Ex.No.: 8	WORKING WITH MULTIPLE TABLES
Date: 03/09/2024	

Objective

After the completion of this exercise, the students will be able to do the following:
• Write SELECT statements to access data from more than one table using equality and nonequality joins
• View data that generally does not meet a join condition by using outer joins
• Join a table to itself by using a self join
Sometimes you need to use data from more than one table.

Cartesian Products
• A Cartesian product is formed when:
– A join condition is omitted
– A join condition is invalid
– All rows in the first table are joined to all rows in the second table
• To avoid a Cartesian product, always include a valid join condition in a WHERE clause.
A Cartesian product tends to generate a large number of rows, and the result is rarely useful. You should always include a valid join condition in a WHERE clause, unless you have a specific need to combine all rows from all tables.
Cartesian products are useful for some tests when you need to generate a large number of rows to simulate a reasonable amount of data.
Example:
To displays employee last name and department name from the EMPLOYEES and DEPARTMENTS tables.

```
SELECT last_name, department_name dept_name
FROM   employees, departments;
```

Types of Joins

• Equijoin
• Non-equijoin
• Outer join
• Self join
• Cross joins
• Natural joins
• Using clause
• Full or two sided outer joins
• Arbitrary join conditions for outer joins

Joining Tables Using Oracle Syntax

```
SELECT table1.column, table2.column
```

This query was completed in earlier releases as follows:

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  d.department_id = e.department_id  (+);
```

FULL OUTER JOIN
Example:

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
FULL OUTER JOIN departments d
ON   (e.department_id = d.department_id) ;
```

This query retrieves all rows in the EMPLOYEES table, even if there is no match in the DEPARTMENTS table. It alslso retrieves all rows in the DEPARTMENTS table, even if there is no match in the EMPLOYEES table.

Find the Solution for the following:

1. Write a query to display the last name, department number, and department name for all employees.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

2. Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

```
SELECT DISTINCT j.job_title, l.city FROM employees e
JOIN jobs j ON e.job_id = j.job_id
JOIN departments d ON e.department_id = d.department_id
JOIN location l ON d.location_id = l.location_id
WHERE d.department_id = 80;
```

3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

```
SELECT e.last_name, d.department_name, d.location_id, l.
city FROM employees e
JOIN departments d ON e.department_id = d.department_
id
JOIN location l ON d.location_id = l.location_id
WHERE e.commission_pct IS NOT NULL;
```

2. Display the employee last name and department name for all employees who have an a(lowercase) in their last names. P

```
SELECT e.last_name, d.department_name FROM
employees e JOIN departments d ON e.department_id
=d.department_id WHERE LOWER(e.last_name) like '%a%';
```

5. Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

```
SELECT e.last_name, j.job_title, d.department_id,
d.department_name FROM employees e JOIN job j ON
e.job_id = j.job_id JOIN departments d ON e.
department_id = d.department_id JOIN locations l
ON d.location_id = l.location_id where l.city='Toronto';
```

6. Display the employee last name and employee number along with their manager's last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, Respectively

```
SELECT e.last_name AS Employee, e.employee_id AS Emp#,
       m.last_name AS Manager, m.employee_id AS Mgr#
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id;
```

7. Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

```
SELECT e.last_name AS Employee, e.employee_id AS
Emp#, m.last_name AS Manager, m.employee_id AS
Mgr# FROM employees e LEFT JOIN employees m ON
e.manager_id = m.employee_id ORDER BY e.employee_id
```

8. Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label

```
SELECT e1.last_name AS Employee, e1.department_id,
       e2.last_name AS colleague
FROM employees e1 JOIN employees e2 ON e1.department
_id = e2.department_id where e1.employee_id !=
e2.employee_id ORDER BY e1.department_id;
```

9. Show the structure of the JOB_GRADES table. Create a query that displays the name, job, department name, salary, and grade for all employees

```
DESC job_grades;
SELECT e.last_name, j.job_title, d.department_name,
e.salary, g.grade_level FROM employees e
JOIN jobs j ON e.job_id = j.job_id
JOIN departments d ON e.department_id = d.department
_id
JOIN job_grades g ON e.salary BETWEEN g.lowest
```

10. Create a query to display the name and hire date of any employee hired after employee Davies.

```
SELECT e.last_name, e.hire_date
FROM employees e
WHERE e.hire_date > (SELECT hire_date FROM
employees WHERE last_name = 'Davie');
```

11. Display the names and hire dates for all employees who were hired before their managers, along with their manager's names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

```
SELECT e.last_name AS Employee, e.hiredate AS
"Emp Hired", m.last_name AS manager, m.hire_
date AS "Mgr Hired" FROM employees e JOIN
employees m ON e.manager_id = m.employee_id
WHERE e.hire_date < m.hire_date ORDER BY
e.hire_date;
```

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	