

## 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.

This query retrieves all rows in the EMPLOYEES table, even if there is no match in the DEPARTMENTS table. It also 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 e.job-id, d.location-id, from employees e  
JOIN departments 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, d.location-id,  
l.city, from employees e JOIN departments d on e.  
department-id = d.department-id JOIN locations l 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-name, d.department-name from employees e JOIN  
departments 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, e.department-id, d.department-name  
from employees e 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 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 EmpID,  
NVL(m.last-name, 'No Manager') as manager, NVL(jd.department\_id, -1) as Mgr # from employees e LEFT  
JOIN employees m ON e.manager\_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 e2.last-name as colleague, e2.department\_id from  
employees e1 JOIN employees e2 ON e1.department\_id =  
e2.department\_id WHERE e1.employee\_id = 144 AND e2.employee\_id <> 144;

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, e.job\_id, d.department\_name, e.salary, j.grade\_level from employees JOIN

departments d ON e.department\_id = d.department\_id JOIN  
job\_grades j ON e.salary BETWEEN j.lower\_bound AND j.highest\_bound

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

Select last-name, hire-date from employees WHERE  
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.hiredate as 'Emp-Hire',  
m.last-name as manager, m.hire-date as 'Mgr-Hire'  
from employees e JOIN employees m ON e.manager\_id =  
m.employee\_id WHERE e.hire-date < m.hire-date;

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	RJ 8/9/2K