# **My NOTES**

# Introduction

MY IS USED TO CREATE **THE CRUD OPERATIONS**:

1. CREATE
2. READ
3. UPDATE
4. DELETE

Inside a database :

Inter-related data is stored inside the table in the form of rows and columns.

Columns tells about the data structure or often called schema of our table.

## Creating a database ;

CREATE DATABASE DB-NAME;

Drop database db\_name; //deletes the databases;

Use db\_name to select a database ;

## Creating a table inside the database;

Use db\_name;

Create table table\_name ( column\_name1 datatype constraint );

For example:

Create table table\_name ( id int primary key ,

Name varchar(50),

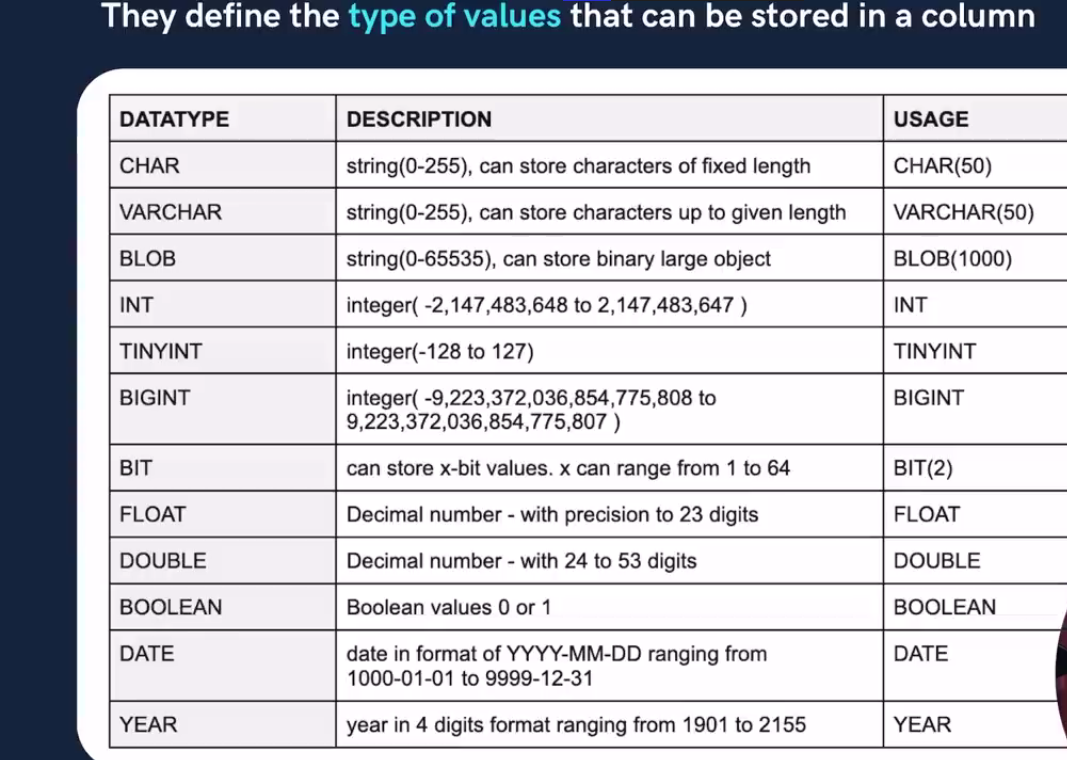
Age int not null //age can’t be null;

);

## inserting value

Insert into student (id , name , age) values (1, Ahtisham , 21 ), (2, ayti, 21);

## Most common data types;



# DATA TYPES IN MY

he following is a list of datatypes available in My , which includes string, numeric, date/time, and large object datatypes.

## Sring datatypes

The following are the **String Datatypes** in My :

|  |  |  |
| --- | --- | --- |
| **Data Type Syntax** | **Maximum Size** | **Explanation** |
| CHAR(size) | 5Maximum size of 255 characters. | Where **size** is the number of characters to store. Fixed-length strings. Space padded on right to equal **size** characters. |
| VARCHAR(size) | Maximum size of 255 characters. | Where **size** is the number of characters to store. Variable-length string. |
| TINYTEXT(size) | Maximum size of 255 characters. | Where **size** is the number of characters to store. |
| TEXT(size) | Maximum size of 65,535 characters. | Where **size** is the number of characters to store. |
| MEDIUMTEXT(size) | Maximum size of 16,777,215 characters. | Where **size** is the number of characters to store. |
| LONGTEXT(size) | Maximum size of 4GB or 4,294,967,295 characters. | Where **size** is the number of characters to store. |
| BINARY(size) | Maximum size of 255 characters. | Where **size** is the number of binary characters to store. Fixed-length strings. Space padded on right to equal **size** characters. (Introduced in My 4.1.2) |
| VARBINARY(size) | Maximum size of 255 characters. | Where **size** is the number of characters to store. Variable-length string. (Introduced in My 4.1.2) |

**Numeric Datatypes**

The following are the **Numeric Datatypes** in My :

|  |  |  |
| --- | --- | --- |
| **Data Type Syntax** | **Maximum Size** | **Explanation** |
| BIT | Very small integer value that is equivalent to TINYINT(1). Signed values range from -128 to 127. Unsigned values range from 0 to 255. |  |
| TINYINT(m) | Very small integer value. Signed values range from -128 to 127. Unsigned values range from 0 to 255. |  |
| SMALLINT(m) | Small integer value. Signed values range from -32768 to 32767. Unsigned values range from 0 to 65535. |  |
| MEDIUMINT(m) | Medium integer value. Signed values range from -8388608 to 8388607. Unsigned values range from 0 to 16777215. |  |
| INT(m) | Standard integer value. Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295. |  |
| BIGINT(m) | Big integer value. Signed values range from -9223372036854775808 to 1. Unsigned values range from 0 to 18446744073709551615. |  |
| DECIMAL(m,d) | Unpacked fixed point number. m defaults to 10, if not specified. d defaults to 0, if not specified. | Where m is the total digits and d is the number ofdigits after the decimal. |
| FLOAT(m,d) | Single precision floating point number. | Where m is the total digits and d is the number ofdigits after the decimal. |
| DOUBLE(m,d) | Double precision floating point number. | Where m is the total digits and d is the number ofdigits after the decimal. |
| BOOLEAN | Synonym for TINYINT(1) | Treated as a boolean data type where a value of 0 is considered to be FALSE and any other value isconsidered to be TRUE. |
|  |  |  |

**Date/Time Datatypes**

The following are the **Date/Time Datatypes** in My :

|  |  |  |
| --- | --- | --- |
| **Data Type Syntax** | **Maximum Size** | **Explanation** |
| DATE | Values range from '1000-01-01' to '9999-12-31'. | Displayed as 'YYYY-MM-DD'. |
| DATETIME | Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. | Displayed as 'YYYY-MM-DD HH:MM:SS'. |
| TIMESTAMP(m) | Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC. | Displayed as 'YYYY-MM-DD HH:MM:SS'. |
| TIME | Values range from '-838:59:59' to '838:59:59'. | Displayed as 'HH:MM:SS'. |
| YEAR[(2|4)] | Year value as 2 digits or 4 digits. | Default is 4 digits. |

**Large Object (LOB) Datatypes**

The following are the **LOB Datatypes** in My :

|  |  |  |
| --- | --- | --- |
| Data Type Syntax | Maximum Size | Explanation |
| TINYBLOB | Maximum size of 255 bytes. |  |
| BLOB(size) | Maximum size of 65,535 bytes. | Where size is the number of characters to store. *(size is optional and was introduced in My 4.1)* |
| MEDIUMBLOB | Maximum size of 16,777,215 bytes. |  |
| LONGTEXT | Maximum size of 4GB or 4,294,967,295 characters. |  |

# Storing positive numbers only

If we have to store positive numbers only we can increase the range of numbers by using

Int unsigned / tinyint unsigned

COMMAND TYPES :



# Database related queries

IF NOT EXITS IS USED TO AVOID ERRORS I.E IF WE TRY TO CREATE A DATABASE BUT IT ALREADY EXISTS SO THEIR WILL A WARNING AND QUERY WILL BE EXECUTED;

## 

Similary drop If exists is used.

# Primary and Foreign key ;

### **What is a Primary Key in ?**

A Primary key is a unique column we set in a table to easily identify and locate data in queries. A table can have only one primary key.

The primary key column has a unique value and doesn’t store repeating values. A Primary key can never take NULL values.

**For example,** in the case of a student when identification needs to be done in the class, the roll number of the student plays the role of Primary key.

Similarly, when we talk about employees in a company, the employee ID is functioning as the Primary key for identification.

Let us now understand the Syntax of creating the table with the Primary key specified.

**Syntax:**

CREATE TABLE tableName (

col1 int NOT **NULL**,

col2 varchar(50) NOT **NULL**,

col3 int,

…………….

PRIMARY KEY (col1)

);

Note :

**A combination of two columns can also be made a primary key . This is because id or name of two students may or may not be same but combinations of their id and name will always be unique.**

### **What is a Foreign key in ?**

A Foreign key is beneficial when we connect two or more tables so that data from both can be put to use parallelly.

A foreign key is a field or collection of fields in a table that refers to the Primary key of the other table. It is responsible for managing the relationship between the tables.

The table which contains the foreign key is often called the child table, and the table whose primary key is being referred by the foreign key is called the Parent Table.

**For example:** When we talk about students and the courses they have enrolled in, now if we try to store all the data in a single table, the problem of redundancy arises.

To solve this table, we make two tables, one the student detail table and the other department table. In the student table, we store the details of students and the courses they have enrolled in.

And in the department table, we store all the details of the department. Here the courseId acts as the Primary key for the department table whereas it acts as the Foreign key in the student table.

Let us now look at the syntax of creating a table with a foreign key.

**Syntax:**

CREATE TABLE childTable (

col1 int NOT **NULL**,

col2 int NOT **NULL**,

col3 int,

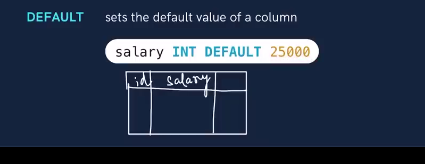
………...

PRIMARY KEY (col1),

FOREIGN KEY (col3) REFERENCES parentTable(parent\_Primary\_key)

);

If we don’t insert a value a default value is inserted if we use the default keyword;



# Playing with the table

## 1.Altering

### 1. Rename a Column (and optionally change its data type)

If you want to rename a column, use the CHANGE clause:

ALTER TABLE table\_name

CHANGE old\_column\_name new\_column\_name new\_data\_type;

Example: Renaming a column from username to user\_name:

ALTER TABLE users

CHANGE username user\_name VARCHAR(100);

### 2. Modify a Column's Data Type

If you only want to change the data type of the column without renaming it, use MODIFY:

ALTER TABLE table\_name

MODIFY column\_name new\_data\_type;

Example: Changing a column age from INT to BIGINT:

ALTER TABLE users

MODIFY age BIGINT;

### 3. Add or Remove Constraints

To add or remove a constraint (e.g., a NOT NULL or DEFAULT constraint), you can use MODIFY as well. For example, to make a column NOT NULL:

ALTER TABLE users

MODIFY email VARCHAR(255) NOT NULL;

## 2. RETRIEVING

To retrieve data based on a specific id (which is usually the primary key), you can use the SELECT statement in . The syntax is:

SELECT column\_name(s)

FROM table\_name

WHERE primary\_key\_column = primary\_key\_value;

### Example:

If you want to get the name and age from the Parent table where id = 1, you can write:

SELECT name, age

FROM Parent

WHERE id = 1;

If you want to retrieve all columns for that specific id, use \*:

SELECT \*

FROM Parent

WHERE id = 1;

This query will return the row where the id is 1. Let me know if you need further assistance!

## Retrieving more values based on id

**Select \* from tablename where Id in (id1, id2… ,id n);**

## 3. INSERTING /updating

To change or update the value of an attribute (column) in a row based on a primary key, you can use the UPDATE statement. The syntax for updating a value in a specific row is:

UPDATE table\_name

SET column\_name = new\_value

WHERE primary\_key\_column = primary\_key\_value;

### Example:

Suppose you have a table Parent with columns id (which is the primary key) and name. If you want to change the name of the row where id = 1 to "Ahmed," you would do the following:

UPDATE Parent

SET name = 'Ahmed'

WHERE id = 1;

### Multiple Updates:

You can also update multiple columns in the same row by separating them with commas:

UPDATE table\_name

SET column1 = value1, column2 = value2

WHERE primary\_key\_column = primary\_key\_value;

Example: Updating both name and another column, age, in the Parent table where id = 1:

UPDATE Parent

SET name = 'Ahmed', age = 35

WHERE id = 1;

Make sure to always specify the WHERE condition; otherwise, the UPDATE statement will modify all rows in the table.

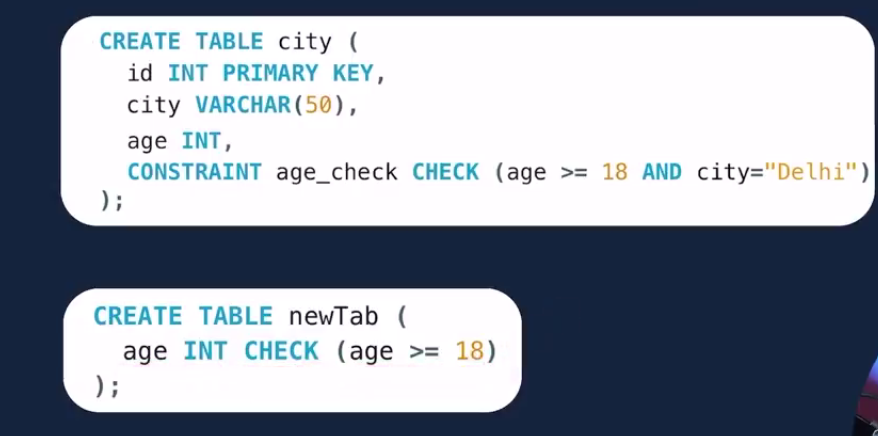
# Constraint in tables

We can add constraints on single or multiple columns.

## Table constraints

For table constraints we define them inside the table as follows.

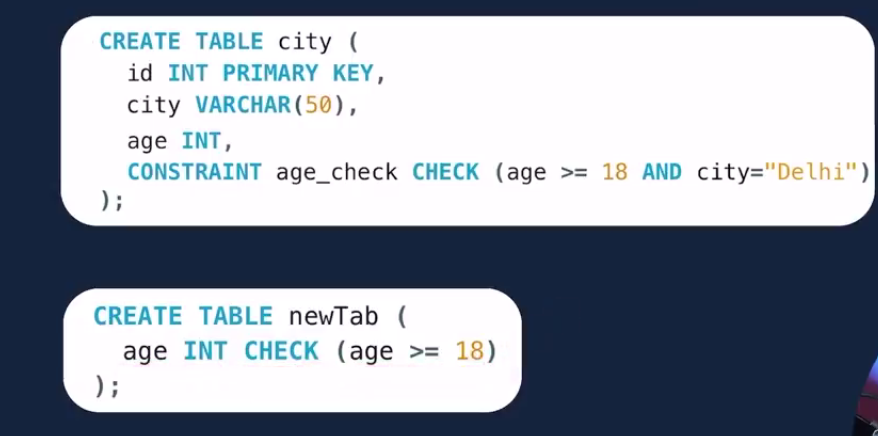
Constraint constraint\_name check ( condition(s) );



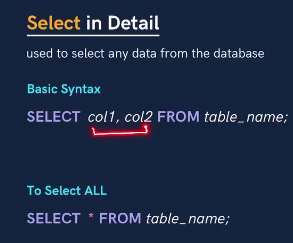
## Row constraint

In case of row constraints we define the constraint in the same row as follows ;

City varchar(10) check (city=”Washington DC” );



# Select



### Distinct :

Select distinct city from students ;

gives the distinct city names;

# where clause

it is used to select based on special condtion;

example :

select \* from students where marks > 80 and city=”Multan” ;

we can use

1. Comparison
2. Arithmetic
3. Logical operators
4. Bitwise operators