Table of Contents

1. Data types for temporal data

MySQL supports the following data types for temporal data: Date, DateTime. Year, Time, TimeStamp

In class we will work mostly with the Date and DateTime types. You may ignore the other types mentioned here.

Date values have the format CCYY-MM-DD with a range of 1000-01-01 to 9999-12-31. (You can insert earlier date values - such as '0005-11-26' but the MySQL manual states that these dates are not supported and might not work as expected.)

DateTime values have the format CCYY-MM-DD hh:mm:ss. The time component defaults to midnight. In the DateTime type, the time component is time of day.

2. Getting the current date: CURDATE(), CURRENT_DATE() , NOW()

Most of the demos will focus on the date and datetime types. Most functions treat these interchangeably. There are similar functions for time which are not generally mentioned here.

Demo 01: Current date functions

```
select curdate(), current_date(), current_date;  -- these are synonyms
+------------+---------------+--------------+
| curdate()  | current_date() | current_date |
+------------+---------------+--------------+
| 2016-02-13 | 2016-02-13     | 2016-02-13   |
+------------+---------------+--------------+

select now();  --- includes the time
+---------------------+
| now()               |
+---------------------+
| 2016-02-13 20:31:26 |
+---------------------+
```

3. Date values versus string values

Hopefully you noticed that the date literal '2015-05-31' is really a string literal. This is common in database systems and the string '2015-05-31' will be treated as a date if it is used in an expression where MySQL expects a date. We have seen this when we do an insert statement, inserting a value such as '2015-05-31' into a date attribute.

Demo 02: We can use an explicit cast if we want. But you do not commonly see this done.

```
select
  cast('2015-05-31' as date)
, cast('2015-05-31' as datetime);
```

```
+----------------------------+---------------------------------+
| cast('2015-05-31' as date) | cast('2015-05-31' as datetime) |
+----------------------------+---------------------------------+
| 2015-05-31                 | 2015-05-31 00:00:00             |
+----------------------------+---------------------------------+
```

You will see a lot of MySQL code which treats date values as if they are strings, depending on the format of YYYY-MM-DD. We will also see more standard ways of doing these manipulations.

Demo 03: MySQL is willing to do cast from dates to strings depending on the usage.
These functions return numbers.

```
select
  current_date
, left(current_date,4) as theYear
, substring(current_date,6,2) as theMonth
, substring(current_date,9,2) as theDay;
+--------------+---------+----------+--------+
| current_date | theYear | theMonth | theDay |
+--------------+---------+----------+--------+
| 2016-02-13   | 2016    | 02       | 13     |
+--------------+---------+----------+--------+
```

Demo 04: But this is easier and clearer to do. These functions return numbers.

```
select
  current_date
, year(current_date) as theYear
, month(current_date) as theMonth
, dayofmonth(current_date) as theDay;
+--------------+---------+----------+--------+
| current_date | theYear | theMonth | theDay |
+--------------+---------+----------+--------+
| 2016-02-13   |    2016 |        2 |     13 |
+--------------+---------+----------+--------+
```

Question: since the substring expression here returns a two character string value (such as '06' and the month function returns a number, what will the following query return?

```
select *
from vets.vt_animals
where month(an_dob) =substring(an_dob,6,2);
```

4. MySQL issues with temporal data

For these demos use a full date value such as '2009-06-25'. All of the date values we use in class will be complete date values, but you do need to be aware of other MySQL date issues such as values such as 2010-00-00.

Prior to version 5.0.2, MySQL checked date values for validity by checking that the month component is between 1 and 12 and the date component is between 1 and 31. This allowed for date values such as February 31, 2009. The current version of MySQL rejects invalid dates.

The temporal types have a 0 value used for invalid dates.

Some date functions accept a 0 value for the date components.

Some date functions return 0 for incomplete dates.

Demo 05:

```
select  month ('2525-00-00');
+----------------------+
| month ('2525-00-00') |
+----------------------+
|                    0 |
+----------------------+
```

In general, if you supply invalid date values to functions, the results could be unexpected.

5. How temporal values are stored

MySQL stores the temporal values as integers. The manual explains this as:

MySQL temporal values Storage:

- **DATE**: A three-byte integer packed as **DD** + **MM**×32 + **YYYY**×16×32
- **DATETIME**: Eight bytes:
- A four-byte integer packed as **YYYY**×10000 + **MM**×100 + **DD**
- A four-byte integer packed as **HH**×10000 + **MM**×100 + **SS**
- **YEAR**: A one-byte integer
- **TIME**: A three-byte integer packed as **DD**×24×3600 + **HH**×3600 + **MM**×60 + **SS**
- **TIMESTAMP**: A four-byte integer representing seconds UTC since the epoch (**'1970-01-01 00:00:00'** UTC)

Since one of the reasons for using a dbms is to isolate the user from the physical storage patterns of the data you should not really need to know that- but in reality sometimes things make more sense when you have some idea about data storage. Don't memorize these!