**22F3661**

**Imama Kainat**

**SE-4A**

**My notes (Trigger)**

A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs.

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

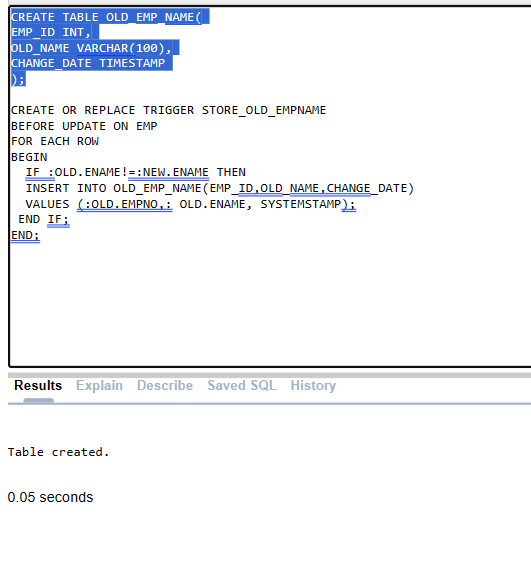
[trigger\_body]

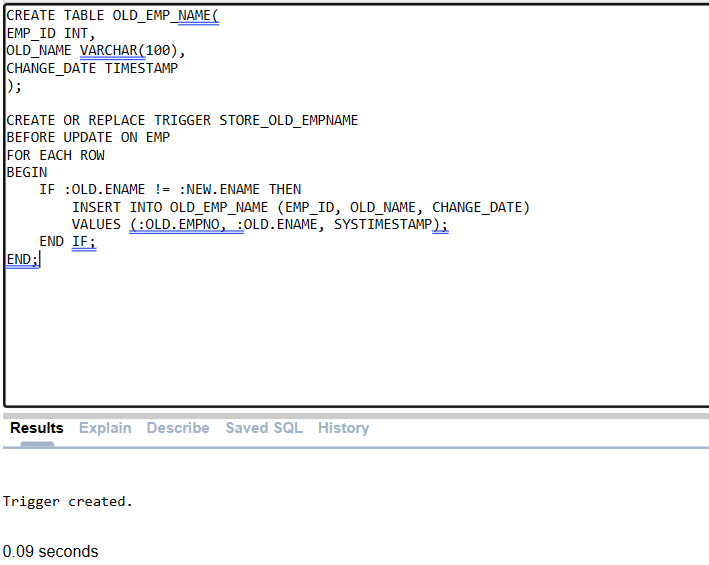
**Explanation of Syntax**

1. Create trigger [trigger\_name]: Creates or replaces an existing trigger with the trigger\_name.
2. [before | after]: This specifies when the trigger will be executed.
3. {insert | update | delete}: This specifies the DML operation.
4. On [table\_name]: This specifies the name of the table associated with the trigger.
5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each affected row.
6. [trigger\_body]: This provides the operation to be performed as the trigger is fired

**Create a trigger that will fire when employee’s name is about to update and store old value**

**in another table before new updating in employee table.**





CREATE TABLE OLD\_EMP\_NAME(

EMP\_ID INT,

OLD\_NAME VARCHAR(100),

CHANGE\_DATE TIMESTAMP

);

CREATE OR REPLACE TRIGGER STORE\_OLD\_EMPNAME

BEFORE UPDATE ON EMP

FOR EACH ROW

BEGIN

IF :OLD.ENAME != :NEW.ENAME THEN

INSERT INTO OLD\_EMP\_NAME (EMP\_ID, OLD\_NAME, CHANGE\_DATE)

VALUES (:OLD.EMPNO, :OLD.ENAME, SYSTIMESTAMP);

END IF;

END;

**QUESTION 2**

**Create a trigger that will fire if the increase in an employee salary is greater than 10% of his**

**salary. Trigger will display the old &amp; new salary of the employee with his complete info.**

CREATE OR REPLACE TRIGGER check\_salary

BEFORE UPDATE OF SAL ON EMP

FOR EACH ROW

DECLARE

SAL\_INCREASE NUMBER;

BEGIN

IF :NEW.SAL > :OLD.SAL THEN

SAL\_INCREASE := ((:NEW.SAL - :OLD.SAL) / :OLD.SAL) \* 100;

IF SAL\_INCREASE > 10 THEN

DBMS\_OUTPUT.PUT\_LINE('HIRE DATE: ' || :OLD.HIREDATE);

DBMS\_OUTPUT.PUT\_LINE('OLD SALARY: ' || :OLD.SAL);

DBMS\_OUTPUT.PUT\_LINE('NEW SALARY: ' || :NEW.SAL);

DBMS\_OUTPUT.PUT\_LINE('DEPARTMENT NUMBER: ' || :OLD.DEPTNO);

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE ID: ' || :OLD.EMPNO);

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE NAME: ' || :OLD.ENAME);

DBMS\_OUTPUT.PUT\_LINE('JOB: ' || :OLD.JOB);

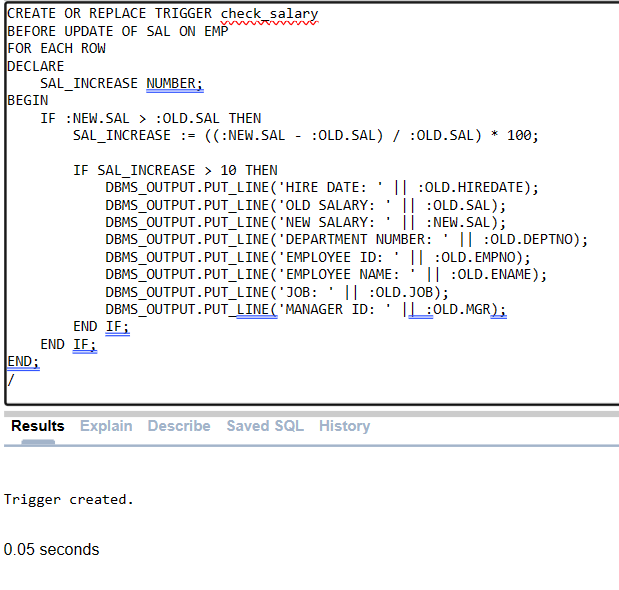
DBMS\_OUTPUT.PUT\_LINE('MANAGER ID: ' || :OLD.MGR);

END IF;

END IF;

END;

/

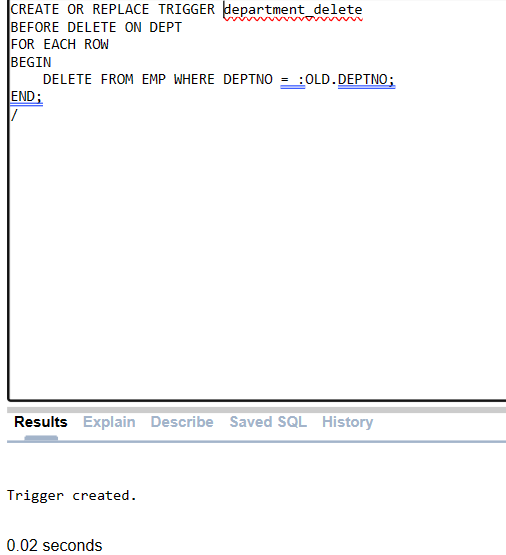


**QUESTION 3**

**Create a trigger that will fire when a department is about to delete. Trigger will fire before**

**the deletion query of department and will delete record of all the employees belonging to**

**that department.**



CREATE OR REPLACE TRIGGER department\_delete

BEFORE DELETE ON DEPT

FOR EACH ROW

BEGIN

DELETE FROM EMP WHERE DEPTNO = :OLD.DEPTNO;

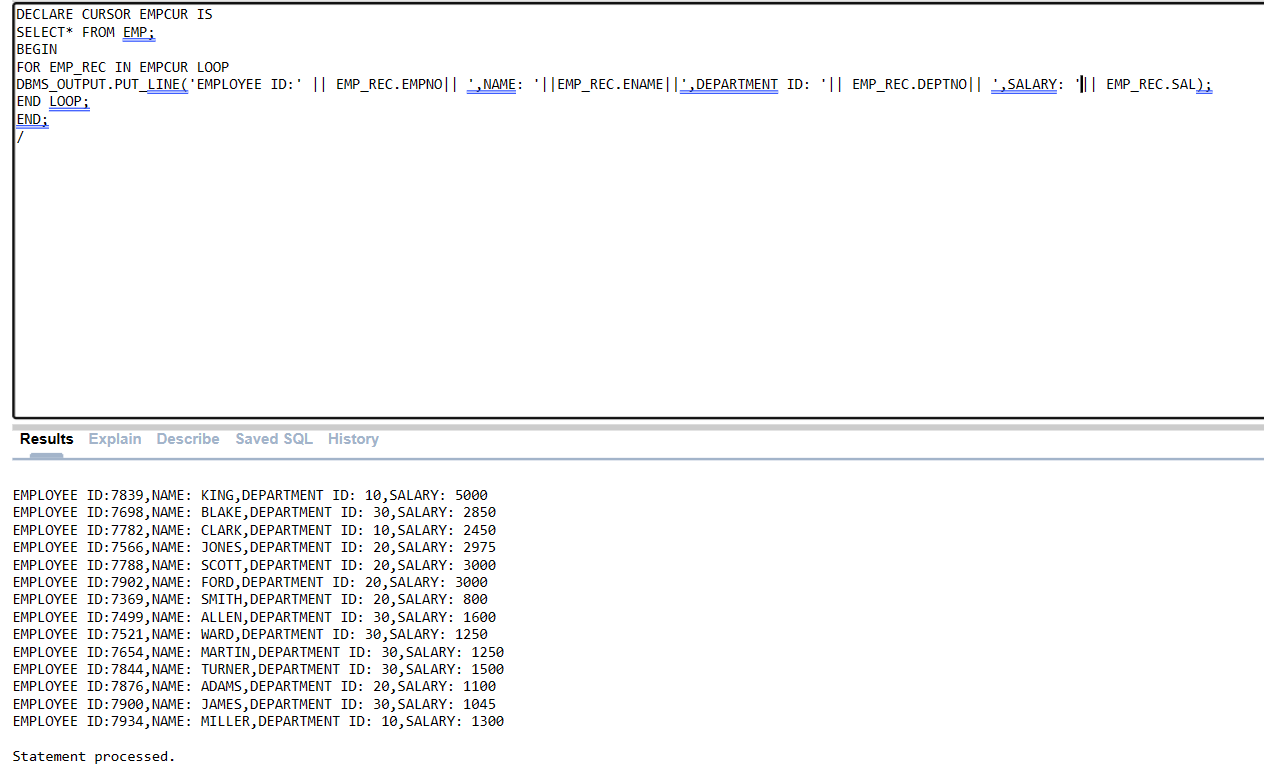
END;

/

**CURSOR**

**QUESTION 1**

**Create a PLSQL query to display complete information of table employee by using Explicit Cursor.**



DECLARE CURSOR EMPCUR IS

SELECT\* FROM EMP;

BEGIN

FOR EMP\_REC IN EMPCUR LOOP

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE ID:' || EMP\_REC.EMPNO|| ',NAME: '||EMP\_REC.ENAME||',DEPARTMENT ID: '|| EMP\_REC.DEPTNO|| ',SALARY: '|| EMP\_REC.SAL);

END LOOP;

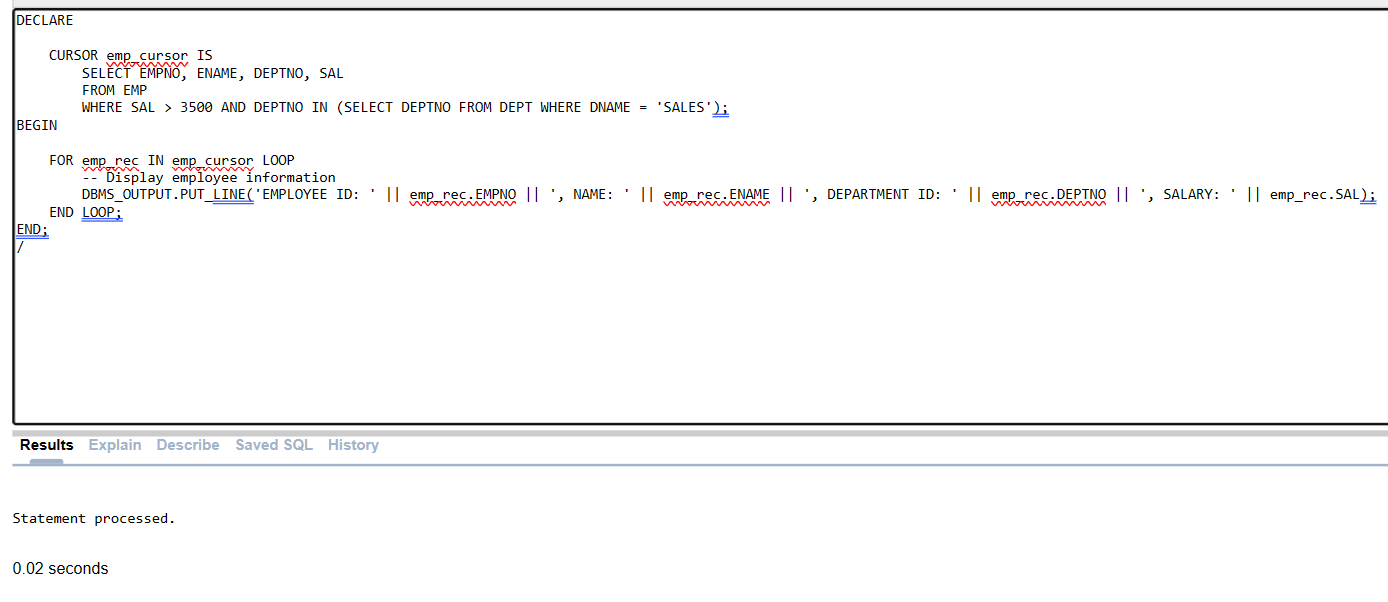
END;

/

**QUESTION 2**

**Create a cursor to display the records of the employees whose salary is greater than 3500 AND are**

**working in Sales Department. (By using Cursor for Loop)**



DECLARE

CURSOR emp\_cursor IS

SELECT EMPNO, ENAME, DEPTNO, SAL

FROM EMP

WHERE SAL > 3500 AND DEPTNO IN (SELECT DEPTNO FROM DEPT WHERE DNAME = 'SALES');

BEGIN

FOR emp\_rec IN emp\_cursor LOOP

-- Display employee information

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE ID: ' || emp\_rec.EMPNO || ', NAME: ' || emp\_rec.ENAME || ', DEPARTMENT ID: ' || emp\_rec.DEPTNO || ', SALARY: ' || emp\_rec.SAL);

END LOOP;

END;

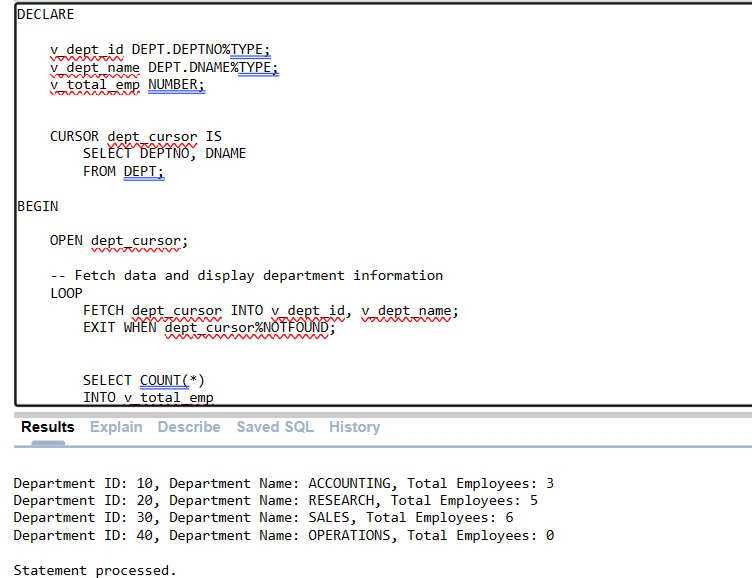
/

**QUSTION 3**

**Write a cursor that selects all the departments from the department table and displays their**

**department\_id, department\_name, and the total number of employees in each department. (by using**

**Cursor open-fetch method).**



DECLARE

v\_dept\_id DEPT.DEPTNO%TYPE;

v\_dept\_name DEPT.DNAME%TYPE;

v\_total\_emp NUMBER;

CURSOR dept\_cursor IS

SELECT DEPTNO, DNAME

FROM DEPT;

BEGIN

OPEN dept\_cursor;

-- Fetch data and display department information

LOOP

FETCH dept\_cursor INTO v\_dept\_id, v\_dept\_name;

EXIT WHEN dept\_cursor%NOTFOUND;

SELECT COUNT(\*)

INTO v\_total\_emp

FROM EMP

WHERE DEPTNO = v\_dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || v\_dept\_id || ', Department Name: ' || v\_dept\_name || ', Total Employees: ' || v\_total\_emp);

END LOOP;

-- Close cursor

CLOSE dept\_cursor;

END;

/