1. Basic Structure of a Relation

- **Relation**: In the relational model, a relation is represented as a table. The table consists of:
    - **Attributes** (columns): These define the type of data that can be stored. Each attribute has a name and a domain (set of allowed values).
    - **Tuples** (rows): Each row represents a single data record. The number of rows is called the **cardinality** of the relation, and the number of columns is called the **degree** of the relation.
- Example: In a table called Students, the attributes could be StudentID, Name, Age, and the tuples represent individual students:

StudentID	Name	Age
1	Ritesh	20
2	Priya	19
3	Aman	22

2. Domain

- A **domain** is the set of all possible values an attribute can take. For example:
    - The domain of the Age attribute might be integers between 0 and 120.
    - The domain of Name might be strings up to 50 characters.
- Every attribute in a relation is associated with a domain to ensure consistency and avoid invalid data.

3. Keys

- **Primary Key**: This is a unique identifier for each tuple in a relation. It ensures that no two rows have the same value for the primary key.
    - Example: In the Students table, StudentID is the primary key, meaning every student must have a unique StudentID. No two rows can have the same ID, ensuring uniqueness.

- **Candidate Key**: A candidate key is any set of attributes that could serve as a primary key. Out of all candidate keys, one is chosen as the primary key.
    - Example: StudentID and maybe an email address could be candidate keys because both uniquely identify a student.
- **Foreign Key**: This key is used to create a relationship between two tables. A foreign key in one table refers to the primary key in another table. It helps establish relationships and maintain referential integrity.
    - Example: A Grades table might have a StudentID column as a foreign key, linking it to the primary key StudentID in the Students table.

4. Relational Database Constraints

These constraints enforce rules on the data to maintain accuracy and consistency:

- **Domain Constraint**: This ensures that values for an attribute must come from the specified domain.
    - Example: If the domain of the Age attribute is restricted to integers from 0 to 120, entering 200 as an age would violate the domain constraint.
- **Key Constraint**: This constraint ensures that no two tuples (rows) have the same value for the primary key.
    - Example: In the Students table, no two students can have the same StudentID.
- **Entity Integrity**: This rule ensures that no primary key can have a NULL value, which means every tuple must have a valid, non-null identifier.
    - Example: You cannot have a student record without a StudentID in the Students table.
- **Referential Integrity**: This ensures that a foreign key must match an existing primary key value or be NULL. It enforces consistency between related tables.
    - Example: In the Grades table, if a StudentID references the Students table, every StudentID in Grades must exist in the Students table. If you try to insert a StudentID in Grades that doesn't exist in Students, it will violate referential integrity.