

Structured Query Language

SQL - Day 6

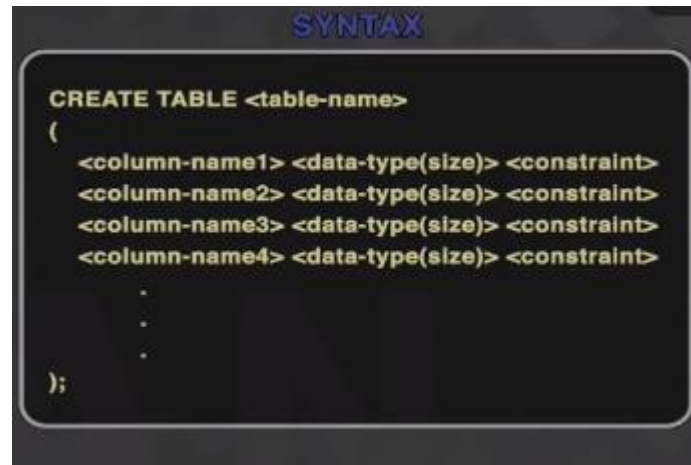
Agenda

- Different DATA TYPES in SQL
- Different Constraints in SQL



How to create table in database and add data inside table.?

Here is the **SYNTAX** for creating table .



```
SYNTAX

CREATE TABLE <table-name>
(
  <column-name1> <data-type(size)> <constraint>
  <column-name2> <data-type(size)> <constraint>
  <column-name3> <data-type(size)> <constraint>
  <column-name4> <data-type(size)> <constraint>
  .
  .
  .
);
```

Before creating table Lets first understand what are Data types and Constraints.

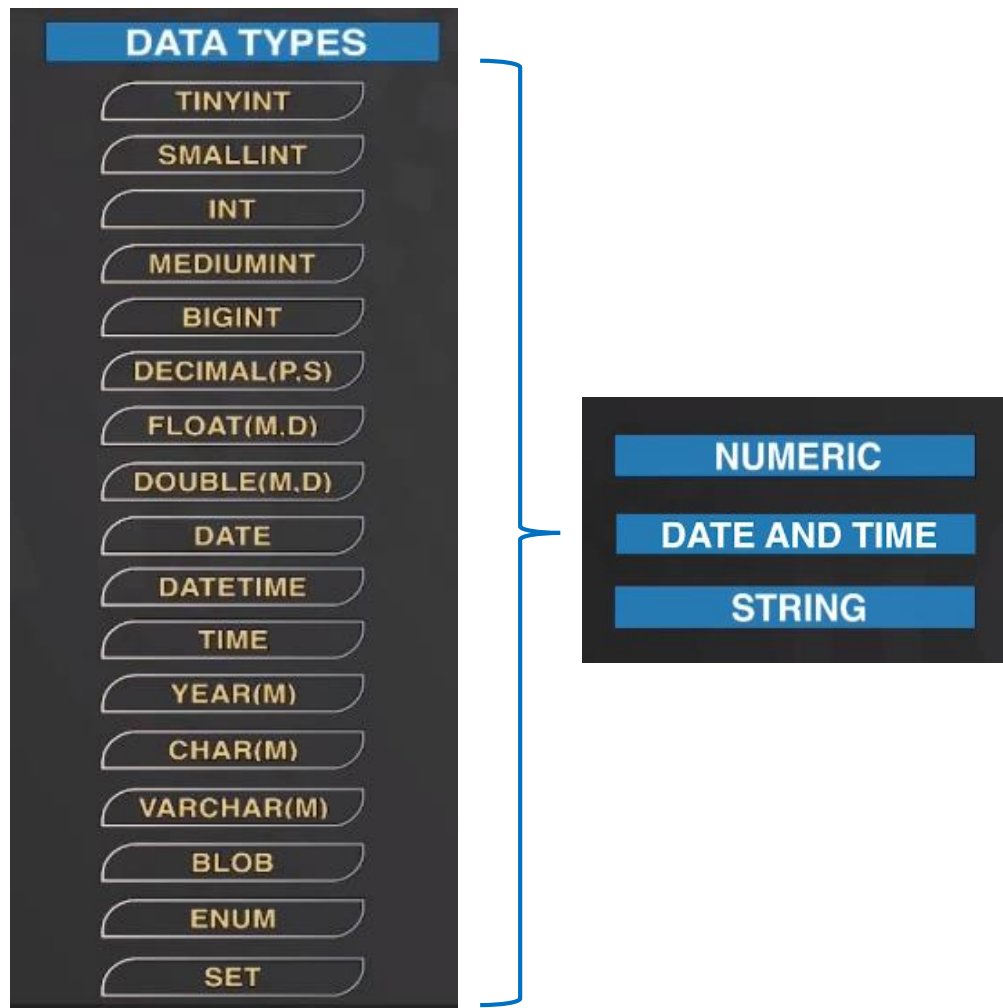
DATA TYPES in SQL:

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.

Note: Data types might have different names in different database. And even if the name is the same, the size and other details may be different! Always check the documentation!

MySQL provides different data types and each data type is categorised into three main data types such as string, numeric, and date and time as shown below:



Let's know see which data type fall under which category and what is the purpose of using it to store data in SQL database.

NUMERIC DATA TYPE.

Numeric data type is used to store numeric data/numbers.

Note: There are two types of number **whole number** and **decimal number**. There are different data types available for different types of number.

Whole Number.

Whole numbers are numbers without any decimal point.

Example: Age, mobile number, pin code, order number etc.

Data types which is used to store the **whole numbers** is listed below.

NUMERIC			
DATATYPE	RANGE(signed)	RANGE(unsigned)	Width
TINYINT	-128 to 127	0 to 255	4
SMALLINT	-32768 to 32767	0 to 65535	5
MEDIUMINT	-8388608 to 8388607	0 to 16777215	9
INT	-2147483648 to 2147483647	0 to 4294967295	11
BIGINT	-9223372036854775808 to 9223372036854775807	0 to 18446744073709551615	20

TINYINT:

TINYINT stores 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 4.

SMALLINT:

SMALLINT stores small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width 5.

MEDIUMINT:

MEDIUMINT stores medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width 9.

INT:

INT stores medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width 11.

BIGINT:

BIGINT stores large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width 20.

Decimal/Real Number.

Decimal numbers are number with precision or decimal values.

Example: Distance, price, height, weight, salary.

Data types which is used to store the **decimal numbers** is listed below.

DECIMAL(P,S) P-> precision S->scale

ex: price DECIMAL(5,2). COLUMN can store any value with 5 digits and 2 decimal

RANGE: -999.99 TO 999.99

FLOAT(M,D) M->length D->decimal. Default values of(M,D)->(10,2)

Decimal precision can go to 24 places for a FLOAT

ex: price FLOAT (6,3). COLUMN can store any value with 6 digits and 3 decimal

RANGE: -999.999 TO 999.999

If you insert 999.0009 into a FLOAT(6,3) column, the approximate result is 999.001.

DOUBLE(M,D) M->length D->decimal. Default values of(M,D)->(16,4)

Decimal precision can go to 53 places for a DOUBLE

Decimal (P,S):

Decimal (P,S) data type stores decimal numbers, where **P is precision** and **S is scale**. Decimal (P,S) can store any values with 5 digits and 2 decimals.

Float (M,D):

Float (M,D) stores decimal numbers, where **M is length** and **D is decimal**. By default **M** can store 10 digits and **D** can store 2 decimal but it can go up to 24places.

Double (M,D):

Double (M,D) stores decimal numbers, **M is length** and **D is decimal**. By default **M** can store 16 digits and **D** can store 4 decimal but it can go up to 53places.

DATE & TIME DATA TYPE.

Date & Time data type is used to store date and time data in the database. For example: Date of Birth, Hiring Date and Salary date etc.

There are different data types available to store Date & Time data. They are listed below:

DATE AND TIME		
	FORMAT	RANGE
DATE	YYYY-MM-DD	1000-01-01 TO 9999-12-31
TIME	HH:MM:SS	-838:59:59 TO 838:59:59
DATETIME	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00 TO 9999-12-31 23:59:59
TIMESTAMP	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:01 UTC TO 2038-01-19 03:14:07 UTC
YEAR	YYYY	1901 TO 2155

Date data type

The **date** data type is used to store **date**. Date data type uses **YYYY-MM-DD** format. The supported range is from **'1000-01-01'** to **'9999-12-31'**,

Time data type

The **time** data type is used to store **TIME**. Time data type uses **hh:mm:ss** format. The supported range is from **'-838:59:59'** to **'838:59:59'**.

DATETIME data type

The **datetime** data type is used to store date and time combination. Time data type uses **YYYY-MM-DD hh:mm:ss** format. The supported range is from **'1000-01-01 00:00:00'** to **'9999-12-31 23:59:59'**.

TIMESTAMP data type

TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). TIMESTAMP data type uses **YYYY-MM-DD hh:mm:ss. format**. The supported range is from **'1970-01-01 00:00:01'** UTC to **'2038-01-09 03:14:07'** UTC.

YEAR data type

Year data type stores year in **four-digit format**. Values allowed in four-digit format: **1901 to 2155, and 0000**.

String Data Type.

STRING		
CHAR(M)	FIXED LENGTH STRINGS	RANGE 255 CHARACTERS
VARCHAR(M)	VARIABLE LENGTH STRINGS	65,535 CHARACTERS
BLOB (BINARY LARGE OBJECT)	LARGE BINARY DATA	65,535 BYTES
ENUM	LIST OF POSSIBLE VALUES	65,535 DISTINCT VALUES
SET	LIST OF POSSIBLE VALUES	65,535 VALUES

Char data type

Char(M) data type stores **fixed length strings**. **M** is the number of characters that can be stored. It can store up to **255 characters**.

Varchar data type

Varchar data type stores **variable length strings**. **M** is the number of characters that can be stored. It can store up to **65535 characters**.

BLOB data type

A BLOB is a binary large object that can hold a variable amount of data/binary data such as audio, video and image.

ENUM data type:

ENUM stands for Enumeration. It allows us to limit the value chosen from a list of permitted values in the column specification at the time of table creation. It can store up to **65535 DISTINCT VALUES**.

SYNTAX: COLUMN_NAME ENUM ('value-1','value-2',.....,'value-n')

For Example: if the user gives the list of values as

COLUMN_NAME ENUM ('ONE','TWO','THREE')

Then user can enter only one of these values **('ONE','TWO','THREE')** inside that column.

SET data type:

A SET is a string object that can have zero or more values, each of which must be chosen from a list of permitted values specified when the table is created. SET column values that consist of multiple set members are specified with members separated by commas (,). A consequence of this is that SET member values should not themselves contain commas.

SYNTAX: COLUMN_NAME SET ('value-1','value-2',.....,'value-n')

For Example: if the user gives the list of values as

COLUMN_NAME SET ('ONE','TWO','THREE')

Then user can choose set of these values **('ONE','TWO','THREE')** inside that column separated with comma like : **'one','two'**.

NOTE: There are other string data type such as **CLOB (Character Large Object)** which is used to store large text like description, text books etc. Anyways using Char and Varchar data type we can store the large text so throughout the course we are using these **Char and Varchar data type instead of CLOB**.

Constraints in SQL.

SQL constraints are used to specify rules for the data in a table.

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 entered, then the data 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.

Below are different types of constraints available in SQL,



NOT NULL CONSTRAINTS:

By default, a column can hold NULL values, but the **NOT NULL** constraint enforces a column to **NOT** accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

UNIQUE CONSTRAINTS:

The **UNIQUE constraint** ensures that all **values in a column are different.**

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

PRIMARY KEY CONSTRAINTS:

The **PRIMARY KEY constraint** uniquely identifies each record in a table.

Primary keys 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).

FOREIGN KEY CONSTRAINTS:

The **FOREIGN KEY** constraint is **used to prevent actions that would destroy links between tables.**

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

CHECK CONSTARINTS:

The **CHECK** constraint is used to limit the value range that can be placed in a column.

If you define a **CHECK** constraint on a column it will allow only certain values for this column.

If you define a **CHECK** constraint on a table it can limit the values in certain columns based on values in other columns in the row.

DEFAULT CONSTARINTS:

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.

AUTO INCREMENT CONSTARINTS:

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.