**Aim**: SQL Triggers.

**Objectives:**

The objectives for learning MySQL triggers are to understand how triggers automatically perform actions when data changes, such as during INSERT, UPDATE, or DELETE events. Triggers are used for tasks like logging changes or updating related tables. The goal is to learn how to use triggers effectively without causing performance issues and ensure data consistency.

**Tools Used:**

* MySQL Workbench

**Concept:**

Triggers in SQL: Triggers are special types of stored procedures that automatically execute predefined actions when certain events (like INSERT, UPDATE, or DELETE) occur in a table. They help automate tasks such as data validation, updating related tables, or logging changes, ensuring data consistency and reducing the need for manual interventions.

**Example:**

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_BEFORE\_INSERT` BEFORE INSERT ON `emp` FOR EACH ROW BEGIN

if new.Working\_hours < 0

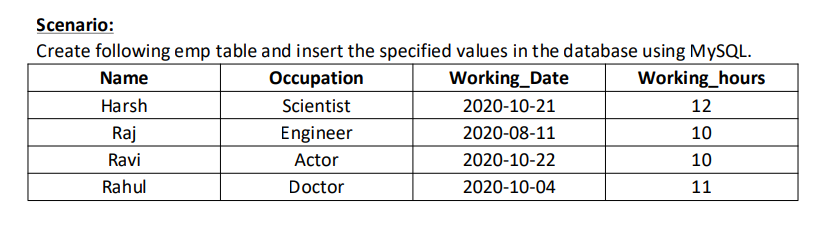
then

set new.Working\_hours = 0;

end if;

END

**Problem Statement:**

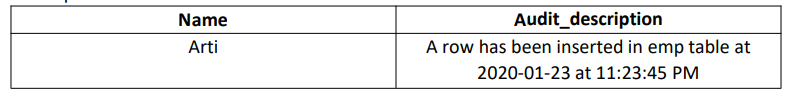
****

1) Question on Before Insert Trigger:

Write a trigger which ensures that if user enters negative value in Working\_hours the value is set to 0. 2) Question on After Insert Trigger:

Create a table emp\_audit(name, audit\_description)

Create a trigger to make sure If any employee information is inserted in emp table then trigger is inserting the row in emp\_audit table automatically. Output should look like:

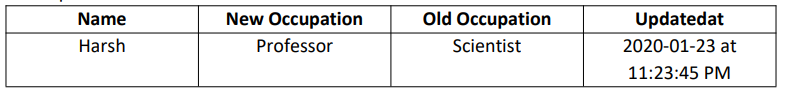


3) Question on Before Update Trigger:

Create a trigger if a new working date is greater than today’s date to raise an error message.

4) Question on After Update Trigger:

Create a table EmpChanges(Name, New Occupation, Old Occupation,Updatedate as shown in following output. Create a trigger that will keep history of changes in the EmpChange table when you change data in Emp table. Output should look like:



5) Question on Before Delete Trigger:

Create a table Emp\_archeives (Name,Occupation,Working\_date,WorkingHours, Deletedat)

Create trigger to ensure before removing data from Emp table, the record should be entered in Emp\_archieves table.

6) Question on After Delete Trigger:

Consider you have two tables Emp Table(Original Table) and Total\_working\_hours\_table which looks like



Create a trigger that changes the Total of above table when Emp leaves the company.

**Solution:**

1)

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_BEFORE\_INSERT` BEFORE INSERT ON `emp` FOR EACH ROW BEGIN

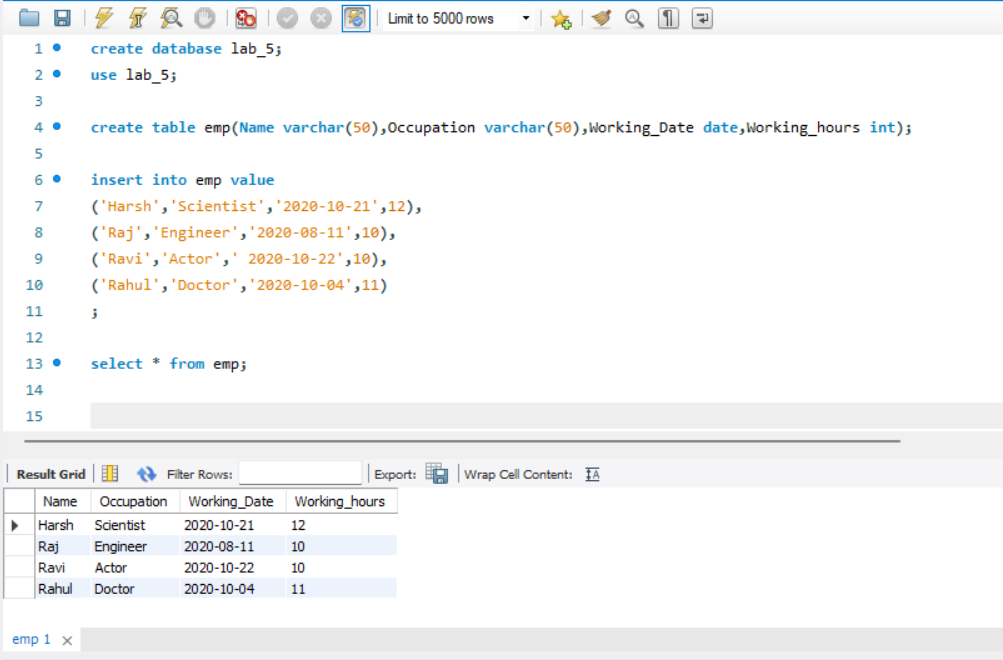
if new.Working\_hours < 0

then

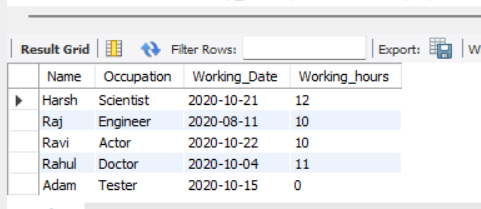
set new.Working\_hours = 0;

end if;

END



insert into emp value('Adam','Tester','2020-10-15',-20);



2)

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_AFTER\_INSERT` AFTER INSERT ON `emp` FOR EACH ROW BEGIN

insert into emp\_audit values

(new.Name, concat(

'A row has been inserted in emp table at ',

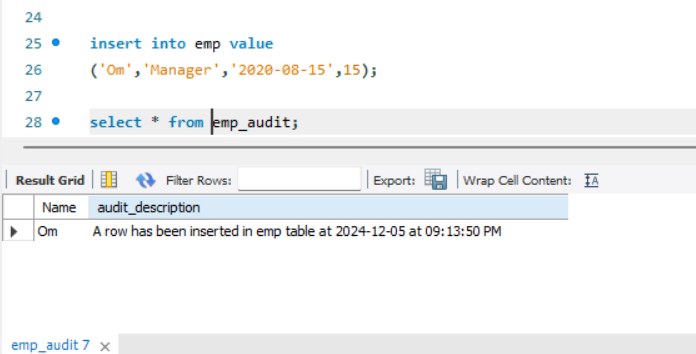
Date\_Format(NOW(),'%Y-%m-%d'),

' at ',

Date\_Format(NOW(),'%h:%i:%s %p')

));

END



3)

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_BEFORE\_UPDATE` BEFORE UPDATE ON `emp` FOR EACH ROW BEGIN

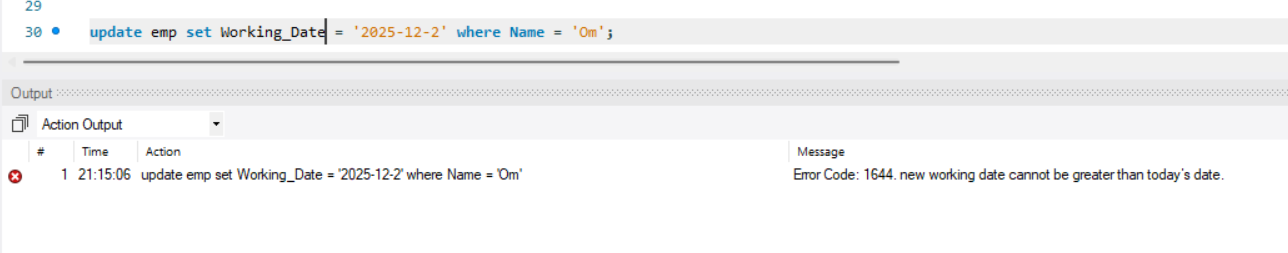
if new.working\_date > date\_format(now(),'%y-%m-%d') then

signal sqlstate '45000'

set message\_text = 'new working date cannot be greater than today’s date.';

end if;

END



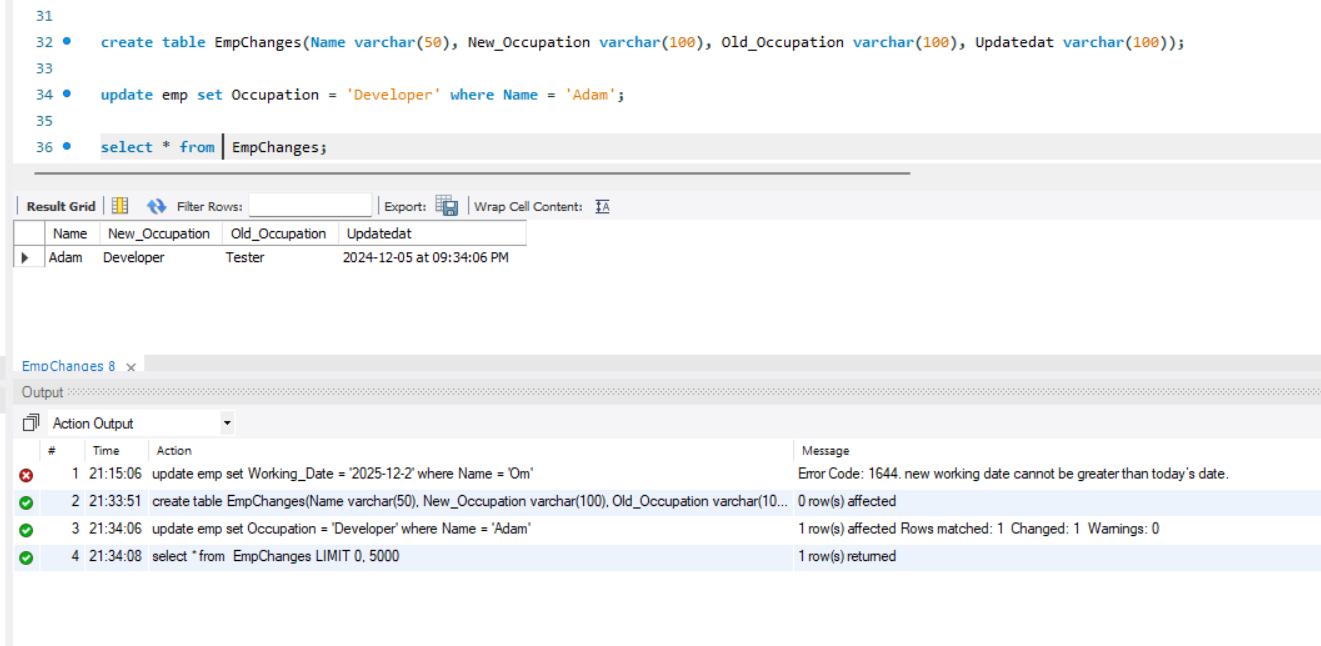
4)

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_AFTER\_UPDATE` AFTER UPDATE ON `emp` FOR EACH ROW BEGIN

insert into EmpChanges values

(new.Name,new.Occupation, old.Occupation, concat(Date\_Format(now(),'%Y-%m-%d'), ' at ',Date\_Format(now(),'%h:%i:%s %p')));

END



5)

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_BEFORE\_DELETE` BEFORE DELETE ON `emp` FOR EACH ROW BEGIN

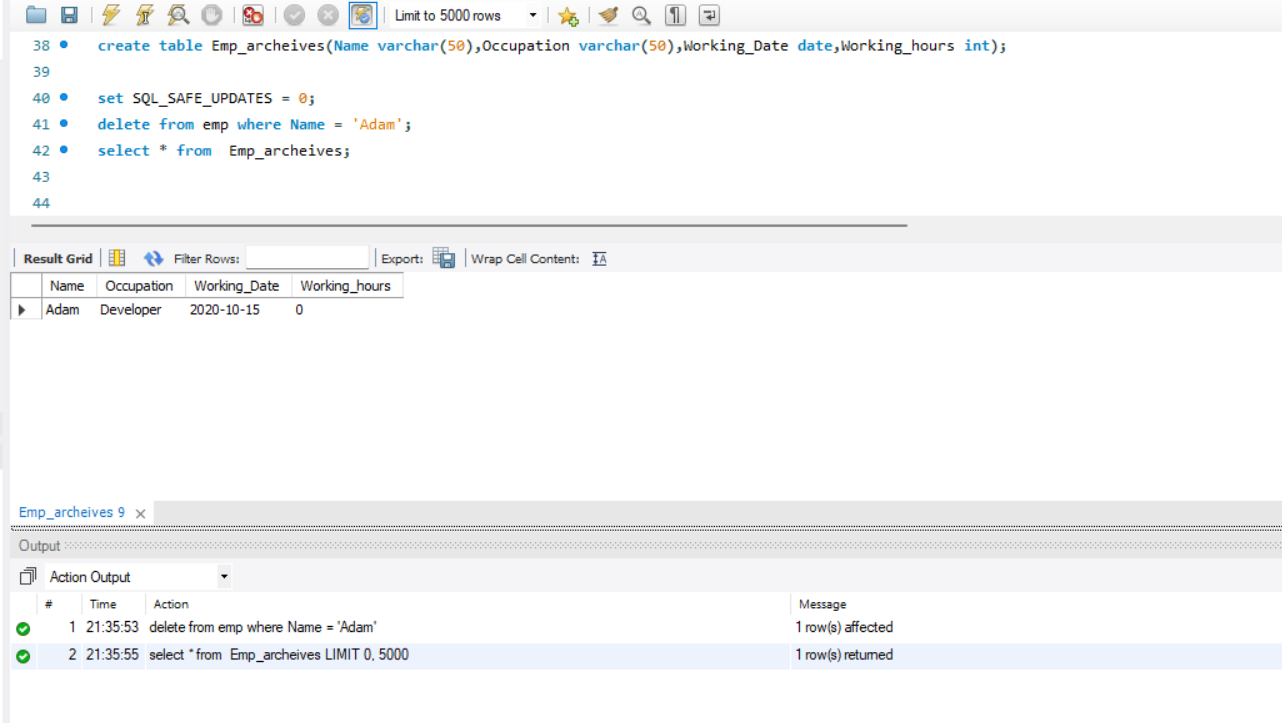
insert into Emp\_archeives values

(

old.Name,old.Occupation,old.Working\_Date,old.Working\_hours

);

END

****

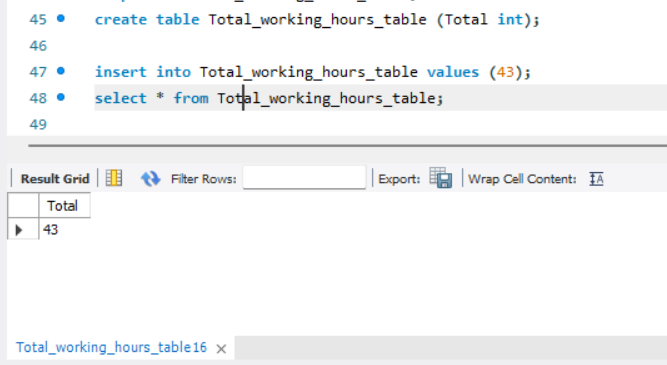
6)

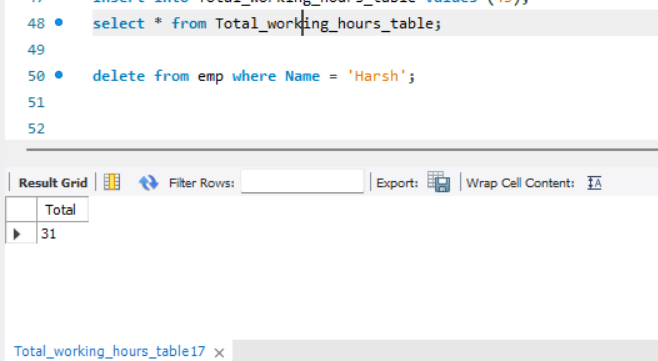
CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_AFTER\_DELETE` AFTER DELETE ON `emp` FOR EACH ROW BEGIN

update total\_working\_hours\_table

set total = total - old.working\_hours;

END





**Observation:**

Triggers are useful tools in SQL that automatically perform actions in response to certain events, such as INSERT, UPDATE, or DELETE. They help automate tasks like updating related tables, enforcing data integrity, and logging changes, without requiring manual intervention.