



**Vidyavardhini's College of Engineering and Technology**

**Department of Artificial Intelligence & Data Science**

---

<b>Experiment No.8</b>
Implementation of Views and Triggers
Date of Performance:
Date of Submission:



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

**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;



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

**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

### VIEWS:-

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The SQL editor contains the following queries:

```
3 SELECT user_id, username, email
4 FROM social_media_db.users;
5 • SELECT * FROM user_view;
6
7
8
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input and an 'Export' button. The grid displays the following data:

	user_id	username	email
▶	107	Aaru	aradhya@gmail.com
	108	Ekansh	ansha@gmail.com
	109	Luv	luv@gmail.com
	112	Sans	sanskriti@gmail.com
	113	Parnu	Parnika@gmail.com
	123	Roohi	roohi@gmail.com
	124	Shreya	shreya@gmail.com



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

### TRIGGERS:-

Limit to 1000 rows

```
17 • CREATE TRIGGER after_insert_users
18 AFTER INSERT ON users
19 FOR EACH ROW
20 BEGIN
21     INSERT INTO user_log (user_id, username, email)
22     VALUES (NEW.user_id, NEW.username, NEW.email);
23 END//
24
25 DELIMITER ;
```

Result Grid

	user_id	username	email	password	created_at
▶	107	Aaru	aradhya@gmail.com	wewy2vduqw	2024-04-22 19:41:14
	108	Ekansh	ansha@gmail.com	wewqwy2vduqw	2024-04-22 19:13:09
	109	Luv	luv@gmail.com	wadewy2vduqwwr4c	2024-04-22 19:47:54
	112	Sans	sanskriti@gmail.com	wewy2vggduqw	2024-04-22 19:46:10
	113	Parnu	Parnika@gmail.com	wewy2vdue5qw	2024-04-22 19:44:57
	123	Roohi	roohi@gmail.com	sbfd62435	2024-04-23 01:03:24
	124	Shreya	shreya@gmail.com	sbfd62006	2024-04-23 01:03:05
*	NULL	NULL	NULL	NULL	NULL



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

### Conclusion:

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

##### triggers. Views:

- a) Simplified Data Access\*\*: Views allow users to access and retrieve data from multiple tables using a single, simplified interface. This can enhance productivity by reducing the complexity of queries.
- b) Data Security\*\*: Views can be used to restrict access to specific columns or rows of data, providing a layer of security by hiding sensitive information from unauthorized users.
- c) Data Abstraction\*\*: Views abstract the underlying structure of the database, allowing changes to the database schema without affecting applications that rely on the views. This enhances flexibility and maintainability.
- d) Performance Optimization\*\*: Views can be precomputed and cached, improving query performance by reducing the need for repetitive joins or complex calculations.

##### Triggers:

- a) Data Integrity Enforcement\*\*: Triggers can enforce data integrity rules by automatically performing actions, such as validation checks or data modifications, before or after data manipulation operations (INSERT, UPDATE, DELETE).
- b) Business Logic Implementation\*\*: Triggers can implement complex business logic directly in the database, ensuring consistency and enforcing business rules across applications.
- c) Auditing and Logging\*\*: Triggers can be used to log changes made to the database, providing an audit trail of data modifications for compliance or troubleshooting purposes.
- d) Data Synchronization\*\*: Triggers can synchronize data between tables or databases, ensuring consistency and maintaining data integrity across distributed systems.

#### 2) Explain different strategies to update views

##### a) Simple Views:

In some database systems, you can update a view if it is based on a single table and does not contain any aggregate functions or grouping.

The update is directly applied to the underlying table(s) that the view references.

##### b) Updatable Views:

Some views are explicitly designed to be updatable, allowing you to update data through the view itself.

These views must meet certain criteria, such as:

- Contain all columns from a single base table.

- Not include aggregate functions, GROUP BY clauses, or DISTINCT.

- Not contain joins, subqueries in the SELECT list, or set operations (UNION, INTERSECT,



EXCEPT).

c) Instead Of Triggers:

Instead Of triggers are used to enable updates on non-updatable views by intercepting the attempted update operations and providing custom logic to handle them.

These triggers can be defined to execute custom insert, update, or delete operations on the underlying tables when the view is modified.

d) Materialized Views:

Materialized views are physical copies of query results stored as tables.

They can be updated by refreshing the materialized view, which recalculates the data based on the underlying tables.

The refresh operation can be performed manually or scheduled to occur automatically at specified intervals.