# **MySQL Primary Key**

**Summary**: in this tutorial, you will learn how to use the **MySQL primary key** constraint to create the primary key for a table.

## **Introduction to the MySQL primary key**

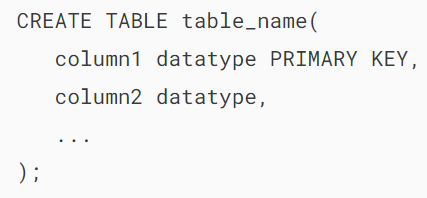
In MySQL, a primary key is a column or a set of columns that uniquely identifies each row in the table. A primary key column must contain unique values.

If the primary key consists of multiple columns, the combination of values in these columns must be unique. Additionally, a primary key column cannot contain NULL.

A table can have either zero or one primary key, but not more than one.

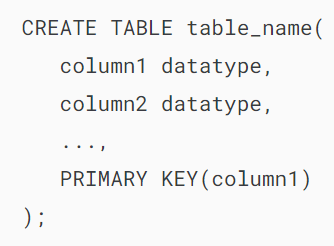
### **Defining a single-column primary key**

Typically, you define a primary key for a table when you [create the table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/). Here’s the syntax for defining the primary key that consists of one column:



In this syntax, you define the PRIMARY KEY constraint as a column constraint.

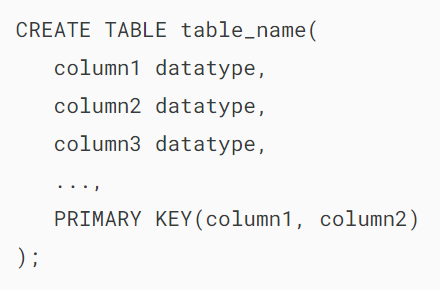
Additionally, you can put the PRIMARY KEY at the end of the column list:



In this syntax, you define the PRIMARY KEY constraint as a table constraint.

### **Defining a multi-column primary key**

If the primary key consists of two or more columns, you need to use a table constraint to define the primary key:



In this syntax, you list the primary key columns inside parentheses, separated by commas, followed by the PRIMARY KEY keywords.

### **Adding a primary key to an existing table**

If an existing table does not have a primary key, you can add a primary key to the table using the ALTER TABLE ... ADD PRIMARY KEY statement:

### 

### **Removing a primary key**

In practice, you’ll rarely remove a primary key. However, if you want to do so, you can use the ALTER TABLE ... DROP PRIMARY KEY statement:

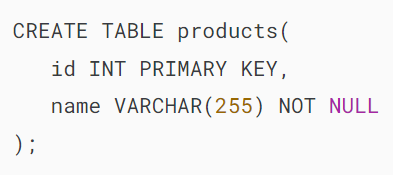
## 

## **MySQL PRIMARY KEY examples**

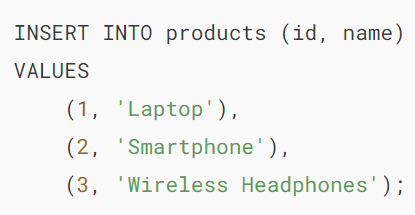
We’ll explore some examples of defining primary keys.

### **1) Defining a single-column primary key example**

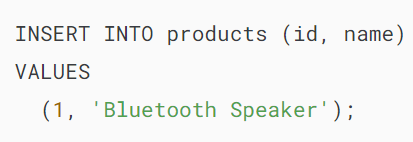
The following example [creates a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) called products, which has the id column as the primary key:



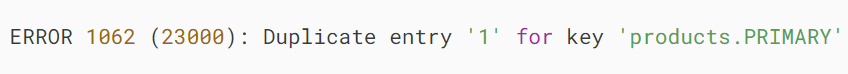
When you [insert data](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into the products table, you need to ensure the uniqueness of values in the id column. For example:



If you attempt to insert a duplicate value into the primary key column, you’ll get an error. For example:



Error:

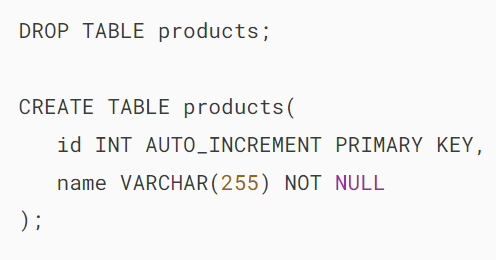


The output indicates that MySQL found a duplicate entry 1 for the primary key of the products table.

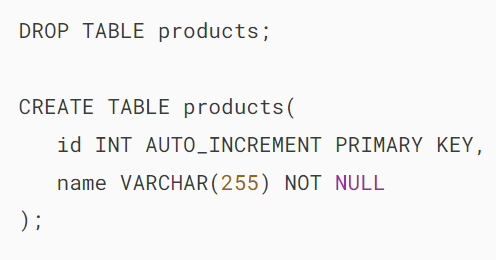
Keeping track of primary key values manually can be challenging. To simplify this process, MySQL provides the [AUTO\_INCREMENT](https://www.mysqltutorial.org/mysql-basics/mysql-auto_increment/) attribute, which automatically assigns a unique value to the primary key each time you insert a new record.

### **2) Defining a single-column primary key with AUTO\_INCREMENT attribute example**

The following statements re-create the products table with the primary key that uses the AUTO\_INCREMENT attribute:



Now, you can insert new rows into the products table without having to provide the values for the primary key column. For example:

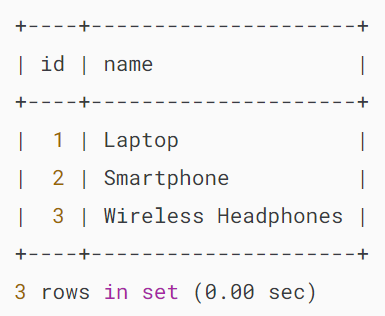


MySQL automatically generates sequential integer values for the id column when a new row is inserted.

Here’s the contents of the products table:

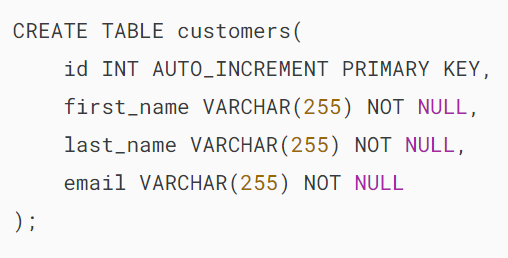


Output:



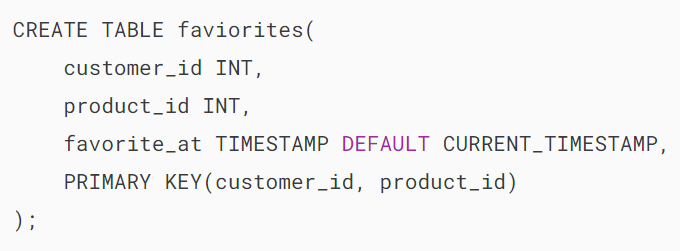
### **3) Defining a multi-column primary key example**

We’ll create a new table called customers:



Suppose each customer has some favorite products and each product is favored by some customers.

To model this relationship, you need to create a table called favorites:

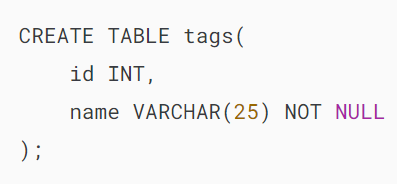


The favorites table has a primary that consists of two columns customer\_id and product\_id.

Note that in the [foreign key tutorial](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/), you’ll learn how to define a foreign key for the customer\_id column that references the id column of the customers table and a foreign key for the product\_id column that references the id column of the products table.

### **4) Adding a primary key to a table example**

The following statement creates a table called tags without a primary key:



To make the id column the primary key, you use the ALTER TABLE ... ADD PRIMARY KEY statement:

### 

### **5) Removing the primary key from a table**

The following statement removes the primary key from the tags table:

## 

## **Summary**

* A primary key is a unique identifier for a row in a table.
* Use the PRIMARY KEY constraint to define a primary key for a table.

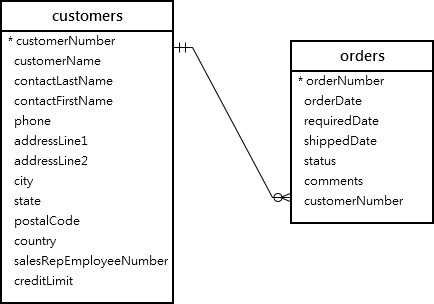
# **MySQL Foreign Key**

**Summary**: in this tutorial, you will learn about **MySQL foreign key** and how to create, drop, and disable a foreign key constraint.

## **Introduction to MySQL foreign key**

A foreign key is a column or group of columns in a table that links to a column or group of columns in another table. The foreign key places constraints on data in the related tables, which allows MySQL to maintain referential integrity.

Let’s take a look at the following customers and orders tables from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).



In this diagram, each customer can have zero or many orders and each order belongs to one customer.

The relationship between customers table and orders table is **one-to-many**. This relationship is established via the foreign key in the orders table specified by the customerNumber column.

The customerNumber column in the orders table links to the customerNumber primary key column in the customers table.

The customers table is called the parent table or referenced table, and the orders table is known as the child table or referencing table.

Typically, the foreign key columns of the child table often refer to the [primary key](https://www.mysqltutorial.org/mysql-basics/mysql-primary-key/) columns of the parent table.

A table can have more than one foreign key where each foreign key references a primary key of the different parent tables.

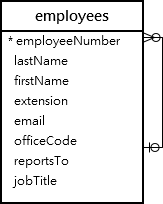
Once a foreign key constraint is in place, the foreign key columns from the child table must have the corresponding row in the parent key columns of the parent table, or values in these foreign key columns must be NULL (see the SET NULL action example below).

For example, each row in the orders table has a customerNumber that exists in the customerNumber column of the customers table. Multiple rows in the orders table can have the same customerNumber.

### **Self-referencing foreign key**

Sometimes, the child and parent tables may refer to the same table. In this case, the foreign key references back to the primary key within the same table.

See the following employees table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).



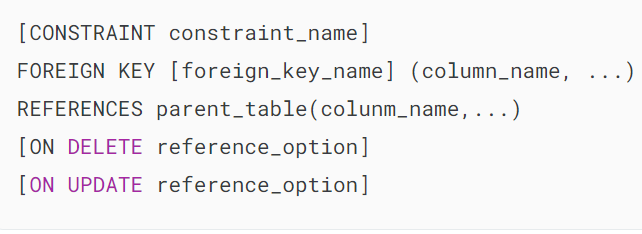
The reportTo column is a foreign key that refers to the employeeNumber column which is the primary key of the employees table.

This relationship allows the employees table to store the reporting structure between employees and managers. Each employee reports to zero or one employee and an employee can have zero or many subordinates.

The foreign key on the column reportTo is known as a recursive or self-referencing foreign key.

## **MySQL FOREIGN KEY syntax**

Here is the basic syntax of defining a foreign key constraint in the [CREATE TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) or [ALTER TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/) statement:



In this syntax:

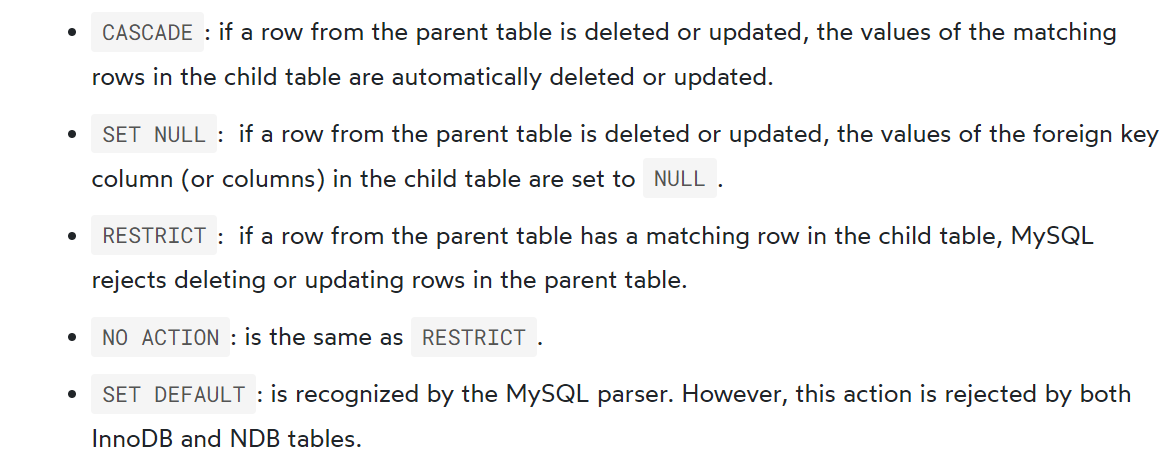
First, specify the name of the foreign key constraint that you want to create after the CONSTRAINT keyword. If you omit the constraint name, MySQL automatically generates a name for the foreign key constraint.

Second, specify a list of comma-separated foreign key columns after the FOREIGN KEY keywords. The foreign key name is also optional and is generated automatically if you skip it.

Third, specify the parent table followed by a list of comma-separated columns to which the foreign key columns reference.

Finally, specify how the foreign key maintains the referential integrity between the child and parent tables by using the ON DELETE and ON UPDATE clauses. The reference\_option determines the action that MySQL will take when values in the parent key columns are deleted (ON DELETE) or updated (ON UPDATE).

MySQL has five reference options: CASCADE, SET NULL, NO ACTION, RESTRICT, and SET DEFAULT.



MySQL fully supports three actions: RESTRICT, CASCADE and SET NULL.

If you don’t specify the ON DELETE and ON UPDATE clause, the default action is RESTRICT.

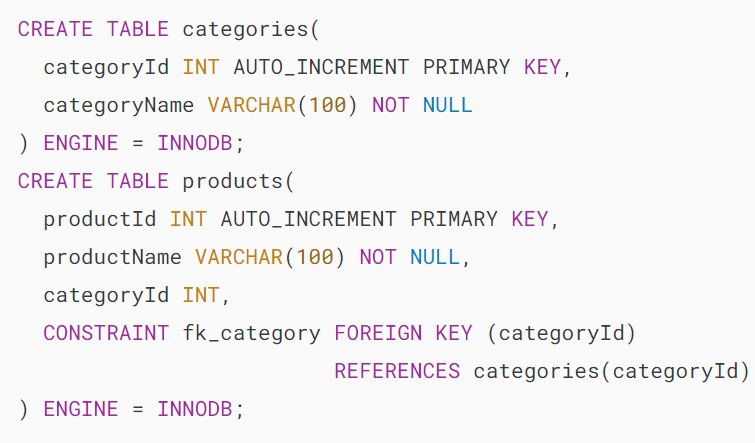
## **MySQL FOREIGN KEY examples**

Let’s [create a new database](https://www.mysqltutorial.org/mysql-basics/mysql-create-database/) called fkdemo for the demonstration.

### 

### **1) RESTRICT & NO ACTION actions**

Inside the fkdemo database, create two tables categories and products:



The categoryId in the products table is the foreign key column that refers to the categoryId column in the  categories table.

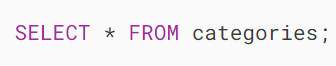
Because we don’t specify any ON UPDATE and ON DELETE clauses, the default action is RESTRICT for both update and delete operations.

The following steps illustrate the RESTRICT action.

1) [Insert two rows](https://www.mysqltutorial.org/mysql-basics/mysql-insert-multiple-rows/) into the categories table:

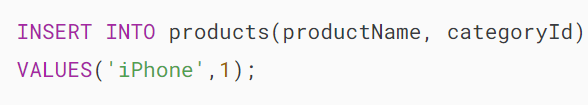


2) [Select](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) data from the categories table:



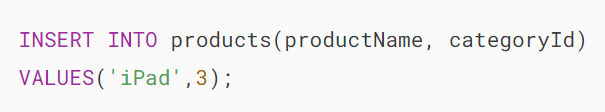


3) [Insert a new row](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into the products table:

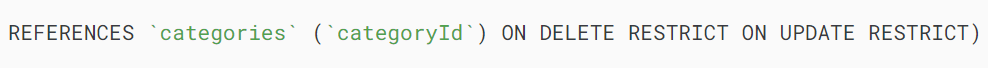
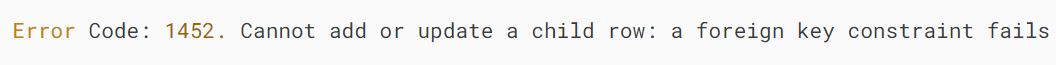


It works because the categoryId 1 exists in the categories table.

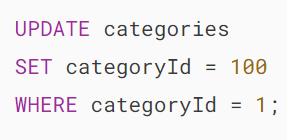
4) Attempt to insert a new row into the products table with a categoryId  value does not exist in the categories table:



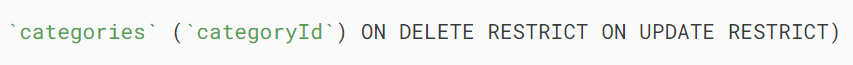
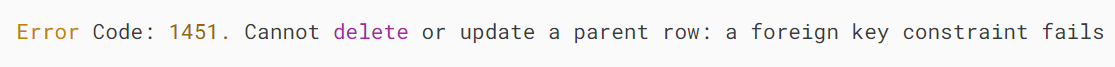
MySQL issued the following error:



5) Update the value in the categoryId column in the categories table to 100:



MySQL issued this error:



Because of the RESTRICT option, you cannot delete or update categoryId 1 since it is referenced by the productId 1 in the products table.

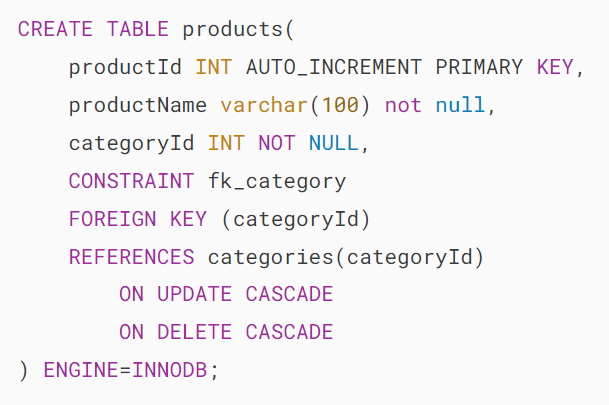
### **2) CASCADE action**

These steps illustrate how ON UPDATE CASCADE and [ON DELETE CASCADE](https://www.mysqltutorial.org/mysql-basics/mysql-on-delete-cascade/) actions work.

1) [Drop](https://www.mysqltutorial.org/mysql-drop-table) the products table:



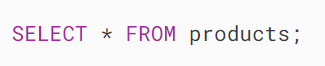
2) Create the products table with the ON UPDATE CASCADE and ON DELETE CASCADE options for the foreign key:

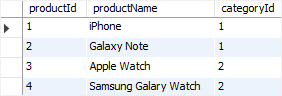


3) Insert four rows into the products table:

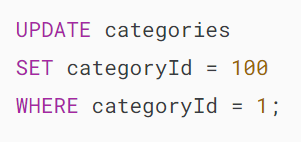


4) Select data from the products table:





5) Update categoryId 1 to 100 in the categories table:

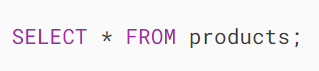


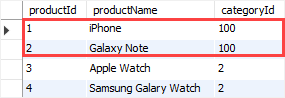
6) Verify the update:





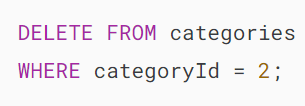
7) Get data from the products table:



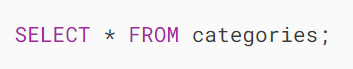


As you can see, two rows with value 1 in the categoryId column of the products table was automatically updated to 100 because of the ON UPDATE CASCADE action.

8) Delete categoryId 2 from the categories table:



9) Verify the deletion:





10) Check the products table:



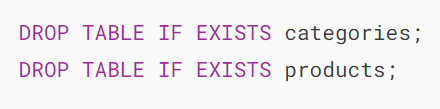


All products with categoryId 2 from the products table was automatically deleted because of the ON DELETE CASCADE action.

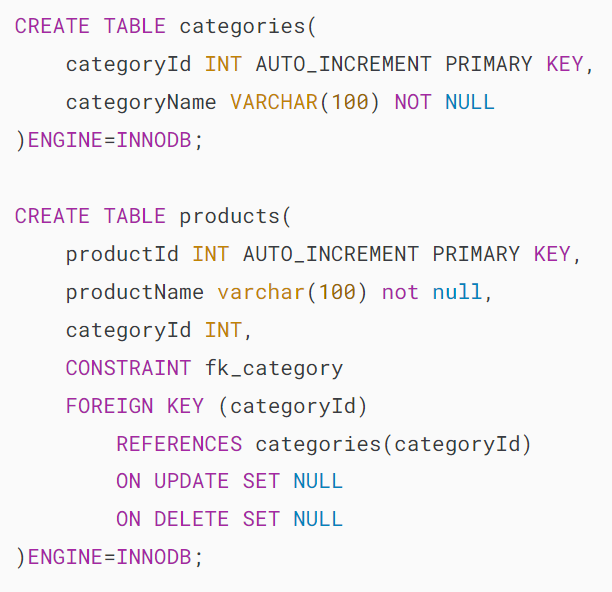
### **3) SET NULL action**

These steps illustrate how the ON UPDATE SET NULL and ON DELETE SET NULL actions work.

1) Drop both categories and products tables:

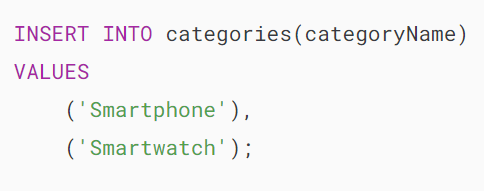


2) Create the categories and products tables:

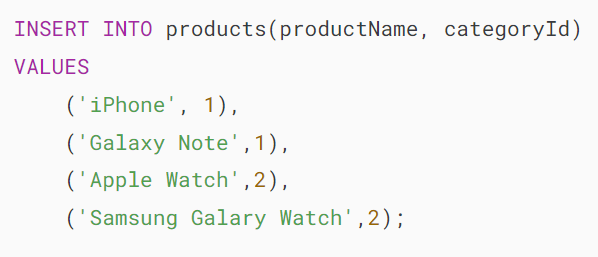


The foreign key in the products table changed to ON UPDATE SET NULL and ON DELETE SET NULL options.

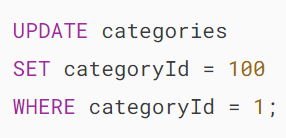
3) Insert rows into the categories table:



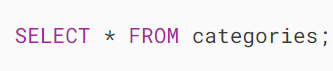
4) Insert rows into the products table:



5) Update categoryId from 1 to 100 in the categories table:

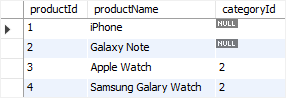


6) Verify the update:



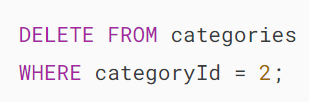


7) Select data from the products table:



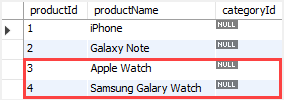
The rows with the categoryId 1 in the products table was automatically set to NULL due to the ON UPDATE SET NULL action.

8) Delete the categoryId 2 from the categories table:



9) Check the products table:

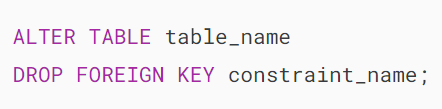




The values in the categoryId column of the rows with categoryId 2 in the products table was automatically set to NULL due to the ON DELETE SET NULL action.

## **Drop MySQL foreign key constraints**

To drop a foreign key constraint, you use the ALTER TABLE statement:



In this syntax:

* First, specify the name of the table from which you want to drop the foreign key after the ALTER TABLE keywords.
* Second, specify  the constraint name after the DROP FOREIGN KEY keywords.

Notice that constraint\_name is the name of the foreign key constraint specified when you created or added the foreign key constraint to the table.

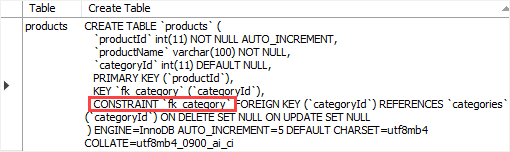
To obtain the generated constraint name of a table, you use the SHOW CREATE TABLE statement:



For example, to see the foreign keys of the products table, you use the following statement:

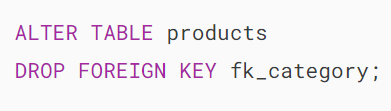


The following is the output of the statement:



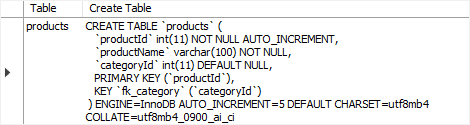
As you can see clearly from the output, the table products table has one foreign key constraint: fk\_category

This statement drops the foreign key constraint of the products table:



To ensure that the foreign key constraint has been dropped, you can view the structure of the products table:





## **Disabling foreign key checks**

Sometimes, it is very useful to disable foreign key checks e.g., when you [import data from a CSV file into a table](https://www.mysqltutorial.org/mysql-basics/import-csv-file-mysql-table/).

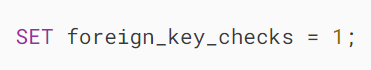
If you don’t disable foreign key checks, you have to load data into a proper order i.e., you have to load data into parent tables first and then child tables, which can be tedious.

However, if you disable the foreign key checks, you can load data into tables in any order.

To disable foreign key checks, you use the following statement:



And you can enable it by using the following statement:



# **MySQL Disable Foreign Key Checks**

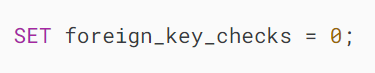
**Summary**: in this tutorial, you will learn how to disable foreign key constraint checks in MySQL.

Sometimes, it is very useful to disable [foreign key](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/) checks. For example, you can load data to the parent and child tables in any order with the foreign key constraint check disabled.

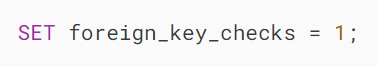
If you don’t disable foreign key checks, you have to load data into the parent tables first and then the child tables in sequence, which can be tedious.

Another scenario in which you want to disable the foreign key check is when you want to [drop a table](https://www.mysqltutorial.org/mysql-drop-table). Unless you disable the foreign key checks, you cannot [drop a table](https://www.mysqltutorial.org/mysql-drop-table) referenced by a foreign key constraint.

To disable foreign key checks, you set the foreign\_key\_checks variable to zero as follows:



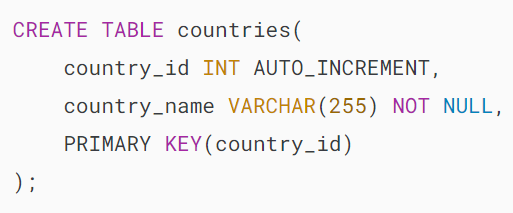
To enable the foreign key constraint check, you set the value of the foreign\_key\_checks to 1:



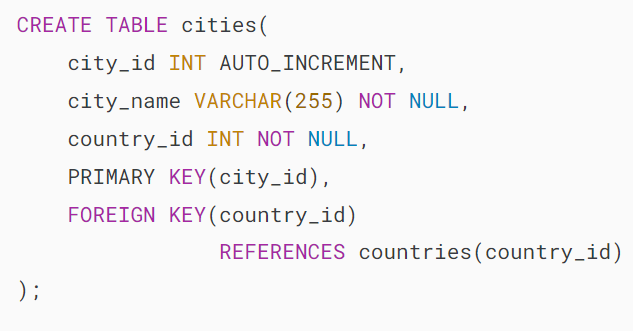
Notice that setting foreign\_key\_checks to 1 does not trigger any validation of the existing table data. In other words, MySQL will not verify the consistency of the data that was added during the foreign key check disabled.

## **Disable foreign key check example**

First, [create a new table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named countries:

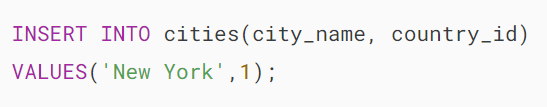


Second, [create another table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named cities:



The table cities has a [foreign key constraint](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/) that references the column country\_id of the table countries.

Third, [insert a new row](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into the cities table:



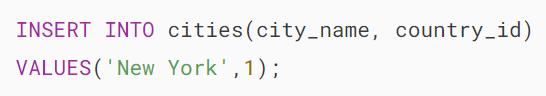
MySQL issued the following error:



Fourth, disable foreign key checks:

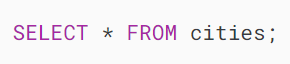


Fifth, insert a new row into the cities table:



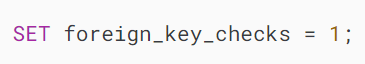
This time the INSERT statement executed successfully due to the foreign key check being disabled.

The following [query](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) returns the contents of the table cities:



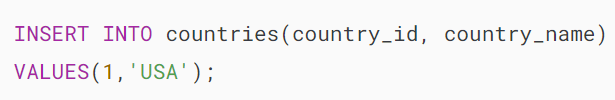
MySQL Disable Foreign Key Checks example

Sixth, re-enable foreign key constraint check:



When the foreign key checks were re-enabled, MySQL did not re-validate data in the table. However, it won’t allow you to insert or update data that violates the foreign key constraint.

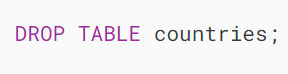
Finally, insert a row into the countries table whose value in the column country\_id is 1 to make the data consistent in both tables:



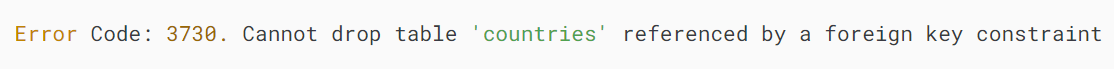
## **Drop tables that have foreign key constraints**

Suppose that you want to drop the countries and cities tables.

First, drop the table countries :



MySQL issued this error:

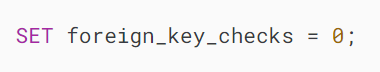


To fix this, you have two options:

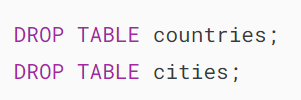
* Drop the table cities first and then remove the table countries.
* Disable foreign key checks and drop tables in any sequence.

We’ll demonstrate the second way which disables the foreign key constraint check before dropping the tables.

Second, disable the foreign key check:



Third, drop both tables countries and cities:



Both statements were executed successfully.

Finally, enable the foreign key check:

## 

## **Summary**

* Use the SET foreign\_key\_checks = 0 to disable foreign key checks in MySQL.

# **MySQL UNIQUE Constraint**

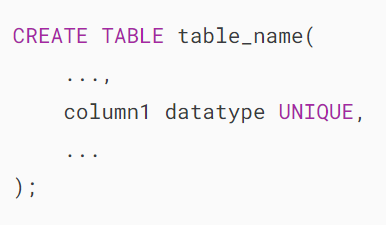
**Summary**: in this tutorial, you will learn about MySQL UNIQUE constraint and how to use it to enforce the uniqueness of values in a column or a group of columns in a table.

## **Introduction to MySQL UNIQUE constraint**

Sometimes, you want to ensure values in a column or a group of columns are unique. For example, email addresses of users in the users table, or phone numbers of customers in the customers table should be unique. To enforce this rule, you use a UNIQUE constraint.

A UNIQUE constraint is an integrity constraint that ensures the uniqueness of values in a column or group of columns. A UNIQUE constraint can be either a column constraint or a table constraint.

To define a UNIQUE constraint for a column when [creating a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/), you use the following syntax:

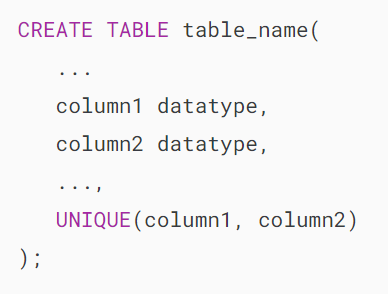


In this syntax, you include the UNIQUE keyword in the definition of the column that you want to enforce the uniqueness.

If you [insert](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) or [update](https://www.mysqltutorial.org/mysql-basics/mysql-update/) a value that causes a duplicate in the column1, MySQL rejects the change and issues an error.

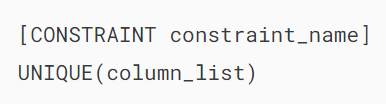
This UNIQUE constraint is a column constraint. And you can use it to enforce the unique rule for one column.

To define a UNIQUE  constraint for two or more columns, you use the following syntax:



In this syntax, you add a comma-separated list of columns in parentheses after the UNIQUE keyword. In this case, MySQL will use the combination of values in both columns column1 and column2 to evaluate the uniqueness.

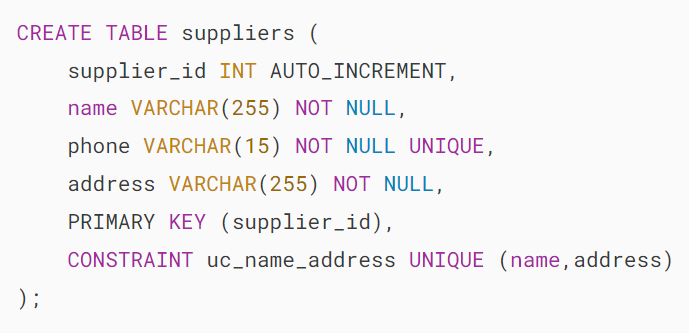
If you define a UNIQUE constraint without specifying a name, MySQL automatically generates a name for it. To define a UNIQUE constraint with a name, you use this syntax:



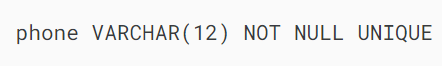
In this syntax, you specify the name of the UNIQUE constraint after the CONSTRAINT keyword.

## **MySQL UNIQUE constraint example**

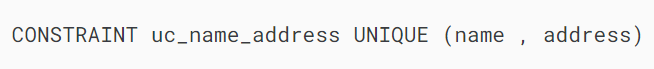
First, [creates a new table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named suppliers with the two UNIQUE constraints:



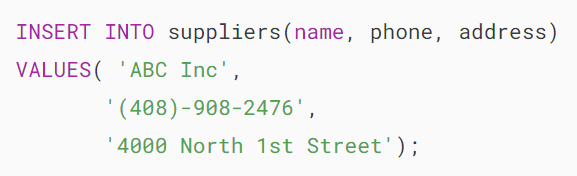
In this example, the first UNIQUE constraint is defined for the phone column:



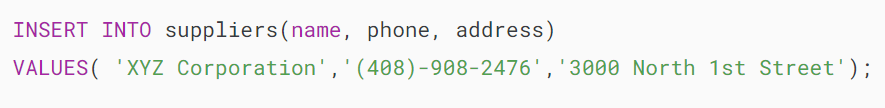
The second UNIQUE constraint includes both name and address columns:



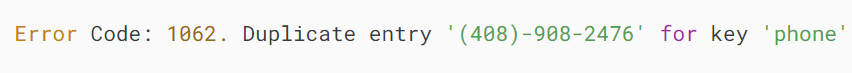
Second, [insert a row](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into the suppliers table:



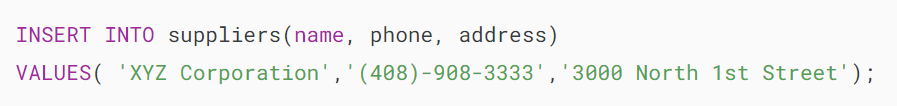
Third, attempt to insert a different supplier but has the phone number that already exists in the suppliers table.



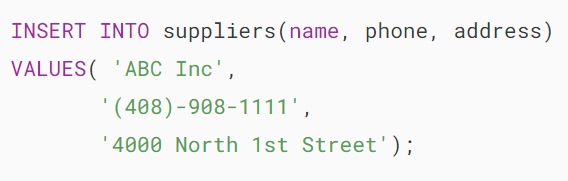
MySQL issued the following error:



Fourth, change the phone number to a different one and execute the insert statement again.



Fifth, insert a row into the suppliers table with values that already exist in the columns name and address :



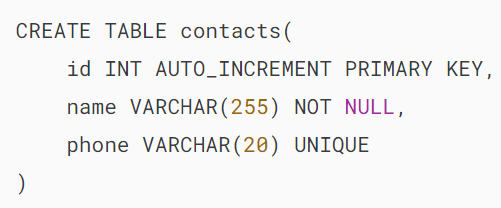
MySQL issued an error because the UNIQUE constraint uc\_name\_address was violated.

## 

## **MySQL UNIQUE constraint & NULL**

In MySQL, NULL values are treated as distinct when it comes to unique constraints. Therefore, if you have a column that accepts NULL values, you can insert multiple values into the column.

First, [create a new table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) called contacts:



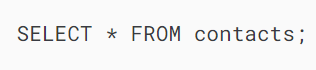
The contacts table has a phone column with a UNIQUE constraint. Also, the phone column can accept NULL values.

Second, insert some rows into the contacts table:



In this example, we can insert two NULL values into the phone column without causing a duplicate.

Third, retrieve data from the contacts table:



Output:

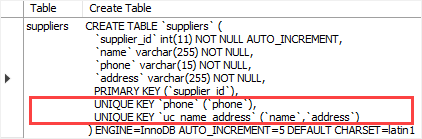
## 

## **MySQL UNIQUE constraints and indexes**

When you define a unique constraint for a column or a group of columns, MySQL creates a corresponding [UNIQUE index](https://www.mysqltutorial.org/mysql-unique/) and uses this index to enforce the rule.

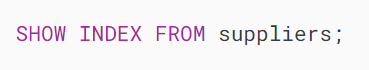
The SHOW CREATE TABLE statement shows the definition of the suppliers table:

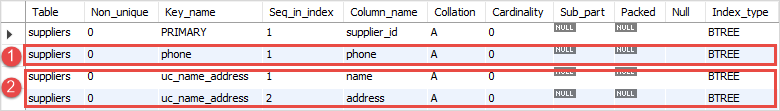




The output indicates that MySQL created two UNIQUE indexes on the suppliers table: phone and uc\_name\_address.

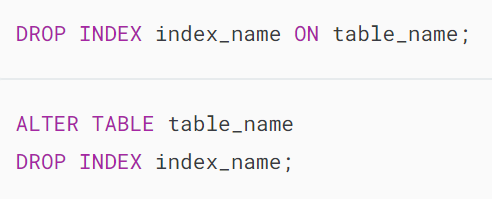
The following [SHOW INDEX](https://www.mysqltutorial.org/mysql-index/mysql-show-indexes/) statement displays all indexes associated with the suppliers table.



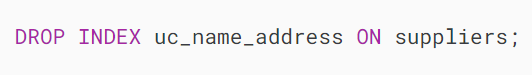


## **Drop a unique constraint**

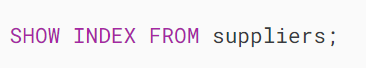
To drop a UNIQUE constraint, you can use [DROP INDEX](https://www.mysqltutorial.org/mysql-index/mysql-drop-index/) or [ALTER TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/) statement:

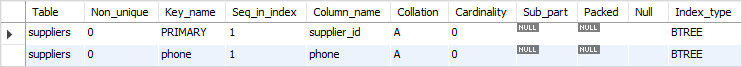


For example, the following statement drops the uc\_name\_address constraint on the suppliers table:



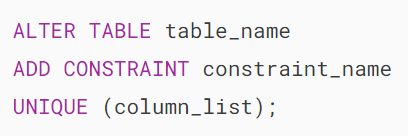
Execute the SHOW INDEX statement again to verify if the uc\_name\_unique constraint has been removed.



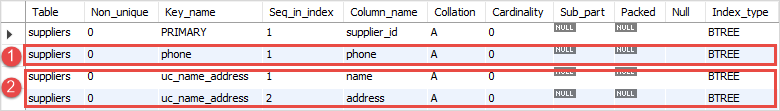
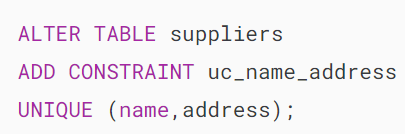


## **Add new unique constraint**

The following ALTER TABLE ADD CONSTRAINT adds a unique constraint to a column of an existing table:



This statement adds a UNIQUE constraint uc\_name\_address back to the suppliers table:



Note that MySQL will not add a unique constraint if the existing data in the columns of specified in the unique constraint does not comply with the uniqueness rule.

## **Summary**

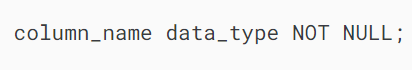
* Use MySQL UNIQUE constraint to enforce the uniqueness of values in a column or group of columns of a table.

# **MySQL NOT NULL Constraint**

**Summary**: in this tutorial, you will learn about MySQL NOT NULL constraints including defining a NOT NULL constraint for a column, adding a NOT NULL constraint to an existing column, and removing a NOT NULL constraint from a column.

## **Introduction to MySQL NOT NULL constraints**

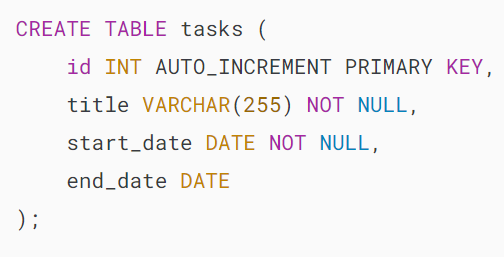
A NOT NULL constraint ensures that values stored in a column are not [NULL](https://www.mysqltutorial.org/mysql-basics/mysql-null/). The syntax for defining a NOT NULL constraint is as follows:



A column may have only one NOT NULL constraint, which enforces the rule that the column must not contain any NULL values.

In other words, if you attempt to [update](https://www.mysqltutorial.org/mysql-basics/mysql-update/) or [insert](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) a NULL value into a NOT NULL column, MySQL will issue an error.

For example, the following creates the tasks table using the CREATE TABLE statement:

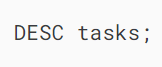


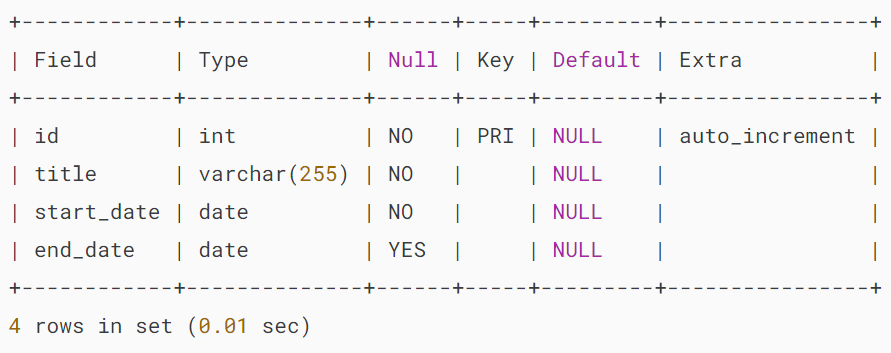
In the tasks table, we explicitly define the title and start\_date columns with NOT NULL constraints.

The id column has the [PRIMARY KEY](https://www.mysqltutorial.org/mysql-basics/mysql-primary-key/) constraint, therefore, it implicitly includes a NOT NULL constraint.

The end\_date column can have NULL values, as when creating a new task, you may not know its completion date

The following shows the structure of the tasks table:





It’s a good practice to have the NOT NULL constraint in every column of a table unless you have a specific reason not to.

Generally, NULL values may complicate your queries because you need to use NULL-related functions such as [ISNULL()](https://www.mysqltutorial.org/mysql-comparison-functions/mysql-isnull-function/), [IFNULL()](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-ifnull/), and [NULLIF()](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-nullif/) to handle them.

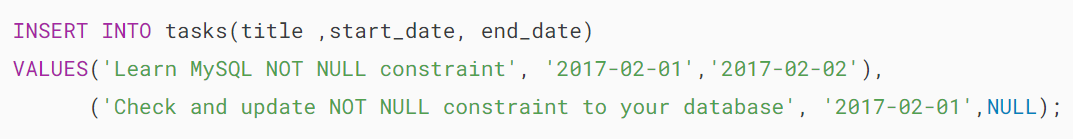
## **Adding a NOT NULL constraint to an existing column**

Typically, you add NOT NULL constraints to columns when you create the table. However, you may want to add a NOT NULL constraint to a column of an existing table. In this case, you use the following steps:

1. First, check the current values of the column if there are any NULL values.
2. Second, update the NULL to non-NULL.
3. Third, modify the column with a NOT NULL constraint.

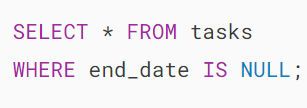
Consider the following example.

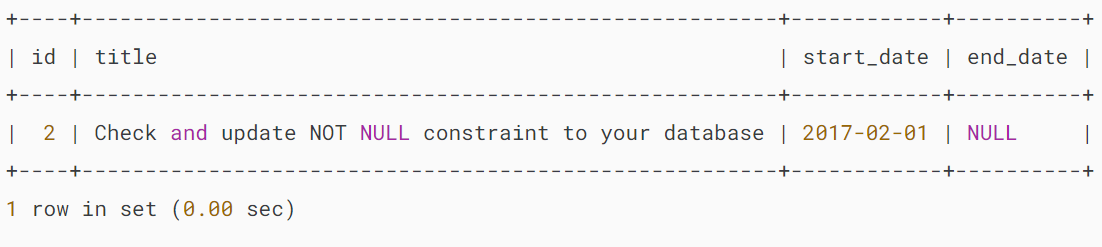
First, [insert some rows](https://www.mysqltutorial.org/mysql-basics/mysql-insert-multiple-rows/) into the tasks table:



If you want to require users to provide an estimated end date when creating a new task, you can add a NOT NULL constraint to the end\_date column of the tasks table.

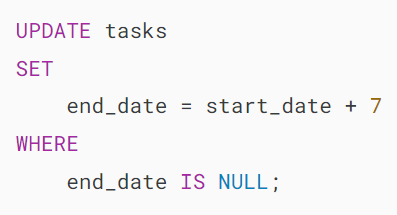
Second, find rows with NULLs in the column end\_date using the [IS NULL](https://www.mysqltutorial.org/mysql-basics/mysql-is-null/) operator:





The [query](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) returned one row with NULL in the column end\_date.

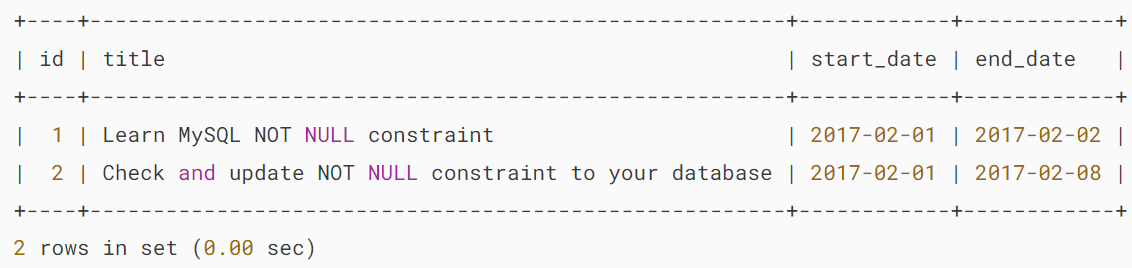
Third, [update](https://www.mysqltutorial.org/mysql-basics/mysql-update/) the NULL values to non-null values. In this case, you can create a rule that sets to one week after the start date when the end\_date is NULL.

’

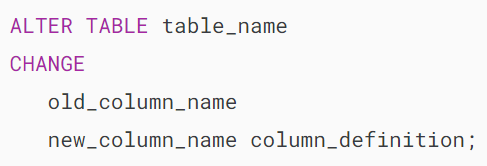
This query verifies the update:



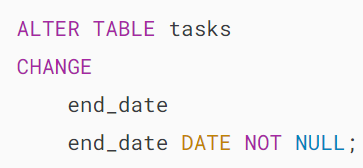
Output:



Third, add a NOT NULL constraint to the end\_date column using the following [ALTER TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/) statement:



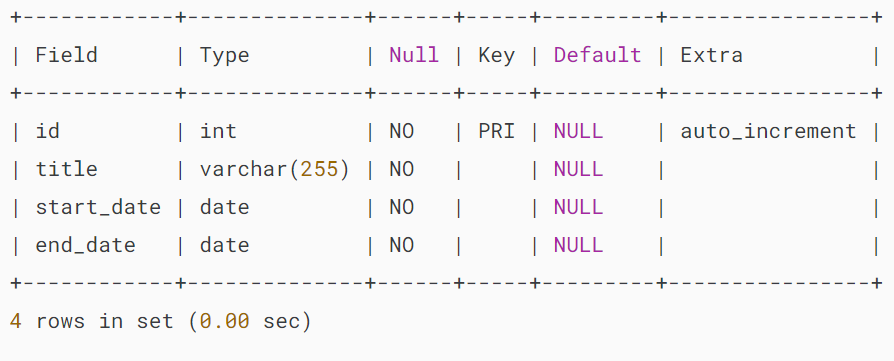
In this case, the name of the old and new column names are the same except that the column must have a NOT NULL constraint:



Finally, verify the change using the [DESCRIBE](https://www.mysqltutorial.org/mysql-administration/mysql-show-columns/) statement:



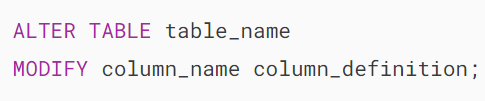
Output:



The output indicates that the NOT NULL constraint was added to the end\_date column successfully.

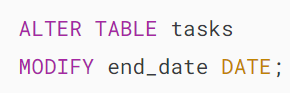
## **Removing a NOT NULL constraint**

To drop a NOT NULL constraint for a column, you use the ALTER TABLE..MODIFY statement:



Note that the column definition (column\_definition) must restate the original column definition without the NOT NULL constraint.

For example, the following statement removes the NOT NULL constraint from the end\_date column in the tasks table:



Here’s the structure of the tasks table:



Output:

## 

## **Summary**

* Use NOT NULL constraint to ensure that a column does not contain any NULL values.
* Use ALTER TABLE ... CHANGE statement to add a NOT NULL constraint to an existing column.
* Use ALTER TABLE ... MODIFY to drop a NOT NULL constraint from a column.

# **MySQL DEFAULT**

**Summary**: in this tutorial, you’ll learn about MySQL DEFAULT constraint and how to use it effectively.

## **Introduction to the MySQL DEFAULT constraint**

MySQL DEFAULT constraint allows you to specify a default value for a column. Here’s the syntax of the DEFAULT constraint:



In this syntax, you specify the DEFAULT keyword followed by the default value for the column. The type of the default value matches the data type of the column.

The default\_value must be a literal constant, e.g., a number or a string. It cannot be a function or an expression. However, MySQL allows you to set the current date and time (CURRENT\_TIMESTAMP) to the [TIMESTAMP](https://www.mysqltutorial.org/mysql-basics/understanding-mysql-timestamp/) and [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) columns.

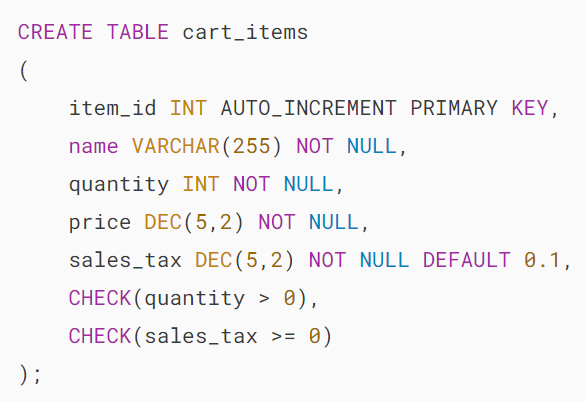
When you define a column without the [NOT NULL](https://www.mysqltutorial.org/mysql-basics/mysql-not-null-constraint/) constraint, the column will implicitly take NULL as the default value.

If a column has a DEFAULT constraint and the [INSERT](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) or [UPDATE](https://www.mysqltutorial.org/mysql-basics/mysql-update/) statement doesn’t provide the value for that column, MySQL will use the default value specified in the DEFAULT constraint.

Typically, you set the DEFAULT constraints for columns when you [create the table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/). MySQL also allows you to add default constraints to the columns of existing tables. If you don’t want to use default values for columns, you can remove the default constraints.

## **MySQL DEFAULT constraint example**

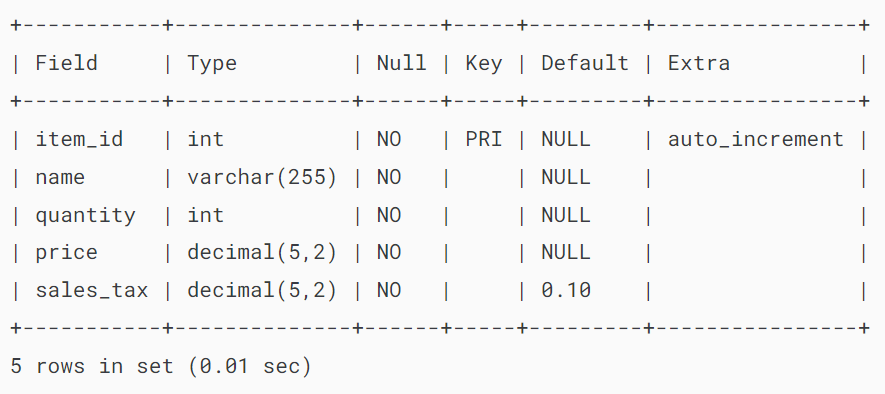
The following example creates a new table named cart\_items with four columns item\_id, name, quantity, and sales\_tax:



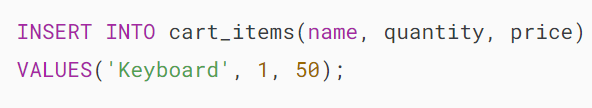
The sales\_tax column has a default value 0.1 (10%). The following statement shows the cart\_items table:



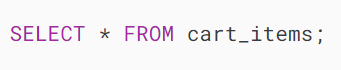
Output:



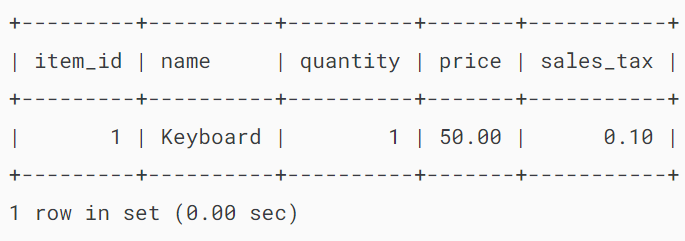
The following INSERT statement adds a new item to the cart\_items table:



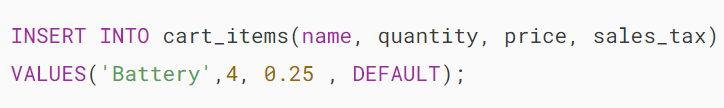
In this example, the INSERT statement doesn’t provide a value for the sales\_tax column. The sales\_tax column useS the default value specified in the DEFAULT constraint:



Output:



Also, you can explicitly use the DEFAULT keyword when you insert a new row into the cart\_items table:



In this case, the sales\_tax column takes the default value:

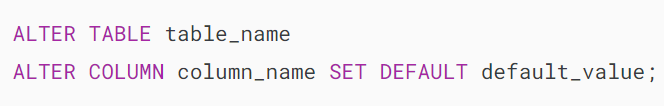


Output:

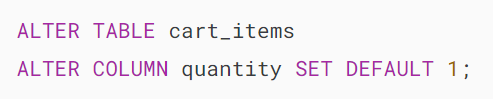
## 

## **Adding a DEFAULT constraint to a column**

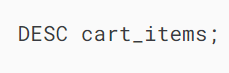
To add a default constraint to a column of an existing table, you use the [ALTER TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/) statement:



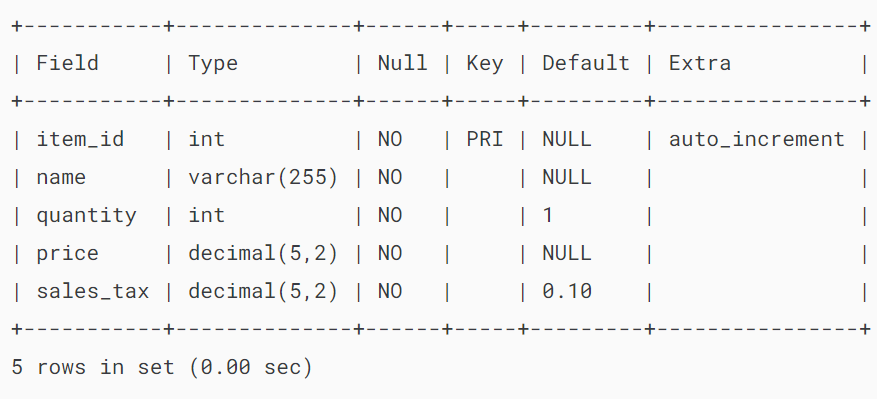
The following example adds a DEFAULT constraint to the quantity column of the cart\_itesm table:



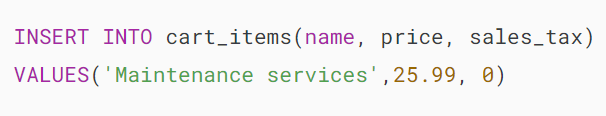
If you describe the cart\_items table, you’ll see the changes:



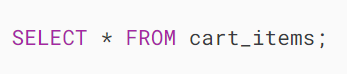
Output:



The following statement inserts a new row into the cart\_items table without specifying a value for the quantity column:



The value of the quantity column will default to 1:

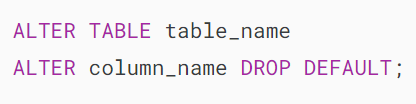


Output:

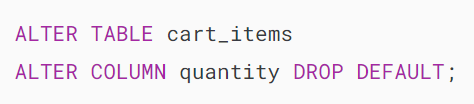
## 

## **Removing a DEFAULT constraint from a column**

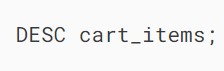
To remove a DEFAULT constraint from a column, you use the ALTER TABLE statement:



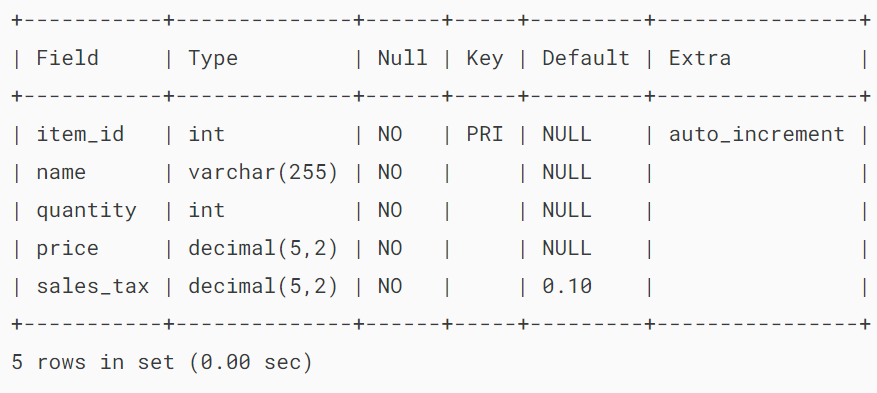
The following example removes the DEFAULT constraint from the quantity column of the cart\_items table:



And here’s the new cart\_items structure:



Output:



## **Summary**

* MySQL DEFAULT constraints set default values for columns.
* Use DEFAULT default\_value to set a default constraint to a column.
* Use ALTER TABLE ... ALTER COLUMN ... SET DEFAULT to add a DEFAULT constraint to a column of an existing table.
* Use ALTER TABLE ... ALTER COLUMN ... DROP DEFAULT to drop a DEFAULT constraint from a column of an existing table.

# **MySQL CHECK Constraint**

**Summary**: in this tutorial, you will learn how to use MySQL CHECK constraint to ensure that values stored in a column or group of columns satisfy a Boolean expression.

MySQL 8.0.16 implemented the SQL check constraint. If you use MySQL with the earlier versions, you can [emulate a CHECK constraint](https://www.mysqltutorial.org/mysql-basics/mysql-check-constraint-emulation/) using a [view](https://www.mysqltutorial.org/mysql-views/) WITH CHECK OPTION or a [trigger](https://www.mysqltutorial.org/mysql-triggers/).

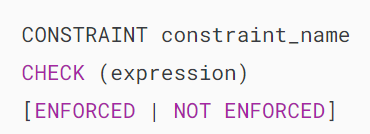
## **Introduction to the MySQL CHECK constraint**

Before MySQL 8.0.16, the [CREATE TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) allows you to include a table CHECK constraint. However, MySQL ignores all the CHECK constraints:



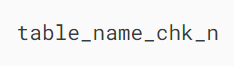
As of MySQL 8.0.16, the [CREATE TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) supported essential features of table and column CHECK constraints for all [storage engines](https://www.mysqltutorial.org/mysql-administration/mysql-storage-engines/).

Here is the basic syntax:



In this syntax:

First, specify the name for the check constraint that you want to create after the CONSTRAINT keyword. If you omit the constraint name, MySQL automatically generates a name with the following convention:



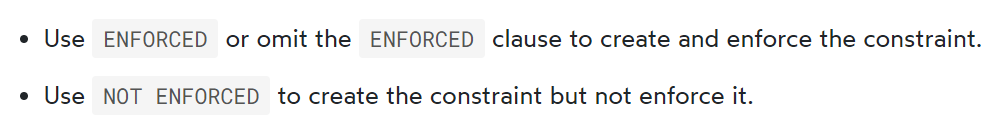
In this convention, n is an ordinal number such as 1,2, and 3. For example, the automatically generated names of CHECK constraints of the parts table will be parts\_chk\_1, parts\_chk\_2, and so on.

Second, specify a Boolean expression which must be evaluated to TRUE or UNKNOWN for each row of the table inside the parentheses after the CHECK keyword.

If the expression evaluates to FALSE, the values violate the constraint or a constraint violation occurs.

Note that MySQL treats 1 as true and 0 as false.

Third, optionally specify the enforcement clause to indicate whether the check constraint is enforced:



As mentioned earlier, you can define a CHECK constraint as a table constraint or column constraint.

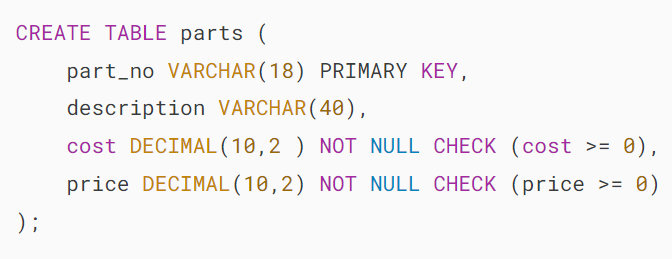
A table CHECK constraint can reference multiple columns whereas the column CHECK constraint can refer to the only column where it is defined.

## **MySQL CHECK constraint examples**

Let’s take some examples of using the CHECK constraints.

### **1) Creating CHECK constraints as column constraints**

The following [CREATE TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) statement creates a new table called parts:



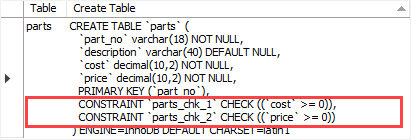
The parts table has two column CHECK constraints: one for the cost column and the other for the price column.

Because we did not explicitly specify the names of the CHECK constraints, MySQL automatically generated names for them.

To view the table definition with the CHECK constraint name, you use the SHOW CREATE TABLE statement:



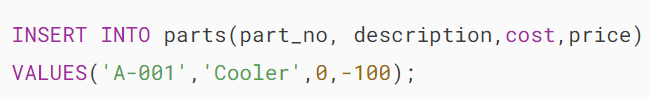
Output:



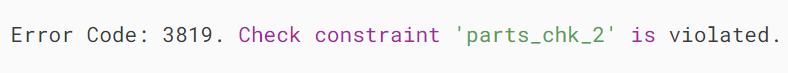
The output indicates that MySQL generated the names (parts\_chk\_1 and parts\_chk\_2) for the check constraints.

After creating CHECK constraints, if you [insert](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) or [update](https://www.mysqltutorial.org/mysql-basics/mysql-update/) a value that causes the Boolean expression to be false, MySQL rejects the change and issues an error.

This statement inserts a new row into the parts table:



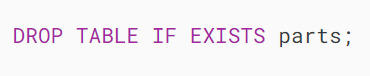
MySQL issued an error:



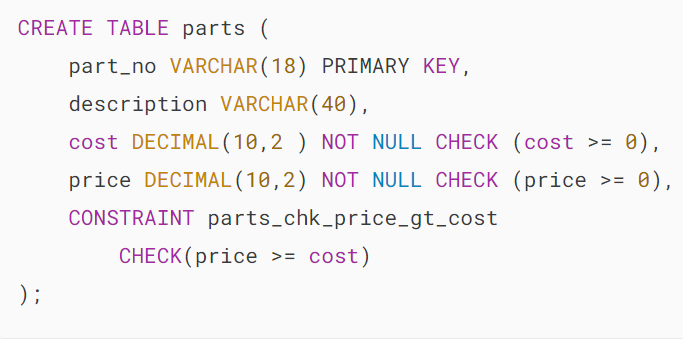
Because the value of the price column is negative which causes the expression price > 0 evaluates to FALSE that results in a constraint violation.

### **2) Creating CHECK constraints as a table constraints**

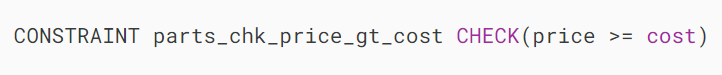
First, drop the parts table:



Then, create a new parts table with one more table CHECK constraint:



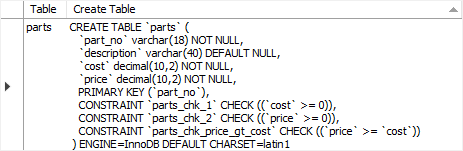
The following new clause defines a table CHECK constraint that ensures the price is always greater than or equal to the cost:



Because we explicitly specify the name of the CHECK constraint, MySQL creates the new constraint with the specified name.

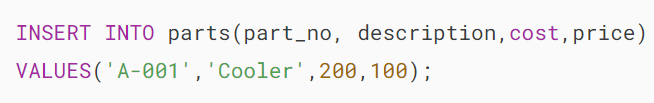
Here is the definition of the parts table:





The table CHECK constraint appears at the end of the table definition after the column list.

This statement attempts to [insert](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) a new part whose price is less than the cost:

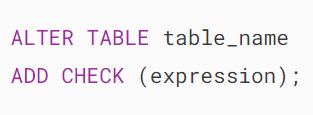


Here is the error due to the constraint violation:

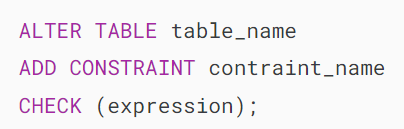
## 

## **Adding a check constraint to a table**

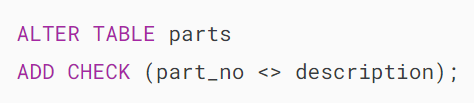
To add a check constraint to an existing table, you use the ALTER TABLE ... ADD CHECK statement:



If you want to explicitly specify the name of the CHECK constraint, you can use the ALTER TABLE ... ADD CONSTRAINT ... CHECK statement:

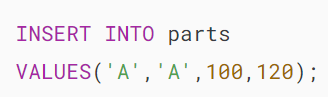


For example, the following statement adds a CHECK constraint to the parts table:

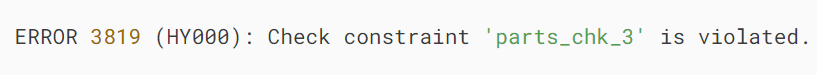


This CHECK constraint prevents you from having the part\_no identical to the description.

For example, the following INSERT statement will be rejected:

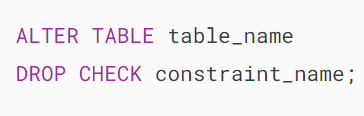


Output:



## **Removing a check constraint from a table**

To remove a CHECK constraint from a table, you use the ALTER TABLE ... DROP CHECK statement:



For example, the following statement removes the CHECK constraint parts\_chk\_3 from the parts table:

## 

## **Summary**

* Use CHECK constraints to ensure values stored in a column satisfy a Boolean condition.
* Use the CHECK(expression) to define a CHECK constraint.
* Use the ALTER TABLE ... ADD CHECK to add a CHECK constraint to a table.
* Use the ALTER TABLE ... DROP CHECK to remove a CHECK constraint from a table.