MySQL ON DELETE CASCADE

ON DELETE CASCADE clause in MySQL is used to automatically **remove** the matching records from the child table when we delete the rows from the parent table. It is a kind of referential action related to the **foreign key**.

Suppose we have created two tables with a FOREIGN KEY in a foreign key relationship, making both tables a parent and child. Next, we define an ON DELETE CASCADE clause for one FOREIGN KEY that must be set for the other to succeed in the cascading operations. If the ON DELETE CASCADE is defined for one FOREIGN KEY clause only, then cascading operations will throw an error.

MySQL ON DELETE CASCADE Example

Let us understand how we can use the ON DELETE CASCADE clause in the MySQL table. First, we are going to create two tables named **Employee and Payment**. Both tables are related through a foreign key with on delete cascade operation. Here, an Employee is the **parent table**, and Payment is the **child table**. The following scripts create both tables along with their records.

Table: Employee

The following statement creates a table Employee:

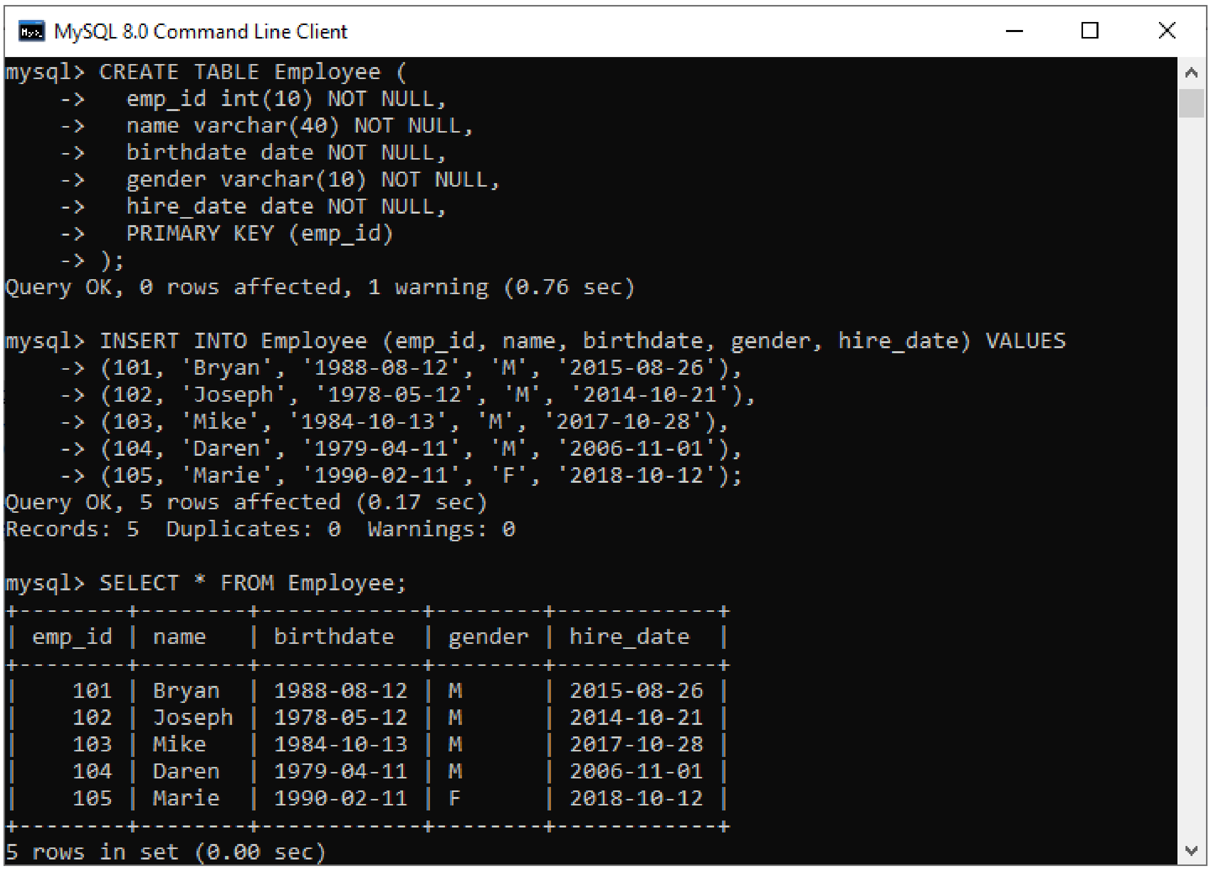
```
CREATE TABLE Employee (
  emp_id int(10) NOT NULL,
  name varchar(40) NOT NULL,
  birthdate date NOT NULL,
  gender varchar(10) NOT NULL,
  hire_date date NOT NULL,
  PRIMARY KEY (emp_id)
);
```

Next, execute the insert query to fill the records.

```
INSERT INTO Employee (emp_id, name, birthdate, gender, hire_date) VALUES
(101, 'Bryan', '1988-08-12', 'M', '2015-08-26'),
(102, 'Joseph', '1978-05-12', 'M', '2014-10-21'),
(103, 'Mike', '1984-10-13', 'M', '2017-10-28'),
(104, 'Daren', '1979-04-11', 'M', '2006-11-01'),
```

6. (105, 'Marie', '1990-02-11', 'F', '2018-10-12');

Execute the SELECT query to verify the data into a table, which can be shown below:



```
mysql> CREATE TABLE Employee (
    ->   emp_id int(10) NOT NULL,
    ->   name varchar(40) NOT NULL,
    ->   birthdate date NOT NULL,
    ->   gender varchar(10) NOT NULL,
    ->   hire_date date NOT NULL,
    ->   PRIMARY KEY (emp_id)
    -> );
Query OK, 0 rows affected, 1 warning (0.76 sec)

mysql> INSERT INTO Employee (emp_id, name, birthdate, gender, hire_date) VALUES
    -> (101, 'Bryan', '1988-08-12', 'M', '2015-08-26'),
    -> (102, 'Joseph', '1978-05-12', 'M', '2014-10-21'),
    -> (103, 'Mike', '1984-10-13', 'M', '2017-10-28'),
    -> (104, 'Daren', '1979-04-11', 'M', '2006-11-01'),
    -> (105, 'Marie', '1990-02-11', 'F', '2018-10-12');
Query OK, 5 rows affected (0.17 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Employee;
+--------+--------+------------+--------+------------+
| emp_id | name   | birthdate  | gender | hire_date  |
+--------+--------+------------+--------+------------+
|    101 | Bryan  | 1988-08-12 | M      | 2015-08-26 |
|    102 | Joseph | 1978-05-12 | M      | 2014-10-21 |
|    103 | Mike   | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren  | 1979-04-11 | M      | 2006-11-01 |
|    105 | Marie  | 1990-02-11 | F      | 2018-10-12 |
+--------+--------+------------+--------+------------+
5 rows in set (0.00 sec)
```

Table: Payment

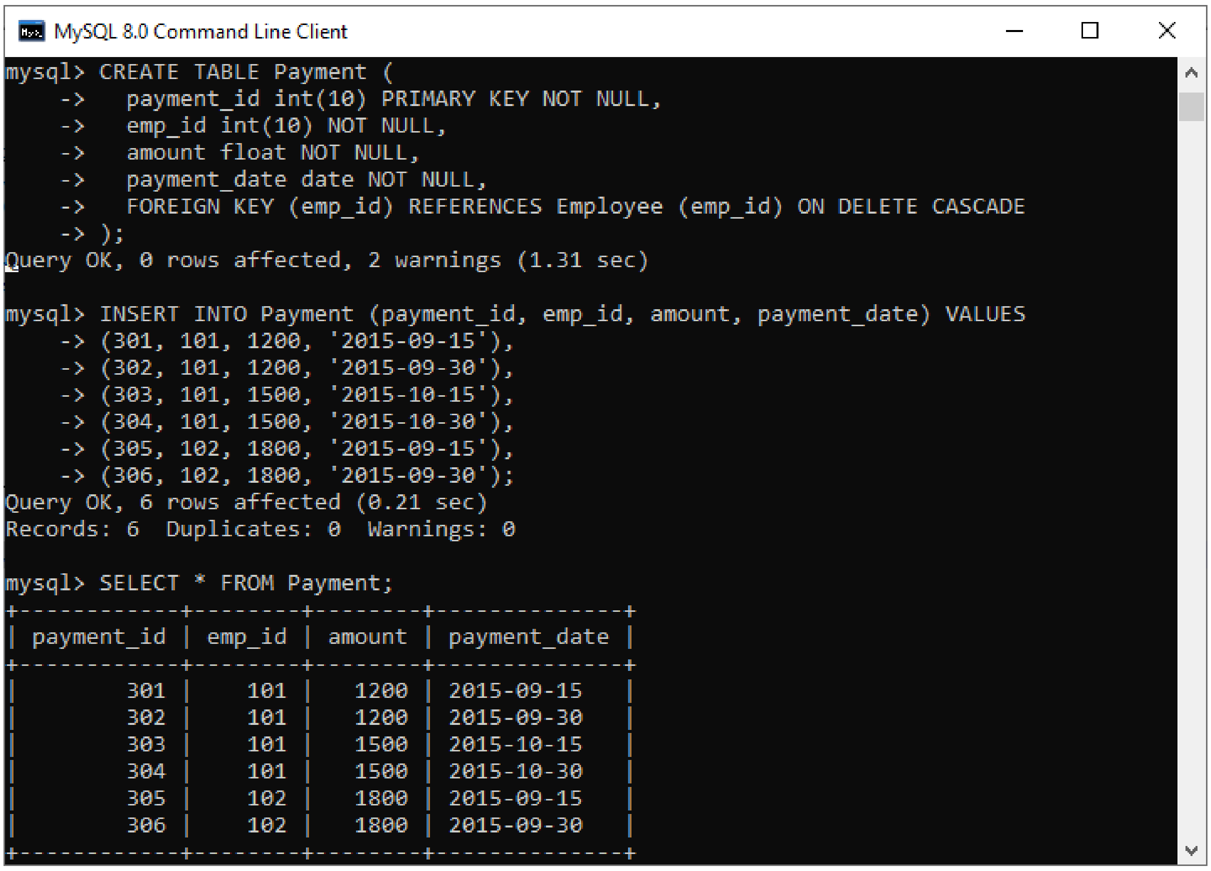
The below statement creates a table Payment:

```
1. CREATE TABLE Payment (
2.   payment_id int(10) PRIMARY KEY NOT NULL,
3.   emp_id int(10) NOT NULL,
4.   amount float NOT NULL,
5.   payment_date date NOT NULL,
6.   FOREIGN KEY (emp_id) REFERENCES Employee (emp_id) ON DELETE CASC
   ADE
7. );
```

Next, execute the insert statement to fill the records into a table.

```
1. INSERT INTO Payment (payment_id, emp_id, amount, payment_date) VALUES
2. (301, 101, 1200, '2015-09-15'),
3. (302, 101, 1200, '2015-09-30'),
4. (303, 101, 1500, '2015-10-15'),
5. (304, 101, 1500, '2015-10-30'),
6. (305, 102, 1800, '2015-09-15'),
7. (306, 102, 1800, '2015-09-30');
```

Execute the SELECT query to verify the data into a table, which can be shown below:

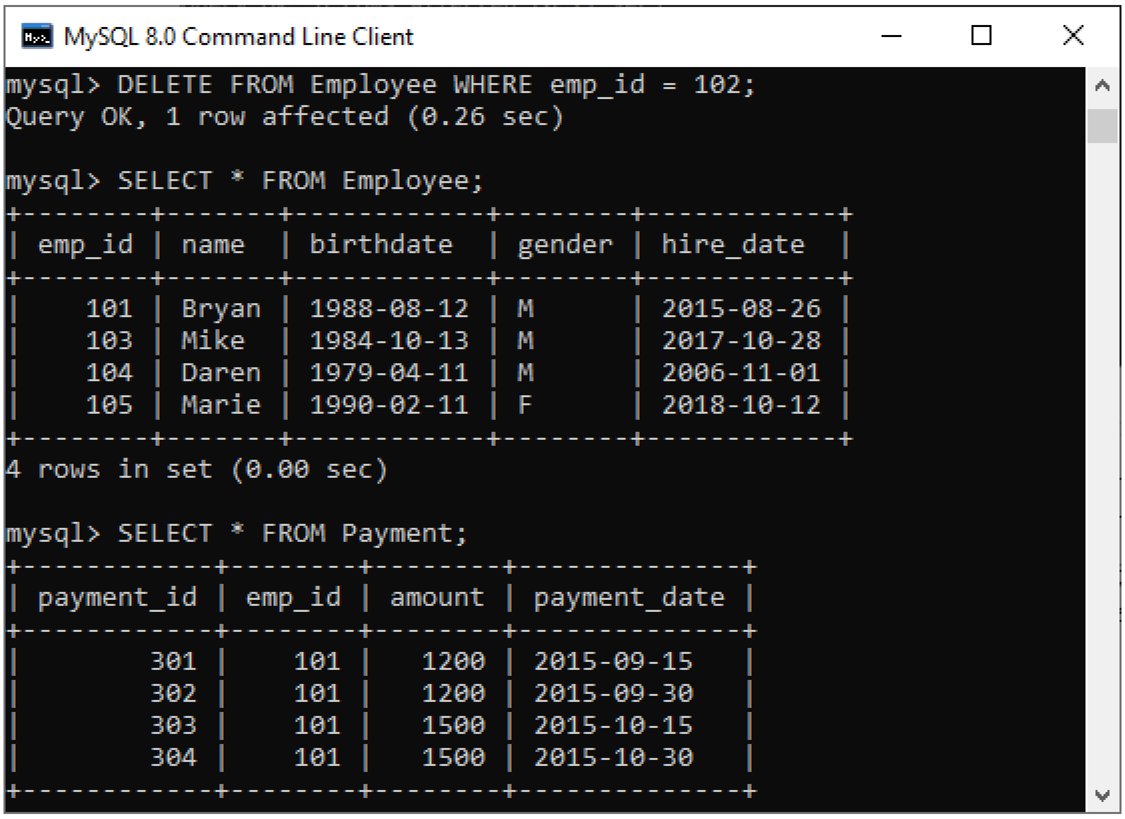


```
mysql> CREATE TABLE Payment (
    ->   payment_id int(10) PRIMARY KEY NOT NULL,
    ->   emp_id int(10) NOT NULL,
    ->   amount float NOT NULL,
    ->   payment_date date NOT NULL,
    ->   FOREIGN KEY (emp_id) REFERENCES Employee (emp_id) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected, 2 warnings (1.31 sec)

mysql> INSERT INTO Payment (payment_id, emp_id, amount, payment_date) VALUES
    -> (301, 101, 1200, '2015-09-15'),
    -> (302, 101, 1200, '2015-09-30'),
    -> (303, 101, 1500, '2015-10-15'),
    -> (304, 101, 1500, '2015-10-30'),
    -> (305, 102, 1800, '2015-09-15'),
    -> (306, 102, 1800, '2015-09-30');
Query OK, 6 rows affected (0.21 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Payment;
+------------+--------+--------+--------------+
| payment_id | emp_id | amount | payment_date |
+------------+--------+--------+--------------+
|        301 |    101 |   1200 | 2015-09-15   |
|        302 |    101 |   1200 | 2015-09-30   |
|        303 |    101 |   1500 | 2015-10-15   |
|        304 |    101 |   1500 | 2015-10-30   |
|        305 |    102 |   1800 | 2015-09-15   |
|        306 |    102 |   1800 | 2015-09-30   |
+------------+--------+--------+--------------+
```

Let us **delete** data from the parent table Employee. To do this, execute the following statement:

```
1. mysql> DELETE FROM Employee WHERE emp_id = 102;
```

The above statement will delete the employee records whose **emp_id = 102** and **referencing** data into the child table. We can verify the data using the SELECT statement that will give the following output:


MySQL 8.0 Command Line Client

```
mysql> DELETE FROM Employee WHERE emp_id = 102;
Query OK, 1 row affected (0.26 sec)

mysql> SELECT * FROM Employee;
+--------+-------+------------+--------+------------+
| emp_id | name  | birthdate  | gender | hire_date  |
+--------+-------+------------+--------+------------+
|    101 | Bryan | 1988-08-12 | M      | 2015-08-26 |
|    103 | Mike  | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren | 1979-04-11 | M      | 2006-11-01 |
|    105 | Marie | 1990-02-11 | F      | 2018-10-12 |
+--------+-------+------------+--------+------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Payment;
+------------+--------+--------+--------------+
| payment_id | emp_id | amount | payment_date |
+------------+--------+--------+--------------+
|        301 |    101 |   1200 | 2015-09-15   |
|        302 |    101 |   1200 | 2015-09-30   |
|        303 |    101 |   1500 | 2015-10-15   |
|        304 |    101 |   1500 | 2015-10-30   |
+------------+--------+--------+--------------+
```

In the above output, we can see that all the rows referencing to emp_id = 102 were automatically deleted from both tables.

How to find the affected table by ON DELETE CASCADE action?

Sometimes, before deleting records from the table, we want to know the affected table by the ON DELETE CASCADE referential action. We can find this information by querying from the referential_constraints in the information_schema database as follows:

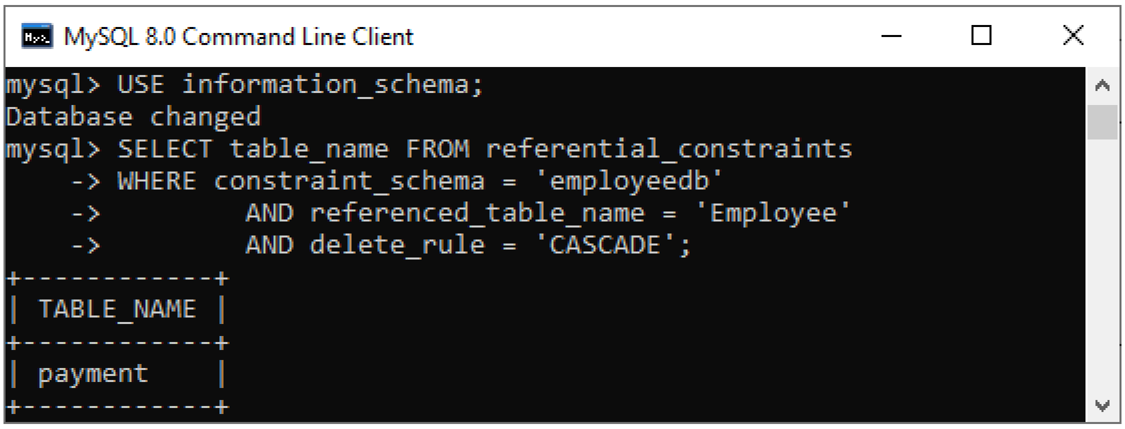
```
USE information_schema;

SELECT table_name FROM referential_constraints
WHERE constraint_schema = 'database_name'
      AND referenced_table_name = 'parent_table'
      AND delete_rule = 'CASCADE'
```

The below statement produces the result about the tables associated with the Employee table with the ON DELETE CASCADE rule in the **employeedb** database:

```
USE information_schema;

```

3. **SELECT** table_name **FROM** referential_constraints
4. **WHERE** constraint_schema = 'employeedb'
5.   AND referenced_table_name = 'Employee'
6.   AND delete_rule = 'CASCADE';

After executing the above command, we will get the output below:



MySQL 8.0 Command Line Client

```
mysql> USE information_schema;
Database changed
mysql> SELECT table_name FROM referential_constraints
    -> WHERE constraint_schema = 'employeedb'
    ->         AND referenced_table_name = 'Employee'
    ->         AND delete_rule = 'CASCADE';
+------------+
| TABLE_NAME |
+------------+
| payment    |
+------------+
```

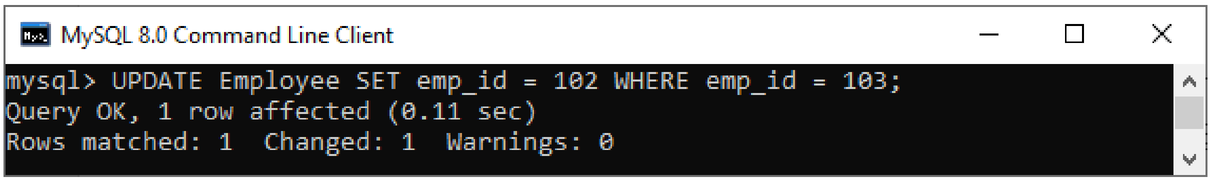
MySQL ON UPDATE CASCADE

ON UPDATE CASCADE clause in MySQL is used to **update** the matching records from the child table automatically when we update the rows in the parent table. The following example explains it more clearly.

First, we need to use the **ALTER TABLE** statement to add the ON UPDATE CASCADE clause in the table Payment as below:

1. **ALTER TABLE** Payment **ADD CONSTRAINT** `payment_fk`
2. **FOREIGN KEY**(emp_id) **REFERENCES** Employee (emp_id) **ON UPDATE CASCA DE**;

It will give the following output:



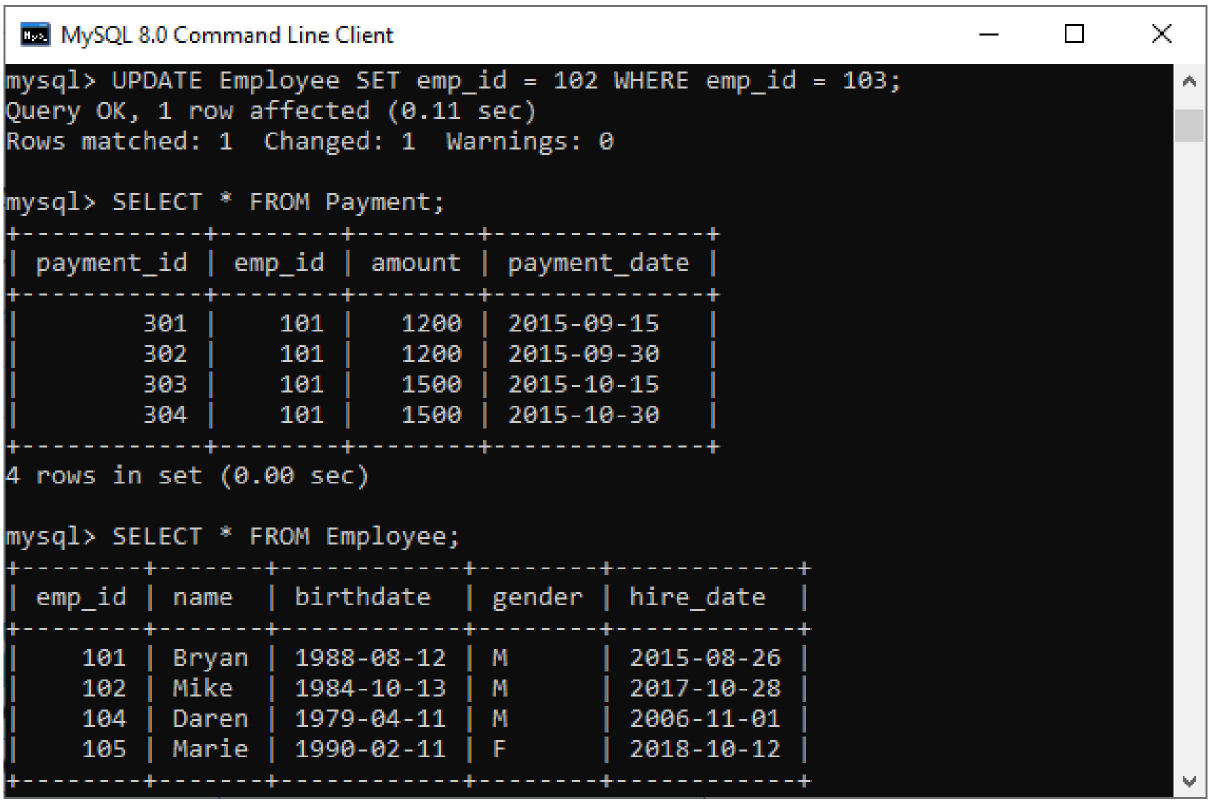
MySQL 8.0 Command Line Client

```
mysql> UPDATE Employee SET emp_id = 102 WHERE emp_id = 103;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

In the below script, we will update the id of the employee in the Parent Table, and it will automatically reflect this change in the child table as well:

1. mysql> **UPDATE** Employee **SET** emp_id = 102 **WHERE** emp_id = 103;

Verifying the content of the Employee and Payment table, we will see that **emp_id** column values will be updated successfully.


MySQL 8.0 Command Line Client

```
mysql> UPDATE Employee SET emp_id = 102 WHERE emp_id = 103;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Payment;
+------------+--------+--------+--------------+
| payment_id | emp_id | amount | payment_date |
+------------+--------+--------+--------------+
|        301 |    101 |   1200 | 2015-09-15   |
|        302 |    101 |   1200 | 2015-09-30   |
|        303 |    101 |   1500 | 2015-10-15   |
|        304 |    101 |   1500 | 2015-10-30   |
+------------+--------+--------+--------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Employee;
+--------+-------+------------+--------+------------+
| emp_id | name  | birthdate  | gender | hire_date  |
+--------+-------+------------+--------+------------+
|    101 | Bryan | 1988-08-12 | M      | 2015-08-26 |
|    102 | Mike  | 1984-10-13 | M      | 2017-10-28 |
|    104 | Daren | 1979-04-11 | M      | 2006-11-01 |
|    105 | Marie | 1990-02-11 | F      | 2018-10-12 |
+--------+-------+------------+--------+------------+
```