Lab3

Objectives:

At the end of this lab, you should be able to

- Write select statement to retrieve data from more one table.
- Use subquires

Cartesian Product

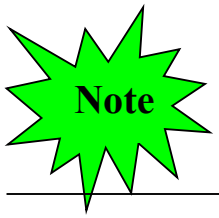
- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table

Example:

```
SELECT  ename, dname  FROM   emp, dept;
```

You can explicitly create Cartesian product using "cross join:
For example

```
Select ename, dname from emp cross join dept;
```



To avoid a Cartesian product, always include a valid join condition.

Question:
Create a query to display employee name, his department name for all employees.

To get such a report, the data is in separate tables.

- Ename exists in the EMP table.
- Dname exists in DEPT table.

To produce the report, you need to link EMP and DEPT tables and access data from both of them, write a simple join condition in the WHERE clause.

```
select  emp.ename,dept.dname from emp,dept where emp.deptno=dept.deptno;
```

Notes:

- 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.

Prepared by :Dr. Saleh Alhazbi

Types of Joins

The types of join conditions:

1. **Equijoins**
2. **Non-equijoins**
3. **Outer joins**
4. **Self joins**

1) Equijoins:

When the query is relating two tables using an equality operator(=) it is an equijoin or (inner join).

Example: select emp.ename,dept.dname from emp,dept where emp.deptno=dept.deptno;

Note: You can join tables automatically based on the columns in the two tables that have matching data types and names using NATURAL JOIN keywords.

Selet ename, dname from emp natural join dept;

Note:

- If the columns have the same name but different types then it causes syntax error.
- If the two tables have multiple similar columns, and you want to join based only one table, then you should use clause ON

```
Select ename, dept from emp
join  dept on (emp.deptno=dept.deptno);
```

Applying additional conditions with Equijoin

Select ename, dname from emp natural join dept where mgr=7839;

Which is equivalent to

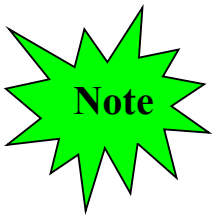
```
Select  emp.ename,dept.dname from emp,dept where emp.deptno=dept.deptno
and
mgr=7839;
```

---

2) Self joins:

How to retrieve the name of each employee with name of his manager? Managers also are employees and their data is in the table emp.

```
SELECT worker.ename,manager.ename
FROM        emp worker, emp manager
WHERE       worker.mgr = manager.empno;
```



With self-join, you need to use alias to refer to the same table twice.

3) Non-Equijoins:

Example :

```
SELECT     e.ename, e.sal, s.grade FROM emp e,  salgrade s
WHERE      e.sal
BETWEEN  s.losal AND s.hisal;
```

Notes : In the above example, we use table aliases, in this case they are not optional.

---

4) Outer joins:

Note: Joining two tables using equijoin will not display unmatched rows in both tables, this is called "Inner join". To return the unmatched rows, we use Outer join.

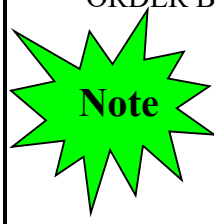
Example :

If we want to display all departments including those that do not have employees.

```
Select ename, dname from emp
right outer join dept on (emp.deptno=dept.deptno);
```

Note: in Oracle, you can use + operator for outer join

```
SELECT       e.ename, d.deptno, d.dname  FROM  emp e,   dept d
  WHERE      e.deptno(+) = d.deptno
   ORDER BY          e.deptno;
```



The outer join operator (+)can appear on only one side of the expression—the side that has information missing. It returns those rows from one table that have no direct match in the other table.
The (+) after deptno in emp table (left side) indicates that employees will be displyay as null for the departments that do not have employees.

Practice:

Write a query to display employee name and manger name including the employees who have no managers.

Note: To include data from both sides, even if they have no match on the other side, use "Full outer"

```
Select e.name, m.ename from emp e
Full outer join emp m on (e.mgr=m.empno);
```

Practice

1. Write a query to display the employee name, department name, and location of all employees who earn a commission.
2. Show the structure of the SALGRADE table. Create a query that will display the name, job, department name, salary, and grade for employees who are in grade 3.
3. Create a query to display the name and hire date of any employee hired after employee Blake.
4. Display all employees' names and hire dates along with their manager's name and hire date for all employees who were hired before their managers. Label the columns Employee, Emp, Hiredate, Manager, and Mgr Hiredate, respectively.

Subqueries

```
SELECT    select_list
FROM      table
WHERE     expr operator
                    (SELECT      select_list
                    FROM         table);
```

Notes:

- The subquery (inner query) executes before the outer query.
- The result of the subquery is used by the main query.
- The subquery can be used in SQL clauses including: where clause, having clause, from clause.
- *Operator* includes comparison conditions (>,>=,<,<=,=,<>) for single row subquery and (IN, ANY, ALL) for multiple-row subquery.

Single Row subquery:

The subquery returns only one row.

Example: Assume we want to display names of all employees whose salaries are greater than the salary of employee no. 7566

```
SELECT ename  FROM   emp
WHERE  sal >   (SELECT sal      FROM   emp
                WHERE  empno=7566);
```

In the example, the inner query determines the salary of employee 7566. The outer query takes the result of the inner query and uses this result to display all the employees who earn more than this amount.

Practice:

Write a query to display name of the employee who was hired in the same day as "JAMES', but his salary is greater than "JAMES" salary.

Using Group Functions in a Subquery

To display name, job and salary of employees whose salary is the minimum , we use min group function in the subquery as follows :

```
SELECT      ename, job, sal  FROM emp
  WHERE     sal =
      (SELECT     MIN(sal)
              FROM        emp);
```

Using HAVING clause in the subquery

```
SELECT DEPTNO,MIN(SAL) FROM EMP
GROUP BY DEPTNO
HAVING MIN(SAL)>(SELECT MIN(SAL) FROM EMP WHERE DEPTNO=20);
```

Multiple-Row Subqueries

Subqueries that return more than one row are called *multiple-row subqueries*. You use a multiple-row operator, instead of a single-row operator, with a multiple-row subquery. The multiple-row operator expects one or more values.

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

Examples

Display employees numbers ,names and jobs for all employees whose salaries are less than any clerk

```
SELECT  empno, ename, job
 FROM    emp
 WHERE   sal < ANY
              (SELECT      sal
                      FROM emp
              WHERE       job = 'CLERK')
```

Display employees numbers ,names and jobs for all employees whose salaries are greater than all departments' averages .

```
SELECT  empno, ename, job
  FROM    emp
  WHERE   sal > ALL
               (SELECT      avg(sal)
                       FROM        emp
                       GROUP BY    deptno);
```

Practice:

1. Write a query to display the employee name and hire date for all employees in the same department as Blake. Exclude Blake.
2. Create a query to display the employee number and name for all employees who earn more than the average salary. Sort the results in descending order of salary.
3. Write a query that will display the employee number and name for all employees who work in a department with any employee whose name contains a *T*.
4. Display the employee name, department number, and job title for all employees whose department location is Dallas.
5. Display the department number, name, and job for all employees in the Sales department.