CONSTRAINTS

SQL Constraints are rules used to limit the type of data that can go into a table and to maintain the accuracy and integrity of the data inside table. In a way they define certain properties that data in a database must comply with.

NOT NULL

NOT NULL constraint restricts a column from having a NULL value (no value). Once NOT NULL constraint is applied to a column, you cannot pass a NULL value to that column. It enforces a column to contain a proper value.

Syntax:

```
CREATE TABLE table_name
(
       column_name1 data_type(size) NOT NULL,
       column_name2 data_type(size),
       column_name3 data_type(size),
       ....
);
```

Example 1.1: Create a table employee1 with id of type integer and name, a variable character field. Now insert a row into this table only giving value to name field. Display the resultant table.

Query:

```
CREATE TABLE employee1
(
    id INTEGER,
    name VARCHAR(20)
);

INSERT INTO employee1 (name)
VALUES ('Rajesh');

INSERT INTO employee1 (id, name)
VALUES (2, NULL);

SELECT * FROM employee1;
```

Result:

```
+------+--------+
| id   | name   |
+------+--------+
| NULL | Rajesh |
|    2 | NULL   |
+------+--------+
2 rows in set (0.01 sec)
```

Explanation:
Since, we didn't provide any value to id field in our first INSERT query our row got NULL value for the field id.

Example 1.2: Create a table employee2 with id of type integer and name, a variable character field. Field id cannot be null. Now try to insert a row into this table only giving value to name field.

Query:

```
CREATE TABLE employee2
(
   id INTEGER NOT NULL,
   name VARCHAR(20)
);

INSERT INTO employee2 (name)
VALUES ('Mahesh');
```

Result:

```
SQL Error line 7:
Column 'id' cannot be NULL
```

Explanation:
We have applied NOT NULL constraint to our field id. Now we cannot write any SQL query that tries to provide NULL value to id. Hence, the INSERT query above generated error.

DEFAULT

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

Syntax:

```
CREATE TABLE table_name
(
      column_name1 data_type(size) DEFAULT value,
      column_name2 data_type(size),
      column_name3 data_type(size),
      ....
);
```

Example 2.1: Without default

Query:

```
CREATE TABLE employee3
(
   id INTEGER,
   name VARCHAR(20)
);

INSERT INTO employee3 (name)
VALUES ('Vishal');

SELECT * FROM employee3;
```

Result:

```
+------+--------+
| id   | name   |
+------+--------+
| NULL | Vishal |
+------+--------+
1 row in set (0.00 sec)
```

Explanation: Without default we get NULL values if we skip column value in INSERT query.
Example 2.2: With default

Query:

```
CREATE TABLE employee4
(
   id INTEGER DEFAULT 0,
   name VARCHAR(20)
);

INSERT INTO employee4 (name)
VALUES ('Vishal');

INSERT INTO employee4 (name)
VALUES ('Priya');

SELECT * FROM employee4;
```

Result:

```
+------+--------+
| id   | name   |
+------+--------+
|    0 | Vishal |
|    0 | Priya  |
+------+--------+
2 rows in set (0.00 sec)
```

Explanation:
With DEFAULT value if we do not provide any value for the column the DEFAULT value will be used. We can use DEFAULT constraint with NOT NULL constraint. It can avoid SQL errors that are generated if we do not provide any value to a column with NOT NULL constraint applied to it.

CHECK

CHECK constraint is used to restrict the value of a column between a specific range. It performs check on the values, before storing them into the database. It’s like condition checking before saving data into a column.

Note: CHECK constraint is not supported by MySQL.

Syntax:

```
CREATE TABLE table_name
(
      column_name1 data_type(size),
      column_name2 data_type(size),
      column_name3 data_type(size),
      CHECK (column_name condition)
      ....
);
```

Example 3: Working with CHECK constraint

Query:

```
1  CREATE TABLE employee5
2  (
3      id INTEGER,
4      name VARCHAR(20),
5      CHECK (id > 0)
6  );
7
```

Result:

```
SQL Error line 8
```

```
INSERT INTO employee1 (id, name)
VALUES (-1, 'Hari');
```

UNIQUE

The UNIQUE Constraint prevents two records from having identical values in a particular column.

Syntax:

```
CREATE TABLE table_name
(
       column_name1 data_type(size) UNIQUE,
       column_name2 data_type(size),
       column_name3 data_type(size),
       ....
);
```

Example 4: Working with UNIQUE constraint

Query:

```
1  CREATE TABLE employee6
2  (
3     id INTEGER UNIQUE,
4     name VARCHAR(20)
5  );
6
7  INSERT INTO employee6 (id, name)
8  VALUES (1,'Priya');
9
10 INSERT INTO employee6 (name)
11 VALUES (1, 'Neha');
```

Result:

```
SQL Error line 10:
Duplicate entry '1' for key 'id'
```

PRIMARY KEY

Primary key constraint uniquely identifies each record in a database. A Primary key must contain unique value and it cannot contain null value. Usually Primary key is used to index the data inside the table.
To define a field as primary key, following conditions have to be met:

a) No two rows can have the same primary key value.
b) Every row must have a primary key value.
c) The primary key field cannot be null.
d) Value in a primary key column can never be modified or updated, if any foreign key refers to that primary key.

Syntax:

```
CREATE TABLE table_name
(
       column_name1 data_type(size),
       column_name2 data_type(size),
       column_name3 data_type(size),
       ....,
        PRIMARY KEY (column_name)
);
```

Example 5: Working with UNIQUE constraint

Query:

```
1  CREATE TABLE employee7
2  (
3     id INTEGER,
4     name VARCHAR(20),
5     PRIMARY KEY (id)
6  );
7
8  INSERT INTO employee7 (id)
9  VALUES (1);
10
11 SELECT * FROM employee7;
12
13 INSERT INTO employee7 (name)
14 VALUES ('Neha');
15
16 INSERT INTO employee7 (id, name)
17 VALUES (1, 'Priya');
```

Result:

```
Query OK, 1 row affected (0.00 sec)

+----+------+
| id | name |
+----+------+
|  1 | NULL |
+----+------+
1 row in set (0.00 sec)

SQL Error line 13:
Column 'id' cannot be NULL

SQL Error line 16:
Duplicate entry '1' for key 'PRIMARY'
```

Difference between primary key and unique key constraint:

S. No.	Primary Key	Unique Key
1	It doesn't allow NULL values.	Allows NULL value. But only one NULL value.
2	By default it adds a clustered index.	By default it adds a UNIQUE non-clustered index.
3	A table can have only one PRIMARY KEY Column.	A table can have more than one UNIQUE Key Column.

FOREIGN KEY

FOREIGN KEY is used to relate two tables. A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

Example 6.1: Create a table employee9 with id of type integer and name, a variable character field and id as PRIMARY KEY. Now insert two rows into this table. Display the resultant table. Create another table orders with o_id of type integer as PRIMARY KEY and name, a variable character field and id of type integer as FOREIGN KEY.

Query:

```
CREATE TABLE employee9
(
   id INTEGER NOT NULL,
   name VARCHAR(20)
   PRIMARY KEY(id)
);

CREATE TABLE orders
(
   o_id INTEGER NOT NULL,
   date VARCHAR(20) NOT NULL,
   id INTEGER,
   PRIMARY KEY(o_id),
   FOREIGN KEY(id)
   REFERENCES employee9(id)
);

INSERT INTO employee9 (id, name)
VALUES (1, 'Priya');

INSERT INTO employee9 (id, name)
VALUES (2, 'Neha');

INSERT INTO orders (o_id, date, id)
VALUES (10, 'March', 1);

INSERT INTO orders (o_id, date, id)
VALUES (11,'Feb',2);

SELECT * FROM employee9;
SELECT * FROM orders;
```

Result:

```
Query OK, 0 rows affected (0.47
sec)

+----+-------+
| id | name  |
+----+-------+
|  1 | priya |
|  2 | Neha  |
+----+-------+
2 rows in set (0.00 sec)

+------+-------+------+
| o_id | date  | id   |
+------+-------+------+
|   10 | March |    1 |
|   11 | Feb   |    2 |
+------+-------+------+
2 rows in set (0.02 sec)
```

Example 6.2: Continuing from above tables created delete from employee9 one row whose id is present in orders table. Display the output. Inserting a row in orders whose id is not present in employee1.Display the output.

Query:

Result:

```sql
DELETE FROM employee9 WHERE id = 1;

INSERT INTO orders (o_id, date, id)
VALUES (13, 'Feb', 4);
```

```
SQL Error line 1:
Cannot delete or update a parent
row: a foreign key constraint fails
(`sheet`.`orders`, CONSTRAINT
`orders_ibfk_1` FOREIGN KEY (`id`)
REFERENCES `employee9` (`id`))

SQL Error line 3:
Cannot add or update a child row: a
foreign key constraint fails
(`sheet`.`orders`, CONSTRAINT
`orders_ibfk_1` FOREIGN KEY (`id`)
REFERENCES `employee9` (`id`))
```

Problem 1	**In each of the following problems well refer the following table structure: table name: employee9 columns: e_id of type integer and name of variable character type**
Problem 1.2	**Create the table structure. Insert two rows in the table with e_id only and display the output. Write your output.**

Problem 1.3	**What is primary key?
	Make e_id column primary key and write output for following queries:
	➢ INSERT INTO employee9 (e_id, name) VALUES (1, 'Shreya');
	➢ INSERT INTO employee9 (e_id, name) VALUES (2, 'Shreya');
	➢ INSERT INTO employee9 (name) VALUES ('Shriya');
	➢ INSERT INTO employee9 (e_id, name) VALUES (4, 'Shashank');
	➢ SELECT * FROM employee9;**

Problem 1.4	**Write a query for creating table such that it will not allow any duplicate value for any column.**
Problem 1.5	**Create one more table events with id, winner_name and e_id . Write a query for linking it with employee9 table created. Insert values ('a','First',1) and ('b','Second',2)**
Problem 1.6	**Continuing Problem 1.5. What will happen if: ➢ INSERT INTO events VALUES('c', 'Third', 3); ➢ Delete from employee9 having e_id as 1. ➢ Delete from events having e_id as 1. ➢ Delete from employee9 having e_id as 1.**
Problem 1.7	**What is the difference between DEFAULT and NOT NULL constraint? Give examples illustrating the difference.**