

**Note**

It is also possible in MySQL 8.0 to use `UNIX_TIMESTAMP(timestamp_column)` as a partitioning expression for tables that are partitioned by `LIST`. However, it is usually not practical to do so.

- Partition the table by `RANGE COLUMNS`, using a `DATE` or `DATETIME` column as the partitioning column. For example, the `members` table could be defined using the `joined` column directly, as shown here:

```
CREATE TABLE members (
  firstname VARCHAR(25) NOT NULL,
  lastname VARCHAR(25) NOT NULL,
  username VARCHAR(16) NOT NULL,
  email VARCHAR(35),
  joined DATE NOT NULL
)
PARTITION BY RANGE COLUMNS(joined) (
  PARTITION p0 VALUES LESS THAN ('1960-01-01'),
  PARTITION p1 VALUES LESS THAN ('1970-01-01'),
  PARTITION p2 VALUES LESS THAN ('1980-01-01'),
  PARTITION p3 VALUES LESS THAN ('1990-01-01'),
  PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

**Note**

The use of partitioning columns employing date or time types other than `DATE` or `DATETIME` is not supported with `RANGE COLUMNS`.

24.2.2 LIST Partitioning

List partitioning in MySQL is similar to range partitioning in many ways. As in partitioning by `RANGE`, each partition must be explicitly defined. The chief difference between the two types of partitioning is that, in list partitioning, each partition is defined and selected based on the membership of a column value in one of a set of value lists, rather than in one of a set of contiguous ranges of values. This is done by using `PARTITION BY LIST(expr)` where `expr` is a column value or an expression based on a column value and returning an integer value, and then defining each partition by means of a `VALUES IN (value_list)`, where `value_list` is a comma-separated list of integers.

**Note**

In MySQL 8.0, it is possible to match against only a list of integers (and possibly `NULL`—see [Section 24.2.7, “How MySQL Partitioning Handles NULL”](#)) when partitioning by `LIST`.

However, other column types may be used in value lists when employing `LIST COLUMN` partitioning, which is described later in this section.

Unlike the case with partitions defined by range, list partitions do not need to be declared in any particular order. For more detailed syntactical information, see [Section 13.1.20, “CREATE TABLE Statement”](#).

For the examples that follow, we assume that the basic definition of the table to be partitioned is provided by the `CREATE TABLE` statement shown here:

```
CREATE TABLE employees (
  id INT NOT NULL,
  fname VARCHAR(30),
  lname VARCHAR(30),
  hired DATE NOT NULL DEFAULT '1970-01-01',
  separated DATE NOT NULL DEFAULT '9999-12-31',
```