# **MySQL Date Functions**

This page shows you the most commonly used MySQL Date functions that allow you to manipulate date and time data effectively.

## **Section 1. Getting the current Date & Time**

This section explains the functions that allow you to retrieve the current date, time, or both.

* [CURDATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-curdate/) – Return the current date. ( synonyms: CURRENT\_DATE() & CURRENT\_DATE).
* [CURRENT\_TIME](https://www.mysqltutorial.org/mysql-date-functions/mysql-current_time/) – Return the current time ( synonyms: CURRENT\_TIME() & CURTIME() ).
* [NOW()](https://www.mysqltutorial.org/mysql-date-functions/mysql-now-function/) – Return the current date and time ( synonyms: CURRENT\_TIMESTAMP(), CURRENT\_TIMESTAMP, LOCALTIME(), LOCALTIMESTAMP()).
* [SYSDATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-sysdate/) – Return the time at which it executes.
* [UTC\_TIMESTAMP()](https://www.mysqltutorial.org/mysql-date-functions/mysql-utc_timestamp/) – Return the current UTC date and time.
* [UTC\_DATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-utc_date/) – Return the current UTC date.
* [UTC\_TIME()](https://www.mysqltutorial.org/mysql-date-functions/mysql-utc_time/) – Return the current UTC time.

## **Section 2. Calculating Date and Time**

* [ADDTIME()](https://www.mysqltutorial.org/mysql-date-functions/mysql-addtime/) – Add a time interval to a time value or datetime value.
* [DATE\_ADD()](https://www.mysqltutorial.org/mysql-date-functions/mysql-date_add/) – Add a time value to a date (synonyms: ADDDATE()).
* [DATE\_SUB()](https://www.mysqltutorial.org/mysql-date-functions/mysql-date_sub/) – Subtract a time value (interval) from a date.
* [DATEDIFF()](https://www.mysqltutorial.org/mysql-date-functions/mysql-datediff-function/) – Return the difference in days of two date values.
* [TIMEDIFF()](https://www.mysqltutorial.org/mysql-date-functions/mysql-timediff/) – Return the difference of two time values.
* [TIMESTAMPADD()](https://www.mysqltutorial.org/mysql-date-functions/mysql-timestampadd/) – Add or subtract an interval from a timestamp or date.
* [TIMESTAMPDIFF()](https://www.mysqltutorial.org/mysql-date-functions/mysql-timestampdiff/) – Return the difference between two timestamp values.
* [TIME\_TO\_SEC()](https://www.mysqltutorial.org/mysql-date-functions/mysql-time_to_sec/) – Return the number of seconds from a time argument.
* [TO\_DAYS()](https://www.mysqltutorial.org/mysql-date-functions/mysql-to_days/) – Return a day number (the number of days since year 0) from a given date.

## **Section 3. Converting Functions**

* [CONVERT\_TZ()](https://www.mysqltutorial.org/mysql-date-functions/mysql-convert_tz/) – Convert a datetime value from one time zone to another.
* [FROM\_DAYS()](https://www.mysqltutorial.org/mysql-date-functions/mysql-from_days/) – Convert a numeric day count into a date.
* [STR\_TO\_DATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-str_to_date/) – Convert a string to date.
* [FROM\_UNIXTIME()](https://www.mysqltutorial.org/mysql-date-functions/mysql-from_unixtime/) – Convert UNIX timestamps into a readable date and time format.
* [UNIX\_TIMESTAMP()](https://www.mysqltutorial.org/mysql-date-functions/mysql-unix_timestamp/) – Convert a datetime to a UNIX timestamp.

## **Section 4. Formatting Date & Time functions**

* [DATE\_FORMAT()](https://www.mysqltutorial.org/mysql-date-functions/mysql-date_format/) – Return a string representation of a date based on a format.
* [TIME\_FORMAT()](https://www.mysqltutorial.org/mysql-date-functions/mysql-time_format-function/) – Return a string representation of a time based on a format.
* [GET\_FORMAT()](https://www.mysqltutorial.org/mysql-date-functions/mysql-get_format/) – Return a format string for a date, time, datetime, or timestamp.

## **Section 5. Extracting Date & Time Functions**

The extraction functions allow you to extract date and time components from a date and time.

* [DATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-date-function/) – Extract the date component from a date.
* [EXTRACT()](https://www.mysqltutorial.org/mysql-date-functions/mysql-extract/) – Extract a component of a date.
* [YEAR()](https://www.mysqltutorial.org/mysql-year/) – Return the year component of a date.
* [YEARWEEK()](https://www.mysqltutorial.org/mysql-date-functions/mysql-yearweek/) – Return the year and week for a date.
* [QUARTER()](https://www.mysqltutorial.org/mysql-date-functions/mysql-quarter/) – Return the quarter of the year for a date.
* [MONTH()](https://www.mysqltutorial.org/mysql-month/) – Return the month component of a date.
* [WEEK()](https://www.mysqltutorial.org/mysql-date-functions/mysql-week/) – Return the week component of a date.
* [WEEKDAY()](https://www.mysqltutorial.org/mysql-weekday/) – Return the weekday index of a date.
* [WEEKOFYEAR()](https://www.mysqltutorial.org/mysql-date-functions/mysql-week/) – Return the calendar week of the date (1-53) – equivalent to WEEK(date, 3).
* [DAY()](https://www.mysqltutorial.org/mysql-date-functions/mysql-day/) – Return the day of the month for a specific date (1-31). DAYOFMONTH is the synonym for DAY.
* [DAYOFYEAR()](https://www.mysqltutorial.org/mysql-date-functions/mysql-dayofyear/) – Return the day of the year (1-366).
* [DAYOFWEEK()](https://www.mysqltutorial.org/mysql-dayofweek/) – Return the day of the week (1-7).
* [HOUR()](https://www.mysqltutorial.org/mysql-date-functions/mysql-hour/) – Return the hour for a time.
* [MINUTE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-minute-function/) – Return the minute for a time.
* [SECOND()](https://www.mysqltutorial.org/mysql-date-functions/mysql-second/) – Return the second for a time.
* [LAST\_DAY()](https://www.mysqltutorial.org/mysql-date-functions/mysql-last_day/) – Return an integer that represents the last day of the month for a specific date.

## **Section 6. Getting Month & Day Names**

This section shows you how to use functions to get the month and day names.

* [DAYNAME()](https://www.mysqltutorial.org/mysql-date-functions/mysql-dayname/) – Return the name of the day for a specific date.
* [MONTHNAME()](https://www.mysqltutorial.org/mysql-date-functions/mysql-monthname/) – Return the name of the month for a specific date.

## **Section 7. Creating Date & Time Functions**

* [MAKEDATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-makedate/) – create a date based on a given year and the number of days.
* [MAKETIME()](https://www.mysqltutorial.org/mysql-date-functions/mysql-maketime/) – create a time based on hour, minute, and second.

## **Section 8. Handling Period Functions**

This section covers the function that manipulates the periods in the format YYMM or YYMMMM.

* [PERIOD\_ADD()](https://www.mysqltutorial.org/mysql-date-functions/mysql-period_add/) – add a number of months to a period in the format YYMM or YYMMMM.
* [PERIOD\_DIFF()](https://www.mysqltutorial.org/mysql-date-functions/mysql-period_diff/) – calculate the difference in months of two periods represented in the format YYMM or YYYYMM.

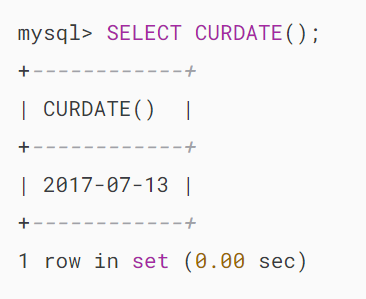
# **MySQL CURDATE() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL CURDATE() function to get the current date.

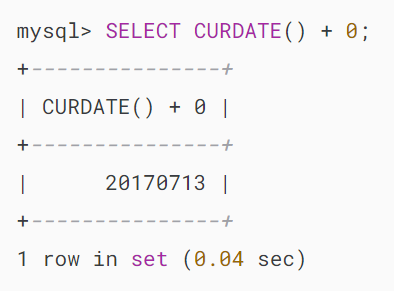
## **Introduction to MySQL CURDATE() function**

The CURDATE() function returns the current [date](https://www.mysqltutorial.org/mysql-basics/mysql-date/) as a value in the 'YYYY-MM-DD' format if it is used in a string context or YYYMMDD format if it is used in a numeric context.

The following example shows how the CURDATE() function is used in the string context.



The following example illustrates how the CURDATE() function is used in a numeric context:



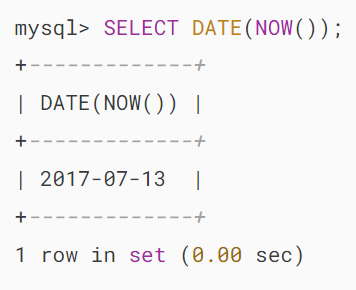
The CURRENT\_DATE and CURRENT\_DATE() are synonyms for CURDATE().

## 

## **CURDATE vs. NOW**

The CURDATE() function returns the current date with the date part only while the [NOW()](https://www.mysqltutorial.org/mysql-date-functions/mysql-now-function/) function returns both date and time parts of the current time.

The result of the CURDATE() function is equivalent to the following expression:



# **MySQL CURRENT\_TIME**

**Summary**: in this tutorial, you will learn how to use the MySQL CURRENT\_TIME function to get the current time.

## **Introduction to MySQL CURRENT\_TIME function**

The CURRENT\_TIME function returns the current time. Here’s the syntax of the CURRENT\_TIME function:



The CURRENT\_TIME doesn’t accept any arguments and returns the current time value in 'hh:mm:ss' if it is used in a string context and hhmmss if it is used in a numeric context.

Besides the CURRENT\_TIME function, you can use the CURRENT\_TIME() function or CURTIME() function to get the current time. And you can use these functions interchangeably because they’re synonyms.

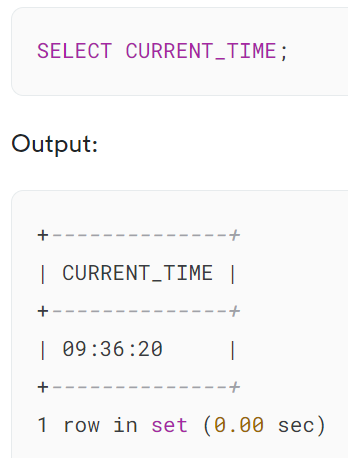
Note that you cannot use

## **MySQL CURRENT\_TIME function example**

Let’s take some examples of using the CURRENT\_TIME function.

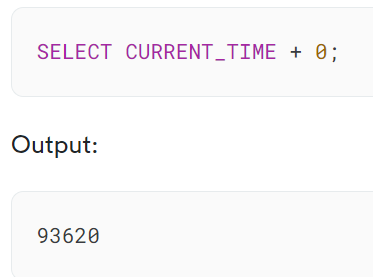
### **1) Simple CURRENT\_TIME function examples**

The following example uses the CURRENT\_TIME function to get the current time:



In this example, the CURRENT\_TIME function returns the current time as a string in the format 'hh:mm:ss'. The reason is that we use the CURRENT\_TIME function in the string context.

If we use the CURRENT\_TIME function in a numeric context, it’ll return the current time in the hhmmss format as shown in the following example:



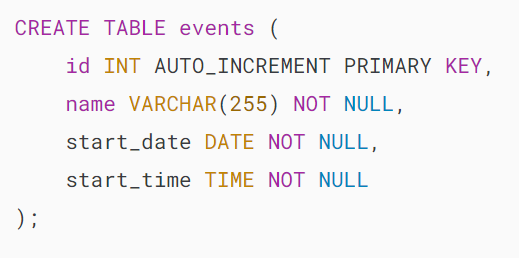
The output 93620 has three parts:

* Hour: 9
* Minutes: 36
* Seconds: 20

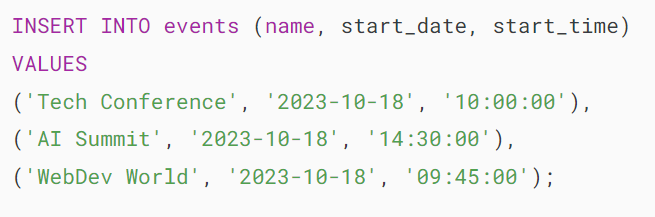
### **2) Using the MySQL CURRENT\_TIME with table data**

In the following example, we’ll show you how to use the CURRENT\_TIME function to query data from a table.

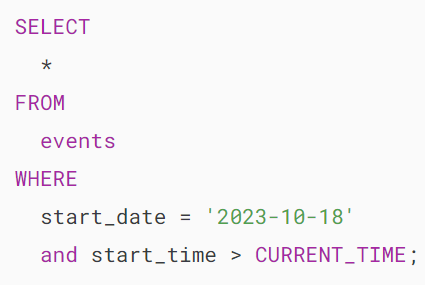
First, [create a new table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) called events with the following structure:



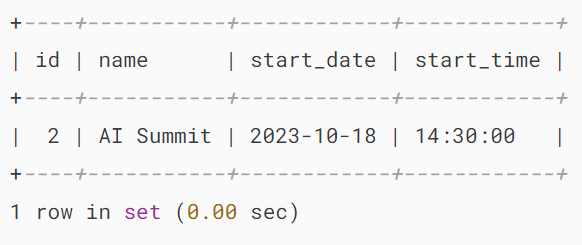
Second, [insert some rows](https://www.mysqltutorial.org/mysql-basics/mysql-insert-multiple-rows/) into the events table:



Third, use the CURRENT\_TIME function to get the events that have not yet started on 2023-10-18:



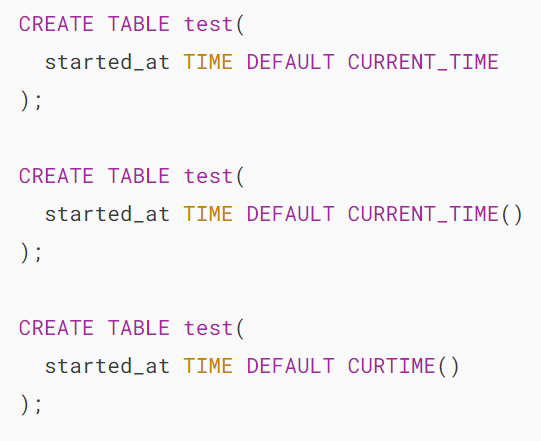
Output:



Note that the query will return the result based on your current time when you run it.

## **Default current time value**

Note that you cannot use the CURRENT\_TIME, CURRENT\_TIME(), or CURTIME() function as a default value for a column in a table. Hence, the following statements will result in an error:



To use the current time as a default value for a column, you use the [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) or [TIMESTAMP](https://www.mysqltutorial.org/mysql-basics/understanding-mysql-timestamp/) as the column’s type and the [NOW()](https://www.mysqltutorial.org/mysql-date-functions/mysql-now-function/) function as the default value. For example:

## 

## **Summary**

* Use the CURRENT\_TIME function to get the current time as a string in the format 'hh:mm:ss' or as a number in the format hhmmss.

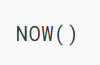
# **MySQL NOW() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL NOW() function to get the current date and time of the server.

## **Introduction to MySQL NOW() function**

The MySQL NOW() function returns the current date and time in the configured time zone as a string or a number in the 'YYYY-MM-DD HH:MM:DD' or 'YYYYMMDDHHMMSS.uuuuuu' format.

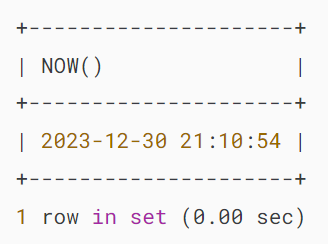
Here’s the syntax of the NOW() function:



The returned type of the NOW() function depends on the context where you use it.

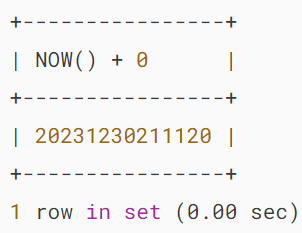
For example, in the following statement, the NOW() function returns the current date and time as a string:



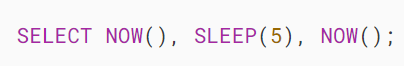


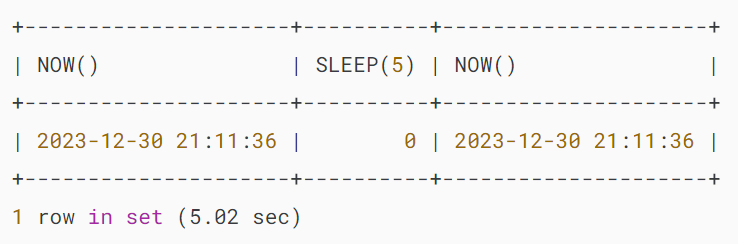
However, in the numeric context, the NOW() function returns the current date and time as a number as shown in the following example:





Notice that the NOW() function returns the date and time at which the statement started executing. For example:

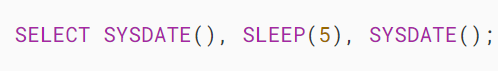


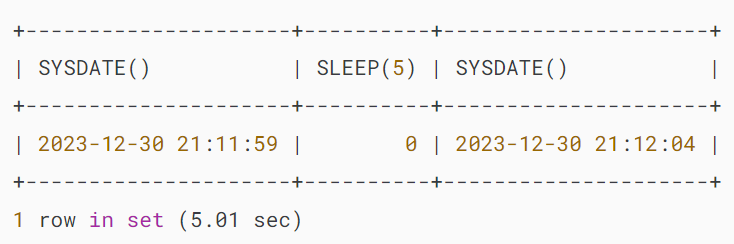


In the query, the first NOW() function executed, and the SLEEP(5) function paused the execution of the query for 5 seconds, and the second NOW() function executed.

However, both NOW() functions return the same value, even when they are executed at different times.

If you want to get the exact time at which the statement executes, you use the [SYSDATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-sysdate/) function instead. For example:





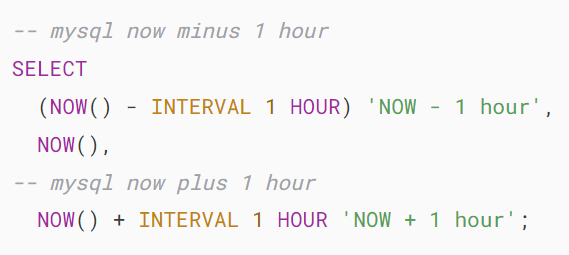
If you want to change the MySQL server’s time zone to adjust the current date and time returned by the NOW() function, you use the following statement:

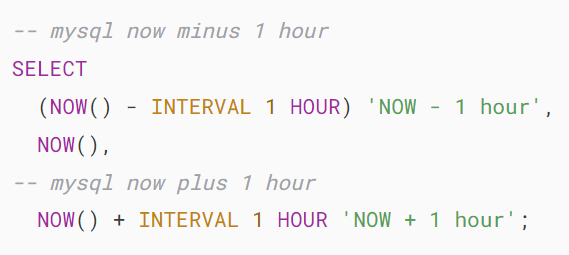
## 

## **MySQL NOW() function calculations**

Because the NOW() function returns a number when it is used in a numeric context, you can use it in calculations e.g., now plus 1 hour, now minus 1 hour, and now plus 1 day.

The following statement returns the current date and time, now minus 1 hour and now plus 1 hour:





The following statement returns the current date and time, now minus 1 day and now plus 1 day:

## 

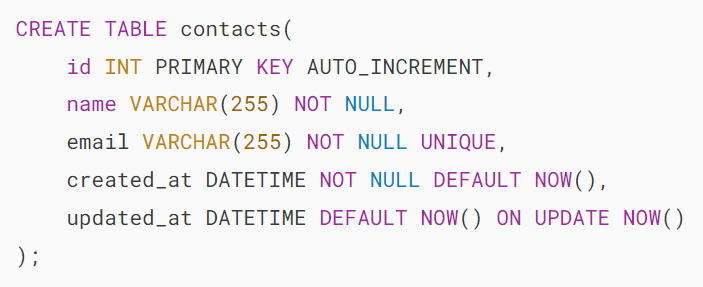
## 

## **Using the NOW() function as the default value for a column**

You can use the NOW() function to provide a default value for a DATETIME or TIMESTAMP column.

When you omit the date or time value in the [INSERT](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) statement, MySQL automatically inserts the current date and time into the column whose default value is NOW(). For example:

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

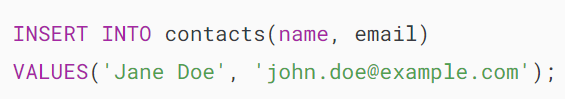


The contacts table includes the following columns:

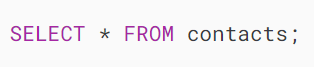
* id : This is the primary key column of the contacts table.
* name: This represents the name of the contact.
* email: This column stores the contact’s email.
* created\_on : This column has the default value specified by the NOW() function.
* updated\_at: This column sets updated\_at with a default value of the current date and time when a new row is inserted. Additionally, the ON UPDATE NOW() clause ensures that the column is automatically updated to the current date and tie whenever the row is updated.

Notice that CURRENT\_TIMESTAMP and CURRENT\_TIMESTAMP() are synonyms for NOW() so you can use them interchangeably.

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



Third, retrieve data from the contacts table:

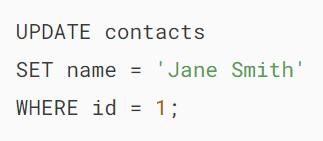


Output:



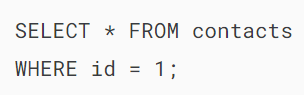
In the INSERT statement, we don’t provide the values for the created\_at and updated\_at columns. Therefore, they use the default value which is the current date and time returned by the NOW() function.

Fourth, update the row with id 1:



In the UPDATE statement, we don’t specify the value for the updated\_at column so it is updated to the time the row is updated.

Finally, retrieve the data from the contacts table:



Output:

## 

## **Summary**

* Use the NOW() function to get the current date and time of the server.
* Use the DEFAULT NOW() for a column to set the default value for the column to the current date and time when a row is inserted.
* Use the ON UPDATE NOW() for a column to set a default for a column to the current date and time when a row is updated.

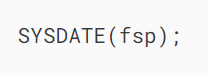
# **MySQL SYSDATE Function**

**Summary**: in this tutorial, you will learn about the MySQL SYSDATE() function and its caveat.

## **Introduction to MySQL SYSDATE() function**

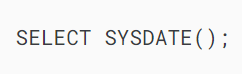
The SYSDATE() function returns the current [date](https://www.mysqltutorial.org/mysql-basics/mysql-date/) and [time](https://www.mysqltutorial.org/mysql-basics/mysql-time/) at which it is executed. If you use the function in the string context, the return value in the 'YYYY-MM-DD HH:MM:SS' format. However, if you use the function in a numeric context, it returns a value in the YYYYMMDDHHMMSS format.

Here’s the basic syntax of the SYSDATE() function:

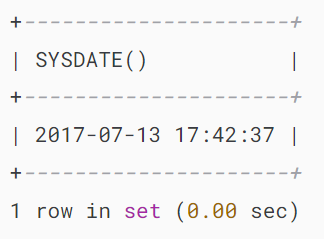


The SYSDATE() function accepts an optional argument fsp that determines whether the result should include a fractional seconds precision which ranges from 0 to 6.

See the following example:



Output:

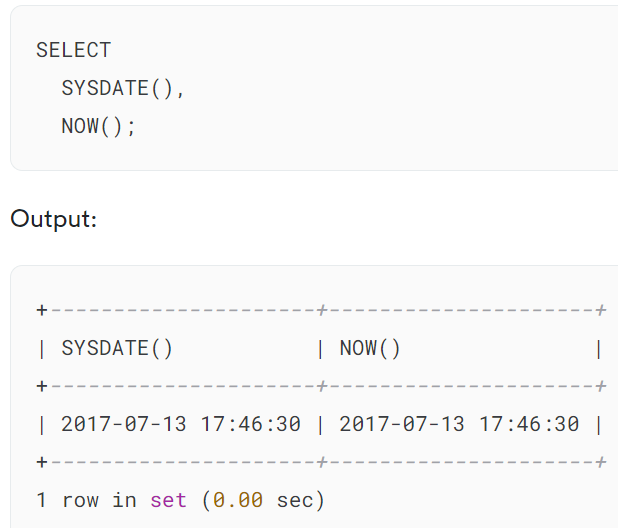


If you pass the fsp argument, the result will include the fractional seconds precision as shown in the following example:

## 

## **SYSDATE vs. NOW**

The following example uses the SYSDATE() and NOW() functions in the same query:



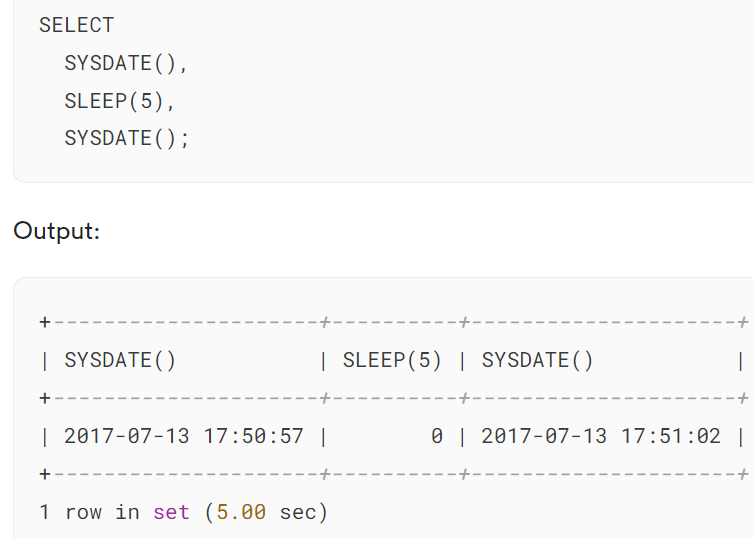
It seems that both SYSDATE() and [NOW()](https://www.mysqltutorial.org/mysql-date-functions/mysql-now-function/) functions return the same value which is the current date and time at which it is executed.

However, the SYSDATE() function actually returns the time at which it executes while the NOW() function returns a constant time at which the statement began to execute. For example:



In this example, we use the SLEEP() function to pause the query for 5 seconds. Within the same statement, the NOW() function always returns a constant which is the time at which the statement starts.

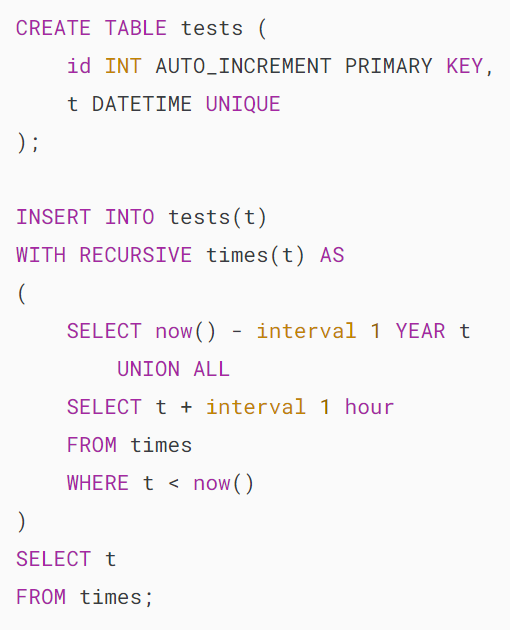
Let’s change the NOW() function to SYSDATE() function:



Within the same statement, SYSDATE() function returns different time values that reflect the time at which the SYSDATE() function was executed.

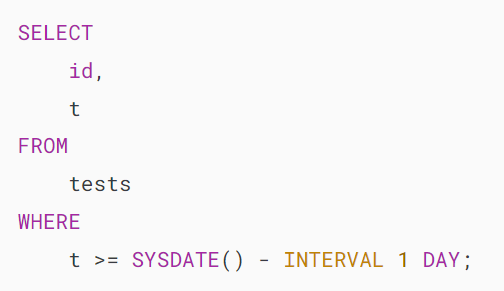
Because the SYSDATE() function is non-deterministic, [indexes](https://www.mysqltutorial.org/mysql-index/mysql-create-index/) cannot be utilized for evaluating expressions that refer to it.

To demonstrate this, we will [create a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named tests and [insert some data](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) into this table.

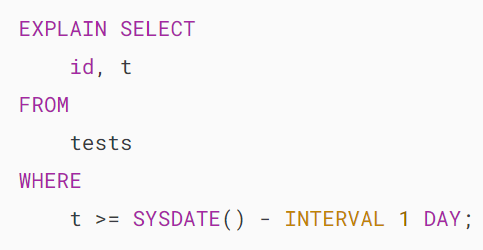


Notice that we used a [recursive CTE](https://www.mysqltutorial.org/mysql-basics/mysql-recursive-cte/) for generating time series. The CTE has been available since MySQL 8.0

Because the t column has a [unique index](https://www.mysqltutorial.org/mysql-unique/), the following query should execute fast:



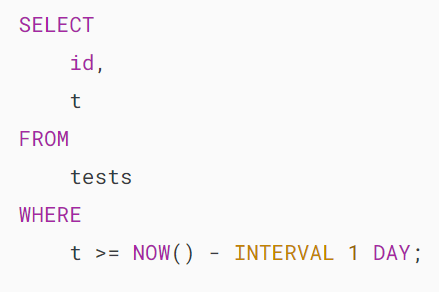
However, it took 15ms to complete. Let’s see the detail using the EXPLAIN statement.



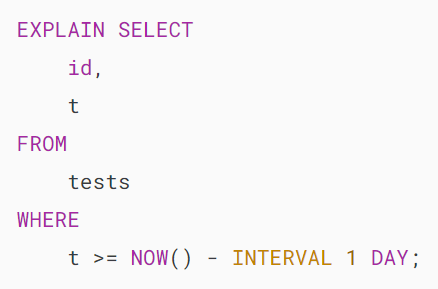
MySQL SYSDATE - SYSDATE function

It turned out that MySQL had to scan all the rows in the table to get the data. The index could not utilized.

If you change the SYSDATE() to NOW() function in the query:



With the NOW() function, the index has been used for querying data as demonstrated in the result of the EXPLAIN statement below:



MySQL SYSDATE - NOW function

Note that MySQL provides you with the --sysdate-is-now option that can make the SYSDATE() function behaves the same as the NOW() function.

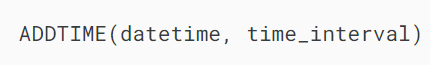
In this tutorial, you have learned about the MySQL SYSDATE() function and the reasons why you should consider using it.

# **MySQL ADDTIME() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL ADDTIME() function to add a time to another.

## **Introduction to MySQL ADDTIME() function**

The ADDTIME() function allows you to add a time interval to a [time](https://www.mysqltutorial.org/mysql-basics/mysql-time/) or [datetime](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) expression. Here’s the syntax of the ADDTIME() function:



In this syntax:

* datetime: The initial datetime or time value you want to add the time interval.
* time\_interval: The time interval you want to add, specified as a time expression in the format ‘HH:MM:SS‘.

In practice, you often use the ADDTIME() function to add hours, minutes, or seconds to a timestamp.

## **MySQL ADDTIME() function examples**

Let’s take some examples of using the ADDTIME() function.

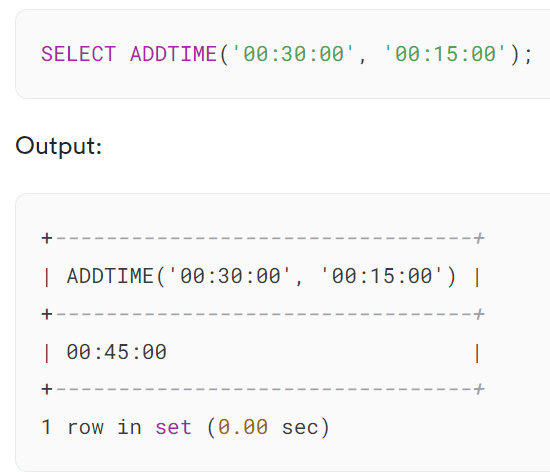
### **1) Basic MySQL ADDTIME() function example**

The following example uses the ADDTIME() function to add 3 hours, 30 minutes, and 45 seconds to the date time ‘2023-10-23 14:30:00’:

### 

### **2) Adding Minutes to a Time Value**

The following example uses the ADDTIME() function to add 15 minutes to a time value that represents the duration of a meeting:

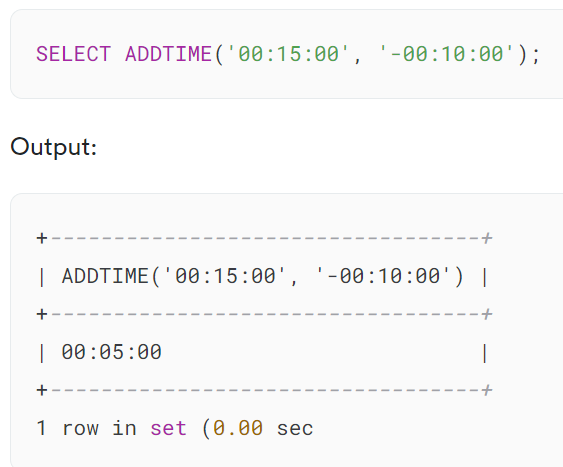


The result is ’00:45:00′, which represents the updated meeting duration.

### **3) Using the ADDTIME() function with negative time intervals**

The ADDTIME() function allows you to subtract a time by using negative time intervals.

For example, if you have a time value representing the duration of a break and you want to subtract 10 minutes from it:



The result is ’00:05:00′ which indicates the new duration after subtracting 10 minutes.

## **Summary**

* Use the ADDTIME() function to add a time interval to a time or datetime value.

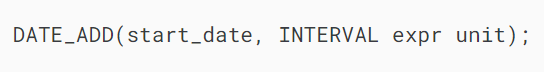
# **MySQL DATE\_ADD() Function**

**Summary**: in this tutorial, you will learn how to use MySQL DATE\_ADD() function to add a time value to a DATE or DATETIME value.

## **Introduction to MySQL DATE\_ADD function**

The DATE\_ADD function adds an [interval](https://www.mysqltutorial.org/mysql-basics/mysql-interval/) to a [DATE](https://www.mysqltutorial.org/mysql-basics/mysql-date/) or [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) value.

The following illustrates the syntax of the DATE\_ADD function:



The DATE\_ADD function takes two arguments:

* start\_date is a starting DATE or DATETIME value.
* INTERVAL expr unit is an interval value to be added to the starting date value.

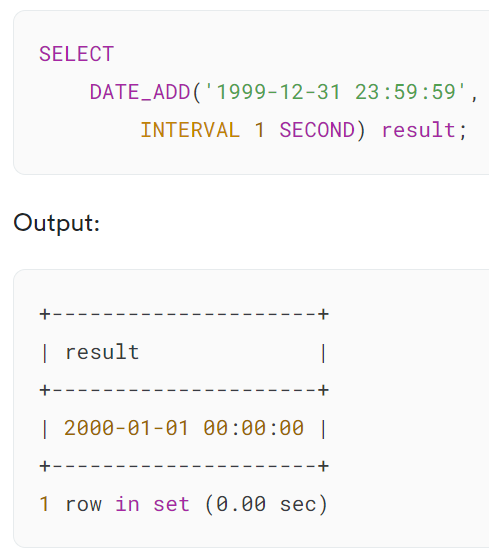
The DATE\_ADD() function may return a DATETIME value or a string, depending on the arguments:

* DATETIME if the first argument is a DATETIME value or if the interval value has a time element such as hour, minute, second, etc.
* String otherwise.

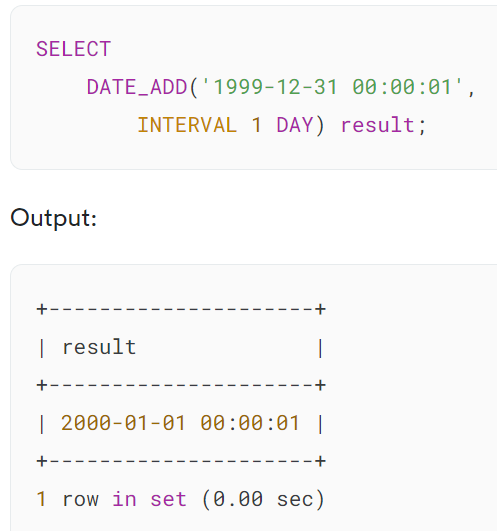
## **MySQL DATE\_ADD function examples**

Let’s take a look at some examples to understand how DATE\_ADD() function works.

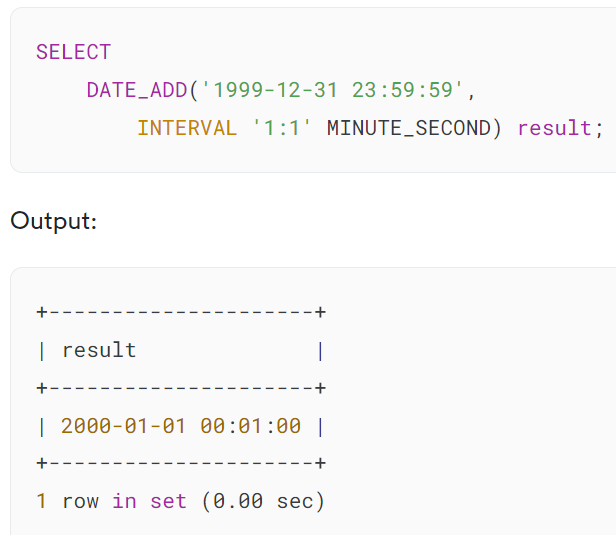
The following statement uses the DATE\_ADD() function to add one second to 1999-12-31 23:59:59:



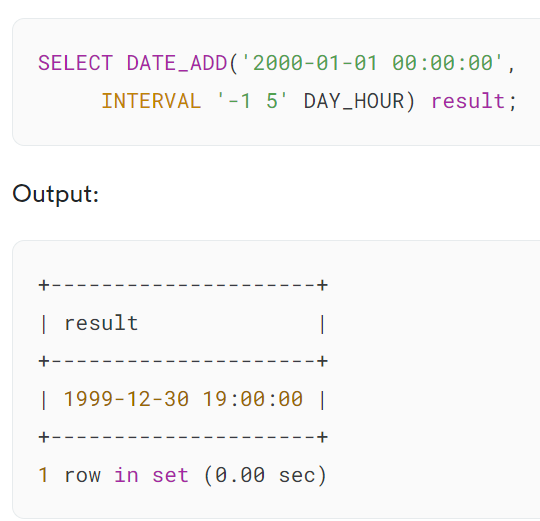
The following example uses the DATE\_ADD() function to add one day to 1999-12-31 00:00:01:



The following example uses the DATE\_ADD() function to add 1 minute and 1 second to 1999-12-31 23:59:59:



The following example adds -1 day and 5 hours to 2000-01-01 00:00:00.



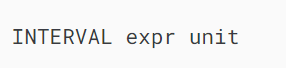
The following example adds 1 second and 999999 microseconds to 1999-12-31 23:59:59.000002:

## 

## **MySQL DATE\_ADD function usage notes**

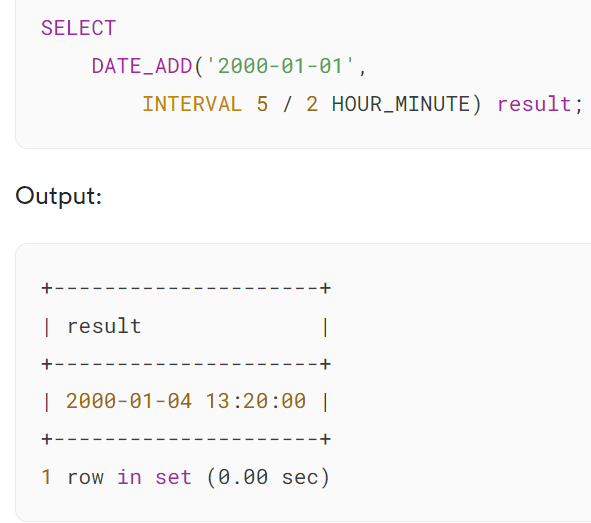
### **Interval Handling**

In the interval:



The expr is treated as a string, therefore, you should be careful when you use a non-string value for the expr.

For example, with an interval of HOUR\_MINUTE, 5/2 evaluates to 2.5000 (not 2.5) and is treated as 2 hours 5000 minutes as in the following statement:



To ensure the correct interpretation of a non-string interval value, you should use the [CAST](https://www.mysqltutorial.org/mysql-cast/) function as follows:

### 

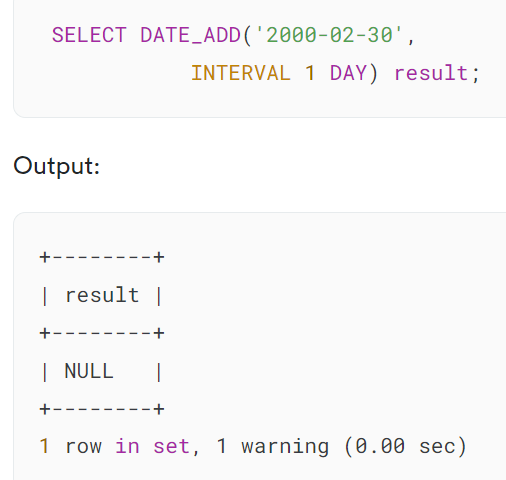
### **Automatic DATETIME conversion**

If you add a time value to a date value, the result is a [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) value as shown in the following example:



### **Invalid starting date**

The DATE\_ADD function returns [NULL](https://www.mysqltutorial.org/mysql-basics/mysql-null/) if you use an invalid date for the first argument, for example:



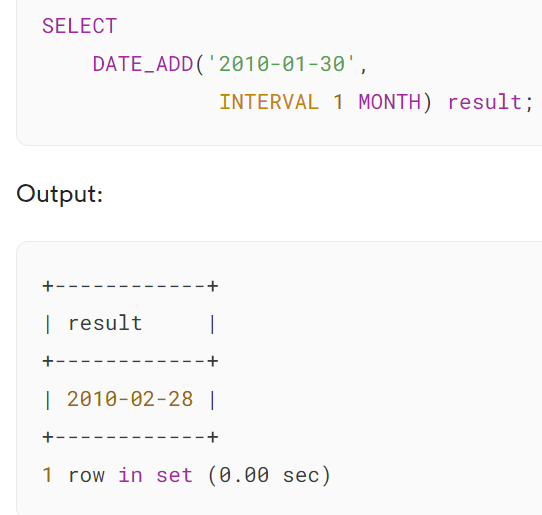
If you want to see the warning in detail, you use the SHOW WARNINGS statement:



### **Adjusted day, month, or year**

If you add an interval of MONTH, YEAR, or YEAR\_MONTH to a date that results in a date that has a day larger than the maximum day for the new month, the day will be adjusted to the maximum day in the new month.

Consider the following example:



In this example, we added 1 month to the January 30th 2010 that results in February 28th 2010. The day was adjusted to the maximum day inFebruary 2010.

In the year that February has 29 days, the date will be also adjusted to the 29th as shown below:

## 

## **Summary**

* Use the MySQL DATE\_ADD function to add an interval to a DATE or DATETIME value.

# **MySQL DATE\_SUB() Function**

**Summary**: in this tutorial, you will learn how to subtract a time from a date using the MySQL DATE\_SUB() function.

## **Introduction to MySQL DATE\_SUB() function**

The DATE\_SUB() function subtracts a time value (or an [interval](https://www.mysqltutorial.org/mysql-basics/mysql-interval/)) from a [DATE](https://www.mysqltutorial.org/mysql-basics/mysql-date/) or [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) value.

Here’s the syntax of the DATE\_SUB() function:

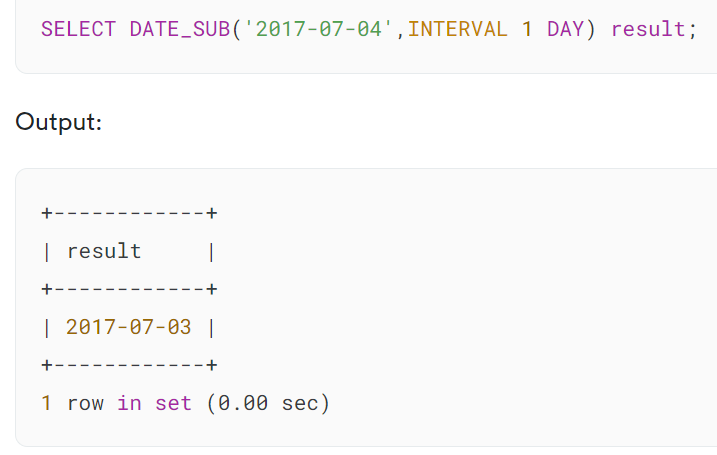


The DATE\_SUB() function accepts two arguments:

* date: This is the date that you want to subtract a value.
* expr: This is a string that determines an interval value that you want to subtract from the date. The unit is the interval unit that expr should be interpreted e.g., DAY, HOUR, etc.

The DATE\_SUB() function returns NULL if the date is NULL.

The following statement uses the DATE\_SUB() function to subtract one day from the July-4th-2017:



In this example:

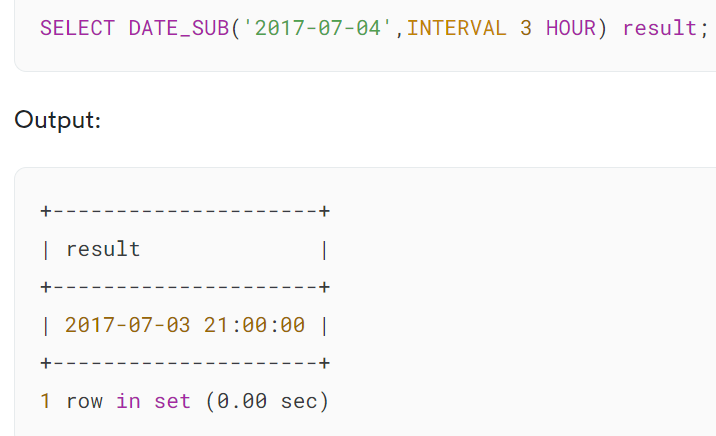
* The date is 2017-07-04, which is in the yyyy-mm-dd format.
* The INTERVAL 1 DAY is interpreted as 1 day interval.

The DATE\_SUB() function returns a string value that represents the date July, 3rd 2017

Similar to the [DATE\_ADD()](https://www.mysqltutorial.org/mysql-date-functions/mysql-date_add/) function, the data type of the return value of the DATE\_SUB() function can be:

* a DATETIME value if the first argument is a DATETIME or the interval has time elements such as the hour, minute, second, etc.
* a string otherwise.

See the following example:



Because the interval is 3 hours, the result of the DATE\_SUB function is a DATETIME value.

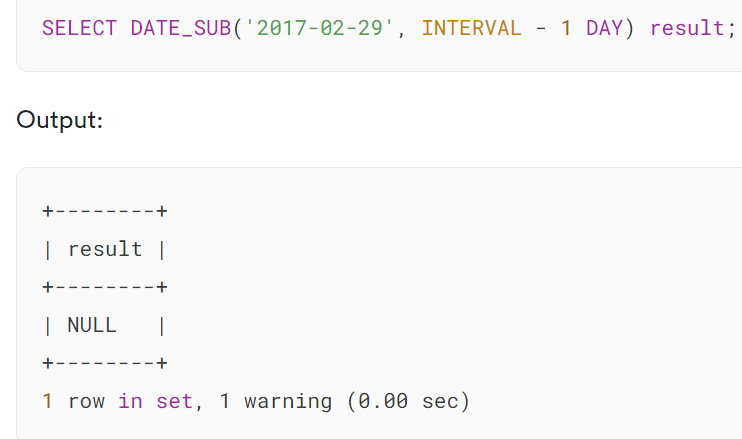
## **Negative interval**

The expr in the interval can be positive or negative. In case the expr is negative, the DATE\_SUB() function behaves like the DATE\_ADD() function as shown in the following example:

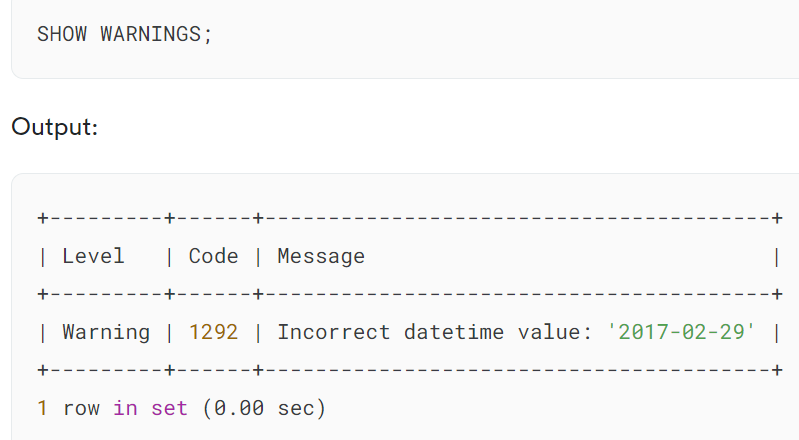
## 

## **Invalid or malformed date**

If the first argument of the DATE\_SUB() function is a malformed, invalid date, or NULL, the DATE\_SUB() function returns NULL.

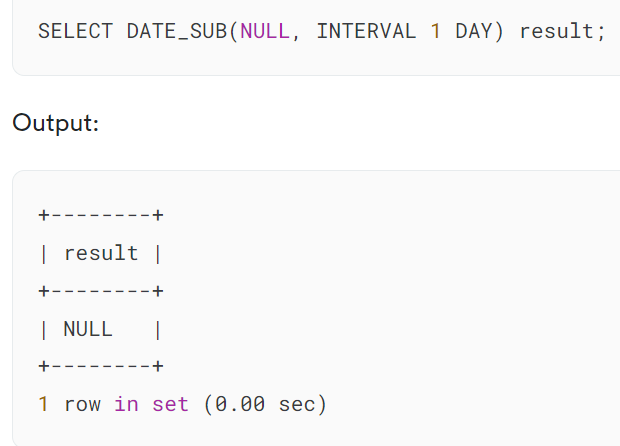


In this example, 2017-02-03 is an invalid date, therefore, the result is NULL. In addition, MySQL produced a warning.



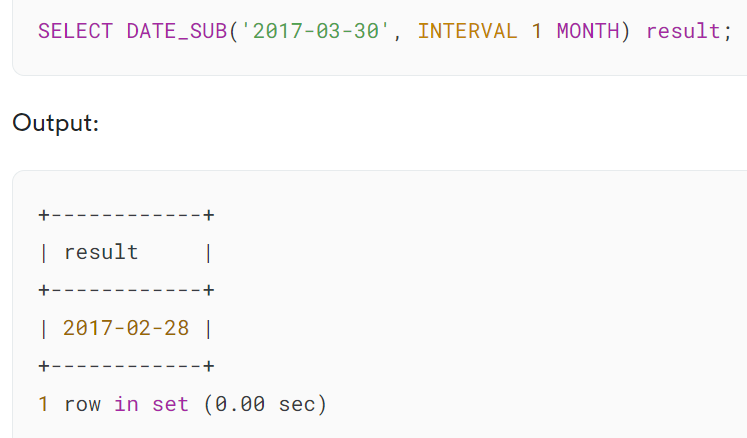
The following examples demonstrate the effects when passing a malformed date or NULL to the DATE\_SUB function:

## 



## **MySQL DATE\_SUB: automatically adjusted day**

If you subtract MONTH, YEAR, or YEAR\_MONTH from a date and the new date has a day that’s too big for the new month, the DATE\_SUB() will adjust the day to the latest day of the new month. For example:



In this example, we subtracted 1 month from March 30th 2017, therefore, the result is February 28th 2017.

However, the DATE\_SUB() function adjusted the day to 28 instead of 30 because February 2017 has 28 days only.

## **Summary**

* Use the MySQL DATE\_SUB() function to subtract an interval from a DATE or DATETIME value.

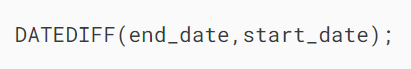
# **MySQL DATEDIFF() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL DATEDIFF()function to calculate the number of days between two DATE values.

## **Introduction to MySQL DATEDIFF() function**

The DATEDIFF() function calculates the difference in days between two dates.

Here’s the basic syntax of the DATEDIFF() function:



In this syntax:

* end\_date: The date to which you want to calculate the difference.
* start\_date: The date from which you want to calculate the difference.

The DATEDIFF() function returns an integer that represents the number of days between two dates.

If end\_date or start\_date is NULL, the DATEDIFF() function returns NULL.

Notice that the DATEDIFF() function considers only the date components for calculation and disregards the time components.

## **MySQL DATEDIFF examples**

Let’s take some examples of using the DATEDIFF() function

### **1) Simple DATEDIFF() function example**

The following example uses the DATEDIFF() function to calculate the difference between two DATE literal values:

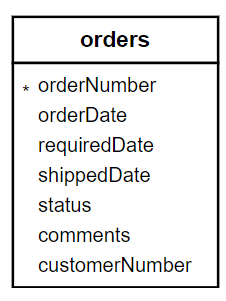


The following example uses the DATEDIFF() function to calculate the number of days between 2011-08-17 and 2011-08-08:

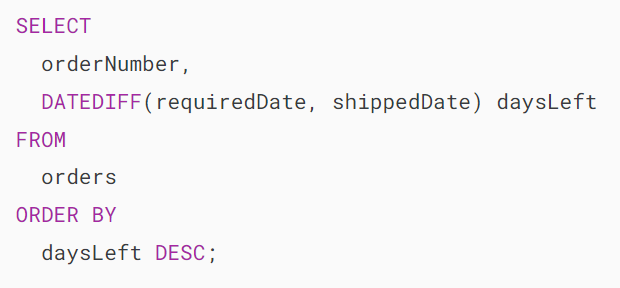
### 

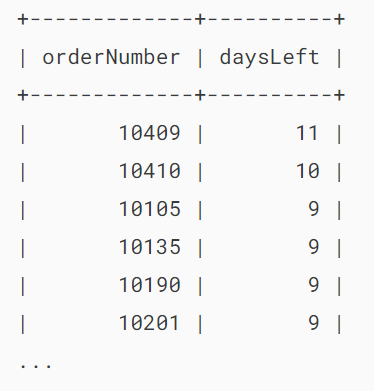
### **2) Using DATEDIFF() function with table data**

We’ll use the orders table in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):

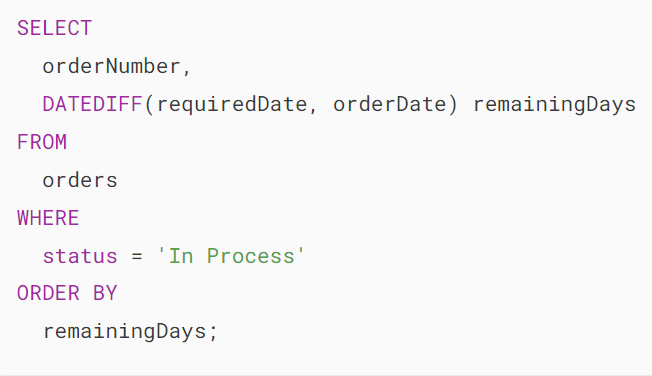


The following example uses the DATEDIFF() function to calculate the number of days between the required date and the shipped date of each order::

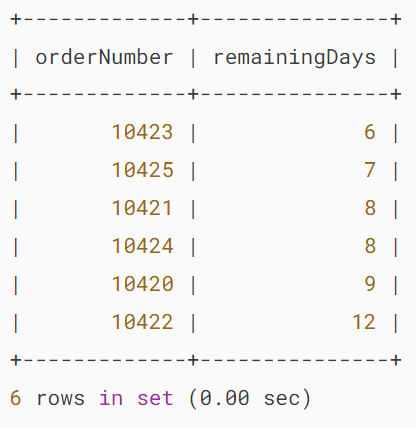


...

The following statement gets all orders whose statuses are in process and calculates the number of days between the ordered date and the required date:

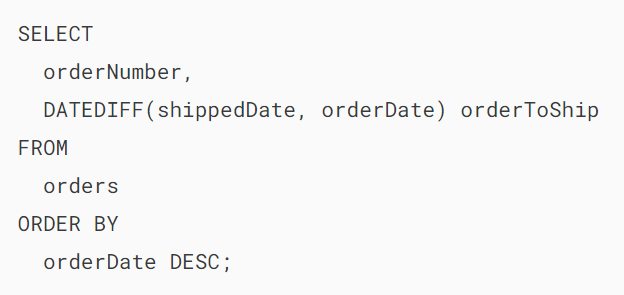


Output:

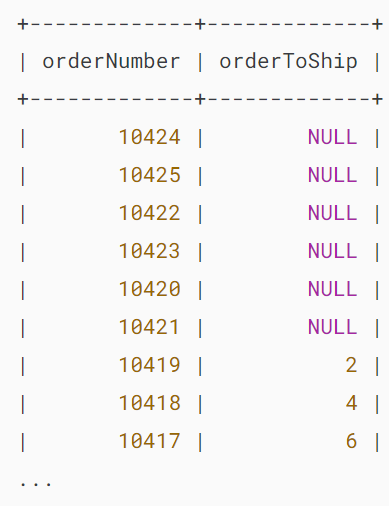


### **3) Dealing with NULL values**

The following example uses the DATEDIFF() function to calculate the days it takes from the order date to the shipped date:



Output:



The following example uses the IFNULL() function to display N/A when the shipped date is NULL



Output:

## 

## **Summary**

* Use MySQL DATEDIFF() function to calculate the number of days between two date values.

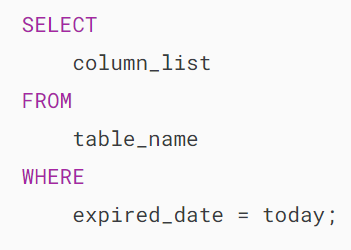
# **How to Get MySQL Today’s Date**



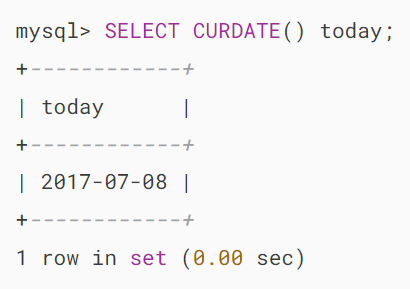
**Summary**: in this tutorial, you will learn how to query data that matches **MySQL today**‘s date by using built-in date functions.

## **Getting MySQL today’s date using built-in date functions**

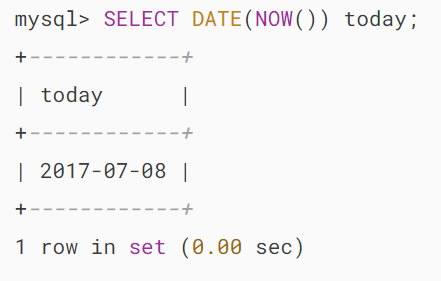
Sometimes, you may want to query data from a table to get rows with the date column is today, for example:



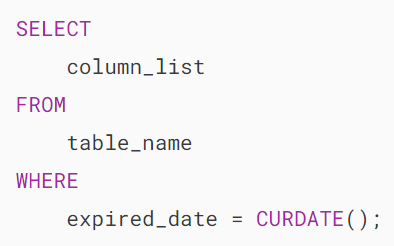
To get today’s date, you use the CURDATE() function as follows:



Alternatively, you can get the date part from the current time returned by the [NOW()](https://www.mysqltutorial.org/mysql-date-functions/mysql-now-function/) function:



So the query should change to:

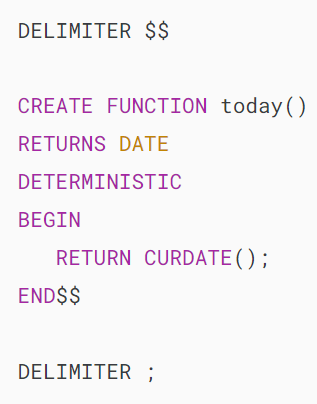


If the expired\_date column contains both the date and time part, you should use the DATE() function to extract only the date part and compare it with the current date:

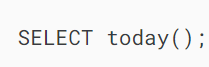
## 

## **Creating your MySQL today stored function**

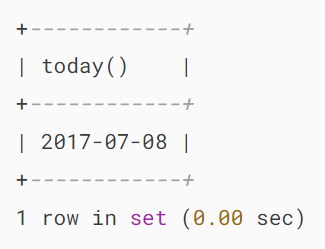
If you use frequently the [CURDATE()](https://www.mysqltutorial.org/mysql-date-functions/mysql-curdate/) function in your queries and want to replace them with the today() function for enhanced readability, you can create your own [stored function](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-stored-function/) named today() as follows:



Now, you can use the today() function as follows:



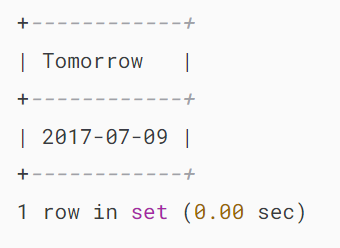
Output:



How about tomorrow? It should be as simple as the following query:



Output:



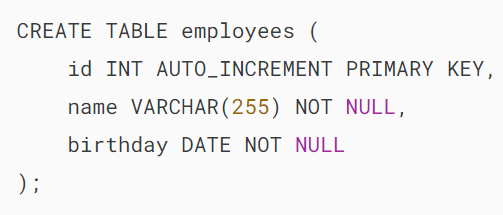
And also yesterday is easy as well:

## 

## **Getting employees whose birthday is today**

Sometimes, you want to retrieve employees whose birthday is today. Let’s see an example.

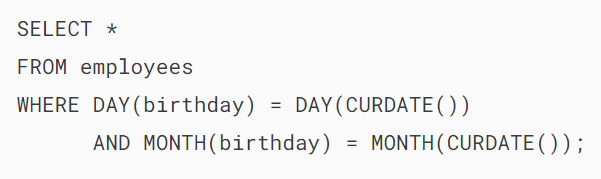
First, create a table called employees:



Second, insert some rows into the employees table:



Third, find the employees whose birthday is today:



The query retrieves employees whose month and day of birthday are the same as the month and day of today.

## **Summary**

* Use the CURDATE() or NOW() function to return the today’s date.

<https://www.mysqltutorial.org/mysql-date-functions/mysql-timediff/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-timestampdiff/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-timestampadd/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-timestampdiff/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-time_to_sec/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-to_days/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-convert_tz/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-from_days/>

# **MySQL STR\_TO\_DATE() Function**

**Summary**: in this tutorial, we will show you how to use the **MySQL STR\_TO\_DATE()**function to convert a string into a date, time, or datetime value.

## **Introduction to MySQL STR\_TO\_DATE function**

The STR\_TO\_DATE() converts a string into a date value based on a specified format string.

Here’s the syntax of the STR\_TO\_DATE() function:



In this syntax:

* str: This is the input string that you want to convert.
* fmt: This is the format string that includes format specifiers. For example, %d for day, %m for month and %y for year.

The STR\_TO\_DATE() function may return a DATE , TIME, or DATETIME value based on the input and format strings.

If the input string is illegal, the STR\_TO\_DATE() function returns NULL.

The STR\_TO\_DATE() function scans the input string to match the format string. The format string may contain literal characters and format specifiers that begin with a percentage (%) character.

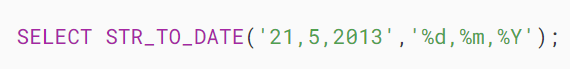
Refer to the [DATE\_FORMAT function](https://www.mysqltutorial.org/mysql-date-functions/mysql-date_format/)for the list of format specifiers.

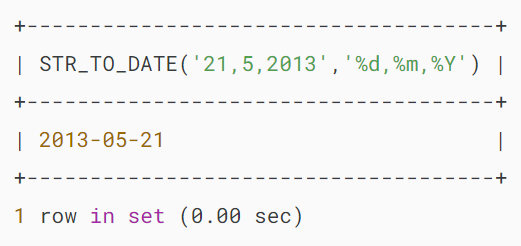
The STR\_TO\_DATE() function is very useful in data migration that involves temporal data conversion from an external format to MySQL temporal data format.

## **MySQL STR\_TO\_DATE examples**

Let’s look at some examples of using STR\_TO\_DATE() function to convert strings into date and/or time values

The following statement converts a string into a DATE value.

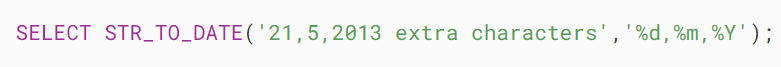


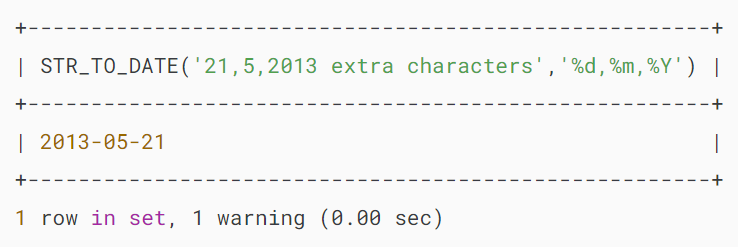


Based on the format string ‘%d, %m, %Y’, the STR\_TO\_DATE() function scans the ‘21,5,2013’ input string.

* First, it attempts to find a match for the %d format specifier, which is a day of the month (01…31), in the input string. Because the number 21 matches with the %d specifier, the function takes 21 as the day value.
* Second, because the comma (,) literal character in the format string matches with the comma in the input string, the function continues to check the second format specifier %m , which is a month (01…12), and finds that the number 5 matches with the %m format specifier. It takes the number 5 as the month value.
* Third, after matching the second comma (,), the STR\_TO\_DATE() function keeps finding a match for the third format specifier %Y , which is a four-digit year e.g., 2012,2013, etc., and it takes the number 2013 as the year value.

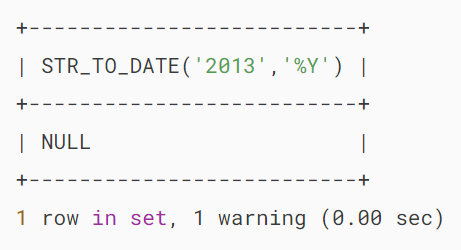
The STR\_TO\_DATE() function ignores extra characters at the end of the input string when it parses the input string based on the format string. See the following example:





The STR\_TO\_DATE() sets all incomplete date values, which are not provided by the input string, to NULL. See the following example:

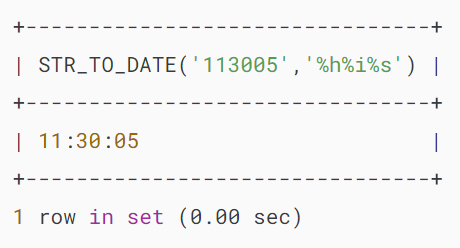




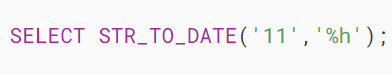
Because the input string only provides a year value, the STR\_TO\_DATE() function returns a date value that has month and day set to NULL. Notice that in MySQL 5.7 or earlier, the STR\_TO\_DATE() sets month and day to zero.

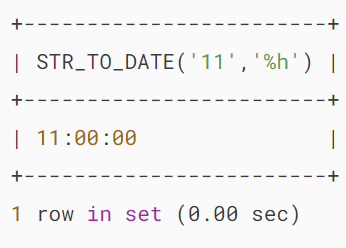
The following example converts a time string into a TIME value:





Similar to the unspecified date part, the STR\_TO\_DATE() function sets unspecified time part to zero, see the following example:





The following example converts the string into a DATETIME value because the input string provides both date and time parts.



## 

## **Summary**

* Use the MySQL STR\_TO\_DATE() function to convert strings to date and time values.

<https://www.mysqltutorial.org/mysql-date-functions/mysql-from_unixtime/>

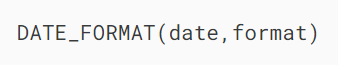
<https://www.mysqltutorial.org/mysql-date-functions/mysql-unix_timestamp/>

# **MySQL DATE\_FORMAT() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL DATE\_FORMAT()function to format a date value based on a specific format.

## **Introduction to MySQL DATE\_FORMAT function**

To format a [date](https://www.mysqltutorial.org/mysql-basics/mysql-date/) value to a specific format, you use the DATE\_FORMAT() function. The syntax of the DATE\_FORMAT function is as follows:



The DATE\_FORMAT() function accepts two arguments:

* date : is a valid date value that you want to format
* format : is a format string that consists of predefined specifiers. Each specifier is preceded by a percentage character ( % ). See the table below for a list of predefined specifiers.

The DATE\_FORMAT function returns a string whose [character set](https://www.mysqltutorial.org/mysql-basics/mysql-character-set/) and [collation](https://www.mysqltutorial.org/mysql-basics/mysql-collation/) depend on the settings of the client’s connection.

The following table illustrates the specifiers and their meanings that you can use to construct a date format string:

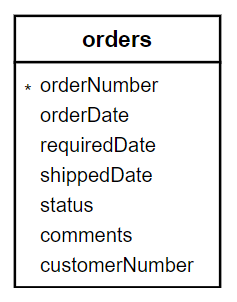
| **Specifier** | **Meaning** |
| --- | --- |
| %a | Three-characters abbreviated weekday name e.g., Mon, Tue, Wed, etc. |
| %b | Three-characters abbreviated month name e.g., Jan, Feb, Mar, etc. |
| %c | Month in numeric e.g., 1, 2, 3…12 |
| %D | Week number with leading zero when the first day of the week is Sunday e.g., 00,01,02…53 |
| %d | Day of the month with leading zero if it is 1 number e.g., 00, 01,02, …31 |
| %e | Day of the month without leading zero e.g., 1,2,…31 |
| %f | Microseconds in the range of 000000..999999 |
| %H | Hour in 24-hour format with leading zero e.g., 00..23 |
| %h | Hour in 12-hour format with leading zero e.g., 01, 02…12 |
| %I | Same as %h |
| %i | Minutes with leading zero e.g., 00, 01,…59 |
| %j | Day of year with leading zero e.g., 001,002,…366 |
| %k | Hour in 24-hour format without leading zero e.g., 0,1,2…23 |
| %l | Hour in 12-hour format without leading zero e.g., 1,2…12 |
| %M | Full month name e.g., January, February,…December |
| %m | Month name with leading zero e.g., 00,01,02,…12 |
| %p | AM or PM, depending on other time specifiers |
| %r | Time in 12-hour format hh:mm:ss AM or PM |
| %S | Seconds with leading zero 00,01,…59 |
| %s | Same as %S |
| %T | Time in 24-hour format hh:mm:ss |
| %U | Weekday in number (0=Sunday, 1= Monday, etc.) |
| %u | Week number with leading zero when the first day of the week is Monday e.g., 00,01,02…53 |
| %V | Same as %U; it is used with %X |
| %v | Same as %u; it is used with %x |
| %W | Full name of weekday e.g., Sunday, Monday,…, Saturday |
| %w | Two digits year e.g., 10,11, and 12. |
| %X | Year for the week in four digits where the first day of the week is Sunday; often used with %V |
| %x | Year for the week, where the first day of the week is Monday, four digits; used with %v |
| %Y | Four digits year e.g., 2000 and 2001. |
| %y | Add a percentage (%) character to the output |
| %% | Add percentage (%) character to the output |

The following are some commonly used date format strings:

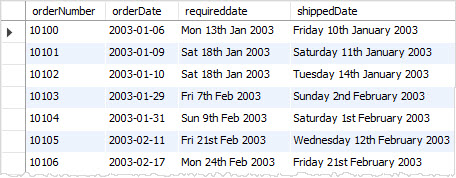
| **DATE\_FORMAT string** | **Formatted date** |
| --- | --- |
| %Y-%m-%d | 2013-07-04 |
| %e/%c/%Y | 4/7/2013 |
| %c/%e/%Y | 7/4/2013 |
| %d/%m/%Y | 4/7/2013 |
| %m/%d/%Y | 7/4/2013 |
| %e/%c/%Y %H:%i | 4/7/2013 11:20 |
| %c/%e/%Y %H:%i | 7/4/2013 11:20 |
| %d/%m/%Y %H:%i | 4/7/2013 11:20 |
| %m/%d/%Y %H:%i | 7/4/2013 11:20 |
| %e/%c/%Y %T | 4/7/2013 11:20 |
| %c/%e/%Y %T | 7/4/2013 11:20 |
| %d/%m/%Y %T | 4/7/2013 11:20 |
| %m/%d/%Y %T | 7/4/2013 11:20 |
| %a %D %b %Y | Thu 4th Jul 2013 |
| %a %D %b %Y %H:%i | Thu 4th Jul 2013 11:20 |
| %a %D %b %Y %T | Thu 4th Jul 2013 11:20:05 |
| %a %b %e %Y | Thu Jul 4 2013 |
| %a %b %e %Y %H:%i | Thu Jul 4 2013 11:20 |
| %a %b %e %Y %T | Thu Jul 4 2013 11:20:05 |
| %W %D %M %Y | Thursday 4th July 2013 |
| %W %D %M %Y %H:%i | Thursday 4th July 2013 11:20 |
| %W %D %M %Y %T | Thursday 4th July 2013 11:20:05 |
| %l:%i %p %b %e, %Y | 7/4/2013 11:20 |
| %M %e, %Y | 4-Jul-13 |
| %a, %d %b %Y %T | Thu, 04 Jul 2013 11:20:05 |

## **MySQL DATE\_FORMAT examples**

Let’s take a look at the orders table in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).



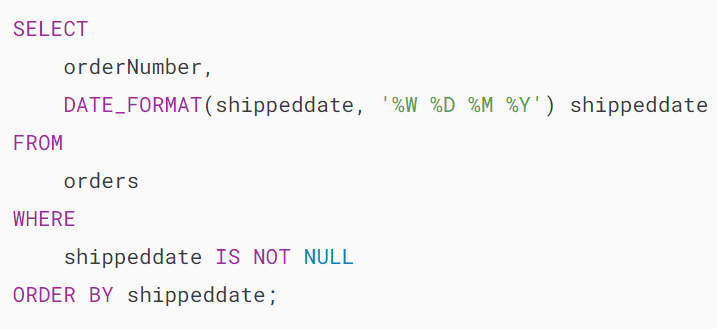
To select the order’s data and format the date value, you use the following statement:



We formatted the order date, required date, and shipped date of each order based on different date formats specified by the format strings.

### **MySQL DATE\_FORMAT with ORDER BY**

See the following example:

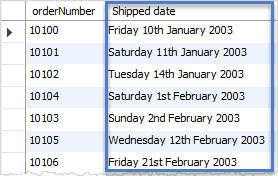
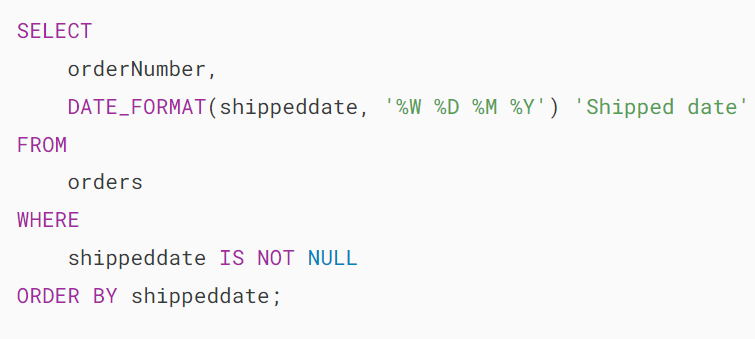




In the query, we selected all orders whose shipped dates were not NULL and sorted the orders by the shipped date. However, the orders were not sorted correctly.

The reason is that we used shippeddate as the [alias](https://www.mysqltutorial.org/mysql-basics/mysql-alias/)for the output of the DATE\_FORMAT function, which is a string, the [ORDER BY](https://www.mysqltutorial.org/mysql-order-by) clause took the alias and sorted the orders based on string values, not date values.

To fix this problem, we have to use an alias that is different from the column name; see the following statement:



<https://www.mysqltutorial.org/mysql-date-functions/mysql-time_format-function/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-get_format/>

# **MySQL DATE() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL DATE() function to extract the date part of a datetime or timestamp value.

## **Introduction to MySQL DATE() function**

The DATE() function allows you to extract the date component from a [datetime](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) or [timestamp](https://www.mysqltutorial.org/mysql-basics/understanding-mysql-timestamp/) expression.

Here’s the syntax of the DATE() function:



In this syntax:

* expression: This is an expression that evaluates to a DATE, a DATETIME, or a TIMESTAMP value from which you want to extract the value.

The DATE() function returns the DATE value. It returns NULL if the expression is NULL.

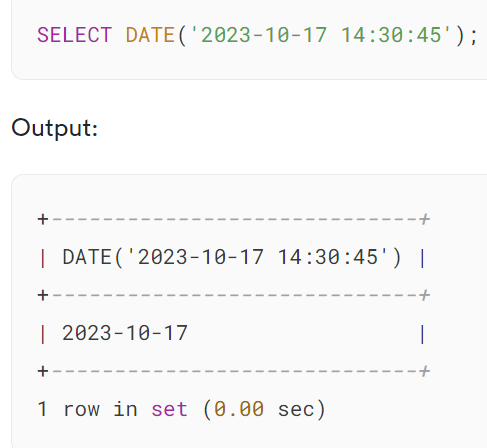
The DATE() function is a convenient tool for working with DATETIME and TIMESTAMP values in the database. It allows you to extract, filter, group, and perform date arithmetic on date expressions. By using the DATE() function in your queries, you can manage and analyze date-related data more effectively.

## **MySQL DATE() function examples**

Let’s take some examples of using the MySQL DATE() function.

### **1) Extracting a date from a DATETIME value**

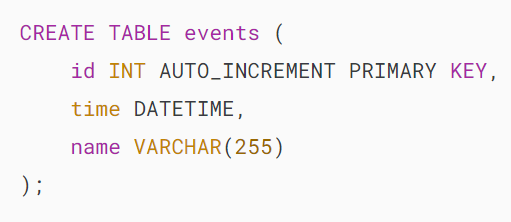
The following example uses the DATE() function to extract the date from a DATETIME literal value:



In this example, the DATE() function returns a DATE value ('2023-10-17') from the DATETIME value '2023-10-17 14:30:45'.

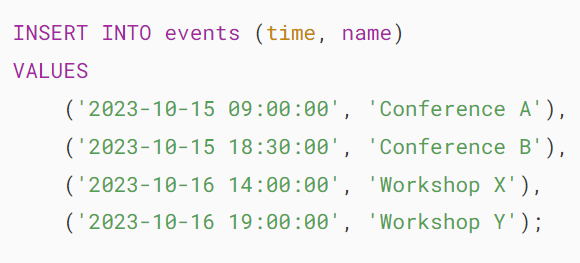
### **2) Using the DATE() function with table data**

First, create a new table called events with the following structure:

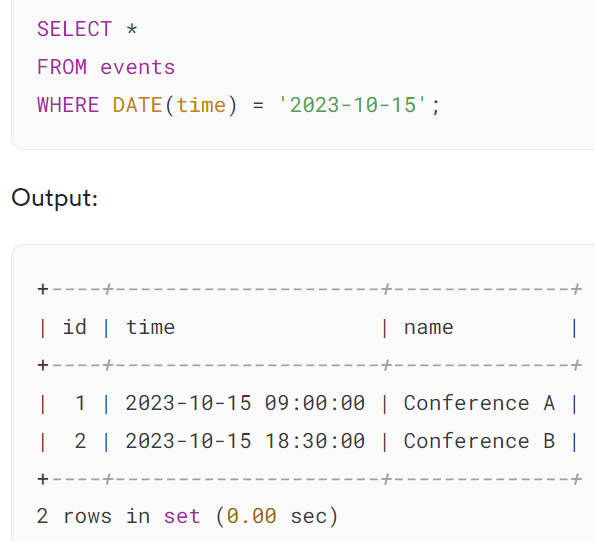


The events table has three columns id, time, and name. The time column has the DATETIME type.

Second, insert four rows into the events table:



Third, query data from the events table:



In this example, we use the DATE() function to extract the date part of the time column and use its result to filter the events based on a specific date.

## **Summary**

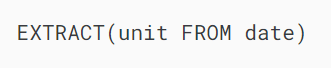
* Use the DATE() function to extract the date part of a datetime or timestamp value.

# **MySQL EXTRACT() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL EXTRACT() function to extract part of a [DATE](https://www.mysqltutorial.org/mysql-basics/mysql-date/) or [DATETIME](https://www.mysqltutorial.org/mysql-basics/mysql-datetime/) value.

## **Introduction to the MySQL EXTRACT() function**

The EXTRACT() function extracts part of a date. The following illustrates the syntax of the EXTRACT() function.



The EXTRACT() function requires two arguments unit and date.

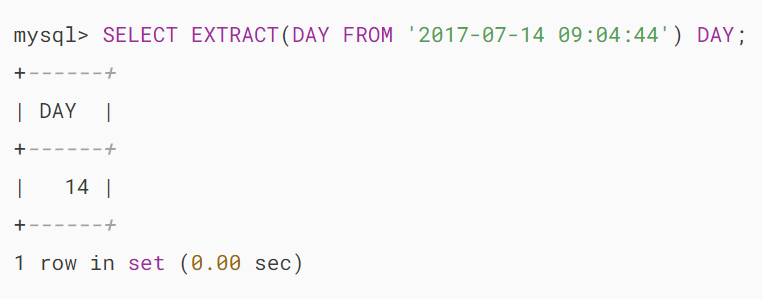
The unit is the [interval](https://www.mysqltutorial.org/mysql-basics/mysql-interval/) that you want to extract from the date. The following are the valid intervals for the unit argument.

* DAY
* DAY\_HOUR
* DAY\_MICROSECOND
* DAY\_MINUTE
* DAY\_SECOND
* HOUR
* HOUR\_MICROSECOND
* HOUR\_MINUTE
* HOUR\_SECOND
* MICROSECOND
* MINUTE
* MINUTE\_MICROSECOND
* MINUTE\_SECOND
* MONTH
* QUARTER
* SECOND
* SECOND\_MICROSECOND
* WEEK
* YEAR
* YEAR\_MONTH

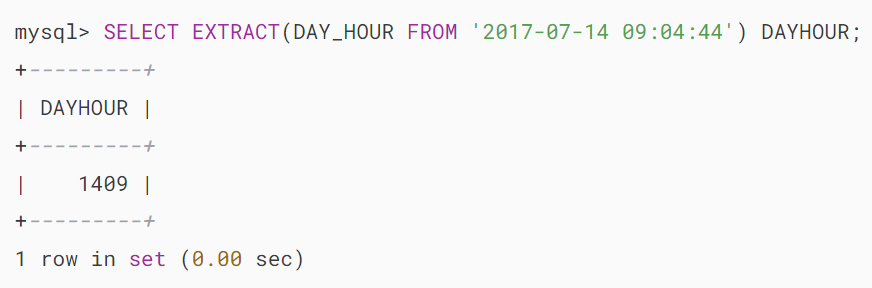
The date is a DATE or DATETIME value from which you extract an interval.

## **MySQL EXTRACT() function examples**

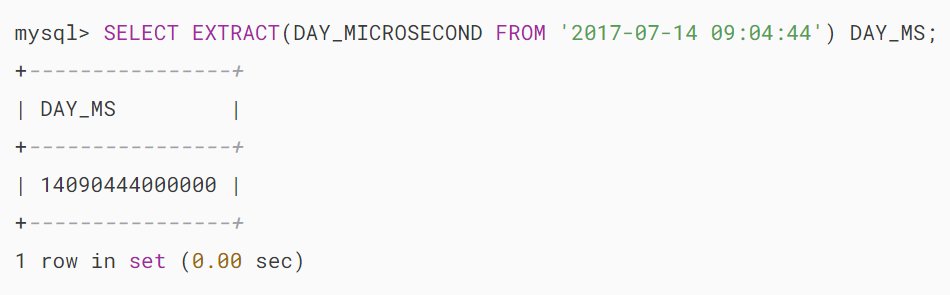
Extract day from a datetime:



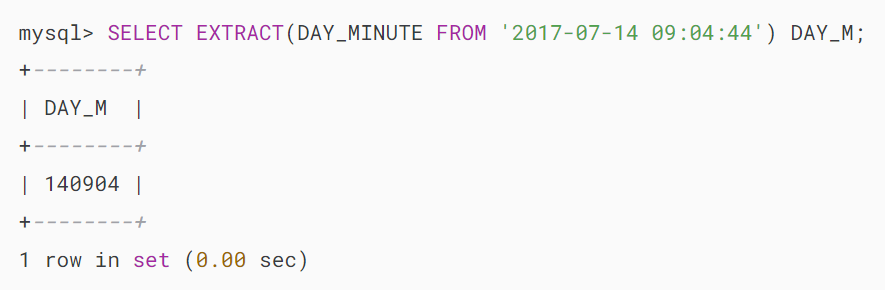
Extract day\_hour from a datetime:



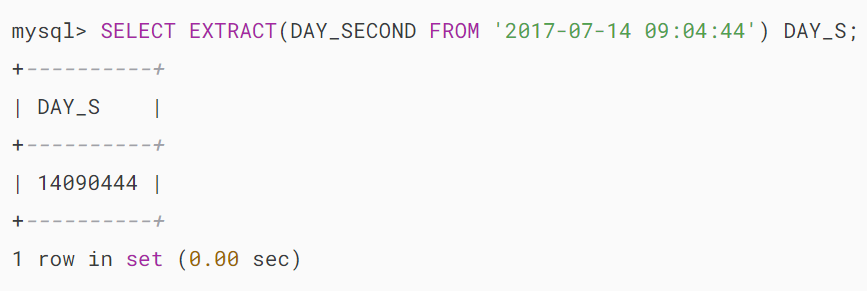
Extract day\_microsecond from a datetime:



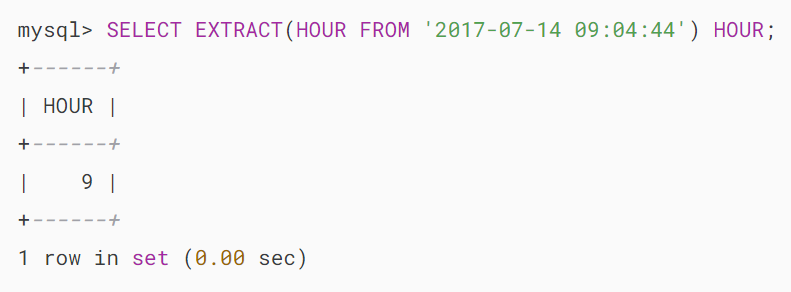
Extract day\_minute from a datetime:



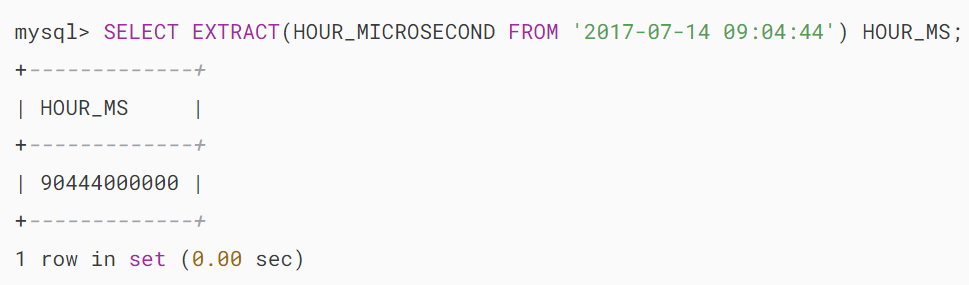
Extract day\_second from a datetime



Extract hour from a datetime:



Extract hour\_microsecond from a datetime:



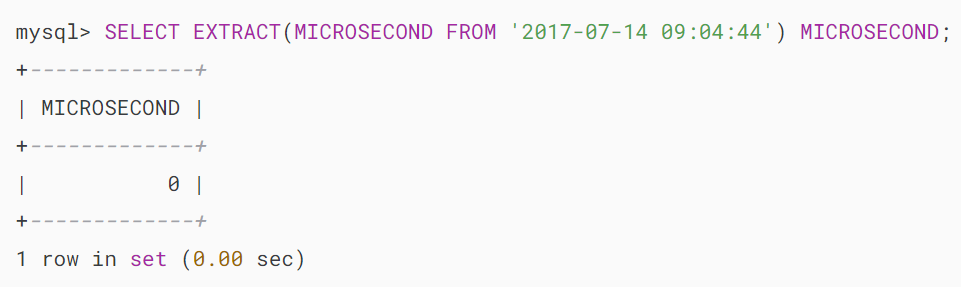
Extract hour\_minute from a datetime:



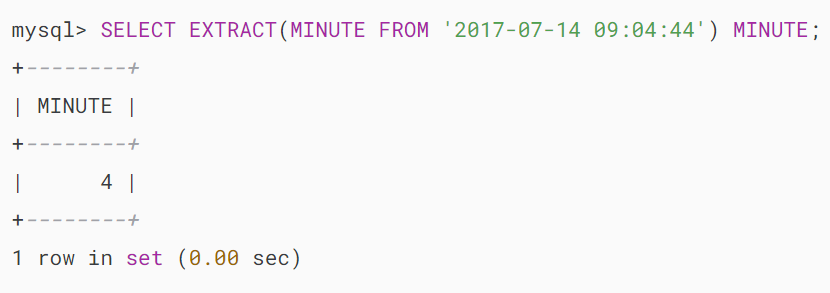
Extract hour\_second from a datetime:



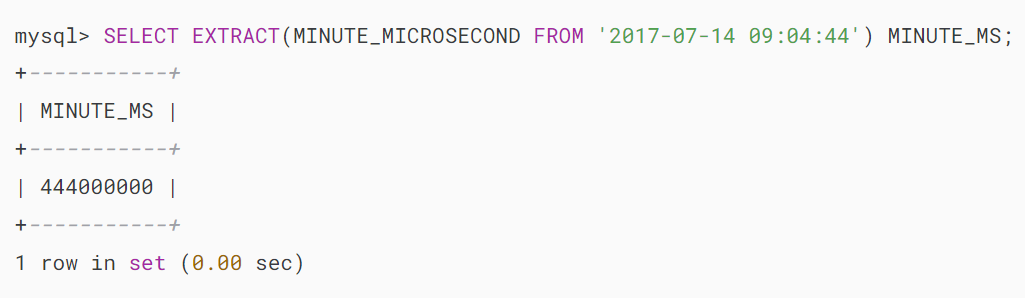
Extract microsecond from a datetime:



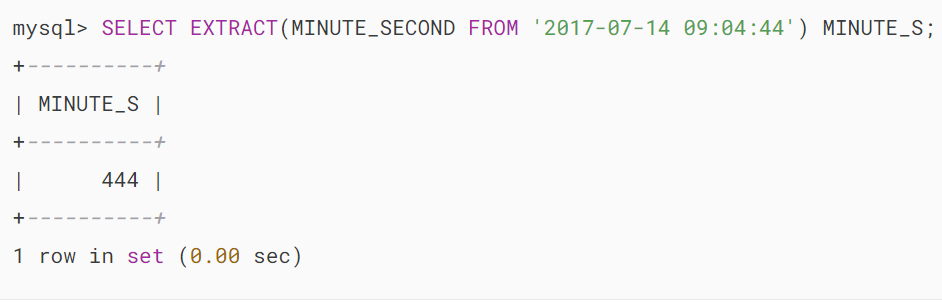
Extract minute from a datetime:



Extract minute\_microsecond from a datetime:



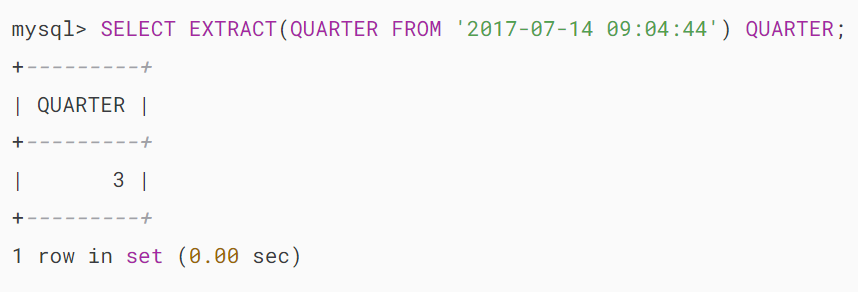
Extract minute\_second from a datetime:



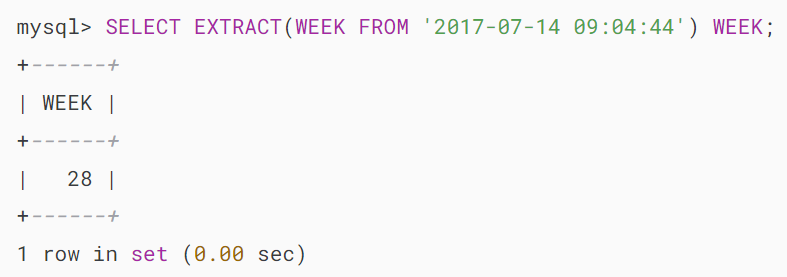
Extract month from a datetime:



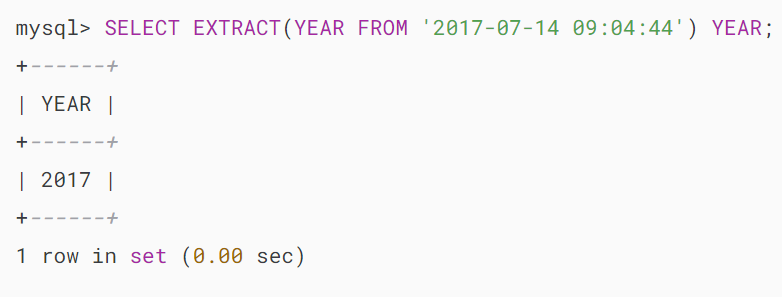
Extract quarter from a datetime:



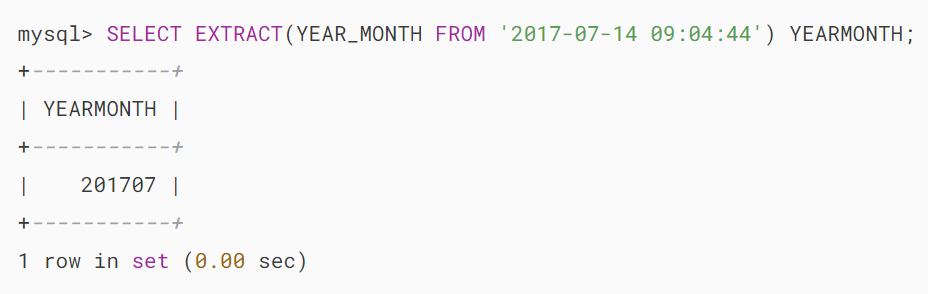
Extract week from a datetime:



Extract year from a datetime:



Extract year\_month from a datetime

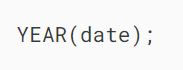


# **MySQL YEAR() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL YEAR() function to get the year out of a date value.

## **Introduction to MySQL YEAR() function**

The YEAR() function returns a year from a [date](https://www.mysqltutorial.org/mysql-basics/mysql-date/) value. Here’s the basic syntax of the YEAR() function:



In this syntax:

* date: This is the date or datetime value from which you want to get the year.

The YEAR() function returns an integer that represents the year part of the provided date. It has a range of 1000 and 9999.

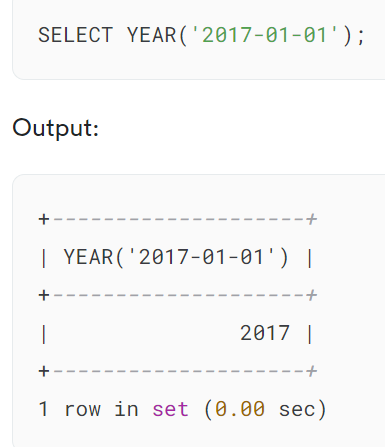
If the date is zero, the YEAR() function returns 0. If the date is NULL, the YEAR() function returns NULL.

## **MySQL YEAR() function examples**

Let’s take some examples of using the YEAR() function.

### **1) Simple YEAR() function examples**

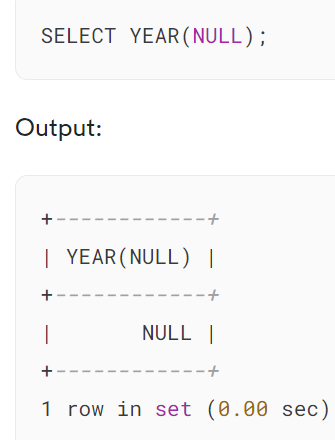
The following example uses the YEAR() function to extract the year of January 1st 2017:



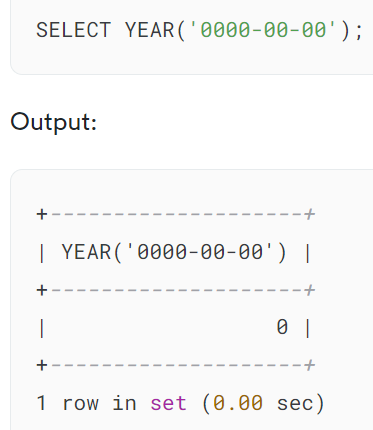
To get the current year, you use the NOW() function to get the current date and time and pass it to the YEAR() function as follows:



If the date is NULL, the YEAR() function will return NULL as shown in the following example:

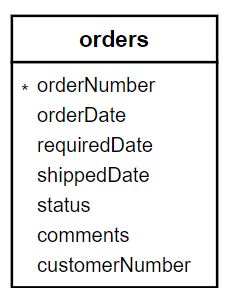


As mentioned earlier, the YEAR() of a zero date is zero:

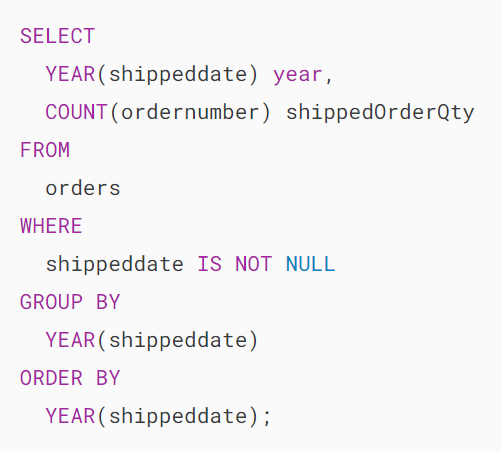


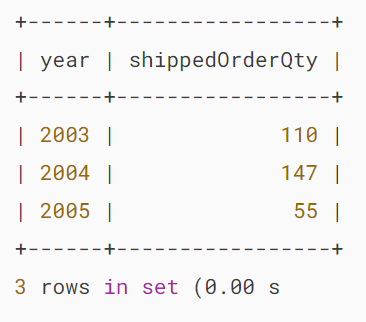
### **2) Using the YEAR() function with table data**

We’ll use the orders table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):



The following query uses the YEAR() function to retrieve the number of orders shipped per year





The query does the following:

* First, select data from the orders table.
* Second, filter out orders that haven’t been shipped.
* Third, group the remaining rows by the year in which they were shipped and count the number of orders for each year.
* Finally, sort the result set by year in ascending order

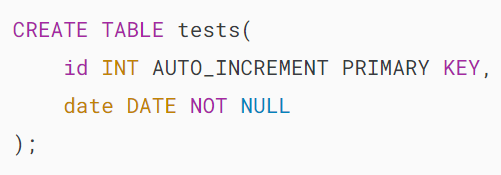
It provides useful insights into the trend of order shipments over the years.

## **MySQL YEAR function and indexes**

When you use the YEAR() function in a query, MySQL may not use an [index](https://www.mysqltutorial.org/mysql-index/). This can lead to a decrease in the speed of data retrieval. It’ll be fine if the table has a few rows.

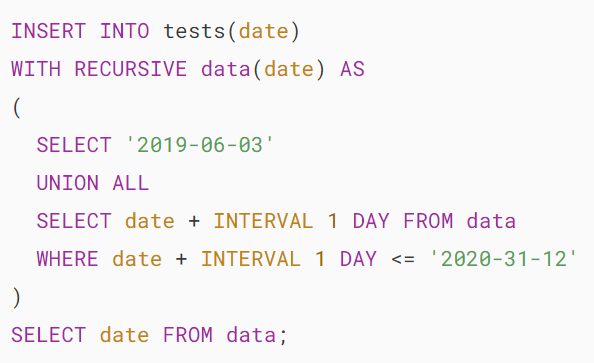
But it’ll be an issue when the number of rows in the table grows. To fix this, you need to use a functional index on the date column for using the YEAR() function. Consider the following example.

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

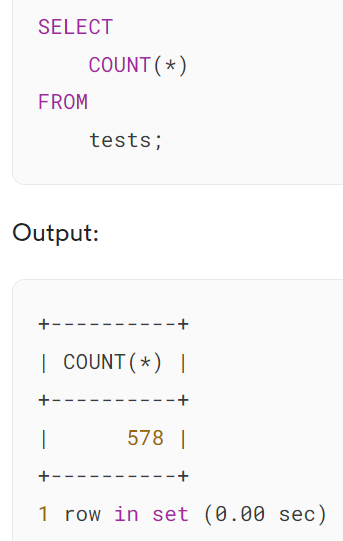


The date column will store the date data.

Second, insert many rows into the tests table using a [recursive CTE](https://www.mysqltutorial.org/mysql-basics/mysql-recursive-cte/) that generates a series of dates:

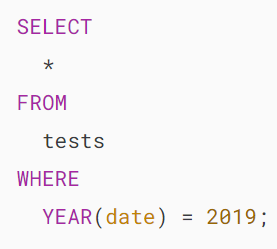


Third, get the number of rows in the tests table:



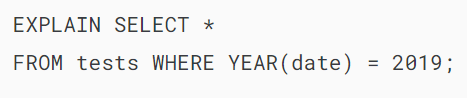
The tests table has 578 rows.

Finally, retrieve all the rows whose dates are in 2019:

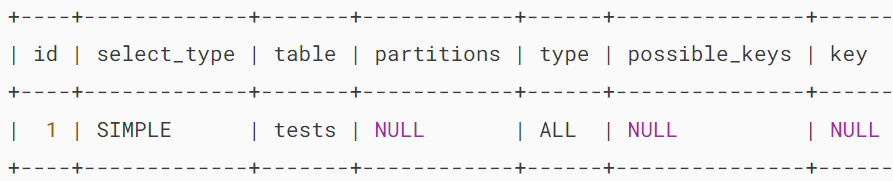


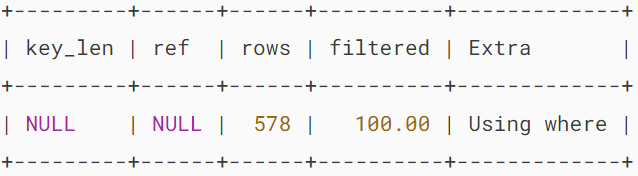
It returned 212 rows.

The following statement explains how MySQL retrieves data:



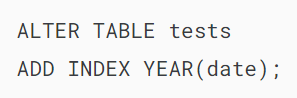
Output:



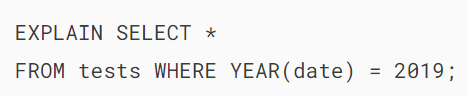


The query does not use an index.

To improve the performance, you can create a functional index on the date column of the tests table as follows:



The following query shows that querying data from the tests table based on the YEAR() function of the date column will use the index:



Output:

## 

## **Summary**

* Use the YEAR() function to extract the year of a date value.

<https://www.mysqltutorial.org/mysql-date-functions/mysql-yearweek/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-quarter/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-month-function/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-weekday-function/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-week/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-day/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-dayofyear/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-dayofweek-function/>

…

<https://www.mysqltutorial.org/mysql-date-functions/mysql-last_day/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-dayname/>

# **MySQL MAKEDATE() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL MAKEDATE() function to return a date based on a specified year and the day of the year.

## **Introduction to MySQL MAKEDATE() function**

The MAKEDATE() function returns a [date](https://www.mysqltutorial.org/mysql-basics/mysql-date/) based on a specified year and day of the year. Here’s the MAKEDATE() function syntax:



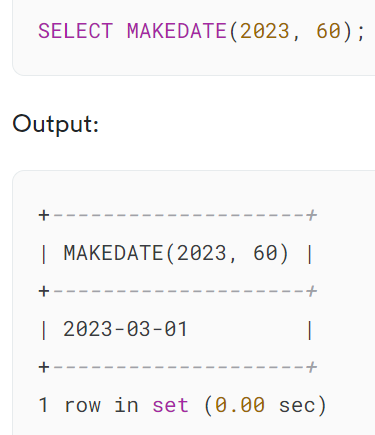
In this syntax:

* year: This is the year for which you want to create a date.
* dayofyear: This is the day of the year for which you want to create a date.

If year or dayofyear is NULL, the MAKEDATE() function returns NULL.

## **MySQL MAKEDATE() function example**

The following example uses the MAKEDATE() function to calculate the date on the 60 days of 2023:



## **Summary**

* Use the MAKEDATE() function to make a date based on the year and day of the year.

<https://www.mysqltutorial.org/mysql-date-functions/mysql-maketime/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-period_add/>

<https://www.mysqltutorial.org/mysql-date-functions/mysql-period_diff/>