Ryan Paw
Week 2, Lab
ANA660

I, Ryan Paw, finished and completed this week's reading requirements and lab activities. I created and ran the code in SQL Plus myself.

Complete the following Lab Exercises in the textbook
Lab 10.1 Exercises (question c)
Lab 10.2 Exercises (all the questions)
Lab 8.1 Exercises (all the questions)
Lab 8.2 Exercises (all the questions)

10.1, c)

c) Rewrite the following SQL statement using an outer join.

```
SELECT course_no, description
  FROM course c
 WHERE NOT EXISTS
       (SELECT 'X'
          FROM section
         WHERE c.course_no = course_no)
COURSE_NO DESCRIPTION
--------- --------------------------------
       80 Programming Techniques
      430 Java Developer III

2 rows selected.
```

Code/Output:

```
select c.course_no, c.description
from course c, section s
where c.course_no = s.course_no(+)
and s.course_no is NULL
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.173 seconds

	COURSE_NO	DESCRIPTION
1	80	Programming Techniques
2	430	Java Developer III

Lab 10.2 Exercises

a) For SECTION_ID 86, determine which students received a lower grade on their final than on their midterm. In your result, list the STUDENT_ID and the grade for the midterm and final.

b) What problem does the following query solve?

```
SELECT DISTINCT a.student_id, a.first_name, a.salutation
  FROM student a, student b
 WHERE a.salutation <> b.salutation
   AND b.first_name = a.first_name
   AND a.student_id <> b.student_id
 ORDER BY a.first_name
```

c) Display the student ID, last name, and street address of students living at the same address and zip code.

d) Write a query that shows the course number, course description, prerequisite, and description of the prerequisite. Include courses without any prerequisites. (This requires a self-join and an outer join.)

10.2, a)
Code/Output:

```
select g1.student_id, g2.numeric_grade "midterm grade",
       g1.numeric_grade "final grade"
from grade g1 join grade g2
on (g1.section_id = g2.section_id
and g1.student_id = g2.student_id)
where g1.grade_type_code = 'FI'
and g1.section_id = 86
and g2.grade_type_code = 'MT'
and g1.numeric_grade < g2.numeric_grade;
```

Query Result

All Rows Fetched: 3 in 0.295 seconds

	STUDENT_ID	midterm grade	final grade
1	102	90	85
2	108	91	76
3	211	92	77

10.2, b)
Code/Output:

```
/*10.2, b*/
select distinct s1.student_id, s1.first_name, s1.salutation
from student s1, student s2
where s1.salutation <> s2.salutation
and s1.first_name = s2.first_name
and s1.student_id <> s2.student_id
order by s1.first_name;
```

Query Result

All Rows Fetched: 17 in 0.155 seconds

	STUDENT_ID	FIRST_NAME	SALUTATION
1	124	Daniel	Mr.
2	242	Daniel	Mr.
3	315	Daniel	Ms.
4	230	George	Mr.
5	259	George	Mr.
6	275	George	Mr.
7	279	George	Ms.
8	303	George	Mr.
9	319	George	Mr.
10	103	J.	Ms.
11	197	J.	Mr.

10.2, c)
Code/Output:

```sql
/*10.2, c*/
select distinct s1.student_id, s1.last_name, s1.street_address
from student s1, student s2
where s1.street_address = s2.street_address
and s1.zip = s2.zip
and s1.student_id <> s2.student_id
order by s1.street_address;
```

Query Result

All Rows Fetched: 22 in 0.199 seconds

	STUDENT_ID	LAST_NAME	STREET_ADDRESS
1	390	Greenberg	105-34 65th Ave. #6B
2	392	Saliternan	105-34 65th Ave. #6B
3	234	Brendler	111 Village Hill Dr.
4	380	Krot	111 Village Hill Dr.
5	242	Ordes	117 Knapp Ave.
6	148	Orent	117 Knapp Ave.
7	337	Cross	131-57 230th
8	305	Larcia	131-57 230th
9	284	Lindeman	1614 64th St.
10	184	Zuckerberg	1614 64th St.
11	371	Anglin	199-46 21st Ave.

10.2, d)

Code/Output:

```sql
/*10.2, d*/
select c1.course_no, substr(c1.description,1,15) course_descr,
    c1.prerequisite, substr(c2.description,1,15) pre_req_descr
from course c1 left outer join course c2
on c1.prerequisite = c2.course_no
order by 1;
```

Query Result

All Rows Fetched: 30 in 0.133 seconds

	COURSE_NO	COURSE_DESCR	PREREQUISITE	PRE_REQ_DESCR
1	10	Technology Conc	(null)	(null)
2	20	Intro to Inform	(null)	(null)
3	25	Intro to Progra	140	Systems Analysi
4	80	Programming Tec	204	Intro to SQL
5	100	Hands-On Window	20	Intro to Inform
6	120	Intro to Java P	80	Programming Tec
7	122	Intermediate Ja	120	Intro to Java P
8	124	Advanced Java P	122	Intermediate Ja
9	125	Java Developer	122	Intermediate Ja
10	130	Intro to Unix	310	Operating Syste
11	132	Basics of Unix	130	Intro to Unix

Lab 8.1 Exercises

a) Write a SQL statement that displays the first and last names of the students who registered first.

b) Show the sections with the lowest course cost and a capacity equal to or lower than the average capacity. Also display the course description, section number, capacity, and cost.

c) Select the course number and total capacity for each course. Show only the courses with a total capacity less than the average capacity of all the sections.

d) Choose the most ambitious students: Display the STUDENT_ID for the students enrolled in the most sections.

e) Select the STUDENT_ID and SECTION_ID of enrolled students living in zip code 06820.

f) Display the course number and course description of the courses taught by instructor Fernand Hanks.

g) Select the last names and first names of students not enrolled in any class.

h) Determine the STUDENT_ID and last name of students with the highest FINAL_GRADE for each section. Also include the SECTION_ID and the FINAL_GRADE columns in the result.

i) Select the sections and their capacity, where the capacity equals the number of students enrolled.

8.1, a)

Code/Output:

```
/*8.1, a*/
select first_name, last_name
from student
where registration_date =
    (select min(registration_date)
    from student);
```

Query Result

SQL | All Rows Fetched: 8 in 0.202 seconds

	FIRST_NAME	LAST_NAME
1	Fred	Crocitto
2	J.	Landry
3	Laetia	Enison
4	Angel	Moskowitz
5	Judith	Olvsade
6	Catherine	Mierzwa
7	Judy	Sethi
8	Larry	Walter

8.1, b)

Code/Output:

```sql
/*8.1, b*/
select c.description, s.section_no, c.cost, s.capacity
from course c, section s
where c.course_no = s.course_no
and s.capacity <=
    (select avg(capacity)
    from section)
and c.cost=
    (select min(cost)
    from course);
```

Query Result

SQL | All Rows Fetched: 6 in 0.483 seconds

	DESCRIPTION	SECTION_NO	COST	CAPACITY
1	Unix Tips and Techniques	2	1095	15
2	Unix Tips and Techniques	4	1095	15
3	Intro to the Internet	1	1095	12
4	Intro to the Internet	2	1095	15
5	Intro to the BASIC Language	1	1095	10
6	Intro to the BASIC Language	2	1095	15

8.1, c)
Code/Output:

```sql
/* 8.1, c*/
select course_no, sum(capacity)
from section
group by course_no
having sum(capacity) <
    (select avg(capacity)
    from section);
```

Query Result

SQL | All Rows Fetched: 2 in 0.06 seconds

	COURSE_NO	SUM(CAPACITY)
1	10	15
2	144	15

8.1, d)
Code/Output:

```
/*8.1, d) */
select student_id, count(*)
from enrollment
group by student_id
having count(*) =
    (select max(count(*))
    from enrollment
    group by student_id)
```

Query Result

SQL | All Rows Fetched: 2 in 2

	STUDENT_ID	COUNT(*)
1	124	4
2	214	4

8.1, e)
Code/Output:

```
/*8.1, e)*/
select student_id, section_id
from enrollment
where student_id in
    (select student_id
    from student
    where zip = '06820');
```

Query Result

SQL | All Rows Fetched: 1 in 0.45 seconds

	STUDENT_ID	SECTION_ID
1	240	81

8.1, f)
Code/Output:

```sql
/*8.1, f)*/
select course_no, description
from course
where course_no in
    (select course_no
    from section
    where instructor_id in
        (select instructor_id
        from instructor
        where last_name = 'Hanks'
        and first_name = 'Fernand'));
```

Query Result

SQL | All Rows Fetched: 9 in 0.752 seconds

	COURSE_NO	DESCRIPTION
1	25	Intro to Programming
2	120	Intro to Java Programming
3	122	Intermediate Java Programming
4	125	Java Developer I
5	134	Advanced Unix Admin
6	140	Systems Analysis
7	146	Java for C/C++ Programmers
8	240	Intro to the BASIC Language
9	450	DB Programming with Java

8.1, g)

Code/Output:

```sql
/*8.1, g)*/
select last_name, first_name
from student
where student_id not in
    (select student_id
    from enrollment);
```

Query Result

SQL | Fetched 50 rows in 0.504 seconds

	LAST_NAME	FIRST_NAME
1	Lindeman	Salewa
2	Sikinger	Paul
3	Kelly	Robin
4	Ellman	Rosemary
5	Murray	Shirley
6	Robles	Brian
7	Dewitt	D.
8	Cadet	Austin V.
9	M. Orent	Frank
10	Winnicki	Yvonne

8.1, h)

Code/Output:

```
/*8.1, h*/
select s.student_id, s.last_name, e.final_grade,
    e.section_id
from enrollment e, student s
where e.student_id = s.student_id
and (e.final_grade, e.section_id) in
    (select max(final_grade), section_id
    from enrollment
    group by section_id);
```

Query Result

SQL | All Rows Fetched: 1 in 0.926 seconds

	STUDENT_ID	LAST_NAME	FINAL_GRADE	SECTION_ID
1	102	Crocitto	92	89

8.1, i)

Code/Output:

```
/*8.1, i*/
select section_id, capacity
from section
where (section_id, capacity) in
    (select section_id, count(*)
    from enrollment
    group by section_id);
```

Query Result

SQL | All Rows Fetched: 1 in 0.198 seconds

	SECTION_ID	CAPACITY
1	99	12

Lab 8.2 Exercises

a) Explain what the following correlated subquery accomplishes.

```
SELECT section_id, course_no
  FROM section s
 WHERE 2 >
       (SELECT COUNT(*)
          FROM enrollment
         WHERE section_id = s.section_id)
SECTION_ID COURSE_NO
---------- ---------
        79       350
        80        10
...
       145       100
       149       120

27 rows selected.
```

b) List the sections where the enrollment exceeds the capacity of a section and show the number of enrollments for the section using a correlated subquery.

c) Write a SQL statement to determine the total number of students enrolled, using the EXISTS operator. Count students enrolled in more than one course as one.

d) Show the STUDENT_ID, last name, and first name of each student enrolled in three or more classes.

e) Which courses do not have sections assigned? Use a correlated subquery in the solution.

f) Which sections have no students enrolled? Use a correlated subquery in the solution and order the result by the course number, in ascending order.

This subquery shows the course_no and section_id of sections that have less than 2 students enrolled. This is a correlated subquery. In the subquery, it counts the instances in enrollment and compares it if it is less than 2.

Code/Output:

```
/*8.2, a*/
select section_id, course_no
from section s
where 2 >
    (select count(*)
    from enrollment
    where section_id=s.section_id);
```

Query Result

SQL | All Rows Fetched: 27 in 0.164 seconds

	SECTION_ID	COURSE_NO
1	79	350
2	80	10
3	93	25
4	96	204
5	97	210
6	98	220
7	102	240
8	109	450

8.2, b)

Code/Output:

```
/*8.2, b*/
select section_id, count(*)
from enrollment e
group by section_id
having count(*) >
    (select capacity
    from section
    where e.section_id=section_id);
```

Query Result

All Rows Fetched: 1 in 0.344 seconds

	SECTION_ID	COUNT(*)
1	101	12

8.2, c)

Code/Output:

```
/*8.2, c*/
select first_name, last_name, student_id
from student s
where exists
    (select null
    from enrollment
    where s.student_id=student_id
    group by student_id
    having count(*) >= 3);
```

Query Result

All Rows Fetched: 7 in 0.373 seconds

	FIRST_NAME	LAST_NAME	STUDENT_ID
1	Daniel	Wicelinski	124
2	Salewa	Zuckerberg	184
3	Yvonne	Williams	214
4	Reynaldo	Chatman	215
5	Janet	Jung	232
6	Roger	Snow	238
7	Evan	Fielding	250

8.2, d)

Code/Output:

```
/*8.2, d*/
select first_name, last_name, student_id
from student s
where exists
    (select null
    from enrollment
    where s.student_id=student_id
    group by student_id
    having count(*)>=3);
```

Query Result

SQL | All Rows Fetched: 7 in 0.445 seconds

	FIRST_NAME	LAST_NAME	STUDENT_ID
1	Daniel	Wicelinski	124
2	Salewa	Zuckerberg	184
3	Yvonne	Williams	214
4	Reynaldo	Chatman	215
5	Janet	Jung	232
6	Roger	Snow	238
7	Evan	Fielding	250

8.2, e)

Code/Output:

```
/*8.2, e*/
select course_no, description
from course c
where not exists
    (select 'X'
    from section
    where c.course_no=course_no);
```

Query Result

SQL | All Rows Fetched: 2 in 0.291 seconds

	COURSE_NO	DESCRIPTION
1	80	Programming Techniques
2	430	Java Developer III

8.2, f)

Code/Output:

```sql
/*8.2, e*/
select course_no, section_id
from section s
where not exists
    (select null
    from enrollment
    where s.section_id=section_id)
order by course_no;
```

Query Result | Script Output | Query Result 1

SQL | All Rows Fetched: 14 in 0.159 seconds

	COURSE_NO	SECTION_ID
1	25	93
2	124	129
3	125	134
4	130	136
5	132	139
6	134	110
7	135	114
8	140	118