Types of Triggers –
We can define 6 types of triggers for each table:

1. **AFTER INSERT** activated after data is inserted into the table.
2. **AFTER UPDATE:** activated after data in the table is modified.
3. **AFTER DELETE:** activated after data is deleted/removed from the table.
4. **BEFORE INSERT:** activated before data is inserted into the table.
5. **BEFORE UPDATE:** activated before data in the table is modified.
6. **BEFORE DELETE:** activated before data is deleted/removed from the table.

Examples showing implementation of Triggers:

1. Write a trigger to ensure that no employee of age less than 25 can be inserted in the database.

```
delimiter $$
CREATE TRIGGER  Check_age  BEFORE INSERT ON employee
FOR EACH ROW
BEGIN
IF NEW.age < 25 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR:
      AGE MUST BE ATLEAST 25 YEARS!';
END IF;
END; $$
delimiter;
```

Explanation: Whenever we want to insert any tupple to table ’employee’, then before inserting this tupple to the table, trigger named ‘Check_age’ will be executed. This trigger will check the age attribute. If it is greater then 25 then this tupple will be inserted into the tupple otherwise an error message will be printed stating “ERROR: AGE MUST BE ATLEAST 25 YEARS!”

2. Create a trigger which will work before deletion in employee table and create a duplicate copy of the record in another table employee_backup.

Before writing trigger, we need to create table employee_backup

```
create table employee_backup (employee_no int,
    employee_name varchar(40), job varchar(40),
    hiredate date, salary int,
    primary key(employee_no));

delimiter $$
CREATE TRIGGER Backup BEFORE DELETE ON employee
```

```
FOR EACH ROW
BEGIN
INSERT INTO employee_backup
VALUES (OLD.employee_no, OLD.name,
      OLD.job, OLD.hiredate, OLD.salary);
END; $$
delimiter;
```

Explanation: We want to create a backup table that holds the value of those employees who are no more the employee of the institution. So, we create a trigger named Backup that will be executed before the deletion of any Tupple from the table employee. Before deletion, the values of all the attributes of the table employee will be stored in the table employee_backup.

3. Write a trigger to count number of new tupples inserted using each insert statement.

```
Declare count int
Set count=0;
delimiter $$
CREATE TRIGGER Count_tupples
          AFTER INSERT ON employee
FOR EACH ROW
BEGIN
SET count = count + 1;
END; $$
delimiter;
```

Explanation: We want to keep track of the number of new Tupples in the employee table. For that, we first create a variable ‘count’ and initialize it to 0. After that, we create a trigger named Count_tupples that will increment the value of count after insertion of any new Tupple in the table employee.