

```

| NULL | mothra |
+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM t2;
+-----+-----+
| id   | name   |
+-----+-----+
| NULL | mothra |
+-----+-----+
1 row in set (0.00 sec)

```

You can see which partitions are used to store the inserted rows by rerunning the previous query against `INFORMATION_SCHEMA.PARTITIONS` and inspecting the output:

```

mysql> SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
> FROM INFORMATION_SCHEMA.PARTITIONS
> WHERE TABLE_SCHEMA = 'p' AND TABLE_NAME LIKE 't_';
+-----+-----+-----+-----+-----+
| TABLE_NAME | PARTITION_NAME | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+-----+-----+-----+-----+-----+
| t1          | p0             | 1           | 20             | 20           |
| t1          | p1             | 0           | 0              | 0            |
| t1          | p2             | 0           | 0              | 0            |
| t2          | p0             | 1           | 20             | 20           |
| t2          | p1             | 0           | 0              | 0            |
| t2          | p2             | 0           | 0              | 0            |
| t2          | p3             | 0           | 0              | 0            |
+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)

```

You can also demonstrate that these rows were stored in the lowest-numbered partition of each table by dropping these partitions, and then re-running the `SELECT` statements:

```

mysql> ALTER TABLE t1 DROP PARTITION p0;
Query OK, 0 rows affected (0.16 sec)

mysql> ALTER TABLE t2 DROP PARTITION p0;
Query OK, 0 rows affected (0.16 sec)

mysql> SELECT * FROM t1;
Empty set (0.00 sec)

mysql> SELECT * FROM t2;
Empty set (0.00 sec)

```

(For more information on `ALTER TABLE ... DROP PARTITION`, see [Section 13.1.9, “ALTER TABLE Statement”](#).)

`NULL` is also treated in this way for partitioning expressions that use SQL functions. Suppose that we define a table using a `CREATE TABLE` statement such as this one:

```

CREATE TABLE tndate (
  id INT,
  dt DATE
)
PARTITION BY RANGE( YEAR(dt) ) (
  PARTITION p0 VALUES LESS THAN (1990),
  PARTITION p1 VALUES LESS THAN (2000),
  PARTITION p2 VALUES LESS THAN MAXVALUE
);

```

As with other MySQL functions, `YEAR(NULL)` returns `NULL`. A row with a `dt` column value of `NULL` is treated as though the partitioning expression evaluated to a value less than any other value, and so is inserted into partition `p0`.