SYNTAX:

```
REPLACE [LOW_PRIORITY | DELAYED]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),... [RETURNING select_expr[,
select_expr ...]]
```

Or

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, … [RETURNING select_expr[, select_expr ...]]
```

Or

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT … [RETURNING select_expr[, select_expr ...]]
```

DESCRIPTION:

From MariaDB 10.5 the result set of replaced rows can be returned to the client by using the following syntax:

```
REPLACE INTO table_name … RETURNING  select_expr [, select_expr2 …]]
```

It is possible to return all the datatypes

This returns the listed columns for all the rows that are replaced, or more generally, the specified `SELECT` expression. SQL expressions which can be calculated can be used in the select expression for `RETURNING` clause.

`REPLACE...RETURNING` is useful as it eliminates the need to write an additional query.
Example:

```
CREATE TABLE t1(id1 INT PRIMARY KEY);
REPLACE INTO t1 VALUES (1),(2),(3);
SELECT * FROM t1 WHERE id1 IN (1,2,3);
```

The above `INSERT` and `SELECT` statement can be replaced with:

```
REPLACE INTO t1 VALUES (1),(2),(3) RETURNING *;
```

HOW TO USE REPLACE...RETURNING

The expressions can be comprised of bitwise, logical or arithmetic operators, hence virtual columns are allowed. `RETURNING` * works, `AS` keyword is allowed, hence alias can be used.

```
MariaDB [test]> CREATE TABLE t1(id1 INT PRIMARY KEY, animal1
VARCHAR(15));
Query OK, 0 rows affected (0.390 sec)

MariaDB [test]> CREATE TABLE t2(id2 INT PRIMARY KEY, animal2
VARCHAR(15));
Query OK, 0 rows affected (0.463 sec)
```

1)Simple select expression

```
MariaDB [test]> INSERT INTO t2 VALUES
(1,'Lion'),(2,'Tiger'),(3,'Bear'),(4,'Deer');
Query OK, 4 rows affected (0.083 sec)
Records: 4  Duplicates: 0  Warnings: 0

MariaDB [test]> REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog')
RETURNING id2, id2+id2 as Total ,id2|id2, id2&&id2;
+-----+-------+---------+----------+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-------+---------+----------+
|   1 |     2 |       1 |        1 |
|   2 |     4 |       2 |        1 |
+-----+-------+---------+----------+
2 rows in set (0.065 sec)
```

2)Prepared statements and functions

```
MariaDB [test]> DELIMITER |
MariaDB [test]> CREATE FUNCTION f(arg INT) RETURNS INT
    -> BEGIN
    ->   RETURN (SELECT arg+arg);
    -> END|
Query OK, 0 rows affected (0.173 sec)
MariaDB [test]> DELIMITER ;
MariaDB [test]> PREPARE stmt FROM "REPLACE INTO t2 SET id2=3,
animal2='Fox' RETURNING f2(id2), UPPER(animal2)";
Query OK, 0 rows affected (0.002 sec)
Statement prepared
```

```
MariaDB [test]> EXECUTE stmt;
+---------+----------------+
| f2(id2) | UPPER(animal2) |
+---------+----------------+
|       6 | FOX            |
+---------+----------------+
1 row in set (0.094 sec)
```

3) Complex Subquery

```
MariaDB [test]> REPLACE INTO t1 SET id1=1, animal1='Fox' RETURNING
(SELECT GROUP_CONCAT(animal2) FROM t2 GROUP BY id2 HAVING id2=id2+1)
AS new_id;
+--------+
| new_id |
+--------+
| NULL   |
+--------+
1 row in set (0.003 sec)
```

```
MariaDB [test]> REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT
id2 FROM t2 WHERE id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+--------+
| new_id |
+--------+
|      1 |
|      1 |
|      1 |
|      1 |
+--------+
4 rows in set (0.005 sec)
```

4)Special case REPLACE..SELECT..RETURNING

```
MariaDB [test]> REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT
id1+id2 FROM t1 WHERE id1=1);
+---------------------------------------+
| (SELECT id1+id2 FROM t1 WHERE id1=1) |
+---------------------------------------+
|                                     2 |
|                                     3 |
|                                     4 |
|                                     5 |
+---------------------------------------+
4 rows in set (0.009 sec)

MariaDB [test]> REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT
id1+1 FROM t2 WHERE id1=0) as id_2;
+------+
| id_2 |
+------+
| NULL |
| NULL |
| NULL |
| NULL |
+------+
4 rows in set (0.006 sec)
```

Along with these, string function, date-time function,numeric functions, control flow function and secondary functions can also be used in the `RETURNING` clause.

WHAT CANNOT BE USED:

1) Subqueries in `RETURNING` clause that return more than one row or column.
2) Aggregate functions cannot be used in `RETURNING` clause:
   Since aggregate functions work on a set of values and if the purpose is to get the row count, `ROW_COUNT()` with `SELECT` can be used or it can be used in `REPLACE...SELECT...RETURNING` if the table in `RETURNING` clause is not the same as the `REPLACE` table.

Example:

```
REPLACE INTO t2 SELECT * FROM t1 RETURNING (SELECT COUNT(id1) FROM
t1);
```