Heatwave Release Notes

Abstract

This document contains release notes for the changes in each release of HeatWave.

For additional HeatWave documentation, see HeatWave User Guide.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2024-12-24 (revision: 29446)

Table of Contents

Preface and Legal Notices 2
Changes in HeatWave 3
Changes in HeatWave 9.1.2 (2024-12-17, General Availability) 3
Changes in HeatWave 9.1.1 (2024-11-19, General Availability) 4
Changes in HeatWave 9.1.0 (2024-10-15, General Availability) 5
Changes in HeatWave 9.0.1-u1 (2024-09-02, General Availability) 6
Changes in HeatWave 9.0.1 (2024-07-23, General Availability) 7
Changes in HeatWave 9.0.0 (2024-07-02, General Availability) 7
Changes in HeatWave 8.4.0 (2024-04-30, General Availability) 10
Changes in HeatWave 8.3.0-u2 (2024-02-26, General Availability) 11
Changes in HeatWave 8.3.0 (2024-01-17, General Availability) 11
Changes in HeatWave 8.2.0-u2 (2023-12-19, General Availability) 12
Changes in HeatWave 8.2.0-u1 (2023-12-05, General Availability) 12
Changes in HeatWave 8.2.0 (2023-10-25, General Availability) 13
Changes in HeatWave 8.1.0-u4 (2023-09-26, General Availability) 14
Changes in HeatWave 8.1.0-u3 (2023-09-12, General Availability) 15
Changes in HeatWave 8.1.0-u2 (2023-08-08, General Availability) 15
Changes in HeatWave 8.1.0-u1 (2023-07-25, General Availability) 15
Changes in HeatWave 8.1.0 (2023-07-18, General Availability) 16
Changes in HeatWave 8.0.33-u3 (2023-06-09, General Availability) 16
Changes in HeatWave 8.0.33 (2023-04-27, General Availability) 17
Changes in HeatWave 8.0.32-u2 (2023-02-21, General Availability) 17
Changes in HeatWave 8.0.32-u1 (2023-02-21, General Availability) 17
Changes in HeatWave 8.0.32 (2023-01-17, General Availability) 18
Changes in HeatWave 8.0.31 (2022-10-11, General Availability) 19
Changes in HeatWave 8.0.30-u1 (2022-09-06, General Availability) 20
Changes in HeatWave 8.0.30 (2022-07-26, General Availability) 20
Changes in HeatWave 8.0.28-u3 (2022-04-19, General Availability) 23
Changes in HeatWave 8.0.28-u2 (2022-03-29, General Availability) 23
Changes in HeatWave 8.0.28-u1 (2022-02-15, General Availability) 23
Changes in HeatWave 8.0.27-u3 (2021-12-15, General Availability) 23
Changes in HeatWave 8.0.27-u2 (2021-12-07, General Availability) 24

Changes in HeatWave 8.0.26-u2 (2021-09-21, General Availability) 24
Changes in HeatWave 8.0.26-u1 (2021-08-10, General Availability) 25
Changes in HeatWave 8.0.26 (2021-07-23, General Availability) 25
Changes in HeatWave 8.0.25 (2021-05-11, General Availability) 27
Changes in HeatWave 8.0.24 (2021-04-20, General Availability) 27
Changes in HeatWave 8.0.23-u2 (2021-03-15, General Availability) 28
Changes in HeatWave 8.0.23-u1 (2021-02-09, General Availability) 28
Index 29

Preface and Legal Notices

This document contains release notes for the changes in each release of HeatWave.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Changes in HeatWave

Changes in HeatWave 9.1.2 (2024-12-17, General Availability)

Note

These release notes were created with the assistance of HeatWave GenAI.

- HeatWave AutoML

- HeatWave GenAI
- HeatWave Lakehouse

HeatWave AutoML

- HeatWave AutoML now supports more efficient memory usage, reducing the footprint of its processes during training and explanations for large datasets. (WL #16606)

HeatWave GenAI

- HeatWave GenAI supports enhanced unstructured document ingestion with Optical Character Recognition (OCR) now enabled by default. (WL #16584)
- The `ML_RAG` and `ML_RAG_TABLE` routines now supports advanced retrieval options, enabling more precise and context-aware results. Three new parameters have been added to the routines:
  - `max_distance` for filtering segments based on distance from the input query.
  - `percentage_distance` for adaptive distance filtering of segments from the input query.
  - `segment_overlap` for retrieving segments adjacent to the segment nearest to the input query for providing continuous context.

  These new parameters are integrated into the existing `@options` parameter, and can be enabled through a new JSON object parameter, `@retrieval_options`. (WL #16543)

HeatWave Lakehouse

- HeatWave Lakehouse now supports selective load, which lets you update the file paths comprising a Lakehouse table by manipulating Lakehouse metadata. The changes are incrementally applied to the table data. This feature enables you to add or remove Object Storage bucket data directories from a Lakehouse table without reloading the entire table. Changes are executed transactionally, ensuring data consistency and query isolation.

  For more information, see Editing Loaded Data. (WL #16531)
- HeatWave Lakehouse now supports enhanced error and warning messages, providing more data points for easier debugging. This update adds row numbers and character offsets to relevant warning messages in non-strict mode during secondary load, as well as updates byte offset text messages to point to column positions. JSON datatype error and warning messages have also been improved with error and warning offset information within the column.

  For more information, see HeatWave Lakehouse Error Messages. (WL #16530)
- HeatWave Lakehouse now supports efficient loading of Parquet files with large row groups (larger than 500MB and upto 10GB) using large shapes. This helps in overcoming previous memory budget constraints and significantly reducing load times. This enhancement also introduces row group sharing, caching, and synchronization mechanisms to minimize network requests, decoding time, and memory usage during transformation.

  For more information, see Lakehouse Limitations for the Parquet File Format. (WL #16512)

Changes in HeatWave 9.1.1 (2024-11-19, General Availability)

Note

These release notes were created with the assistance of HeatWave GenAI.

- HeatWave AutoML

- HeatWave Lakehouse

HeatWave AutoML

- HeatWave AutoML now supports the Orbit forecasting model, which uses Local Global Trend (LGT). This addition expands the scope of predictive analytics and enables more advanced time-series forecasting for businesses.

  For more information, see Forecasting. (WL #16231)

HeatWave Lakehouse

- HeatWave Lakehouse now supports exporting query results directly to an object store in OCI or AWS. This lets you store, transform, and persist data in CSV and Parquet formats. The new syntaxes introduced make it easier to export data, especially in command-line environments, and provide more control over the output format. This enhancement enables efficient data storage, retrieval, and transformation, improving data management and analysis workflows.

  For more information, see Exporting Query Results to Object Storage. (WL #16214)

Changes in HeatWave 9.1.0 (2024-10-15, General Availability)

- HeatWave GenAI
- HeatWave Lakehouse
- HeatWave MySQL

HeatWave GenAI

- HeatWave GenAI now lets you extract and encode text from images stored in unstructured documents using Optical Character Recognition (OCR). The extracted text is converted into vector embeddings and stored in a vector store the same way regular text in unstructured documents is encoded and stored in a vector store.

  For more information, see About Optical Character Recognition. (WL #16474)

- MySQL JavaScript Stored Programs now include a new GenAI API that you can use to call different HeatWave GenAI routines using JavaScript functions.

  For more information, see JavaScript GenAI API. (WL #16487)

HeatWave Lakehouse

- HeatWave Lakehouse now supports Zone Maps, which helps you improve range queries in OLAP and mixed workloads.

  MySQL HeatWave uses statistics based on minimum and maximum values to create zone maps for every primary key column. It then uses the zone maps for range and point queries to only scan data chunks that are relevant for the query, and accelerates these queries by an order of magnitude. (WL #16414)

HeatWave MySQL

- Auto Parallel Load now uses Autopilot to collect statistics about frequently used InnoDB and HeatWave tables. Auto Parallel Load then automatically loads these tables to HeatWave. If a manually loaded table might cause a memory conflict, Auto Unload unloads automatically loaded tables to free up memory. (WL #15491)

- HeatWave now supports InnoDB partitions, and `INFORMATION_SCHEMA.PARTITIONS` includes a `SECONDARY_LOAD` column that indicates whether a partition has been loaded.

When an unload operation successfully unloads the last partition, it unloads the whole table.

HeatWave MySQL now persists partition information, and during a restart, only the previously loaded partitions are loaded.

For more information, see Loading Partitions (WL #15517, WL #16268)

Changes in HeatWave 9.0.1-u1 (2024-09-02, General Availability)

- HeatWave AutoML
- HeatWave GenAI
- HeatWave Lakehouse
- HeatWave MySQL

HeatWave AutoML

- HeatWave AutoML now supports semi-supervised learning for anomaly detection. This type of machine learning algorithm uses a specific set of labeled data along with unlabeled data to detect anomalies. (WL #16373)
- HeatWave AutoML now supports topic modeling. This is an unsupervised machine learning technique that is capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize the documents. (WL #16275)

HeatWave GenAI

- HeatWave GenAI now supports multiple languages, which lets you use the HeatWave GenAI APIs in non-English languages. This includes ingestion of documents written in languages other than English, vector and similarity search, and querying these documents by giving prompts in the same language.

  For more information, see Languages. (WL #16475)

- HeatWave GenAI now supports large language model (LLM) inference batch processing, which lets you run the HeatWave GenAI routines on multiple queries, in parallel, across different nodes in the HeatWave cluster. Thus, it improves the LLM inference performance and throughput while keeping the inference latency of every request unchanged.

  For more information, see ML_GENERATE_TABLE, ML_RAG_TABLE, and ML_EMBED_TABLE. (WL #16399)

- The HeatWave GenAI `ML_RAG` routine now includes new filtering options, `exclude_vector_store` and `exclude_document_name`, which let you exclude specific vector store tables or documents from context retrieval. (WL #16401)

- You can now use incremental load to refresh vector store tables. For more information, see Loading Data Incrementally into the Vector Store Table. (WL #15817)

- Auto Parallel Load for unstructured data provides a new `format` option setting, `auto_unstructured`, which lets you ingest multiple files with different unstructured data formats in a single load.

  Additionally, for ingesting Microsoft PowerPoint and Microsoft Word files into the vector store, Auto Parallel Load introduces the following new format aliases: `pptx` and `docx`.

  For more information, see Ingesting Files Using Auto Parallel Load. (WL #16433)

HeatWave Lakehouse

- HeatWave Lakehouse now supports `timestamp_format` as a `dialect` parameter to customize the format for columns of the `DATETIME` and `TIMESTAMP` data types. You can also set the formats for specific columns by using the `ENGINE_ATTRIBUTE` option. Formats set at the column level override formats set with the `dialect` parameter. For more information, see Lakehouse External Table Syntax. (WL #16425)
- Some HeatWave Lakehouse error messages include the URL to the external object. The URL can contain up to 1024 characters, and exceed the error message limit of 512 characters. The truncated error message will now include a MySQL command to access the full error message.

  It is now possible to filter out HeatWave Lakehouse warning messages. The console **Total Warnings** will include both the displayed and the filtered warnings count. See: HeatWave Lakehouse Error Messages.

  HeatWave Lakehouse now supports `max_error_count`. (WL #16141)

HeatWave MySQL

- • `GROUP_CONCAT()` function now supports the `CUBE`, `ROLLUP`, and `WITH ROLLUP` options.

  • `COUNT(DISTINCT)` function now supports the `CUBE`, `ROLLUP`, and `WITH ROLLUP` options. (WL #16365)
- HeatWave MySQL now supports the following JSON functions:
  - `JSON_INSERT()`
  - `JSON_REMOVE()`
  - `JSON_REPLACE()`
  - `JSON_SET()`
  - `JSON_ARRAY_APPEND()`
  - `JSON_ARRAY_INSERT()`
  - `JSON_MERGE_PATCH`
  - `JSON_MERGE_PRESERVE`
  - `JSON_MERGE()`
  - `JSON_SCHEMA_VALID()`
  - `JSON_SCHEMA_VALIDATION_REPORT()`

  (WL #16345)

Changes in HeatWave 9.0.1 (2024-07-23, General Availability)

This release does not include any new features.

Changes in HeatWave 9.0.0 (2024-07-02, General Availability)

- HeatWave AutoML
- HeatWave Autopilot
- HeatWave GenAI

- HeatWave Lakehouse
- HeatWave MySQL
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML has increased the model size from 900MB to 4GB. (WL #15931)

HeatWave Autopilot

- HeatWave Autopilot includes improvements to cardinality estimation for queries that run in HeatWave MySQL and HeatWave Lakehouse. This improves the query plan, and provides improved performance (WL #15995)
- HeatWave Autopilot now includes Auto Indexing Advisor that offers indexing suggestions to improve the performance and latency of OLTP workloads running on the HeatWave MySQL primary engine. (WL #15663)

HeatWave GenAI

- HeatWave GenAI introduces similarity search capabilities that support the `DISTANCE` function that measures the distance between two vector values. It can include an optional `distance_metric` that can have a value of `DOT`, `COSINE` or `EUCLIDIAN`. This is available in the primary and secondary engines. (WL #16081)
- HeatWave GenAI introduces the new native `VECTOR` data type and in-database vector store support. This provides a fast, end-to-end, fully integrated pipeline which automates the vector store creation:
  - Reading unstructured data in PDF, HTML, TXT, PPT or DOCX format from object store.
  - Parsing the text in the documents.
  - Partitioning the text into smaller paragraphs and segments.
  - Encoding the paragraphs.
  - Storing the encoded paragraphs in a standard MySQL table in HeatWave.

  (WL #16106)
- HeatWave GenAI introduces a set of intuitive chat functionalities that enable enterprises to quickly build their own AI powered chatbot. It also introduces an out-of-the-box chatbot – HeatWave Chat, which integrates the chat functionalities that enable users to interact with HeatWave using natural language in addition to SQL. (WL #16248)
- HeatWave GenAI introduces in-database LLM with support for the following LLMs, see HeatWave GenAI:
  - Mistral-7B-Instruct.
  - Llama-2-8B-Instruct.
  - LLMs from OCI GenAI.

  (WL #15844, WL #16145)

HeatWave Lakehouse

- HeatWave Lakehouse now supports unstructured documents, and loads them to the GenAI vector store. Supported formats include PDF, HTML, TXT, PPT and DOCX. (WL #16100)

- HeatWave Lakehouse introduces Lakehouse Incremental Load that uses change propagation to refresh data views following changes to a loaded table. (WL #15817)

HeatWave MySQL

- HeatWave MySQL includes improvements to query optimization that lead to better performance for low latency queries in HeatWave MySQL and HeatWave Lakehouse. (WL #16222)
- HeatWave MySQL includes improvements to change propagation performance. (WL #16148)
- HeatWave MySQL improves the debugging of offloaded queries. (WL #15988)
- HeatWave MySQL now supports the use of cursors from stored procedures. (WL #16142)
- HeatWave MySQL now has support for the `JSON` data type with the following functions:
  - Cast Functions and Operators.
  - `COALESCE()` and `IN()`, see: Comparison Functions and Operators.
  - Control Flow Functions and Operators.
  - Mathematical Functions.
  - String Functions and Operators.

  (WL #16048)
- HeatWave MySQL now supports the following JSON functions:
  - `JSON_ARRAYAGG()`
  - `JSON_CONTAINS()`
  - `JSON_CONTAINS_PATH()`
  - `JSON_KEYS()`
  - `JSON_OBJECTAGG()`
  - `JSON_OVERLAPS()`
  - `JSON_PRETTY()`
  - `JSON_QUOTE()`
  - `JSON_SEARCH()`
  - `JSON_STORAGE_FREE()`
  - `JSON_STORAGE_SIZE()`
  - `JSON_TYPE()`
  - `JSON_VALID()`
  - `JSON_VALUE()`
  - `MEMBER OF()`

  (WL #16046, WL #16047)
- HeatWave MySQL now includes support for `GROUP_CONCAT()` with `DISTINCT` optimization and the `ORDER BY` clause. (WL #15906)

- HeatWave now supports dynamic query offload which uses machine learning based on query properties to determine the best engine, InnoDB or HeatWave analytic engine, to process the query. (WL #15829)
- HeatWave MySQL now supports concurrent query execution that reduces query latencies, and increases resource utilization. (WL #14826)

Functionality Added or Changed

- MySQL InnoDB on MySQL HeatWave includes a bulk load extension to the `LOAD DATA` statement. This now uses multiple threads to load a series of CSV files to improve performance, and allow the LOAD operation to scale with multiple CPUs. (WL #15611)
- MySQL InnoDB on MySQL HeatWave includes a bulk load extension to the `LOAD DATA` statement. This now supports large objects with `VARCHAR` and `VARBINARY`. It also supports the following data types:
  - `BIT`.
  - `ENUM`.
  - `JSON`.
  - `SET`.
  - `TIMESTAMP`.
  - `YEAR`.
  - `TINYBLOB`, `BLOB`, `MEDIUMBLOB`, and `LONGBLOB`.
  - `TINYTEXT`, `TEXT`, `MEDIUMTEXT`, and `LONGTEXT`.
  - `GEOMETRY`, `GEOMETRYCOLLECTION`, `POINT`, `MULTIPOINT`, `LINESTRING`, `MULTILINESTRING`, `POLYGON`, and `MULTIPOLYGON`,

  See: Bulk Ingest Data to MySQL Server. (WL #16264, WL #16265)

Changes in HeatWave 8.4.0 (2024-04-30, General Availability)

- HeatWave Lakehouse
- Functionality Added or Changed

HeatWave Lakehouse

- HeatWave Lakehouse now supports primary key and unique key constraints. (WL #16094)
- HeatWave Lakehouse now supports the Newline Delimited JSON file format, see HeatWave Lakehouse. (WL #16079)
- HeatWave Lakehouse now has improved support for missing files. Auto Parallel Load now has a `validation` mode to validate files before loading them. (WL #16078)

Functionality Added or Changed

- HeatWave now supports named time zones such as `MET` or `Europe/Amsterdam`. (WL #16098)
- HeatWave Auto Parallel Load and Auto Unload now support a table inclusion list, as well as a table exclusion list. (WL #16070)

- MySQL InnoDB on MySQL HeatWave includes a bulk load extension to the `LOAD DATA` statement. This can now load MySQL Shell dump files and compressed files. There is also a progress monitor. See: Bulk Ingest Data to MySQL Server. (WL #15814, WL #15607, WL #15608, WL #15609)

Changes in HeatWave 8.3.0-u2 (2024-02-26, General Availability)

- HeatWave AutoML
- HeatWave Lakehouse
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML adds prediction intervals to the forecasting task. (WL #16072)
- HeatWave AutoML adds the PCA and GLOF algorithms to the anomaly detection task. (WL #16104)

HeatWave Lakehouse

- HeatWave Lakehouse now supports the JSON data type. (WL #16037)
- HeatWave Lakehouse now supports Guided Load to perform pre-load validation checks and schema inference. (WL #16004)
- HeatWave Lakehouse now supports the following:
  - Point-in-time-recovery.
  - High Availability.
  - Read Replication.
  - Outbound Replication.

  (WL #15926)
- HeatWave Lakehouse now supports Auto Data Compression. This dynamically determines which compression algorithm to use for each column based on its data characteristics. This provides the optimum balance between memory usage and query performance. See: Data Management. (WL #16061)

Functionality Added or Changed

- HeatWave primary and secondary engines now support the HyperLogLog, `HLL()` aggregate function. This is similar to `COUNT(DISTINCT)`, but faster and with user defined precision. This is only available in HeatWave, see: Aggregate Functions. (WL #15992)
- HeatWave can now reload all tables, and unload all tables. (WL #16023)
- HeatWave secondary engine now supports the `TABLESAMPLE` clause to use with `SELECT` statements. (WL #15902)

Changes in HeatWave 8.3.0 (2024-01-17, General Availability)

- HeatWave Lakehouse
- Functionality Added or Changed

HeatWave Lakehouse

- HeatWave Lakehouse extends Auto Parallel Load support for external tables that have a pre-defined schema. It is now able to reconcile the inferred schema with the pre-defined table definition when

there is a column type mismatch between the two. Schema inference adjusts this schema to prevent possible errors during data loading. See: Lakehouse Auto Parallel Load with the external_tables Option. (WL #16040)

- HeatWave Lakehouse now has improved error reporting. See: HeatWave Lakehouse Error Messages. (WL #15951)

Functionality Added or Changed

- `GREATEST()` and `LEAST()` comparison and string functions now support the `YEAR` data type. (WL #15910)

Changes in HeatWave 8.2.0-u2 (2023-12-19, General Availability)

- HeatWave AutoML
- HeatWave Lakehouse

HeatWave AutoML

- The HeatWave AutoML recommendation task now supports content based recommendation models. See Recommendations. (WL #15726)
- HeatWave AutoML introduces data drift detection for classification and regression models. (WL #15866)

HeatWave Lakehouse

- HeatWave Lakehouse now supports all queries that HeatWave supports. See: HeatWave Lakehouse Limitations. (WL #15730)

Changes in HeatWave 8.2.0-u1 (2023-12-05, General Availability)

- HeatWave Lakehouse
- Functionality Added or Changed

HeatWave Lakehouse

- HeatWave Lakehouse now uses Auto Parallel Load to load Avro files. (WL #15778)
- HeatWave Lakehouse schema inference now includes support for unicode column headers. (WL #16011)

Functionality Added or Changed

- HeatWave secondary engine now supports the `QUALIFY` clause to use with `SELECT` statements. (WL #15864)
- Auto Data Compression dynamically determines which compression algorithm to use for each column based on its data characteristics. This provides the optimum balance between memory usage and query performance. See: Data Management. (WL #15248)
- HeatWave now supports the following aggregate functions used as window functions:
  - `STD()`
  - `STDDEV()`
  - `STDDEV_POP()`
  - `STDDEV_SAMP()`

  - `VAR_POP()`
  - `VAR_SAMP()`
  - `VARIANCE()`

  (WL #15367)

- HeatWave secondary engine now supports the HyperLogLog, `HLL()` aggregate function. This is similar to `COUNT(DISTINCT)`, but faster and with user defined precision. This is only available in HeatWave, see: Aggregate Functions. (WL #15914)
- The `GROUP BY` clause permits a `CUBE` modifier. This is only available in HeatWave, see: GROUP BY Modifiers . (WL #15843)
- HeatWave now supports the following JSON functions:
  - `->`
  - `->>`
  - `JSON_ARRAY()`
  - `JSON_DEPTH()`
  - `JSON_EXTRACT()`
  - `JSON_LENGTH()`
  - `JSON_OBJECT()`
  - `JSON_UNQUOTE()`

  See: JSON Functions. (WL #15355)

Changes in HeatWave 8.2.0 (2023-10-25, General Availability)

Functionality Added or Changed

- HeatWave Guided Load reduces the number of steps to manually load data, see Loading Data Manually. (WL #15811)
- HeatWave now supports the following string functions:
  - `BIN()`
  - `BIT_LENGTH()`
  - `ELT()`
  - `EXPORT_SET()`
  - `FIELD()`
  - `MAKE_SET()`

  These functions do not support `ENUM` type columns, see Data Type Limitations. (WL #15628)
- HeatWave now supports full-table aggregation and group-by support for the following temporal data types:
  - `DATE`

  - `TIME`
  - `DATETIME`
  - `TIMESTAMP`
  - `YEAR`

  HeatWave now supports the following functions with temporal data types:

  - `SUM()`
  - `AVG()`
  - `STD()`
  - `STDDEV()`
  - `STDDEV_POP()`
  - `STDDEV_SAMP()`
  - `VAR_POP()`
  - `VAR_SAMP()`
  - `VARIANCE()`

  See: Aggregate Functions. (WL #15637)

- HeatWave now enables the internal conversion of values from one data type to another, including all combinations of these different encodings:

  - Numeric values `INTEGER`, `FLOAT`, and `DECIMAL`.
  - String values, excluding dictionary-encoded strings.
  - Temporal values `DATE`, `TIME`, `DATETIME`, and `YEAR` data types.

  This now permits HeatWave to offload queries that contain functions with unexpected data types by adding an internal cast from the unexpected data type to an expected data type. (WL #15465)

Changes in HeatWave 8.1.0-u4 (2023-09-26, General Availability)

- HeatWave AutoML
- HeatWave Lakehouse
- Functionality Added or Changed

HeatWave AutoML

- The HeatWave AutoML recommendation task now supports implicit feedback. See Recommendations. (WL #15802)

HeatWave Lakehouse

- HeatWave Lakehouse now uses schema inference to create an external table, see How to Load Data from External Storage Using Auto Parallel Load. (WL #15851)
- HeatWave Lakehouse now includes support for HeatWave AutoML. See: HeatWave AutoML and Lakehouse. (WL #15539)

Functionality Added or Changed

- MySQL InnoDB on MySQL HeatWave now has support for a bulk load extension to the `LOAD DATA` statement. This is only available on MySQL HeatWave on AWS. See: MySQL Functionality for HeatWave and Bulk Ingest Data to MySQL Server. (WL #14717, WL #14772, WL #15131, WL #15132, WL #15133, WL #15612)

Changes in HeatWave 8.1.0-u3 (2023-09-12, General Availability)

- HeatWave AutoML
- HeatWave Lakehouse
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML now supports text data types, see: Supported Data Types. (WL #15743)

HeatWave Lakehouse

- HeatWave Lakehouse can now use a regular expression to define a set of files. (WL #15840)
- HeatWave Lakehouse now uses Auto Parallel Load to load Parquet files. (WL #15656)
- HeatWave Lakehouse now supports the Avro file format, see HeatWave Lakehouse. (WL #15494)

Functionality Added or Changed

- Adaptive Query Execution dynamically adjusts the query execution plan based on runtime statistics to improve query execution time. See: Query Processing. (WL #15368)

Changes in HeatWave 8.1.0-u2 (2023-08-08, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML progress tracking can now track the progress with finer granularity, and show the overall progress as well as the progress of subcomponents, see: Progress tracking. (WL #15531)

Functionality Added or Changed

- HeatWave improves time zone support. When the time zone is set to `SYSTEM` or set to an offset from UTC, queries run in HeatWave. `DATE`, `DATETIME`, `TIMESTAMP`, and `TIME` are processed based on the time-zone value. (Bug #35710134)

Changes in HeatWave 8.1.0-u1 (2023-07-25, General Availability)

Functionality Added or Changed

- HeatWave now supports the following functions:
  - `MID`
  - `REGEXP_INSTR`
  - `SOUNDEX`

- `SPACE`
- `WEIGHT_STRING`

(WL #15627)

Changes in HeatWave 8.1.0 (2023-07-18, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML has enhanced recommendation systems that can now recommend new users who are similar to a specific user and new items that are similar to other items. See Recommendations. (WL #15674)
- Improvements to HeatWave AutoML models, the model catalog, and `ML_MODEL_IMPORT` now fully support ONNX and HeatWave AutoML model import. (WL #15383, WL #15559)

Functionality Added or Changed

- Memory-aware query optimization now automatically adjusts HeatWave to optimize query execution time or memory usage and reduce the likelihood of queries running out of memory without user intervention. (WL #15529)
- It is now possible to track the memory used by change propagation and the number of transactions committed to MySQL InnoDB that are waiting to be propagated to HeatWave. (WL #15326, WL #15688)
- HeatWave now accelerates queries with `INTERSECT` and `EXCEPT` clauses. With this support, HeatWave can now offload and accelerate all 97 TPC-DS queries. (WL #15362)
- HeatWave now supports the following temporal functions:
  - `GET_FORMAT()`
  - `MAKETIME()`
  - `PERIOD_ADD()`
  - `PERIOD_DIFF()`
  - `SEC_TO_TIME()`
  - `SUBTIME()`
  - `SYSDATE()`
  - `TIMEDIFF()`

  (WL #14958)

Changes in HeatWave 8.0.33-u3 (2023-06-09, General Availability)

HeatWave Lakehouse

- MySQL HeatWave expands to include HeatWave Lakehouse, letting organizations process and query hundreds of terabytes of data residing in Object Storage—in a variety of file formats, such as CSV and Parquet.

The Lakehouse feature of MySQL HeatWave enables query processing on Object Storage-resident data. The source data is read from Object Storage, transformed to the HeatWave format, stored in the HeatWave persistence storage layer in OCI Object Storage, and then loaded to HeatWave Cluster memory.

- Provides in-memory query processing on Object Storage-resident data.
- Data is not loaded into the MySQL InnoDB storage layer.
- Supports structured and relational data in CSV and Parquet formats.
- With this feature, users can now analyze data in both MySQL InnoDB and an object store using familiar SQL syntax in the same query.

See HeatWave Lakehouse. (WL #15575)

Changes in HeatWave 8.0.33 (2023-04-27, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML now supports a recommendation task, see Recommendations. (WL #15416)

Functionality Added or Changed

- The new HeatWave Auto Unload utility automates the process of unloading tables from HeatWave. (WL #15447)
- The following columns were added to the `performance_schema.rpd_query_stats` table:
  - `CONNECTION_ID`: The ID of the connection.
  - `STATEMENT_ID`: The global query ID.

  (WL #15526)

Changes in HeatWave 8.0.32-u2 (2023-02-21, General Availability)

HeatWave AutoML

- HeatWave AutoML now supports unsupervised anomaly detection, which is the data mining task that finds unusual patterns in data. (WL #15518)

Changes in HeatWave 8.0.32-u1 (2023-02-21, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML adds support for multivariate endogenous forecasting models, and exogenous forecasting models. (WL #15511)

Functionality Added or Changed

- The `rpd_tables` table now supports recovery time measurement with `RECOVERY_SOURCE`, `RECOVERY_START_TIMESTAMP` and `RECOVERY_END_TIMESTAMP`. The

`performance_schema.global_status` now includes the `rapid_recovery_time` variable. (WL #15441)

- Arithmetic operators are now supported with variable-length encoded string columns. Mathematical functions are now supported with variable-length columns. (WL #15405)

Changes in HeatWave 8.0.32 (2023-01-17, General Availability)

- Deprecation and Removal Notes
- HeatWave AutoML
- Functionality Added or Changed

Deprecation and Removal Notes

- MySQL 8.0.32 deprecates the `heatwave_load_report` and `heatwave_advisor_report` tables, and replaces them with the `heatwave_autopilot_report` table in the `sys` schema. A future release will remove them. (Bug #34727481)

HeatWave AutoML

- MySQL 8.0.32 introduces a number of changes for HeatWave AutoML routines:

  - The routines: `ML_PREDICT_ROW`, `ML_PREDICT_TABLE`, `ML_EXPLAIN_ROW`, and `ML_EXPLAIN_TABLE` now include the `ml_results` column, which uses `JSON` format to return the results.

  - The routines: `ML_PREDICT_ROW`, `ML_PREDICT_TABLE`, `ML_EXPLAIN_ROW`, and `ML_EXPLAIN_TABLE` now allow additional columns that are not required for prediction or explanation.

  - The routines: `ML_PREDICT_ROW`, and `ML_PREDICT_TABLE` now allow an *`options`* parameter in `JSON` format.

  - The `ML_TRAIN` routine also runs the `ML_EXPLAIN` routine with the default Permutation Importance model.

  (WL #15420)

- MySQL 8.0.32 introduces progress tracking for `ML_TRAIN`. Use a second MySQL Client window to track the progress of `ML_TRAIN` with calls to the performance schema. It also introduces the `rapid_ml_operation_count` status variable. (WL #15384)

Functionality Added or Changed

- HeatWave now supports the following encryption and compression functions:

  - `COMPRESS()`
  - `MD5()`
  - `RANDOM_BYTES()`
  - `SHA()`
  - `SHA1()`
  - `SHA2()`
  - `UNCOMPRESS()`

- `UNCOMPRESSED_LENGTH()`

(WL #15283)

- HeatWave now supports the following mathematical functions:
  - `CRC32()`
  - `LOG2()`
  - `LOG10()`
  - `RAND()`

(WL #15256)

Changes in HeatWave 8.0.31 (2022-10-11, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML queries are now monitored and recorded in the Performance Schema tables `rpd_query_stats` and `rpd_exec_stats`. Where a single HeatWave AutoML query contains a number of sub-queries, there is one record in `rpd_query_stats` and multiple records in `rpd_exec_stats`. (WL #15243)
- New functions have been added to HeatWave AutoML to help you manage models:
  - When you run the `ML_TRAIN` routine on a training dataset, you can now specify a model handle to use for the model instead of the generated one.
  - A new column `notes` has been added to the `MODEL_CATALOG` table, which you can use to record notes about the models in your model catalog.
  - The new column `model_metadata` in the `MODEL_CATALOG` table records metadata for models, such as the training score, training time, and information about the training dataset. If an error occurs during training or you cancel the training operation, HeatWave AutoML records the error status in this column.

  (WL #15243)
- HeatWave AutoML now supports the upload of pre-trained models in ONNX (Open Neural Network Exchange) format to the model catalog. You can load them using the stored procedure `ML_MODEL_IMPORT` that provides the conversion required to store the model in a MySQL table. (WL #15243)
- A new stored procedure `ML_EXPLAIN` lets you train a variety of model explainers and prediction explainers for HeatWave AutoML, in addition to the default Permutation Importance model and prediction explainers:
  - The Partial Dependence model explainer shows how changing the values of one or more columns will change the value that the model predicts.
  - The SHAP model explainer produces global feature importance values based on Shapley values.
  - The Fast SHAP model explainer is a subsampling version of the SHAP model explainer which usually has a faster runtime.
  - The Permutation Importance prediction explainer explains the prediction for a single row or table.

  - The SHAP prediction explainer uses feature importance values to explain the prediction for a single row or table.

  When you use the `ML_EXPLAIN_TABLE` and `ML_EXPLAIN_ROW` stored procedures to generate explanations for a prediction, you can now use the SHAP prediction explainer as an alternative to the default Permutation Importance prediction explainer. SHAP produces feature importance values (explanations) based on Shapley values. (WL #15243)

- HeatWave AutoML now supports timeseries forecasting using the existing stored procedures `ML_TRAIN`, `ML_PREDICT_TABLE`, and `ML_SCORE`. You can create a forecast for a single column (a univariate endogenous variable) with a numeric data type. The forecasting task is specified as a JSON object when you call the `ML_TRAIN` stored procedure. (WL #15243)

Functionality Added or Changed

- HeatWave uses dictionary encoding to compress string columns (CHAR, VARCHAR, TEXT). These dictionaries are built for each string column with the `RAPID_COLUMN=ENCODING=SORTED` keyword. HeatWave now supports 8.5 billion dictionary entries (up from 4 billion), which means HeatWave can now encode string columns with number of distinct value (NDV) up to 8.5 billion. (WL #14742)

- MySQL HeatWave now uses statistics based on minimum and maximum values to create zone maps for every primary key column. HeatWave then uses the zone maps for range and point queries to only scan data chunks that are relevant for the query, and accelerates these queries by an order of magnitude. This is particularly useful for improving range queries in OLAP and mixed workloads. (WL #14713)

- A new MySQL optimizer is introduced for HeatWave to provide a holistic cost model across MySQL and HeatWave, create better query plans based on statistics used in MySQL Autopilot, reduce compilation time, eliminate the need of query hints for join order, and improve join query performance. With the new optimizer, HeatWave can now run all 22 TPC-H queries without straight join hints. Before 8.0.31, a straight join hint is needed for 10 out of 22 TPC-H to reach peak performance. (WL #14449)

- DDL statements such as `ALTER TABLE`, `RENAME TABLE`, and `TRUNCATE TABLE` are now permitted on a table that has RAPID defined as the secondary engine. If a DDL operation is successfully carried out on a table that is loaded to a HeatWave Cluster at the time, HeatWave automatically reloads the table from MySQL InnoDB. Note that if the DDL operation makes the table's structure incompatible with HeatWave, the table is unloaded from the HeatWave Cluster. (WL #15129)

Changes in HeatWave 8.0.30-u1 (2022-09-06, General Availability)

Functionality Added or Changed

- HeatWave now supports the `ABS()`, `POWER()`, and `SIGN()` mathematical functions. (WL #15246)

- HeatWave now supports the `GROUP_CONCAT()` aggregation function with variable-length (`VARLEN`) string columns. (WL #14767)

- Change propagation memory management was enhanced to improve HeatWave shutdown time. (WL #15306)

- Query performance was optimized for compressed data. Decompression of data that is queried but does not participate in filter evaluation is delayed until after a filter is applied. With this optimization, decompression is avoided entirely for data that is filtered out. (WL #15169)

Changes in HeatWave 8.0.30 (2022-07-26, General Availability)

- Advisor

- HeatWave AutoML
- Functionality Added or Changed

Advisor

- HeatWave Advisor Auto Encoding, which recommends string column encodings, now provides encoding recommendations that optimize query performance. Recommendations are based on performance models that use query execution data. Previously, string column encoding recommendations were optimized for cluster memory usage only. A performance improvement estimate is provided with string column encoding recommendations. (Bug #34145862)

HeatWave AutoML

- You can now train HeatWave AutoML models on tables containing `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, and `YEAR` data types. (Bug #33895503)
- HeatWave AutoML now generates a model explanation when you train a machine learning model. Model explanations help identify the features that are most important to a model. For more information, see The Model Catalog.

  The following columns were added to the `MODEL_CATALOG` table:

  - `column_names`: The feature columns used to train the model.
  - `last_accessed`: The last time the model was accessed. HeatWave AutoML routines update this value to the current timestamp when accessing the model.
  - `model_explanation`: The model explanation generated during training.
  - `model_type`: The type of model (algorithm) selected by `ML_TRAIN` to build the model.
  - `task`: The task type specified in the `ML_TRAIN` query (`classification` or `regression`).

  `ML_PREDICT_*` and `ML_EXPLAIN_*` routine performance was improved, resulting in faster prediction and explanation processing. (WL #15088, WL #15014)
- The following HeatWave AutoML enhancements were implemented:
  - `ML_TRAIN` options for advanced users. These options permit users to customize various aspects of the ML training pipeline including algorithm selection, feature selection, and hyperparameter optimization.
    - The `model_list` option permits specifying the type of model to be trained.
    - The `exclude_model_list` option specifies models types to exclude from consideration during model selection.
    - The `optimization_metric` option specifies the scoring metric to optimize for when training a machine learning model.
    - The `exclude_column_list` option specifies feature columns to exclude from consideration when training a machine learning model.

    For more information, see Advanced ML_TRAIN Options.
  - Support was added for Support Vector Machine `SVC` and `LinearSVC` classification and regression models. For a complete list of supported model types, see Model Types.
  - The `ML_TRAIN` routine now reports a message if a trained model does not meet expected quality criteria.

- `ML_EXPLAIN_ROW` and `ML_EXPLAIN_TABLE` routines now provide information to help interpret explanations. The routines also report a warning when a model quality issue is detected, enabling users to revisit their data in order to improve model quality.

(WL #15089)

Functionality Added or Changed

- The amount of heap memory allocated on the MySQL node for each table loaded into HeatWave was reduced, increasing the maximum number of tables that can be loaded. For `MySQL.HeatWave.VM.E3.Standard` shapes, the maximum was raised from 100k tables to 400k tables. For `MySQL.HeatWave.BM.E3.Standard` shapes, the maximum number was raised from 400k tables to 1600k tables. The actual number of tables that can be loaded is dependent on the table's data. (Bug #33951708)

- The `performance_schema.rpd_column_id` table was modified to remove redundant data. The `NAME`, `SCHEMA_NAME`, `TABLE_NAME` columns were removed, and a `TABLE_ID` column was added. (Bug #33899183)

- Support was added for the `FROM_DAYS()` temporal function, and `GREATEST()` and `LEAST()` comparison and string functions which now support `DATE`, `DATETIME`, `TIME`, and `TIMESTAMP` columns. (WL #14956)

- Support was added for built-in server-side data masking and de-identification to help protect sensitive data from unauthorized uses by hiding and replacing real values with substitutes. Data masking and de-identification operations are performed on the server, and queries involving data masking and de-identification functions are accelerated by HeatWave. The following data masking and de-identification functions are supported:

  - `gen_range()`
  - `gen_rnd_email()`
  - `gen_rnd_pan()`
  - `gen_rnd_ssn()`
  - `gen_rnd_us_phone()`
  - `mask_inner()`
  - `mask_outer()`
  - `mask_pan()`
  - `mask_pan_relaxed()`
  - `mask_ssn()`

  See Data Masking and De-Identification Functions. (WL #15143)

- Optimizations were implemented to improve performance for `JOIN` and `GROUP BY` queries with execution plans involving multiple consecutive rounds of data partitioning. (WL #15143)

- *`expr`* `IN (`*`value`*`,...)` comparisons, where the expression is a single value and compared values are constants of the same data type and encoding, have been optimized. For example, the following `IN()` comparison has been optimized:

```
SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'Spain');
```

(WL #14952)

Changes in HeatWave 8.0.28-u3 (2022-04-19, General Availability)

Functionality Added or Changed

- Tables that have become stale due to a change propagation failure resulting from an out-of-code error are now automatically reloaded. A check for stale tables is performed periodically when the HeatWave Cluster is idle. Previously, identifying and reloading stale tables was a manual process. See Change Propagation. (WL #14914)

Changes in HeatWave 8.0.28-u2 (2022-03-29, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave customers now have access to HeatWave AutoML, which is a fully managed, highly scalable, cost-efficient, machine learning solution for data stored in MySQL. HeatWave AutoML provides a simple SQL interface for training and using predictive machine learning models, which can be used by novice and experienced ML practitioners alike. With HeatWave AutoML, you can train a model with a single call to an SQL routine. Similarly, you can generate predictions with a single `CALL` or `SELECT` statement which can be easily integrated with your applications.

  With HeatWave AutoML, data and models never leave the MySQL Database Service, saving you time and effort while keeping your data and models secure. HeatWave AutoML is optimized for HeatWave shapes and scaling, and all HeatWave AutoML processing is performed on the HeatWave Cluster. ML computation is distributed among HeatWave nodes, taking advantage of HeatWave's scalability and massively parallel processing capabilities. For more information about HeatWave's machine learning capabilities, see HeatWave AutoML. (WL #14661, WL #14836, WL #15014)

Functionality Added or Changed

- HeatWave now compresses data as it is loaded, which permits HeatWave nodes to store more data. More data per node reduces costs by minimizing the size of the HeatWave Cluster required to store your data. Data compression is enabled by default but can be disabled at runtime using the `rapid_compression` session variable. For more information, see Data Compression. (WL #14868)

Changes in HeatWave 8.0.28-u1 (2022-02-15, General Availability)

Functionality Added or Changed

- HeatWave now supports up to 1017 columns for base relations (tables as loaded into HeatWave), and up to 1800 columns for intermediate relations (intermediate tables used during query processing). The maximum column width for base relations and intermediate relations was increased to 65532 bytes. See Column Limits. (WL #14918)

Changes in HeatWave 8.0.27-u3 (2021-12-15, General Availability)

Functionality Added or Changed

- Support was added for the `CONVERT_TZ()` and `LAST_DAY()` functions, which are used to manipulate temporal values. (WL #14768)

- The HeatWave Cluster recovery process was optimized to avoid applying a large volume of changelogs during recovery. Snapshots are now taken when the volume of changelogs and the time required to apply those changes exceed specific thresholds, and recovery from Object Storage is performed using those snapshots. (WL #14615)

- HeatWave now supports automatic data reload when the HeatWave Cluster is restarted. Previously, when a HeatWave Cluster was stopped by a stop or restart action, data had to be reloaded manually after the cluster was restarted. Now, when starting or restarting a HeatWave Cluster, data that was previously loaded is reloaded automatically. Data changes that occur on the DB System while the HeatWave Cluster is offline are included in the reloaded data.

  Automatic data reload does not occur if the HeatWave Cluster was stopped as a result of a stop or restart action performed on the DB System to which the HeatWave Cluster is attached. In this case, data loaded in the HeatWave Cluster must be reloaded manually after the HeatWave Cluster is restarted. (WL #14729)

Changes in HeatWave 8.0.27-u2 (2021-12-07, General Availability)

Functionality Added or Changed

- HeatWave now supports querying views. See Using Views. (WL #13568)
- Bloom filter join optimizations were introduced. For HeatWave queries that join large and small relations, bloom filters reduce the amount of data processed by early filtering and the amount of memory used during query processing. (WL #14752)
- The `rpd_query_stats` table, which stores HeatWave query history (compilation and execution statistics), now stores data for the last 1000 executed queries. Previously, data was stored for the last 200 queries.

  The following columns were added to the `performance_schema.rpd_tables` table:

  - `SIZE_BYTES`: The amount of data loaded per table, in bytes.
  - `QUERY_COUNT`: The number of queries that referenced the table.
  - `LAST_QUERIED`: The timestamp of the last query that referenced the table.
  - `LOAD_END_TIMESTAMP`: The load completion timestamp for the table.

  The following column was added to the `performance_schema.rpd_columns` table:

  - `DICT_SIZE_BTYES`: The dictionary size per column, in bytes.

  The following column was added to the `performance_schema.rpd_nodes` table:

  - `BASEREL_MEMORY_USAGE`: The base relation memory footprint per node.

  The `rapid_query_stats` and `rpd_exec_stats` tables are now synchronized. If a query record is removed from the `rapid_query_stats` table, it is also removed from the `rpd_exec_stats` table. (WL #14759)

Changes in HeatWave 8.0.26-u2 (2021-09-21, General Availability)

Functionality Added or Changed

- The following function support was added:
  - `YEARWEEK(`*`date`*`)`, `YEARWEEK(`*`date,mode`*`)`
  - The mode argument for the two-argument form of the `WEEK()` function: `WEEK(`*`date[,mode]`*`)`
  - `MAKEDATE()`
  - “Zero” handling for dates such as '2001-11-00' was implemented for `WEEK()`, `YEARWEEK()`, and `MAKEDATE()` functions.

  - `CAST()` of `FLOAT` and `DOUBLE` values to `DECIMAL`

  (Bug #33163625, Bug #33138534, WL #14714)

- The new `hw_data_scanned` global status variable tracks the total cumulative megabytes scanned by successfully executed HeatWave queries.

  The number of megabytes scanned by an individual HeatWave query can be obtained by querying the `performance_schema.rpd_query_stats` table.

  An estimated number of megabytes scanned by an individual query can be obtained by running the query with `EXPLAIN` and querying the `performance_schema.rpd_query_stats` table.

  For more information, see Scanned Data Monitoring. (WL #14738)

Changes in HeatWave 8.0.26-u1 (2021-08-10, General Availability)

- HeatWave Network Layer
- HeatWave Data Management Layer
- Functionality Added or Changed

HeatWave Network Layer

- HeatWave network layer optimizations have improved scalability and network performance. (WL #14513)

HeatWave Data Management Layer

- Data loaded into HeatWave, including propagated changes, are now persisted to OCI Object Storage for recovery in case of a HeatWave node or cluster failure. Previously, data was recovered from the MySQL DB System. Loading data from OCI Object Storage is faster because data does not need to be converted to the HeatWave storage format, as is required when loading data from the MySQL DB System. If data recovery from OCI Object Storage fails, HeatWave falls back to recovering data from the MySQL DB System. Data removed from HeatWave when a table is unloaded is removed from OCI Object Storage in a background operation. For related information, see HeatWave Cluster Failure and Recovery. (WL #14478, WL #14046, WL #14541)

Functionality Added or Changed

- HeatWave now supports `COUNT(NULL)`, except in cases where it is used as an input argument for non-aggregate operators. (Bug #33005146)
- Full support was added for the `DISTINCT` modifier. Previously, multiple instances of (`DISTINCT` *`value`*) expressions in a query were only permitted if the same *`value`* was specified. (Bug #32865043, Bug #33007714, WL #14574)
- HeatWave now supports the `WITH ROLLUP` modifier in `GROUP BY` clauses. (WL #14533)
- HeatWave now supports window functions. For optimal performance, window functions in HeatWave utilize a massively parallel, partitioning-based algorithm. For more information, see Window Functions. (WL #14674)

Changes in HeatWave 8.0.26 (2021-07-23, General Availability)

- Advisor
- Auto Parallel Load
- Auto Scheduling

- Functionality Added or Changed

Advisor

- The new HeatWave Advisor provides string column encoding and data placement key recommendations based on machine learning models, data analysis, and HeatWave query history. Implementing HeatWave Advisor recommendations can improve query performance and reduce the amount of memory required on HeatWave nodes.

  The HeatWave Advisor also provides a Query Insights feature, which provides runtimes for successfully executed queries, and runtime estimates for `EXPLAIN` queries, queries canceled using `Ctrl+C`, and queries that fail due to out of memory errors. Runtime data is useful for query optimization, troubleshooting, and estimating the cost of running a particular query or workload.

  The HeatWave Advisor is implemented as a stored procedure named `heatwave_advisor`, which resides in the MySQL `sys` schema. Running Advisor involves issuing a `CALL` statement for the stored procedure with optional arguments.

  ```
  CALL sys.heatwave_advisor (options);
  ```

  For more information about the HeatWave Advisor, see Workload Optimization for OLAP. (WL #14328, WL #14510, WL #14431, WL #14328, WL #14651)

Auto Parallel Load

- The new HeatWave Auto Parallel Load utility automates the process of preparing and loading tables into HeatWave and loads data using an optimized number of parallel load threads.

  The HeatWave Auto Parallel Load utility is implemented as a stored procedure named `heatwave_load`, which resides in the MySQL `sys` schema. Running Auto Parallel Load involves issuing a `CALL` statement for the stored procedure, which takes a list of schemas and options as arguments.

  ```
  CALL sys.heatwave_load (db_list,[options]);
  ```

  For more information about the HeatWave Auto Parallel Load utility, see Loading Data Using Auto Parallel Load. (WL #14149)

Auto Scheduling

- The HeatWave query scheduling algorithm was improved. The revised algorithm prioritizes queries based on estimated cost and wait time in the queue, which enables dynamic, workload-aware query prioritization. Previously, queries were prioritized using a static cost-based prioritization model. (WL #14608)

Functionality Added or Changed

- `DATE_ADD()` and `DATE_SUB()` functions now support precision `INTERVAL` values (`DECIMAL`, `DOUBLE`, and `FLOAT`). (Bug #32725985, Bug #32438123)
- Support was added for multiple instances of `COUNT(DISTINT)` in a query. (Bug #32422984)
- Query compilation and processing was improved to permit combining aggregate operators into a single task in the physical query plan, which avoids fully materializing intermediate result sets. This enhancement reduces memory allocation and deallocation operations, memory usage, and execution time for affected queries. (WL #14614)
- The cost model that estimates HeatWave query runtimes can now use statistics from previously executed queries, which improves the accuracy of query runtime estimates. (WL #14546)
- HeatWave now supports `CREATE TABLE ... SELECT` statements where the `SELECT` query is offloaded to HeatWave and the table is created on the MySQL Database Service instance. This

feature improves `CREATE TABLE ... SELECT` performance in cases where the `SELECT` portion of the statement is a long running, complex query. For more information, see CREATE TABLE ... SELECT Statements. (WL #14516)

- Support was added for `REGEXP_REPLACE()` and `REGEXP_SUBSTR()` regular expression functions, and error messaging was improved for `REGEXP()` function syntax mismatches, expression errors, and input argument errors. (WL #14641)

Changes in HeatWave 8.0.25 (2021-05-11, General Availability)

Functionality Added or Changed

- Support was added for `CAST()` of `ENUM` column values to `CHAR` or `VARCHAR` where the `ENUM` value is cast to a `FLOAT` value, as in the following example:

```
SELECT CAST(CAST(enum_col AS FLOAT) AS CHAR(3)) FROM tbl_name;
```

(Bug #32618454)

- Support was added for `SELECT DISTINCT` queries that order the result set by a column that is not defined in the `SELECT` list. For example, the following query can now be offloaded to HeatWave for execution:

```
SELECT DISTINCT a FROM t1 ORDER BY c DESC;
```

(Bug #32583856)

- Query plan statistics are now collected and stored in a statistics cache when a query is executed in HeatWave. When a new query shares query execution plan nodes with previously executed queries, the actual statistics collected from previously executed queries are used instead of estimated statistics, which improves query execution plans, cost estimations, execution times, and memory efficiency.

  The statistics cache is an LRU structure. When cache capacity is reached, the least recently used entries are evicted from the cache as new entries are added. The maximum number of entries permitted in the statistics cache is defined by the `rapid_stats_cache_max_entries` setting. The number of entries permitted by default is 65536, which is enough to store statistics for 4000 to 5000 unique queries of medium complexity. (WL #14503)

- Support was added for:

  - `CAST() AS YEAR`. Both variable-length and dictionary-encoded string columns values are supported.
  - The `FORMAT()` function. Variable-length-encoded string columns are not supported.

  (WL #14511)

Changes in HeatWave 8.0.24 (2021-04-20, General Availability)

Functionality Added or Changed

- Comparison of different temporal type values is now supported. For example, a query that compares `DATE` values to `TIMESTAMP` values can now be offloaded to HeatWave. (Bug #32420986)

- Range operators on `VARLEN`-encoded string columns are now supported. For example, the following query, where `L_LINESTATUS` is a `VARLEN`-encoded string column, can now be offloaded to HeatWave:

```
SELECT COUNT(*) FROM lineitem WHERE L_LINESTATUS  >= 1 and L_LINESTATUS <= 10;
```

(Bug #31721399)

- HeatWave now supports `INSERT ... SELECT` statements where the `SELECT` query is offloaded to HeatWave and the result set is inserted into a table on the MySQL Database Service instance. This feature improves `INSERT ... SELECT` performance in cases where the `SELECT` portion of the statement is a long running, complex query. For more information, see INSERT ... SELECT Statements. (WL #14299)
- `VARLEN`-encoded columns are now supported as data placement keys. For information about the data placement feature, see Defining Data Placement Keys. (WL #14491)
- Failure handling was improved for queries involving unsupported internal data types. Such queries now exit with an error indicating that the internal data type of the query is not supported. (WL #14483)

Changes in HeatWave 8.0.23-u2 (2021-03-15, General Availability)

Functionality Added or Changed

- Support was added for the following aggregate functions:
  - `STD()`
  - `STDDEV()`
  - `STDDEV_POP()`
  - `STDDEV_SAMP()`
  - `VAR_POP()`
  - `VAR_SAMP()`
  - `VARIANCE()`

  See Aggregate Functions. (WL #14479)
- HeatWave now uses a priority-based scheduling mechanism based on query cost estimates to schedule queries for execution. Previously, queries were executed in the order of arrival. The scheduling mechanism prioritizes short running queries over long running queries to reduce overall query execution wait times. For more information, see Auto Scheduling. (WL #14423)

Changes in HeatWave 8.0.23-u1 (2021-02-09, General Availability)

Functionality Added or Changed

- String column encoding support was added for `TEXT`-type columns. See Encoding String Columns. (WL #14430)
- `UNION` and `UNION ALL` support was extended. The clauses are now supported at any location in a query that is permitted by MySQL. (WL #14455)
- The following temporal functions are now supported:
  - `TO_SECONDS()`
  - `UNIX_TIMESTAMP()`
  - `FROM_UNIXTIME()`
  - `TIME_TO_SEC()`

  See Temporal Functions.

The following temporal functions are now supported with `VARLEN`-encoded columns:

- `TO_DAYS()`
- `DAYOFYEAR()`
- `QUARTER()`
- `TO_SECONDS()`

See Temporal Functions.

The following string functions are now supported with `VARLEN`-encoded columns:

- `ORD()`
- `ASCII()`

See String Functions and Operators.

`SET timezone = timezone` with the *`timezone`* value specified as an offset from UTC in the form of *`[H]H:MM`* and prefixed with a `+` or `-` is now supported with the `UNIX_TIMESTAMP()` and `FROM_UNIXTIME()` functions. (WL #14345)

- Offset is now supported with the `LIMIT` clause:

```
SELECT * FROM tbl LIMIT offset,row_count;
```

The PostgreSQL syntax is also supported:

```
SELECT * FROM tbl LIMIT row_count OFFSET offset;
```

(WL #14341)

- New Performance Schema tables provide access to query and execution statistics:
  - `performance_schema.rpd_exec_stats`
  - `performance_schema.rpd_query_stats`

  Changes to HeatWave Performance Schema tables:
  - An `NDV` (Number of Distinct Values) column was added to the `performance_schema.rpd_columns` table.
  - A `ROWS` column that shows the total number of rows in a table was added to the `performance_schema.rpd_tables` table.
  - A `MEMORY_USAGE` column that shows node memory usage was added to the `performance_schema.rpd_columns` table.
  - The `performance_schema.rpd_nodes DRAM` column was renamed to `MEMORY_TOTAL`. The `MEMORY_TOTAL` column shows the total memory allocated to a HeatWave node.

  See HeatWave Performance Schema Tables. (WL #14386)

Index

A

ABS(), 20
Adaptive Query Execution, 15

Advisor, 20, 25
Aggregate functions, 11, 12, 28
ALTER TABLE, 19
ASCII(), 28
Auto Data Compression, 11, 12
Auto Encoding, 20
Auto Parallel Load, 5, 6, 10, 11, 12, 14, 15, 25
Auto Unload, 10, 17
AutoML, 3, 4, 6, 7, 11, 12, 14, 15, 15, 16, 17, 17, 17, 18, 19, 20, 23
Autopilot, 5, 7
AVG(), 13
Avro file format, 12, 15

B

BIN(), 13
BIT_LENGTH(), 13
Bloom filter optimization, 24
Bulk load, 7, 10, 14

C

CAST(), 13, 24, 27
Change propagation, 16, 20, 23
Column limits, 23
Comparison functions , 11, 20
Comparisons, 20
COMPRESS(), 18
Compression functions, 18
CONVERT_TZ(), 23
COUNT(DISTINCT), 25
COUNT(NULL), 25
CRC32(), 18
CREATE TABLE ... SELECT, 25
CSV file format, 16
CUBE, 12

D

Data compression, 20, 23
Data placement, 27
Data types, 27
DATE, 13, 15
DATETIME, 13, 15
DATE_ADD(), 25
DATE_SUB(), 25
DAYOFYEAR(), 28
Dictionary encoding, 19
DISTINCT, 25

E

ELT(), 13
Encryption functions, 18
Error reporting, 6, 11
EXCEPT, 16
EXPORT_SET(), 13

F

FIELD(), 13

FORMAT(), 27
FROM_DAYS(), 20
FROM_UNIXTIME(), 28

G

GenAI, 3, 5, 6, 7
gen_range(), 20
gen_rnd_email(), 20
gen_rnd_pan(), 20
gen_rnd_ssn(), 20
gen_rnd_us_phone(), 20
GET_FORMAT(), 16
GREATEST(), 11, 20
GROUP BY, 12, 25
GROUP_CONCAT(), 20
Guided Load, 11, 13

H

Heap segment size, 20
HeatWave AutoML, 3, 4, 6, 7, 11, 12, 14, 15, 15, 16, 17, 17, 17, 18, 19, 20, 23
HeatWave Autopilot;, 5, 7
HeatWave GenAI, 3, 5, 6, 7
HeatWave Lakehouse, 3, 4, 5, 6, 7, 10, 11, 11, 12, 12, 14, 15, 16
HeatWave MySQL, 5, 6, 7
heatwave_advisor_report table, 18
heatwave_autopilot_report table, 18
heatwave_load_report table, 18
High Availability, 11
HLL(), 11, 12
HyperLogLog, 11, 12

I

IN(), 20
INSERT ... SELECT, 27
INTERSECT, 16

J

JavaScript, 5
JSON ->, 12
JSON ->>, 12
JSON column path, 12
JSON data type, 11
JSON file format, 10
JSON functions, 12
JSON inline path, 12
JSON_ARRAY(), 12
JSON_DEPTH(), 12
JSON_EXTRACT(), 12
JSON_LENGTH(), 12
JSON_OBJECT(), 12
JSON_UNQUOTE(), 12

L

Lakehouse, 3, 4, 5, 6, 7, 10, 11, 11, 12, 12, 14, 15, 16
Lakehouse Auto Parallel Load, 10, 11, 12, 14, 15

Lakehouse Incremental Load, 7
LAST_DAY, 23
LEAST(), 11, 20
LIMIT, 28
Loading data, 20
LOG10(), 18
LOG2(), 18

M

Machine learning, 3, 4, 6, 7, 11, 12, 14, 15, 15, 16, 17, 17, 17, 18, 19, 20, 23
MAKEDATE(), 24
MAKETIME(), 16
MAKE_SET(), 13
mask_inner(), 20
mask_outer(), 20
mask_pan(), 20
mask_pan_relaxed(), 20
mask_ssn(), 20
Mathematical functions, 17, 18, 20
MD5(), 18
Memory usage, 16
MID, 15
MySQL, 5, 6, 7
MySQL additional functionality, 7, 10, 11, 12, 14
MySQL InnoDB, 7, 10, 14

N

Networking, 25

O

Object Storage, 23
OFFSET, 28
Optimizer, 19
ORD(), 28
Outbound Replication, 11

P

Parquet file format, 15, 16
Partitioning, 20
Partitions, 5
Performance Schema, 17, 17, 20, 24, 28
PERIOD_ADD(), 16
PERIOD_DIFF(), 16
Planned shutdown, 23
Point-in-time-recovery, 11
POWER(), 20

Q

QUALIFY, 12
QUARTER(), 28
Queries, 24, 27
Query cost model, 25
Query optimization, 16
Query plans, 19
Query scheduling, 25, 28
Query support, 12

R

RAND(), 18
RANDOM_BYTES(), 18
rapid_stats_cache_max_entries, 27
Read Replication, 11
Recovery, 23, 25
REGEXP_INSTR, 15
REGEXP_REPLACE(), 25
REGEXP_SUBSTR(), 25
Regular expression functions, 25
Reload tables, 11
RENAME TABLE, 19
rpd_cloumns table, 28
rpd_column_id table, 20
rpd_exec_stats table, 28
rpd_nodes table, 28
rpd_query_stats table, 17, 24, 28
rpd_tables table, 17

S

Scalability, 25
Schema inference, 11, 12, 14
SEC_TO_TIME(), 16
SELECT, 11, 12
SELECT DISTINCT, 27
SET timezone, 28
SHA(), 18
SHA1(), 18
SHA2(), 18
SIGN(), 20
SOUNDEX, 15
SPACE, 15
Statistics, 27
STD(), 12, 13, 28
STDDEV(), 12, 13, 28
STDDEV_POP(), 12, 13, 28
STDDEV_SAMP(), 12, 13, 28
String column encoding, 27
String functions, 11, 13, 20, 28
SUBTIME(), 16
SUM(), 13
SYSDATE(), 16

T

TABLESAMPLE, 11
Temporal data types, 13, 20
Temporal functions, 10, 16, 20, 23, 24, 25, 28
Temporal type comparison, 27
TEXT, 28
TIME, 13, 15
TIMEDIFF(), 16
Timeseries forecasting, 19
TIMESTAMP, 13, 15
Timezone, 15
Timezone functions, 10, 28
TIME_TO_SEC(), 28
TO_DAYS(), 28

TO_SECONDS(), 28
TRUNCATE TABLE, 19

U

UNCOMPRESS(), 18
UNCOMPRESSED_LENGTH(), 18
UNION, 28
UNION ALL, 28
UNIX_TIMESTAMP(), 28
Unload tables, 11

V

Variable-length encoding, 27
VARIANCE(), 12, 13, 28
VARLEN encoding, 28
VAR_POP(), 12, 13, 28
VAR_SAMP(), 12, 13, 28
Views, 24

W

WEEK(), 24
WEIGHT_STRING, 15
Window functions, 12, 25
WITH ROLLUP, 25

Y

YEAR, 13
YEARWEEK(), 24

Z

Zone maps, 5, 19