Chapter 2

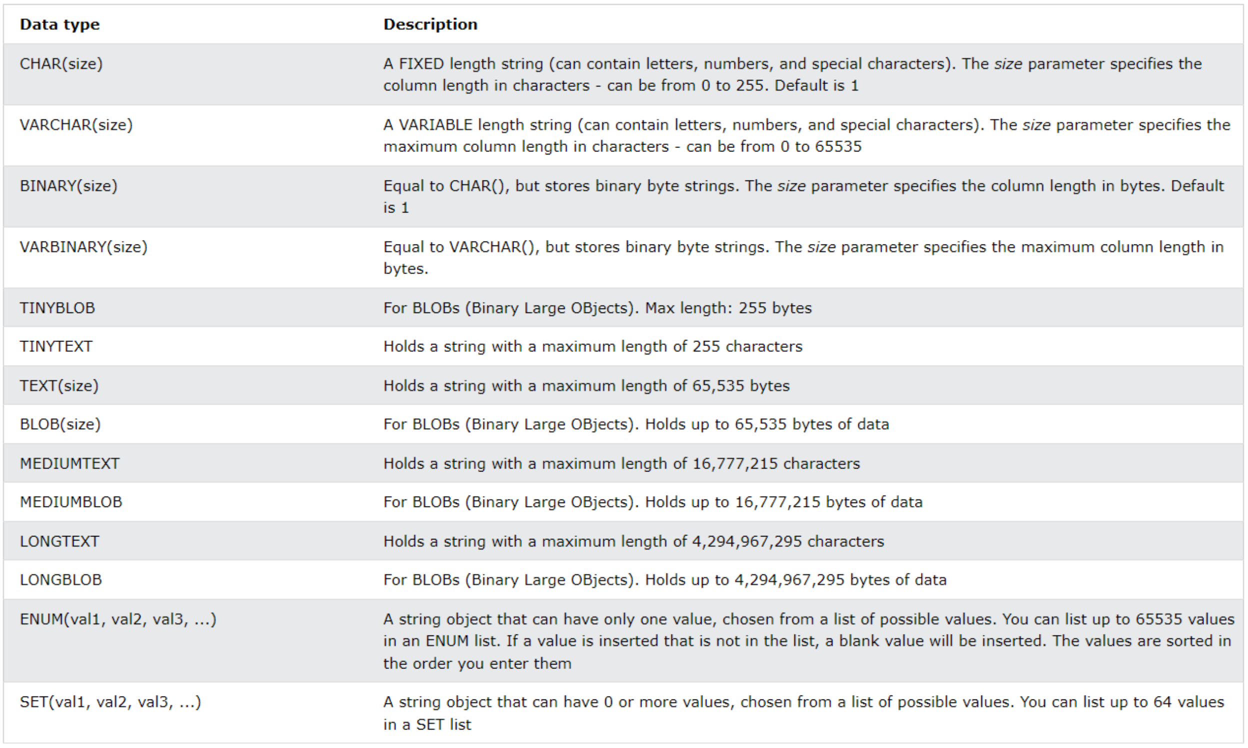
MySQL Data Type And Data Definition Language (DDL)

“In this day and age if you've got the technology then it's vital to use that technology to track people down. The number on the database should be the maximum number you can get.” - Tony Blair

2.1. MySQL Data Types

A data type is a form of data modeling that is declared when creating a table. This data type will affect any data that will be entered into a table. The data to be entered must match the declared data type. Each column in a database table is required to have a name and a data type. An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data. In MySQL, there are three main data types: string, numeric, and date and time.

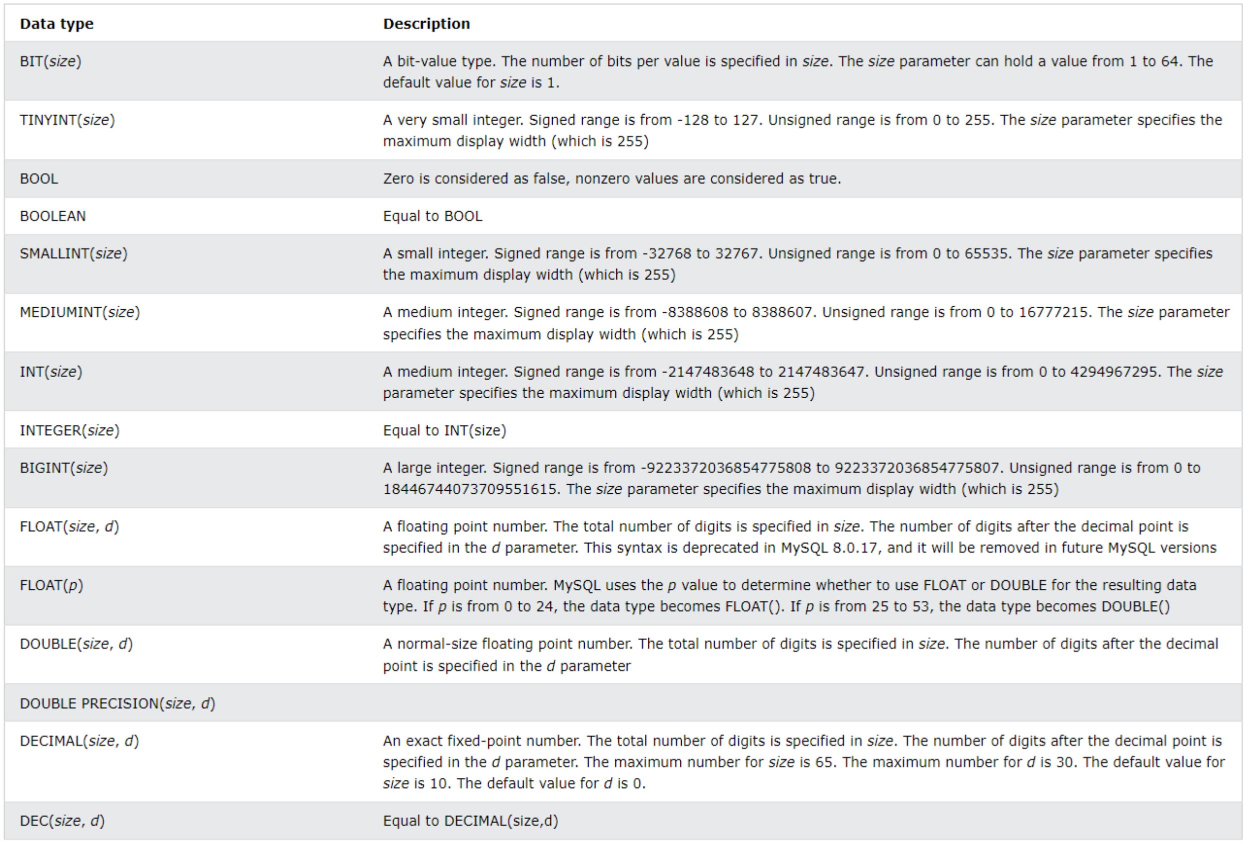
2.2. String Data Types



Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large OBjects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

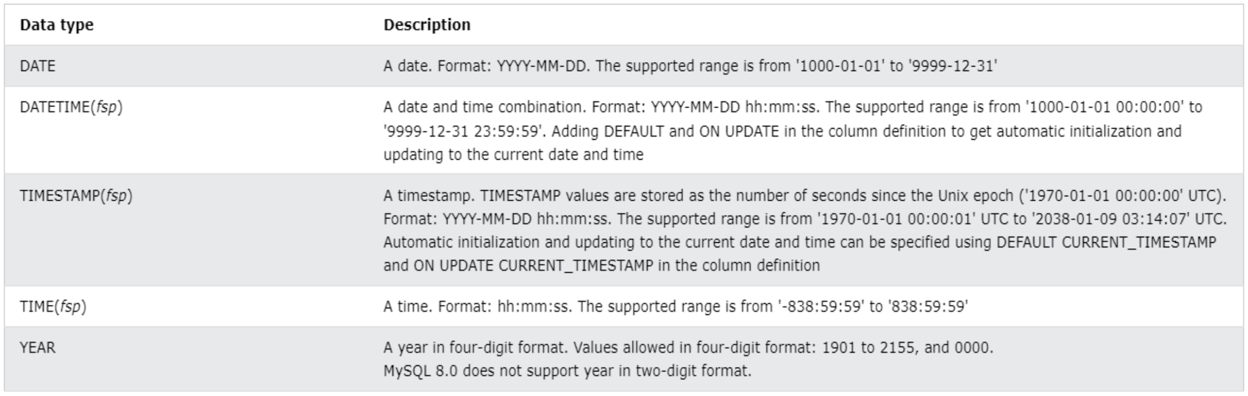
Note: *The difference between **CHAR** and **VARCHAR** is the way MySQL allocates the storage size of the data entered into the column. For example, if we define a table with a column of type CHAR (5), even though the letter or character we enter is only 1 character, MySQL still stores the column for 5 characters. However, if we define it as VARCHAR (5), and we input data with the number of characters 2, then the storage size will only use 2 characters, so VARCHAR is more flexible and efficient.*

2.3. Numeric Data Types



Data type	Description
BIT(*size*)	A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)
MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter
DOUBLE PRECISION(*size*, *d*)	
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.
DEC(*size*, *d*)	Equal to DECIMAL(size,d)

2.4. Date and Time Data Types



Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(*fsp*)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

2.5. Data Definition Language (DDL)

DDL or Data Definition Language consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

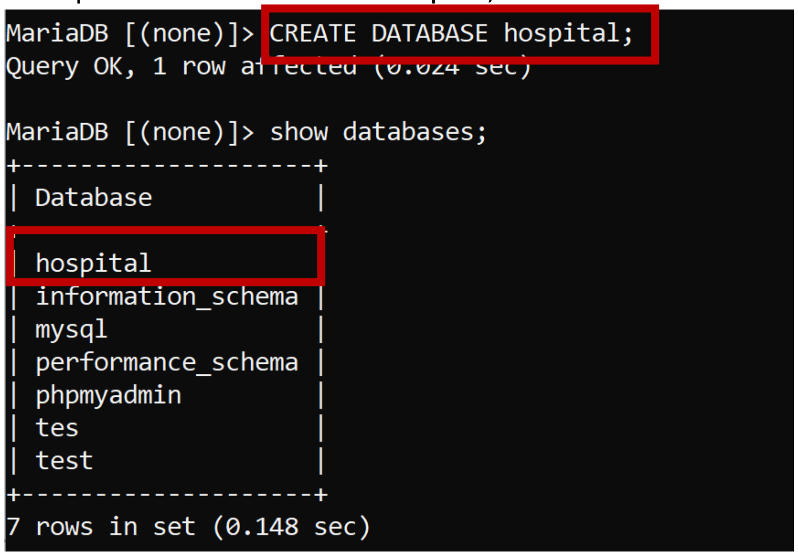
- CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- DROP: This command is used to delete objects from the database.
- ALTER: This is Used to make changes to the table structure that has been created, either adding Fields (Add), renaming Fields (Change) or renaming tables (Rename), and deleting Fields (Drop).

2.6. Create a Database

The database is the main medium that must be made in building a database so that later we can place several tables with their fields. The syntax is as follows:

CREATE DATABASE database_name;

Example: CREATE DATABASE hospital;



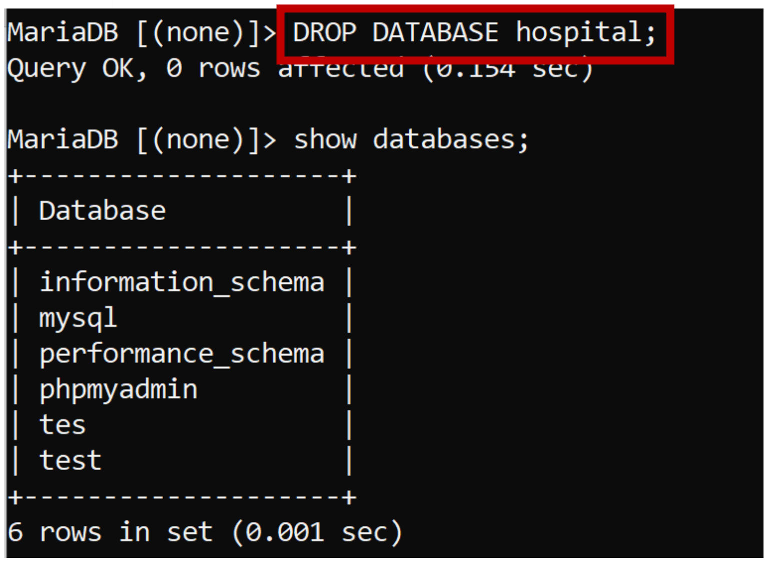
```
MariaDB [(none)]> CREATE DATABASE hospital;
Query OK, 1 row affected (0.024 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| hospital           |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| tes                |
| test               |
+--------------------+
7 rows in set (0.148 sec)
```

2.7. Delete Database

To delete a database that has been created, you can use the following syntax:

DROP DATABASE database_name;



```
MariaDB [(none)]> DROP DATABASE hospital;
Query OK, 0 rows affected (0.154 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| tes                |
| test               |
+--------------------+
6 rows in set (0.001 sec)
```

2.8. Create Table

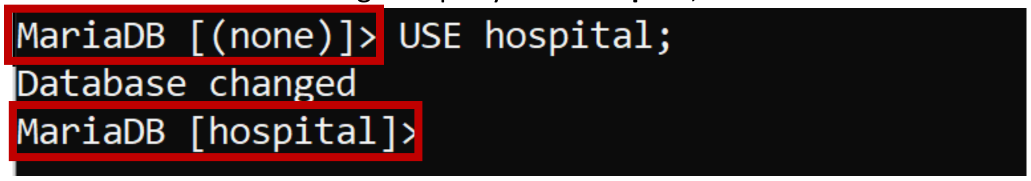
Before creating a table, we must activate the database that has been created using the "**USE**" command.

Syntax: **USE database_name;**

Meanwhile, the syntax for creating a table is: **CREATE TABLE table_name;**

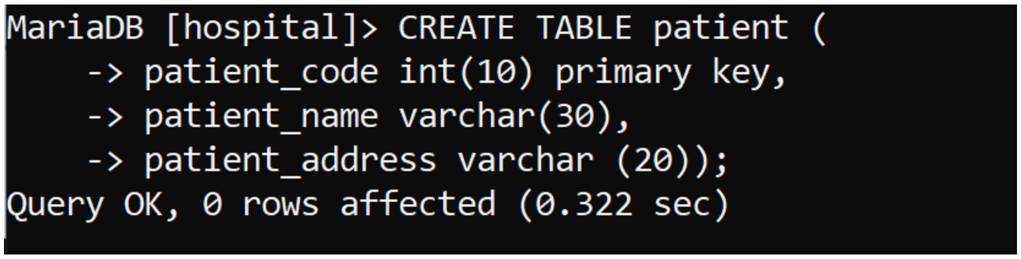
Follow these steps:

1. Create a database (hospital)
2. then enter the query "**SHOW databases**"; to see if the **hospital** database is available.
3. Activate the database using the query "**USE hospital**;"



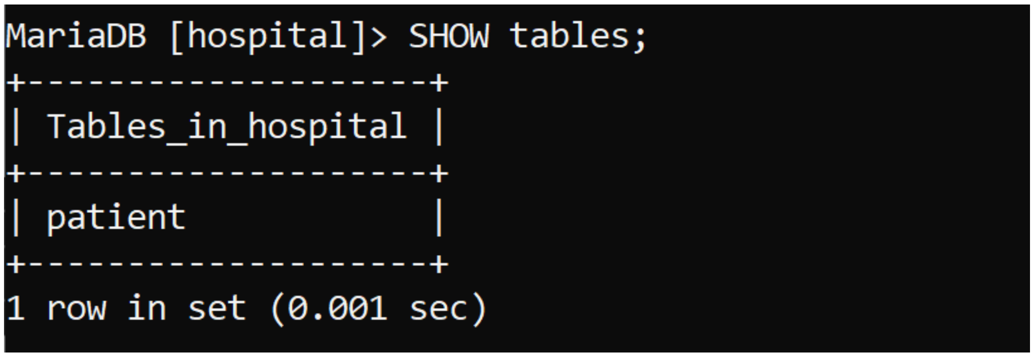
```
MariaDB [(none)]> USE hospital;
Database changed
MariaDB [hospital]>
```

4. Next is to create a **patient** table, which consists of three attributes (patient_code, patient_name and patient_address)



```
MariaDB [hospital]> CREATE TABLE patient (
    -> patient_code int(10) primary key,
    -> patient_name varchar(30),
    -> patient_address varchar (20));
Query OK, 0 rows affected (0.322 sec)
```

5. You can also view the tables that have been created in a database by using the "**SHOW tables**" command.



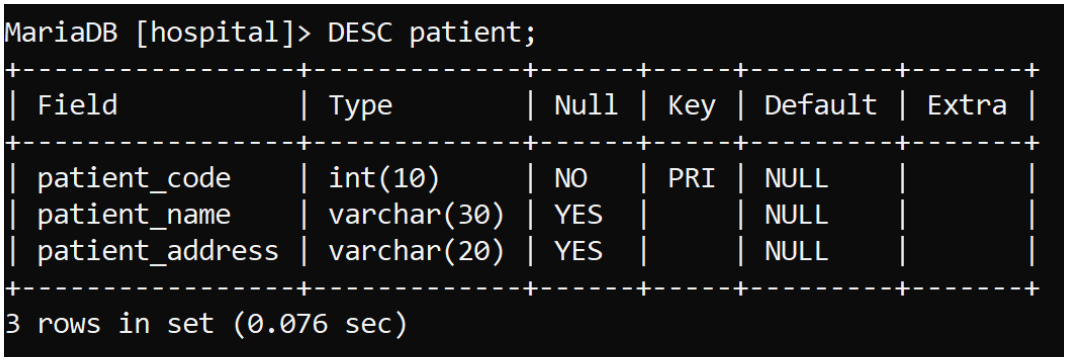
```
MariaDB [hospital]> SHOW tables;
+--------------------+
| Tables_in_hospital |
+--------------------+
| patient            |
+--------------------+
1 row in set (0.001 sec)
```

6. After the table is created, you can see the data type and length of the record set by displaying the table structure.

The syntax is as follows: **DESC table_name;**

or

DESCRIBE table_name;

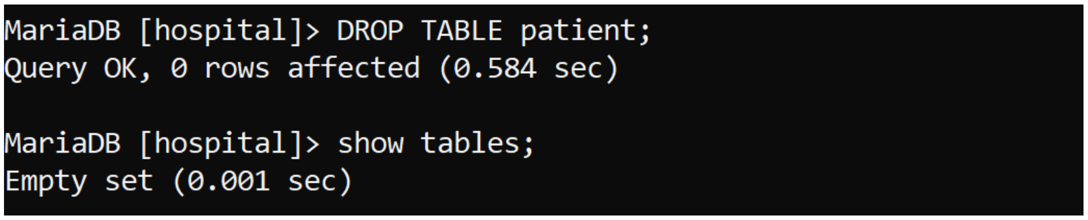


```
MariaDB [hospital]> DESC patient;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| patient_code    | int(10)     | NO   | PRI | NULL    |       |
| patient_name    | varchar(30) | YES  |     | NULL    |       |
| patient_address | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
3 rows in set (0.076 sec)
```

2.9. Delete Table

To delete a table that has been created, you can use the following SQL query:

DROP TABLE table_name;



```
MariaDB [hospital]> DROP TABLE patient;
Query OK, 0 rows affected (0.584 sec)

MariaDB [hospital]> show tables;
Empty set (0.001 sec)
```

<u>TASK 2:</u>

1. Design a database (COURSE) consisting of 3 tables:
   a. Subjects
      - subject_code char (5) primary key
      - subject_name varchar (30) not null
      - semester int (1) not null
      - sks int (1) not null
      - dept varchar (20) not null

   b. Lecturer
      - lecturer_code char (10) primary key
      - lecturer_name varchar (30) not null
      - date of birth date not null
      - gender varchar (2) not null
      - lecturer_address varchar (30) not null
      - phone varchar (15) not null

   c. Students
      - NPM char (10) primary key
      - students_name varchar (30) not null
      - date of birth date not null
      - gender varchar (2) not null
      - dept varchar (20) not null
      - students_address varchar (30) not null

2. Show table structure: (a) Subjects, (b) Lecturer, (c) Students
3. Delete lecturer table