EXERCISE-7

Displaying data from multiple tables

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
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

Write the join condition in the WHERE clause.
• Prefix the column name with the table name when the same column name appears in more than one table.

Guidelines

- When writing a SELECT statement that joins tables, precede the column name with the table name for clarity and to enhance database access.
- If the same column name appears in more than one table, the column name must be prefixed with the table name.
- To join n tables together, you need a minimum of n-1 join conditions. For example, to join four tables, a minimum of three joins is required. This rule may not apply if your table has a concatenated primary key, in which case more than one column is required to uniquely identify each row

What is an Equijoin?

To determine an employee's department name, you compare the value in the DEPARTMENT_ID column in the EMPLOYEES table with the DEPARTMENT_ID values in the DEPARTMENTS table.

The relationship between the EMPLOYEES and DEPARTMENTS tables is an equijoin—that is. values in the DEPARTMENT_ID column on both tables must be equal. Frequently, this type of join involves primary and foreign key complements.

Note: Equijoins are also called simple joins or inner joins

```
SELECTemployees.employee_id,employees.last_name,employees.department_id,
departments.department_id,departments.location_id
FROM   employees, departments
WHERE  employees.department_id = departments.department_id;
```

Additional Search Conditions

Using the AND Operator

Example:

To display employee Matos'department number and department name, you need an additional condition in the WHERE clause.

```
SELECT last_name, employees.department_id,
department_name
FROM   employees, departments
WHERE  employees.department_id = departments.department_id AND   last_name = 'Matos';
```

Qualifying Ambiguous Column Names

- Use table prefixes to qualify column names that are in multiple tables.
- Improve performance by using table prefixes.
- Distinguish columns that have identical names but reside in different tables by using column aliases.

Using Table Aliases

- Simplify queries by using table aliases.
- Improve performance by using table prefixes

Example:

```
SELECT e.employee_id, e.last_name, e.department_id,
d.department_id, d.location_id
FROM   employees e , departments d
WHERE  e.department_id = d.department_id;
```

Joining More than Two Tables

To join n tables together, you need a minimum of n-1 join conditions. For example, to join three

• A join between two tables that returns the results of the inner join as well as unmatched rows left (or
right) tables is a left (or right) outer join.

• A join between two tables that returns the results of an inner join as well as the results of a left and right join is a full outer join.

LEFT OUTER JOIN
Example:

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

Example of LEFT OUTER JOIN
This query retrieves all rows in the EMPLOYEES table, which is the left table even if there is no match in the DEPARTMENTS table.

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;
```

RIGHT OUTER JOIN
Example:

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

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

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 employee e .join department 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 e. job_id , a . location_id from employee
e. join department d on e. department_id = d.department_id
where e. 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 , l. location_id , city
from employee e join departments on e. departments id .
. d. department_id join locations on d. location_id =
l. location_id where e - commission- pct is not Null;
```

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

```
select e. last d. department_name from employee e join department
d on e. department_id = d. department_id where 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 , e. job_id , d . department_id , d. department
name from employee join departments on e. department = d.
department_id join location 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 Empl
m last_name as manager , m. employee_id as mgr # from
employee e left join employee 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.employee_id, e.last_name, e.manager_id.
from employees 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 Name", e1.department_id as "Dept ID",
e2.last_name as "Colleague" from employees e1 join employees
e2 on e1.department_id = e2.department_id order by e1.
department_id, e1.last_name;
```

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

```
describe job_grades.
```

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 = 'Davies');
```

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.hire_date as"
Emp hired", m.last_name as manager, m_hire_date as
"mgr hired" from employee e join employees m
on emanager_id = m.employee_id where e.hire_
date2 . m_hire_date;
```

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