SQL | Constraints

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

- **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
- **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
- **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.

- **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in a another table. And this constraint is used to specify a field as Foreign key.
- **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
- **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

```
CREATE TABLE sample_table
(
column1 data_type(size) constraint_name,
column2 data_type(size) constraint_name,
column3 data_type(size) constraint_name,
....
);
```

sample_table: Name of the table to be created.

data_type: Type of data that can be stored in the field.

constraint_name: Name of the constraint. for example- NOT NULL, UNIQUE, PRIMARY KEY etc.

1. PRIMARY KEY Constraint

A PRIMARY KEY constraint uniquely identifies each record in a table. It ensures that the column (or a combination of columns) has unique values and that no NULL values are allowed.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100)
);
```

2. FOREIGN KEY Constraint

A FOREIGN KEY constraint is used to link two tables together. It ensures that the value in a column (or group of columns) in one table matches values in a column in another table.

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);

CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

3. UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are unique across the table. Unlike PRIMARY KEY, a table can have multiple UNIQUE constraints, and NULL values are allowed.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    email VARCHAR(100) UNIQUE,
    first_name VARCHAR(50),
    last_name VARCHAR(50)
);
```

Note:-

email is unique, so no two employees can have the same email address.

4. NOT NULL Constraint

The NOT NULL constraint ensures that a column cannot have a NULL value. It guarantees that the field must always contain a value.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100)
);
```

Note:- first_name and last_name are marked as NOT NULL, so they cannot be left empty when inserting data.

5. CHECK Constraint

The CHECK constraint ensures that all values in a column satisfy a specific condition.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    age INT CHECK (age >= 18)
);
```

Note:- age must be 18 or older; otherwise, the insertion/update will fail.

6. DEFAULT Constraint

The DEFAULT constraint provides a default value for a column when no value is specified.

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    status VARCHAR(20) DEFAULT 'Active'
);
```

Note:- If no value is provided for status during an insertion, it will automatically be set to 'Active'.

7. INDEX Constraint

An INDEX is not exactly a constraint but is often used to enhance performance by allowing quick access to the data in a database.

```
CREATE INDEX idx_lastname ON
employees(last_name);
```

Note:- This creates an index on the last_name column, speeding up queries that filter by last_name.