VIEWS

A view is a schema object formed of a stored SQL query based on tables or other views.

A view is stored in the database as a SELECT statement. No data is actually stored.

The tables that the views are based on are called **base tables**.

The real reasons to use views is,

1) Restrict data access by concealing sensitive data.
2) Simplify data retrieval by complex queries easier.
3) To present different views of the same data.
4) Provide logical data independence such as if you multiply the sales column with a certain value, we can hide that arithmetic operation.

There are two types of views, SIMPLE and COMPLEX views.

SIMPLE views are that retrieve data from one table, containing no functions and no groupings of data. Also, should not contain DISTINCT or ROWNUM. Can perform DML operations.

COMPLEX views retrieve data from more than one table, can contain functions and groupings, DISTINCT, ROWNUM, ROWID etc. DML operations are not always performed.

CREATING SIMPLE VIEWS:

```
CREATE [OR REPLACE] | [FORCE | NO FORCE] VIEW view_name
[(alias, [, alias]….)] AS subquery
WITH CHECK OPTION [CONSTRAINT constraint_name]
WITH READ ONLY [CONSTRAINT constraint_name];
```

REPLACE:

Modifies the views without the need to re-grant the privileges.

FORCE:

Create a view even if the base table does not exist.

NO FORCE:

Creates the view only if the base table exists.

WITH CHECK OPTION:

Prevents any kind of DML operation that view cannot select.

WITH READ ONLY:

Prevents any DML operation on the view.

```
CREATE VIEW empvw40 (e_id, name, surname, email) AS
SELECT employee_id, first_name, last_name, email
FROM employees WHERE department_id = 40;
SELECT * FROM empvw40;
```

CREATING COMPLEX VIEWS:

```
CREATE VIEW emp_cx_vw (DNAME, MIN_SAL, MAX_SAL) AS
SELECT distinct upper(department_name), min(salary), max(salary)
FROM employees e JOIN departments d
USING(department_id)
GROUP BY department_name;
SELECT * FROM emp_cx_vw;
```

MODIFYING A VIEW:

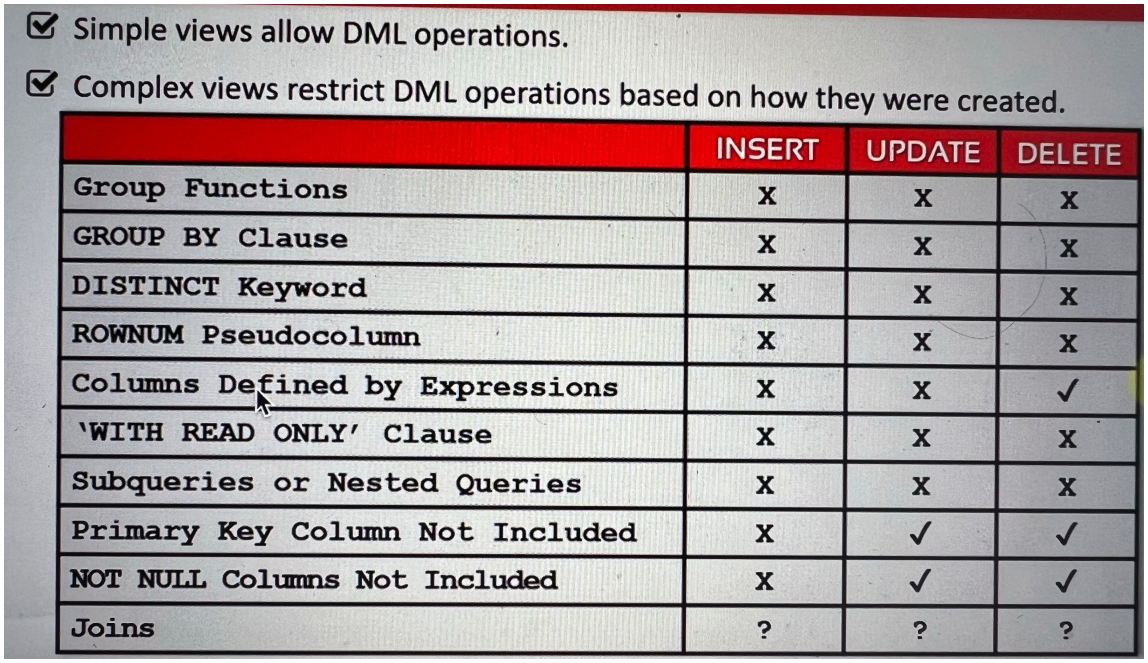
Instead of modifying a view, we can drop and re-create a view but when we drop a view all the privileges are dropped which can create problems like re-granting privileges to the users.

Easier option is to use the CREATE OR REPLACE option to modify the view.

```
CREATE OR REPLACE VIEW empvw30 AS
SELECT employee_id e_id, first_name name, last_name surname, job_id
FROM employees WHERE department_id = 30;
```

PERFORMING DML OPERATIONS ON VIEWS:

Below is a list of operations that can be performed on SIMPLE and COMPLEX views.



	INSERT	UPDATE	DELETE
Group Functions	X	X	X
GROUP BY Clause	X	X	X
DISTINCT Keyword	X	X	X
ROWNUM Pseudocolumn	X	X	X
Columns Defined by Expressions	X	X	✓
'WITH READ ONLY' Clause	X	X	X
Subqueries or Nested Queries	X	X	X
Primary Key Column Not Included	X	✓	✓
NOT NULL Columns Not Included	X	✓	✓
Joins	?	?	?

WITH CHECK OPTION:

Is used to ensure the user cannot perform any DML operations that the view cannot select.

If the DML operation is violating the WHERE clause used in the view, it will return an error.

The with check option will check the WHERE clause when inserting or deleting a record.

WITH READ ONLY:

This will completely prevent DML operations on the views.

WITH READ ONLY cannot be used with WITH CHECK OPTION clause.

DROPPING VIEWS:

Using DROP VIEW view_name statement.

Dropping a view invalidates all the objects created on that view like triggers, PL/SQL package etc.

To drop a view, we need to be owner of the view or DROP ANY VIEW privilege.