TASK -12

Trigger in MySQL

**Summary**: in this task, you will learn how to use the MySQL CREATE TRIGGER statement to create a trigger in the database**.**

Introduction to MySQL CREATE TRIGGER statement

The CREATE TRIGGER statement creates a new trigger. Here is the basic syntax of the CREATE TRIGGER statement:

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE| DELETE }

ON table\_name FOR EACH ROW

trigger\_body;

-

In this syntax:

* First, specify the name of the trigger that you want to create after the CREATE TRIGGER keywords. Note that the trigger name must be unique within a database.
* Next, specify the trigger action time which can be either BEFORE or AFTER which indicates that the trigger is invoked before or after each row is modified.
* Then, specify the operation that activates the trigger, which can be [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx), [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx), or [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx).
* After that, specify the name of the table to which the trigger belongs after the ON keyword.
* Finally, specify the statement to execute when the trigger activates. If you want to execute multiple statements, you use the BEGIN END compound statement.

The trigger body can access the values of the column being affected by the DML statement.

To distinguish between the value of the columns BEFORE and AFTER the DML has fired, you use the NEW and OLD modifiers.

For example, if you update the column description, in the trigger body, you can access the value of the description before the update OLD.description and the new value NEW.description.

The following table illustrates the availability of the OLD and NEW modifiers:

|  |  |  |
| --- | --- | --- |
| Trigger Event | OLD | NEW |
| INSERT | No | Yes |
| UPDATE | Yes | Yes |
| DELETE | Yes | No |

MySQL trigger examples

Let’s start creating a trigger in MySQL to log the changes of the employees table.

Diagram

Description automatically generated with low confidence

Step-1

First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) named employees\_audit to keep the changes to the employees table:

CREATE TABLE employees\_audit (

id INT AUTO\_INCREMENT PRIMARY KEY,

employeeNumber INT NOT NULL,

lastname VARCHAR(50) NOT NULL,

changedat DATETIME DEFAULT NULL,

action VARCHAR(50) DEFAULT NULL

);

Step-2

Next, create a BEFORE UPDATE trigger that is invoked before a change is made to the employees table.

CREATE TRIGGER before\_employee\_update

BEFORE UPDATE ON employees

FOR EACH ROW

INSERT INTO employees\_audit

SET action = 'update',

employeeNumber = OLD.employeeNumber,

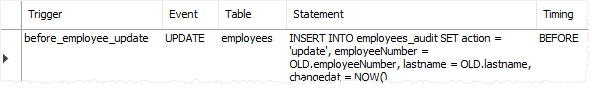
lastname = OLD.lastname,

changedat = NOW();

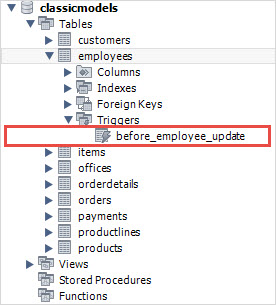
Inside the body of the trigger, we used the OLD keyword to access values of the columns employeeNumber and lastname of the row affected by the trigger.

Then, show all triggers in the current database by using the SHOW TRIGGERS statement:

SHOW TRIGGERS;



In addition, if you look at the schema using MySQL Workbench under the **employees > triggers**, you will see the before\_employee\_update trigger as shown in the screenshot below:



After that, update a row in the employees table:

UPDATE employees

SET

lastName = 'Phan'

WHERE

employeeNumber = 1056;

Finally, query the employees\_audit table to check if the trigger was fired by the UPDATE statement:

SELECT \* FROM employees\_audit;

The following shows the output of the query:

MySQL CREATE TRIGGER example

As you see clearly from the output, the trigger was automatically invoked and inserted a new row into the employees\_audit table.

thus, you have learned how to use the MySQL CREATE TRIGGER statement to create a new trigger in the database.

MySQL DROP TRIGGER

**Summary**: in this , you will learn how to use the MySQL DROP TRIGGER statement to drop a trigger from the database.

Introduction to MySQL DROP TRIGGER statement

The DROP TRIGGER statement deletes a trigger from the database.

Here is the basic syntax of the DROP TRIGGER statement:

DROP TRIGGER [IF EXISTS] [schema\_name.]trigger\_name;

-

In this syntax:

* First, specify the name of the trigger that you want to drop after the DROP TRIGGER keywords.
* Second, specify the name of the schema to which the trigger belongs. If you skip the schema name, the statement will drop the trigger in the current database.
* Third, use IF EXISTS option to conditionally drops the trigger if the trigger exists. The IF EXISTS clause is optional.

If you drop a trigger that does not exist without using the IF EXISTS clause, MySQL issues an error. However, if you use the IF EXISTS clause, MySQL issues a NOTE instead.

The DROP TRIGGER requires the TRIGGER privilege for the table associated with the trigger.

Note that if you [drop a table](https://www.mysqltutorial.org/mysql-drop-table), MySQL will automatically drop all triggers associated with the table.

MySQL DROP TRIGGER example

First, [create a table](https://www.mysqltutorial.org/mysql-create-table/) called billings for demonstration:

CREATE TABLE billings (

billingNo INT AUTO\_INCREMENT,

customerNo INT,

billingDate DATE,

amount DEC(10 , 2 ),

PRIMARY KEY (billingNo)

);

-

Second, [create a new trigger](https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx) called BEFORE UPDATE that is associated with the billings table:

DELIMITER $$

CREATE TRIGGER before\_billing\_update

BEFORE UPDATE

ON billings FOR EACH ROW

BEGIN

IF new.amount > old.amount \* 10 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'New amount cannot be 10 times greater than the current amount.';

END IF;

END$$

DELIMITER ;

-

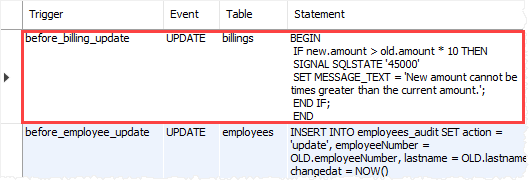
If you are not familiar with the DELIMITER statement, check it out here in the [stored procedure statement](https://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx).

The trigger activates before any update. If the new amount is 10 times greater than the current amount, the trigger raises an error.

Third, show the triggers:

SHOW TRIGGERS;

-



Fourth, drop the before\_billing\_update trigger:

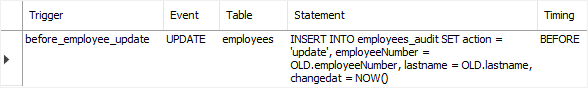
DROP TRIGGER before\_billing\_update;

-

Finally, show the triggers again to verify the removal:

SHOW TRIGGERS;

-



In this tutorial, you have learned how to use the MySQL DROP TRIGGER statement to drop a trigger from the database.

# MySQL BEFORE INSERT Trigger

**Summary**: in this , you will learn how to create a MySQL BEFORE INSERT trigger to maintain a summary table of another table.

## Introduction to MySQL BEFORE INSERT triggers

MySQL BEFORE INSERT [triggers](https://www.mysqltutorial.org/mysql-triggers.aspx) are automatically fired before an [insert](https://www.mysqltutorial.org/mysql-insert-statement.aspx) event occurs on the table.

The following illustrates the basic syntax of creating a MySQL BEFORE INSERT trigger:

CREATE TRIGGER trigger\_name

BEFORE INSERT

ON table\_name FOR EACH ROW

trigger\_body;

-

In this syntax:

First, specify the name of the trigger that you want to create in the [CREATE TRIGGER](https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx) clause.

Second, use BEFORE INSERT clause to specify the time to invoke the trigger.

Third, specify the name of the table that the trigger is associated with after the ON keyword.

Finally, specify the trigger body which contains one or more SQL statements that execute when the trigger is invoked.

If you have multiple statements in the trigger\_body, you have to use the BEGIN END block and change the default [delimiter](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-delimiter/):

DELIMITER $$

CREATE TRIGGER trigger\_name

BEFORE INSERT

ON table\_name FOR EACH ROW

BEGIN

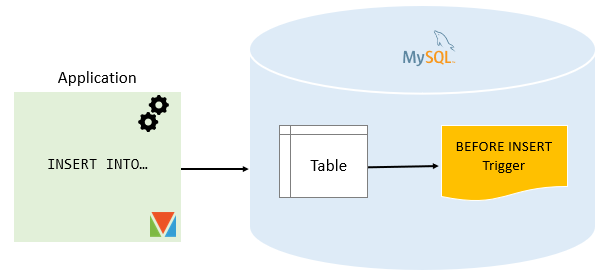
-- statements

END$$

DELIMITER ;

-

Note that in a BEFORE INSERT trigger, you can access and change the NEW values. However, you cannot access the OLD values because OLD values obviously do not exist.



## MySQL BEFORE INSERT trigger example

We will create a BEFORE INSERT trigger to maintain a summary table from another table.

### **Setting up a sample table**

First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) called WorkCenters:

DROP TABLE IF EXISTS WorkCenters;

CREATE TABLE WorkCenters (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

capacity INT NOT NULL

);

-

Second, create another table called WorkCenterStats that stores the summary of the capacity of the work centers:

DROP TABLE IF EXISTS WorkCenterStats;

CREATE TABLE WorkCenterStats(

totalCapacity INT NOT NULL

);

-

### **Creating BEFORE INSERT trigger example**

The following trigger updates the total capacity in the WorkCenterStats table before a new work center is inserted into the WorkCenter table:

DELIMITER $$

CREATE TRIGGER before\_workcenters\_insert

BEFORE INSERT

ON WorkCenters FOR EACH ROW

BEGIN

DECLARE rowcount INT;

SELECT COUNT(\*)

INTO rowcount

FROM WorkCenterStats;

IF rowcount > 0 THEN

UPDATE WorkCenterStats

SET totalCapacity = totalCapacity + new.capacity;

ELSE

INSERT INTO WorkCenterStats(totalCapacity)

VALUES(new.capacity);

END IF;

END $$

DELIMITER ;

-

In this trigger:

First, the name of the trigger is before\_workcenters\_insert specified in the CREATE TRIGGER clause:

CREATE TRIGGER before\_workcenters\_insert

-

Second, the triggering event is:

BEFORE INSERT

-

Third, the table that the trigger associated with is WorkCenters table:

ON WorkCenters FOR EACH ROW

-

Finally, inside the trigger body, we check if there is any row in the WorkCenterStats table.

If the table WorkCenterStats has a row, the trigger adds the capacity to the totalCapacity column. Otherwise, it inserts a new row into the WorkCenterStats table.

### **Testing the MySQL BEFORE INSERT trigger**

First, [insert a new row](https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the WorkCenter table:

INSERT INTO WorkCenters(name, capacity)

VALUES('Mold Machine',100);

-

Second, [query data](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the WorkCenterStats table:

SELECT \* FROM WorkCenterStats;

-



The trigger has been invoked and inserted a new row into the WorkCenterStats table.

Third, insert a new work center:

INSERT INTO WorkCenters(name, capacity)

VALUES('Packing',200);

-

Finally, query data from the WorkCenterStats:

SELECT \* FROM WorkCenterStats;

-

MySQL BEFORE INSERT Trigger Example 2

The trigger has updated the total capacity from 100 to 200 as expected.

Note that to properly maintain the summary table WorkCenterStats, you should also create triggers to handle update and delete events on the WorkCenters table.

In this tutorial, you have learned how to create a MySQL BEFORE INSERT trigger to maintain a summary table of another table.

# MySQL AFTER INSERT Trigger

**Summary**: in this , you will learn how to create a MySQL AFTER INSERT trigger to insert data into a table after inserting data into another table.

## Introduction to MySQL AFTER INSERT triggers

MySQL AFTER INSERT triggers are automatically invoked after an insert event occurs on the table.

The following shows the basic syntax of creating a MySQL AFTER INSERT trigger:

CREATE TRIGGER trigger\_name

AFTER INSERT

ON table\_name FOR EACH ROW

trigger\_body

-

In this syntax:

First, specify the name of the trigger that you want to create after the [CREATE TRIGGER](https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx) keywords.

Second, use AFTER INSERT clause to specify the time to invoke the trigger.

Third, specify the name of the table on which you want to create the trigger after the ON keyword.

Finally, specify the trigger body which consists of one or more statements that execute when the trigger is invoked.

In case the trigger body has multiple statements, you need to use the BEGIN END block and change the default [delimiter](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-delimiter/):

DELIMITER $$

CREATE TRIGGER trigger\_name

AFTER INSERT

ON table\_name FOR EACH ROW

BEGIN

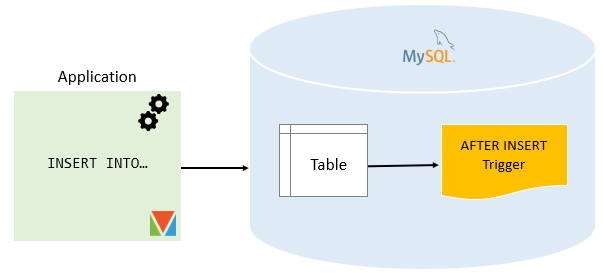
-- statements

END$$

DELIMITER ;

-

In an AFTER INSERT trigger, you can access the NEW values but you cannot change them. Also, you cannot access the OLD values because there is no OLD on INSERT triggers.



## MySQL AFTER INSERT trigger example

Consider the following AFTER INSERT trigger example.

### **Setting up a sample table**

First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) called members:

DROP TABLE IF EXISTS members;

CREATE TABLE members (

id INT AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(255),

birthDate DATE,

PRIMARY KEY (id)

);

-

Second, create another table called reminders that stores reminder messages to members.

DROP TABLE IF EXISTS reminders;

CREATE TABLE reminders (

id INT AUTO\_INCREMENT,

memberId INT,

message VARCHAR(255) NOT NULL,

PRIMARY KEY (id , memberId)

);

-

### **Creating AFTER INSERT trigger example**

The following statement creates an AFTER INSERT trigger that inserts a reminder into the reminders table if the birth date of the member is NULL.

DELIMITER $$

CREATE TRIGGER after\_members\_insert

AFTER INSERT

ON members FOR EACH ROW

BEGIN

IF NEW.birthDate IS NULL THEN

INSERT INTO reminders(memberId, message)

VALUES(new.id,CONCAT('Hi ', NEW.name, ', please update your date of birth.'));

END IF;

END$$

DELIMITER ;

-

In this trigger:

First, the name of the trigger is after\_members\_insert specified in the CREATE TRIGGER clause:

CREATE TRIGGER after\_members\_insert

-

Second, the triggering event is:

AFTER INSERT

-

Third, the table that the trigger associated with is members table:

ON members FOR EACH ROW

-

Finally, inside the trigger body, insert a new row into the reminder table if the birth date of the member is NULL.

### **Testing the MySQL AFTER INSERT trigger**

First, [insert two rows](https://www.mysqltutorial.org/mysql-insert-multiple-rows/) into the members table:

INSERT INTO members(name, email, birthDate)

VALUES

('John Doe', 'john.doe@example.com', NULL),

('Jane Doe', 'jane.doe@example.com','2000-01-01');

-

Second, [query data](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the members table:

SELECT \* FROM members;

-

MySQL AFTER INSERT Trigger example

Third, query data from reminders table:

SELECT \* FROM reminders;

-

MySQL AFTER INSERT Trigger Output

We inserted two rows into the members table. However, only the first row that has a birth date value NULL, therefore, the trigger inserted only one row into the reminders table.

In this tutorial, you have learned how to create a MySQL AFTER INSERT trigger to insert data into a table after inserting data into another table.

# MySQL BEFORE DELETE Trigger

**Summary**: in this tutorial, you will learn how to create a MySQL BEFORE DELETE trigger to add deleted rows into an archive table.

## Introduction to MySQL BEFORE DELETE triggers

MySQL BEFORE DELETE [triggers](https://www.mysqltutorial.org/mysql-triggers.aspx) are fired automatically before a [delete](https://www.mysqltutorial.org/mysql-delete-statement.aspx) event occurs in a table.

Here is the basic syntax of creating a MySQL BEFORE DELETE trigger:

CREATE TRIGGER trigger\_name

BEFORE DELETE

ON table\_name FOR EACH ROW

trigger\_body

-

In this syntax:

First, specify the name of the trigger which you want to create after the CREATE TRIGGER keywords.

Second, use BEFORE DELETE clause to specify that the trigger is invoked right before a delete event.

Third, specify the name of the table that the trigger is associated with after the ON keyword.

Finally, specify the trigger body which consists of one or more statements that execute when the trigger is fired.

Notice that if you have multiple statements in the trigger\_body, you need to use the BEGIN END block to wrap these statements and temporarily change the default delimiter as follows:

DELIMITER $$

CREATE TRIGGER trigger\_name

BEFORE DELETE

ON table\_name FOR EACH ROW

BEGIN

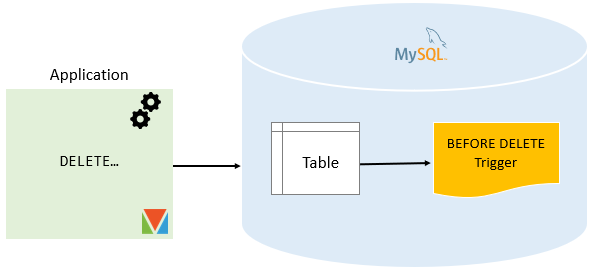
-- statements

END$$

DELIMITER ;

-

In a BEFORE DELETE trigger, you can access the OLD row but cannot update it. Also, there is no NEW row in the BEFORE DELETE trigger.



## MySQL BEFORE DELETE trigger example

Let’s see the following BEFORE DELETE trigger example.

### **Setting up a sample table**

First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) called Salaries that stores salary information of employees

DROP TABLE IF EXISTS Salaries;

CREATE TABLE Salaries (

employeeNumber INT PRIMARY KEY,

validFrom DATE NOT NULL,

amount DEC(12 , 2 ) NOT NULL DEFAULT 0

);

-

Second, [insert some rows](https://www.mysqltutorial.org/mysql-insert-multiple-rows/) into the Salaries table:

INSERT INTO salaries(employeeNumber,validFrom,amount)

VALUES

(1002,'2000-01-01',50000),

(1056,'2000-01-01',60000),

(1076,'2000-01-01',70000);

-

Third, [create a table](https://www.mysqltutorial.org/mysql-create-table/) that stores the deleted salary:

DROP TABLE IF EXISTS SalaryArchives;

CREATE TABLE SalaryArchives (

id INT PRIMARY KEY AUTO\_INCREMENT,

employeeNumber INT PRIMARY KEY,

validFrom DATE NOT NULL,

amount DEC(12 , 2 ) NOT NULL DEFAULT 0,

deletedAt TIMESTAMP DEFAULT NOW()

);

-

### **Creating BEFORE DELETE trigger example**

The following BEFORE DELETE trigger inserts a new row into the SalaryArchives table before a row from the Salaries table is deleted.

DELIMITER $$

CREATE TRIGGER before\_salaries\_delete

BEFORE DELETE

ON salaries FOR EACH ROW

BEGIN

INSERT INTO SalaryArchives(employeeNumber,validFrom,amount)

VALUES(OLD.employeeNumber,OLD.validFrom,OLD.amount);

END$$

DELIMITER ;

-

In this trigger:

First, the name of the trigger is before\_salaries\_delete specified in the CREATE TRIGGER clause:

CREATE TRIGGER before\_salaries\_delete

-

Second, the triggering event is:

BEFORE DELETE

-

Third, the table that the trigger associated with is Salaries table:

ON Salaries FOR EACH ROW

-

Finally, inside the trigger body insert deleted row into the SalaryArchives table.

### **Testing the MySQL BEFORE DELETE trigger**

First, delete a row from the Salaries table:

DELETE FROM salaries

WHERE employeeNumber = 1002;

-

Second, query data from the SalaryArchives table:

SELECT \* FROM SalaryArchives;

-

MySQL BEFORE DELETE Trigger example

The trigger was invoked and inserted a new row into the SalaryArchives table.

Third, delete all rows from the Salaries table:

DELETE FROM salaries;

-

Finally, query data from the SalaryArchives table:

SELECT \* FROM SalaryArchives;

-

Table

Description automatically generated

The trigger was trigger twice because the DELETE statement deleted two rows from the Salaries table.

In this tutorial, you have learned how to create a MySQL BEFORE DELETE trigger to add deleted rows into an archive table.

# MySQL AFTER DELETE Trigger

**Summary**: in this tutorial, you will learn how to create a MySQL AFTER DELETE trigger to maintain a summary table of another table.

## Introduction to MySQL AFTER DELETE triggers

MySQL AFTER DELETE  [triggers](https://www.mysqltutorial.org/mysql-triggers.aspx) are automatically invoked after a [delete](https://www.mysqltutorial.org/mysql-delete-statement.aspx) event occurs on the table.

Here is the basic syntax of creating a MySQL AFTER DELETE trigger:

CREATE TRIGGER trigger\_name

AFTER DELETE

ON table\_name FOR EACH ROW

trigger\_body;

In this syntax:

First, specify the name of the trigger that you want to create in the CREATE TRIGGER clause.

Second, use AFTER DELETE clause to specify the time to invoke the trigger.

Third, specify the name of the table, which the trigger is associated with, after the ON keyword.

Finally, specify the trigger body which contains one or more statements that execute when the trigger is invoked.

If you have multiple statements in the trigger\_body, you need to use the BEGIN END block to wrap them and flip the [default delimiter](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-delimiter/) between $$ and ; as shown in the following:

DELIMITER $$

CREATE TRIGGER trigger\_name

AFTER DELETE

ON table\_name FOR EACH ROW

BEGIN

-- statements

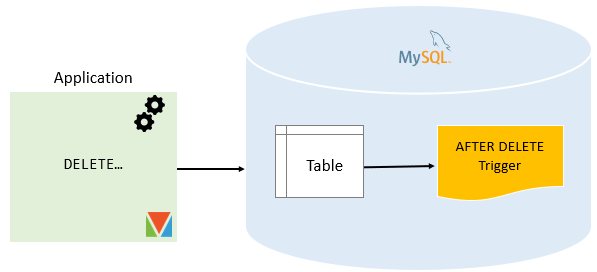
END$$

DELIMITER ;

-

In an AFTER DELETE trigger, you can access the OLD row but cannot change it.

Note that there is no NEW row in the AFTER DELETE trigger.



## MySQL AFTER DELETE trigger example

Consider the following AFTER DELETE trigger example.

### **Setting up a sample table**

First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) called Salaries:

DROP TABLE IF EXISTS Salaries;

CREATE TABLE Salaries (

employeeNumber INT PRIMARY KEY,

salary DECIMAL(10,2) NOT NULL DEFAULT 0

);

-

Second, [insert some rows](https://www.mysqltutorial.org/mysql-insert-multiple-rows/) into the Salaries table:

INSERT INTO Salaries(employeeNumber,salary)

VALUES

(1002,5000),

(1056,7000),

(1076,8000);

-

Third, create another table called SalaryBudgets that stores the total of salaries from the Salaries table:

DROP TABLE IF EXISTS SalaryBudgets;

CREATE TABLE SalaryBudgets(

total DECIMAL(15,2) NOT NULL

);

-

Fourth, use the [SUM()](https://www.mysqltutorial.org/mysql-sum/) function to get the total salary from the Salaries table and insert it into the SalaryBudgets table:

INSERT INTO SalaryBudgets(total)

SELECT SUM(salary)

FROM Salaries;

-

Finally, [query data](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the SalaryBudgets table:

SELECT \* FROM SalaryBudgets;

-



### **Creating AFTER DELETE trigger example**

The following AFTER DELETE trigger updates the total salary in the SalaryBudgets table after a row is deleted from the Salaries table:

CREATE TRIGGER after\_salaries\_delete

AFTER DELETE

ON Salaries FOR EACH ROW

UPDATE SalaryBudgets

SET total = total - old.salary;

-

In this trigger:

First, the name of the trigger is after\_salaries\_delete specified in the CREATE TRIGGER clause:

CREATE TRIGGER after\_salaries\_delete

-

Second, the triggering event is:

AFTER DELETE

-

Third, the table that the trigger associated with is Salaries table:

ON Salaries FOR EACH ROW

-

Finally, inside the trigger body, we subtract deleted salary from the total salary.

### **Testing the MySQL AFTER DELETE trigger**

First, delete a row from the Salaries table:

DELETE FROM Salaries

WHERE employeeNumber = 1002;

-

Second, query total salary from the SalaryBudgets table:

SELECT \* FROM SalaryBudgets;

-



As you can see from the output, the total is reduced by the deleted salary.

Third, delete all rows from the salaries table:

DELETE FROM Salaries;

-

Finally, query the total from the SalaryBudgets table:

SELECT \* FROM SalaryBudgets;

-



The trigger updated the total to zero.

In this tutorial, you have learned how to create a MySQL AFTER DELETE trigger to maintain a summary table of another table.