Table of Contents

This document deals with some of the functions that MySQL provides to work with temporal data.

You should use these functions rather than trying to build your own functions or expressions. For example, if you want to get the month of a date value as the month named spelled out, you use

```
select Date_Format (current_date, '%M'), ;
+---------------------------------+
| Date_Format (current_date, '%M') |
+---------------------------------+
| February                        |
+---------------------------------+
```

or Select MonthName(current_date)

You do not extract the month as number and write a 12 part case expression to get the month name.

If you want to find a date two months after a specified date, you use the Date_Add function.

1. Date formatting

Formatting is tedious but gives you some flexibility in how a date value should be display. **When you format a date, the result is a string.**

1.1. Date _Format()

First an example: this shows formatting codes which start with the % character and literals such as the / character and blanks. This also shows that the formatting codes are case specific %m is a two digit month number and %M is the month named spelled out.

Demo 01: This uses a variable @d which is assigned a string; the string will be treated as a date by the function Date_Format

```
set @d := '2011-02-20';
select   Date_Format(@d, '%Y/%m/%d'), Date_format(@d, '%M %D');
+----------------------------+-------------------------+
| Date_format(@d, '%Y/%m/%d') | Date_format(@d, '%M %D') |
+----------------------------+-------------------------+
| 2011/02/20                 | February 20th           |
+----------------------------+-------------------------+
```

Some of the format codes and their meaning (see the manual for more)

- %Y 4 digit year
- %m two digit month number
- %c one or two digit month number
- %M month name
- %b month name abbreviated
- %d two digit day number
- %e one or two digit day number
- %D day number with suffix as 4th
- %W weekday name
- %a weekday name abbreviated
- %j day of year as number

1.2. Str_To_Date()

The str_to_Date function uses the same format codes to take a string and return a date. This can be useful if you have been supplied date strings in a particular format.

Demo 02:

```
select  str_to_date( 'July 4, 2012', '%M %e, %Y');
+-----------------------------------------+
| str_to_date( 'July 4, 2012', '%M %e, %Y') |
+-----------------------------------------+
| 2012-07-04                              |
+-----------------------------------------+

select   str_to_date('2011,2,20','%Y,%m,%d');
+-----------------------------------+
| str_to_date('2011,2,20','%Y,%m,%d') |
+-----------------------------------+
| 2011-02-20                        |
+-----------------------------------+
```

In this version I skipped the comma after the day **format** and I get a null return value and a warning. Extra blanks do not cause an error.

```
select   str_to_date( 'July 4, 2012', '%M %e %Y');
+----------------------------------------+
| str_to_date( 'July 4, 2012', '%M %e %Y') |
+----------------------------------------+
| NULL                                   |
+----------------------------------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1411): Incorrect datetime value: 'July 4, 2012' for function str_to_date
```

2. Extracting part of a date value

These functions are used to retrieve parts of a data value either as a number or a string

```
EXTRACT( temp_component FROM date_exp)

YEAR(date_exp)
MONTH(date_exp)
WEEK(date_exp)
DAYOFMONTH(date_exp)
DAYOFWEEK(date_exp)
```

```
DAYOFYEAR(date_exp)
MONTHNAME(date_exp)
DAYNAME(date_exp)
```

Demo 03: Date parts using a different value for the variable @d

```
Set @d = '2009-08-15';

select
YEAR(@d), MONTH(@d), WEEK(@d), DAYOFMONTH(@d),
DAYOFWEEK(@d), DAYOFYEAR(@d),
MONTHNAME(@d), DAYNAME(@d)
\G
*************************** 1. row ***************************
         YEAR(@d): 2009
        MONTH(@d): 8
         WEEK(@d): 32
   DAYOFMONTH(@d): 15
    DAYOFWEEK(@d): 7
    DAYOFYEAR(@d): 227
    MONTHNAME(@d): August
      DAYNAME(@d): Saturday
```

Demo 04: Extract date part

```
select  hire_date
, EXTRACT(YEAR   FROM hire_date) AS YearHired
, EXTRACT(MONTH  FROM hire_date) AS MonthHired
, EXTRACT(DAY    FROM hire_date) AS DayHired
from a_emp.employees
limit 4;
+------------+----------+-----------+----------+
| hire_date  | YearHired | MonthHired | DayHired |
+------------+----------+-----------+----------+
| 1989-06-17 |      1989 |          6 |       17 |
| 2008-06-17 |      2008 |          6 |       17 |
| 2010-06-12 |      2010 |          6 |       12 |
| 2010-08-01 |      2010 |          8 |        1 |
+------------+----------+-----------+----------+
```

Demo 05: We want the people hired in August of any year.

```
select emp_id, hire_date
from a_emp.employees
where EXTRACT(MONTH FROM hire_date) = 8;
+--------+------------+
| emp_id | hire_date  |
+--------+------------+
|    103 | 2010-08-01 |
|    201 | 2004-08-25 |
+--------+------------+
```

Demo 06: Who has been hired in the current year?

```
select emp_id, hire_date
from a_emp.employees
where EXTRACT(YEAR FROM hire_date) = EXTRACT(YEAR FROM current_date() );

Empty set (0.01 sec)
```

Who has been hired in the year three years ago?

```
select emp_id, hire_date
from a_emp.employees
where EXTRACT(YEAR FROM hire_date) = EXTRACT(YEAR FROM current_date() ) -3 ;
+--------+------------+
| emp_id | hire_date  |
+--------+------------+
|    104 | 2012-01-25 |
|    109 | 2012-02-29 |
|    110 | 2012-12-31 |
|    146 | 2012-02-29 |
+--------+------------+
```

Demo 07: You can use the simpler functions of Year, Quarter, Month, Day, Hour, Minute, Second to extract parts of the dates.

```
Set @d = '2011-08-15  20:15:33';
select  @d
, year(@d)  as year,    quarter(@d) as quarter
, month(@d) as month,   day(@d)     as day
, hour(@d)  as hour,    minute(@d)  as minute;
+---------------------+------+---------+-------+------+------+--------+
| @d                  | year | quarter | month | day  | hour | minute |
+---------------------+------+---------+-------+------+------+--------+
| 2011-08-15  20:15:33 | 2011 |       3 |     8 |   15 |   20 |     15 |
+---------------------+------+---------+-------+------+------+--------+
```

Demo 08: You can get the word for the date parts.

```
select  @d, dayName(@d), monthname(@d);
+---------------------+-------------+----------------+
| @d                  | dayName(@d) | monthname(@d)  |
+---------------------+-------------+----------------+
| 2011-08-15  20:15:33 | Monday      | August         |
+---------------------+-------------+----------------+
```

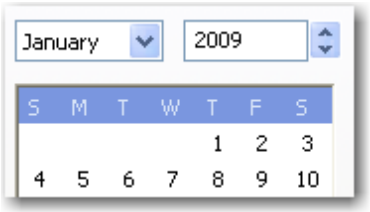
The question- which day of the week is it- as a number- is more complex. Do you consider Sunday to be the first day of the week? Or Monday? Is the first day a numeric value of 0? Or 1?

DayOfWeek uses Sunday =1 and Saturday =7

Weekday uses Monday =0 and Sunday = 6

```
select  @d, dayofWeek(@d), weekday(@d) ;
+---------------------+---------------+-------------+
| @d                  | dayofWeek(@d) | weekday(@d) |
+---------------------+---------------+-------------+
| 2011-08-15  20:15:33 |             2 |           0 |
+---------------------+---------------+-------------+
```

The questions- which week of the year is it is even more complex. In 2009, Jan 1 was a Thursday. So is the first week of the year Jan 1, 2, 3 or Jan 4, 5, 6, 7, 8, 9, 10. Or maybe we start with Monday and the first week is Jan 1, 2, 3, 4 or Jan 5, 6, 7, 8, 9, 10, 11. And is the first week- week 0 or week 1?



MySQL gives you eight choices! You include a mode argument for your choice

Mode	First day of week	Range	Week 1 is the first week
0	Sunday	0-53	with a Sunday in this year
2	Sunday	1-53	with a Sunday in this year
4	Sunday	0-53	with more than 3 days this year
6	Sunday	1-53	with more than 3 days this year
5	Monday	0-53	with a Monday in this year
7	Monday	1-53	with a Monday in this year
1	Monday	0-53	with more than 3 days this year
3	Monday	1-53	with more than 3 days this year

If you do not include a mode then the default is used. What is the default? This is stored in a variable. The value of this variable can be changed.

```
show variables like 'default_week_format';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| default_week_format | 0     |
+----------------------+-------+
```

A few examples:

Demo 09: using Jan 3, 2009 as the date.

```
set @dtm = '2009-01-03';
select  week(@dtm) as m_default
, week(@dtm,0) as m_0,     week(@dtm, 1) as m_1
, week(@dtm,2) as m_2,     week(@dtm, 3) as m_3
, week(@dtm,4) as m_4,     week(@dtm, 5) as m_5
, week(@dtm,6) as m_6,     week(@dtm, 7) as m_7;
+-----------+------+------+------+------+------+------+------+------+
| m_default | m_0  | m_1  | m_2  | m_3  | m_4  | m_5  | m_6  | m_7  |
+-----------+------+------+------+------+------+------+------+------+
|         0 |    0 |    1 |   52 |    1 |    0 |    0 |   53 |   52 |
+-----------+------+------+------+------+------+------+------+------+
```

Demo 10: using Jan 10, 2009 as the date.

```
set @dtm = '2009-01-10';
select  week(@dtm) as m_default
, week(@dtm,0) as m_0,     week(@dtm, 1) as m_1
, week(@dtm,2) as m_2,     week(@dtm, 3) as m_3
, week(@dtm,4) as m_4,     week(@dtm, 5) as m_5
, week(@dtm,6) as m_6,     week(@dtm, 7) as m_7;
+-----------+------+------+------+------+------+------+------+------+
| m_default | m_0  | m_1  | m_2  | m_3  | m_4  | m_5  | m_6  | m_7  |
+-----------+------+------+------+------+------+------+------+------+
|         1 |    1 |    2 |    1 |    2 |    1 |    1 |    1 |    1 |
+-----------+------+------+------+------+------+------+------+------+
```

This is a good time to remind you that you do **not** need to memorize all of these details. I want you to understand that some of these issues can be complex and to be careful when interpreting results from queries that you may be using.

3. Date Arithmetic

To do date manipulation use the date functions. This is a more standard approach to date arithmetic.

- Date_Add(), AddDate()
- Date_Sub(), SubDate()
- DateDiff()

3.1. Intervals

One of the issues with trying to do arithmetic with dates by simply adding a number- such as 40- is that it is not obvious if this is 40 days or 40 years or 40 seconds. A more precise way to handle this is to specify the unit of time you want to add. This is called the Interval. Some of the intervals are shown here. There also are time intervals and more of the combined intervals

- Year
- Quarter
- Month
- Day
- Hour
- Year_Month

Some of the functions default to an interval of Days.

3.2. Date_Add(), AddDate()

Demo 11: Date_Add

```
set @d = '2011-02-20';
select  @d
, Date_add(@d, interval 40 day)   as '40days'
, Date_add(@d, interval 40 year)  as '40years'
, Date_add(@d, interval 51 month) as '51months';
+----------------+-----------+-----------+-----------+
| @d | 40days     | 40years    | 51months   |
+----------------+-----------+-----------+-----------+
| 2011-02-20     | 2011-04-01 | 2051-02-20 | 2015-05-20 |
+----------------+-----------+-----------+-----------+
```

Demo 12: Adding one month to Jan 31 does not result in Feb31- instead the return value is adjusted to the end of the month.

```
select  Date_add('2009-01-31', interval 1 month) as Jan31
, Date_add('2009-01-28', interval 1 month) as Jan28
;
+------------+------------+
| Jan31      | Jan28      |
+------------+------------+
| 2009-02-28 | 2009-02-28 |
+------------+------------+
```

Demo 13: Compound intervals. The delimiter between the components can be any punctuation character

```
set @d := '2011-02-20 19:55:09';
select   @d
, Date_add(@d , interval '4 3' year_month) as '4 years 3 months'
, Date_add(@d , interval '2-23' day_hour)  as '2 days 23 hours'
;
```

```
+---------------------+---------------------+---------------------+
| now()               | 4 years 3 months    | 2 days 23 hours     |
+---------------------+---------------------+---------------------+
| 2011-02-20 19:55:09 | 2015-05-20 19:55:09 | 2011-02-23 18:55:09 |
+---------------------+---------------------+---------------------+
```

Demo 14: The interval value can be negative; if the variable has a time components, then MySQL casts it to a datetime value; if there is no time component, then the value is cast to a date

```
set @d := '2011-02-20';
select   @d
, Date_add(@d, interval '-4 3' year_month) as 'minus 4 years 3 months'
, Date_add(@d, interval '-2' day) as 'minus 2 days'
;
+----------------+-----------------------+--------------+
| @d             | minus 4 years 3 months | minus 2 days |
+----------------+-----------------------+--------------+
| 2011-02-20     | 2006-11-20            | 2011-02-18   |
+----------------+-----------------------+--------------+
```

The AddDate function will default to an interval of days. Date_Add uses the Interval syntax only.

3.3. Date_Sub(), SubDate()

Demo 15: Date_sub is used to subtract a date interval.

```
set @d := '2011-02-20';
select   @d
, Date_sub(@d, interval 40 day)   as '40days'
, Date_sub(@d, interval 40 year)  as '40years'
, Date_sub(@d, interval 51 month) as '51months';
+----------------+------------+------------+------------+
| @d             | 40days     | 40years    | 51months   |
+----------------+------------+------------+------------+
| 2011-02-20     | 2011-01-11 | 1971-02-20 | 2006-11-20 |
+----------------+------------+------------+------------+
```

Demo 16: The SubDate function will default to an interval of days. Date_Sub uses the Interval syntax only.

```
set @d = '2009-07-06' ;
select   @d
, SubDate(@d, interval 40 day)   as '40days'
, SubDate(@d, 40)   as '40days'
;
+------------+------------+------------+
| @d         | 40days     | 40days     |
+------------+------------+------------+
| 2009-07-06 | 2009-05-27 | 2009-05-27 |
+------------+------------+------------+
```

3.4. DateDiff()

Demo 17: The DateDiff function returns the number of days between two dates.

```
select   @d
, DateDiff(@d, '20090720')   as col_2
, DateDiff(@d, '20090620')   as col_3
, DateDiff(@d, '20070706')   as col_4;
```

```
+----------------+-------+-------+-------+
| @d             | col_2 | col_3 | col_4 |
+----------------+-------+-------+-------+
| 2009-07-06     |   -14 |    16 |  1325 |
+----------------+-------+-------+-------+
```

Demo 18: The DateDiff function deals with the date component only. The following returns a value of -1.

```
select   DateDiff('2009-07-06 23:59:59', '2009-07-07 00:00:01')   ;
```

Demo 19: DateDiff with order dates.

```
select   ord_id
, ord_date
from a_oe.order_headers
where dateDiff(date '2014-06-30', ord_date) between 25 and 50;
+--------+---------------------+
| ord_id | ord_date            |
+--------+---------------------+
|    301 | 2014-06-04 00:00:00 |
|    302 | 2014-06-04 00:00:00 |
|    306 | 2014-06-04 00:00:00 |
|    307 | 2014-06-04 00:00:00 |
|    390 | 2014-06-04 00:00:00 |
|    395 | 2014-06-04 00:00:00 |
|    529 | 2014-05-12 00:00:00 |
|    535 | 2014-05-12 00:00:00 |
|    536 | 2014-05-12 00:00:00 |
|    540 | 2014-06-02 00:00:00 |
+--------+---------------------+
```

4. Other temporal functions

4.1. To_Days()

Demo 20: To_Days returns the number of days since the start of the calendar. By itself this is not too interesting.

```
select   current_date()
, To_DAYS (current_date()) as NowCount
, TO_DAYS('2009-07-20') as "20090720"
, TO_DAYS('1066-10-14') as "btlHst"
, TO_DAYS('0079-08-24') as "MtVsv";
+----------------+----------+----------+--------+-------+
| current_date() | NowCount | 20090720 | btlHst | MtVsv |
+----------------+----------+----------+--------+-------+
| 2015-02-11     |   736005 |   733973 | 389635 | 29090 |
+----------------+----------+----------+--------+-------+
```

Demo 21: You can subtract two of the values to get the number of days between the two dates- or use DateDiff.

```
set @d1 := '2011-08-17';
set @d2 := '2011-12-17';

select   To_Days(@d2)- To_Days(@d1);
+----------------------------+
| To_Days(@d2)- To_Days(@d1) |
+----------------------------+
|                        122 |
+----------------------------+
```

4.2. From_Days()

Demo 22: From_Days gets a number and returns a date based on the number of days since the start of the calendar.

```
select From_days(5), From_days(500), From_days(689798), From_days(733736);
+--------------+---------------+-------------------+-------------------+
| From_days(5) | From_days(500) | From_days(689798) | From_days(733736) |
+--------------+---------------+-------------------+-------------------+
| 0000-00-00   | 0001-05-15     | 1888-08-08        | 2008-11-25        |
+--------------+---------------+-------------------+-------------------+
```

4.3. Last_Day

Demo 23: Last_day gets date argument and returns the last day of that month

```
Set @d1 = '2011-03-25';
Set @d2 = '2011-03-31';
select   last_day(@d1), last_day(@d2);
+---------------+---------------+
| last_day(@d1) | last_day(@d2) |
+---------------+---------------+
| 2011-03-31    | 2011-03-31    |
+---------------+---------------+

select   last_day (From_days(689798));
+------------------------------+
| last_day (From_days(689798)) |
+------------------------------+
| 1888-08-31                   |
+------------------------------+
```

5. Direct manipulation of date values

You can do date value plus a number. But you need to pay attention to how this is handled. This is very error prone.

Demo 24: Now() + 40 is **not** 40 days from now and the value returned is numeric in format. Note where the 40 got added.

```
select   now(), now() + 40   as Plus40;
+---------------------+----------------+
| now()               | Plus40         |
+---------------------+----------------+
| 2015-02-11 20:52:11 | 20150211205251 |
+---------------------+----------------+
```

Demo 25: Now try this with CurDate()- this is not a good idea! And the warning does not make a lot of sense.

```
select  curdate()
, curdate()+ 40   as Plus40
, Month(curdate()+ 40)   as Month40;
+------------+----------+---------+
| curdate()  | Plus40   | Month40 |
+------------+----------+---------+
| 2015-02-11 | 20150251 |    NULL |
+------------+----------+---------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Incorrect datetime value: '20150211'
```

Demo 26: Now repeat this by adding 4; In this case we get a date value

```
select  now()
, now()+ 4   as Plus4
, Month(now()+ 4)   as Month4;
+---------------------+----------------+--------+
| now()               | Plus4          | Month4 |
+---------------------+----------------+--------+
| 2015-02-11 20:54:02 | 20150211205406 |      2 |
+---------------------+----------------+--------+
```

You are advised to use the date functions rather than trying to use these manipulations.