TUTORIAL EXERCISES ON SQL (DML)
MULTIPLE TABLE QUERIES

PART A
Consider the ***oracle employees*** database. Write SQL expressions that will retrieve and display:

a. Details of employees who work for the Sales department.

```
SELECT *
FROM emp
WHERE deptno =
            (SELECT deptno
             FROM dept
             WHERE dname = 'SALES');
```

Alternative expressions can be the following

```
SELECT *
FROM emp, dept
WHERE emp.deptno = dept.deptno
  AND dname = 'SALES';

SELECT *
FROM emp NATURAL JOIN dept
WHERE dname = 'SALES';

SELECT *
FROM emp JOIN dept
       USING (deptno)
WHERE dname = 'SALES';
```

b. Details of employees who either have the same job description as that of JONES or whose salary is greater than or equal to that of FORD.

If there is only one employee called FORD and only one employee called JONES in the emp table, then the following expression will return the correct results.
NB: Why is the uniqueness requirement necessary for the following SQL statement?

```
SELECT *
FROM emp
WHERE job =
           (SELECT job
            FROM emp
            WHERE ename = 'JONES')
   OR sal >=
          (SELECT sal
           FROM emp
           WHERE ename = 'FORD');
```

A better solution (that does not require unique employee names) is the following.
You may want to consider how you can use "ALL" instead of "ANY" in relation to the job predicate.

```
SELECT *
FROM emp
WHERE job = ANY
           (SELECT job
            FROM emp
            WHERE ename = 'JONES')
     OR sal >= ANY
          (SELECT sal
           FROM emp
           WHERE ename = 'FORD');
```

An alternative SQL statement can be the following

```
SELECT e.*
FROM emp e CROSS JOIN emp o
WHERE (e. job = o.job
           AND o.ename = 'JONES')
      OR (e.sal >= o.sal
```

```
           AND o.ename = 'FORD');
```

c. Details of employees whose job description is among the job descriptions of employees who work for a department located in CHICAGO.

```
SELECT *
FROM emp
WHERE job =
           (SELECT job
            FROM emp
            WHERE deptno IN
                    (SELECT deptno
                     FROM dept
                     WHERE loc = 'CHICAGO');
```

d. Details of employees who work for a department located in N.Y. or CHICAGO.

```
SELECT *
FROM emp
WHERE deptno IN
         (SELECT deptno
          FROM dept
          WHERE loc = ANY
                 ('NEW YORK', 'CHICAGO'));

SELECT emp.*
FROM emp, dept -- an alternative is to use "emp CROSS JOIN dept"
WHERE emp.deptno = dept.deptno
   AND (loc = 'NEW YORK'
         OR loc = 'CHICAGO');

The following SQL expression are also equivalent to the above
SELECT emp.*
FROM emp CROSS JOIN dept
WHERE emp.deptno = dept.deptno
   AND loc = ANY          -- instead of "=ANY" you can use "IN"
        ('NEW YORK', 'CHICAGO');
```

The following expression that involves the use of the EXISTS predicate is a better answer as it overcomes the problem of subqueries that return no rows (You may want to read about queries returning no results)

```
SELECT *
FROM emp
WHERE EXISTS
         (SELECT *
          FROM dept
          WHERE emp.deptno = deptno
            AND loc = ANY
                    ('NEW YORK', 'CHICAGO'));
```

The above SQL statement also demonstrates the scope of SQL identifiers; i.e. the column names of the emp table mentioned in the outer query are accessible from any subqueries specified in the WHERE clause. Thus, in the WHERE clause of the subquery query to disambiguate to which deptno column we refer to we need to qualify it with the name of the table it comes from, e.g. "WHERE emp.deptno = dept.deptno"; however, as the dept.deptno refers to a column of the dept table, a table that is specified in the subquery, there is no need to qualify the column as the DBMS will assume by default that we are referring to the dept.deptno, hence "WHERE emp.deptno = deptno".

Equivalent to the above answers that involve a "JOIN" expression can be the following (a "*" is used in the SELECT clause a single copy of all common columns and all other columns of the two tables will be returned):

```
SELECT *
FROM emp NATURAL JOIN dept -- or INNER NATURAL JOIN
WHERE loc IN ('NEW YORK', 'CHICAGO');

SELECT *
FROM emp INNER JOIN dept -- the keyword INNER can be omitted
       USING (deptno)
```

```
WHERE loc IN ('NEW YORK', 'CHICAGO');
```
NB: the JOIN with a USING is a form of NATURAL JOIN where only the listed common columns participate in the “natural” join.

```
SELECT *
FROM emp INNER JOIN dept -- the keyword INNER can be omitted
       ON (emp.deptno = dept.deptno)
WHERE loc IN ('NEW YORK', 'CHICAGO');
```

e. Details of employees who are in salary grade 3 and work for a department located in CHICAGO.

```
SELECT *
FROM emp, dept, salgrade
WHERE emp.deptno = dept.deptno
  AND sal between losal and hisal
  AND loc = 'CHICAGO'
  AND grade = 3;
```

An equivalent (but computationally wise more efficient) SQL statement (as selections are done first reducing the size of the tables involved) can be the following

```
SELECT *
FROM emp,
     (SELECT *
      FROM dept
      WHERE loc = 'CHICAGO') d,
     (SELECT *
      FROM salgrade
      WHERE grade = 3) salg,
WHERE emp.deptno = dept.deptno
  AND sal between losal and hisal
```

An equivalent SQL statement using JOINs can be the following

```
SELECT *
FROM (emp NATURAL JOIN dept)
        JOIN salgrade
          ON sal BETWEEN losal AND hisal
WHERE loc = 'CHICAGO'
  AND grade = 3;
```

or the following

```
SELECT *
FROM emp
       NATURAL JOIN
           (SELECT *
            FROM dept
            WHERE loc = 'CHICAGO')
       JOIN
           (SELECT *
            FROM salgrade
            WHERE grade = 3)
         ON sal BETWEEN losal AND hisal
```

f. Details of employees who are in salary grade 3 or 4, work for a department located in N.Y. or CHICAGO, their job description is SALESMAN and whose name contains the letter “N”.

Some equivalent SQL expression that answer the above query can be

```
SELECT *
FROM emp, dept, salgrade
WHERE emp.deptno = dept.deptno
   AND sal between losal and hisal
   AND (loc = 'NEW YORK'
          OR loc = 'CHICAGO') -- or use the IN, =ANY, =SOME
   AND grade IN (3, 4)
   AND job = 'SALESMAN'
   AND ename LIKE '%N%';

SELECT *
FROM emp CROSS JOIN dept CROSS JOIN salgrade
WHERE emp.deptno = dept.deptno
   AND sal between losal and hisal
```

```
   AND (loc = 'NEW YORK'
          OR loc = 'CHICAGO') -- or use the IN, =ANY, =SOME
   AND grade IN (3, 4)
   AND job = 'SALESMAN'
   AND ename LIKE '%N%';

SELECT *
FROM (emp NATURAL JOIN dept)
        JOIN salgrade
          ON sal BETWEEN losal AND hisal
WHERE loc =ANY ('NEW YORK', 'CHICAGO')
  AND grade IN (3, 4)
  AND job = 'SALESMAN'
  AND ename LIKE '%N%';
```

or the following

```
SELECT *
FROM (SELECT *
      FROM emp
      WHERE job = 'SALESMAN'
        AND ename LIKE '%N%')
       NATURAL JOIN
           (SELECT *
            FROM dept
            WHERE loc =ANY ('NEW YORK', 'CHICAGO'))
       JOIN
           (SELECT *
            FROM salgrade
            WHERE grade IN (3, 4))
         ON sal BETWEEN losal AND hisal
```

g. Details of employees who work at the location the SALES department is located and whose job description is not CLERK

```
SELECT emp.*
FROM emp, dept
WHERE emp.deptno = dept.deptno
   AND loc IN
         (SELECT loc
          FROM dept
          WHERE dname = 'SALES')
   AND job != 'CLERK';

SELECT emp.*
FROM emp
WHERE deptno =ANY
         (SELECT deptno
          FROM dept
          WHERE loc = -- the IN predicate is not required – why?
                  (SELECT loc
                   FROM dept
                   WHERE dname = 'SALES'))
  AND job <> 'CLERK'; -- or job != 'CLERK' or NOT job = 'CLERK'
```

h. The EMPNO of any two employees who work at the same location, i.e. pairs of EMPNO.

```
SELECT  e1.empno,  e1.ename,  e1.deptno,  d1.loc,  e2.empno,  e2.ename,
e2.deptno, d2.loc
FROM emp e1, dept d1, emp e2, dept d2
WHERE e1.deptno = d1.deptno
   AND e2.deptno = d2.deptno
   AND d1.loc = d2.loc
   and e1.empno != e2.empno;
```

An improved version, where only one of the pairs (e1, e2) and (e2, e1) will be displayed, is the following:

```
SELECT  e1.empno,  e1.ename,  e1.deptno,  d1.loc,  e2.empno,  e2.ename,
e2.deptno, d2.loc
FROM emp e1, dept d1, emp e2, dept d2
WHERE e1.deptno = d1.deptno
```

```
   AND e2.deptno = d2.deptno
   AND d1.loc = d2.loc
   AND e1.empno < e2.empno;
```

The following SQL statements use JOINs to answer the above query

```
SELECT c1.empno, c1.ename, c1.deptno, c1.loc,
         c2.empno, c2.ename, c2.deptno, c2.loc
FROM (SELECT *
        FROM emp NATURAL JOIN dept) c1,
     (SELECT *
        FROM emp NATURAL JOIN dept) c2
WHERE c1.loc = c2.loc
  AND c1.empno < c2.empno;

SELECT c1.empno, c1.ename, c1.deptno, c1.loc,
         c2.empno, c2.ename, c2.deptno, c2.loc
FROM (SELECT *
        FROM emp NATURAL JOIN dept) c1,
     (SELECT *
        FROM emp NATURAL JOIN dept) c2
WHERE c1.loc = c2.loc
  AND c1.empno < c2.empno;
NB: a JOIN with ON c1.loc = c2.loc cannot be used in this case.

SELECT  e1.empno,  e1.ename,  e1.deptno,  d1.loc,  e2.empno,  e2.ename,
e2.deptno, d2.loc
FROM emp e1 NATURAL JOIN dept d1
       JOIN
         emp e2 NATURAL JOIN dept d2
         USING (loc)
WHERE e1.empno < e2.empno;
NB: Notice the aliases defined and the way columns were qualified.
```

i. Details of CLERKs who work at any location that a SALESMAN works.

```
SELECT *
FROM emp NATURAL JOIN dept
WHERE job = 'CLERK'
  AND loc =ANY
         (SELECT loc
          FROM emp NATURAL JOIN dept
          WHERE job = 'SALESMAN');
```

j. Details of employee whose job description is MANAGERs and who manage at least one employee whose job description is also MANAGER.

```
SELECT *
FROM emp
WHERE job = 'MANAGER'
  AND empno IN
            (SELECT mgr
             FROM emp
             WHERE job = 'MANAGER');
```

-- an equivalent SQL expression that uses the "EXISTS" predicate is the following

```
SELECT *
FROM emp e
WHERE job = 'MANAGER'
  AND EXISTS
         (SELECT *
          FROM emp
          WHERE job = 'MANAGER'
            AND mgr = e.empno);
```

-- another possible SQL statement (depending on how you interpreted the query) can be the following (please note results are NOT the same)

```
SELECT *
FROM emp
WHERE empno IN
           (SELECT mgr
```

```
            FROM emp
            WHERE empno IN
                        (SELECT mgr
                         FROM emp));
```

k. Details of the employees whose grade is unique at the location they work.

```
SELECT e.*
FROM emp e, dept d, salgrade s
WHERE e.deptno = d.deptno
    AND sal BETWEEN losal AND hisal
    AND NOT EXISTS
           (SELECT *
            FROM emp NATURAL JOIN dept
                   JOIN salgrade
                     ON sal BETWEEN losal AND hisal
            WHERE loc = d.loc
              AND grade = s.grade
              AND empno <> e.empno);
```

NB: you can use a JOIN also in the outermost FROM clause but because you need to refer to values of its columns within the subquery an alias is required, so the following statement should be used:

```
SELECT r.*
FROM (SELECT *
      FROM emp NATURAL JOIN dept
             JOIN salgrade
               ON sal BETWEEN losal AND hisal) r
WHERE NOT EXISTS
           (SELECT *
            FROM emp NATURAL JOIN dept
                   JOIN salgrade
                     ON sal BETWEEN losal AND hisal
            WHERE loc = r.loc
              AND grade = r.grade
              AND empno <> r.empno);
```

In this case, an IN predicate you can also be used instead of the EXISTS; the above statement would need to be as follows:

```
SELECT r.*
FROM (SELECT *
      FROM emp NATURAL JOIN dept
             JOIN salgrade
               ON sal BETWEEN losal AND hisal) r
WHERE grade NOT IN
           (SELECT grade
            FROM emp NATURAL JOIN dept
                   JOIN salgrade
                     ON sal BETWEEN losal AND hisal
            WHERE loc = r.loc
              AND empno <> r.empno);
```

NB: the replacement of an EXISTS predicate by an IN predicate in a query does not necessarily result in an equivalent query; i.e. the results produced may differ depending on whether the subquery returns no rows (if the subquery returns no rows then the IN predicate will return a single row with NULL values. – What are the implications of a null row being returned?

l. Details of employees that manage other employees (i.e. their employee number should appear in the *mgr* column of some other employee) and whose *empno* value is greater than the *empno* value of at least one of the employees they manage.

```
SELECT *
FROM emp m, emp e
WHERE e.mgr = m.empno
  AND m.empno > e.empno
```

An equivalent expression using JOINs can be the following

```
SELECT *
FROM emp m JOIN emp e
       ON e.mgr = m.empno
WHERE m.empno > e.empno
```

The following expressions are equivalent to those given above

```
SELECT *
FROM emp m
WHERE empno >ANY
              (SELECT empno
               FROM emp e
               WHERE e.mgr = m.empno);
```

m. Details of employees that manage other employees (i.e. their employee number should appear in the *mgr* column of some other employee) and whose *empno* value is greater than the *empno* value of all employees they manage.

The following will give you the wrong results. Why?

```
SELECT *
FROM emp m
WHERE empno >ALL
              (SELECT empno
               FROM emp e
               WHERE e.mgr = m.empno);

-- The correct answer is the following - Why?
SELECT *
FROM emp m
WHERE empno <ALL
              (SELECT empno
               FROM emp e
               WHERE e.mgr = m.empno)
  AND EXISTS
          (SELECT *
           FROM emp e
           WHERE e.mgr = m.empno);
```

n. Details of employees who are managed directly by the president.

```
SELECT *
FROM emp
WHERE mgr =
          (SELECT empno
           FROM emp
           WHERE job = 'PRESIDENT');
```

o. Details of departments who have employees earning at least as much as MILLER.

```
SELECT dept.*
FROM emp NATURAL JOIN dept
 AND sal >= (SELECT sal
             FROM emp
             WHERE ename = 'MILLER');
```

The assumption made in the above answer is that there are no more than one employees called MILLER in the emp table. If the above assumption cannot be made then the following SQL statement can be one possible answer:

```
SELECT dept.*
FROM dept
WHERE deptno IN
         (SELECT deptno
          FROM emp
          WHERE sal >= ANY
                      (SELECT sal
                       FROM emp
                       WHERE ename = 'MILLER'));
```

The following query is equivalent.

```
SELECT dept.*
FROM dept
WHERE exists
         (SELECT *
          FROM emp
          WHERE emp.deptno = dept.deptno
```

```
           AND sal >=
                 (SELECT MIN(sal)
                  FROM emp
                  WHERE ename = 'MILLER'));
```

Another possible interpretation of the above query is to list employees who earn at least as much as ALL MILLERs, in which case the above SQL statements should be

```
SELECT dept.*
FROM dept
WHERE EXISTS
        (SELECT *
         FROM emp
         WHERE emp.deptno = dept.deptno
           AND sal >= ALL
                 (SELECT sal
                  FROM emp
                  WHERE ename = 'MILLER'));

SELECT dept.*
FROM dept
WHERE EXISTS
        (SELECT *
         FROM emp
         WHERE emp.deptno = dept.deptno
           AND sal >=
                 (SELECT MAX(sal)
                  FROM emp
                  WHERE ename = 'MILLER'));
```

p. Details of employees who are managed by an employee whose job description is not MANAGER.

```
SELECT e.*
FROM emp e JOIN emp m ON e.mgr = m.empno
WHERE m.job <> 'MANAGER';
```

q. Details of employees whose combined earnings (i.e. salary + commissions) are greater than those of clerks.

```
-- consider the various expressions that demonstrate the use of ANY/ALL
-- in the presence of null values (remember that in returns a null row
-- if the subquery returns no rows).

SELECT *
FROM emp
WHERE (sal + comm) >ANY
                      (SELECT sal+comm
                       FROM emp
                       WHERE job = 'CLERK');

SELECT *
FROM emp
WHERE (sal + comm) >ALL
                      (SELECT sal+comm
                       FROM emp
                       WHERE job = 'CLERK');

SELECT *
FROM emp
WHERE (sal + comm) >ANY
                      (SELECT sal+comm
                       FROM emp
                       WHERE job = 'CLERK'
                           AND comm IS NOT NULL);

SELECT *
FROM emp
WHERE (sal + comm) >ALL
                      (SELECT sal+comm
                       FROM emp
                       WHERE job = 'CLERK'
```

```
                                AND comm IS NOT NULL);
```

A better statement for the above query is

```
SELECT *
FROM emp
WHERE (comm IS NULL
         AND sal >ANY
                    (SELECT sal
                     FROM emp
                     WHERE job = 'CLERK'))
   OR (comm IS NOT NULL
         AND sal + comm >ANY
                           (SELECT sal
                            FROM emp
                            WHERE job = 'CLERK'));

SELECT *
FROM emp
WHERE (comm IS NULL
         AND sal >ALL
                    (SELECT sal
                     FROM emp
                     WHERE job = 'CLERK'))
   OR (comm IS NOT NULL
         AND sal + comm >ALL
                           (SELECT sal
                            FROM emp
                            WHERE job = 'CLERK'));
```

r. Descending order of grade and in ascending order of name, the names, salaries and salary grades of all the employees.

```
SELECT *
FROM emp JOIN salgrade
           ON sal BETWEEN losal AND hisal
ORDER BY grade DESC, ename;
```

PART B

Consider the ***oracle employees*** database (emp – dept – salgrade tables). Write SQL expressions that will retrieve:

a. Details (i.e. deptno and dname) of departments along the employee number, name and job description of any employees working for the department (i.e. the details of departments should be displayed irrespective of whether there are any employees working for the department).

```
SELECT dept.deptno, dname, empno, ename, job
FROM dept LEFT OUTER JOIN emp
         ON (dept.deptno = emp.deptno)
```

The different types of JOIN were not part of the SQL standard until around 2000. Thus, although most of the vendors supported some forms of OUTER JOINs not all forms of OUTER JOIN were supported and the syntax varied from vendor to vendor.
To support OUTER JOINs Oracle (up to and including oracle 8i) provided a syntax that involved the use of "(+)"; any unmatched rows of the left operand were compared with a NULL row and were added to the result provided that the column(s) identified by "(+)" involve a comparison with a null:

```
SELECT dept.deptno, dname, empno, ename, job
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO (+)
```

The "(+)" syntax has been kept for backward compatibility reasons event in the latest versions of oracle.

b. Details (i.e. deptno and dname) of all departments along the name of any employees whose job description is 'CLERK' (i.e. the details of departments should be displayed irrespective of whether there are any CLERK employees working for the department)

```
SELECT dept.deptno, dname, empno, ename, job
FROM dept LEFT OUTER JOIN emp
         ON (dept.deptno = emp.deptno)
```

```
WHERE job = 'CLERK' or job is NULL
```
The NULL predicate is required to include unmatched rows in the result.

An alternative SQL statement can be the following
```
SELECT dept.deptno, dname, empno, ename, job
FROM dept LEFT OUTER JOIN (SELECT *
                           FROM emp
                           WHERE JOB = 'CLERK') t
         ON (dept.deptno = t.deptno)
```

The old SQL syntax equivalent is the following
```
SELECT dept.deptno, dname, empno, ename, job
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO (+)
  AND JOB (+) = 'CLERK';
```
Note that the “(+)” had also to be used with the JOB column to allow for unmatched rows that that will have NULL values in the job column

c. Details (i.e. deptno and dname) of the departments along the name of any employees working for the department and the name of their manager (i.e. the details of departments should be displayed irrespective of whether there are any employees working for the department).

```
SELECT dept.deptno, dname, empno, ename, mname
FROM (SELECT e.empno empno, e.ename ename, m.ename mname, e.deptno
      FROM emp e LEFT OUTER JOIN emp m ON (e.mgr = m.empno)) t
           RIGHT OUTER JOIN dept
               ON (dept.deptno = t.deptno)
```

The old SQL syntax equivalent is the following
```
SELECT dept.deptno, dname, e.empno, e.ename, m.ename
FROM emp e, emp m, dept
WHERE dept.deptno = e.deptno (+)
  AND e.mgr = m.empno (+)
```