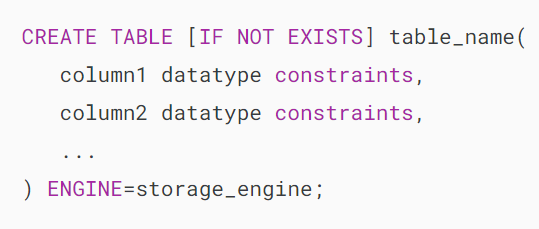
# **MySQL CREATE TABLE**

## **Introduction to MySQL CREATE TABLE statement**

The CREATE TABLE statement allows you to create a new table in a database.

The following illustrates the basic syntax of the CREATE TABLE  statement:



In this syntax:

* table\_name: This is the name of the table that you want to create.
* column1, column2, etc.: The names of the columns in the table.
* datatype: the data of each column such as INT, VARCHAR, DATE, etc.
* constraints: These are optional constraints such as NOT NULL, UNIQUE, PRIMARY KEY, and FOREIGN KEY.

If you create a table with a name that already exists in the database, you’ll get an error. To avoid the error, you can use the IF NOT EXISTS option.

In MySQL, each table has a [storage engine](https://www.mysqltutorial.org/mysql-administration/mysql-storage-engines/) such as InnoDB or MyISAM. The ENGINE clause allows you to specify the storage engine of the table.

If you don’t explicitly specify a storage engine, MySQL will use the default storage engine which is InnoDB.

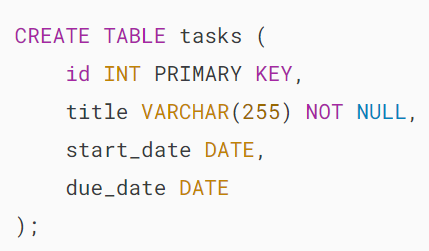
InnoDB became the default storage engine starting with MySQL version 5.5. The InnoDB storage engine offers several advantages of a relational database management system, including ACID transaction support, referential integrity, and crash recovery. In earlier versions, MySQL used MyISAM as the default storage engine.

## **MySQL CREATE TABLE statement examples**

Let’s take some examples of creating new tables.

### **1) Basic CREATE TABLE statement example**

The following example uses the CREATE TABLE statement to create a new table called tasks:

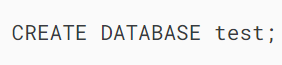


The tasks table has four columns:

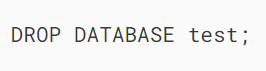
* The id is an [INT](https://www.mysqltutorial.org/mysql-basics/mysql-int/) column and serves as the primary key column.
* The title is a [VARCHAR](https://www.mysqltutorial.org/mysql-basics/mysql-varchar/) column and cannot be NULL.
* The start\_date and end\_date are the [DATE](https://www.mysqltutorial.org/mysql-basics/mysql-date/) column and can be NULL.

To execute the CREATE TABLE statement:

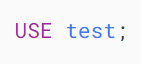
[create a new database](https://www.mysqltutorial.org/mysql-basics/mysql-create-database/) called test:



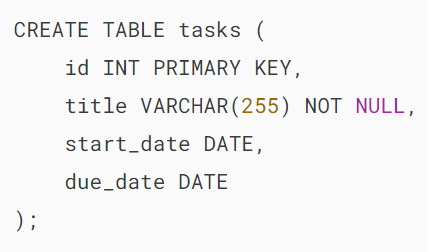
If the database already exists, you can drop it first before executing the above statement:



Then, select the test database to work with:



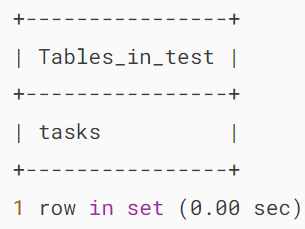
After that, execute the CREATE TABLE statement:



Finally, [list all tables](https://www.mysqltutorial.org/mysql-administration/mysql-show-tables/) in the test database:



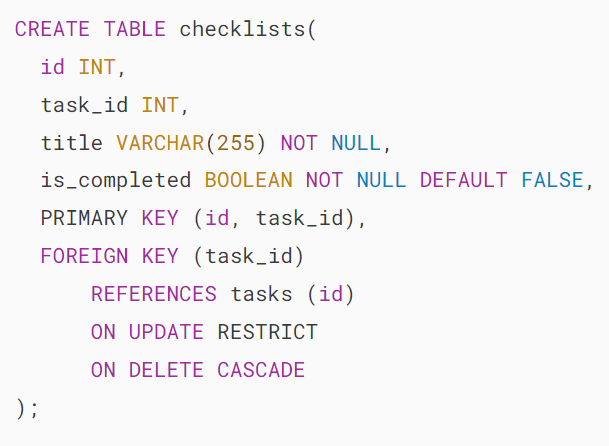
Output:



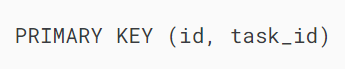
It shows the table tasks that we have created.

### **2) Creating a table with a foreign key example**

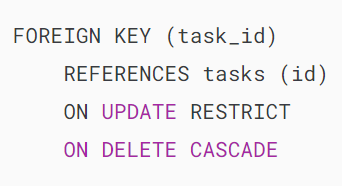
Suppose each task has a checklist. To store the checklists of tasks, you can create a new table called checklists as follows:



The table checklists has a primary key that consists of two columns. Therefore, we need to use a table constraint to define the [primary key](https://www.mysqltutorial.org/mysql-basics/mysql-primary-key/):



In addition, the task\_id is the [foreign key](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/) column that references the id column of the tasks table, therefore, we use a foreign key constraint to establish this relationship:



Note that you will learn more about the [foreign key constraint here](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/).

## **Summary**

* Use CREATE TABLE statement to create a new table.

# **MySQL AUTO\_INCREMENT**

**Summary**: in this tutorial, you will learn how to use the MySQL AUTO\_INCREMENT attribute to automatically generate unique integer values for a column.

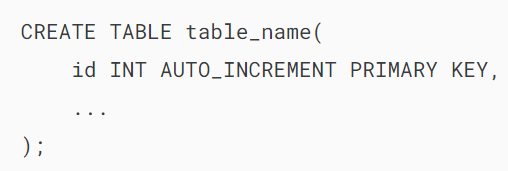
## **Introduction to MySQL AUTO\_INCREMENT attribute**

In MySQL, you use the AUTO\_INCREMENT attribute to automatically generate unique integer values for a column whenever you insert a new row into the table.

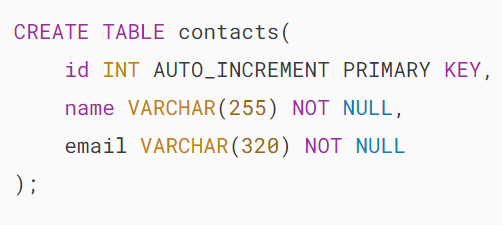
Typically, you use the AUTO\_INCREMENT attribute for the [primary key](https://www.mysqltutorial.org/mysql-basics/mysql-primary-key/) column to ensure each row has a unique identifier.

### **Creating a table with MySQL AUTO\_INCREMENT column**

To [create a table](https://www.mysqltutorial.org/perl-mysql/perl-mysql-create-table/) with an auto-increment column, you use the AUTO\_INCREMENT attribute:



For example, the following statement creates a table called contacts to store contact data:

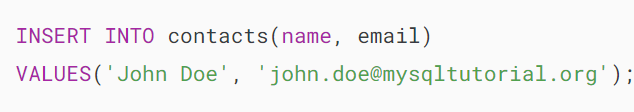


In this example, we assign the AUTO\_INCREMENT attribute to the id column to set it as an auto-increment primary key.

This means that when you insert a new row into the contacts table without providing a value for the id column, MySQL will automatically generate a unique number.

### **Inserting rows with AUTO\_INCREMENT column**

When inserting rows into the table with an AUTO\_INCREMENT column, you don’t need to specify a value for that column. MySQL will automatically generate the value for you. For example:



In the INSERT statement, we don’t specify a value for the id column and only provide the values for the name and email columns. MySQL automatically generated the value 1 for the id column:

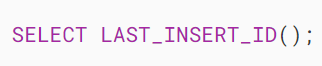


Output:

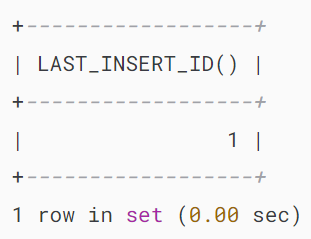
### 

### **Retrieving the last auto-increment value**

To get the AUTO\_INCREMENT value that MySQL generated for the most recent insert, you use the LAST\_INSERT\_ID() function:



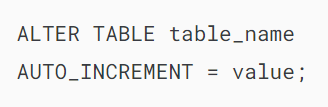
Output:



The query returns the last auto-increment value generated for the ID column, which you can use for other purposes such as inserting into a related table.

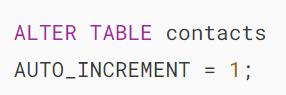
### **Resetting the current auto-increment value**

To reset the AUTO\_INCREMENT value, you use the ALTER TABLE statement:



Note that the ALTER TABLE statement takes effect only if the value that you want to reset to is higher than or equal to the maximum value in the AUTO\_INCREMENT column of the table\_name.

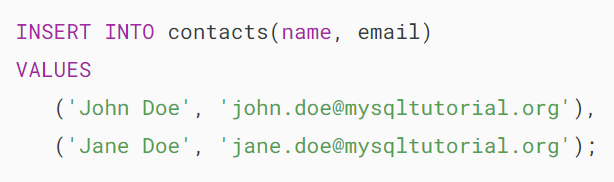
For example, the following statement reset the current auto-increment value to 1:



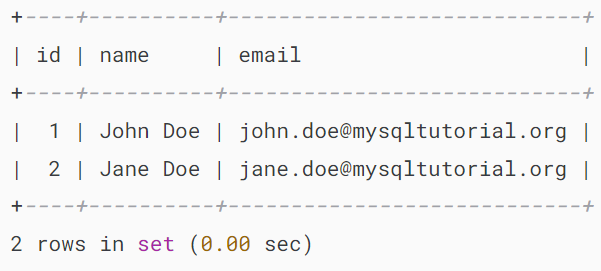
Alternatively, you can delete all rows from the table and reset the AUTO\_INCREMENT value simultaneously. To do that, you use the [TRUNCATE TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-truncate-table/) statement:



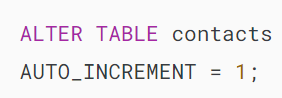
The following example illustrates how to reset the value in the AUTO\_INCREMENT column to an invalid value:



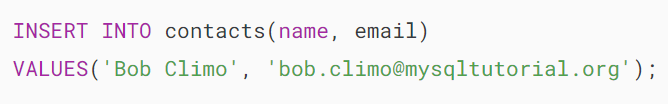
The contacts table now has two rows:



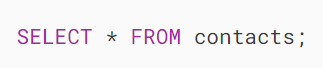
If you reset the AUTO\_INCREMENT column to any number that is less than or equal to 2 using the ALTER TABLE statement, the operation will have no effects. For example:



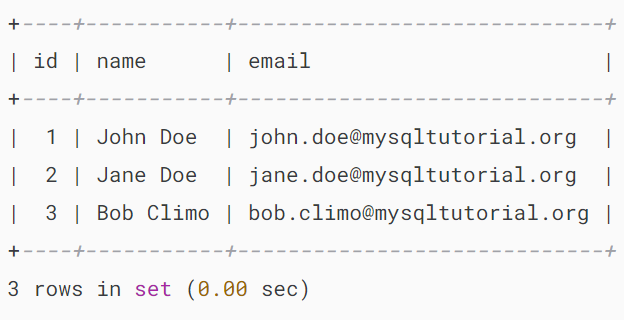
Now, if you insert a new row into the contacts table, MySQL will use the next number 3 for the new row. For example:



The following query returns all rows of the contacts table:



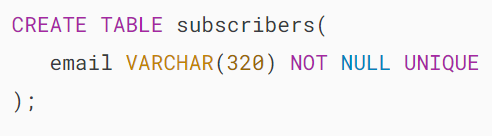
Output:



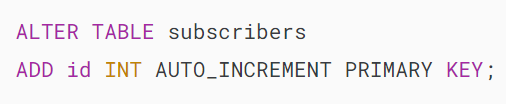
### **Adding an AUTO\_INCREMENT column to an existing table**

To add an AUTO\_INCREMENT to an existing table, you use the ALTER TABLE statement. For example:

First, create a new table without an AUTO\_INCREMENT column:



Second, add the column id to the subscribers table as an AUTO\_INCREMENT column:



## **Summary**

* Assign the AUTO\_INCREMENT attribute to a column to instruct MySQL to automatically generate unique integer values for the column.

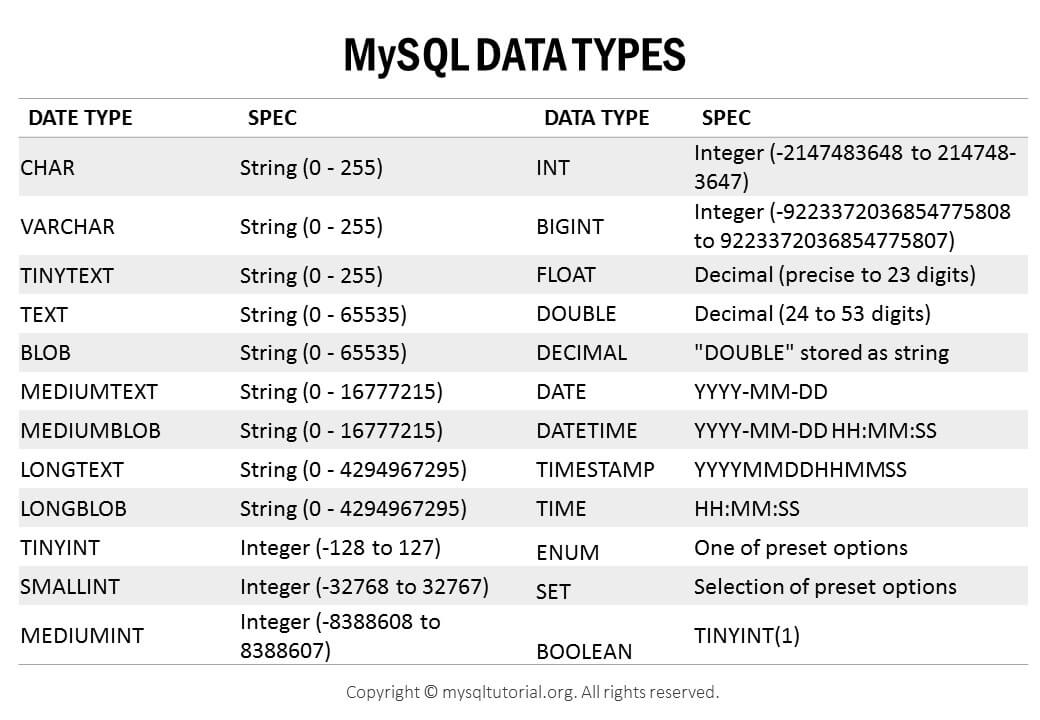
# **MySQL Data Types**

**Summary**: in this tutorial, you will learn about **MySQL data types**and how to use them effectively in designing databases in MySQL.

## **Introduction to MySQL Data Types**

A table contains multiple columns with specific data types such as numeric or string. MySQL provides more data types other than just numeric and string. Each data type in MySQL can be determined by the following characteristics:

* The kind of values it represents.
* The space that takes up and whether the values are a fixed-length or variable length.
* The values of the data type can be indexed or not.
* How MySQL compares the values of a specific data type.



[Download MySQL Data Types Overview](https://www.mysqltutorial.org/wp-content/uploads/2018/03/MySQL-Data-Types.pdf)

## **MySQL numeric data types**

In MySQL, you can find all SQL standard numeric types including exact number data types, and approximate numeric data types including integer, fixed-point, and floating-point.

In addition, MySQL also has [BIT](https://www.mysqltutorial.org/mysql-basics/mysql-bit/) data type for storing bit values. Numeric types can be signed or unsigned except for the BIT type.

The following table shows the summary of numeric types in MySQL:

| **Numeric Types** | **Description** |
| --- | --- |
| [TINYINT](https://www.mysqltutorial.org/mysql-basics/mysql-int/) | A very small integer |
| [SMALLINT](https://www.mysqltutorial.org/mysql-basics/mysql-int/) | A small integer |
| [MEDIUMINT](https://www.mysqltutorial.org/mysql-basics/mysql-int/) | A medium-sized integer |
| [INT](https://www.mysqltutorial.org/mysql-basics/mysql-int/) | A standard integer |
| [BIGINT](https://www.mysqltutorial.org/mysql-basics/mysql-int/) | A large integer |
| [DECIMAL](https://www.mysqltutorial.org/mysql-basics/mysql-decimal/) | A fixed-point number |
| FLOAT | A single-precision floating point number |
| DOUBLE | A double-precision floating point number |
| [BIT](https://www.mysqltutorial.org/mysql-basics/mysql-bit/) | A bit field |

## **MySQL Boolean data type**

MySQL does not have the built-in [BOOLEAN](https://www.mysqltutorial.org/mysql-basics/mysql-boolean/) or [BOOL](https://www.mysqltutorial.org/mysql-basics/mysql-boolean/) data type. To represent boolean values, MySQL uses the smallest integer type which is TINYINT(1). In other words, BOOLEAN and BOOL are synonyms for TINYINT(1).

## **MySQL String data types**

In MySQL, a string can hold anything from plain text to binary data such as images or files. Strings can be compared and searched based on pattern matching by using the [LIKE](https://www.mysqltutorial.org/mysql-basics/mysql-like/) operator, [regular expression](https://www.mysqltutorial.org/mysql-regular-expressions/mysql-regexp/), and [full-text search](https://www.mysqltutorial.org/mysql-full-text-search/).

The following table shows the string data types in MySQL:

| **String Types** | **Description** |
| --- | --- |
| [CHAR](https://www.mysqltutorial.org/mysql-basics/mysql-char-data-type/) | A fixed-length nonbinary (character) string |
| [VARCHAR](https://www.mysqltutorial.org/mysql-basics/mysql-varchar/) | A variable-length non-binary string |
| [BINARY](https://www.mysqltutorial.org/mysql-basics/mysql-binary/) | A fixed-length binary string |
| [VARBINARY](https://www.mysqltutorial.org/mysql-basics/mysql-varbinary/) | A variable-length binary string |
| [TINYBLOB](https://www.mysqltutorial.org/mysql-basics/mysql-blob/) | A very small BLOB (binary large object) |
| [BLOB](https://www.mysqltutorial.org/mysql-basics/mysql-blob/) | A small BLOB |
| [MEDIUMBLOB](https://www.mysqltutorial.org/mysql-basics/mysql-blob/) | A medium-sized BLOB |
| [LONGBLOB](https://www.mysqltutorial.org/mysql-basics/mysql-blob/) | A large BLOB |
| [TINYTEXT](https://www.mysqltutorial.org/mysql-basics/mysql-text/) | A very small non-binary string |
| [TEXT](https://www.mysqltutorial.org/mysql-basics/mysql-text/) | A small non-binary string |
| [MEDIUMTEXT](https://www.mysqltutorial.org/mysql-basics/mysql-text/) | A medium-sized non-binary string |
| [LONGTEXT](https://www.mysqltutorial.org/mysql-basics/mysql-text/) | A large non-binary string |
| [ENUM](https://www.mysqltutorial.org/mysql-basics/mysql-enum/) | An enumeration; each column value may be assigned one enumeration member |
| SET | A set; each column value may be assigned zero or more SET members |

## **MySQL date and time data types**

MySQL provides types for date and time as well as the combination of date and time. In addition, MySQL supports the [timestamp](https://www.mysqltutorial.org/mysql-basics/understanding-mysql-timestamp/) data type for tracking the changes in a row of a table. If you just want to store years without dates and months, you can use the YEAR data type.

The following table illustrates the MySQL date and time data types:

| **Date and Time Types** | **Description** |
| --- | --- |
| [DATE](https://www.mysqltutorial.org/mysql-basics/mysql-date/) | A date value in CCYY-MM-DD format |
| [TIME](https://www.mysqltutorial.org/mysql-basics/mysql-time/) | A time value in hh:mm:ss format |
| [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) | A date and time value in CCYY-MM-DD hh:mm:ssformat |
| [TIMESTAMP](https://www.mysqltutorial.org/mysql-basics/understanding-mysql-timestamp/) | A timestamp value in CCYY-MM-DD hh:mm:ss format |
| YEAR | A year value in CCYY or YY format |

## **MySQL spatial data types**

MySQL supports many spatial data types that contain various kinds of geometrical and geographical values as shown in the following table:

| **Spatial Data Types** | **Description** |
| --- | --- |
| GEOMETRY | A spatial value of any type |
| POINT | A point (a pair of X-Y coordinates) |
| LINESTRING | A curve (one or more POINT values) |
| POLYGON | A polygon |
| GEOMETRYCOLLECTION | A collection of GEOMETRY values |
| MULTILINESTRING | A collection of LINESTRING values |
| MULTIPOINT | A collection of POINT values |
| MULTIPOLYGON | A collection of POLYGON values |

## **JSON data type**

MySQL supported a native [JSON](https://www.mysqltutorial.org/mysql-json/) datatype since version 5.7.8 which allows you to store and manage JSON documents more effectively.

The native JSON data type provides automatic validation of JSON documents and optimal storage format.

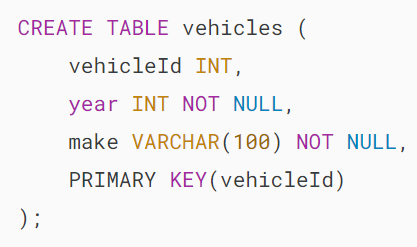
In this tutorial, you have learned various MySQL data types that help you determine which data type you should use for columns when you [create tables](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/).

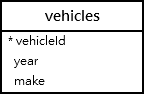
# **MySQL ALTER TABLE**

**Summary**: in this tutorial, you will learn how to use the MySQL ALTER TABLE statement to add a column, alter a column, rename a column, drop a column, and rename a table.

## **Setting up a sample table**

Let’s [create a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named vehicles for the demonstration:



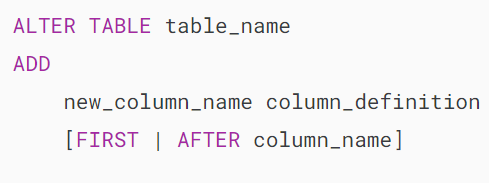


## **MySQL ALTER TABLE – Add columns to a table**

The ALTER TABLE ADD statement allows you to add one or more columns to a table.

### **1) Add a column to a table**

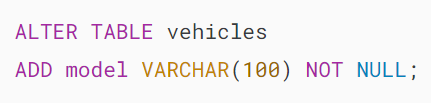
To [add a column](https://www.mysqltutorial.org/mysql-basics/mysql-add-column/) to a table, you use the ALTER TABLE ADD syntax:



In this syntax:

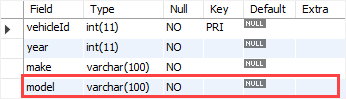
* table\_name – specify the name of the table to which you want to add a new column or columns after the ALTER TABLE keywords.
* new\_column\_name –  specify the name of the new column.
* column\_definition– specify the datatype, maximum size, and column constraint of the new column
* FIRST | AFTER column\_name specify the position of the new column in the table. You can add a column after an existing column (ATER column\_name) or as the first column (FIRST). If you omit this clause, the column is appended at the end of the column list of the table.

The following example uses the ALTER TABLE ADD statement to add a column at the end of the vehicles table:



This statement shows the column list of the vehicles table:

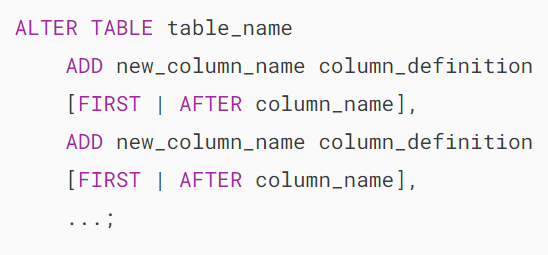




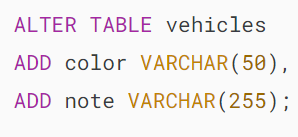
As shown clearly from the output, the column model has been added to the vehicles table.

### **2) Add multiple columns to a table**

To [add multiple columns](https://www.mysqltutorial.org/mysql-basics/mysql-add-column/) to a table, you use the following form of the ALTER TALE ADD statement:



For example, this statement adds two columns color and note to the vehicles table:



This statement shows the new structure of the vehicles table:

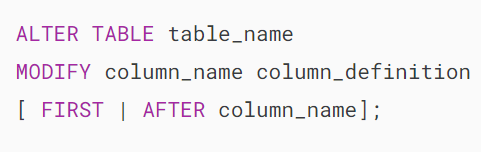
## 

## **MySQL ALTER TABLE - add multiple columns example**

## **MySQL ALTER TABLE – Modify columns**

### **1) Modify a column**

Here is the basic syntax for modifying a column in a table:

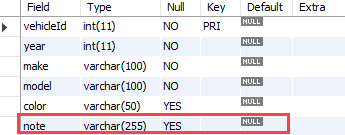


It’s a good practice to view the attributes of a column before modifying it.

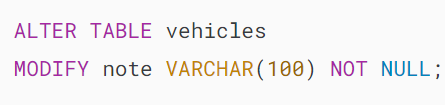
Suppose that you want to change the note column a [NOT NULL](https://www.mysqltutorial.org/mysql-basics/mysql-not-null-constraint/) column with a maximum of 100 characters.

First, show the column list of the vehicles table:



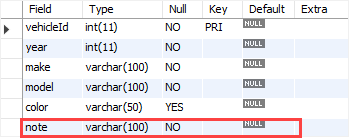


Then, modify the note column:



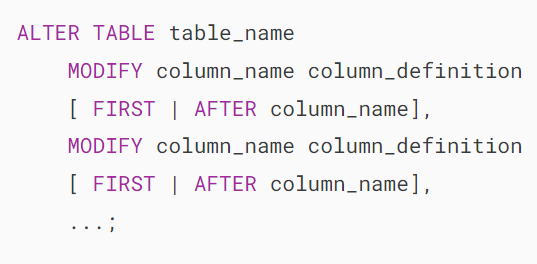
Finally, show the column list of the vehicles table to verify the change:



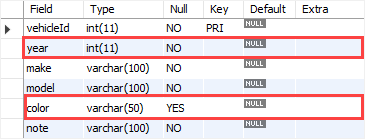


### **2) Modify multiple columns**

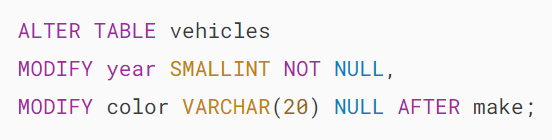
The following statement allows you to modify multiple columns:



First, show the current columns of the vehicles table:



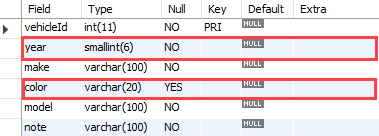
Second, use the ALTER TABLE MODIFY statement to modify multiple columns:



In this example:

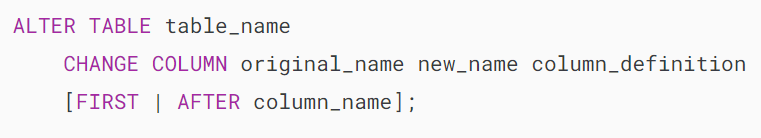
* First, modify the data type of the year column from INT to SMALLINT
* Second, modify the color column by setting the maximum length to 20, removing the NOT NULL constraint, and changing its position to appear after the make column.

Third, show the new column list of the vehicles table to verify the modifications:



## **MySQL ALTER TABLE – Rename a column in a table**

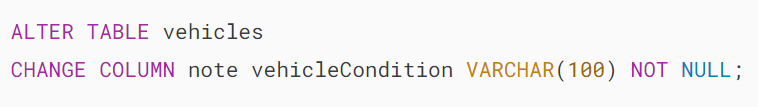
To rename a column, you use the following statement:



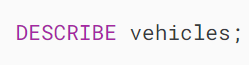
In this syntax:

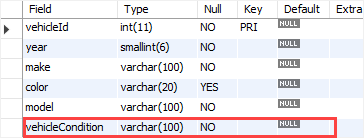
* First, specify the name of the table to which the column belongs.
* Second, specify the column name and the new name followed by the column definition after the CHANGE COLUMN keywords.
* Third, use the FIRST or AFTER column\_name option to determine the new position of the column.

The following example uses the ALTER TABLE CHANGE COLUMN statement to rename the column note to vehicleCondition:



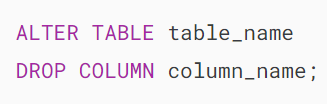
Let’s review the column list of the vehicles table:





## **MySQL ALTER TABLE – Drop a column**

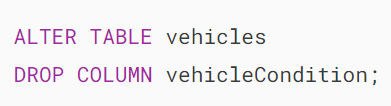
To [drop a column](https://www.mysqltutorial.org/mysql-drop-column/) in a table, you use the ALTER TABLE DROP COLUMN statement:



In this syntax:

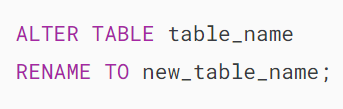
* First, specify the name of the table that you want to drop a column after the ALTER TABLE keywords.
* Second, specify the name of the column that you want to drop after the DROP COLUMN keywords.

This example shows how to remove the vehicleCondition column from the vehicles table:



## **MySQL ALTER TABLE – Rename table**

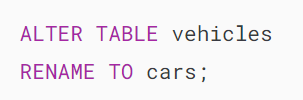
To [rename a table](https://www.mysqltutorial.org/mysql-basics/mysql-rename-table/), you use the ALTER TABLE RENAME TO statement:



In this syntax:

* First, specify the name of the table that you want to rename after the ALTER TABLE keywords.
* Second, specify the new name for the table after the RENAME TO keywords.

This example renames the vehicles table to cars:



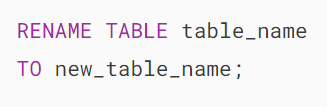
# **MySQL RENAME TABLE Statement**

**Summary**: in this tutorial, you will learn how to rename tables using MySQL RENAME TABLE statement and ALTER TABLE statement.

## **Introduction to MySQL RENAME TABLE statement**

Due to evolving business requirements, you need to rename the existing table to better align with the new situation. MySQL offers a valuable statement for renaming one or more tables.

To rename one or more tables, you can use the RENAME TABLE statement as follows:



In this syntax:

* table\_name: This is the name of the table that you want to rename.
* new\_table\_name: This is the new table name.

The table with the table\_name must exist or the RENAME statement will fail with an error.

While executing the RENAME TABLE statement, you need to ensure that there are no active transactions or [locked tables](https://www.mysqltutorial.org/mysql-basics/mysql-table-locking/).

Note that you cannot use the RENAME TABLE statement to rename a [temporary table](https://www.mysqltutorial.org/mysql-basics/mysql-temporary-table/), but you can use the [ALTER TABLE statement](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/) to rename a temporary table.

In terms of security, any existing [privileges that you granted to the old table](https://www.mysqltutorial.org/mysql-administration/mysql-grant/) must be manually migrated to the new table.

Before renaming a table, it’s important to thoroughly evaluate the potential impact.

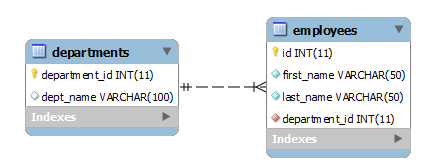
For example, you should investigate which applications are currently using the table. Changing the table name would necessitate corresponding changes in the application code that references it.

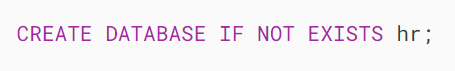
Additionally, you’ll need to manually adjust other database objects, including [views](https://www.mysqltutorial.org/mysql-views/), [stored procedures](https://www.mysqltutorial.org/mysql-stored-procedure/), [triggers](https://www.mysqltutorial.org/mysql-triggers/), and [foreign key constraints](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/) that reference the table.

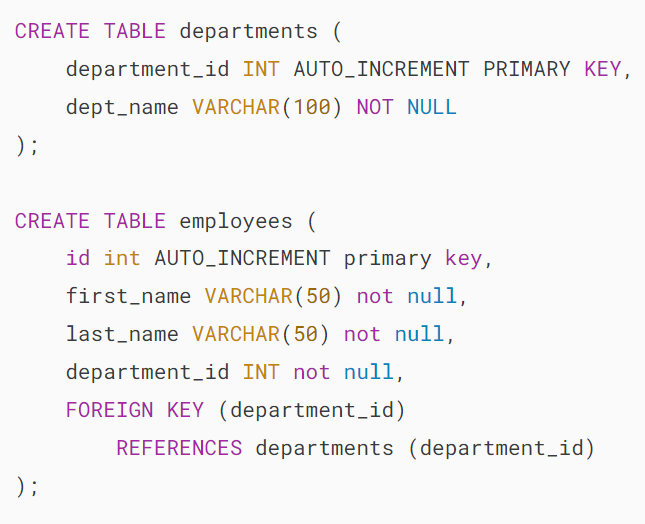
We will delve into this in more detail in the following examples.

## **MySQL RENAME TABLE examples**

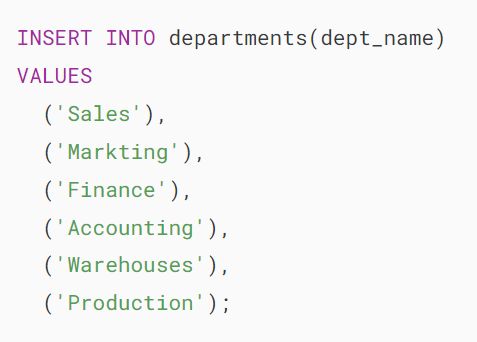
First, [create a new database](https://www.mysqltutorial.org/mysql-basics/mysql-create-database/) named hr that includes two tables: employees and departments for the demonstration.





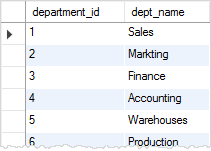
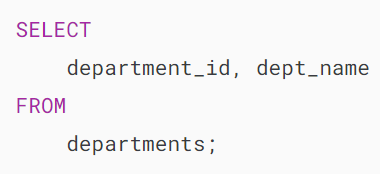


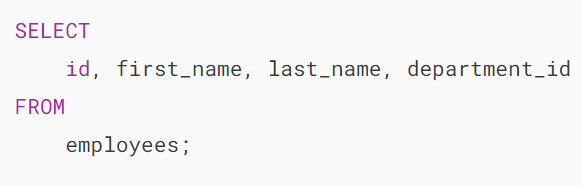
Second, [insert sample data](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into both employees and departments tables:

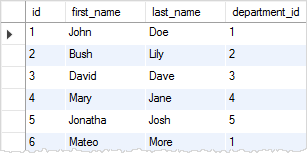




Third, query data from the departments and employees tables:



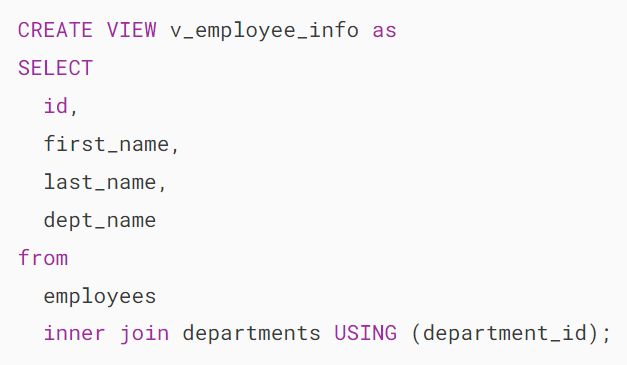




### **1) Renaming a table referenced by a view**

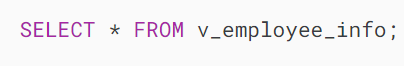
If the table you are going to rename is referenced by a [view](https://www.mysqltutorial.org/mysql-views/), the view will become invalid, and you have to adjust the view manually.

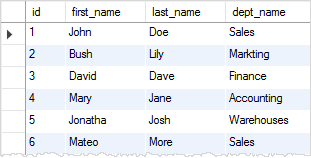
For example, we create a view named v\_employee\_info based on the employees and departments tables as follows:



The views use the [inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/) clause to join departments and employees tables.

The following [SELECT statement](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) returns all data from the v\_employee\_info view.

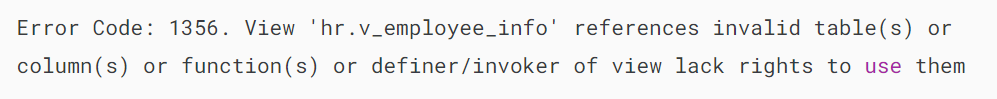




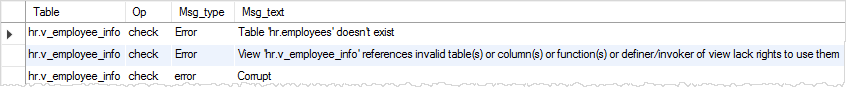
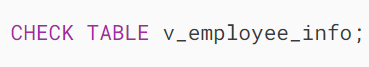
Now we rename the employees to people table and query data from the v\_employee\_info view again.



MySQL returns the following error message:



We can use the CHECK TABLE statement to check the status of the v\_employee\_info view as follows:

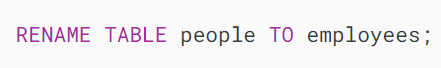


We need to manually change the v\_employee\_info view so that it refers to the people table instead of the employees table.

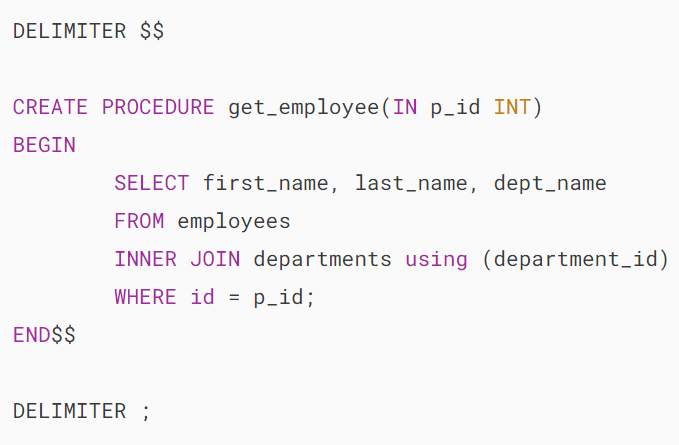
### **2) Renaming a table that is referenced by a stored procedure**

In case the table that you are going to rename is referenced by a [stored procedure](https://www.mysqltutorial.org/mysql-stored-procedure/), you have to manually adjust it like you did with the view.

First, rename the people table back to the employees table.



Then, [create a new stored procedure](https://www.mysqltutorial.org/mysql-stored-procedure/getting-started-with-mysql-stored-procedures/) named get\_employee that refers to the employees table.

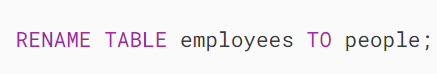


Next, execute the get\_employee table to get the data of the employee with ID 1 as follows:



MySQL RENAME TABLE with Stored Procedure

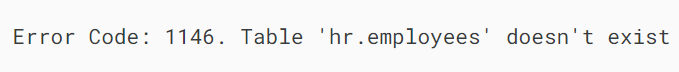
After that, rename the employees to the people table again.



Finally, call the get\_employee stored procedure to get the information of the employee with the ID 2:



MySQL returns the following error message:



To fix this, you need to manually change the employees table in the stored procedure to people table.

### **3) Renaming a table that is referenced by foreign key constraints**

The departments table links to the employees table using the department\_id column.

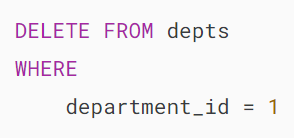
The department\_id column in the employees table is the [foreign key](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/) that references to the departments table.

If you rename the departments table, all the foreign keys that reference the departments table will not be automatically updated. In such cases, you must drop and recreate the foreign keys manually.

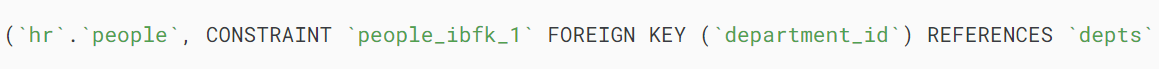


We delete a department with id 1, because of the foreign key constraint, all rows in the people table should be also deleted.

However, we renamed the departments table to the depts table without updating the foreign key manually, MySQL returns an error as illustrated below:

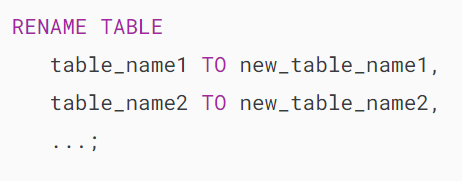


Output:

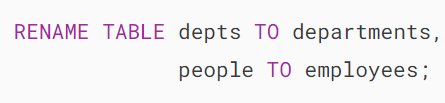


## **Renaming multiple tables**

We can also use the RENAME TABLE statement to rename multiple tables at a time. See the following statement:



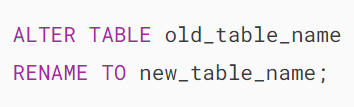
The following statement renames the people and depts tables to employees and departments tables:



Note the RENAME TABLE statement is not atomic. It means that if any errors occur, MySQL performs a rollback of all renamed tables to their old names.

## **Renaming tables using ALTER TABLE statement**

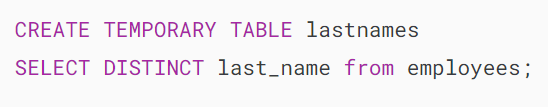
You can rename a table using the ALTER TABLE statement as follows:



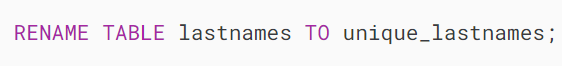
The ALTER TABLE statement can rename a temporary table while the RENAME TABLE statement cannot.

### **Renaming temporary table example**

First, create a [temporary table](https://www.mysqltutorial.org/mysql-basics/mysql-temporary-table/) that contains all unique last names that come from the last\_name column of the employees table:



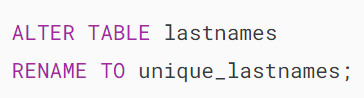
Second, use the RENAME TABLE to rename the lastnames table:



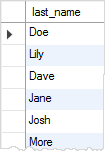
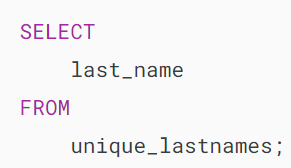
MySQL returns the following error message:



Third, use the ALTER TABLE statement to rename the lastnames table.



Fourth, query data from the unique\_lastnames temporary table:



## **Summary**

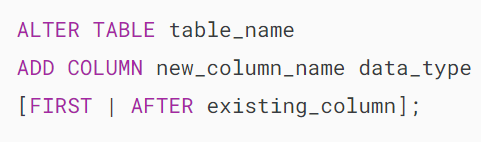
* Use the RENAME TABLE or ALTER TABLE statement to rename a table.

# **How to Add Columns to a Table Using MySQL ADD COLUMN Statement**

**Summary**: in this tutorial, you will learn how to add a column to a table using MySQL ADD COLUMN statement.

## **Introduction to MySQL ADD COLUMN statement**

To add a new column to an existing table, you use the [ALTER TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/) … ADD COLUMN statement as follows:



In this syntax:

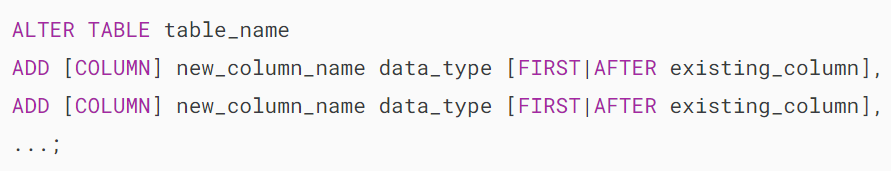
* First, provide the table name to which you want to add a new column after the ALTER TABLE clause.
* Second, define the new column and its attributes after the ADD COLUMN clause. Note that COLUMN keyword is optional so you can omit it.
* Third, specify the position of the new column in the table.

When adding a new column to a table, you can specify its position within the table. You can use the keyword FIRST if you want the new column to be positioned as the first column in the table.

Alternatively, you can use the AFTER existing\_column clause to specify that you want to add a new column after an existing column.

If you do not explicitly specify the position of the new column, the statement will automatically add it as the last column in the table.

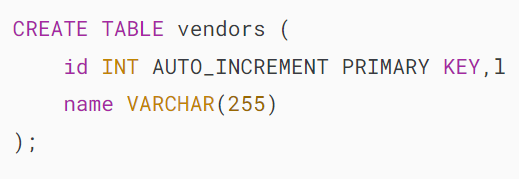
To add two or more columns to a table at the same time, you use multiple ADD COLUMN clauses like this:



## **MySQL ADD COLUMN examples**

Let’s look at examples of adding one or more columns to a table.

We’ll [create a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) called vendors with two columns id and name:



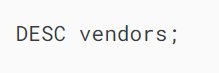
### **1) Adding one column example**

First, add a new column phone to the vendors table:

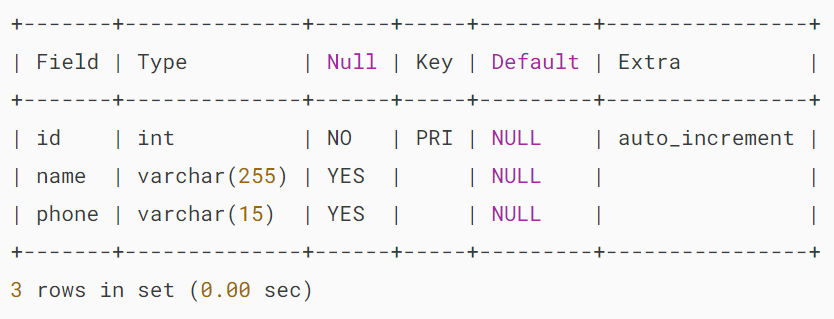


Because we specify the position of the phone column explicitly after the name column, the statement places the phone column after the name column.

Second, view the columns list of the vendor table using the DESC statement:

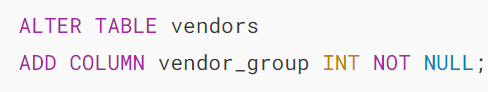


Output:



### **2) Adding a column as the last column**

First, add a new column vendor\_group to the vendors table:

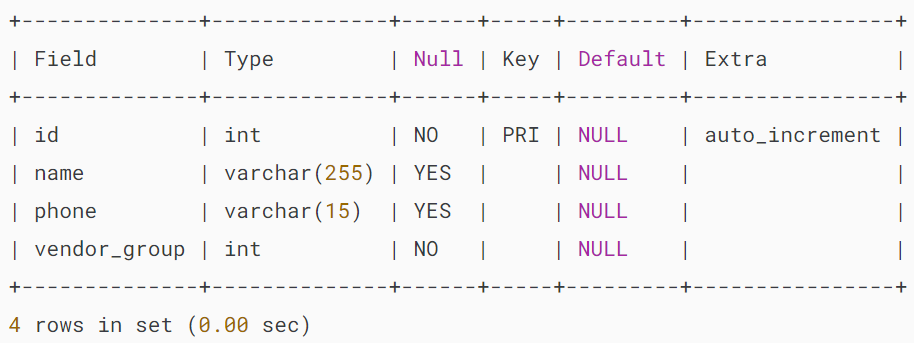


In this statement, we don’t specify the new column’s position so it adds the vendor\_group column as the last column of the vendors table.

Second, view the vendors table:

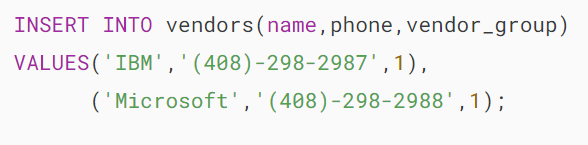


Output:

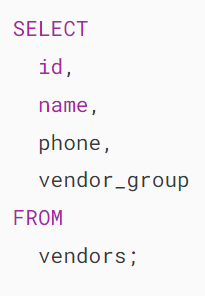


### **3) Adding two columns example**

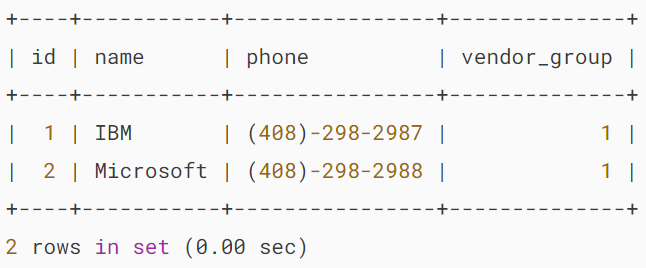
First, [insert some rows](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into the vendors table.



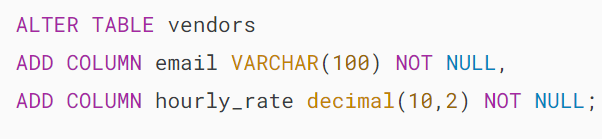
Second, [query the data](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) of the vendors table:



Output:

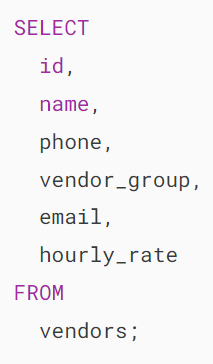


Third, add two more columns email and hourly\_rate to the vendors table using two ADD clauses:

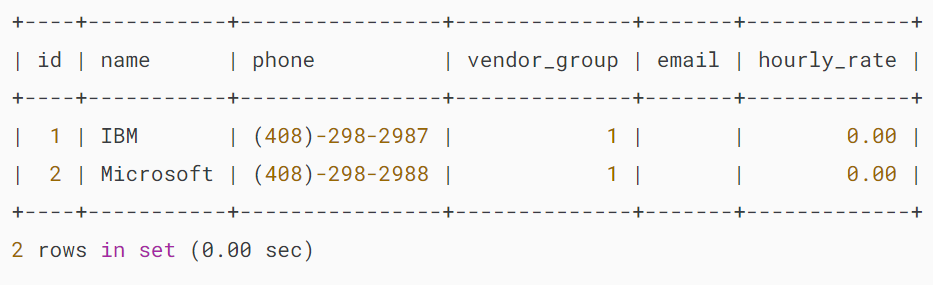


Note that both email and hourly\_rate columns are NOT NULL. However, the vendors table already has data. In this case, MySQL will use default values for these new columns.

Finally, retrieve data from the vendors table:



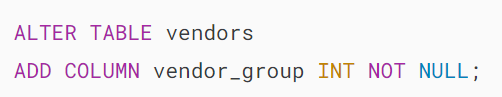
Output:



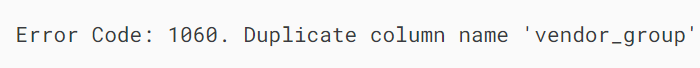
The output indicates that the email column is populated with blank, not the NULL values. And the hourly\_rate column is filled with 0.00.

### **4) Adding a column that already exists**

If you add a column that already exists in the table, MySQL will issue an error. For example, if you execute the following statement:



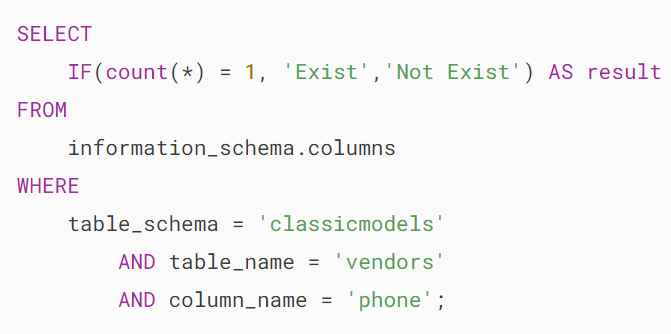
Output:



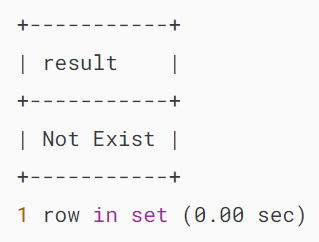
For the table with a few columns, it is easy to see which columns are already there. However, it becomes more difficult for a big table with hundreds of columns.

In this case, you want to check whether a column exists in a table before adding it.

However, there is no statement like ADD COLUMN IF NOT EXISTS available. Fortunately, you can get this information from the columns table of the information\_schema database as the following query:



Output:



In the [WHERE clause](https://www.mysqltutorial.org/mysql-basics/mysql-where/), we passed three arguments: table schema or database, table name, and column name. We used the [IF function](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-if-function/) to check whether the column exists or not.

### **5) Adding an auto-increment column**

MySQL allows a table to have up to one [auto-increment](https://www.mysqltutorial.org/mysql-basics/mysql-auto_increment/) column and that column must be defined as a key. For example:

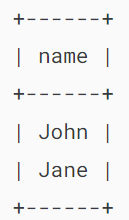
First, create a new table called contacts:



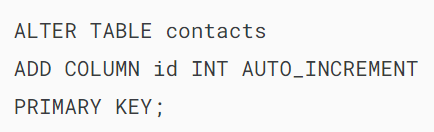
Second, insert some rows into contacts table:



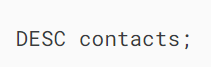
Third, retrieve the data from the contacts table:



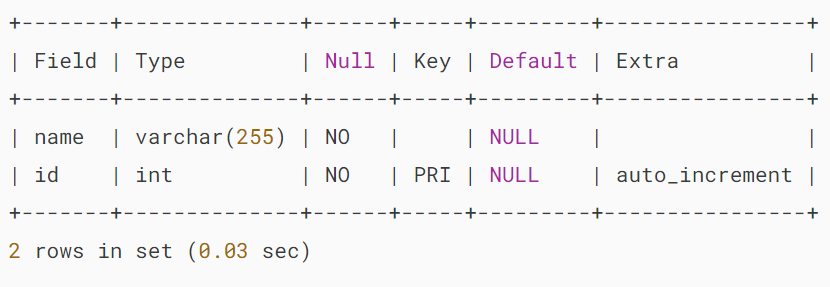
Fourth, add id column as the auto-increment primary key column to the contacts table:



Fifth, show the contacts table:



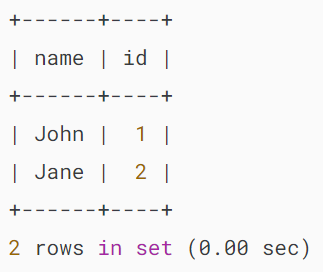
Output:



Finally, retrieve the data from the contacts table:



Output:



The output indicates that MySQL automatically generates value for the id column.

## **Summary**

* Use MySQL ADD COLUMN clause in the ALTER TABLE statement to add one or more columns to a table.