```
1 row in set (0.00 sec)
```

Such casts can also be seen by executing `EXPLAIN [FORMAT=TRADITIONAL]`, in which case it is also necessary to issue `SHOW WARNINGS` after executing the `EXPLAIN` statement.

- **Time zone support for TIMESTAMP and DATETIME.** As of MySQL 8.0.19, the server accepts a time zone offset with inserted datetime (`TIMESTAMP` and `DATETIME`) values. This offset uses the same format as that employed when setting the `time_zone` system variable, except that a leading zero is required when the hours portion of the offset is less than 10, and `'-00:00'` is not allowed. Examples of datetime literals that include time zone offsets are `'2019-12-11 10:40:30-05:00'`, `'2003-04-14 03:30:00+10:00'`, and `'2020-01-01 15:35:45+05:30'`.

  Time zone offsets are not displayed when selecting datetime values.

  Datetime literals incorporating time zone offsets can be used as prepared statement parameter values.

  As part of this work, the value used to set the `time_zone` system variable is now also restricted to the range `-14:00` to `+14:00`, inclusive. (It remains possible to assign name values to `time_zone` such as `'EST'`, `'Posix/Australia/Brisbane'`, and `'Europe/Stockholm'` to this variable, provided that the MySQL time zone tables are loaded; see Populating the Time Zone Tables).

  For more information and examples, see Section 5.1.15, “MySQL Server Time Zone Support”, as well as Section 11.2.2, “The DATE, DATETIME, and TIMESTAMP Types”.

- **Precise information for JSON schema CHECK constraint failures.** When using `JSON_SCHEMA_VALID()` to specify a `CHECK` constraint, MySQL 8.0.19 and later provides precise information about the reasons for failures of such constraints.

  For examples and more information, see JSON_SCHEMA_VALID() and CHECK constraints. See also Section 13.1.20.6, “CHECK Constraints”.

- **Row and column aliases with ON DUPLICATE KEY UPDATE.** Beginning with MySQL 8.0.19, it is possible to reference the row to be inserted, and, optionally, its columns, using aliases. Consider the following `INSERT` statement on a table `t` having columns `a` and `b`:

  ```
  INSERT INTO t SET a=9,b=5
      ON DUPLICATE KEY UPDATE a=VALUES(a)+VALUES(b);
  ```

  Using the alias `new` for the new row, and, in some cases, the aliases `m` and `n` for this row's columns, the `INSERT` statement can be rewritten in many different ways, some examples of which are shown here:

  ```
  INSERT INTO t SET a=9,b=5 AS new
      ON DUPLICATE KEY UPDATE a=new.a+new.b;

  INSERT INTO t VALUES(9,5) AS new
      ON DUPLICATE KEY UPDATE a=new.a+new.b;

  INSERT INTO t SET a=9,b=5 AS new(m,n)
      ON DUPLICATE KEY UPDATE a=m+n;

  INSERT INTO t VALUES(9,5) AS new(m,n)
      ON DUPLICATE KEY UPDATE a=m+n;
  ```

  For more information and examples, see Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”.