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 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]

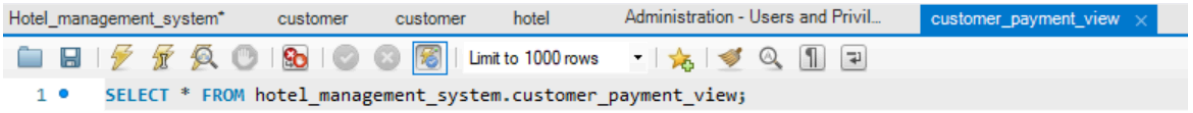
Explanation of syntax:

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 : View

```
CREATE VIEW customer_payment_view AS
SELECT c.Custermor_id, c.Cust_name, c.cu_address, c.phone_no, c.aadhar_no, c.DOB,
       p.payement_id, p.amount, p.payment_type
FROM customer c
JOIN payement p ON c.Custermor_id = p.payement_id;
```

Output:



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

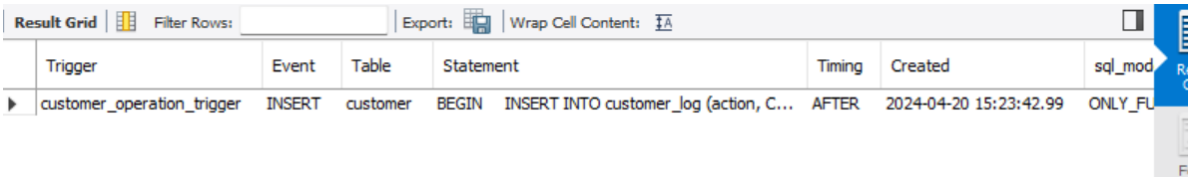
Custermor_id	Cust_name	cu_address	phone_no	aadhar_no	DOB	payement_id	amount	payment_type
1	John Doe	123 Main St	1234567	123456789	1990-01-01	1	500	Credit Card
2	Jane Smith	456 Elm St	987654	98765432	1985-05-15	2	750	Debit Card
3	Alice Johnson	789 Oak St	555123	55512345	2000-10-20	3	1000	Cash
4	Bob Brown	321 Pine St	999888	99988877	1978-12-30	4	300	Online Transfer
5	Emily Wilson	654 Birch St	11122	11122233	1995-08-25	5	900	Cheque

Impelentaion : Trigger

```
DELIMITER $$
CREATE TRIGGER customer_operation_trigger
AFTER INSERT ON customer
FOR EACH ROW
BEGIN
    INSERT INTO customer_log (action, Custermor_id, Cust_name, cu_address, phone_no, aadhar_no, DOB)
    VALUES ('INSERT', NEW.Custermor_id, NEW.Cust_name, NEW.cu_address, NEW.phone_no, NEW.aadhar_no, NEW.DOB);
END$$
DELIMITER ;

SHOW TRIGGERS;
```

Output:



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

Trigger	Event	Table	Statement	Timing	Created	sql_mod
customer_operation_trigger	INSERT	customer	BEGIN INSERT INTO customer_log (action, C...	AFTER	2024-04-20 15:23:42.99	ONLY_FU

Conclusion:

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

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

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.