SE IT	Roll number :
Experiment no. : 8	Date of Implementation :
Aim : To implement Functions, Triggers and Cursors	
Tool Used : PostgreSQL	
Related Course outcome : At the end of the course, Students should be able to 4. Write queries in SQL to retrieve any type of information from a data base	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment(3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism(2)	Copied it from someone else(0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment(2)	Unable to answer any questions(0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks :

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Total : **(Out of 10)**

Teacher's Sign :

EXPERIMENT 8	*Functions and Triggers*
Aim	To implement PL/pgSQL function and trigger
Tools	PostgreSQL

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 to remove a trigger.
	
	Cursors:
	A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.
	
	Types of cursors
	Implicit cursors
	• Created by default when statements like, INSERT, UPDATE, and DELETE executed.
	• Also created when a SELECT statement that returns just one row.
	Explicit cursors
	• They must be created when you are executing a SELECT statement that returns more than one row.
	• Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row.
	• When you fetch a row the current row position moves to next row.
	
	Cursor Syntax
	    DECLARE
	    variables;
	    records;
	    create a cursor;
	    BEGIN
	    OPEN cursor;
	    FETCH cursor;
	    process the records;
	    CLOSE cursor;
	    END;

Procedure	1. Write a function to find factorial of a number 2. Create table emp(id,name,salary) and insert 3 records in it. 3. Write a function find average salary from emp table 4. 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. 5. Write a row level trigger that would fire after insert/update/delete operations performed on emp table displaying date on which data manipulation performed. 6. Create an explicit cursor to fetch all employee records having

	salary above 50000/-
Post Lab Questions:	1. Explain TCL commands 2. Explain DCL commands