Assignment No. 8

**Name: Tejaswini Durge**

**Roll No: 331020**

**PRN: 22210270**

**Div: TY-A1**

Execute DDL statements which demonstrate the use of views. Try to update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.

Aim:-Execute DDL statements which demonstrate the use of views. Try to update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.

Objective - To study and learn view implementation.

Theory:

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view. Views, which are a type of virtual tables allow users to do the following –

● Structure data in a way that users or classes of users find natural or intuitive.

● Restrict access to the data in such a way that a user can see and (sometimes) modify

exactly what they need and no more.

● Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific

implementation.

The basic CREATE VIEW syntax is as follows −

CREATEVIEWview\_name AS

SELECT column1, column2.....

FROMtable\_name

WHERE[condition];

You can include multiple tables in your SELECT statement in a similar way as you use them in a

normal SQL SELECT query

Let’s start by :-

Creating views, updating the base table through those views, and discussing the restrictions on updatable views.

**Step 1: Create a View from a Single Table**

We can create a view from the events table to demonstrate a simple view.

**Create a View**

CREATE VIEW ActiveProjects AS

SELECT ProjectID, ProjectName, Deadline

FROM Projects

WHERE Deadline > NOW();

**Explanation**:

It selects the ProjectID, ProjectName, and Deadline from the Projects table.

The WHERE Deadline > NOW() clause ensures that only projects with a future deadline (i.e., projects that are still active) are included in the view.

CREATE VIEW UserDetails AS

SELECT UserID, UserName, Role

FROM Users;

**Explanation:**

* It selects the UserID, UserName, and Role from the Users table.
* This view helps in quickly accessing user information without querying the entire Users table every time.

CREATE VIEW TeamProjectDetails AS

SELECT

T.TeamID,

T.TeamName,

P.ProjectName,

U.UserName

FROM Teams T

JOIN Projects P ON T.ProjectID = P.ProjectID

JOIN TeamMembers TM ON T.TeamID = TM.TeamID

JOIN Users U ON TM.UserID = U.UserID;

**Explanation:**

* The view joins four tables: Teams, Projects, TeamMembers, and Users.
* It retrieves:
  + TeamID and TeamName from the Teams table.
  + ProjectName from the Projects table (based on the relationship between Teams and Projects via ProjectID).
  + UserName from the Users table (based on the relationship between TeamMembers and Users via UserID).
* This view essentially maps teams to their respective projects and users, helping to track which teams are working on which projects and who the team members are.

UPDATE Users u

JOIN (SELECT UserID FROM Users WHERE UserName = 'Tejaswini Durge') AS sub

ON u.UserID = sub.UserID

SET u.Role = 'Faculty';

**Explanation:**

* The subquery SELECT UserID FROM Users WHERE UserName = 'Tejaswini Durge' retrieves the UserID of the user with the name Tejaswini Durge.
* The JOIN clause connects this subquery result with the Users table, ensuring that only the relevant user’s record is updated.
* The SET u.Role = 'Faculty' updates the Role field for the selected user, changing it to 'Faculty'.

CREATE VIEW ProjectCounts AS

SELECT

FacultyID,

COUNT(ProjectID) AS TotalProjects

FROM Projects

GROUP BY FacultyID;

**Explanation:**

* The COUNT(ProjectID) function counts the number of projects each faculty member is associated with.
* The results are grouped by FacultyID, which means each row in the view corresponds to a faculty member, along with the number of projects they are managing.

SELECT \* FROM Users WHERE UserName = 'Tejaswini Durge';

**Explanation:**

* It retrieves all columns (\*) from the Users table where the UserName matches 'Tejaswini Durge'.
* This is a basic SELECT query used to fetch all details for a specific user from the Users table.