**Types of constraints**

* NOT NULL
* UNIQUE
* DEFAULT
* CHECK
* Key Constraints – PRIMARY KEY, FOREIGN KEY
* Domain constraints
* Mapping constraints

**NOT NULL:**

NOT NULL constraint makes sure that a column does not hold NULL value. When we don’t provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

Example:

CREATE TABLE STUDENT(

ROLL\_NO INT **NOT NULL**,

STU\_NAME VARCHAR (35) **NOT NULL**,

STU\_AGE INT **NOT NULL**,

STU\_ADDRESS VARCHAR (235),

PRIMARY KEY (ROLL\_NO)

);

#### UNIQUE:

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

CREATE TABLE STUDENT(

ROLL\_NO INT NOT NULL,

STU\_NAME VARCHAR (35) NOT NULL **UNIQUE**,

STU\_AGE INT NOT NULL,

STU\_ADDRESS VARCHAR (35) **UNIQUE**,

PRIMARY KEY (ROLL\_NO)

);

#### DEFAULT:

The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.

CREATE TABLE STUDENT(

ROLL\_NO   INT  NOT NULL,

STU\_NAME VARCHAR (35) NOT NULL,

STU\_AGE INT NOT NULL,

EXAM\_FEE INT  **DEFAULT** 10000,

STU\_ADDRESS VARCHAR (35) ,

PRIMARY KEY (ROLL\_NO)

);

#### CHECK:

This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

CREATE TABLE STUDENT(

ROLL\_NO   INT  NOT NULL CHECK(ROLL\_NO >1000) ,

STU\_NAME VARCHAR (35)  NOT NULL,

STU\_AGE INT  NOT NULL,

EXAM\_FEE INT DEFAULT 10000,

STU\_ADDRESS VARCHAR (35) ,

PRIMARY KEY (ROLL\_NO)

);

In the above example we have set the check constraint on ROLL\_NO column of STUDENT table. Now, the ROLL\_NO field must have the value greater than 1000.

## Key constraints:

#### PRIMARY KEY:

[Primary key](https://beginnersbook.com/2015/04/primary-key-in-dbms/) uniquely identifies each record in a table. It must have unique values and cannot contain nulls. In the below example the ROLL\_NO field is marked as primary key, that means the ROLL\_NO field cannot have duplicate and null values.

CREATE TABLE STUDENT(

ROLL\_NO   INT  NOT NULL,

STU\_NAME VARCHAR (35)  NOT NULL UNIQUE,

STU\_AGE INT NOT NULL,

STU\_ADDRESS VARCHAR (35) UNIQUE,

**PRIMARY KEY** (ROLL\_NO)

);

#### FOREIGN KEY:

Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.  
Read more about it [here](https://beginnersbook.com/2015/04/foreign-key-in-dbms/).

#### Domain constraints:

Each table has certain set of columns and each column allows a same type of data, based on its data type. The column does not accept values of any other data type.  
[Domain constraints](https://beginnersbook.com/2015/04/domain-constraints-in-dbms/) are **user defined data type** and we can define them like this:

Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT)

**Mapping Cardinality**:  
**One to One**: An entity of entity-set A can be associated with at most one entity of entity-set B and an entity in entity-set B can be associated with at most one entity of entity-set A.

**One to Many**: An entity of entity-set A can be associated with any number of entities of entity-set B and an entity in entity-set B can be associated with at most one entity of entity-set A.

**Many to One**: An entity of entity-set A can be associated with at most one entity of entity-set B and an entity in entity-set B can be associated with any number of entities of entity-set A.

**Many to Many**: An entity of entity-set A can be associated with any number of entities of entity-set B and an entity in entity-set B can be associated with any number of entities of entity-set A.

We can have these constraints in place while creating tables in database.

**Example**:

CREATE TABLE Customer (

customer\_id int PRIMARY KEY NOT NULL,

first\_name varchar(20),

last\_name varchar(20)

);

CREATE TABLE Order (

order\_id int PRIMARY KEY NOT NULL,

customer\_id int,

order\_details varchar(50),

constraint fk\_Customers foreign key (customer\_id)

references dbo.Customer

);

Assuming, that a customer orders more than once, the above relation represents **one to many**relation. Similarly we can achieve other mapping constraints based on the requirements.