



# Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

|                     |               |                     |                                       |
|---------------------|---------------|---------------------|---------------------------------------|
| <b>Class:</b>       | <b>SE</b>     | <b>Semester:</b>    | <b>IV</b>                             |
| <b>Course Code:</b> | <b>CSL402</b> | <b>Course Name:</b> | <b>Database Management System Lab</b> |

|                                 |   |
|---------------------------------|---|
| <b>Name of Student:</b>         | <b>Sharvari Bhondekar</b>                   |
| <b>Roll No. :</b>               | <b>7</b>                                    |
| <b>Experiment No.:</b>          | <b>8</b>                                    |
| <b>Title of the Experiment:</b> | <b>Implementation of Views and Triggers</b> |
| <b>Date of Performance:</b>     | <b>20/02/25</b>                             |
| <b>Date of Submission:</b>      | <b>06/03/25</b>                             |

## Evaluation

| <b>Performance Indicator</b>       | <b>Max. Marks</b> | <b>Marks Obtained</b> |
|------------------------------------|-------------------|-----------------------|
| Performance                        | 5                 |                       |
| Understanding                      | 5                 |                       |
| Journal work and timely submission | 10                |                       |
| <b>Total</b>                       | <b>20</b>         |                       |

| <b>Performance Indicator</b>       | <b>Exceed Expectations (EE)</b> | <b>Meet Expectations (ME)</b> | <b>Below Expectations (BE)</b> |
|------------------------------------|---------------------------------|-------------------------------|--------------------------------|
| Performance                        | 4-5                             | 2-3                           | 1                              |
| Understanding                      | 4-5                             | 2-3                           | 1                              |
| Journal work and timely submission | 8-10                            | 5-8                           | 1-4                            |

Checked by

Name of Faculty : Ms. Neha Raut

Signature :

Date:



# Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

---

## 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;
```



# 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:**

For VIEWS:

**Code:**

|   | payment_id | ticket_id | amount | payment_status | payment_time |
|---|------------|-----------|--------|----------------|--------------|
| ▶ | 1          | NULL      | 30     | Success        | 10:30:00     |
|   | 2          | NULL      | 44     | Success        | 11:00:00     |
|   | 3          | NULL      | 300    | Failed         | 10:30:00     |
|   | 4          | NULL      | 50     | Success        | 13:45:00     |
|   | 5          | NULL      | 450    | Pending        | 11:00:00     |
|   | 6          | NULL      | 550    | Success        | 13:45:00     |
| * | NULL       | NULL      | NULL   | NULL           | NULL         |

**CREATE VIEW View\_SuccessPayments AS**

**SELECT \***



# Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

**FROM Payment**

**WHERE payment\_status = 'Success';**

**SELECT \* FROM View\_SuccessPayments;**

Output:

|   | payment_id | ticket_id | amount | payment_status | payment_time |
|---|------------|-----------|--------|----------------|--------------|
| ▶ | 1          | NULL      | 30     | Success        | 10:30:00     |
|   | 2          | NULL      | 44     | Success        | 11:00:00     |
|   | 4          | NULL      | 50     | Success        | 13:45:00     |
|   | 6          | NULL      | 550    | Success        | 13:45:00     |

**For TRIGGER:**

**Code:**

|   | payment_id | ticket_id | amount | payment_status | payment_time |
|---|------------|-----------|--------|----------------|--------------|
| ▶ | 1          | NULL      | 300.00 | Success        | 10:30:00     |
|   | 2          | NULL      | 400.00 | Success        | 11:00:00     |
|   | 3          | NULL      | 250.00 | Pending        | 12:15:00     |
|   | 4          | NULL      | 500.00 | Success        | 13:45:00     |
|   | 5          | NULL      | 350.00 | Failed         | 15:00:00     |
| + | NULL       | NULL      | NULL   | NULL           | NULL         |

Payment 9 x

**CREATE TRIGGER apply\_discount**

**BEFORE UPDATE ON Payment**

**FOR EACH ROW**

**BEGIN**

**SET NEW.amount = NEW.amount \* 0.10;**

**END apply\_discount;**

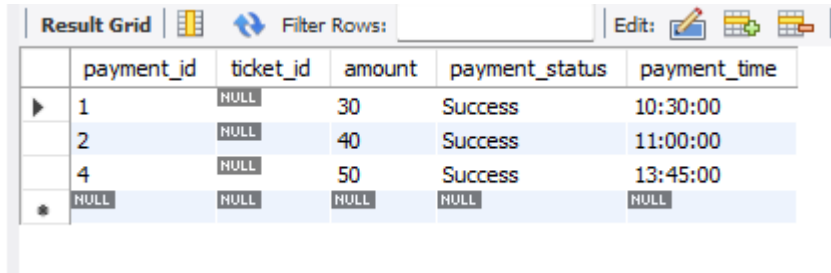


# Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

**SELECT \* FROM Payment;**

**Output:**



|   | payment_id | ticket_id | amount | payment_status | payment_time |
|---|------------|-----------|--------|----------------|--------------|
| ▶ | 1          | NULL      | 30     | Success        | 10:30:00     |
|   | 2          | NULL      | 40     | Success        | 11:00:00     |
|   | 4          | NULL      | 50     | Success        | 13:45:00     |
| * | NULL       | NULL      | NULL   | NULL           | NULL         |

**Conclusion:**

The implementation of views and triggers plays a vital role in enhancing the functionality, security, and integrity of a database system. Views provide a powerful mechanism to simplify complex queries, restrict user access to sensitive data, and present customized representations of the underlying data. On the other hand, triggers automate actions in response to specific database events, ensuring consistency, enforcing business rules, and reducing the need for repetitive application logic. Together, views and triggers significantly contribute to building robust, efficient, and maintainable database applications.

A) Brief about the benefits for using views and triggers.

Ans. **Benefits of Views:**

## 1. Data Abstraction

- Views allow users to see specific data from one or more tables without exposing the full structure of the database.
- This simplifies the user's interaction with complex data.

## 2. Enhanced Security

- By creating views, access can be restricted to specific rows or columns, thus protecting sensitive data.
- Users can only access the view without knowing the underlying table details.

## 3. Simplified Querying



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

- **Complex joins and filters can be written once in a view and reused multiple times, making queries easier and cleaner.**
- **Views act like shortcuts for repeated queries.**

### **4. Logical Data Independence**

- **Changes in base table structure (like adding columns) don't affect applications if they use views.**
- **This provides flexibility and reduces dependencies.**

### **5. Data Consistency**

- **Views provide a consistent and uniform representation of data even if it is retrieved from multiple tables.**

## **Benefits of Triggers:**

### **1. Automatic Execution**

- **Triggers are automatically fired when certain events (INSERT, UPDATE, DELETE) occur on a table.**
- **This reduces the need for manual intervention.**

### **2. Data Integrity**

- **Triggers help maintain consistency and correctness in data by validating or checking conditions before changes are made.**

### **3. Audit and Logging**

- **Useful for keeping track of user activity or changes made to tables by automatically logging such events.**

### **4. Enforcement of Business Rules**

- **Triggers can enforce business logic at the database level, such as preventing transactions outside working hours.**

### **5. Preventing Invalid Transactions**

- **Triggers can cancel harmful or invalid data operations, ensuring better control over the database.**



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

B) Explain different strategies to update views

Ans. In SQL, views are virtual tables representing the result of a predefined query.

Although they do not store data physically, they can often be used and manipulated like base tables. However, updating views is subject to certain rules and limitations. Below are the key strategies used to update views effectively:

### 1. Direct Update on Simple Views

If a view is derived from a single base table and does not contain aggregations, GROUP BY, DISTINCT, joins, or subqueries, it can be updated directly.

*Example:*

```
CREATE VIEW user_view AS  
SELECT user_id, uname FROM users;
```

```
UPDATE user_view  
SET uname = 'Alice'  
WHERE user_id = 1;
```

This will successfully update the uname column in the base users table.

### 2. Using INSTEAD OF Triggers

For complex views (especially those involving joins or aggregated data), direct updates are not allowed. In such cases, INSTEAD OF triggers (supported in some DBMS like SQL Server and Oracle) are used to define custom logic for update operations.

*Usage:*

When a user attempts to update a view, the trigger intercepts the operation and applies the necessary changes to the underlying tables.

### 3. View Update Using WITH CHECK OPTION

The WITH CHECK OPTION ensures that updates or inserts made through the view adhere to the view's WHERE clause conditions. It prevents modifications that would result in rows no longer being part of the view.

*Example:*

```
CREATE VIEW high_salary AS
```



# Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

---

```
SELECT * FROM employees WHERE salary > 50000  
WITH CHECK OPTION;
```

This ensures that no update through `high_salary` can set a salary below 50000.

## 4. Modifying the View Definition Using CREATE OR REPLACE VIEW

Instead of changing the data, a developer may choose to modify the structure or logic of the view itself using the `CREATE OR REPLACE VIEW` command.

*Example:*

```
CREATE OR REPLACE VIEW user_view AS  
SELECT user_id, uname, email FROM users;
```

This redefines the view without affecting existing data in the base table.

## 5. Update on Views with Joins (Partially Updatable Views)

In some DBMS, views based on joins may allow updates only to one of the underlying tables, provided the columns from that table are uniquely identifiable.

If ambiguity exists, such updates are restricted unless explicitly handled via triggers.