Name	**Hatim Yusuf Sawai**
UID no.	**2021300108**
Experiment No.	**8**

AIM:	**Implement Triggers in MySql**
Program 1	
PROBLEM STATEMENT:	Implement different types of triggers on tables in the existing database in mysql
THEORY:	**TRIGGERS:** A trigger in MySQL is a set of SQL statements that reside in a system catalog. **It is a special type of stored procedure that is invoked automatically in response to an event**. Each trigger is associated with a table, which is activated on any DML statement such as **INSERT, UPDATE**, or **DELETE**. **Limitations of Using Triggers in MySQL** 1. MySQL triggers do not allow to use of all validations; they only provide extended validations. **For example**, we can use the NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints for simple validations. 2. Triggers are invoked and executed invisibly from the client application. Therefore, it isn't easy to troubleshoot what happens in the database layer. 3. Triggers may increase the overhead of the database server.

Types of Triggers in MySQL:

1. BEFORE INSERT:

It is activated before the insertion of data into the table.

Syntax:

CREATE TRIGGER trigger_name
BEFORE INSERT ON table_name
FOR EACH ROW
BEGIN
trigger code
END;

2. AFTER INSERT:

It is activated after the insertion of data into the table.

Syntax:

CREATE TRIGGER trigger_name
AFTER INSERT ON table_name
FOR EACH ROW
BEGIN
trigger code
END;

3. BEFORE UPDATE:

It is activated before the update of data in the table.

Syntax:

CREATE TRIGGER trigger_name
BEFORE UPDATE ON table_name
FOR EACH ROW
BEGIN
trigger code
END;

4. AFTER UPDATE

It is activated after the update of the data in the table.

Syntax:

	CREATE TRIGGER trigger_name
	AFTER UPDATE ON table_name
	FOR EACH ROW
	BEGIN
	trigger code
	END;
	
	5. BEFORE DELETE
	It is activated before the data is removed from the table.
	Syntax:
	CREATE TRIGGER trigger_name
	BEFORE DELETE ON table_name
	FOR EACH ROW
	BEGIN
	trigger code
	END;
	
	6. AFTER DELETE
	It is activated after the deletion of data from the table.
	Syntax:
	CREATE TRIGGER trigger_name
	AFTER DELETE ON table_name
	FOR EACH ROW
	BEGIN
	trigger code
	END;

QUERIES:	**Before Insert Trigger**
	1. Create a trigger to check if new record contains patient age less than 18, if so, throw an error:
	
	CREATE trigger patient_trigger
	BEFORE INSERT ON patient
	FOR EACH ROW
	BEGIN
	IF NEW.Age<18 THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Age must be greater than 18';
	END IF;
	END;
	
	▷ Run on active connection | ≡ Select block
	-- @BLOCK
	INSERT INTO patient VALUES (16,'Vanshika',17,
	'Andheri',9479574411,null);
	
	Query with errors. Please, check the error below.
	
	ER_SIGNAL_EXCEPTION: Age must be greater than 18
	
	2. Create a trigger to check if new salary being inserted is always over 0 (+ve) and if not then default the salary column to 0:
	CREATE trigger doc_salary_trigger
	BEFORE INSERT ON doctor
	FOR EACH ROW
	BEGIN
	IF NEW.Salary<0 THEN
	SET NEW.Salary = 0;
	END IF;
	END;

```
▷ Run on active connection | ≡ Select block
-- @BLOCK
INSERT INTO doctor VALUES (13,'Dr. Rajesh',
7876539261,-100,'Cardiologist','Andheri');
```

D_id	Dname	Ph_no	Salary	Field	Address
abc Filte	abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
13	Dr. Rajesh	7876539261	0	Cardiologist	Andheri

Before Update Trigger:

3. Create a trigger to change the address to "Andheri West" whenever the user updates the address to "Andheri":

```
CREATE trigger pat_update_trigger
BEFORE UPDATE ON patient
FOR EACH ROW
BEGIN
IF NEW.address='Andheri' THEN
SET NEW.address='Andheri West';
END IF;
END;
```

```
▷ Run on active connection | ≡ Select block
-- @BLOCK
UPDATE patient SET address='Andheri' WHERE P_id=5;
SELECT * FROM patient WHERE P_id=5;
```

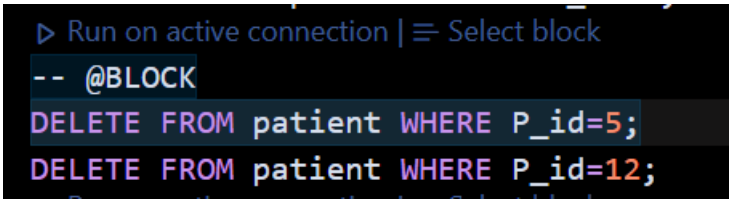
P_id	Pname	Age	Address	Ph_no	D_id
abc Filter.	abc Filter...	abc Filter.	abc Filter...	abc Filter...	abc Filte
5	Hatim	45	Andheri West	9876543210	5

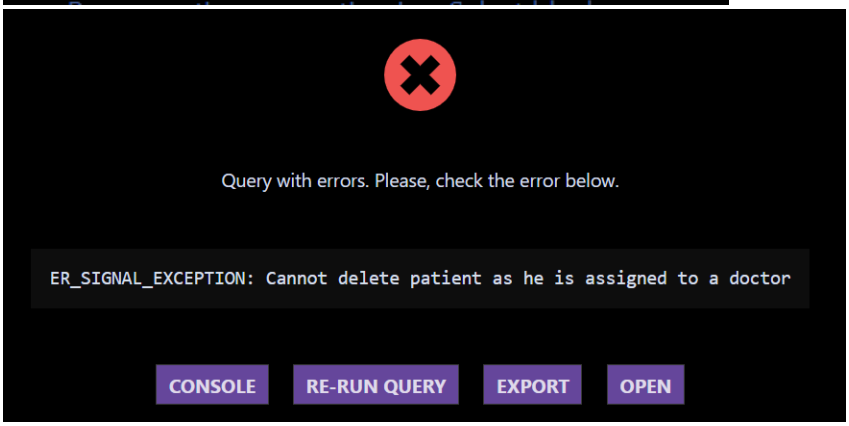
Before delete trigger:

4. Create a trigger to check if patient is not assigned to any doctor before deletion:

```
CREATE trigger pat_delete_trigger
BEFORE DELETE ON patient
FOR EACH ROW
BEGIN
```

If OLD.D_id is not null THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Cannot delete patient as he is assigned to a doctor';
END IF;
END;





5. Create a trigger to check if a doctor is assigned to a patient before deleting the doctor record:

CREATE trigger doc_delete_trigger
BEFORE DELETE ON doctor
FOR EACH ROW
BEGIN
If OLD.D_id in (SELECT D_id FROM patient) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Cannot delete doctor as he is assigned to a patient';
END IF;
END;

```
▷ Run on active connection | ≡ Select block
-- @BLOCK
DELETE FROM doctor WHERE D_id=5;
```

Query with errors. Please, check the error below.

```
ER_SIGNAL_EXCEPTION: Cannot delete doctor as he is assigned to a patient
```

CONSOLE | RE-RUN QUERY | EXPORT | OPEN

After Delete Trigger:

6. Create a trigger to increase the salary of a doctor who's assigned patient was removed from the database after treatment:

```
CREATE trigger del_trigger
AFTER DELETE ON patient
FOR EACH ROW
BEGIN
IF OLD.D_id is not null THEN
UPDATE doctor SET Salary=Salary+1000 WHERE D_id=OLD.D_id;
END IF;
END;
```

D_id	Dname	Ph_no	Sal...	Field	Address
Filter...	Filter...	Filter...	Filter	Filter...	Filter...
1	akash	5748364582	500000	Cardiologist	Marol
2	pramod	8965735643	721000	Neurologist	Andheri

D_id	Dname	Ph_no	Salary	Field	Address
Filter.	Filter...	Filter...	Filter.	Filter...	Filter...
1	akash	5748364582	500000	Cardiologist	Marol
2	pramod	8965735643	722000	Neurologist	Andheri

CONCLUSION:	In this experiment, we learned how to implement different type of triggers in MySQL and how to use thr triggers to maintain consistent data across all tables