See Obtaining Information with EXPLAIN ANALYZE, for more information.

- **Query cast injection.** In version 8.0.18 and later, MySQL injects cast operations into the query item tree inside expressions and conditions in which the data type of the argument and the expected data type do not match. This has no effect on query results or speed of execution, but makes the query as executed equivalent to one which is compliant with the SQL standard while maintaining backwards compatibility with previous releases of MySQL.

  Such implicit casts are now performed between temporal types (`DATE`, `DATETIME`, `TIMESTAMP`, `TIME`) and numeric types (`SMALLINT`, `TINYINT`, `MEDIUMINT`, `INT`/`INTEGER`, `BIGINT`; `DECIMAL`/`NUMERIC`; `FLOAT`, `DOUBLE`, `REAL`; `BIT`) whenever they are compared using any of the standard numeric comparison operators (`=`, `>=`, `>`, `<`, `<=`, `<>`/`!=`, or `<=>`). In this case, any value that is not already a `DOUBLE` is cast as one. Cast injection is also now performed for comparisons between `DATE` or `TIME` values and `DATETIME` values, where the arguments are cast whenever necessary as `DATETIME`.

  Beginning with MySQL 8.0.21, such casts are also performed when comparing string types with other types. String types that are cast include `CHAR`, `VARCHAR`, `BINARY`, `VARBINARY`, `BLOB`, `TEXT`, `ENUM`, and `SET`. When comparing a value of a string type with a numeric type or `YEAR`, the string cast is to `DOUBLE`; if the type of the other argument is not `FLOAT`, `DOUBLE`, or `REAL`, it is also cast to `DOUBLE`. When comparing a string type to a `DATETIME` or `TIMESTAMP` value, the string is cast is to `DATETIME`; when comparing a string type with `DATE`, the string is cast to `DATE`.

  It is possible to see when casts are injected into a given query by viewing the output of `EXPLAIN ANALYZE`, `EXPLAIN FORMAT=JSON`, or, as shown here, `EXPLAIN FORMAT=TREE`:

```
mysql> CREATE TABLE d (dt DATETIME, d DATE, t TIME);
Query OK, 0 rows affected (0.62 sec)

mysql> CREATE TABLE n (i INT, d DECIMAL, f FLOAT, dc DECIMAL);
Query OK, 0 rows affected (0.51 sec)

mysql> CREATE TABLE s (c CHAR(25), vc VARCHAR(25),
    ->     bn BINARY(50), vb VARBINARY(50), b BLOB, t TEXT,
    ->     e ENUM('a', 'b', 'c'), se SET('x' ,'y', 'z'));
Query OK, 0 rows affected (0.50 sec)

mysql> EXPLAIN FORMAT=TREE SELECT * from d JOIN n ON d.dt = n.i\G
*************************** 1. row ***************************
EXPLAIN: -> Inner hash join (cast(d.dt as double) = cast(n.i as double))
(cost=0.70 rows=1)
    -> Table scan on n  (cost=0.35 rows=1)
    -> Hash
        -> Table scan on d  (cost=0.35 rows=1)

mysql> EXPLAIN FORMAT=TREE SELECT * from s JOIN d ON d.dt = s.c\G
*************************** 1. row ***************************
EXPLAIN: -> Inner hash join (d.dt = cast(s.c as datetime(6)))  (cost=0.72 rows=1)
    -> Table scan on d  (cost=0.37 rows=1)
    -> Hash
        -> Table scan on s  (cost=0.35 rows=1)

1 row in set (0.01 sec)

mysql> EXPLAIN FORMAT=TREE SELECT * from n JOIN s ON n.d = s.c\G
*************************** 1. row ***************************
EXPLAIN: -> Inner hash join (cast(n.d as double) = cast(s.c as double))  (cost=0.70 rows=1)
    -> Table scan on s  (cost=0.35 rows=1)
    -> Hash
        -> Table scan on n  (cost=0.35 rows=1)
```