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

Experiment No.8

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 column1, 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 column1, 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:

```
CREATE VIEW PilotInfo AS
SELECT airplane_id, model
FROM Airplane;
SELECT * FROM PilotInfo;
```

12 22:35:31 CREATE VIEW PilotInfo AS SELECT airplane_id, model FROM Airplane

airplane_id	model
22546	BX1221
22547	BX1231
22552	BX1521
22563	BX1621

2.) Drop View:

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

14 22:40:07 DROP VIEW IF EXISTS PilotInfo

SQL Trigger:

```
CREATE TRIGGER capitalize_name_before_insert
BEFORE INSERT ON AIRPLANE
FOR EACH ROW
BEGIN
    SET NEW.FirstName = CONCAT(UPPER(SUBSTRING(NEW.FirstName,1 ,1)), LOWER(SUBSTRING(NEW.FirstName, 2)));
    SET NEW.LastName = CONCAT(UPPER(SUBSTRING(NEW>LastName, 1, 1)), LOWER(SUBSTRING(NEW.LastName, 2)));
END;

INSERT INTO AIRPLANE (AirplaneID, FirstName, LastName)
VALUES ('1213', 'Chris', 'Dsouza')
```

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.