# MySQL - ALTER Command


**MySQL Database Training For Beginners**

39 Lectures 5.5 hours

Simon Sez IT

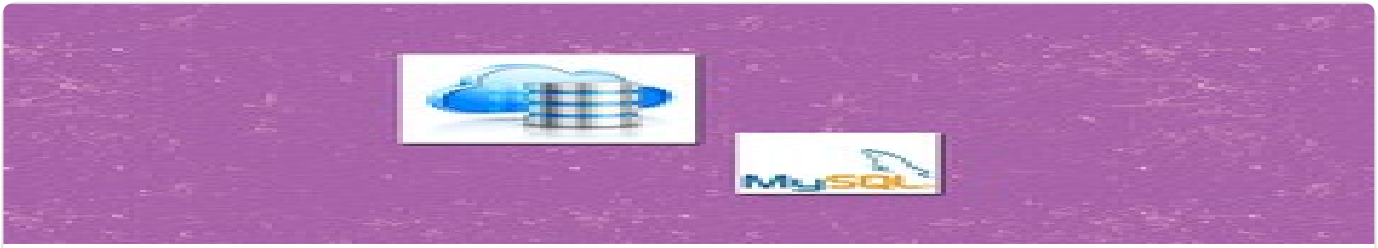
More Detail


**Python Programming With MySQL Database: From Scratch**

152 Lectures 16 hours

Metla Sudha Sekhar

More Detail




**Learn MySQL From Scratch For Data Science And Analytics**

87 Lectures 5.5 hours

Metla Sudha Sekhar

More Detail

The MySQL **ALTER** command is very useful when you want to change a name of your table, any table field or if you want to add or delete an existing column in a table.

Let us begin with the creation of a table called **testalter_tbl**.

```
root@host# mysql -u root -p password;
Enter password:*******

mysql> use TUTORIALS;
Database changed

mysql> create table testalter_tbl
   -> (
   -> i INT,
   -> c CHAR(1)
   -> );
Query OK, 0 rows affected (0.05 sec)
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   i   | int(11) | YES  |     |   NULL  |       |
|   c   | char(1) | YES  |     |   NULL  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

## Dropping, Adding or Repositioning a Column

If you want to drop an existing column i from the above MySQL table, then you will use the **DROP** clause along with the **ALTER** command as shown below −

```
mysql> ALTER TABLE testalter_tbl  DROP i;
```

A **DROP** clause will not work if the column is the only one left in the table.

To add a column, use ADD and specify the column definition. The following statement restores the **i** column to the testalter_tbl −

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

After issuing this statement, testalter will contain the same two columns that it had when you first created the table, but will not have the same structure. This is because there are new colun

that are added to the end of the table by default. So even though **i** originally was the first column in mytbl, now it is the last one.

```
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   c   | char(1) | YES  |     |   NULL  |       |
|   i   | int(11) | YES  |     |   NULL  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

To indicate that you want a column at a specific position within the table, either use FIRST to make it the first column or **AFTER col_name** to indicate that the new column should be placed after the col_name.

Try the following **ALTER TABLE** statements, using **SHOW COLUMNS** after each one to see what effect each one has −

```
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT FIRST;
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

The FIRST and AFTER specifiers work only with the ADD clause. This means that if you want to reposition an existing column within a table, you first must **DROP** it and then **ADD** it at the new position.

## Altering (Changing) a Column Definition or a Name

To change a column's definition, use **MODIFY** or **CHANGE** clause along with the ALTER command.

For example, to change column **c** from CHAR(1) to CHAR(10), you can use the following command −

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

With **CHANGE**, the syntax is a bit different. After the CHANGE keyword, you name the column you want to change, then specify the new definition, which includes the new name.

Try out the following example −

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

If you now use CHANGE to convert **j** from **BIGINT** back to **INT** without changing the column name, the statement will be as shown below −

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

**The Effect of ALTER TABLE on Null and Default Value Attributes** − When you MODIFY or CHANGE a column, you can also specify whether or not the column can contain NULL values and what its default value is. In fact, if you don't do this, MySQL automatically assigns values for these attributes.

The following code block is an example, where the **NOT NULL** column will have the value as 100 by default.

```
mysql> ALTER TABLE testalter_tbl
   -> MODIFY j BIGINT NOT NULL DEFAULT 100;
```

If you don't use the above command, then MySQL will fill up NULL values in all the columns.

## Altering (Changing) a Column's Default Value

You can change a default value for any column by using the **ALTER** command.

Try out the following example.

```
mysql> ALTER TABLE testalter_tbl ALTER i SET DEFAULT 1000;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   c   | char(1) | YES  |     |   NULL  |       |
|   i   | int(11) | YES  |     |   1000  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

You can remove the default constraint from any column by using DROP clause along with the **ALTER** command.

```
mysql> ALTER TABLE testalter_tbl ALTER i DROP DEFAULT;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field |  Type   | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
|   c   | char(1) | YES  |     |   NULL  |       |
|   i   | int(11) | YES  |     |   NULL  |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

## Altering (Changing) a Table Type

You can use a table type by using the **TYPE** clause along with the ALTER command. Try out the following example to change the **testalter_tbl** to **MYISAM** table type.

To find out the current type of a table, use the **SHOW TABLE STATUS** statement.

```
mysql> ALTER TABLE testalter_tbl TYPE = MYISAM;
mysql>  SHOW TABLE STATUS LIKE 'testalter_tbl'\G
*************************** 1. row ***************
           Name: testalter_tbl
           Type: MyISAM
     Row_format: Fixed
           Rows: 0
 Avg_row_length: 0
    Data_length: 0
Max_data_length: 25769803775
   Index_length: 1024
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2007-06-03 08:04:36
    Update_time: 2007-06-03 08:04:36
     Check_time: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

## Renaming (Altering) a Table

To rename a table, use the **RENAME** option of the **ALTER TABLE** statement.

Try out the following example to rename **testalter_tbl** to **alter_tbl**.

```
mysql> ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

You can use the ALTER command to create and drop the INDEX command on a MySQL file. We will discuss in detail about this command in the next chapter.