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.

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 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 last_name, e. department_id). d. department_name.
From employees
Join department 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, l. 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. 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, l.city from employees. Join department d. department_id = d department join location on d. location_id = l. location_id where 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 department d on. e. department_id = d. department_id.

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
Join location 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
left Join employees 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 last name As employee , e. employee id as "Emp#" ,
m. last name as manager , m. employee_id as "Mgr#" ;
From employee left Join employee 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" , e. department _id as
"order" , c. last name As "Colleague" from employee e c ,
order By e. deptment _id. c. 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

Desc Job- grade

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 last name as "Employee" , e. hire - date as
"Emp Hired" from employees
Join employee on e. manager _id m. employee . id
where e. hire _ date (m. hire date");

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