ST. FRANCIS INSTITUTE OF TECHNOLOGY MT. POINSUR, BORIVALI (W), MUMBAI

LAB MANUAL

EXPERIMENT NO. 7

Aim: - Perform View and Triggers in SQL.

Theory:-

1. Views. Its advantages and dis advantages
2. Triggers.

Lab Manual:

Views: In database theory, a view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary.

SQL CREATE VIEW Statement:

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

SQL CREATE VIEW Syntax:

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition;
```

Note: A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

SQL CREATE VIEW Examples

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued=No;
```

We can query the view above as follows:

```
SELECT * FROM [Current Product List];
```

SQL Updating a View:

You can update a view by using the following syntax:

SQL CREATE OR REPLACE VIEW Syntax

```
CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

SQL Dropping a View:
You can delete a view with the DROP VIEW command.

SQL DROP VIEW Syntax

```
DROP VIEW view_name;
```

Triggers:

Triggers are similar to stored procedures. A trigger stored in the database can include SQL and PL/SQL or Java statements to run as a unit and can invoke stored procedures. However, procedures and triggers differ in the way that they are invoked. A procedure is explicitly run by a user, application, or trigger. Triggers are implicitly fired by Oracle when a triggering event occurs, no matter which user is connected or which application is being used. A trigger can also call out to a C procedure, which is useful for computationally intensive operations. The events that fire a trigger include the following:

· DML statements that modify data in a table (INSERT, UPDATE, or DELETE) ·
DDL statements

· System events such as startup, shutdown, and error messages

· User events such as logon and logoff

How Triggers Are Used

Triggers supplement the standard capabilities of Oracle to provide a highly customized database management system. For example, a trigger can restrict DML operations against a table to those issued during regular business hours. You can also use triggers to:

· Automatically generate derived column values

· Prevent invalid transactions

· Enforce complex security authorizations

· Enforce referential integrity across nodes in a distributed database

· Enforce complex business rules

· Provide transparent event logging

· Provide auditing

· Maintain synchronous table replicates

· Gather statistics on table access

· Modify table data when DML statements are issued against views

· Publish information about database events, user events, and SQL statements to subscribing applications

There are three components in trigger

Event: When this event happens, the trigger is activated

Condition (optional): If the condition is true, the trigger executes, otherwise skipped Action: The actions performed by the trigger

Semantics: When the Event occurs and Condition is true, execute the Action

Trigger syntax:

```
CREATE TRIGGER <triggerName>
BEFORE|AFTER INSERT|DELETE|UPDATE
    [OF <columnList>] ON <tableName>|<viewName>
    [REFERENCING [OLD AS <oldName>] [NEW AS <newName>]]
[FOR EACH ROW] (default is "FOR EACH STATEMENT")
[WHEN (<condition>)]
< trigger body>;
```

In SQL*Plus, you can also use the following shortcut to view compilation errors:

SQL> **SHOW ERRORS TRIGGER** MY_TRIGGER

ST. FRANCIS INSTITUTE OF TECHNOLOGY

MT. POINSUR, BORIVALI (W), MUMBAI

Lab Exercise on Views:-

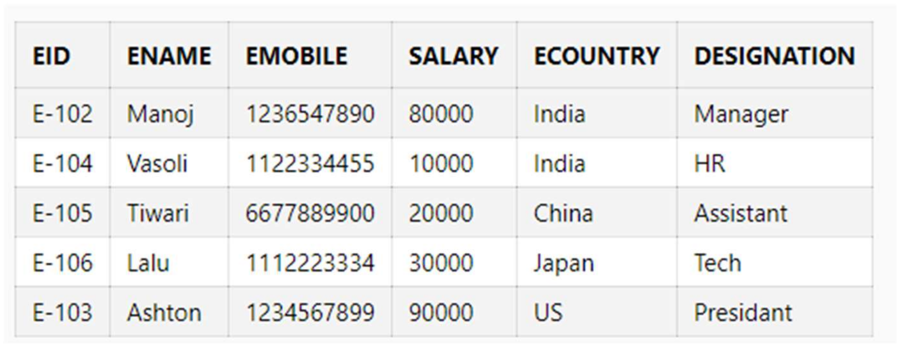
1. Create table Employee having eid varchar (5), ename varchar(15), emobile number(10), salary number(10), ecountry varchar(10), designation varchar(10).

```
create table Employees(eid varchar(5),ename varchar(15),emobile number(10), salary number(10), ecountry varchar(10), designation varchar(10))
```

```
Table created.
```

2. Insert 5 values inside the Employee table.

```
select * from Employees
```



EID	ENAME	EMOBILE	SALARY	ECOUNTRY	DESIGNATION
E-102	Manoj	1236547890	80000	India	Manager
E-104	Vasoli	1122334455	10000	India	HR
E-105	Tiwari	6677889900	20000	China	Assistant
E-106	Lalu	1112223334	30000	Japan	Tech
E-103	Ashton	1234567899	90000	US	Presidant

3. Create view India_emp_view, list all from Employee table where ecountry='India'

```
create view XYZ as select * from employees where ecountry = 'India'
```

```
View created.
```

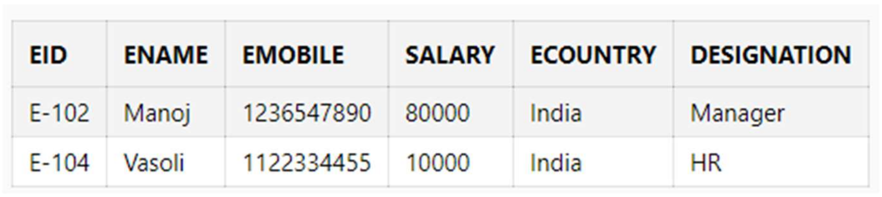
4. Create view Manager_emp_view, list all details for employee where designation is manager.

```
create view Manager_Emp_View as select * from employees where designation='Manager'
```

```
View created.
```

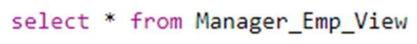
5. Display India_emp_view

```
select * from xyz
```

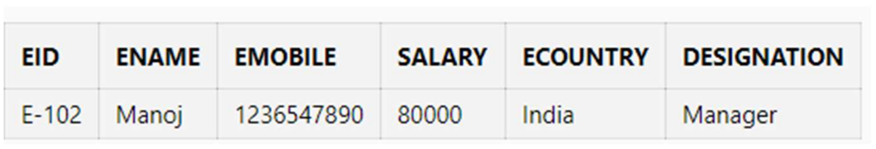


EID	ENAME	EMOBILE	SALARY	ECOUNTRY	DESIGNATION
E-102	Manoj	1236547890	80000	India	Manager
E-104	Vasoli	1122334455	10000	India	HR

6. Display Manager_emp_view.

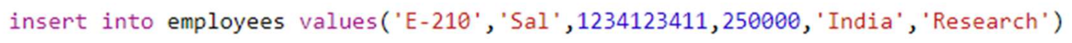


```
select * from Manager_Emp_View
```

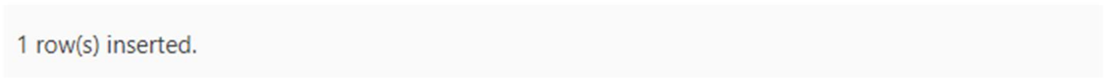


EID	ENAME	EMOBILE	SALARY	ECOUNTRY	DESIGNATION
E-102	Manoj	1236547890	80000	India	Manager

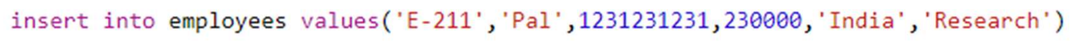
7. Insert 2 tuples in Employee where country of employee is India, and then check no. of records in India_emp_view.



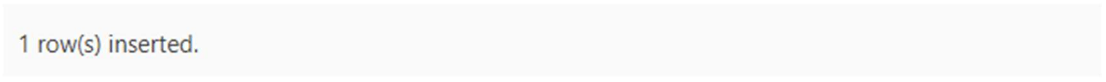
```
insert into employees values('E-210','Sal',1234123411,250000,'India','Research')
```



```
1 row(s) inserted.
```

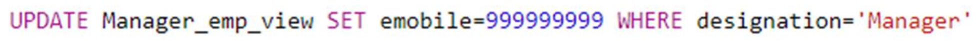


```
insert into employees values('E-211','Pal',1231231231,230000,'India','Research')
```

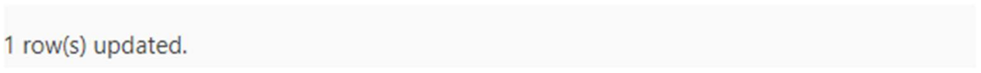


```
1 row(s) inserted.
```

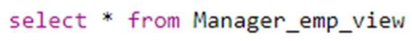
8. Update Manager_emp_view, update its emobile, and then check the record in Employee table.



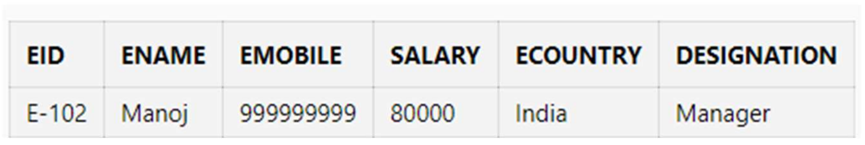
```
UPDATE Manager_emp_view SET emobile=999999999 WHERE designation='Manager'
```



```
1 row(s) updated.
```



```
select * from Manager_emp_view
```

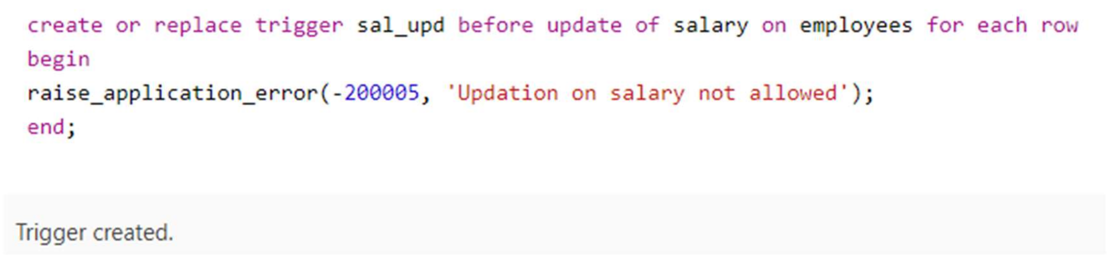


EID	ENAME	EMOBILE	SALARY	ECOUNTRY	DESIGNATION
E-102	Manoj	999999999	80000	India	Manager

Lab Exercise on Triggers:-

1. For Relational Schema Employee (Eid, ename, emobile, salary, ecountry, designation),create following triggers:

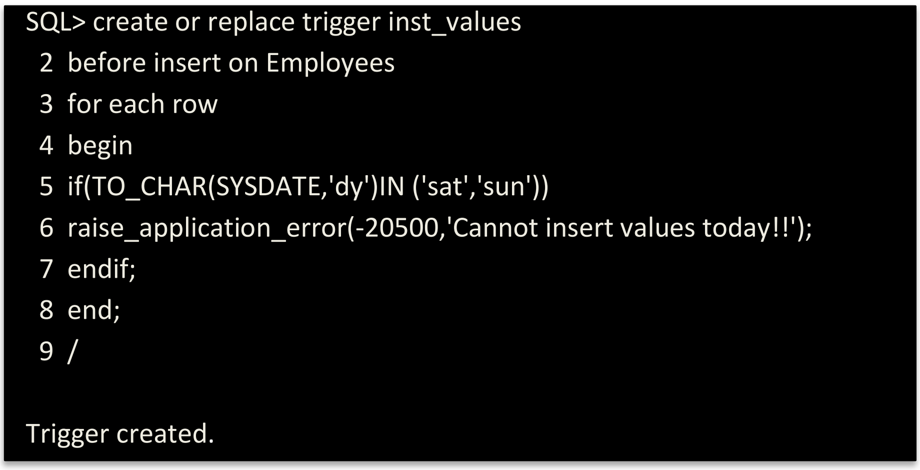
   a. Write a trigger to avoid updating on Salary attribute for employee relation.



```
create or replace trigger sal_upd before update of salary on employees for each row
begin
raise_application_error(-200005, 'Updation on salary not allowed');
end;
```

```
Trigger created.
```

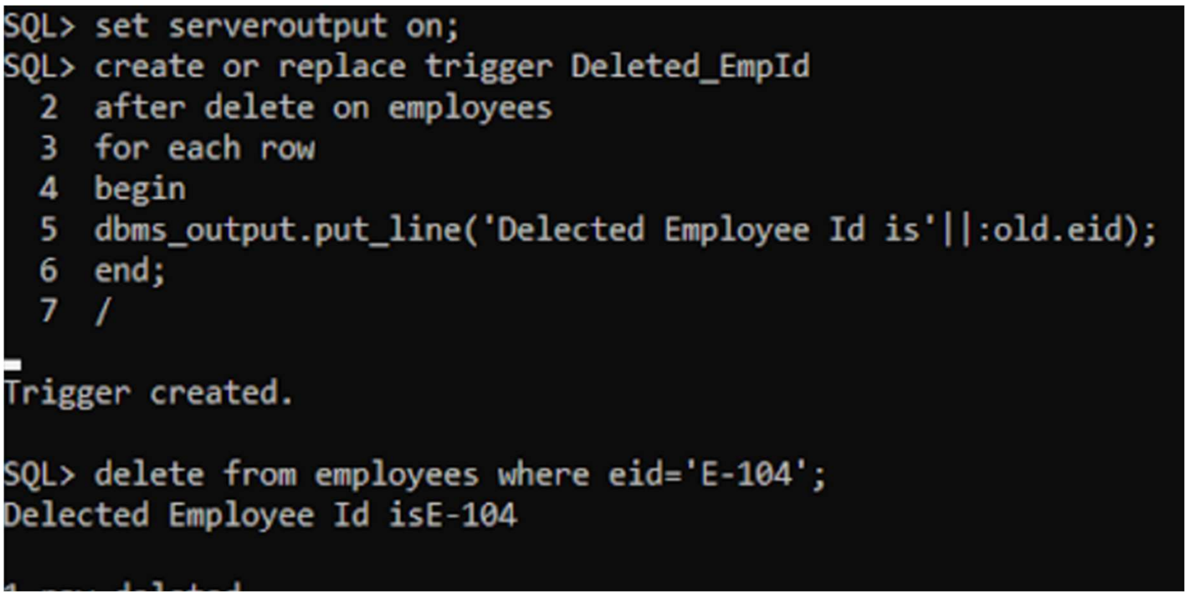
   b. Write a trigger to avoid insert on employee relation on Weekends.



```
SQL> create or replace trigger inst_values
before insert on Employees
for each row
begin
if(TO_CHAR(SYSDATE,'dy')IN ('sat','sun'))
raise_application_error(-20500,'Cannot insert values today!!');
endif;
end;
/

Trigger created.
```

   c. Write a trigger that displays the employee id for the record which gets deleted.

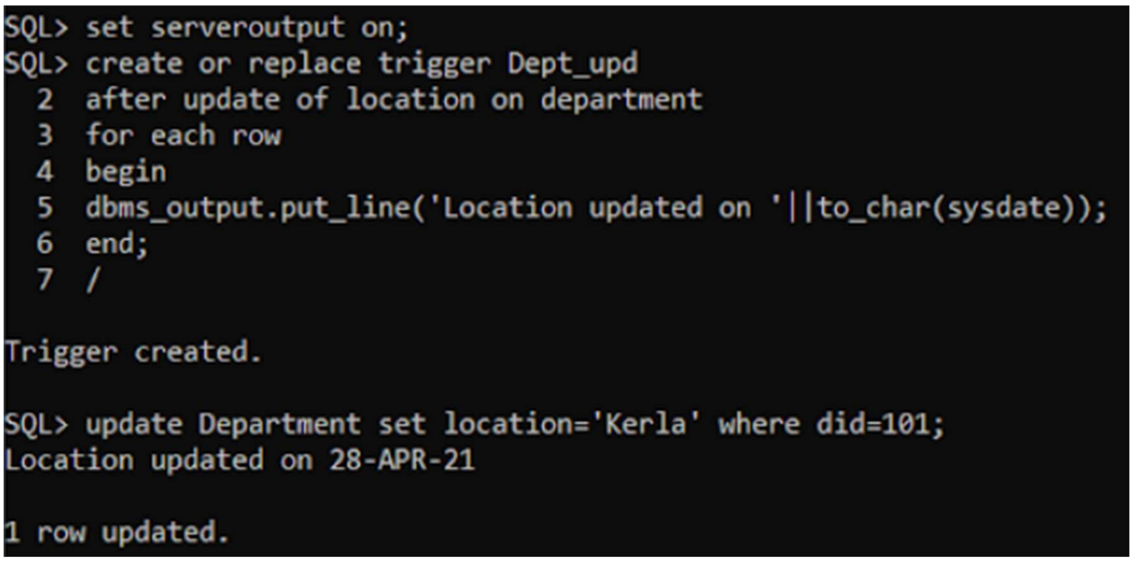


```
SQL> set serveroutput on;
SQL> create or replace trigger Deleted_EmpId
after delete on employees
for each row
begin
dbms_output.put_line('Delected Employee Id is'||:old.eid);
end;
/

Trigger created.

SQL> delete from employees where eid='E-104';
Delected Employee Id isE-104
```

2. For Relational Schema Department (Did, Dname, Location,Dmgr),create following triggers:
   a. Write a trigger that displays the system date whenever there is an update on Location attribute for department relation.



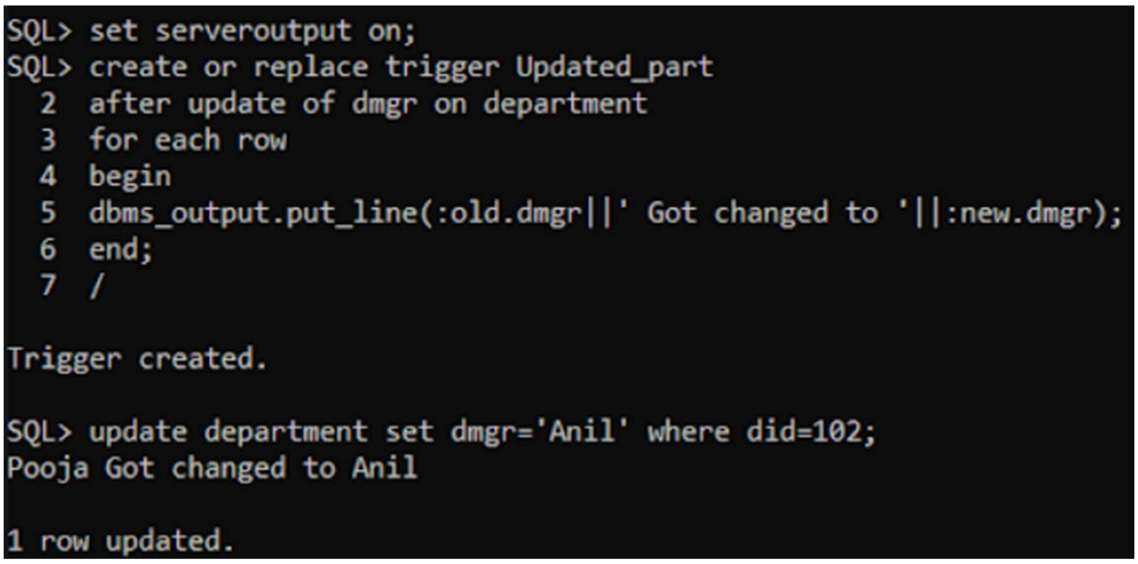
```
SQL> set serveroutput on;
SQL> create or replace trigger Dept_upd
  2  after update of location on department
  3  for each row
  4  begin
  5  dbms_output.put_line('Location updated on '||to_char(sysdate));
  6  end;
  7  /

Trigger created.

SQL> update Department set location='Kerla' where did=101;
Location updated on 28-APR-21

1 row updated.
```

   b. Write a trigger that outputs a statement stating old name which got updated by the new name whenever the Dmgr gets updated for department relation.



```
SQL> set serveroutput on;
SQL> create or replace trigger Updated_part
  2  after update of dmgr on department
  3  for each row
  4  begin
  5  dbms_output.put_line(:old.dmgr||' Got changed to '||:new.dmgr);
  6  end;
  7  /

Trigger created.

SQL> update department set dmgr='Anil' where did=102;
Pooja Got changed to Anil

1 row updated.
```

Conclusion:

Views and triggers are created in the above experiment. Views in SQL are a kind of virtual table. A View can either have all the rows of a table or specific rows based on certain conditions. Triggers are stored programs, which are automatically executed or fired when some events occur.