# *Database Management I (420-D10-HR)*

# *Lab 13 - Multiple Table Queries II*

Date assigned: Monday, November 14, 2016

Date due: **Monday, November 14, 2016, 3:50pm**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

* 1. write SQL queries using a sub-query
  2. create a table using a sub-query
  3. insert data into a table using a sub-query
  4. update data in a table using a sub-query
  5. delete a row from a table using a sub-query
  6. use IN, ALL and ANY in a sub-query
  7. create a query to do top-n analysis
  8. write a correlated subquery
  9. use the UNION, INTERSECT and MINUS set operations in a query

***Reference:*** SQL Cheat Sheet (Moodle)

[Sub-queries](https://www.techonthenet.com/oracle/subqueries.php)

[Merge SQL Command](https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_9016.htm#SQLRF01606)

ALL, ANY and SOME [Comparison](https://oracle-base.com/articles/misc/all-any-some-comparison-conditions-in-sql) Conditions in SQL

[SQL: CREATE TABLE AS Statement](https://www.techonthenet.com/sql/tables/create_table2.php)

**To uploaded to Moodle:**

1. Rename this document to ***username\_*D10\_L14\_Multiple\_Queries\_II.docx** file containing the SELECT statements you wrote for this lab.

**To Start:**

Copy the **SELECT** statements that you code for the questions in this lab into the appropriate locations in this file.

I also want to see a sample of your output. Include the column headers and the first 5-20 lines of output.

**Marks:**

|  |  |  |
| --- | --- | --- |
| **Question** | **Mark** | **Out of** |
| Part A |  |  |
| 1 |  | 2 |
| 2 |  | 3 |
| 3 |  | 8 |
| 4 |  | 3 |
| 5 |  | 5 |
| 6 |  | 5 |
| 7 |  | 13 |
| 8 |  | 4 |
| 9 |  | 5 |
| 10 |  | 6 |
| Part B |  |  |
| 1 |  | 8 |
| 2 |  | 2 |
| 3 |  | 2 |
| 4 |  | 7 |
| 5 |  | 5 |
| 6a |  | 3 |
| 6b |  | 5 |
| 7 |  | 6 |
| Organization |  | 5 |
| **Total** |  | **97** |

**To Do:**

# Part A

Using the N2 Corporation database from the Shah database package, answer the following.

All queries **must** use a subquery (i.e. a nested query). None of the queries or subqueries should use a join. When asked to find employees, display the employee's last name followed by a comma and the employee's first name.

1. Find the name of the supervisor for employee number 433.

SELECT e1.fName||' '||e1.lName AS "Employee"

FROM nn\_employee e1

WHERE e1.employeeid = (

SELECT e2.supervisor

FROM nn\_employee e2

WHERE e2.employeeid = 433

);



1. Who has the same qualification as Stanley Garner?

SELECT e1.lname||', '||e1.fname AS "Employee"

FROM nn\_employee e1

WHERE e1.qualid = (

SELECT e2.qualid

FROM nn\_employee e2

WHERE e2.fname = 'Stanley'

AND e2.lname = 'Garner'

);



1. Which department has more employees than Department 20?

SELECT deptname, deptid

FROM nn\_dept

WHERE (

SELECT DISTINCT COUNT(deptid)

FROM nn\_employee

WHERE deptid = nn\_dept.deptid

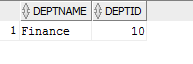
) > (

SELECT COUNT(deptid)

FROM nn\_employee

WHERE deptid = 20

);



1. Which employees have been working in the company longer than Larry Houston?

SELECT e1.lname||', '||e1.fname AS "employee"

FROM nn\_employee e1

WHERE e1.hiredate < (

SELECT e2.hiredate

FROM nn\_employee e2

WHERE e2.fname = 'Larry'

AND e2.lname = 'Houston'

);



1. Create a new table, EMP30, and populate it with employees in Department 30 by using and esiting table and a subquery. Use EmployeeId, Lname,Fname, HireDate, and Salary columns.

DROP TABLE EMP30; --Just so that it doesn’t give errors when run if table exists

CREATE TABLE EMP30 AS (

SELECT EMPLOYEEID, LNAME, FNAME,

HIREDATE, SALARY

FROM nn\_employee

WHERE deptid = 30

1. Add more rows to the EMP30 table with employees in Dpeartment 40. Do not transfer the employee’s salary.

INSERT INTO EMP30 (

SELECT EMPLOYEEID, LNAME, FNAME,

HIREDATE, null

FROM nn\_employee

WHERE deptid = 40

);

1. Update the salary of the newly transferred employees from the EMPLOYEE table to the EMP30 table with a MERGE statement, and insert employees who are not in the EMP30 table.

MERGE INTO EMP30 e1

USING (

SELECT employeeid, lname, fname,

hiredate, salary

FROM nn\_employee

) e2

ON (e1.employeeid = e2.employeeid)

WHEN MATCHED THEN UPDATE SET

e1.fname = e2.fname,

e1.lname = e2.lname,

e1.hiredate = e2.hiredate,

e1.salary = e2.salary

WHEN NOT MATCHED THEN INSERT (e1.employeeid, e1.fname, e1.lname, e1.hiredate, e1.salary)

VALUES (e2.employeeid, e2.fname, e2.lname, e2.hiredate, e2.salary);

1. Find employees with the minimum salary in their own department with the use of a correlated subquery.

SELECT e1.lname||', '||e1.fname AS "Name"

FROM nn\_employee e1

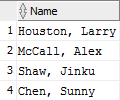
WHERE e1.salary IN (

SELECT MIN(salary)

FROM nn\_employee

GROUP BY deptid

);



1. Use set operator and subquery to find employees who do not have any dependents.

SELECT DISTINCT e1.lname||', '||e1.fname AS "Name"

FROM nn\_employee e1

WHERE e1.employeeid NOT IN (

SELECT DISTINCT employeeid

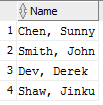
FROM nn\_employee

minus

SELECT employeeid

FROM nn\_dependent

);



1. Write a subquery that finds the average salary by each department. Check to find if employee 543’s salary satisfies the =ANY, <ANY,>ANY, <ALL,or >ALL condition against those departmental average salaries.

SELECT fname||', '||lname AS "Name"

FROM nn\_employee

WHERE employeeid = 543 AND salary = ANY

(SELECT AVG(salary) FROM nn\_employee GROUP BY deptid);



SELECT fname||', '||lname AS "Name"

FROM nn\_employee

WHERE employeeid = 543 AND salary < ANY

(SELECT AVG(salary) FROM nn\_employee GROUP BY deptid);



SELECT fname||', '||lname AS "Name"

FROM nn\_employee

WHERE employeeid = 543 AND salary > ANY

(SELECT AVG(salary) FROM nn\_employee GROUP BY deptid);



SELECT fname||', '||lname AS "Name"

FROM nn\_employee

WHERE employeeid = 543 AND salary < ALL

(SELECT AVG(salary) FROM nn\_employee GROUP BY deptid);

SELECT fname||', '||lname AS "Name"

FROM nn\_employee

WHERE employeeid = 543 AND salary > ALL

(SELECT AVG(salary) FROM nn\_employee GROUP BY deptid);



# Part B

## List the id, name, phone number and status of all faculty members who are either a department co-ordinator or who are not teaching any courses. (***Hint***: First write two separate queries – one to find all co-ordinators and then one to find all teachers not teaching any courses. To find the teachers who are not teaching any courses, use a sub-query with **not in**. When using **not in**, values cannot be null. To output the *Status*, simply code the literal you want to output in the select clause.) Sort the output by faculty name. A sample run is shown here.

|  |  |  |  |
| --- | --- | --- | --- |
| **Faculty Id** | **Name** | **Phone** | **Status** |
| 111 | Jones | 525 | Co-ordinator |
| 222 | Williams | 533 | Co-ordinator |
| 222 | Williams | 533 | Not teaching |
| 235 | Vajpayee | 577 | Not teaching |
| 333 | Collins | 599 | Co-ordinator |
| 333 | Collins | 599 | Not teaching |
| 391 | Turanyi | 2071 | Co-ordinator |
| 391 | Turanyi | 2071 | Not teaching |
| 402 | McDonald | 2083 | Not teaching |
| 407 | Patterson | 2701 | Not teaching |
| 444 | Rivera | 544 | Co-ordinator |
| 555 | Chang | 587 | Co-ordinator |
| 555 | Chang | 587 | Not teaching |

SELECT DISTINCT f.facultyid, name, f.phone, 'Co-ordinator'

FROM iu\_faculty f, iu\_department d

WHERE d.facultyid = f.facultyid

UNION

SELECT DISTINCT f.facultyid, name, f.phone, 'Not teaching'

FROM iu\_faculty f, iu\_crssection c

WHERE f.facultyid NOT IN (

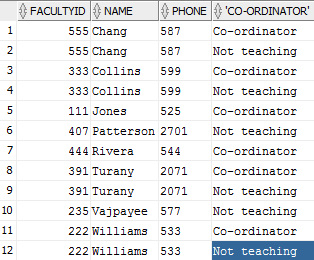
SELECT f.facultyid

FROM iu\_faculty f, iu\_crssection c

WHERE f.facultyid = c.facultyid

)

ORDER BY name;



## List the id, name and phone number all the co-ordinators who are also teaching. Use Intersect. (Do not list the status.)

|  |  |  |
| --- | --- | --- |
| **Faculty Id** | **Name** | **Phone** |
| 111 | Jones | 525 |
| 444 | Rivera | 544 |

SELECT DISTINCT f.facultyid, f.name, f.phone

FROM iu\_faculty f, iu\_department d, iu\_crssection c

WHERE d.facultyid = f.facultyid AND

f.facultyid IN c.facultyid;

## 

## Change the previous query to use a minus to list all co-ordinators who are not teaching.

|  |  |  |
| --- | --- | --- |
| **Faculty Id** | **Name** | **Phone** |
| 222 | Williams | 533 |
| 333 | Collins | 599 |
| 391 | Turanyi | 2071 |
| 555 | Chang | 587 |

SELECT DISTINCT f.facultyid, f.name, f.phone

FROM iu\_faculty f, iu\_department d, iu\_crssection c

WHERE d.facultyid = f.facultyid

minus

SELECT f.facultyid, f.name, f.phone

FROM iu\_faculty f, iu\_crssection c

WHERE f.facultyid = c.facultyid;

## 

## List the course section id, the course number and name, the section number and the maxcount for all course sections that will fit in all the labs and classrooms. Use ALL.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CSID** | **COURSEID** | **TITLE** | **SECTION** | **MAXCOUNT** |
| 1210 | CIS253 | Database Systems | 31 | 1 |
| 4202 | 420C30 | Web Programming III | 01 | 24 |
| 4201 | 420B31 | Programming III | 01 | 24 |
| 4203 | 420D10 | Database Management I | 01 | 24 |
| 4204 | 420E11 | Systems I | 01 | 24 |

SELECT DISTINCT c1.csid, c1.courseid, c2.title, c1.section, c1.maxcount

FROM iu\_crssection c1, iu\_course c2

WHERE c1.courseid = c2.courseid

AND c1.maxcount <= ALL (

SELECT l.capacity

FROM iu\_location l

);

## 

## List the student id and name of all students who have completed any courses. (A course is completed if it has a value in the final column.) Use ANY.

|  |  |
| --- | --- |
| **Student Id** | **Name** |
| 00100 | Diaz, Jose |
| 00101 | Tyler, Mickey |
| 00103 | Rickles, Deborah |

SELECT DISTINCT s.studentid, s.last||', '||s.first AS "Name"

FROM iu\_student s, iu\_registration

WHERE s.studentid = ANY (

SELECT r.studentid

FROM iu\_registration r

WHERE r.final IS NOT NULL

);

## 

## a. Add a row to the term table for the winter 2017 semester. The start date is Jan. 19 and the end date is May 12.

INSERT INTO iu\_term (termid, termdesc, startdate, enddate)

VALUES ('WN17', 'Winter 2017',

to\_date('19-JAN-17','DD-MON-RR'),

to\_date('12-MAY-17','DD-MON-RR'));

### Add rows to the table for course sections for the winter 2017 semester using a sub-query. All the courses that were offered in winter 2003 will be offered in winter 2017. The csid should be the 300 higher than the csid for the equivalent course in winter 2017. All other attributes will be the same.

## Brian Lee has decided to take the term off. Delete all his registration records for the 'SP03' term. Use subqueries to get his studentid and the spring 2003 course sections.

DELETE FROM iu\_registration

WHERE studentid = (

SELECT s.studentid

FROM iu\_student s

WHERE s.last = 'Lee'

AND s.first = 'Brian'

)

AND csid = ANY(

SELECT c.csid

FROM iu\_crssection c

WHERE c.termid='SP03'

);

