- **Optimizer hints for FORCE INDEX, IGNORE INDEX.** MySQL 8.0 introduces index-level optimizer hints which serve as analogs to the traditional index hints as described in Section 8.9.4, “Index Hints”. The new hints are listed here, along with their `FORCE INDEX` or `IGNORE INDEX` equivalents:

  - `GROUP_INDEX`: Equivalent to `FORCE INDEX FOR GROUP BY`

    `NO_GROUP_INDEX`: Equivalent to `IGNORE INDEX FOR GROUP BY`

  - `JOIN_INDEX`: Equivalent to `FORCE INDEX FOR JOIN`

    `NO_JOIN_INDEX`: Equivalent to `IGNORE INDEX FOR JOIN`

  - `ORDER_INDEX`: Equivalent to `FORCE INDEX FOR ORDER BY`

    `NO_ORDER_INDEX`: Equivalent to `IGNORE INDEX FOR ORDER BY`

  - `INDEX`: Same as `GROUP_INDEX` plus `JOIN_INDEX` plus `ORDER_INDEX`; equivalent to `FORCE INDEX` with no modifier

    `NO_INDEX`: Same as `NO_GROUP_INDEX` plus `NO_JOIN_INDEX` plus `NO_ORDER_INDEX`; equivalent to `IGNORE INDEX` with no modifier

  For example, the following two queries are equivalent:

  ```
  SELECT a FROM t1 FORCE INDEX (i_a) FOR JOIN WHERE a=1 AND b=2;

  SELECT /*+ JOIN_INDEX(t1 i_a) */ a FROM t1 WHERE a=1 AND b=2;
  ```

  The optimizer hints listed previously follow the same basic rules for syntax and usage as existing index-level optimizer hints.

  These optimizer hints are intended to replace `FORCE INDEX` and `IGNORE INDEX`, which we plan to deprecate in a future MySQL release, and subsequently to remove from MySQL. They do not implement a single exact equivalent for `USE INDEX`; instead, you can employ one or more of `NO_INDEX`, `NO_JOIN_INDEX`, `NO_GROUP_INDEX`, or `NO_ORDER_INDEX` to achieve the same effect.

  For further information and examples of use, see Index-Level Optimizer Hints.

- **JSON_VALUE() function.** MySQL 8.0.21 implements a new function `JSON_VALUE()` intended to simplify indexing of `JSON` columns. In its most basic form, it takes as arguments a JSON document and a JSON path pointing to a single value in that document, as well as (optionally) allowing you to specify a return type with the `RETURNING` keyword. `JSON_VALUE(`*`json_doc`*`, `*`path`*` RETURNING `*`type`*`)` is equivalent to this:

  ```
  CAST(
      JSON_UNQUOTE( JSON_EXTRACT(json_doc, path) )
      AS type
  );
  ```

  You can also specify `ON EMPTY`, `ON ERROR`, or both clauses, similar to those employed with `JSON_TABLE()`.

  You can use `JSON_VALUE()` to create an index on an expression on a `JSON` column like this:

  ```
  CREATE TABLE t1(
      j JSON,
      INDEX i1 ( (JSON_VALUE(j, '$.id' RETURNING UNSIGNED)) )
  );
  ```