**Lab 11 – Working with Triggers**

PostgreSQL **Triggers** are database callback functions, which are automatically performed/invoked when a specified database event occurs.

The following are important points about PostgreSQL triggers −

* PostgreSQL trigger can be specified to fire …
  + **Before** the operation is attempted on a row (before constraints are checked and the INSERT, UPDATE or DELETE is attempted)
  + **After** the operation has completed (after constraints are checked and the INSERT, UPDATE, or DELETE has completed)
  + **Instead of** the operation (in the case of inserts, updates or deletes on a view)

Basic syntax to create a Postgres trigger is…

CREATE TRIGGER trigger\_name [BEFORE|AFTER|INSTEAD OF] event\_name

ON table\_name

[

-- Trigger logic goes here....

];

*An event\_name is INSERT/UPDATE/DELETE*

**This link will support your understanding of what a database TRIGGER is, how to create a TRIGGER, how to use a TRIGGER and when these database objects can become extremely useful.**

<https://www.postgresqltutorial.com/introduction-postgresql-trigger/>

<https://www.tutorialspoint.com/postgresql/postgresql_triggers.htm>

<https://www.edureka.co/blog/triggers-in-sql/>

**Steps for creating and testing a trigger**

Run the following commands in the order they are listed here and examine the results after each step.

============================================================================

-- 1 - create or use existing table

============================================================================

DROP TABLE employee;

DROP TABLE employee\_history;

CREATE TABLE employee ( id smallint NOT NULL, emp\_name character(100) NOT NULL, emp\_email character(20) NOT NULL, emp\_phone character(14));

CREATE TABLE employee\_history ( log\_id smallint NOT NULL);

============================================================================

-- 1.a (prepare sample data for test)

============================================================================

INSERT INTO EMPLOYEE (id, emp\_name, emp\_email, emp\_phone) VALUES (1,'ABC', 'abc@gmail.com', '1234567890');

INSERT INTO EMPLOYEE (id, emp\_name, emp\_email, emp\_phone) VALUES (2,'PQR', 'pqr@gmail.com', '1234567890');

INSERT INTO EMPLOYEE (id, emp\_name, emp\_email, emp\_phone) VALUES (3,'XYZ', 'xyz@gmail.com', '1234567890');

============================================================================

============================================================================

-- 2 - query the data prior to executing the trigger

============================================================================

select \* from employee;

select \* from employee\_history;

============================================================================

-- 3 - create the function (logic to be used when the trigger fires)

============================================================================

CREATE OR REPLACE FUNCTION trigger\_testing()

RETURNS trigger AS $test\_trigger$

BEGIN

INSERT INTO employee\_history (log\_id) VALUES (new.id);

RETURN NEW;

END; $test\_trigger$

LANGUAGE plpgsql;

============================================================================

-- 4 - create the trigger on the table

-- Note that the trigger is only going to fire AFTER INSERT statement

============================================================================

DROP TRIGGER trigger\_test ON employee;

CREATE TRIGGER trigger\_test

AFTER INSERT ON employee

FOR EACH ROW

EXECUTE PROCEDURE trigger\_testing();

============================================================================

-- 5 - test the trigger by performing the DML statement

============================================================================

INSERT INTO EMPLOYEE (id,emp\_name, emp\_email, emp\_phone) VALUES (6,'AXX', 'axx@gmail.com', '1234567890');

INSERT INTO EMPLOYEE (id,emp\_name, emp\_email, emp\_phone) VALUES (7,'AYY', 'ayy@gmail.com', '1234567890');

INSERT INTO EMPLOYEE (id,emp\_name, emp\_email, emp\_phone) VALUES (8,'AZZ', 'azz@gmail.com', '1234567890');

============================================================================

-- 6 - query the data to verify that the trigger fired and the function executed correctly

============================================================================

select \* from employee;

select \* from employee\_history;