SQL CONSTRAINTS

WHAT ARE SQL CONSTRAINTS?

- SQL constraints are a **set of rules** implemented on tables in relational databases.
- Constraints are **used to limit the type of data** that can go into a table.
- This ensures the **accuracy** and **reliability** of the data in the table.
- If there is any violation between the constraint and the data action, the action is aborted.

SQL CONSTRAINTS

- Constraints can be column level or table level. The **column-level constraints** can apply only to one column where as **table-level constraints** are applied to the entire table.
- Constraints are declared at the time of table creation.
- Constraints which are commonly used in SQL:
  - Not null
  - Check
  - Default
  - Unique
  - Primary Key
  - Foreign Key

1) NOT NULL CONSTRAINT

- By default, a **COLUMN** can hold **NULL values**. If you do not want a column to have a NULL value, use the **NOT NULL** constraint.
- It restricts a column from having a NULL value.

Note: This constraint cannot be defined at the table level.

Syntax :

```
CREATE TABLE table_name
(
  column_name1  DATATYPE NOT NULL,
  column_name2  DATATYPE NOT NULL,
  column_name3 DATATYPE
);
```

Example :

```
CREATE TABLE student_table
(
student_id INT NOT NULL,
student_name VARCHAR(255) NOT NULL,
student_age INT
);
```

2) CHECK CONSTRAINT

- A CHECK constraint checks for a **specific condition** before inserting data into a table.
- If the data passes all the Check constraints then the data will be inserted into the table otherwise the data for insertion will be discarded.

Syntax :

```
CREATE TABLE table_name
(
  column_name1  DATATYPE NOT NULL,
  column_name2  DATATYPE,
  column_name3 DATATYPE CHECK ()
);
```

Example :

```
CREATE TABLE student_table
(
  student_id INT NOT NULL,
  student_name VARCHAR(255) ,
  Student_age INT CHECK (Student_age >18)
);
```

2) CHECK CONSTRAINT...

- To apply **multiple** conditions

Example :

```
CREATE TABLE student_table
(
  student_id INT NOT NULL,
  student_name VARCHAR(255) ,
  student_age INT,
  CHECK (student_age >18 AND student_id >2 )
);
```

3) DEFAULT CONSTRAINT

- The DEFAULT constraint is used **to set a default value** for a column.
- The default value will be added to all new records **if no other value** is specified.

Syntax :

```
CREATE TABLE table_name
(
  column_name1  DATATYPE NOT NULL,
  column_name2  DATATYPE,
  column_name3 DATATYPE DEFAULT
);
```

Example :

```
CREATE TABLE student_table
(
  student_id INT NOT NULL,
  student_name VARCHAR(255) ,
  Student_age INT DEFAULT 18
);
```

3) DEFAULT CONSTRAINT…

- INSERTING VALUES in a TABLE:

Syntax :

INSERT INTO table_name
VALUES
(value1, value2, value3) ,
(value1.1, value2.1, value3.1),
(value1.2, value2.2, value3.2);

Example :

INSERT INTO student_table
VALUES
(1, 'Raju', 20),
(2, 'Rancho', 19),
(3, 'Farhan', DEFAULT);

4) UNIQUE CONSTRAINT

- UNIQUE constraints enforce the **uniqueness of values** in a column or a group of columns in a table.
- A UNIQUE constraint can be either a column constraint or a table constraint.

Syntax :

```
CREATE TABLE table_name
(
  column_name1  DATATYPE UNIQUE,
  column_name2  DATATYPE,
  column_name3 DATATYPE
);
```

Example :

```
CREATE TABLE student_table
(
  student_id INT UNIQUE,
  student_name VARCHAR(255) ,
  Student_age INT
);
```

4) UNIQUE CONSTRAINT...

- Applying UNIQUE constraints on multiple columns together.

Example :

```
CREATE TABLE student_table
(
  student_id INT,
  student_name VARCHAR(255) ,
  Student_age INT,
  UNIQUE ( student_id, student_age)
);
```

5) PRIMARY KEY CONSTRAINT

- A Primary key **uniquely identifies** each row in a table.
- Primary Key must contain **UNIQUE** values, and cannot contain **NULL** values.
- A table can have **only ONE** primary key; and in the table, this primary key can consist of single or multiple columns (fields).
- If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- Note that MySQL implicitly adds a NOT NULL constraint to primary key columns.

5) PRIMARY KEY CONSTRAINT…

- Defining Primary Key Constraints:
- If the primary key has **one column**, you can use the PRIMARY KEY constraint as a column constraint:

Syntax :

```
CREATE TABLE table_name
(
  column_name1  DATATYPE PRIMARY KEY,
  column_name2  DATATYPE,
  column_name3 DATATYPE
);
```

Example :

```
CREATE TABLE student_table
(
  student_id INT PRIMARY KEY,
  student_name VARCHAR(255),
  Student_age INT
);
```

5) PRIMARY KEY CONSTRAINT…

- Defining Primary Key Constraints:
- When the primary key **has more than one column**, you must use the PRIMARY KEY constraint as a table constraint.

Syntax :

```
CREATE TABLE table_name
(
  column_name1  DATATYPE,
  column_name2  DATATYPE,
  column_name3 DATATYPE,
  PRIMARY KEY (column_list)
);
```

Example :

```
CREATE TABLE student_table
(
  student_id INT,
  student_name VARCHAR(255),
  Student_age INT,
  PRIMARY KEY(student_id, student_age)
);
```

6) FOREIGN KEY CONSTRAINT

- A Foreign Key is a field in a database table that is a Primary key in another table.
- A Foreign key **creates a relation between two tables**.
- The first table contains a primary key and the second table contains a foreign key.
- The foreign key places constraints on data in the related tables, which allows MySQL to maintain referential integrity.

6) FOREIGN KEY CONSTRAINT

- The customer_table is called the ***parent table*** **or** ***referenced table***, and the orders_table is known as the ***child table*** **or** ***referencing table***.
- A table can have **more than one foreign key** where each foreign key references to a primary key of the different parent tables.

Table 1:

```
CREATE TABLE customer_table
(
  customer_id INT PRIMARY KEY,
  customer_name VARCHAR(255),
  city VARCHAR(255),
  state VARCHAR(255)
);
```

Table 2 :

```
CREATE TABLE order_table
(
  order_id INT PRIMARY KEY,
  order_number INT,
  customer_id INT,
  FOREIGN KEY(customer_id) REFERENCES customer_table(customer_id)
);
```

Thank You