- When sorting `JSON` values in a query using `ORDER BY`, each value is now represented by a variable-length part of the sort key, rather than a part of a fixed 1K in size. In many cases this can reduce excessive usage. For example, a scalar `INT` or even `BIGINT` value actually requires very few bytes, so that the remainder of this space (up to 90% or more) was taken up by padding. This change has the following benefits for performance:

  - Sort buffer space is now used more effectively, so that filesorts need not flush to disk as early or often as with fixed-length sort keys. This means that more data can be sorted in memory, avoiding unnecessary disk access.

  - Shorter keys can be compared more quickly than longer ones, providing a noticeable improvement in performance. This is true for sorts performed entirely in memory as well as for sorts that require writing to and reading from disk.

- Added support in MySQL 8.0.2 for partial, in-place updates of `JSON` column values, which is more efficient than completely removing an existing JSON value and writing a new one in its place, as was done previously when updating any `JSON` column. For this optimization to be applied, the update must be applied using `JSON_SET()`, `JSON_REPLACE()`, or `JSON_REMOVE()`. New elements cannot be added to the JSON document being updated; values within the document cannot take more space than they did before the update. See Partial Updates of JSON Values, for a detailed discussion of the requirements.

  Partial updates of JSON documents can be written to the binary log, taking up less space than logging complete JSON documents. Partial updates are always logged as such when statement-based replication is in use. For this to work with row-based replication, you must first set `binlog_row_value_options=PARTIAL_JSON`; see this variable's description for more information.

- Added the JSON utility functions `JSON_STORAGE_SIZE()` and `JSON_STORAGE_FREE()`. `JSON_STORAGE_SIZE()` returns the storage space in bytes used for the binary representation of a JSON document prior to any partial update (see previous item). `JSON_STORAGE_FREE()` shows the amount of space remaining in a table column of type `JSON` after it has been partially updated using `JSON_SET()` or `JSON_REPLACE()`; this is greater than zero if the binary representation of the new value is less than that of the previous value.

  Each of these functions also accepts a valid string representation of a JSON document. For such a value, `JSON_STORAGE_SIZE()` returns the space used by its binary representation following its conversion to a JSON document. For a variable containing the string representation of a JSON document, `JSON_STORAGE_FREE()` returns zero. Either function produces an error if its (non-null) argument cannot be parsed as a valid JSON document, and `NULL` if the argument is `NULL`.

  For more information and examples, see Section 12.18.8, “JSON Utility Functions”.

  `JSON_STORAGE_SIZE()` and `JSON_STORAGE_FREE()` were implemented in MySQL 8.0.2.

- Added support in MySQL 8.0.2 for ranges such as `$[1 to 5]` in XPath expressions. Also added support in this version for the `last` keyword and relative addressing, such that `$[last]` always selects the last (highest-numbered) element in the array and `$[last-1]` the next to last element. `last` and expressions using it can also be included in range definitions. For example, `$[last-2 to last-1]` returns the last two elements but one from an array. See Searching and Modifying JSON Values, for additional information and examples.