27. MySQL Date and Time Function

Date & Time function allows you to manipulate date and time data effectively. This article provides detailed information and step-by-step examples of functions such as YEAR(), MONTH(), WEEK(), DAY(), DAYOFMONTH(), HOUR(), MINUTE(), SECOND(), TIME_TO_SEC(), YEARWEEK(), ADDTIME(), SUBTIME() and TIMESTAMP(). End of this article you can identify to compare dates and times using above functions.

DATE()

The DATE(Input) function returns the date in yyyy-mm-dd format. The input parameter can be given as date/datetime.

SQL Statement / Example(s)	Result(s)

`SELECT DATE("2019-12-30 17:45:59");`
`SELECT DATE("2019-12-30 17:45:59.123456");`
`SELECT DATE(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT DATE(CURDATE()); -- E.g. 2019-12-30`
`SELECT DATE("2019-12-30");`
	2019-12-30
	2019-12-30
	2019-12-30
	2019-12-30
	2019-12-30

Query 1. Using DATE in SQL

YEAR()

The YEAR(Input) function returns the year in yyyy format. The input parameter can be given as date/datetime.

SQL Statement / Example(s)	Result(s)

`SELECT YEAR("2019-12-30 17:45:59");`
`SELECT YEAR("2019-12-30 17:45:59.123456");`
`SELECT YEAR(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT YEAR(CURDATE()); -- E.g. 2019-12-30`
`SELECT YEAR("10000-12-30 17:45:59");`
	2019
	2019
	2019
	2019
	NULL

Query 2. Using YEAR in SQL

MONTH()

The MONTH(Input) function returns the month (1~12). The input parameter can be given as date/datetime.

SQL Statement / Example(s)	Result(s)

`SELECT MONTH("2019-12-30 17:45:59");`
`SELECT MONTH("2019-12-30 17:45:59.123456");`
`SELECT MONTH(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT MONTH(CURDATE()); -- E.g. 2019-12-30`
`SELECT MONTH("2019-0-0 0:0:0");`
	12
	12
	12
	12
	0

Query 3. Using MONTH in SQL

WEEK()

The WEEK(Input) function returns the week (0~53). The input parameter can be given as date/datetime.

SQL Statement / Example(s)	Result(s)

`SELECT WEEK("2019-1-5 23:59:59");`
`SELECT WEEK("2019-1-6 0:0:0");`
`SELECT WEEK(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT WEEK(CURDATE()); -- E.g. 2019-12-30`
`SELECT WEEK("2019-0-0 0:0:0");`
	0
	1
	52
	52
	NULL

Query 4. Using WEEKin SQL

DAY()

The DAY(Input) function returns the day (1~31). The input parameter can be given as date/datetime. It is equal to function DAYOFMONTH(Input).

SQL Statement / Example(s)	Result(s)

`SELECT DAY("2019-12-30");`
`SELECT DAY("2019-12-30 17:45:59");`
`SELECT DAY("2019-11-31 17:45:59");`
`SELECT DAY("2019-12-30 17:45:59.123456");`
`SELECT DAY(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT DAY(CURDATE()); -- E.g. 2019-12-30`
	30
	30
	NULL
	30
	30
	30

Query 5. Using DAY in SQL

DAYOFMONTH()

The DAYOFMONTH(Input) function returns the day (1~31). The input parameter can be given as date/datetime. It is equal to function DAY(Input).

SQL Statement / Example(s)	Result(s)
`SELECT DAYOFMONTH ("2019-12-30");` `SELECT DAYOFMONTH ("2019-12-30 17:45:59.123456");`	30 30

Query 6. Using DATEOFMONTH in SQL

HOUR()

The HOUR(Input) function returns the hour (0~23). The input parameter can be given as datetime/time.

SQL Statement / Example(s)	Result(s)

`SELECT HOUR("2019-12-30 17:45:59");`
`SELECT HOUR("2019-12-30 17:45:59.123456");`
`SELECT HOUR(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT HOUR(CURDATE()); -- E.g. 2019-12-30`
`SELECT HOUR("2019-12-30 0:0:0");`
	17
	17
	17
	0
	0

Query 7. Using HOUR in SQL

MINUTE()

The MINUTE(Input) function returns minutes (0~59). The input parameter can be given as datetime/time.

SQL Statement / Example(s)	Result(s)

`SELECT MINUTE("2019-12-30 17:45:59");`
`SELECT MINUTE("2019-12-30 17:45:59.123456");`
`SELECT MINUTE(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
`SELECT MINUTE("2019-12-30 24:0:0");`
	45
	45
	45
	NULL

Query 8. Using MINUTE in SQL

SECOND()

The SECOND(Input) function returns seconds (0~59). The input parameter can be given as datetime/time.

SQL Statement / Example(s)	Result(s)
`SELECT SECOND("2019-12-30 17:45:59");`	59

```
SELECT SECOND("2019-12-30 17:45:59.123456");
SELECT SECOND(NOW()); -- E.g. 2019-12-30 17:45:59.123456
SELECT SECOND("2019-12-30 0:60:00");
```

```
59
59
NULL
```

Query 9. Using SECOND in SQL

TIME()

The TIME(Input) function casts a string into time or extract time from date time. The input parameter can be given as datetime/time.

SQL Statement / Example(s)	Result(s)

`SELECT TIME("17:45:59");`
`SELECT TIME("2019-12-30 3:4:59");`
`SELECT TIME("2019-12-30 3:4:59.123456");`
`SELECT TIME(CURDATE()); -- E.g. 2019-12-30`
`SELECT TIME(NOW()); -- E.g. 2019-12-30 17:45:59.123456`
	`17.45.59`
	`03.04.59`
	`03.04.59.123456`
	`00:00:00`
	`17.45.59`

Query 10. Using TIME in SQL

TIME_TO_SEC()

The TIME_TO_SEC(Input) function converts time to seconds. The input parameter can be given as datetime/time.

SQL Statement / Example(s)	Result(s)

`SELECT TIME_TO_SEC("1:0:0");`
`SELECT TIME_TO_SEC("-1:1:1");`
`SELECT TIME_TO_SEC("2019-12-30 0:1:1.123456");`
`SELECT TIME_TO_SEC(CURDATE()); -- E.g. 2019-12-30`
`SELECT TIME_TO_SEC(NOW()); -- E.g. 2019-12-30 0:1:1.123456`
	`3600`
	`-3661`
	`61`
	`0`
	`61`

Query 11. Using TIME_TO_SEC in SQL

YEARWEEK()

The YEARWEEK(Input) function returns year with week (0~53). The input parameter can be given as date/datetime.

SQL Statement / Example(s)	Result(s)
`SELECT YEARWEEK ("2019-12-30 17:45:59.123456");` `SELECT YEARWEEK ("2019-1-5 17:45:59.123456");`	`201952` `201852`

Query 12. Using YEARWEEK in SQL

ADDTIME()

The ADDTIME(Input1, Input2) function adds time to a time or datetime and returns as time or datetime format.

Input1 parameter can be given as date/datetime/time.
Input2 parameter can be given as time.

SQL Statement / Example(s)	Result(s)

`SELECT ADDTIME("2019-12-30 23:59:59", "1");`
`SELECT ADDTIME("2019-12-30 23:59:59", "1 1:1:1.000001");`
`SELECT ADDTIME("23:59:59", "1 1:1:1.000001");`
	2019-12-31 00:00:00
	2020-01-01 01:01:00.000001
	49:01:00.000001

Query 13. Using ADDTIME in SQL

SUBTIME()

The SUBTIME(Input1, Input2) function adds time to a time or datetime and returns as time or datetime format.

Input1 parameter can be given as date/datetime.
Input2 parameter can be given as time.

SQL Statement / Example(s)	Result(s)

`SELECT SUBTIME("2019-12-30 23:59:59", "1");`
`SELECT SUBTIME("2019-12-30 0:0:0", "1 1:1:1.000001");`
`SELECT SUBTIME("1 11:11:11", "1 1:1:1");`
`SELECT SUBTIME("0 0:0:0", "1 1:1:1");`
	2019-12-30 23:59:58
	2019-12-28 22:58:58.999999
	10:10:10
	-25:01:01

Query 14. Using SUBTIME in SQL

TIMESTAMP()

The TIMESTAMP(Input1, Input2) function concatenates date with time and returns in yyyy-mm-dd hh:mm:ss format.

Input1 parameter can be given as date/datetime.
Input2 parameter can be given as time.

SQL Statement / Example(s)	Result(s)

`SELECT TIMESTAMP ("2019-12-30 23:59:59", "1");`
`SELECT TIMESTAMP ("2019-12-30", "23:59:59");`
`SELECT TIMESTAMP ("2019-12-30");`
	`2019-12-31 00:00:00`
	`2019-12-30 23:59:59`
	`2019-12-30 00:00:00`

Query 15. Using TIMESTAMP in SQL

Date Comparison

Dates can compare without formatting. Consider the below examples.

Table: Trips

Id	StartDate	EndDate
1	2019-01-20	2019-01-23
2	2019-01-21	2019-01-25
3	2019-01-22	2019-01-25
4	2019-01-22	2019-01-24
5	2019-01-24	2019-01-24
6	2019-01-25	2019-01-25

Table 1. Sample data for trips

Example #1: Compare using “=”

SQL Statement:

```
SELECT * FROM Trips WHERE StartDate = "2019-1-22";
```

Compare using “=” in SQL

Result:

Id	StartDate	EndDate
3	2019-01-22	2019-01-25
4	2019-01-22	2019-01-24

Table 2. Query Result

Example #2: Compare using “>” or “<”

SQL Statement:

```
SELECT * FROM Trips WHERE StartDate > "2019-1-22";
```

Compare using “>” in SQL

Result:

Id	StartDate	EndDate
5	2019-01-24	2019-01-24
6	2019-01-25	2019-01-25

Table 3. Query Result

Example #3: Compare using Between

SQL Statement:

```
SELECT * FROM Trips WHERE StartDate BETWEEN "2019-1-22" AND "2019-1-24";
```

Compare using BETWEEN in SQL

Alternative:

```
SELECT * FROM Trips WHERE StartDate >= "2019-1-22" AND StartDate <=
"2019-1-24";
```

Alternative SQL for BETWEEN

Result:

Id	StartDate	EndDate
3	2019-01-22	2019-01-25
4	2019-01-22	2019-01-24
5	2019-01-24	2019-01-24

Table 4. Query Result

Datetime Comparison

Datetimes can compare by formatting. Consider the below examples.

Table: Trips

Id	StartDate	EndDate
1	2019-01-20 9:00:10	2019-01-23 17:30:11
2	2019-01-21 9:01:35	2019-01-25 17:28:17
3	2019-01-22 9:13:12	2019-01-25 17:10:34
4	2019-01-22 9:14:00	2019-01-24 17:20:20
5	2019-01-24 9:20:11	2019-01-24 17:14:13
6	2019-01-25 9:25:21	2019-01-25 17:13:18

Table 5. Sample data for trips

Example #4: Compare using "="

SQL Statement:

```
SELECT * FROM Trips WHERE Date(StartDate) = "2019-1-22";
```

Compare using "=" in SQL

Result:

Id	StartDate	EndDate
3	2019-01-22 9:13:12	2019-01-25 17:10:34
4	2019-01-22 9:14:00	2019-01-24 17:20:20

Table 6. Query Result

Example #5: Compare using ">" or "<"

SQL Statement:

```
SELECT * FROM Trips WHERE Date(StartDate) > "2019-1-22";
```

Compare using ">" in SQL

Result:

Id	StartDate	EndDate
5	2019-01-24 9:20:11	2019-01-24 17:14:13
6	2019-01-25 9:25:21	2019-01-25 17:13:18

Table 7. Query Result

Example #6: Compare using Between

SQL Statement:

```
SELECT * FROM Trips WHERE Date(StartDate) BETWEEN "2019-1-22" AND
"2019-1-24";
```

Compare using BETWEEN in SQL

Alternative-1:

```
SELECT * FROM Trips WHERE Date(StartDate) >= "2019-1-22" AND Date(StartDate)
<= "2019-1-24";
```

Alternative SQL for BETWEEN

Alternative-2:

```
SELECT * FROM Trips WHERE  DATE_FORMAT(StartDate, "%Y-%m-%d") BETWEEN
"2019-01-22" AND "2019-01-24";
```

Compare using BETWEEN in SQL

Result:

Id	StartDate	EndDate
3	2019-01-22 9:13:12	2019-01-25 17:10:34
4	2019-01-22 9:14:00	2019-01-24 17:20:20
5	2019-01-24 9:20:11	2019-01-24 17:14:13

Table 8. Query Result