Experiment No.8
Implementation of Views and Triggers.
Date of Performance:
Date of Submission:

Aim :- Write a SQL query to implement views and triggers

Objective :- To learn about virtual tables in the database and also PLSQL constructs

Theory:

SQL Views:

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 statements and functions to a view and present the data as if the data were coming from one single table.
A view is created with the CREATE VIEW statement.

CREATE VIEW syntax

CREATE VIEW view name AS

SELECT columni, column2, ...

FROM table name

WHERE condition;

SQL Updating a View

A view can be updated with the CREATE OR REPLACE VIEW statement.

SQL CREATE OR REPLACE VIEW Syntax

CREATE OR REPLACE VIEW view name AS

SELECT columnl, column2,...

FROM table name

WHERE condition;

SQL Dropping a View

A view is deleted with the DROP VIEW statement.

SQL DROP VIEW syntax

DROP VIEW view name;

Trigger: A trigger is a stored procedure in the database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Syntax:

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 row being affected.
6. [trigger_body]: This provides the operation to be performed as trigger is fired

Implementation:

SQL View:

1)Create View:

```
show databases;
use hospitalmgmt;
show tables;
DESC PATIENTS;
CREATE VIEW PatientInfo AS
SELECT Patient_id,Name,Doctor_id
FROM PATIENTS;
SELECT * FROM PatientInfo;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Patient_id	Name	Doctor_id
201	John Doe	101
202	Jane Smith	102
203	Michael Johnson	103
204	Emily Brown	104

PatientInfo 8 ×

Output

Action Output

#	Time	Action	Message
11	12:35:15	CREATE VIEW PatientInfo AS SELECT Patient_id,Name,Doctor_id FROM PATIENTS	0 row(s) affected
12	12:35:22	SELECT * FROM PatientInfo LIMIT 0, 1000	4 row(s) returned

```
22 •    SELECT * FROM EmployeeInfo;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EmployeeID	FirstName	LastName	Department	Position
1	John	Doe	Sales	Sales Representative
2	Jane	Smith	Marketing	Marketing Manager
3	Alice	Johnson	HR	HR Coordinator
4	Bob	Williams	Finance	Financial Analyst
5	Emily	Brown	IT	Software Developer

2)Drop View:

```
DROP VIEW IF EXISTS EmployeeInfo;
SELECT * FROM EmployeeInfo;
```

✓	9	11:52:00	DROP VIEW IF EXISTS EmployeeInfo	0 row(s) affected
✗	10	11:52:15	SELECT * FROM EmployeeInfo LIMIT 0, 1000	Error Code: 1146. Table 'office_management.employeeinfo' doesn't exist

SQL Trigger:

```
CREATE TRIGGER capitalize_patient_name
BEFORE INSERT ON PATIENTS
FOR EACH ROW
    SET NEW.name = CONCAT(UPPER(LEFT(NEW.name, 1)), LOWER(SUBSTRING(NEW.name, 2)));
```

Patient_id	Name	Doctor_id
201	John Doe	101
202	Jane Smith	102
203	Michael Johnson	103
204	Emily Brown	104

tientInfo 9 ×

utput

Action Output

#	Time	Action	Message
16	12:42:08	CREATE TRIGGER capitalize_patient_name BEFORE INSERT ON PATIENTS FOR EACH ROW SET NEW.name = CONCAT(U...	0 row(s) affected
17	12:42:51	SELECT * FROM PatientInfo LIMIT 0, 1000	4 row(s) returned

Conclusion:

1. Brief about the benefits for using views and triggers.

Ans.: Views simplify queries, enhance security, abstract table structures, and optimize performance. Triggers enforce data integrity, audit changes, enforce business logic, and support replication.

2. Explain different strategies to update views.

Ans.: Updating views can be done directly, by updating base tables, using triggers, or by recreating views. These methods offer varying degrees of control and are applied based on the view's complexity and update requirements.