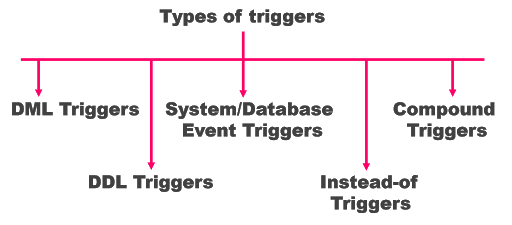
A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server.



A MySQL **trigger** is a stored program (with queries) which is executed automatically to respond to a specific event such as insertion, updation or deletion occurring in a table.

**Before Update Trigger:**  
It is a trigger which enacts before an update is invoked. If we write an update statement, then the actions of the trigger will be performed before the update is implemented

Step-1 Creating Table

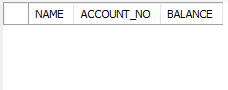
CREATE TABLE account\_holder(

NAME VARCHAR(255),

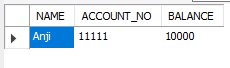
ACCOUNT\_NO INT(255),

BALANCE INT(255)

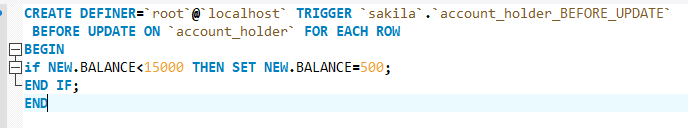
);



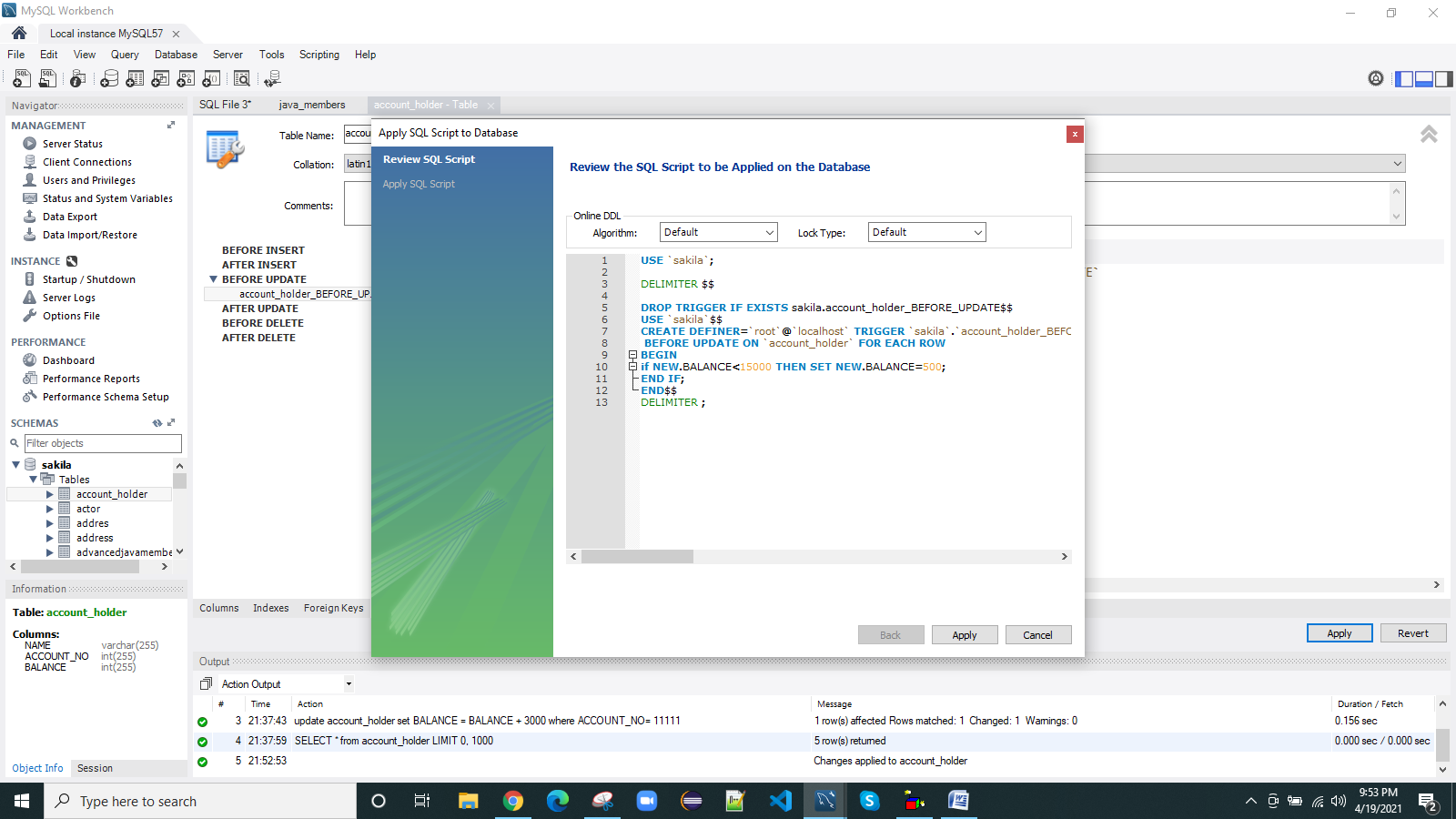
Step-2: insert data into table



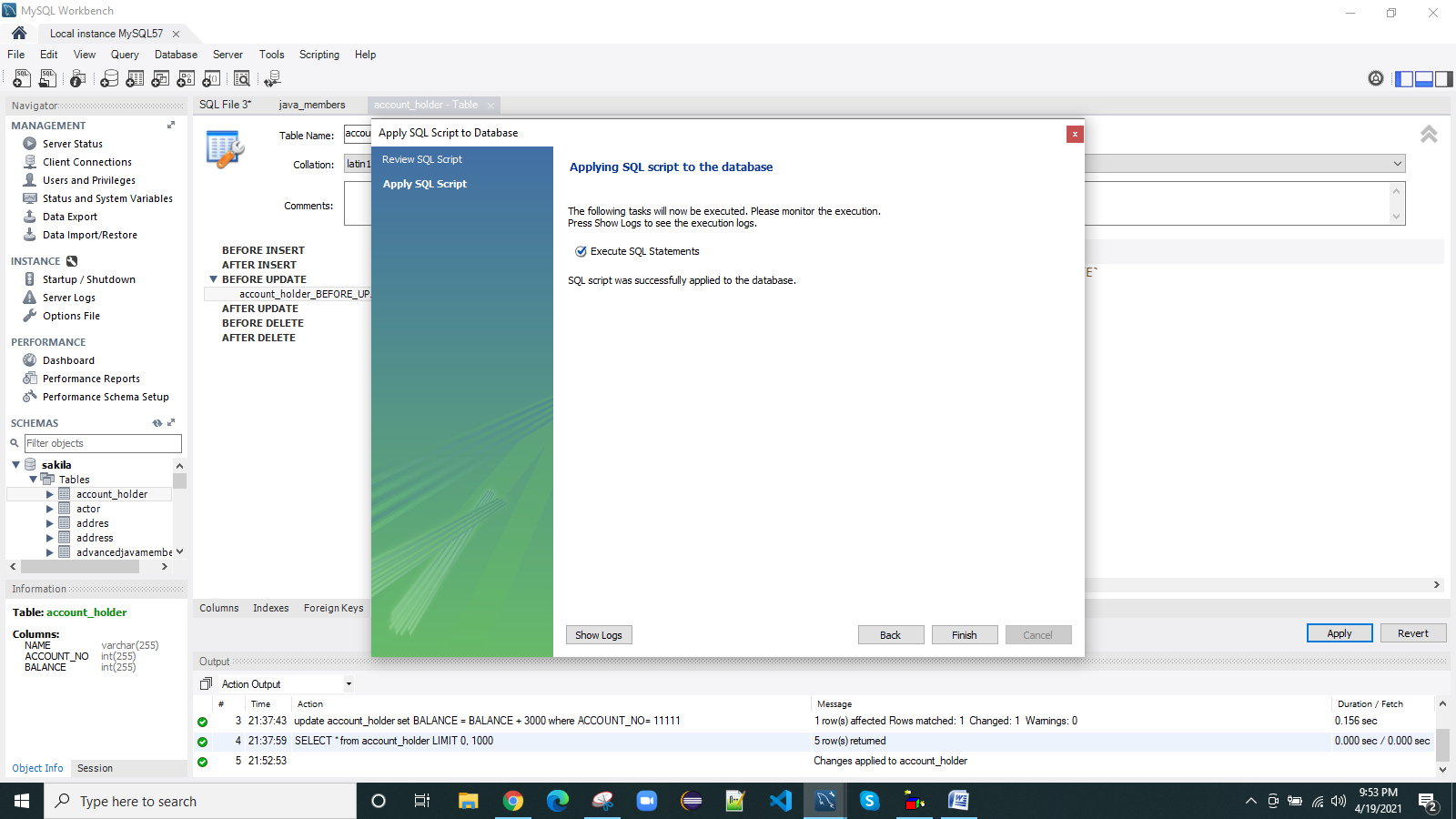
Step-3 creating trigger



Now click on Apply



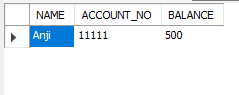
Now click on Finish , Trigger is created



Step-4 updating

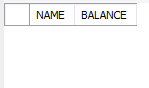
update account\_holder set BALANCE = BALANCE + 3000 where ACCOUNT\_NO= 11111;

step-5 check the updates in the table

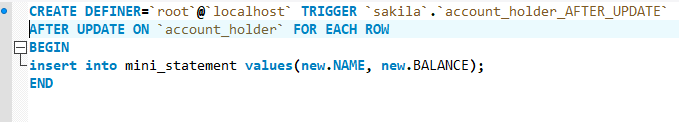


**After Update Trigger:**  
This trigger is invoked after an updation occurs. (i.e., it gets implemented after an update statement is executed.).

Step-1 create a table with table name mini\_statement

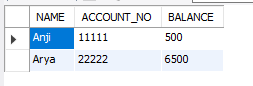


Step-2 create trigger

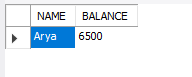


Repeat the same process again

STEP-3 INSERTING DATA INTO TABLE WITH TABLE NAME account\_holder



Step-4 data inside the mini\_statement table.(after update)

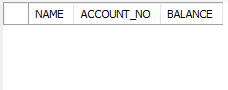
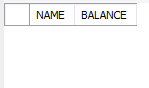


**Before Insert Trigger:**

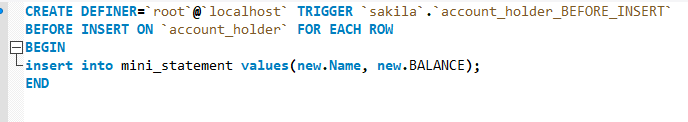
This trigger is invoked before an insert, or before an insert statement is executed.

STEP-1:CREATING TABLES AS account\_holder and mini\_statement

account\_holder mini\_statement

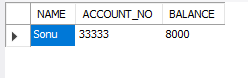
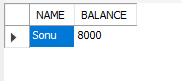
Step-2 : CREATING TRIGGER



Step -3 INSERT DATA INTO THE TABLE account\_holder

insert into account\_holder values('Sonu','33333','8000');

TABLE NAME: account\_holder TABLE NAME: mini\_statement

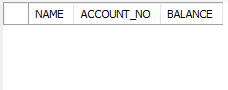
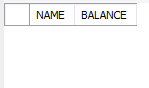
Here ,data is inserted into the table account\_holder but not in the table mini\_statement.

But data also be inserted into the table mini\_statement because of TRIGGER.

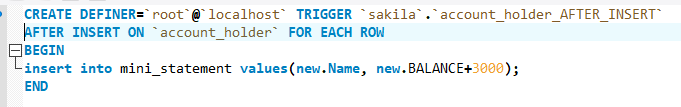
**After Insert Trigger:**  
This trigger gets invoked after an insert is implemented.

STEP-1:CREATING TABLES AS account\_holder and mini\_statement

account\_holder mini\_statement

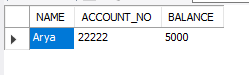
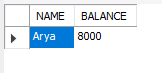
Step-2 : CREATE TRIGGER



Step -3 INSERT DATA INTO THE TABLE account\_holder

insert into account\_holder values('Arya','22222','5000');

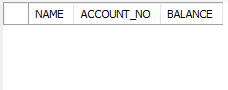
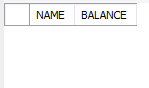
TABLE NAME: account\_holder TABLE NAME: mini\_statement

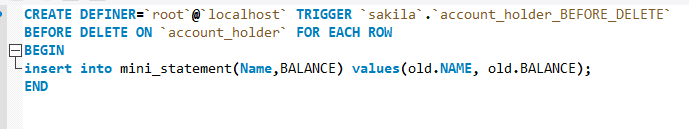
**Before Delete Trigger:**  
This trigger is invoked before a delete occurs, or before deletion statement is implemented.

STEP-1:CREATING TABLES AS account\_holder and mini\_statement

account\_holder mini\_statement

Step-2 : CREATE TRIGGER



Step -3 insert data into account\_holder

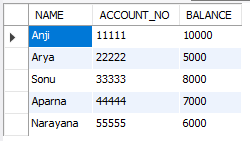
insert into account\_holder values('Anji','11111','10000');

insert into account\_holder values('Arya','22222','5000');

insert into account\_holder values('Sonu','33333','8000');

insert into account\_holder values('Aparna','44444','7000');

insert into account\_holder values('Narayana','55555','6000');



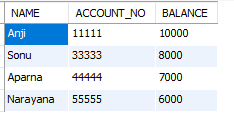
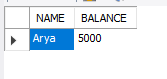
STEP-4:

DELETE FROM account\_holder

WHERE

ACCOUNT\_NO = 22222;

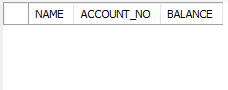
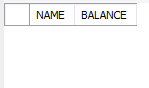
STEP-5:

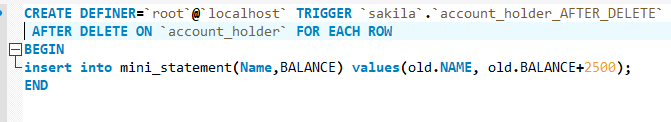
**After Delete Trigger:**  
As the name implies, this trigger is invoked after a delete occurs, or after a delete operation is implemented.

STEP-1:CREATING TABLES AS account\_holder and mini\_statement

account\_holder mini\_statement

Step-2 : CREATE TRIGGER



Step -3 insert data into account\_holder

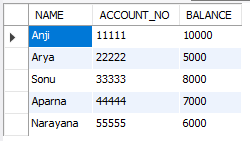
insert into account\_holder values('Anji','11111','10000');

insert into account\_holder values('Arya','22222','5000');

insert into account\_holder values('Sonu','33333','8000');

insert into account\_holder values('Aparna','44444','7000');

insert into account\_holder values('Narayana','55555','6000');



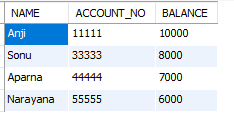
STEP-4:

DELETE FROM account\_holder

WHERE

ACCOUNT\_NO = 22222;

STEP-:

**System/database event triggers**

**LOG-ON AND LOG-OFF TRIGGERS**

Step-1 create table with name sakila.logon\_logoff\_event

CREATE TABLE sakila.logon\_logoff\_event (

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

connection\_id INT UNSIGNED NOT NULL,

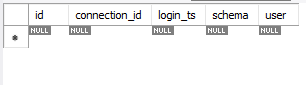
login\_ts TIMESTAMP NOT NULL,

`schema` VARCHAR(64) NULL,

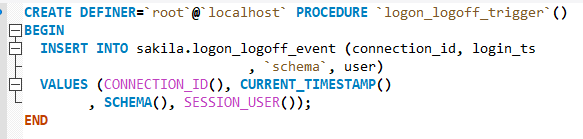
user VARCHAR(77) NOT NULL,

PRIMARY KEY (id)

);



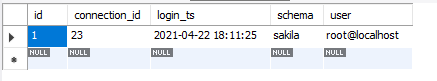
STEP-2 Create trigger as stored procedure



STEP-3 Call the stored procedure

call sakila.logon\_logoff\_trigger;

STEP-4 SEE THE DATA INSIDE TABLE



THE DATA INSIDE THE TABLE IS LOGGING DETAILS OF SERVER

**STARTUP AND SHUTDOWN TRIGGER**

NOTE: SAME PROCEDURE THAT HOW THE LOGON AND LOGOFF TRIGGERS CAN BE EXECUTED

What is an INSTEAD OF trigger

An INSTEAD OF trigger is a trigger that allows you to skip an [INSERT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/), [DELETE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-delete/), or [UPDATE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-update/) statement to a table or a view and execute other statements defined in the trigger instead. The actual insert, delete, or update operation does not occur at all.

In other words, an INSTEAD OF trigger skips a DML statement and execute other statements.

Syntax;

CREATE TRIGGER [schema\_name.] trigger\_name

ON {table\_name | view\_name }

INSTEAD OF {[INSERT] [,] [UPDATE] [,] [DELETE] }

AS

{sql\_statements}