|  |  |  |  |
| --- | --- | --- | --- |
| Indicator | Poor | Average | Good |
| Timeliness   * Maintains assignment   deadline (3) | Assignment not done (0) | One or More than One week late (1-2) | Maintains deadline (3) |
| Completeness and neatness   * Complete all parts of   assignment(3) | N/A | < 80% complete  (1-2) | 100%  complete (3) |
| Originality   * Extent of plagiarism(2) | Copied it from someone  else(0) | At least few questions have been done without copying(1) | Assignment has been  solved  completely without  copying (2) |
| Knowledge   * In depth knowledge of the assignment(2) | Unable to answer 2  questions(0) | Unable to answer 1 question (1) | Able to answer 2 questions (2) |

|  |  |
| --- | --- |
| Timeliness |  |
| Completeness and neatness |  |
| Originality |  |
| Knowledge |  |
| Total |  |

|  |
| --- |
| SE Comp A Roll number : 9913 |
| Experiment no. : 9 Date of Implementation : 26/ 03/ 2024 |
| Aim : To implement Functions and Triggers |
| Tool Used : PostgreSQL |
| Related Course outcome : At the end of the course, Students will be able to Use SQL : Standard language of relational database |
| **Rubrics for assessment of Experiment:** |
| **Assessment Marks :** |
| **Total : (Out of 10)** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **Teacher's Sign :** | | | |
|  | ***EXPERIMENT 09*** | ***Functions and Triggers*** |  |
|  | Aim | To implement PL/pgSQL function and trigger |  |
|  | Tools | PostgreSQL  <http://www.postgresqltutorial.com/postgresql-create-function/> <http://www.postgresqltutorial.com/plpgsql-function-overloading/> <http://www.postgresqltutorial.com/plpgsql-function-returns-a-table/> <http://www.postgresqltutorial.com/creating-first-trigger-postgresql/> [PostgreSQL: Documentation: 15: 43.10. Trigger Functions](https://www.postgresql.org/docs/current/plpgsql-trigger.html) |  |

|  |  |  |
| --- | --- | --- |
|  | Theory | CREATE FUNCTION defines a new function. CREATE OR REPLACE FUNCTION will either create a new function, or replace an existing definition. To be able to define a function, the user must have the USAGE privilege on the language. If a schema name is included, then the function is created in the specified schema. Otherwise it is created in the current schema. The name of the new function must not match any existing function with the same input argument types in the same schema. However, functions of different argument types can share a name (this is called *overloading*).  **Syntax for Function**  CREATE [ OR REPLACE ] FUNCTION  name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default\_expr ] [,  ...] ] )  [ RETURNS rettype  | RETURNS TABLE ( column\_name column\_type [, ...] ) ]  { LANGUAGE lang\_name  | WINDOW  | IMMUTABLE | STABLE | VOLATILE  | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT  | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER  | COST execution\_cost  | ROWS result\_rows  | SET configuration\_parameter { TO value | = value | FROM CURRENT }  | AS 'definition'  | AS 'obj\_file', 'link\_symbol'  } ...  [ WITH ( attribute [, ...] ) ]  If you drop and then recreate a function, the new function is not the same entity as the old; you will have to drop existing rules, views, triggers, etc. that refer to the old function. Use CREATE OR REPLACE FUNCTION to change a function definition without breaking objects that refer to the function.  The 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); or after the operation has completed (after constraints are checked and the INSERT, UPDATE, or DELETE has completed); or instead of the operation (in the case of inserts, updates or deletes on a view). If the trigger fires before or instead of the event, the trigger can skip the operation for the current row, or change the row being inserted (for INSERT and UPDATE operations only). If the trigger fires after the event, all changes, including the effects of other triggers, are "visible" to the trigger. |

Syntax of Trigger

CREATE [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [ OR

... ] }

ON table

[ FROM referenced\_table\_name ]

[ NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED }

]

[ FOR [ EACH ] { ROW | STATEMENT } ]

[ WHEN ( condition ) ]

EXECUTE PROCEDURE function\_name ( arguments ) where event can be one of:

INSERT

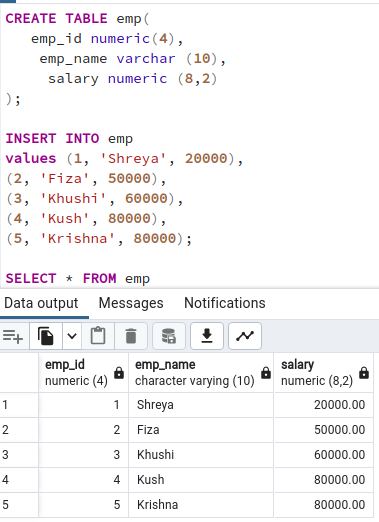
UPDATE [ OF column\_name [, ... ] ] DELETE

TRUNCATE

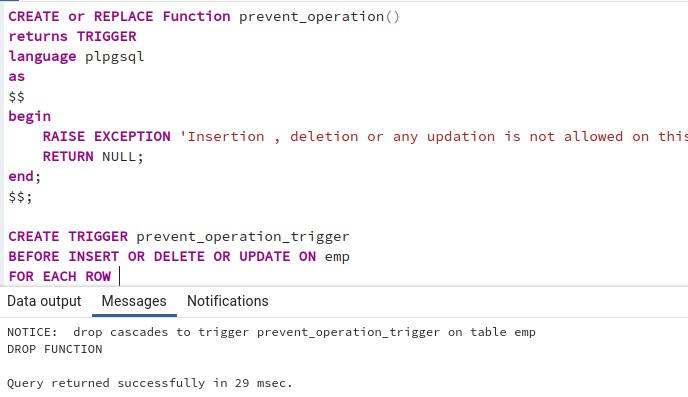
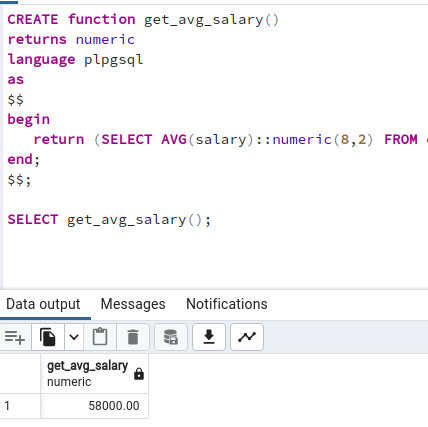
To create a trigger on a table, the user must have the TRIGGER privilege on the table. The user must also have EXECUTE privilege on the trigger function.

Use [DROP TRIGGER](https://www.postgresql.org/docs/9.1/static/sql-droptrigger.html) to remove a trigger.

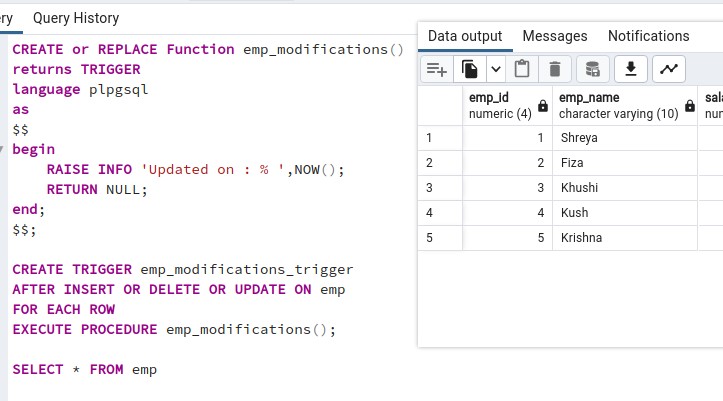
|  |  |  |
| --- | --- | --- |
|  | **AIM** | 1. Write a function to find factorial of a number use it and write the observation for it.      1. Create table emp(id,name,salary) and insert 3 records in it. |



3. Write a function find average salary from emp table



1. Write a row level trigger that would fire before insert/ update/delete operations performed on emp table, not allowing these operations and display the appropriate message.
2. Write a row level trigger that would fire after insert/update/delete operations performed on emp table displaying date on which data manipulation performed.



|  |  |  |
| --- | --- | --- |
|  | **Post Lab Questions:** | 1. Explain syntax of function in Mysql /PostgreSQL with example  ⇒ The general structure of mysql function is: DELIMITER //  CREATE FUNCTION function\_name(parameter INT) RETURNS INT  BEGIN  DECLARE variable\_name INT;  -- Function logic RETURN variable\_name;  END;  // DELIMITER ;  Here is an example of creating a function in MySQL: DELIMITER //  CREATE FUNCTION CalcIncome (starting\_value INT) RETURNS INT  BEGIN  DECLARE income INT; SET income = 0;  label1: WHILE income <= 3000 DO  SET income = income + starting\_value; END WHILE label1;  RETURN income; END;  // DELIMITER ;  **The general structure of a function in postgresql is:**  CREATE FUNCTION somefunc(integer, text) RETURNS integer AS  $$ DECLARE  -- Local variables declaration BEGIN  -- Function logic END;  $$  LANGUAGE plpgsql;  Here is an example of creating a function in PostgreSQL: CREATE FUNCTION totalRecords() RETURNS integer AS $total$ DECLARE  total integer;  BEGIN  SELECT count(\*) INTO total FROM COMPANY; RETURN total; |

END;

$total$ LANGUAGE plpgsql;

2. Explain trigger example with syntax in Mysql/postgreSQL.

# ⇒ MySQL Trigger Example:

In MySQL, to create a trigger, you use the CREATE TRIGGER statement. Below is an example of a trigger that updates a timestamp column whenever a row is inserted into a table:

sql

CREATE TRIGGER update\_timestamp BEFORE INSERT ON table\_name

FOR EACH ROW

SET NEW.timestamp\_column = NOW();

* Explanation:
  + CREATE TRIGGER: Initiates the trigger creation.
  + update\_timestamp: Name of the trigger.
  + BEFORE INSERT ON table\_name: Specifies the trigger to execute before an insert operation on a specific table.
  + FOR EACH ROW: Indicates that the trigger should be executed for each row affected by the operation.

SET NEW.timestamp\_column = NOW(): Sets the timestamp\_column to the current timestamp when a new row is inserted.

# PostgreSQL Trigger Example:

In PostgreSQL, triggers are created using the CREATE TRIGGER statement. Here is an example of a trigger that logs changes made to a specific column in a table:

CREATE TRIGGER log\_changes

AFTER UPDATE OF column\_name ON table\_name FOR EACH ROW

EXECUTE FUNCTION log\_update(); Explanation:

* CREATE TRIGGER: Starts the trigger creation.
* log\_changes: Name of the trigger.
* AFTER UPDATE OF column\_name ON table\_name: Specifies the trigger to execute after an update operation on a specific column in a table.
* FOR EACH ROW: Indicates that the trigger should be executed for each row affected by the operation.
* EXECUTE FUNCTION log\_update(): Calls the log\_update function to log the changes made.