# EX.NO:5 DATE:4.09.2024

**CREATING VIEWS**

After the completion of this exercise, students will be able to do the following:

* Describe a view
* Create, alter the definition of, and drop a view
* Retrieve data through a view
* Insert, update, and delete data through a view
* Create and use an inline view

# View:

A view is a logical table based on a table or another view. A view contains no data but is like a window through which data from tables can be viewed or changed.The tables on which a view is based are called base tables.

# Advantages of Views:

* To restrict data access
* To make complex queries easy
* To provide data independence
* To present different views of the same data

# Classification of views:

1. Simple view
2. Complex view

CREATE OR REPLACE FORCE/NOFORCE VIEW view\_name AS Subquery WITH CHECK

OPTION CONSTRAINT constraint WITH READ ONLY CONSTRAINT constraint; FORCE - Creates the view regardless of whether or not the base tables exist.

NOFORCE - Creates the view only if the ase table exist.

WITH CHECK OPTION CONSTRAINT-specifies that only rows accessible to the view can be inserted or updated.

WITH READ ONLY CONSTRAINT-ensures that no DML operations can be performed on the View.

**Example: 1** (Without using Column aliases)

Create a view EMPVU80 that contains details of employees in department80.

# Example 2:

CREATE VIEW empvu80 AS SELECT employee\_id, last\_name, salary FROM employees WHERE department\_id=80;

**Example:1** (Using column aliases) CREATE VIEW salvu50

AS SELECT employee\_id,id\_number, last\_name NAME, salary \*12 ANN\_SALARY FROM employees

WHERE department\_id=50; Retrieving data from a view Example:

SELECT \* from salvu50; Modifying a view

A view can be altered without dropping, re-creating. Example: (Simple view)

Modify the EMPVU80 view by using CREATE OR REPLACE.

CREATE OR REPLACE VIEW empvu80 (id\_number, name, sal, department\_id) AS SELECT employee\_id,first\_name, last\_name, salary, department\_id

FROM employees

WHERE department\_id=80;

**Example:** (complex view)

CREATE VIEW dept\_sum\_vu (name, minsal, maxsal,avgsal)

AS SELECT d.department\_name, MIN(e.salary), MAX(e.salary), AVG(e.salary) FROM employees e, department d

WHERE e.deparment\_id=d.deparment\_id GROUP BY d.department\_name;

# Rules for performing DML operations on view

* Can perform operations on simple views
* Cannot remove a row if the view contains the following:
* Group functions
* Group By clause
* Distinct keyword
* Cannot modify data in a view if it contains
* Group functions
* Group By clause
* Distinct keyword
* Columns contain by expressions
* Cannot add data thr‘ a view if it contains
* Group functions
* Group By clause
* Distinct keyword
* Columns contain by expressions
* NOT NULL columns in the base table that are not selected by the view

**Example:** (Using the WITH CHECK OPTION clause) CREATE OR REPLACE VIEW empvu20

AS SELECT \*

FROM employees WHERE department\_id=20

WITH CHECK OPTION CONSTRAINT empvu20\_ck;

Note:Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint.

**Example** – (Execute this and note the error)

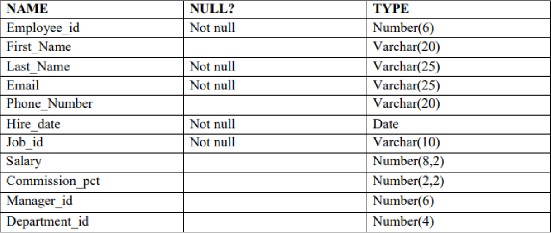
UPDATE empvu20 SET department\_id=10 WHERE employee\_id=201;

# Denying DML operations

**Use of the WITH READ ONLY option.**

Any attempt to perform a DML on any row in the view results in an oracle server error. Try this code:

CREATE OR REPLACE VIEW empvu10(employee\_number, employee\_name,job\_title) AS SELECT employee\_id, last\_name, job\_id

FROM employees WHERE department\_id=10 WITH READ ONLY;

CREATE TABLE Employee (

Employee\_id NUMBER(6) NOT NULL, First\_Name VARCHAR2(20),

Last\_Name VARCHAR2(25) NOT NULL, Email VARCHAR2(25) NOT NULL,

Phone\_Number VARCHAR2(20), Hire\_date DATE NOT NULL, Job\_id VARCHAR2(10) NOT NULL, Salary NUMBER(8,2),

Commission\_pct NUMBER(2,2), Manager\_id NUMBER(6), Department\_id NUMBER(4),

CONSTRAINT pk\_employee PRIMARY KEY (Employee\_id)

);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id)

VALUES (101, 'Amit', 'Sharma', 'amit.sharma@example.com', '9123456789', TO\_DATE('2021- 03-01', 'YYYY-MM-DD'), 'IT\_PROG', 60000, NULL, 100, 50);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id)

VALUES (102, 'Sita', 'Patel', 'sita.patel@example.com', '9123456790', TO\_DATE('2020-05-12', 'YYYY-MM-DD'), 'HR\_REP', 45000, NULL, 101, 30);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id) VALUES (103, 'Rahul', 'Verma', 'rahul.verma@example.com', '9123456791', TO\_DATE('2019-07-23', 'YYYY- MM-DD'), 'SA\_MAN', 70000, 0.1, 101, 80);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id) VALUES (104, 'Priya', 'Singh', 'priya.singh@example.com', '9123456792', TO\_DATE('2022-08-17', 'YYYY- MM-DD'), 'FI\_ACCOUNT', 50000, NULL, 102, 50);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id) VALUES (105, 'Rohan', 'Reddy', 'rohan.reddy@example.com', '9123456793', TO\_DATE('2020-11-02', 'YYYY- MM-DD'), 'ST\_CLERK', 35000, NULL, 101, 50);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id) VALUES (106, 'Kiran', 'Nair', 'kiran.nair@example.com', '9123456794', TO\_DATE('2018-09-25', 'YYYY-MM- DD'), 'IT\_PROG', 65000, NULL, 100, 60);

INSERT INTO Employee (Employee\_id, First\_Name, Last\_Name, Email, Phone\_Number, Hire\_date, Job\_id, Salary, Commission\_pct, Manager\_id, Department\_id) VALUES (107, 'Anjali', 'Desai', 'anjali.desai@example.com', '9123456795', TO\_DATE('2021-12-05', 'YYYY- MM-DD'), 'HR\_REP', 47000, NULL, 102, 30);



# Find the Solution for the following:

1. **Create a view called EMPLOYEE\_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee**

# name to EMPLOYEE.

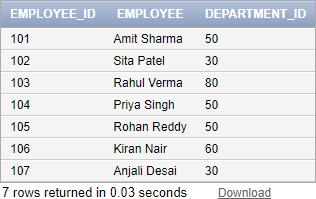
CREATE VIEW EMPLOYEE\_VU AS

SELECT Employee\_id,

First\_Name || ' ' || Last\_Name AS EMPLOYEE, Department\_id

FROM Employee;

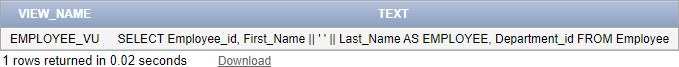
# Display the contents of the EMPLOYEES\_VU view.

SELECT \* FROM EMPLOYEE\_VU;

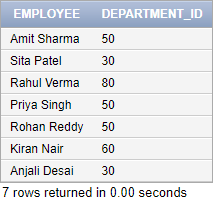
# Select the view name and text from the USER\_VIEWS data dictionary views.

SELECT view\_name, text FROM user\_views

WHERE view\_name = 'EMPLOYEE\_VU';



# Using your EMPLOYEES\_VU view, enter a query to display all employees names and department.

SELECT EMPLOYEE, Department\_id FROM EMPLOYEE\_VU;

# Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50.Label the view columns EMPNO,

**EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department**

# through the view.

CREATE VIEW DEPT50 AS

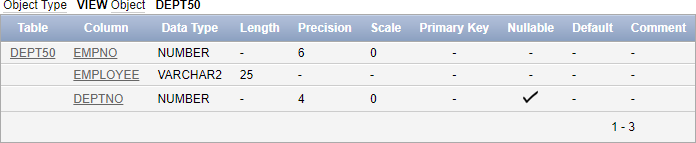
SELECT Employee\_id AS EMPNO, Last\_Name AS EMPLOYEE, Department\_id AS DEPTNO

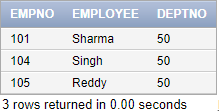
FROM Employee

WHERE Department\_id = 50 WITH READ ONLY;

# Display the structure and contents of the DEPT50 view.

DESC DEPT50;

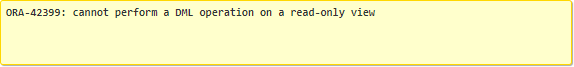


SELECT \* FROM DEPT50;

# Attempt to reassign Matos to department 80.

UPDATE DEPT50 SET DEPTNO = 80

WHERE EMPLOYEE = 'Matos';

This should fail due to the READ ONLY constraint.

# Create a view called SALARY\_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and

**JOB\_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.**

CREATE VIEW SALARY\_VU AS SELECT

e.Last\_Name AS Employee, d.Dept\_name AS Department, e.Salary AS Salary, jg.Grade\_level AS Grade

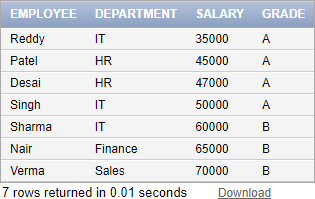
FROM

Employee e JOIN

Department d ON e.Department\_id = d.Dept\_id JOIN

Job\_Grade jg ON e.Salary BETWEEN jg.Lowest\_sal AND jg.Highest\_sal;

SELECT \* FROM SALARY\_VU;



# RESULT:

Hence, views are created and executed successfully.