5. SQL DATABASE

The SQL CREATE DATABASE STATEMENT

The CREATE DATABASE statement in SQL is used to create a new SQL database.

Syntax

CREATE DATABASE database_name;

Let's create a database and give name as testdb

CREATE database testdb;

```
mysql> create database testdb;
Query OK, 1 row affected (0.28 sec)
```

Now, let's check the databases in MySQL by using **show databases** query.

Show databases;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| testdb             |
| world              |
+--------------------+
7 rows in set (0.06 sec)
```

The SQL DROP DATABASE STATEMENT

The DROP DATABASE statement in SQL is used to drop an existing SQL database.

Syntax

DROP DATABASE database_name;

Let's drop the created database by using drop database testdb.

```
DROP database testdb;
```

```
mysql> drop database testdb;
Query OK, 0 rows affected (0.78 sec)
```

Now, let's check the databases in MySQL by using **show databases** query after dropping the testdb.

```
SHOW databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)
```

The created database(testdb) has been dropped.

The SQL CREATE TABLE

The CREATE TABLE statement in SQL is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
   column1 data_type,
   column2 data_type,
   column3 data_type,
  ....
);
```

The column1, column2, ….., specify the names of the columns of the table. The datatype parameter specifies the type of data the column can hold (e.g., varchar, integer, date, etc.)

Let's create a customer table

CREATE TABLE cutomer(id integer, first_name varchar(10), last_name varchar(10), city varchar(10), country varchar(15), phone varchar(15));

```
mysql> create table customer (id integer, first_name varchar(10),last_name varchar(10),city varchar(10),country varchar(15),phone varchar(15));
Query OK, 0 rows affected (1.85 sec)
```

To check the schema of the table, use desc table_name.

DESC customer;

```
mysql> desc customer;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | YES  |     | NULL    |       |
| first_name | varchar(10) | YES  |     | NULL    |       |
| last_name  | varchar(10) | YES  |     | NULL    |       |
| city       | varchar(10) | YES  |     | NULL    |       |
| country    | varchar(15) | YES  |     | NULL    |       |
| phone      | varchar(15) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.04 sec)
```

The SQL DROP TABLE STATEMENT

The DROP TABLE statement in SQL is used to drop an existing table in a database.

DROP TABLE customer;

```
mysql> drop table customer;
Query OK, 0 rows affected (1.24 sec)

mysql> desc customer;
ERROR 1146 (42S02): Table 'testdb.customer' doesn't exist
```

The table has dropped after running the query drop table table_name. As we can see, the table does not exist after dropped.

Now we are going to create the same table again to insert the values in that table.

The SQL INSERT INTO STATEMENT

The INSERT INTO statement in SQL is used to insert new records in a table.

INSERT INTO query

We can write the INSERT INTO statement in two ways. The first way is to specify both the column names and the values to be inserted:

INSERT INTO customer(id , first_name, last_name ,city ,country,phone)VALUES (2, 'Ana', 'Trujillo', 'Mexico', 'Mexico', (5) 555-4729);

```
mysql> INSERT INTO customer (id,first_name,last_name,city,country,phone) VALUES(2,'Ana','Trujillo','México','Mexico','(5) 555-4729');
Query OK, 1 row affected (0.12 sec)

mysql> select * from customer;
+------+------------+-----------+--------+---------+--------------+
| id   | first_name | last_name | city   | country | phone        |
+------+------------+-----------+--------+---------+--------------+
|    2 | Ana        | Trujillo  | México | Mexico  | (5) 555-4729 |
+------+------------+-----------+--------+---------+--------------+
1 row in set (0.00 sec)
```

If users are adding values for all the columns of the table, you don't need to specify the particular column names in the SQL query. However, ensure the order of the values is in the same order as the columns in the table.

The INSERT INTO query would be as follows:

INSERT INTO customer

VALUES (3, 'Antonio, 'Moreno, 'Mexico', 'Mexico', (5) 555-3932);

```
mysql> select * from customer;
+------+------------+-----------+--------+---------+--------------+
| id   | first_name | last_name | city   | country | phone        |
+------+------------+-----------+--------+---------+--------------+
|    2 | Ana        | Trujillo  | México | Mexico  | (5) 555-4729 |
|    3 | Antonio    | Moreno    | México | Mexico  | (5) 555-3932 |
+------+------------+-----------+--------+---------+--------------+
2 rows in set (0.00 sec)
```

We have inserted two rows yet. Similarly, we can insert many rows in the table. Finally, we have added ten rows as we can see in the picture below.

SELECT * FROM customer;

```
mysql> select * from customer;
+------+------------+-----------+------------+---------+----------------+
| id   | first_name | last_name | city       | country | phone          |
+------+------------+-----------+------------+---------+----------------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 |
|   11 | Victoria   | Ashworth  | London     | UK      | (171) 555-1212 |
+------+------------+-----------+------------+---------+----------------+
10 rows in set (0.00 sec)
```

The SQL NULL VALUES

What is a NULL Value?

The field with a NULL value is a field with no value. If the field in a table is optional, to insert new data or update data without adding a value to this field and Then, the field will be saved as a NULL value.

Note: A NULL value is not the same as a zero value, or we can say a field that holds spaces. The field with a NULL value is one that has been left blank during record creation!

Insert the NULL values in tables

INSERT INTO customer VALUES(11, ‘Victoria’, ‘Ashworth’, ‘London’, NULL, ‘(171) 555-1212’)

```
mysql> INSERT INTO customer VALUES(11,'Victoria','Ashworth','London',NULL,'(171) 555-1212');
Query OK, 1 row affected (0.16 sec)

mysql> select * from customer;
+------+------------+-----------+------------+---------+----------------+
| id   | first_name | last_name | city       | country | phone          |
+------+------------+-----------+------------+---------+----------------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 |
|   11 | Victoria   | Ashworth  | London     | NULL    | (171) 555-1212 |
+------+------------+-----------+------------+---------+----------------+
10 rows in set (0.00 sec)
```

As we can able to see, the last row contains one NULL value.

How to check for NULL Values?

To test for NULL values in the table has to use the **IS NULL** and **IS NOT NULL** operators instead.

IS NULL Syntax

SELECT *

FROM customer WHERE country IS NULL;

```
mysql> select * from customer where country IS NULL;
+------+------------+-----------+--------+---------+----------------+
| id   | first_name | last_name | city   | country | phone          |
+------+------------+-----------+--------+---------+----------------+
|   11 | Victoria   | Ashworth  | London | NULL    | (171) 555-1212 |
+------+------------+-----------+--------+---------+----------------+
1 row in set (0.00 sec)
```

IS NOT NULL Syntax

SELECT * FROM customer

WHERE country IS NOT NULL;

```
mysql> select * from customer where country IS NOT NULL;
+------+------------+-----------+------------+---------+----------------+
| id   | first_name | last_name | city       | country | phone          |
+------+------------+-----------+------------+---------+----------------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 |
+------+------------+-----------+------------+---------+----------------+
9 rows in set (0.00 sec)
```

It will return those countries which have some values(expect Null values).

The SQL UPDATE STATEMENT

The UPDATE statement in SQL is used to modify the existing records in a table.

UPDATE Syntax

UPDATE customer

SET country = ‘Mexico’ WHERE id = 11;

```
mysql> update customer set country = 'maxico' where id = 11;
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from customer;
+------+------------+-----------+------------+---------+----------------+
| id   | first_name | last_name | city       | country | phone          |
+------+------------+-----------+------------+---------+----------------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 |
|   11 | Victoria   | Ashworth  | London     | maxico  | (171) 555-1212 |
+------+------------+-----------+------------+---------+----------------+
10 rows in set (0.00 sec)
```

We have updated the null value of the country with Mexico.

The SQL DELETE STATEMENT

The DELETE statement in SQL is used to delete existing records in a table.

DELETE Syntax

DELETE FROM customer WHERE id = 11;

```
mysql> delete from customer where id = 11;
Query OK, 1 row affected (0.15 sec)

mysql> select * from customer;
+------+------------+-----------+------------+---------+----------------+
| id   | first_name | last_name | city       | country | phone          |
+------+------------+-----------+------------+---------+----------------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 |
+------+------------+-----------+------------+---------+----------------+
9 rows in set (0.00 sec)
```

We have deleted one row, which contains id = 11.

The SQL ALTER TABLE STATEMENT

The ALTER TABLE statement in SQL is used to add, modify, or delete columns in an existing table. And it also used to add and drop various constraints on a current table.

5.1.1. ALTER TABLE - ADD COLUMN IN EXISTING TABLE

To add a new column in a table, use the SQL query

ALTER TABLE customer

ADD email varchar(25);

```
mysql> alter table customer add email varchar(25);
Query OK, 0 rows affected (2.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from customer;
+------+------------+-----------+------------+---------+----------------+-------+
| id   | first_name | last_name | city       | country | phone          | email |
+------+------------+-----------+------------+---------+----------------+-------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   | NULL  |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   | NULL  |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 | NULL  |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  | NULL  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     | NULL  |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    | NULL  |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 | NULL  |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    | NULL  |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 | NULL  |
+------+------------+-----------+------------+---------+----------------+-------+
9 rows in set (0.00 sec)
```

5.1.2. ALTER TABLE – MODIFY/ALTER COLUMN

To change the data type of column values in a table, use the following syntax:

ALTER TABLE customer ADD COLUMN dob date;

```
mysql> alter table customer add dob date;
Query OK, 0 rows affected (1.83 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

We have assigned the dob with the datatype date. But now we want to change the datatype from date to year.

ALTER TABLE customer MODIFY dob year;

```
mysql> alter table customer modify dob year;
Query OK, 9 rows affected (3.68 sec)
Records: 9  Duplicates: 0  Warnings: 0
```

5.1.3. ALTER TABLE - DROP COLUMN

To delete a specific column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

Syntax:

ALTER TABLE customer

DROP COLUMN email;

```
mysql> alter table customer drop column email;
Query OK, 0 rows affected (2.40 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from customer;
+------+------------+-----------+------------+---------+----------------+
| id   | first_name | last_name | city       | country | phone          |
+------+------------+-----------+------------+---------+----------------+
|    2 | Ana        | Trujillo  | México     | Mexico  | (5) 555-4729   |
|    3 | Antonio    | Moreno    | México     | Mexico  | (5) 555-3932   |
|    4 | Thomas     | Hardy     | London     | UK      | (171) 555-7788 |
|    5 | Christina  | Berglund  | Luleå      | Sweden  | 0921-12 34 65  |
|    6 | Hanna      | Moos      | Mannheim   | Germany | 0621-08460     |
|    7 | Frédérique | Citeaux   | Strasbourg | France  | 88.60.15.31    |
|    8 | Martín     | Sommer    | Madrid     | Spain   | (91) 555 22 82 |
|    9 | Laurence   | Lebihan   | Marseille  | France  | 91.24.45.40    |
|   10 | Elizabeth  | Lincoln   | Tsawassen  | Canada  | (604) 555-4729 |
+------+------------+-----------+------------+---------+----------------+
9 rows in set (0.00 sec)
```

6. The SQL CONSTRAINTS Datatypes

The Constraints in SQL can be specified when the table is created with the CREATE TABLE statement, or after the table is altered with the ALTER TABLE statement.

Syntax:

```
CREATE TABLE table_name (
 column1 datatype constraint,
 column2 datatype constraint,
column3 datatype constraint,
 ....
);
```

SQL Constraints

SQL constraints are used to specify any rules for the records in a table. Constraints can be used to limit the type of data that can go into a table. It ensures the accuracy and reliability of the records in the table, and if there is any violation between the constraint and the record action, the action is aborted. Constraints can be column level or table level. Column level constraints apply to a column, and table-level constraints apply to the whole table.

The constraints are commonly used in SQL

CONSTRAINTS	DESCRIPTION
Not Null	It Ensures that a column cannot have a NULL value.
Unique	It Ensures that all the values in a column are unique.
Primary Key	It is a combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.
Foreign Key	Uniquely identifies a record /row in another table
Check	It checks that all values in a column satisfy a specific condition
Default	It gives a default value for a column when no value is specified
Index	It is Used to create and retrieve data from the database quickly.

NOT NULL CONSTRAINTS

The NOT NULL constraint enforces a column NOT to accept NULL values. This imposes a field always to contain a value, which means that the user cannot insert a new record in a table or update a record without adding a value to this field.

NOTE: By default, a column can hold NULL values.

Create a table using SQL not null constraints

The following SQL ensures that the "id", "First_name" and "Last_name" columns will NOT accept NULL values when the "student" table is created:

Example

```
CREATE TABLE student(
id int NOT NULL,
first_name varchar(25) NOT NULL,
last_name varchar(25) NOT NULL,
age int
);
```

```
mysql> create table student (id int not null, first_name varchar(25) not null, last_name varchar(25) not null, age int);
Query OK, 0 rows affected (1.16 sec)

mysql> desc student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   |     | NULL    |       |
| first_name | varchar(25) | NO   |     | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.24 sec)
```

In the above table, it has specified the id, first_name, and last_name as not null and age as null.

SQL NOT NULL on ALTER table Statement

To make a NOT NULL constraint on the "age" column when the "student" table is already created, use the following SQL:

Example:

```
ALTER TABLE student

MODIFY age int NOT NULL;
```

```
mysql> alter table student modify age int not null;
Query OK, 0 rows affected (1.93 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc customer;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | YES  |     | NULL    |       |
| first_name | varchar(10) | YES  |     | NULL    |       |
| last_name  | varchar(10) | YES  |     | NULL    |       |
| city       | varchar(10) | YES  |     | NULL    |       |
| country    | varchar(15) | YES  |     | NULL    |       |
| phone      | varchar(15) | YES  |     | NULL    |       |
| dob        | year(4)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

In the above table, it has specified the id, first_name,last_name, and age as not null.

SQL UNIQUE CONSTRAINT

The **UNIQUE** constraint in SQL ensures that all values in a column are distinct. UNIQUE and PRIMARY KEY constraints both provides a guarantee for **uniqueness** for a column or group of columns. A PRIMARY KEY constraint, by default, has a UNIQUE constraint. However, the user can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Creates UNIQUE constraint on the "id" column when the "person" table is created

```
CREATE TABLE person (

id int NOT NULL,
```

last_name varchar(255) NOT NULL,

first_name varchar(255),

age int,

UNIQUE (ID)

);

```
mysql> create table person(id int not null, first_name varchar(25) not null, last_name varchar(25) not null,age int, unique(id));
Query OK, 0 rows affected (1.29 sec)

mysql> desc person;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| first_name | varchar(25) | NO   |     | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

We have applied unique constraints on id, and as we can see, it is showing as the primary key.

Create a UNIQUE constraint on the "first_name" column when the "persons" table already exists.

ALTER TABLE persons

ADD UNIQUE (first_name);

```
mysql> alter table person add unique(first_name);
Query OK, 0 rows affected (0.76 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc person;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| first_name | varchar(25) | NO   | UNI | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Now we have two unique constraints(id and first_name) in the person table.

To name the UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

ALTER TABLE person

ADD CONSTRAINT UC_person UNIQUE (age, last_name);

```
mysql> ALTER TABLE person
    -> ADD CONSTRAINT UC_Persons UNIQUE (age,last_name);
Query OK, 0 rows affected (1.65 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc person;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| first_name | varchar(25) | NO   | UNI | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Here the age and last_name are converted as unique constraints.

DROP A UNIQUE CONSTRAINT

To drop a UNIQUE constraint, use the SQL query

ALTER TABLE person

DROP INDEX UC_Person;

```
mysql> ALTER TABLE person
    -> drop index UC_Persons;
Query OK, 0 rows affected (0.38 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc person;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| first_name | varchar(25) | NO   | UNI | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.07 sec)
```

As we can see in the person table The unique constraint(UC_Persons) has been dropped.

SQL PRIMARY KEY CONSTRAINTS

The PRIMARY KEY constraint uniquely identifies each of the records in a table. Only ONE primary key can have in a table. And also, in the table, this primary key can consist of single or multiple columns (fields). Primary keys should contain UNIQUE values, and cannot contain **NULL** values.

CREATE TABLE person(ID int NOT NULL, last_name varchar(255) NOT NULL, first_name varchar(255), age int, PRIMARY KEY(ID));

```
mysql> CREATE TABLE person(ID int NOT NULL,
    ->     last_name varchar(255) NOT NULL,
    ->     first_name varchar(255),
    ->     age int,
    ->     PRIMARY KEY (ID)
    -> );
Query OK, 0 rows affected (0.61 sec)

mysql> desc person;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| ID         | int(11)      | NO   | PRI | NULL    |       |
| last_name  | varchar(255) | NO   |     | NULL    |       |
| first_name | varchar(255) | YES  |     | NULL    |       |
| age        | int(11)      | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

To allow the naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the SQL syntax.

```
CREATE TABLE person (
 id int NOT NULL,
 last_name varchar(255) NOT NULL,
 first_name varchar(255),
age int,
 CONSTRAINT PK_person PRIMARY KEY (id,last_name)
);
```

```
mysql> CREATE TABLE Person1 (
    ->     id int NOT NULL,
    ->     last_name varchar(25) NOT NULL,
    ->     first_name varchar(25),
    ->     age int,
    ->     CONSTRAINT PK_Person PRIMARY KEY (id,last_name)
    -> );
Query OK, 0 rows affected (0.94 sec)

mysql> desc Person1
    -> ;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| last_name  | varchar(25) | NO   | PRI | NULL    |       |
| first_name | varchar(25) | YES  |     | NULL    |       |
| age        | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Note: In this example, there is only ONE PRIMARY KEY as PK_Person. And the VALUE of the primary key is made up of **two columns** (id+ last_name).

SQL PRIMARY KEY on ALTER TABLE

Create a PRIMARY KEY constraint on the column_name "**id**" when the table_name(student) is already created, use the following SQL:

ALTER TABLE student

ADD PRIMARY KEY (id);

```
mysql> alter table student add primary key(id);
Query OK, 0 rows affected (1.71 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| first_name | varchar(25) | NO   |     | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Here we have assigned the primary key as “id” on the student table.

Allow the naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the SQL query:

ALTER TABLE student

ADD CONSTRAINT PK_student PRIMARY KEY (id,first_name);

```
mysql> desc student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   |     | NULL    |       |
| first_name | varchar(25) | NO   |     | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> alter table student
    -> ADD CONSTRAINT PK_student PRIMARY KEY (id,first_name);
Query OK, 0 rows affected (1.38 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

DROP PRIMARY KEY CONSTRAINTS

To drop the PRIMARY KEY constraint from the table, use the SQL Query:

ALTER TABLE student

DROP PRIMARY KEY;

```
mysql> alter table student
    -> drop primary key;
Query OK, 0 rows affected (2.44 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   |     | NULL    |       |
| first_name | varchar(25) | NO   |     | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| age        | int(11)     | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

As we can see from the student table, the primary key has been dropped from the table.

SQL FOREIGN KEY CONSTRAINT

A FOREIGN KEY is used to link two tables together. It is sometimes also called a referencing key. Foreign Key is a combination of columns (can be single column) whose value matches a Primary Key in the different tables. The relationship between two tables matches the Primary Key in one of the tables with a Foreign Key in the second table. If the table contains a primary key defined on any field, then the user should not have two records having the equal value of that field.

Let's create two tables using the foreign key.

CUSTOMER table

```
CREATE TABLE customer(
 Id int NOT NULL,
 Name varchar(20)   NOT NULL,
Age int NOT NULL,
Address varchar(25) ,
 Salary decimal (18, 2),
  PRIMARY KEY (id)
);
```

```
mysql> CREATE TABLE customer(
    ->     Id int NOT NULL,
    ->     Name varchar(20)    NOT NULL,
    ->     Age int NOT NULL,
    ->     Address varchar(25) ,
    ->     Salary decimal (18, 2),
    ->     PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (1.05 sec)

mysql>
mysql> desc customer;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| Id      | int(11)       | NO   | PRI | NULL    |       |
| Name    | varchar(20)   | NO   |     | NULL    |       |
| Age     | int(11)       | NO   |     | NULL    |       |
| Address | varchar(25)   | YES  |     | NULL    |       |
| Salary  | decimal(18,2) | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
5 rows in set (0.08 sec)
```

Order Table with Foreign key

CREATE TABLE Orders (OrderID int NOT NULL, OrderNumber int NOT NULL, Id int,

PRIMARY KEY(OrderID), CONSTRAINT FK_customerOrder FOREIGN KEY(Id));

```
mysql> CREATE TABLE Orders (
    ->     OrderID int NOT NULL,
    ->     OrderNumber int NOT NULL,
    ->     Id int,
    ->     PRIMARY KEY (OrderID),
    ->     CONSTRAINT FK_customerOrder FOREIGN KEY (Id)
    ->     REFERENCES customer(Id)
    -> );
Query OK, 0 rows affected (1.08 sec)

mysql> desc orders;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| OrderID     | int(11) | NO   | PRI | NULL    |       |
| OrderNumber | int(11) | NO   |     | NULL    |       |
| Id          | int(11) | YES  | MUL | NULL    |       |
+-------------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Here the Id is the primary key for the customer table and foreign key for orders table.

FOREIGN KEY on ALTER TABLE

To create the FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the SQL query:

ALTER TABLE Orders

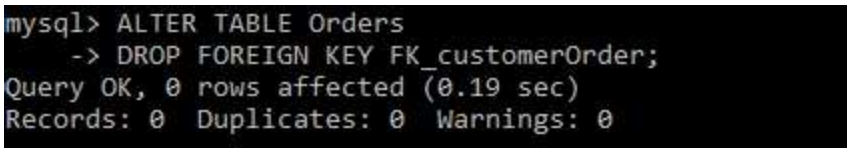
ADD FOREIGN KEY (ID) REFERENCES customer(id);

```
mysql> ALTER TABLE Orders
    -> ADD FOREIGN KEY (ID) REFERENCES customer(id);
Query OK, 0 rows affected (2.38 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> desc orders;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| OrderID     | int(11) | NO   | PRI | NULL    |       |
| OrderNumber | int(11) | NO   |     | NULL    |       |
| Id          | int(11) | YES  | MUL | NULL    |       |
+-------------+---------+------+-----+---------+-------+
3 rows in set (0.03 sec)
```

DROP A FOREIGN KEY CONSTRAINT

To drop a FOREIGN KEY constraint from the table, use the SQL query:

```
ALTER TABLE Orders

DROP FOREIGN KEY FK_PersonOrder;
```



SQL CHECK CONSTRAINTS

The CHECK CONSTRAINTS is used to limit the range of value that can be placed in a column if the user defines a CHECK constraint on a single column, it allows only specific values for the column. If the user defines a CHECK constraint on a table, it can limit the values in particular columns based on values in another column in the row.

SQL CHECK on CREATE TABLE

SQL Query to creates a CHECK constraint on the column "Age" when the table "Persons" is created. The CHECK constraint makes sure that the user can not have any person below 18 years:

```
CREATE TABLE Persons (

 ID int NOT NULL,

 LastName varchar(255) NOT NULL,

 FirstName varchar(255),

Age int,

 CHECK (Age>=18)

);
```

```
mysql> CREATE TABLE Persons (
    ->     ID int NOT NULL,
    ->     LastName varchar(255) NOT NULL,
    ->     FirstName varchar(255),
    ->     Age int,
    ->     CHECK (Age>=18)
    -> );
Query OK, 0 rows affected (1.19 sec)

mysql> insert into Persons values(1, 'abc', 'aaa', 17);
ERROR 3819 (HY000): Check constraint 'persons_chk_1' is violated.
```

Here we have created the Persons table and given a check constraint on the Age column. If the Age<18, then it will throw an error, as shown below.

INSERT INTO Persons VALUES(1, 'abc', 'aaa', 17);

```
mysql> insert into Persons values(1, 'abc', 'aaa', 17);
ERROR 3819 (HY000): Check constraint 'persons_chk_1' is violated.
```

For creating a CHECK constraint on multiple columns in the table, use the SQL syntax:

```
mysql> CREATE TABLE Persons (
    ->     ID int NOT NULL,
    ->     LastName varchar(255) NOT NULL,
    ->     FirstName varchar(255),
    ->     Age int,
    ->     City varchar(255),
    ->     CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
    -> );
Query OK, 0 rows affected (1.20 sec)
```

CHECK on ALTER TABLE

Create a CHECK constraint on the column "Age" when the table is already created, use the following SQL:

ALTER TABLE Persons

ADD CHECK (Age >= 18)

```
mysql> ALTER TABLE Persons
    -> ADD CHECK (Age>=18)
    -> ;
Query OK, 0 rows affected (2.58 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Defining CHECK constraint on multiple columns of a table, use the SQL query:

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
```

```
mysql> ALTER TABLE Persons
    -> ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
Query OK, 0 rows affected (2.31 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

DROP A CHECK CONSTRAINT

To drop a CHECK constraint from the table, use the following SQL:

```
ALTER TABLE Persons
DROP CHECK CHK_PersonAge;
```

```
mysql> ALTER TABLE Persons
    -> DROP CHECK CHK_PersonAge;
Query OK, 0 rows affected (0.38 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Here we have dropped the CHK_PersonAge constraints by using the drop statement.

SQL DEFAULT CONSTRAINT

The DEFAULT constraint in SQL is used to provide a default value for a column of the table. The default value will be added to every new record if no other value is mentioned.

SQL DEFAULT on CREATE TABLE

The SQL query to sets a DEFAULT value for the "City" column when the "Persons" table is created

```
CREATE TABLE Persons (
 ID int NOT NULL,
```

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

City varchar(255) DEFAULT 'Sandnes'

);

```
mysql> CREATE TABLE Persons (
    ->     ID int NOT NULL,
    ->     LastName varchar(255) NOT NULL,
    ->     FirstName varchar(255),
    ->     Age int,
    ->     City varchar(255) DEFAULT 'Sandnes'
    -> );
Query OK, 0 rows affected (1.06 sec)

mysql> desc persons
    -> ;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ID        | int(11)      | NO   |     | NULL    |       |
| LastName  | varchar(255) | NO   |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Age       | int(11)      | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | Sandnes |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.06 sec)
```

As we can see in the Persons table, the city name is written as Sandnes by Default.

SQL DEFAULT on ALTER TABLE

To create a DEFAULT constraint on the column "City" when the table is already created, use the following SQL:

ALTER TABLE Persons

ALTER Age SET DEFAULT 20;

```
mysql> ALTER TABLE Persons
    -> ALTER Age SET DEFAULT 20;
Query OK, 0 rows affected (0.44 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc persons;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ID        | int(11)      | NO   |     | NULL    |       |
| LastName  | varchar(255) | NO   |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Age       | int(11)      | YES  |     | 20      |       |
| City      | varchar(255) | YES  |     | Sandnes |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.04 sec)
```

DROP A DEFAULT CONSTRAINT

To drop a DEFAULT constraint from the table, use the SQL query:

ALTER TABLE Persons

ALTER City DROP DEFAULT;

```
mysql> ALTER TABLE Persons
    -> ALTER City DROP DEFAULT;
Query OK, 0 rows affected (0.18 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc persons;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ID        | int(11)      | NO   |     | NULL    |       |
| LastName  | varchar(255) | NO   |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Age       | int(11)      | YES  |     | 20      |       |
| City      | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

As we can see in the Persons table, the default value of the city has been removed.