- **New optimizer_switch flags.** MySQL 8.0.21 adds two new flags for the `optimizer_switch` system variable, as described in the following list:

  - `prefer_ordering_index` flag

    By default, MySQL attempts to use an ordered index for any `ORDER BY` or `GROUP BY` query that has a `LIMIT` clause, whenever the optimizer determines that this would result in faster execution. Because it is possible in some cases that choosing a different optimization for such queries actually performs better, it is now possible to disable this optimization by setting the `prefer_ordering_index` flag to `off`.

    The default value for this flag is `on`.

  - `subquery_to_derived` flag

    When this flag is set to `on`, the optimizer transforms eligible scalar subqueries into joins on derived tables. For example, the query `SELECT * FROM t1 WHERE t1.a > (SELECT COUNT(a) FROM t2)` is rewritten as `SELECT t1.a FROM t1 JOIN ( SELECT COUNT(t2.a) AS c FROM t2 ) AS d WHERE t1.a > d.c`.

    This optimization can be applied to a subquery which is part of a `SELECT`, `WHERE`, `JOIN`, or `HAVING` clause; contains one or more aggregate functions but no `GROUP BY` clause; is not correlated; and does not use any nondeterministic functions.

    The optimization can also be applied to a table subquery which is the argument to `IN`, `NOT IN`, `EXISTS`, or `NOT EXISTS`, and which does not contain a `GROUP BY`. For example, the query `SELECT * FROM t1 WHERE t1.b < 0 OR t1.a IN (SELECT t2.a + 1 FROM t2)` is rewritten as `SELECT a, b FROM t1 LEFT JOIN (SELECT DISTINCT 1 AS e1, t2.a AS e2 FROM t2) d ON t1.a + 1 = d.e2 WHERE t1.b < 0 OR d.e1 IS NOT NULL`.

    This optimization is normally disabled, as it does not yield a noticeable performance benefit in most cases, and so the flag is set to `off` by default.

  For more information, see Section 8.9.2, “Switchable Optimizations”. See also Section 8.2.1.19, “LIMIT Query Optimization”, Section 8.2.2.1, “Optimizing IN and EXISTS Subquery Predicates with Semijoin Transformations”, and Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”.

- **XML enhancements.** As of MySQL 8.0.21, the `LOAD XML` statement now supports `CDATA` sections in the XML to be imported.

- **Casting to the YEAR type now supported.** Beginning with MySQL 8.0.22, the server allows casting to `YEAR`. Both the `CAST()` and `CONVERT()` functions support single-digit, two-digit, and four-digit `YEAR` values. For one-digit and two-digit values, the allowed range is 0-99. Four-digit values must be in the range 1901-2155. `YEAR` can also be used as the return type for the `JSON_VALUE()` function; this function supports four-digit years only.

  String, time-and-date, and floating-point values can all be cast to `YEAR`. Casting of `GEOMETRY` values to `YEAR` is not supported.

  For more information, including conversion rules, see the description of the `CONVERT()` function.

- **Retrieval of TIMESTAMP values as UTC.** MySQL 8.0.22 and later supports conversion of a `TIMESTAMP` column value from the system time zone to a UTC `DATETIME` on retrieval, using `CAST(value AT TIME ZONE specifier AS DATETIME)`, where the specifier is one of