```
mysql> SET sql_mode = '';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT CAST(0 AS UNSIGNED) - 1;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(cast(0 as unsigned) - 1)'
```

If the NO_UNSIGNED_SUBTRACTION SQL mode is enabled, the result is negative:

```
mysql> SET sql_mode = 'NO_UNSIGNED_SUBTRACTION';
mysql> SELECT CAST(0 AS UNSIGNED) - 1;
+-------------------------+
| CAST(0 AS UNSIGNED) - 1 |
+-------------------------+
|                      -1 |
+-------------------------+
```

If the result of such an operation is used to update an UNSIGNED integer column, the result is clipped to the maximum value for the column type, or clipped to 0 if NO_UNSIGNED_SUBTRACTION is enabled. If strict SQL mode is enabled, an error occurs and the column remains unchanged.

11.2 Date and Time Data Types

The date and time data types for representing temporal values are DATE, TIME, DATETIME, TIMESTAMP, and YEAR. Each temporal type has a range of valid values, as well as a “zero” value that may be used when you specify an invalid value that MySQL cannot represent. The TIMESTAMP and DATETIME types have special automatic updating behavior, described in Section 11.2.5, “Automatic Initialization and Updating for TIMESTAMP and DATETIME”.

For information about storage requirements of the temporal data types, see Section 11.7, “Data Type Storage Requirements”.

For descriptions of functions that operate on temporal values, see Section 12.7, “Date and Time Functions”.

Keep in mind these general considerations when working with date and time types:

- MySQL retrieves values for a given date or time type in a standard output format, but it attempts to interpret a variety of formats for input values that you supply (for example, when you specify a value to be assigned to or compared to a date or time type). For a description of the permitted formats for date and time types, see Section 9.1.3, “Date and Time Literals”. It is expected that you supply valid values. Unpredictable results may occur if you use values in other formats.

- Although MySQL tries to interpret values in several formats, date parts must always be given in year-month-day order (for example, '98-09-04'), rather than in the month-day-year or day-month-year orders commonly used elsewhere (for example, '09-04-98', '04-09-98'). To convert strings in other orders to year-month-day order, the STR_TO_DATE() function may be useful.

- Dates containing 2-digit year values are ambiguous because the century is unknown. MySQL interprets 2-digit year values using these rules:

  - Year values in the range 70-99 become 1970-1999.

  - Year values in the range 00-69 become 2000-2069.

  See also Section 11.2.8, “2-Digit Years in Dates”.

- Conversion of values from one temporal type to another occurs according to the rules in Section 11.2.7, “Conversion Between Date and Time Types”.