# **MySQL Character Sets**

**Summary**: in this tutorial, we’ll explore MySQL character sets and their importance in how MySQL stores text data.

In MySQL, the **character set** and [**collation**](https://www.mysqltutorial.org/mysql-basics/mysql-collation/) are essential concepts that define how data is stored and sorted in text columns.

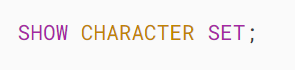
Understanding character sets and collations are crucial for designing databases that can handle different languages and text requirements.

## **Introduction to MySQL Character Sets**

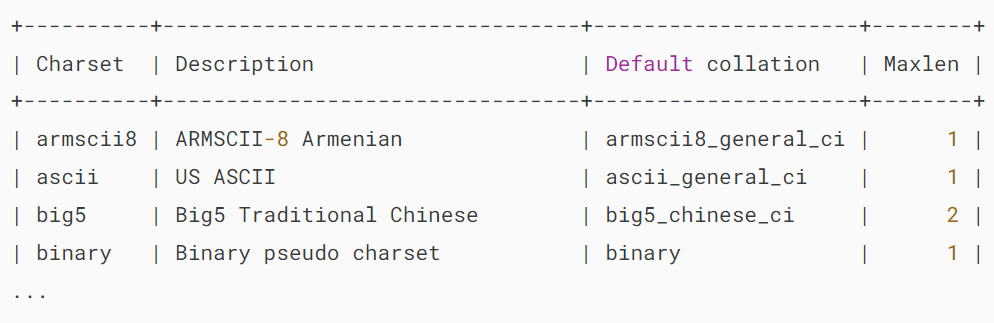
A character set is a collection of characters with a unique encoding. It defines the set of characters that can be used in a text column, such as letters, numbers, symbols, and special characters.

MySQL supports various character sets, and the choice of character set determines **the range of characters**that can be stored in a column.

Common character sets include utf8, utf8mb4, latin1, utf16, and many others. To list all character sets in the current MySQL server, you use the following statement:



Here’s the partial output:



The Maxlen column specifies the number of bytes that a character in a character set holds.

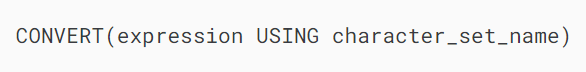
Some character sets contain single-byte characters e.g., latin1 , latin2 , cp850 , etc., whereas other character sets contain multi-byte characters.

The default character sets are utf8mb4 and utf8mb4\_0900\_ai\_ci. However, you can change them accordingly.

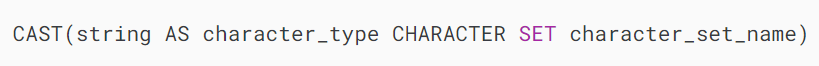
## **Converting between different character sets**

MySQL provides two functions that allow you to convert strings between different character sets: CONVERT and CAST.

The syntax of the CONVERT function is as follows:



The CAST function is similar to the CONVERT function. It converts a string to a different character set:



Take a look at the following example of using the CAST function:

## 

## **Setting character sets for client connections**

To configure a character set for a client connection, you can do one of the following ways:

### **1) Using the SET NAMES statement**

Issue the SET NAME  statement after the client connected to the MySQL database server. For example, to set a Unicode character set utf8mb4, you use the following statement:



### **2) Using –default-character-set option**

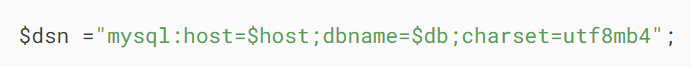
If the application supports the --default-character-set  option, you can use it to set the character set.

For example, mysql client tool supports --default-character-set option and you can set it up in the configuration file as follows:

### 

### **3) Using the charset in connection strings**

Some MySQL connectors allow you to specify a character set. For example, if you use [PHP PDO](https://www.phptutorial.net/php-pdo/), you can set the character set in the data source name as follows:



Regardless of which way you use it, make sure that the character set used by the application matches the character set stored in the MySQL database server.

## **Summary**

* Character sets define how MySQL stores data in text columns.
* Use the SHOW CHARACTER SET statement to list all character sets.
* Use the CONVERT() or CAST() function to convert between character sets.

# **MySQL Collation**

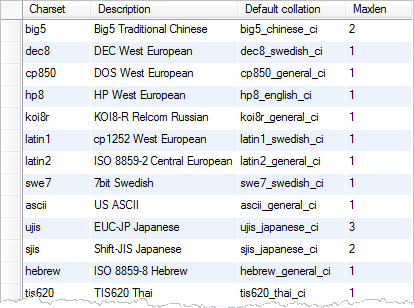
**Summary**: in this tutorial, you will learn about **MySQL collation** and how to set character sets and collations for the MySQL server, database, table, and column.

## **Introduction to MySQL collation**

A MySQL collation is a set of rules used to compare characters in a particular [character set](https://www.mysqltutorial.org/mysql-basics/mysql-character-set/). Each character set in MySQL has at least one default collation. It can have more than one collation. However, two character sets cannot have the same collation.

MySQL provides the SHOW CHARACTER SET  statement that allows you to get the default collations of character sets:

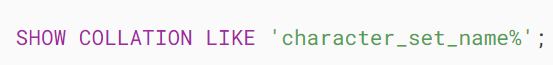




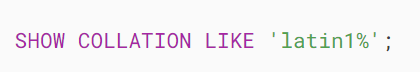
The values of the default collation column specify the default collations for the character sets.

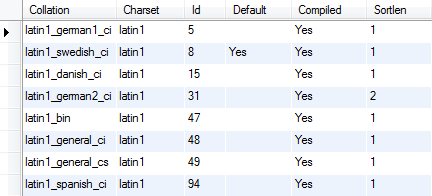
By convention, a collation for a character set begins with the character set name and ends with \_ci (case insensitive) \_cs  (case sensitive) or \_bin  (binary).

To get all collations for a given character set, you use the SHOW COLLATION  statement as follows:



For example, the following statement gets all collations for the latin1 character set:





MySQL Collations for latin1 Character Set

As mentioned above, each character set has a default collation e.g., latin1\_swedish\_ci is the default collation of the latin1 character set.

## **Setting character sets and collations**

MySQL allows you to specify character sets and collations at four levels:

1. Server
2. Database
3. Table
4. Column

### **1) Setting character sets and collations at the server Level**

MySQL uses the latin1 as the default character set. Therefore, the default collation is latin1\_swedish\_ci. You can change these settings at server startup.

If you specify one character set at server startup, MySQL will use the default collation of that character set. However, if you specify both a character set and a collation explicitly, MySQL will use the character set and collation for all databases that you will create.

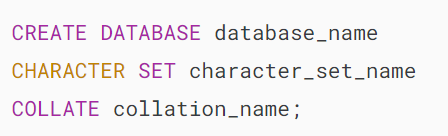
The following statement sets the utf8 character set and utf8\_unicode\_cs collation for the server via the command line:

### 

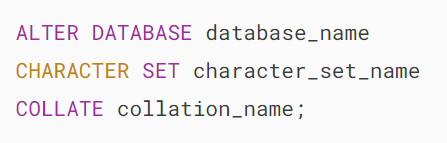
### **2) Setting character sets and collations at the database level**

When you [create a database](https://www.mysqltutorial.org/mysql-basics/mysql-create-database/) but do not specify the character set and collation, MySQL will use the default character set and collation of the server for the new database.

You can override the default settings at the database level by using [CREATE DATABASE](https://www.mysqltutorial.org/mysql-basics/mysql-create-database/) statement:



or using the ALTER DATABASE statement:

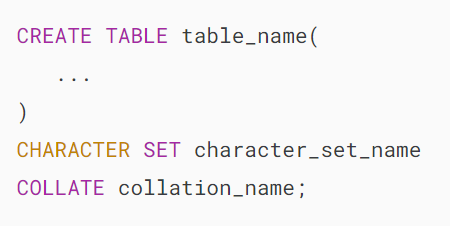


MySQL will use the database’s character set and collation by default for the tables that you create in the database.

### **3) Setting character sets and collations at the table level**

A database may contain tables with character sets and collations that are different from the database’s character set and collation.

You can specify the default character set and collation for a table when you create the table by using the [CREATE TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) statement:



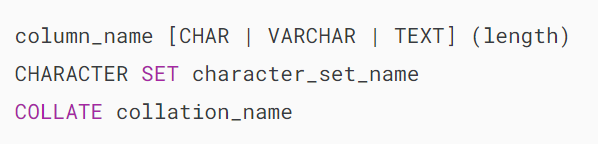
or when you alter the table using the [ALTER TABLE](https://www.mysqltutorial.org/mysql-basics/mysql-alter-table/)  statement:

### 

### **4) Setting character set and collation at column level**

A column of type [CHAR](https://www.mysqltutorial.org/mysql-basics/mysql-char-data-type/) , [VARCHAR](https://www.mysqltutorial.org/mysql-basics/mysql-varchar/) or [TEXT](https://www.mysqltutorial.org/mysql-basics/mysql-text/) can have its own character set and collation different from the table’s character set and collation.

The CREATE TABLE and ALTER TABLE statement allows you to override the character set and collation for a specific column:



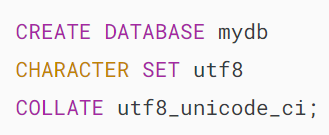
The rules for setting the character set and collation are:

* If you specify both a character set and a collation explicitly, the character set and collation are used.
* If you specify a character set and omit the collation, the default collation of the character set is used.
* If you specify a collation without a character set, the character set associated with the collation is used.
* If you omit both character set and collation, the default character set and collation are used.

Let’s take a look at some examples of setting the character sets and collations.

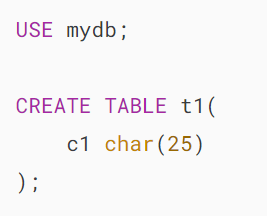
## **Examples of setting character sets and collations**

First, create a new database with utf8 as the character set and utf8\_unicode\_ci as the default collation:



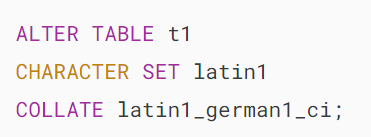
Because we specify the character set and collation for the mydb database explicitly, the mydb won’t take the default character set and collation of the database server.

Second, create a new table named t1 in the mydb database:



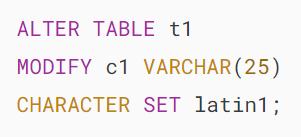
In this CREATE TABLE statement, we don’t specify the character set and collation for the t1 table. Therefore, MySQL will use the character set and collation of the database for the t1 table. In this case, the t1  table will have utf8 as the default character set and utf8\_unicode\_ci as the default collation.

Third, change the character set and collation of the table t1 to latin1 and latin1\_german1\_ci:



The c1 column in the t1 table has latin1 as the character set and latin1\_german1\_ci as the collation.

Finally, change the character set of the c1 column to latin1 :



Now, the c1 column has the latin1  character set, but what about its collation? Is it inheriting the latin1\_german1\_ci collation from the table’s collation?

The answer is no. The reason is that the default collation of the latin1 character set is latin1\_swedish\_ci, therefore, the c1 column will have the latin1\_swedish\_ci collation.

In this tutorial, you have learned about MySQL collation and how to specify character sets and collations for the MySQL server, databases, tables, and columns.

## **Reference**

* <https://dev.mysql.com/doc/refman/8.0/en/charset.html> – MySQL character set support
* <http://collation-charts.org/mysql60/> – MySQL collation charts

# **Import CSV File into MySQL Table**

Summary: in this tutorial, you will learn how to import a CSV file into a MySQL table using the LOAD DATA INFILE statement and MySQL Workbench.

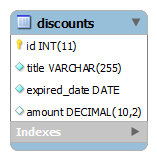
## **1) Importing a CSV file on the MySQL server into a table using LOAD DATA INFILE statement**

The LOAD DATA INFILE statement allows you to read data from a CSV file in a specified directory on the MySQL server and import its contents to into a table.

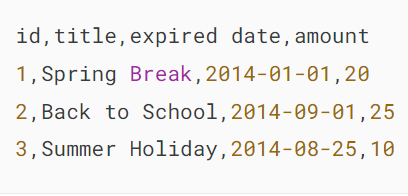
Before importing the file, you need to prepare the following:

* A table that you want to import data into.
* A CSV file with data that matches the number of columns of the table and the type of data in each column.
* A MySQL user account that has FILE and INSERT privileges.

Suppose you have a table called discounts with the following structure:

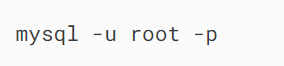


And the following  discounts.csv file contains the first line as column headings and the other three lines of data:



To import the discounts.csv file into the discounts table, you follow these steps:

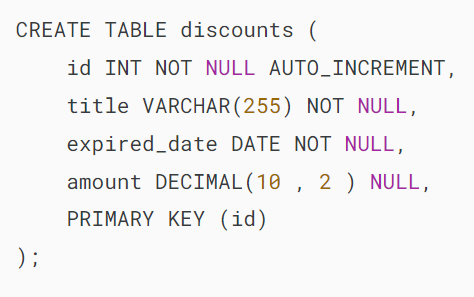
First, open the Command Prompt on Windows or the Terminal on Unix-like systems and [connect to the MySQL server](https://www.mysqltutorial.org/getting-started-with-mysql/connect-to-mysql-server/):



Second, change the current database to classicmodels:



Third, create a discounts table:



Fourth, show the value of the @@secure\_file\_priv variable:



The secure\_file\_priv option indicates the directory where you are allowed to store the input file and execute it with the LOAD DATA INFILE statement.

Fifth, copy the discounts.csv file to the directory specified by the secure\_file\_priv option.

Sixth, import data from the discounts.csv file into the discounts table by executing the following statement:

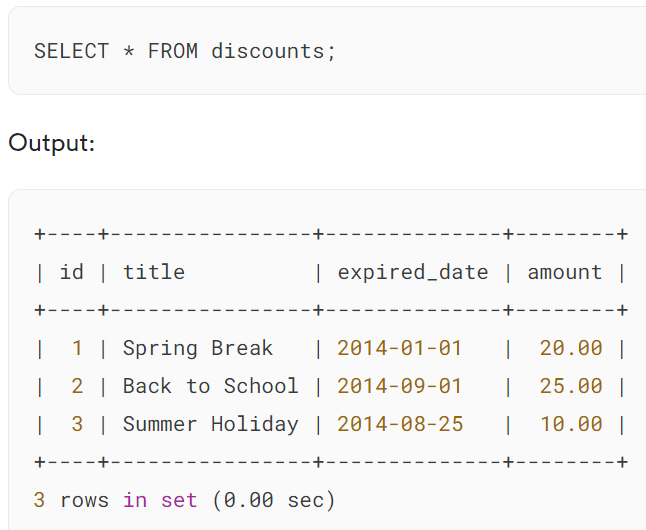


The output indicates that the statement has loaded the file with three rows successfully.

Here’s how the LOAD DATA INFILE works:

* LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/discounts.csv': Specifies the path to the input file (discounts.csv) that contains the data to be loaded into the table.
* INTO TABLE discounts: Specifies the target table (discounts) where you want to load the data
* FIELDS TERMINATED BY ',': Specifies that the fields in the input file are separated (terminated) by a comma (,). This indicates that the file is a CSV (Comma-Separated Values) file.
* ENCLOSED BY '"': Specifies that the fields in the input file are enclosed by double quotation marks ("). This is common in CSV files to handle cases where a field may contain the delimiter ,.
* LINES TERMINATED BY '\n': Specifies that each line in the input file is terminated by a newline character (\n). This indicates the end of a record (row) in the CSV file.
* IGNORE 1 ROWS: Instructs the statement to ignore the first row in the input file. This is useful when the first row contains headers and should not be imported as data.

Finally, retrieve the data from the discounts table to verify the import:



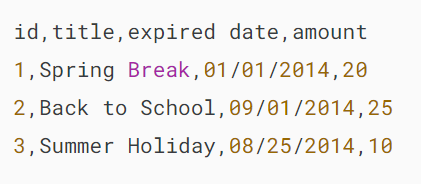
Notice that if you don’t place the file in the directory specified by the secure\_file\_priv variable, you’ll get the following error:

### 

### **Transforming data while importing**

Sometimes the format of the data does not match the target columns in the table. In simple cases, you can transform it by using the SET clause in the  LOAD DATA INFILE statement.

Suppose the expired date column in the  discount2.csv file is in the  mm/dd/yyyy format:



When importing data into the discounts table, you can transform it into MySQL date format by using [STR\_TO\_DATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-str_to_date/) function:

First, [truncate](https://www.mysqltutorial.org/mysql-basics/mysql-truncate-table/) the discounts table:



Second, transform and load data from the discount2.csv file into the discounts table:

## 

## **2) Importing a CSV file from a local computer to a table on a remote MySQL server**

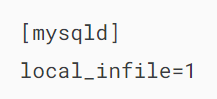
The LOAD DATA INFILE allows you to import a CSV file from your local computer to a table in a remote MySQL server via the LOCAL option.

### **Configuration**

MySQL disables loading a local file to the server by default. To load a local file to the MySQL server, you need to enable the option on both the client and server sides.

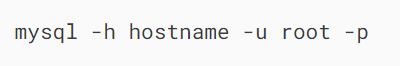
The local\_infile controls whether the client and server permit the use of the LOCAL option in the LOAD DATA INFILE statement.

First, open the MySQL configuration file (my.ini or my.cnf) and add the following line to enable loading the local file on the MySQL server:



Second, restart the MySQL server to apply the change.

Third, open the mysql client program on the local computer to connect to the MySQL server



You need to replace the hostname with your remote MySQL server.

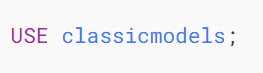
Fourth, set the global variable local\_infile to 1 (or ON):



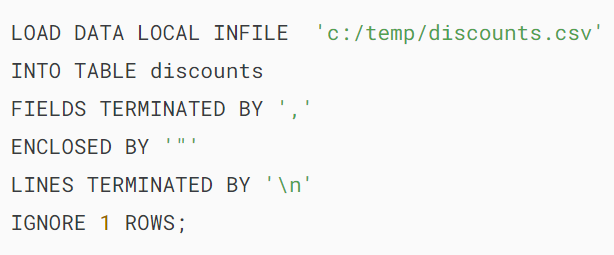
We have configured the local\_infile on both the server and client.

### **Loading the local file**

First, change the current database to the one that contains the discounts table:



Second, execute the LOAD DATA INFILE statement to import data from a CSV file located in the C:\temp\ directory to the MySQL server:



The only addition to the LOAD DATA INFILE statement is the LOCAL option.

If you load a big CSV file, you will see that with the LOCAL option, it will take time to transfer the file to the MySQL server.

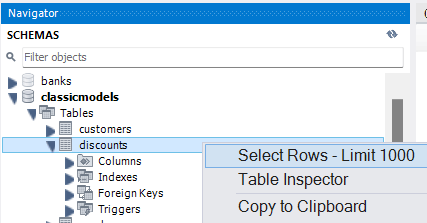
Note that if you don’t configure both client and server properly, you will get the following message:

## 

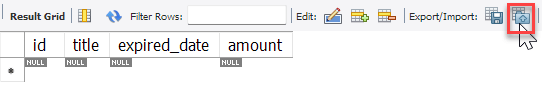
## **3) Importing CSV files into a table using MySQL Workbench**

MySQL workbench provides a tool to import data into a table

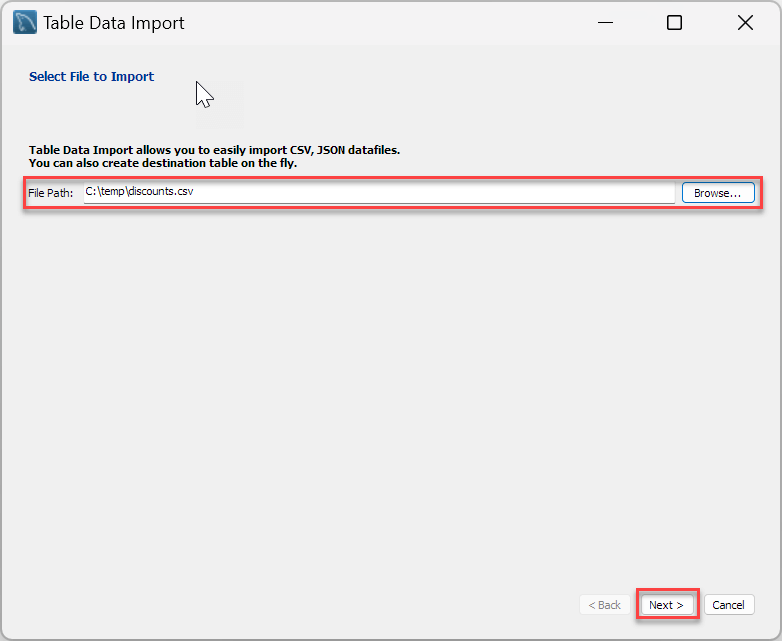
First, open the discounts table:



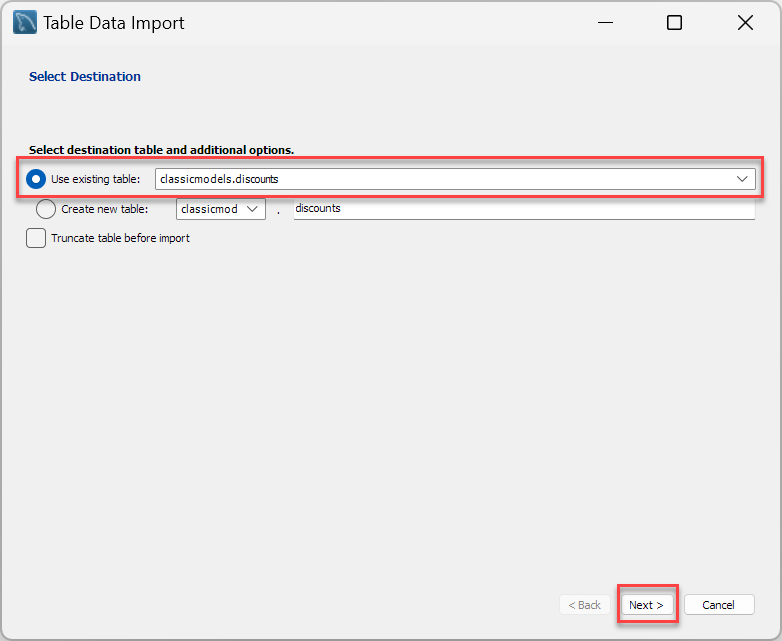
Second, click the import button:



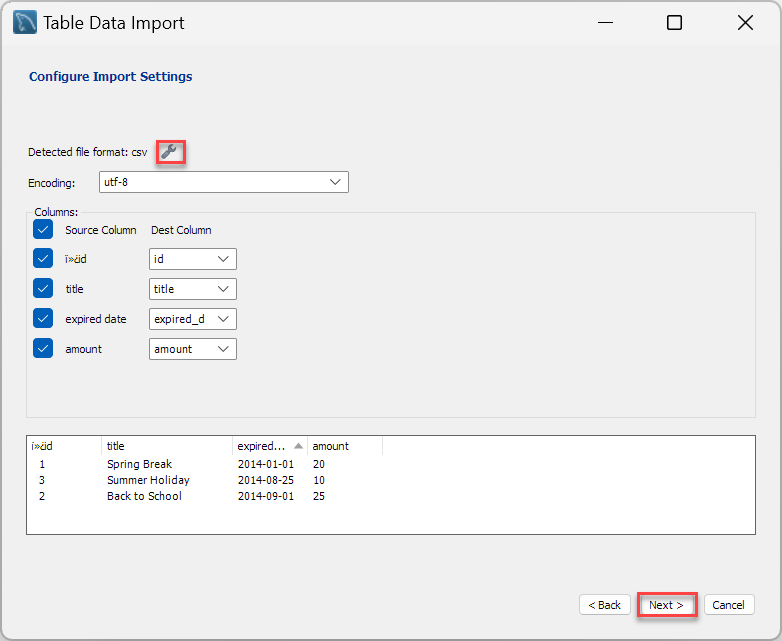
Third, select the path to the CSV file and click the Next button:



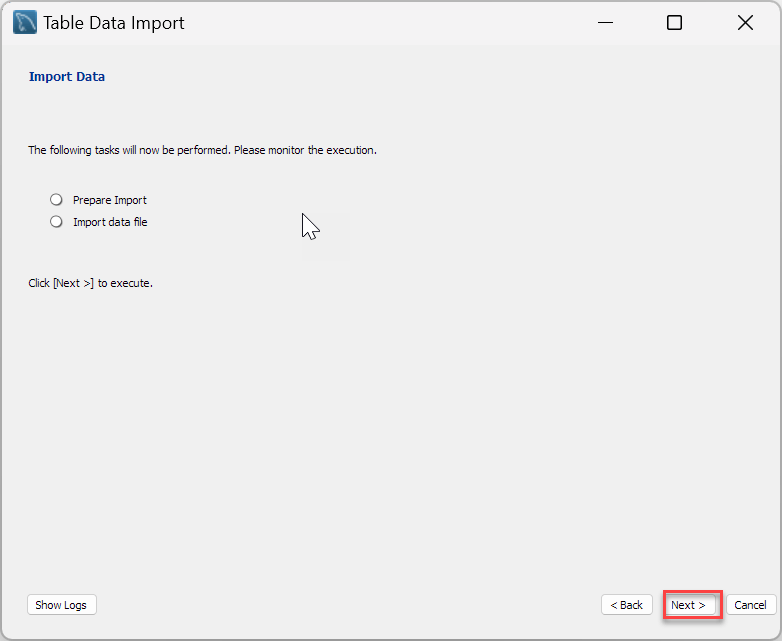
Fourth, select the destination table, which is the classicmodels.discounts in this case. Note that you can create a new table before importing the file and/or truncate the table before import:



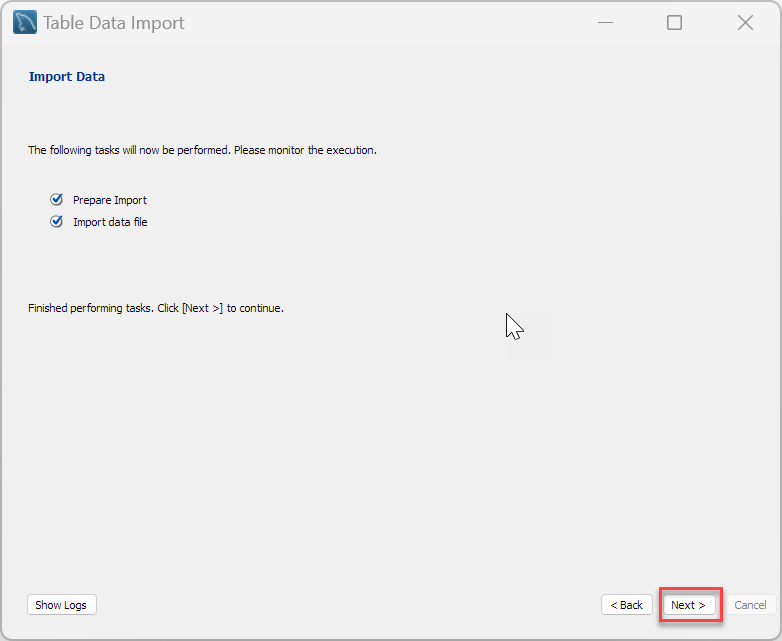
Fifth, map the columns in the source file with the columns in the destination table and click the Next button:



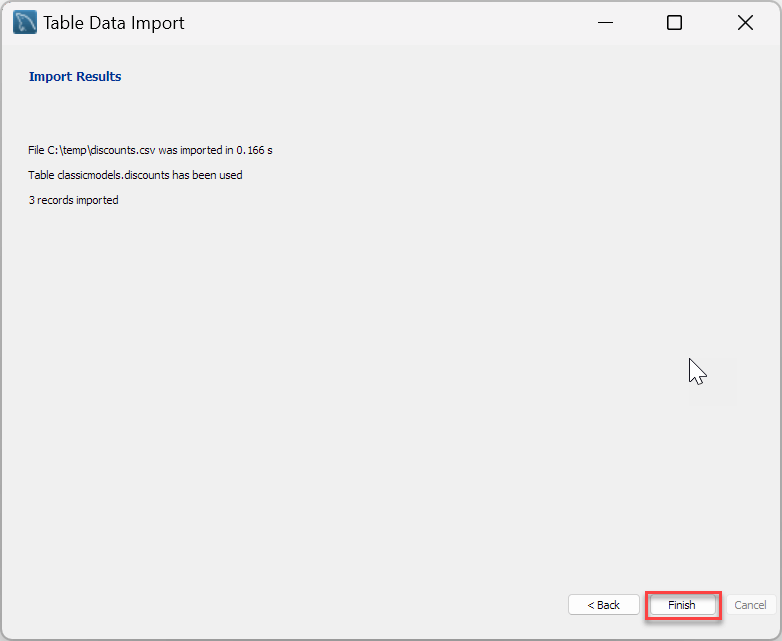
Sixth, review the steps that the Workbench will do and click the Next button:



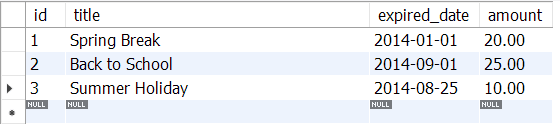
Seventh, review the steps that the Workbench will do and click the Next button:



Eight, review the import result and click the Finish button:



Finally, show the contents of the discounts table:



## **Summary**

* Use the LOAD DATA INFILE statement to import a CSV file into a table.
* Use MySQL Workbench to import a CSV file from the local computer to a table on a remote MySQL server.

# **MySQL Export Table to CSV**

**Summary**: in this tutorial, you will learn various techniques for exporting a MySQL table to a CSV file.

The CSV stands for comma-separated values. The CSV file format is often used to exchange data between applications such as Microsoft Excel, Open Office, Google Docs, and so on.

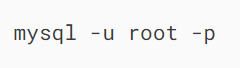
Having data from the MySQL database in CSV file format will be useful because you can analyze and format the data in the way you want.

MySQL provides an easy way to export the query’s result into a CSV file that resides on the database server.

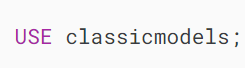
## **1) Exporting a table to a CSV file using SELECT … INTO OUTFILE statement**

We’ll illustrate how to export the orders table from the classicmodels [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) into a CSV file located on the MySQL Server.

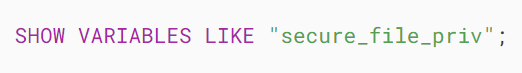
First, open the Command Prompt on Windows or Terminal on Unix-like systems and connect to the MySQL server:



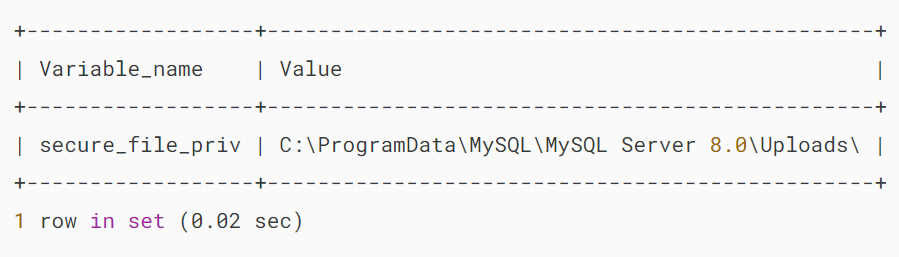
Second, change the current database to classicmodels:



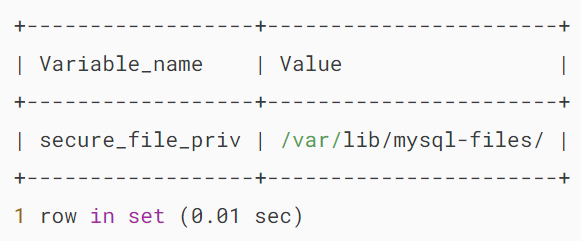
Third, show the value of the secure\_file\_priv variable:



On Windows server, the output is like the following:



On Ubuntu, the output will look like:



The secure\_file\_priv setting indicates the directory where you are allowed to store the output file.

Finally, retrieve data from the orders table and export the result set into the orders.csv file:



Notice that the orders.csv file must not exist in the directory C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/. Otherwise, the statement will issue an error:

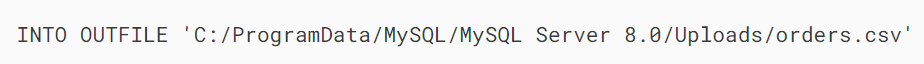
 

How it works:

First, retrieve data from the orders table:

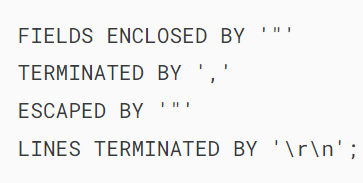


Second, specify the CSV output file (orders.csv) stored in the allowed directory (C:/ProgramData/MySQL/MySQL Server 8.0/Uploads) in the INTO OUTFILE clause:



Notice you have to replace the backslash \ by the forward slash / in the directory on Windows to make it work.

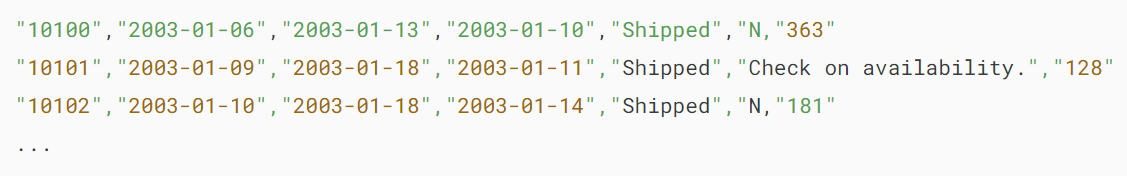
Third, define the format of the output file as CSV by specifying how fields and lines in the output are formatted:



Here is the detail of each option:

* **FIELDS ENCLOSED BY ‘”‘:** specifies that each field in the output file will be enclosed by double quotation marks (").
* **TERMINATED BY ‘,’:** specifies that fields in the output file are separated (terminated) by a comma (,).
* **ESCAPED BY ‘”‘:** specifies the character used to escape special characters. In this case, it’s a double quotation mark (").
* **LINES TERMINATED BY ‘\r\n’:** specifies that each line in the output file is terminated by a carriage return (\r) followed by a newline character (\n), which is a common line-ending sequence in Windows environments.

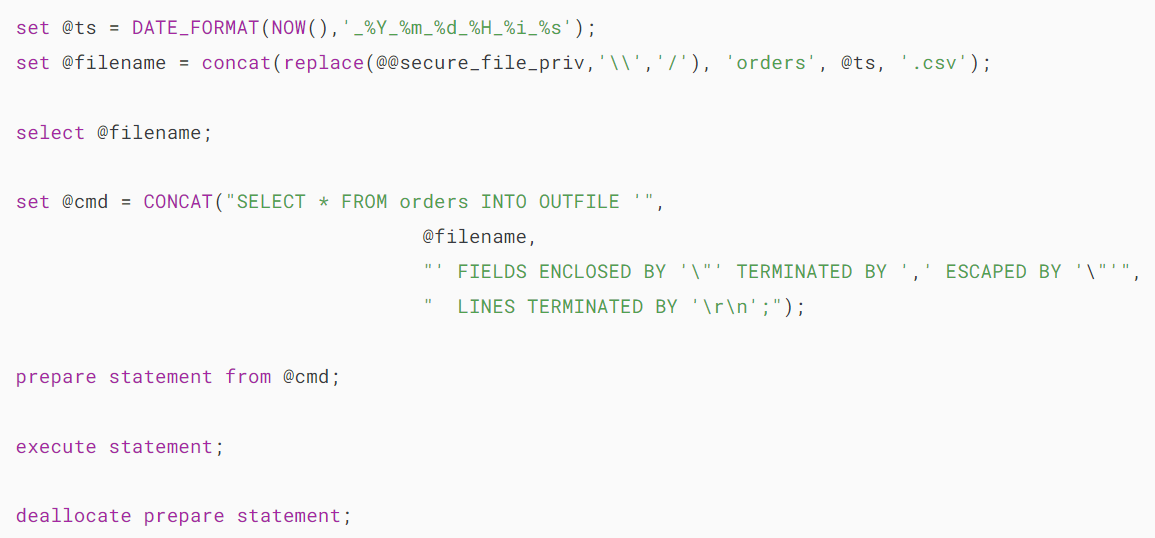
Here is the excerpt from the orders.csv file:



### **Adding a timestamp to the CSV output file**

You often need to export data into a CSV file whose name contains a timestamp indicating when the file is created. To achieve this, you need to use a [MySQL prepared statement](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-prepared-statement/).

The following commands export the entire orders table into a CSV file with a timestamp as a part of the file name:

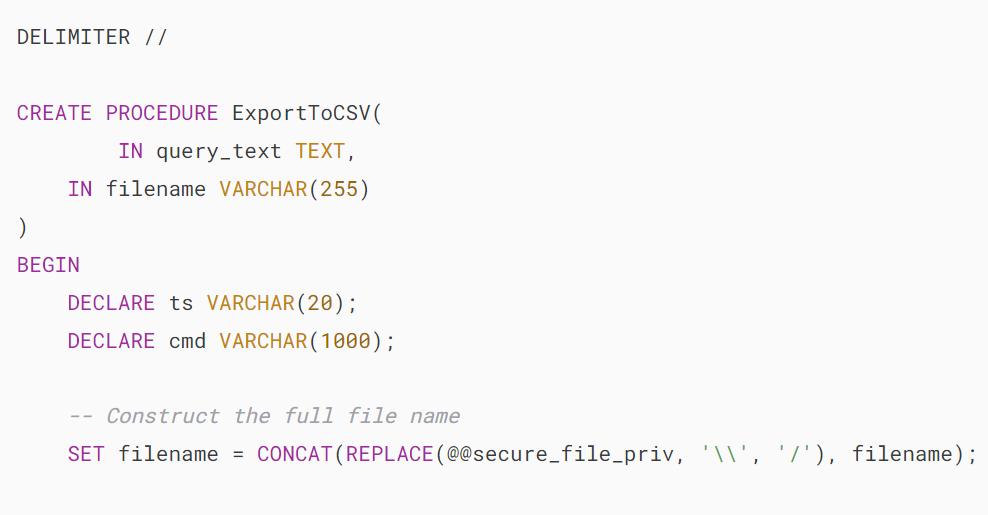


In these statements:

* First, construct a query with the current timestamp as a part of the file name.
* Second, prepare the statement for execution by using a PREPARE statement.
* Third, execute the statement by using the EXECUTE command.

### **Creating a stored procedure that exports a query result into a CSV file**

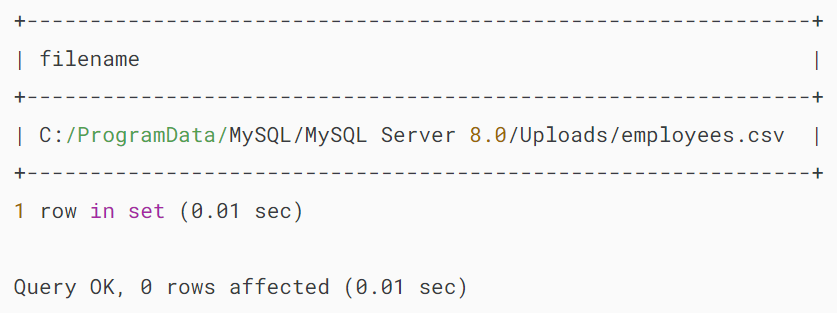
To simplify the process of exporting the result set of a query into a CSV file, we can create a stored procedure that generates a CSV file from a query’s result set:

For example, you can export the employees table to employees.csv file as follows:



Output:

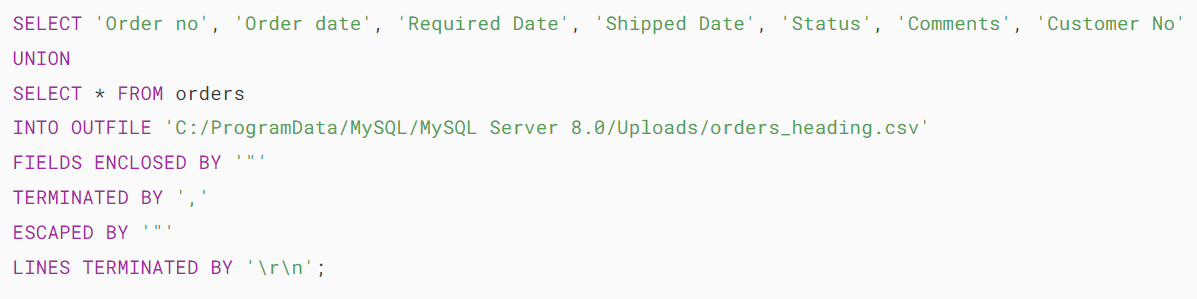


You can retrieve the CSV file in the output: C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/employees.csv

### **Adding column heading to the CSV output file**

It would be convenient if the CSV file contains the first line as the column headings so that the file is more understandable.

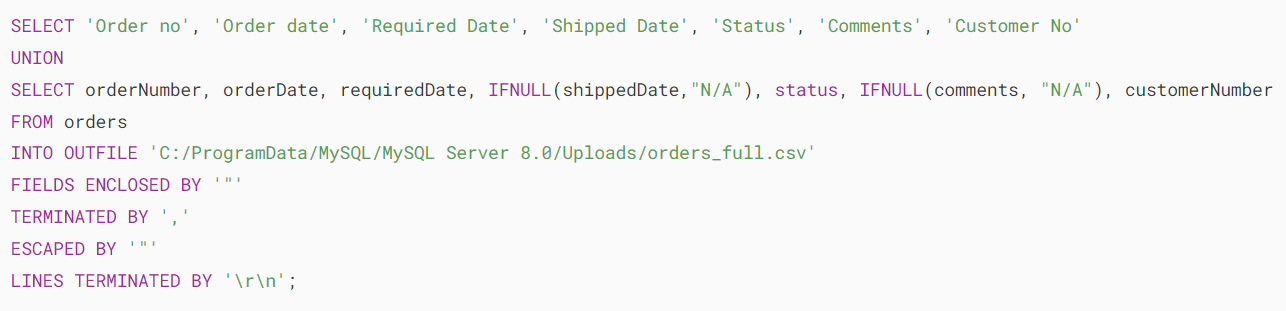
To add the column headings, you can use a [UNION](https://www.mysqltutorial.org/mysql-basics/mysql-union/) operator as follows:



### **Mapping NULL to other values**

If the values in the result set contain [NULL](https://www.mysqltutorial.org/mysql-basics/mysql-null/) values, the target file will contain  "N instead of NULL.

To fix this issue, you need to replace the NULL value by another value, for example, not available ( N/A ) by using the [IFNULL function](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-ifnull/) as shown in the following query:

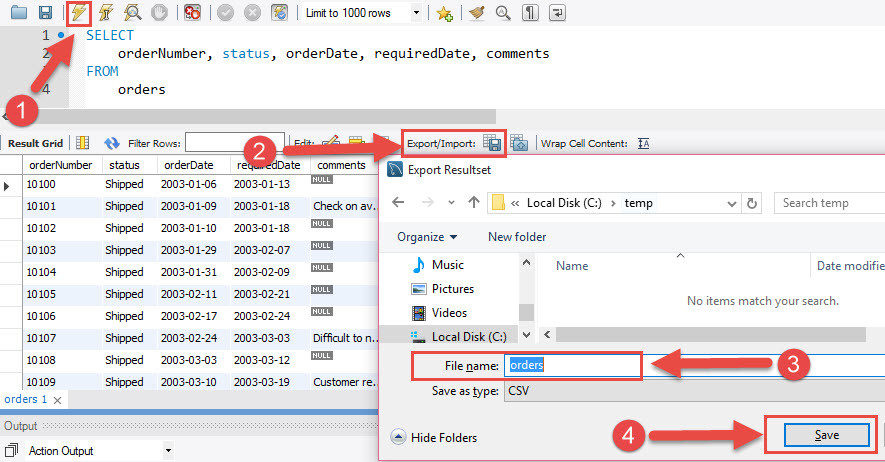


In the SELECT clause, we replace NULL in the shippedDate and comments columns with the string "N/A".

## **2) Exporting data to CSV file using MySQL Workbench**

If you don’t have access to the MySQL server to retrieve the exported CSV file, you can use MySQL Workbench to export the result set of a query to a CSV file in your local computer as follows:

* First, execute a query to get its result set.
* Second, from the result panel, click “export recordset to an external file”. Note that a result set is also known as a recordset.
* Third, a new dialog is displayed and asks you for a filename and file format. Enter the file name, choose CSV as the file format, and click the Save button.



The CSV file exported by MySQL Workbench supports column headings, NULL values and other features.

## **Summary**

* Use the SELECT ... INTO OUTFILE statement to export a table to a CSV file on the MySQL Server.
* Use MySQL Workbench to export a table to a CSV file on your local computer.