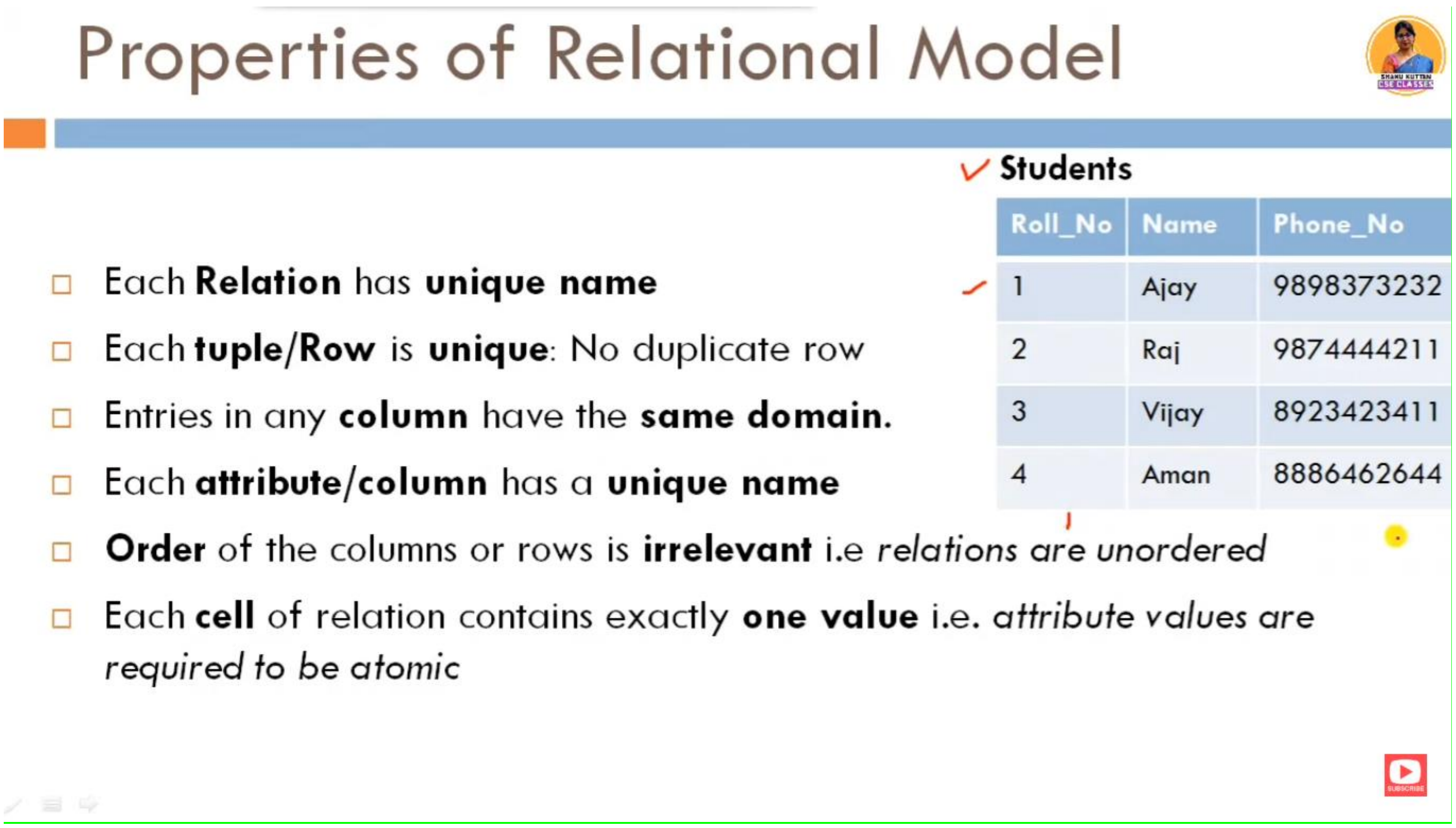
- **Relation:** A relation is a table with columns and rows.
- **Attribute:** An attribute is a named column of a relation.
- **Domain:** A domain is the set of allowable values for one or more attributes.
- **Tuple:** A tuple is a row of a relation.
- **Relation Schema:** A relation schema represents the name of the relation with its attributes.
- **Relation instance** (State): Relation instance is a finite set of tuples. Relation instances never have duplicate tuples.
- **Degree:** The total number of columns or attributes in the relation
- **Cardinality:** Total number of rows present in the Table.
- **Relation key:** Every row has one or multiple attributes, that can uniquely identify the row in the relation, which is called relation key (Primary key).
- **Tuple Variable:** it is the data stored in a record of the table

Roll_No	Name	Phone_N
1	Ajay	9898373232
2	Raj	9874444211
3	Vijay	8923423411
4	Aman	8886462644



Roll_No	Name	Phone_No
1	Ajay	9898373232
2	Raj	9874444211
3	Vijay	8923423411
4	Aman	8886462644

Relation Schema: Students (Roll_No, Name, Phone_No)

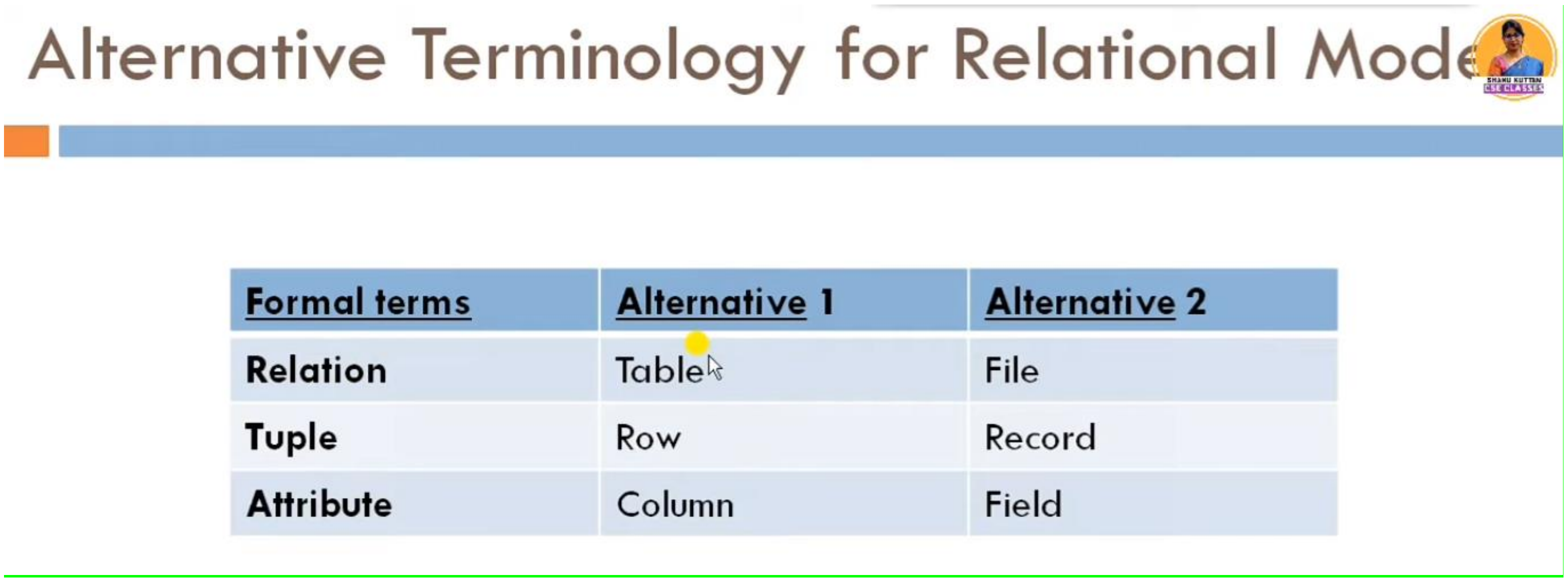


Properties of Relational Model

Students

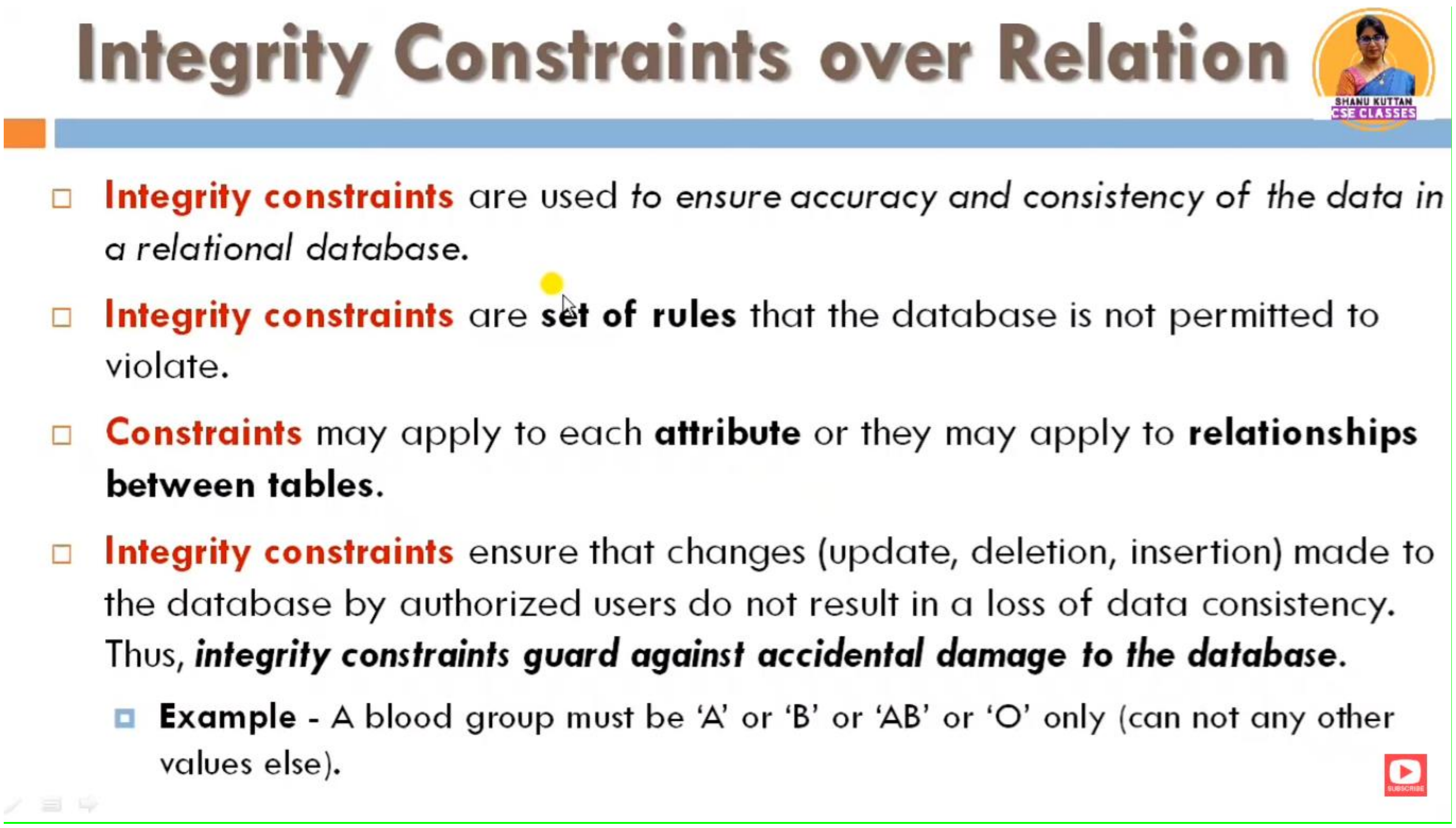
Roll_No	Name	Phone_No
1	Ajay	9898373232
2	Raj	9874444211
3	Vijay	8923423411
4	Aman	8886462644

- Each **Relation** has **unique name**
- Each **tuple/Row** is **unique**: No duplicate row
- Entries in any **column** have the **same domain.**
- Each **attribute/column** has a **unique name**
- **Order** of the columns or rows is **irrelevant** i.e *relations are unordered*
- Each **cell** of relation contains exactly **one value** i.e. *attribute values are required to be atomic*



Alternative Terminology for Relational Mode

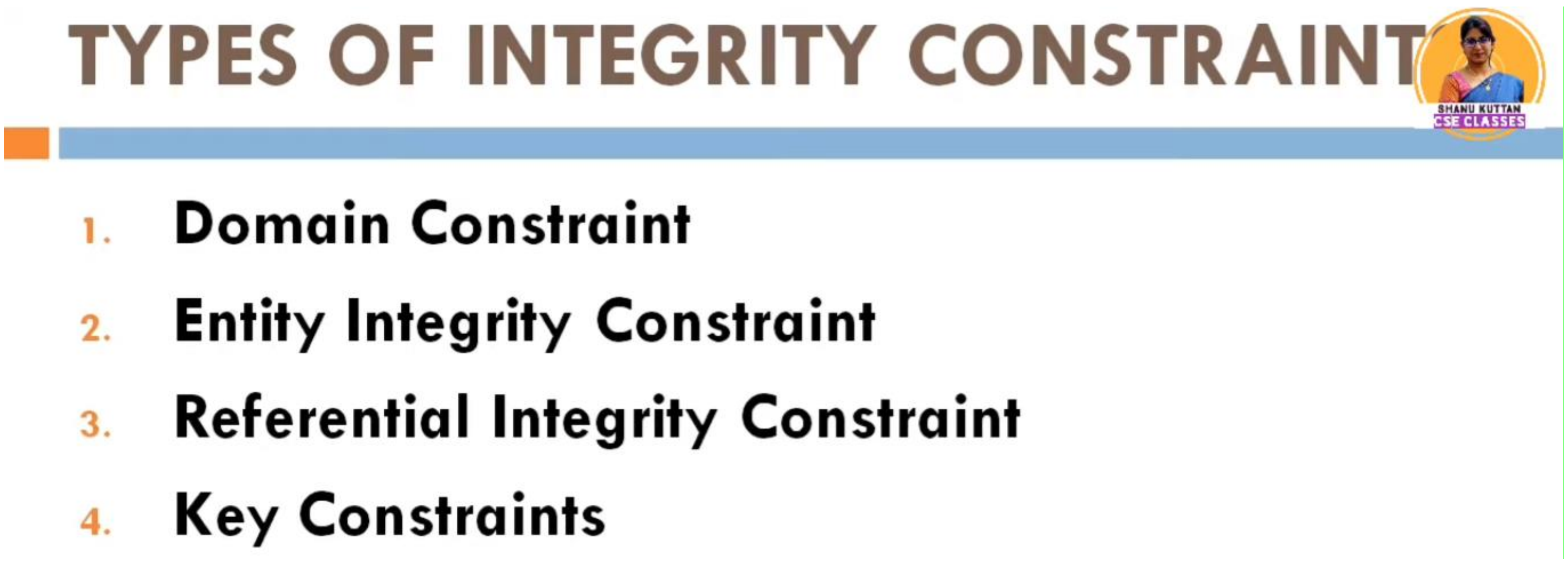
Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field



Integrity Constraints over Relation

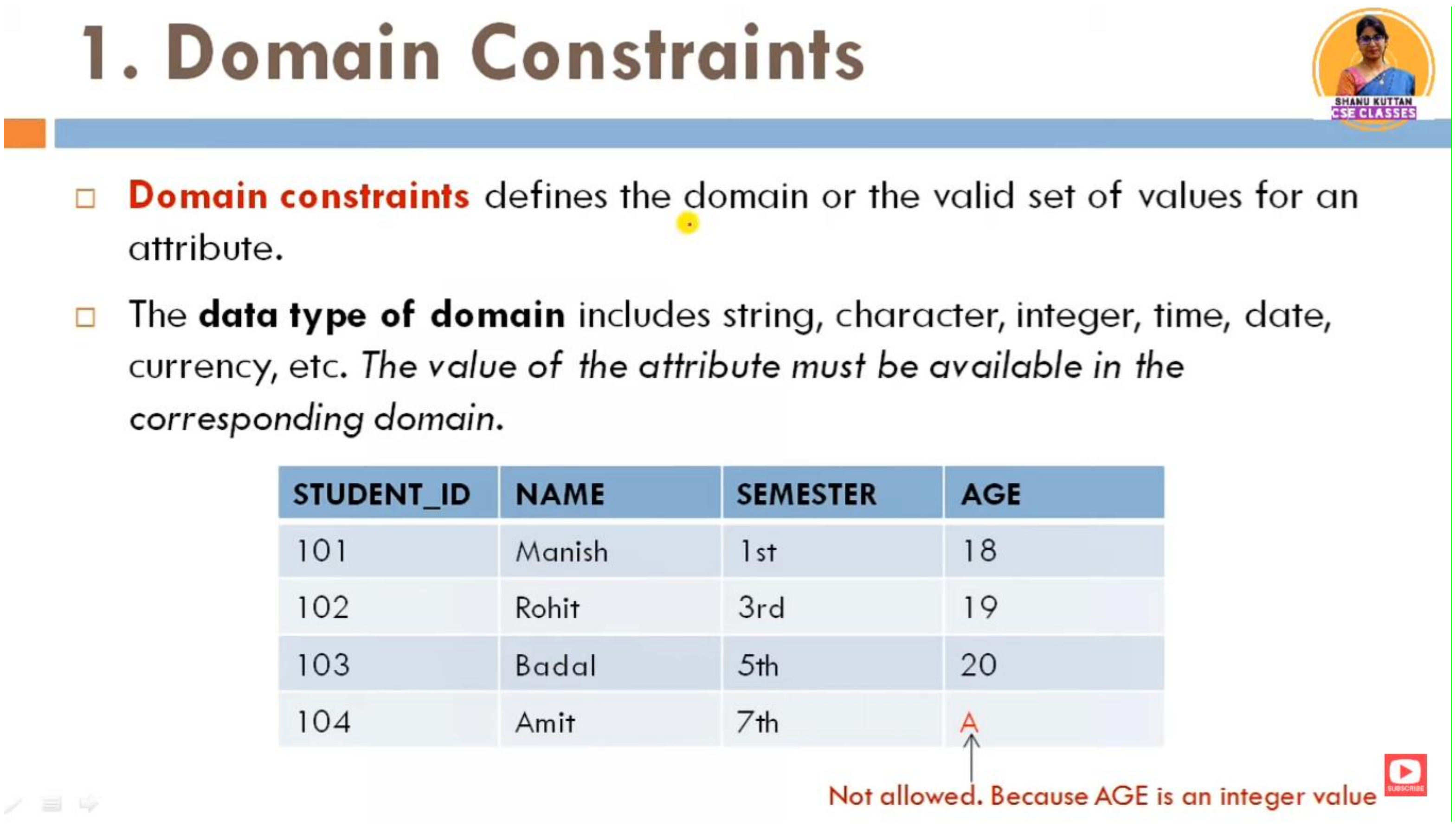
- **Integrity constraints** are used *to ensure accuracy and consistency of the data in a relational database.*
- **Integrity constraints** are **set of rules** that the database is not permitted to violate.
- **Constraints** may apply to each **attribute** or they may apply to **relationships between tables.**
- **Integrity constraints** ensure that changes (update, deletion, insertion) made to the database by authorized users do not result in a loss of data consistency. Thus, ***integrity constraints guard against accidental damage to the database.***
  - **Example -** A blood group must be 'A' or 'B' or 'AB' or 'O' only (can not any other values else).

- *In DBMS (Database Management System), constraints are rules or restrictions enforced on the data in a database to ensure its accuracy, consistency, and reliability.*
- *Constraints are used to prevent invalid data from being entered into the database and to maintain the integrity of the database*



TYPES OF INTEGRITY CONSTRAINT

1. **Domain Constraint**
2. **Entity Integrity Constraint**
3. **Referential Integrity Constraint**
4. **Key Constraints**

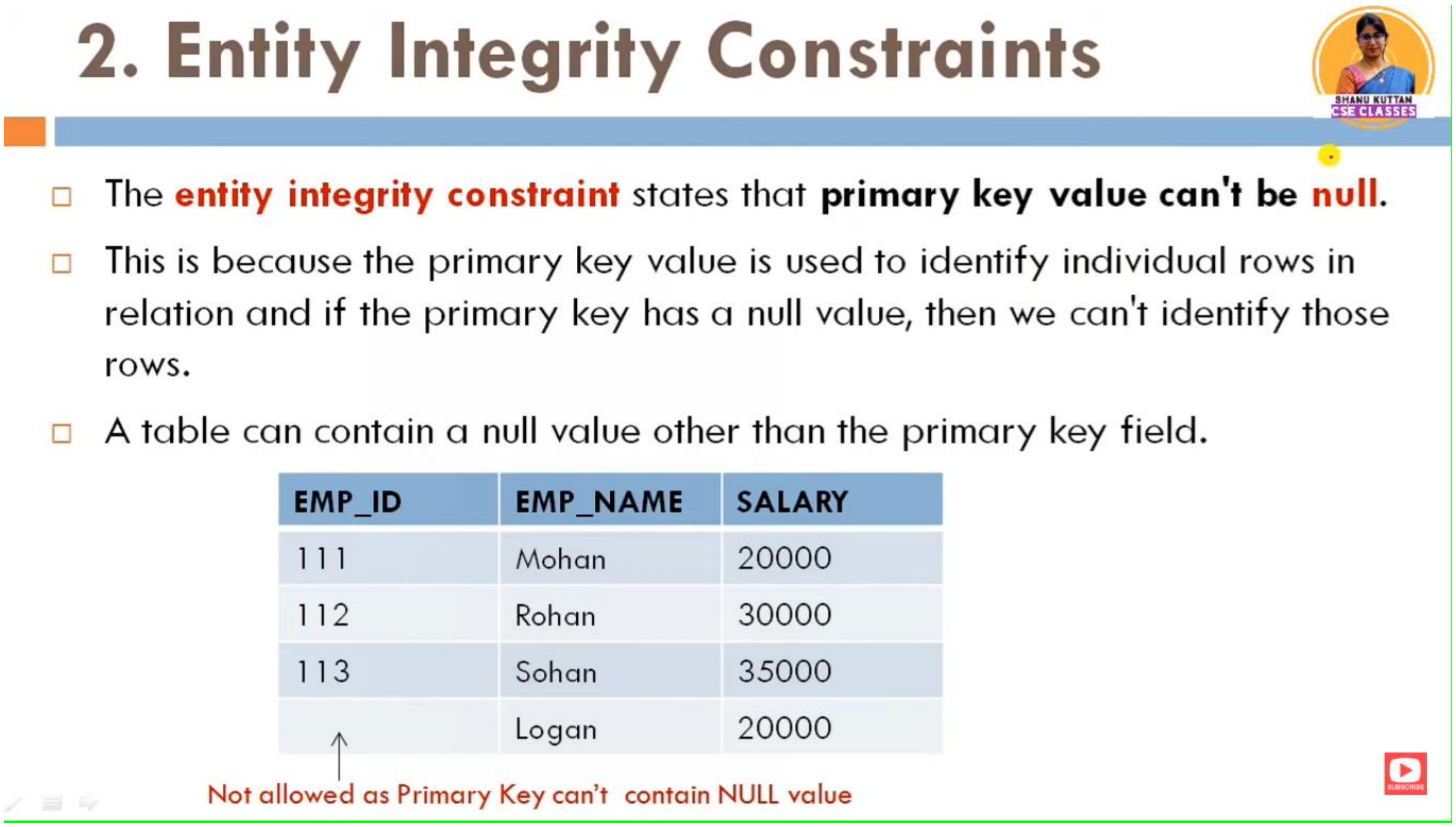


1. Domain Constraints

- **Domain constraints** defines the domain or the valid set of values for an attribute.
- The **data type of domain** includes string, character, integer, time, date, currency, etc. *The value of the attribute must be available in the corresponding domain.*

STUDENT_ID	NAME	SEMESTER	AGE
101	Manish	1st	18
102	Rohit	3rd	19
103	Badal	5th	20
104	Amit	7th	A

Not allowed. Because AGE is an integer value

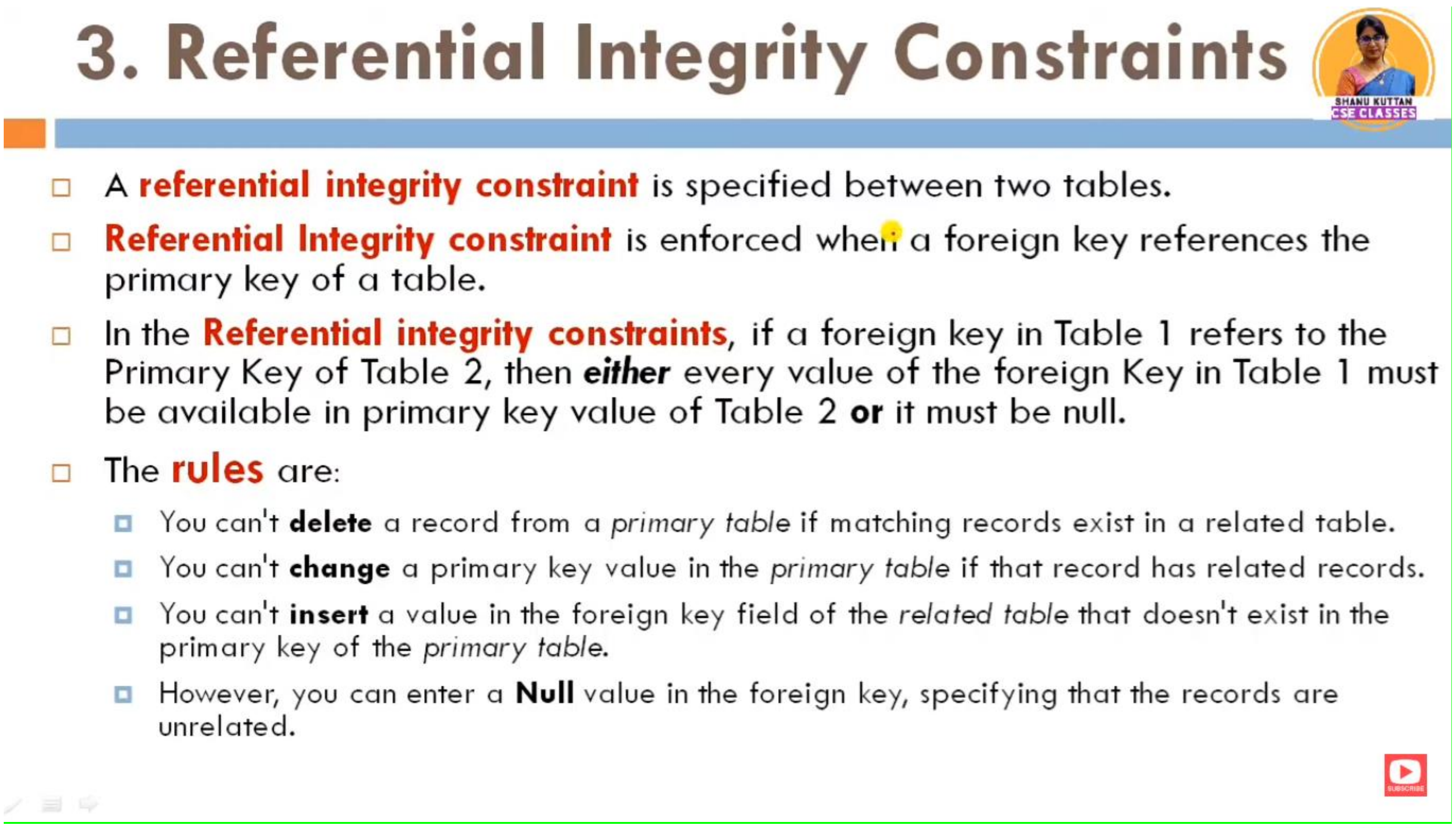


2. Entity Integrity Constraints

- The **entity integrity constraint** states that **primary key value can't be null**.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

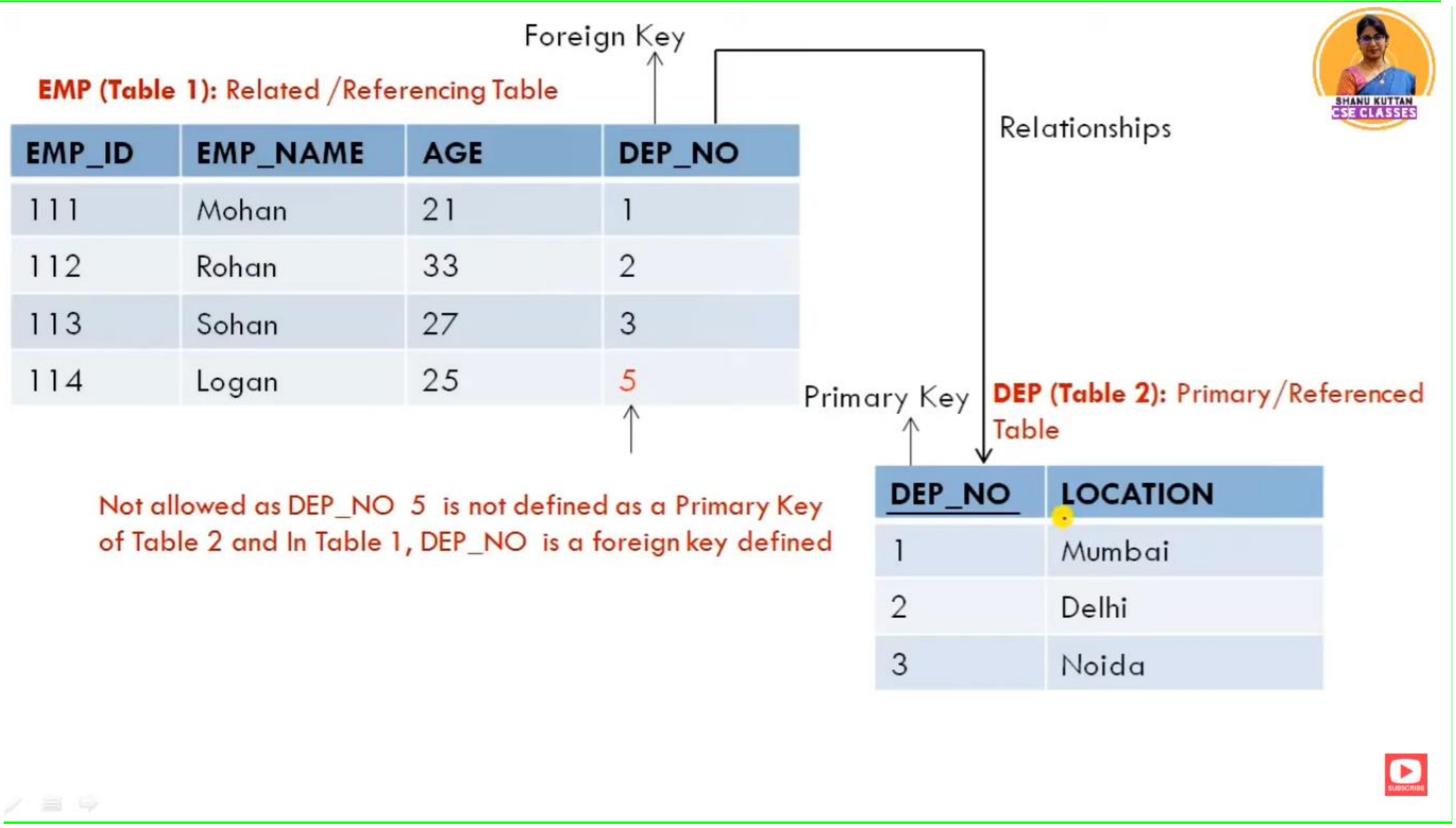
EMP_ID	EMP_NAME	SALARY
111	Mohan	20000
112	Rohan	30000
113	Sohan	35000
	Logan	20000

Not allowed as Primary Key can't contain NULL value



3. Referential Integrity Constraints

- A **referential integrity constraint** is specified between two tables.
- **Referential Integrity constraint** is enforced when a foreign key references the primary key of a table.
- In the **Referential integrity constraints**, if a foreign key in Table 1 refers to the Primary Key of Table 2, then ***either*** every value of the foreign Key in Table 1 must be available in primary key value of Table 2 **or** it must be null.
- The **rules** are:
  - You can't **delete** a record from a *primary table* if matching records exist in a related table.
  - You can't **change** a primary key value in the *primary table* if that record has related records.
  - You can't **insert** a value in the foreign key field of the *related table* that doesn't exist in the primary key of the *primary table*.
  - However, you can enter a **Null** value in the foreign key, specifying that the records are unrelated.



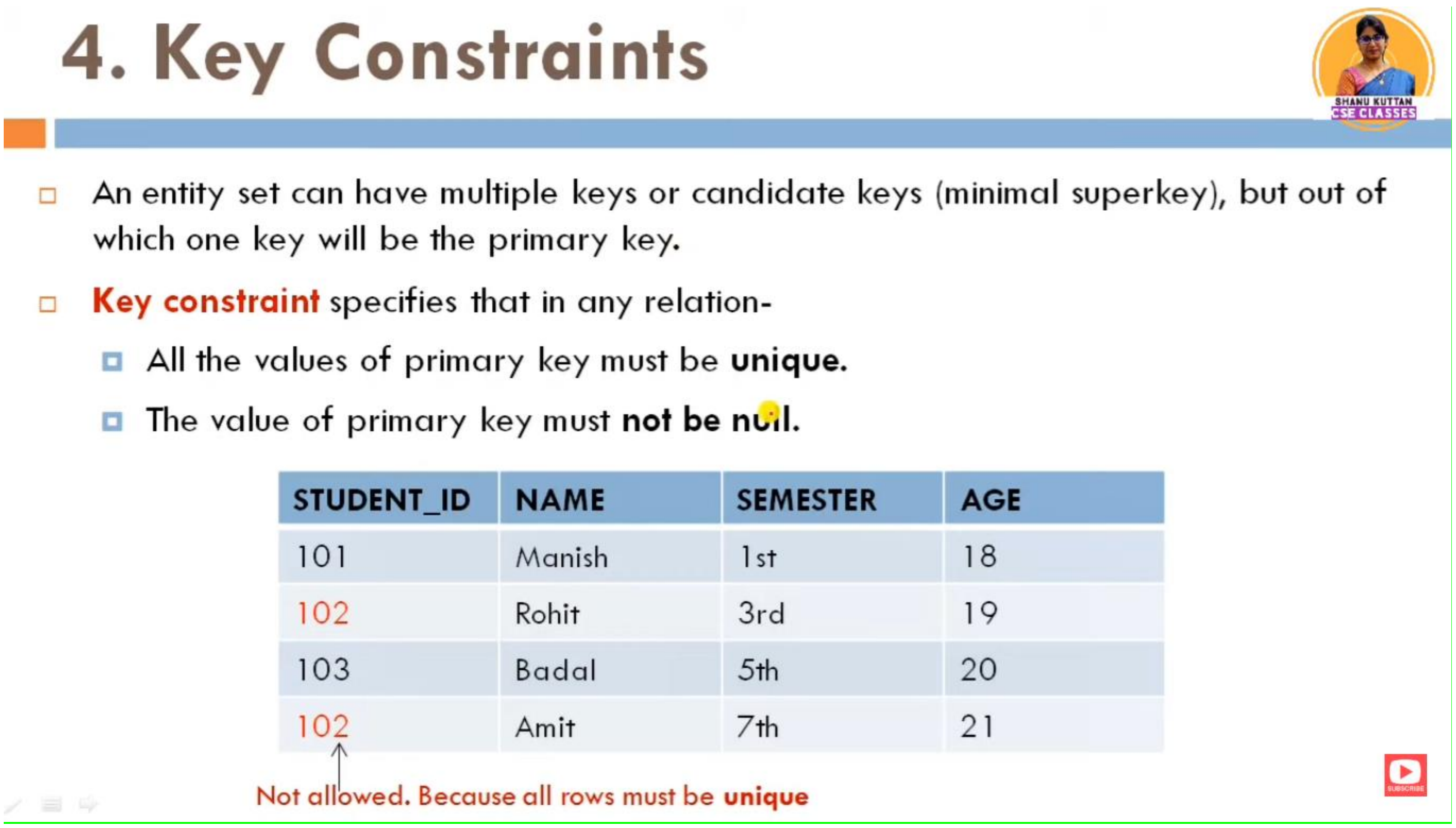
EMP (Table 1): Related /Referencing Table

EMP_ID	EMP_NAME	AGE	DEP_NO
111	Mohan	21	1
112	Rohan	33	2
113	Sohan	27	3
114	Logan	25	5

Not allowed as DEP_NO 5 is not defined as a Primary Key of Table 2 and In Table 1, DEP_NO is a foreign key defined

DEP (Table 2): Primary/Referenced Table

DEP_NO	LOCATION
1	Mumbai
2	Delhi
3	Noida



4. Key Constraints

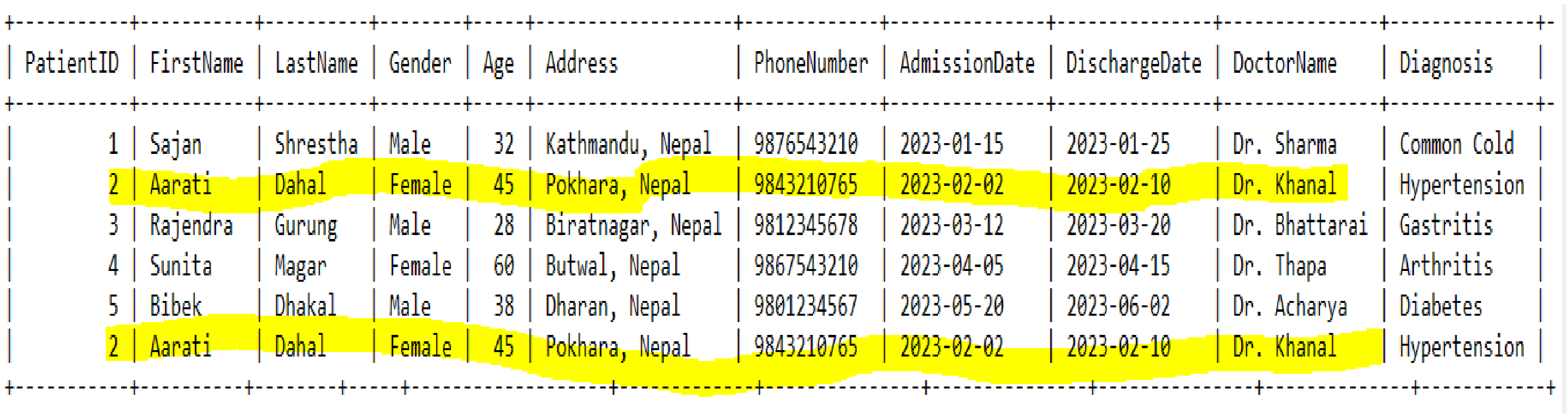
- An entity set can have multiple keys or candidate keys (minimal superkey), but out of which one key will be the primary key.
- **Key constraint** specifies that in any relation-
  - All the values of primary key must be **unique.**
  - The value of primary key must **not be null.**

STUDENT_ID	NAME	SEMESTER	AGE
101	Manish	1st	18
102	Rohit	3rd	19
103	Badal	5th	20
102	Amit	7th	21

Not allowed. Because all rows must be **unique**

3.1. Importance of Normalization

Normalization is a process in a Database Management System (DBMS) used to organize data in a database by *reducing redundancy and improving data integrity*. It involves dividing large tables into smaller ones and defining relationships between them to minimize duplicate data.



```
+-----------+-----------+----------+--------+-----+------------------+------------+----------------+---------------+--------------+-------------+
| PatientID | FirstName | LastName | Gender | Age | Address          | PhoneNumber | AdmissionDate | DischargeDate | DoctorName   | Diagnosis   |
+-----------+-----------+----------+--------+-----+------------------+------------+----------------+---------------+--------------+-------------+
|         1 | Sajan     | Shrestha | Male   |  32 | Kathmandu, Nepal | 9876543210 | 2023-01-15     | 2023-01-25    | Dr. Sharma   | Common Cold |
|         2 | Aarati    | Dahal    | Female |  45 | Pokhara, Nepal   | 9843210765 | 2023-02-02     | 2023-02-10    | Dr. Khanal   | Hypertension |
|         3 | Rajendra  | Gurung   | Male   |  28 | Biratnagar, Nepal | 9812345678 | 2023-03-12    | 2023-03-20    | Dr. Bhattarai | Gastritis  |
|         4 | Sunita    | Magar    | Female |  60 | Butwal, Nepal    | 9867543210 | 2023-04-05     | 2023-04-15    | Dr. Thapa    | Arthritis   |
|         5 | Bibek     | Dhakal   | Male   |  38 | Dharan, Nepal    | 9801234567 | 2023-05-20     | 2023-06-02    | Dr. Acharya  | Diabetes    |
|         2 | Aarati    | Dahal    | Female |  45 | Pokhara, Nepal   | 9843210765 | 2023-02-02     | 2023-02-10    | Dr. Khanal   | Hypertension |
+-----------+-----------+----------+--------+-----+------------------+------------+----------------+---------------+--------------+-------------+
```

Objectives of Normalization

1. Eliminate redundant data.
2. Ensure data dependency is logical.
3. Reduce the chances of data anomalies (Insertion, Update, Deletion anomalies).

- Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like ***Insertion, Update and Deletion Anomalies.***

- Normalization rules divides larger tables into smaller tables and links them using relationships.

- The purpose of Normalization in SQL is to ***eliminate redundant (repetitive) data and ensure datais stored logically.***

- A large database defined as a single relation may result in data duplication. This repetition of data may result in:

  - Making relations very large.
  - It isn't easy to maintain and update data as it would involve searching many records in relation.
  - Wastage and poor utilization of disk space and resources.
  - The likelihood of errors and inconsistencies increases.

- So to handle these problems, ***we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations*** that are satisfy desirable properties.

- Normalization is a process of ***decomposing the relations into relations with fewer attributes***.

```
| PatientID | FirstName | LastName | Gender | Age | Address | PhoneNumber |

| PatientID | DoctorName | Diagnosis

| PatientID || RoomNumber |  DischargeDate | AdmissionDate
```

Why Do We Need Normalization?

- As we have discussed above, normalization is used to reduce data redundancy.
- It provides a method to remove the following anomalies from the database and bring it to a more consistent state.

- Below are the key reasons why normalization is needed:

1. Eliminate Data Redundancy

- **Problem**: Redundant data leads to wastage of storage space and the risk of inconsistencies.
- **Solution**: Normalization removes duplicate data by splitting large tables into smaller, related tables.
- **Example**: Instead of repeating instructor details in every row of a course table, store it in a separate Instructor table.

2. Avoid Data Anomalies

Normalization resolves three types of anomalies:

1. **Insertion Anomaly**:

- Without normalization, adding new data might require entering irrelevant data.
- Example: Adding a new course without students creates NULL values in a non-normalized table.

2. **Update Anomaly**:
    - Changes in one place must be updated everywhere, risking inconsistencies.
    - Example: Updating an instructor's contact information in multiple rows.

3. **Deletion Anomaly**:
    - Deleting specific data might accidentally remove important related data.
    - Example: Deleting a student's enrollment removes the entire course record.

3. Improve Data Integrity

- Normalization ensures data is logically stored and adheres to predefined rules (constraints).
- Dependencies like foreign keys ensure relationships remain valid across tables.

4. Enhance Query Performance

- Although normalization might involve more joins between tables, it ensures data retrieval is more consistent and meaningful, improving query accuracy.

5. Maintain Scalability

- A normalized database can easily accommodate changes like adding new fields, relationships, or tables without disrupting the existing structure.

6. Enforce Business Rules

- Normalization reflects real-world relationships, making it easier to enforce business rules within the database.

- Example: Ensuring that every enrolled student must belong to a valid course.

3.2. Functional Dependencies –

definition,

trivial and non-trivial FD,

closure of FD set,

closure of attributes

https://github.com/sanjeevlcc/notes_2081/blob/main/DBMS_BIM_BSCIT_BCA/notes/Normalization/Functional%20Dependency%20and%20Normalisation.txt

3.4. Normal forms (1NF, 2NF, 3NF, BCNF)

- The inventor of the relational model Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form.
- Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

Database Normal Forms

1NF (First Normal Form)

2NF (Second Normal Form)

3NF (Third Normal Form)

BCNF (Boyce-Codd Normal Form)

4NF (Fourth Normal Form)

5NF (Fifth Normal Form)

6NF (Sixth Normal Form)

- The Theory of Data Normalization in MySQL server is still being developed further. For example, there are discussions even on 6th Normal Form.
- However, in most practical applications, normalization achieves its best in 3rd Normal Form.

Summary:

- **1NF**: Atomicity (no repeating groups).
- **2NF**: No partial dependencies (all non-key attributes depend on the whole key).
- **3NF**: No transitive dependencies (non-key attributes depend only on the key).
- **BCNF**: Every determinant is a superkey.
- **4NF**: No multi-valued dependencies.
- **5NF**: Decomposition without loss of information.

Q/A

Fill in the Blanks (20 Questions)

Multiple Choice Questions (MCQ) (20 Questions)

Short Questions (20 Questions)

Comprehensive Questions (20 Questions)

ANSWER of (MCQ)

ANSWER of Short Questions