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 x Query Result x

SCRIPT Output x Query Result x

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.)

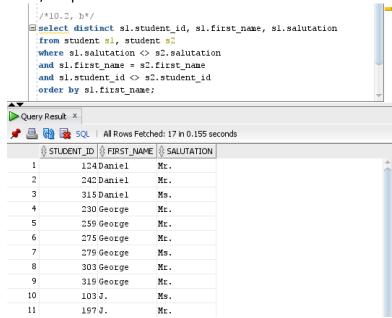
10.2, a)

Code/Output:

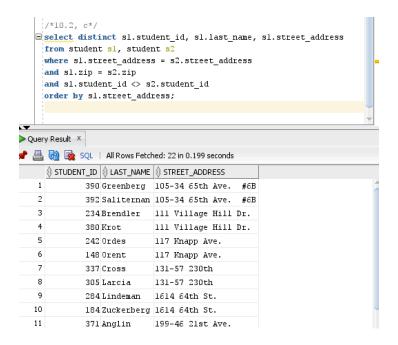
```
select gl.student_id, g2.numeric_grade "midterm grade",
            gl.numeric_grade "final grade"
     from grade gl join grade g2
     on (gl.section_id = g2.section_id
     and gl.student_id = g2.student_id)
     where gl.grade_type_code = 'FI'
     and gl.section_id = 86
     and g2.grade_type_code = 'MT'
     and gl.numeric_grade < g2.numeric_grade;</pre>
➤ Query Result ×
📌 搗 祔 攻 SQL | All Rows Fetched: 3 in 0.295 seconds
      1
             102
                          90
                                    85
    2
             108
                                    76
    3
                          92
             211
                                    77
```

10.2, b)

Code/Output:



10.2, c) Code/Output:



10.2, d)

```
/*10.2, d*/
  select cl.course_no, substr(cl.description,1,15) course_descr,
        cl.prerequisite, substr(c2.description,1,15) pre_req_descr
    from course cl left outer join course c2
    on cl.prerequisite = c2.course_no
    order by 1;
▶ Query Result ×
🏲 📇 🙌 퀋 SQL | 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
                                            120 Intro to Java P
             122 Intermediate Ja
   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 Rasics of Haiv
                                            130 Intro to Univ
```

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 o6820.
- **1)** 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 ×
r 🖶 🙌 嶳 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
                 01vsade
   6 Catherine
                 Mierzwa
   7 Judy
                 Sethi
   8 Larry
                 Walter
```

8.1, b)

```
/*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 ×
🏲 搗 🙌 퀋 SQL | All Rows Fetched: 6 in 0.483 seconds
    ⊕ DESCRIPTION

⊕ SECTION_NO | ⊕ COST | ⊕ CAPACITY

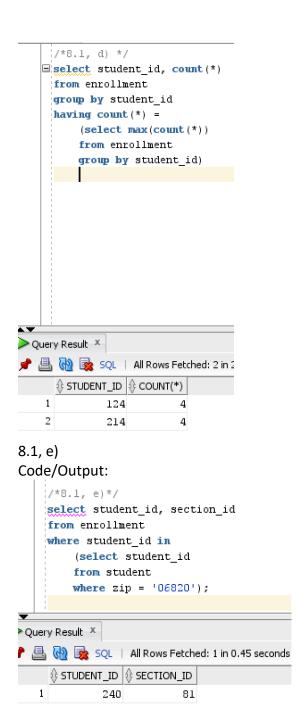
   1 Unix Tips and Techniques
                                               1095
                                                           15
   2 Unix Tips and Techniques
                                              1095
                                                           15
   3 Intro to the Internet
                                              1095
                                                           12
   4 Intro to the Internet
                                             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:
    /* 8.1, c*/
  □|select course_no, sum(capacity)
    from section
    group by course_no
    having swm(capacity) <
         (select avg(capacity)
         from section);
Query Result X
 📇 🙌 嶳 SQL | All Rows Fetched: 2 in 0.06 seconds
     $ COURSE_NO | $ SUM(CAPACITY)
  1
                                15
               10
```

15

8.1, d) Code/Output:

144

2



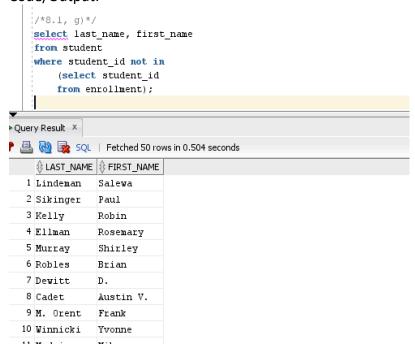
8.1, f) Code/Output:

```
/*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 X
🥜 🖺 🙌 🔯 SQL | All Rows Fetched: 9 in 0.752 seconds

⊕ COURSE_NO | ⊕ DESCRIPTION

               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:



8.1, h)

Code/Output:

8.1, i)

Lab 8.2 Exercises

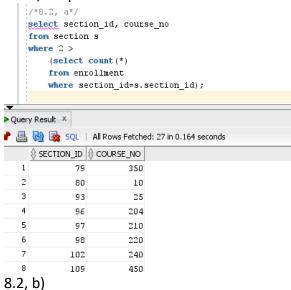
27 rows selected.

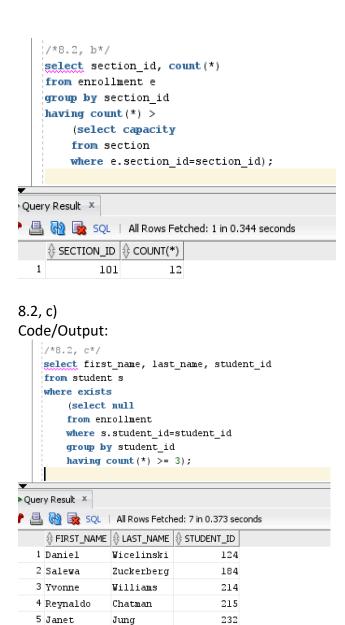
a) Explain what the following correlated subquery accomplishes.

- **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.
- $\underline{\textbf{d}})$ Show the STUDENT_ID, last name, and first name of each student enrolled in three or more classes.
- $\underline{\mathbf{e}})$ Which courses do not have sections assigned? Use a correlated subquery in the solution.
- **1** 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, d) Code/Output:

6 Roger

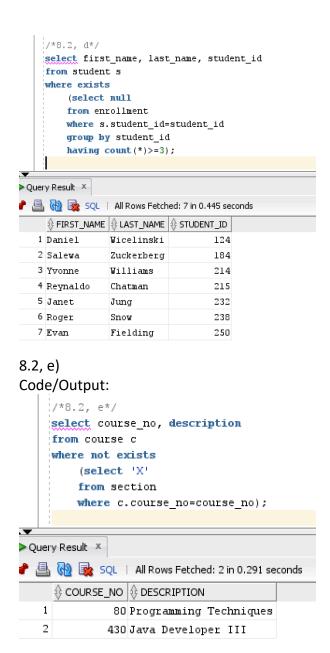
7 Evan

Snow

Fielding

238

250



8.2, f) Code/Output:

```
/*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
    1
            25
                       93
  2
           124
                      129
  3
           125
                      134
  4
           130
                      136
  5
           132
                      139
  6
           134
                      110
  7
                      114
           135
           140
                      118
```