Contenido

[Capitulo 2 2](#_Toc89732677)

[**Chapter 3.** 15](#_Toc89732678)

[**Character, Number and Miscellaneous Functions** 15](#_Toc89732679)

[**3.1.2. Concatenate Strings** 19](#_Toc89732680)

[**Lab 3.2 3.2.1. Use Number Functions and Perform Mathematical** 20](#_Toc89732681)

[**Computations**. 20](#_Toc89732682)

[**3.3.1. Apply Substitution Functions and Other Miscellaneous Functions** 21](#_Toc89732683)

[3.3.2. Utilize the Power of the DECODE Function and the CASE Expression 22](#_Toc89732684)

[**Chapter 4. Date and Conversion Functions** 24](#_Toc89732685)

[Chapter 5 Equijoins 27](#_Toc89732686)

[Lab 6.1 The Two-Table Join 27](#_Toc89732687)

[Lab 6.1 Exercises 29](#_Toc89732688)

[Lab 6.2 Exercises 32](#_Toc89732689)

[6.2.1. Join Three or More Tables 32](#_Toc89732690)

[Chapter 7. Subqueries 34](#_Toc89732691)

[Lab 7.1 Exercises 34](#_Toc89732692)

[7.2 exercises 43](#_Toc89732693)

[Lab 7.4 Exercise 45](#_Toc89732694)

[Chapter 8. Set Operators 48](#_Toc89732695)

[Lab 8.1 Exercises 49](#_Toc89732696)

[Lab 8.2 Exercises 52](#_Toc89732697)

[Chapter 9. Complex Joins 54](#_Toc89732698)

[Lab 9.1 Exercises 56](#_Toc89732699)

# Capitulo 2

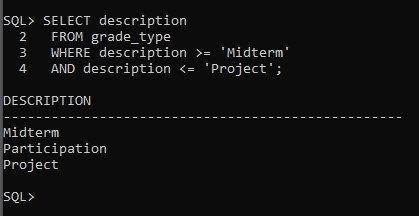
rem [Rischert, 2004,148]

SELECT description

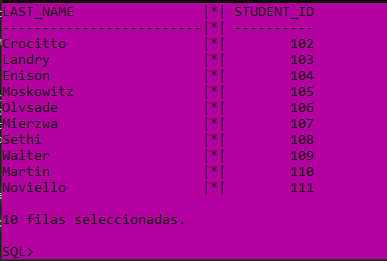
FROM grade\_type

WHERE description >= 'Midterm'

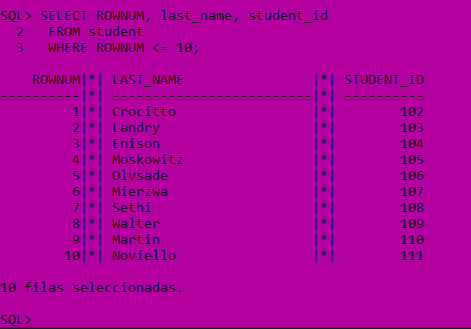
AND description <= 'Project';

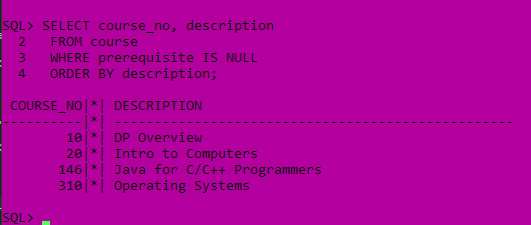


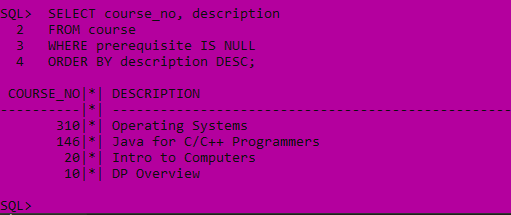
e) Execute the following query and determine how many rows the query returns. SELECT last\_name, student\_id FROM student WHERE ROWNUM <= 10

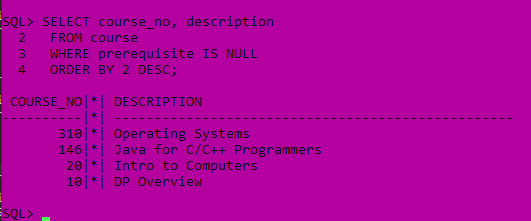


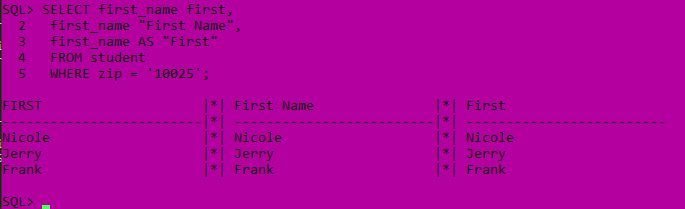
SELECT ROWNUM, last\_name, student\_id FROM student WHERE ROWNUM <= 10

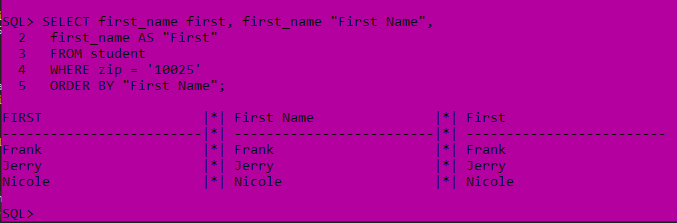


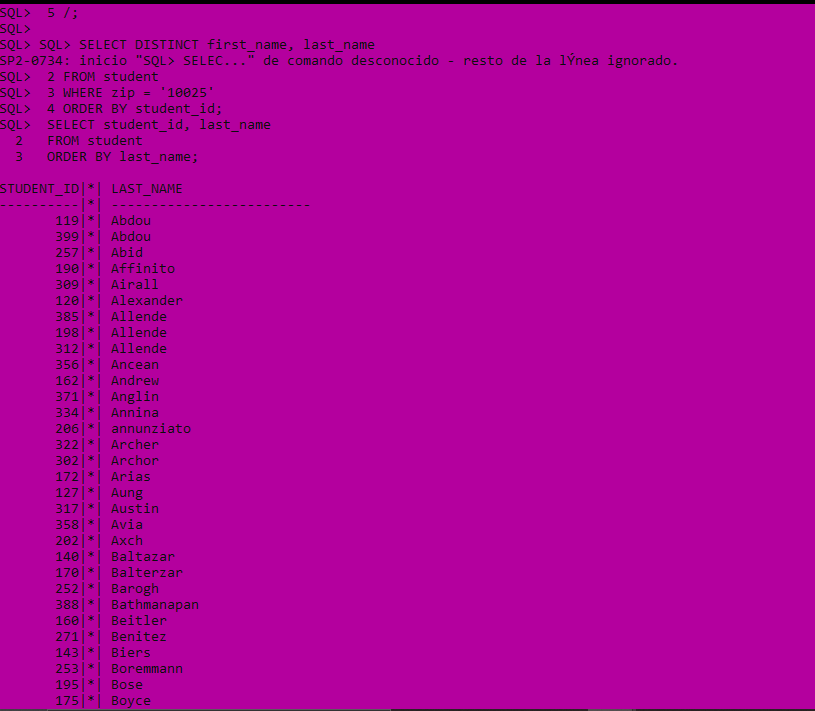


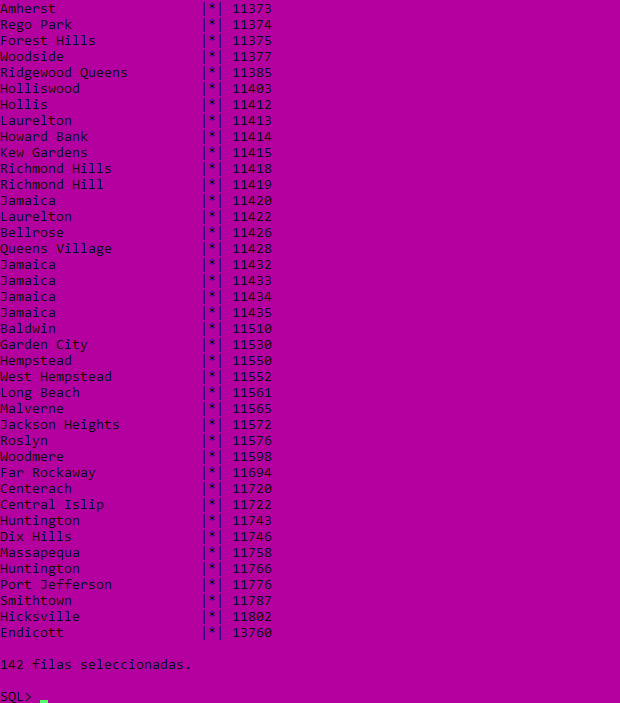


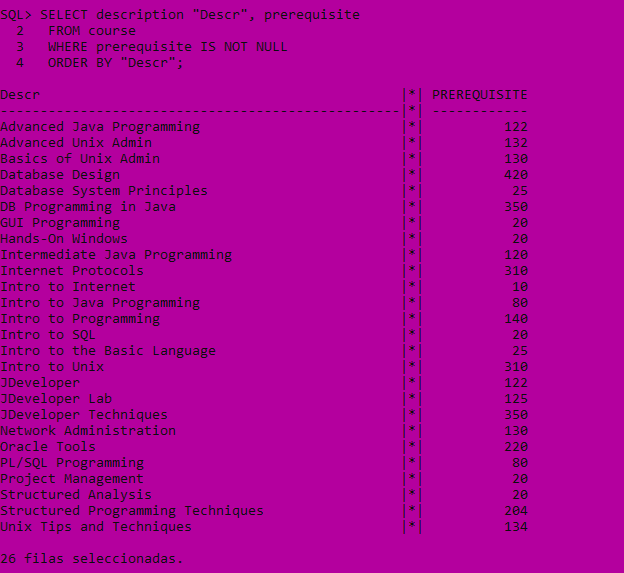


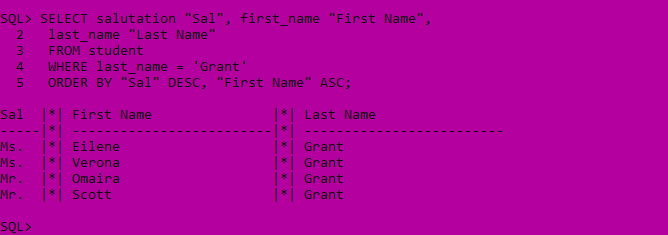




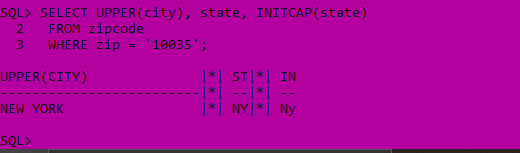


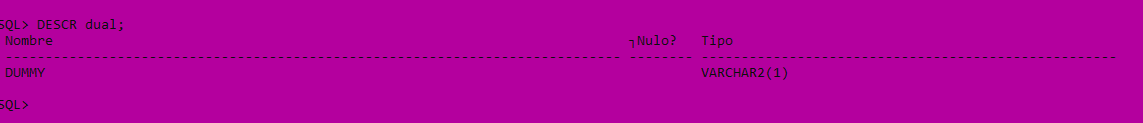


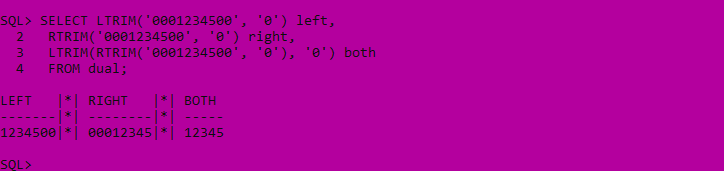


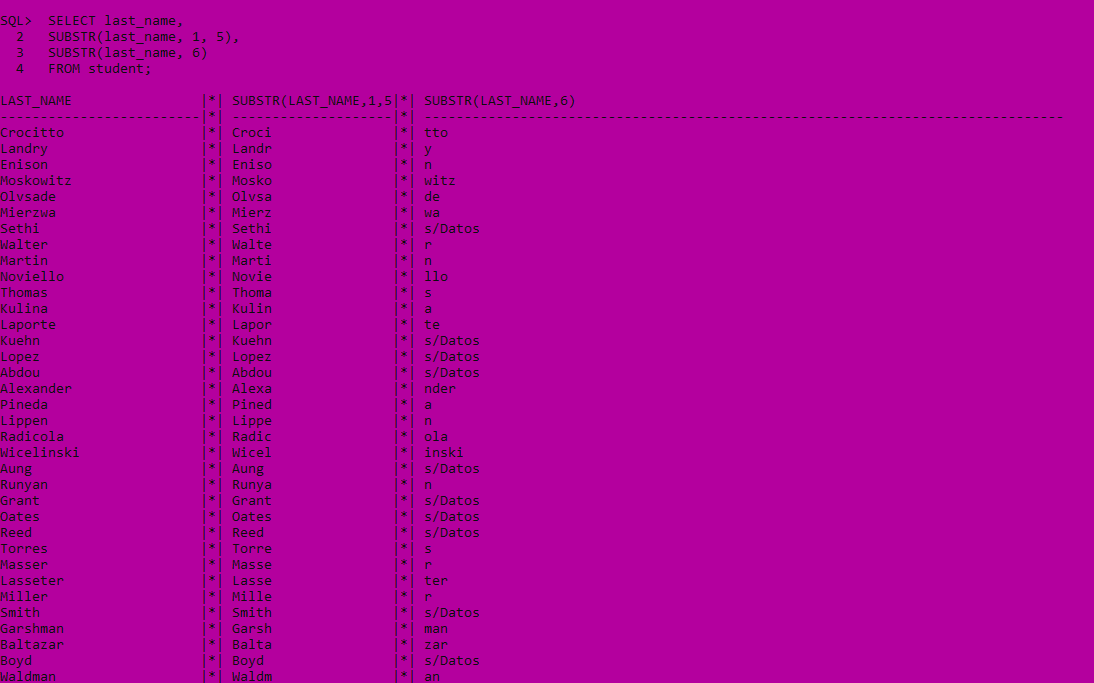




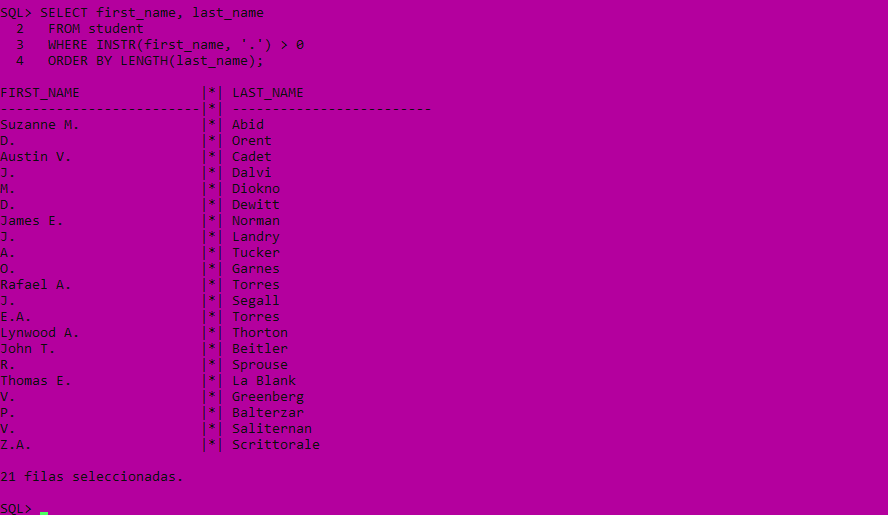


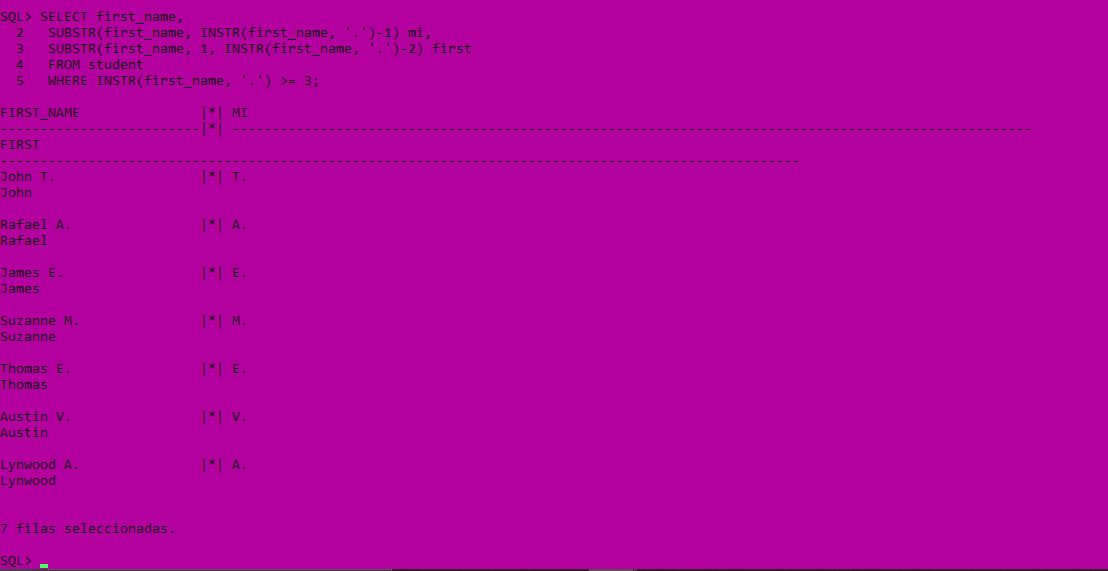




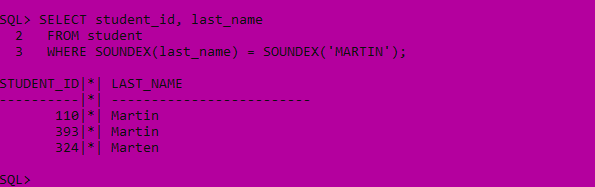


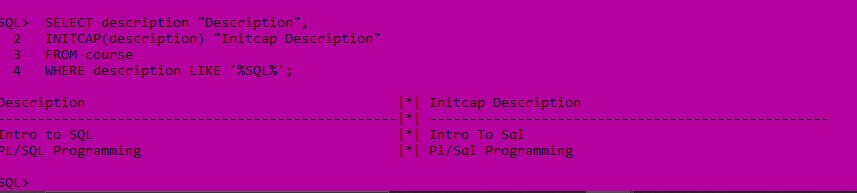


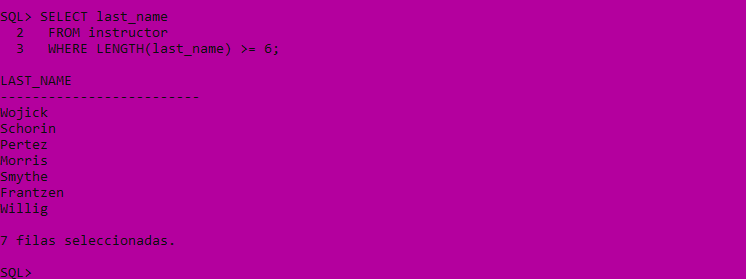






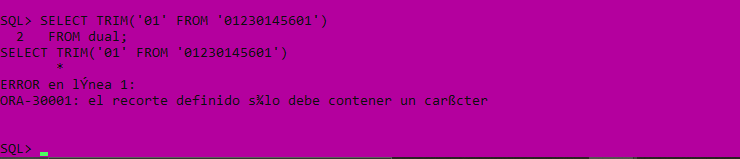


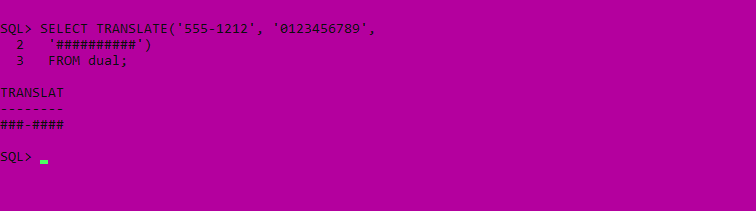


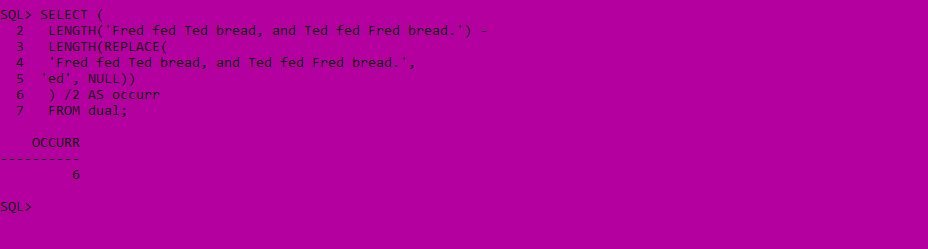


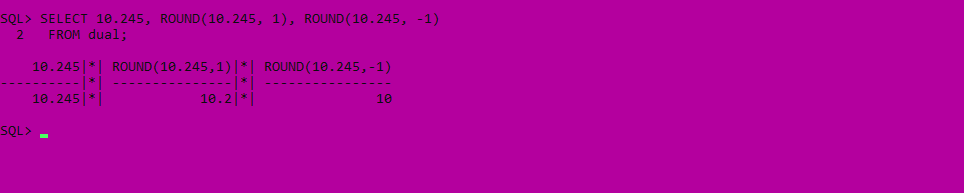




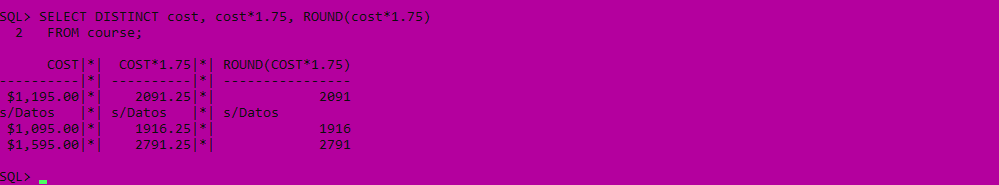


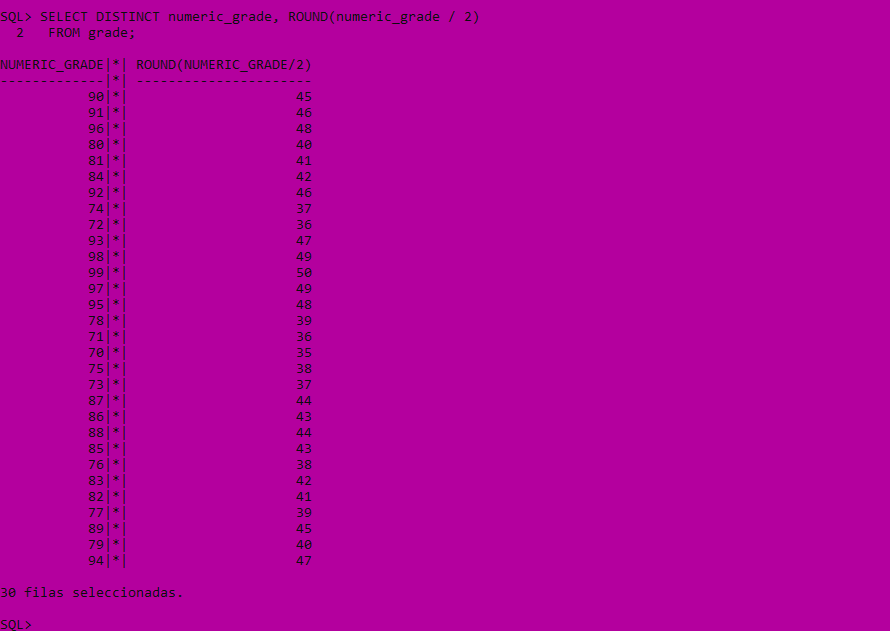


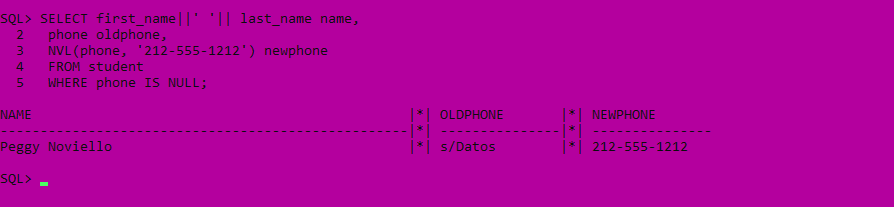


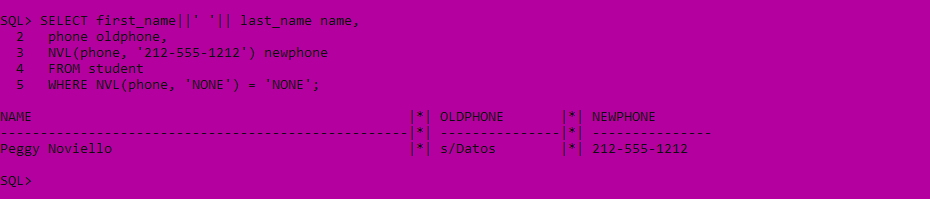














# **Chapter 3.**

## **Character, Number and Miscellaneous Functions**

Lab 3.1 Exercises

3.1.1. Use a Character Function in a SQL Statement

a) Execute the following SQL statement. Based on the result, what is the purpose of the INITCAP function?

SELECT description "Description",

INITCAP(description) "Initcap Description"

FROM course

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente WHERE description LIKE '%SQL%'

Con la primer sentencia podemos observar que al momento de ingresarla no funciona.

b) Write the question answered by the following SQL statement.

SELECT last\_name

FROM instructor

WHERE LENGTH(last\_name) >= 6

Texto

Descripción generada automáticamente

La sentencia nos devuelve los apellidos que tengan seis o mas letras.

c) Describe the result of the following SQL statement. Pay particular attention to the egative number parameter.

SELECT SUBSTR('12345', 3),

SUBSTR('12345', 3, 2),

SUBSTR('12345', -4, 3)

FROM dual

Texto

Descripción generada automáticamente

En la Columba SUBSTR toma los números a partir de la tercera posición, en la segunda Columba, toma los primeros dos valores a partir del tercero y en la tercer columna como es un número negativo empieza desde el final de la cadena, por lo tanto comienza a contar a partir del dos, tomando solamente tres caracteres.

d) Based on the result of the following SQL statement, describe the purpose of the LTRIM and RTRIM functions.

SELECT zip, LTRIM(zip, '0'), RTRIM(ZIP, '4')

FROM zipcode

ORDER BY zip

Gráfico, Texto, Gráfico de rectángulos

Descripción generada automáticamente

e) What do you observe when you execute the next statement? How would you change the statement to achieve the desired result?

SELECT TRIM('01' FROM '01230145601')

FROM dual

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

f) What is the result of the following statement?

SELECT TRANSLATE('555-1212', '0123456789',

'##########')

FROM dual

Imagen que contiene Texto

Descripción generada automáticamente

g) Write the SQL statement to retrieve those students that have a last name with the lowercase letter 'o' occurring three or more times.

Texto

Descripción generada automáticamente

h) The following statement determines how many times the string 'ed' occurs in the phrase 'Fred fed Ted bread, and Ted fed Fred bread.' Explain how this is accomplished.

SELECT (

LENGTH('Fred fed Ted bread, and Ted fed Fred bread.') -

LENGTH(REPLACE('Fred fed Ted bread, and Ted fed Fred bread.',

'ed', NULL))

) /2 AS occurr

FROM dual

OCCURR

----------

6

1 row selected.

Texto

Descripción generada automáticamente con confianza media

## **3.1.2. Concatenate Strings**

1. Write a SELECT statement that returns each instructor's last name, followed by a comma and a space, followed by the instructor's first name, all in a single column in the result set.

Texto

Descripción generada automáticamente con confianza media

b) Using functions in the SELECT list, WHERE, and ORDER BY clauses, write the SELECT statement that returns course numbers and course descriptions from the COURSE table and looks like the following result set:

Description

-------------------------------------

204.......Intro to SQL

130.......Intro to Unix

230.......Intro to Internet

20........Intro to Computers

25........Intro to Programming

120.......Intro to Java Programming

240.......Intro to the Basic Language

7 rows selected.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## **Lab 3.2 3.2.1. Use Number Functions and Perform Mathematical**

## **Computations**.

a) Describe the effect of the negative precision as a parameter of the ROUND function in

the following SQL statement.

SELECT 10.245, ROUND(10.245, 1), ROUND(10.245, -1)

FROM dual

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteb) Write a SELECT statement that displays distinct course costs. In a separate column, show the COST increased by 75% and round the decimals to the nearest dollar.

c) Write a SELECT statement that displays distinct numeric grades from the GRADE table and half those values expressed as a whole number in a separate column.

Imagen que contiene Texto

Descripción generada automáticamente

## **3.3.1. Apply Substitution Functions and Other Miscellaneous Functions**

a) List the last name, first name, and phone number of students who do not have a phone

number. Display '212-555-1212' for the phone number.

Texto

Descripción generada automáticamente

b) For course numbers 430 and greater, show the course cost. Add another columna reflecting a discount of 10% off the cost and substitute any NULL values in the COST column with the number 1000. The result should look similar to the following output.

Texto

Descripción generada automáticamenteCOURSE\_NO COST NEW

-------------- ------- ------

430 1195 1075.5

450 900

2 rows selected.

c) Write the query to accomplish the following output using the NVL2 function in the column 'Get this result'.

ID NAME PHONE Get this result

------------------ ------------ -----------------

112 Thomas Thomas 201-555-5555 Phone# exists.

111 Peggy Noviello No phone# exists.

2 rows selected.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## 3.3.2. Utilize the Power of the DECODE Function and the CASE Expression

Texto

Descripción generada automáticamentea) Rewrite the query from Exercise 3.3.1 c) using the DECODE function instead.

b) For course numbers 20, 120, 122, and 132, display the description, course number, and

prerequisite course number. If the prerequisite is course number 120, display 200; if the

prerequisite is 130, display 'N/A'. For courses with no prerequisites, display 'None'.

Otherwise, list the current prerequisite. The result should look like the one listed below.

COURSE\_NO DESCRIPTION ORIGINAL NEW

--------- ------------------------------ -------- ----

132 Basics of Unix Admin 130 N/A

122 Intermediate Java Programming 120 200

120 Intro to Java Programming 80 80

20 Intro to Computers None

4 rows selected.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

c) Display the student ID, zip code, and phone number for students with student IDs 145, 150, or 325. For those students living in the 212 area code and in zip code 10048, display 'North Campus'. List students living in the 212 area code but in a different zip code as 'West Campus'. Display students outside the 212 area code as 'Off Campus'. The result should look like the following output. Hint: The solution to this query requires nested DECODE functions or nested CASE expressions.

STUDENT\_ID ZIP PHONE LOC

---------- ----- --------------- ------------

145 10048 212-555-5555 North Campus

150 11787 718-555-5555 Off Campus

325 10954 212-555-5555 West Campus

Texto

Descripción generada automáticamente3 rows selected

d) Display all the distinct salutations used in the INSTRUCTOR table. Order them alphabetically except for female salutations, which should be listed first. Hint: Use the DECODE function or CASE expression in the ORDER BY clause.

Texto

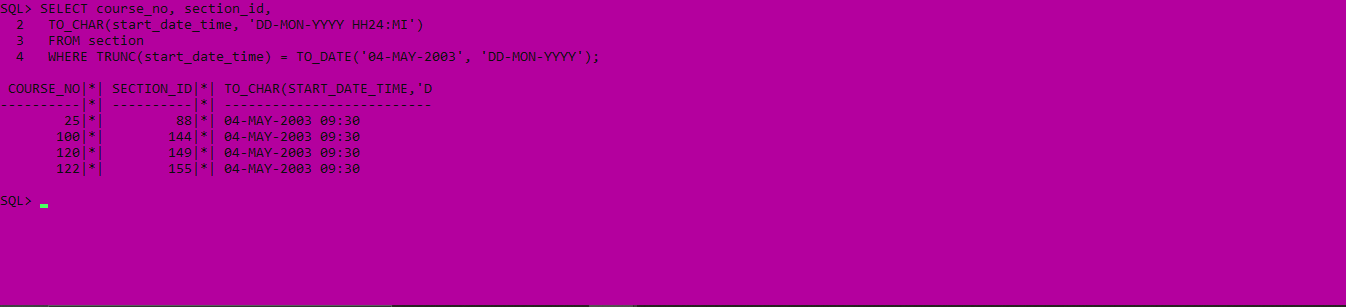
Descripción generada automáticamente

# **Chapter 4. Date and Conversion Functions**

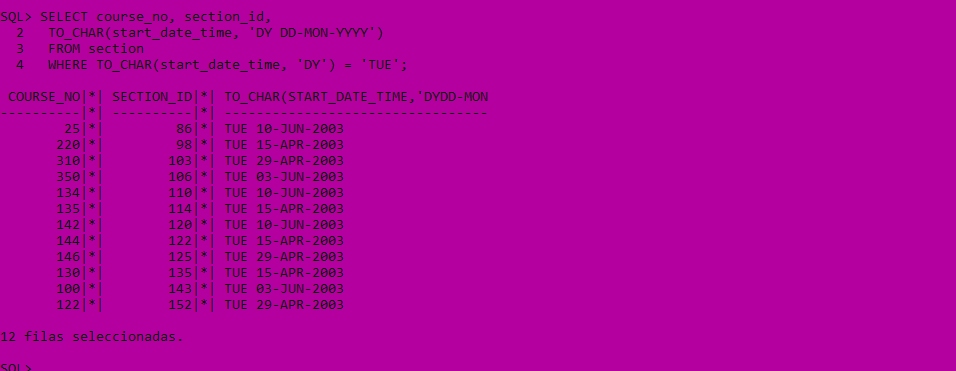
****

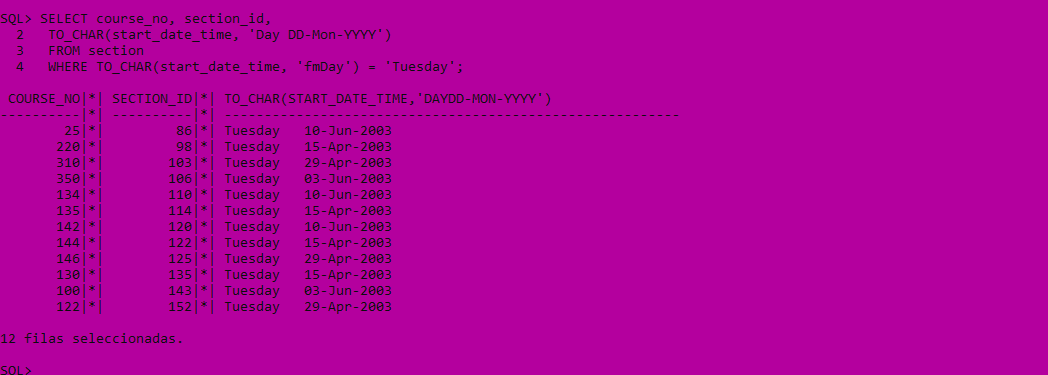
****

****

****

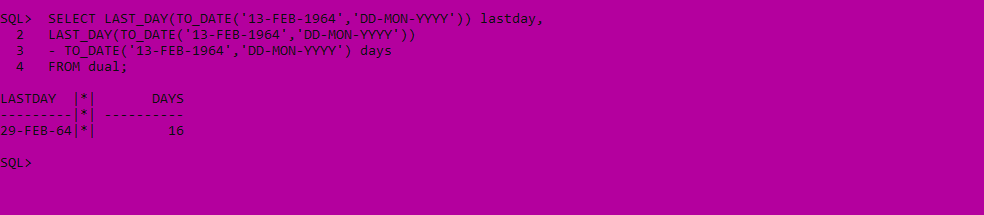
****

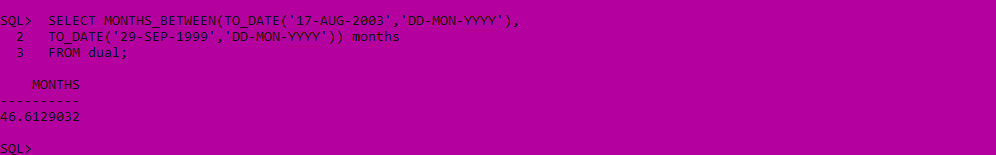
****

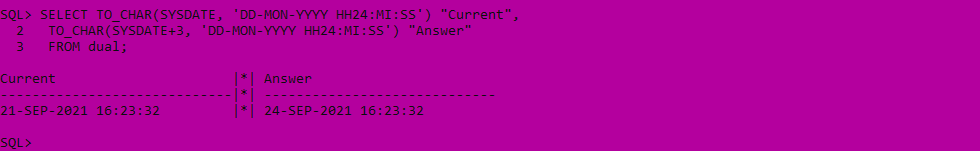
****

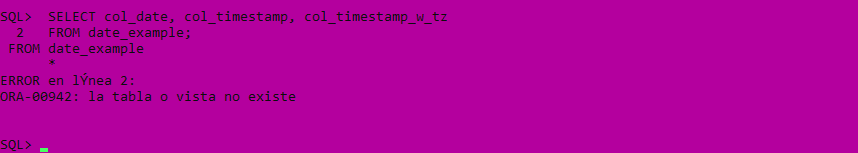
Captura de pantalla de computadora

Descripción generada automáticamente

****

****

****

****

****

# Chapter 5 Equijoins

## Lab 6.1 The Two-Table Join

rem [Rischert, 2004,330]

SELECT c.course\_no, s.section\_no, c.description,

s.location, s.instructor\_id

FROM course c, section s

WHERE c.course\_no = s.course\_no; Imagen que contiene Texto

Descripción generada automáticamente

rem [Rischert, 2004,331]

SELECT course\_no, s.section\_no, c.description,

s.location, s.instructor\_id

FROM course c JOIN section s

USING (course\_no);

Texto

Descripción generada automáticamente

rem [Rischert, 2004,334]

SELECT COUNT(\*)

Captura de pantalla de computadora

Descripción generada automáticamente FROM section, instructor;

rem [Rischert, 2004,334]

SELECT s.instructor\_id s\_instructor\_id,

i.instructor\_id i\_instructor\_id

FROM section s, instructor i;

Captura de pantalla de computadora

Descripción generada automáticamente

## Lab 6.1 Exercises

6.1.1. Write Simple Join Constructs

a) For all students, display last name, city, state, and zip code. Show the result ordered by zip code.

SELECT s.last\_name, s.zip, z.state, z.city

FROM student s, zipcode z

WHERE s.zip = z.zip ORDER BY s.zip

6.1.2. Narrow Down Your Result Set

a) Execute the following SQL statement. Explain your observations about the WHERE clause and the resulting output.

SELECT c.course\_no, c.description, s.section\_no

FROM course c, section s

WHERE c.course\_no = s.course\_no AND c.prerequisite

IS NULL ORDER BY c.course\_no, s.section\_no

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

rem [Rischert, 2004,339]

SELECT c.course\_no, c.description, s.section\_no

FROM course c, section s

WHERE c.course\_no = s.course\_no

AND c.prerequisite IS NULL

ORDER BY c.course\_no, s.section\_no;

Pantalla de computadora

Descripción generada automáticamente con confianza media

rem [Rischert, 2004,340]

SELECT e.student\_id, s.course\_no,

TO\_CHAR(e.enroll\_date,'MM/DD/YYYY HH:MI PM'),

e.section\_id

FROM enrollment e JOIN section s

ON (e.section\_id = s.section\_id)

WHERE s.course\_no = 20

AND e.enroll\_date >= TO\_DATE('01/30/2003','MM/DD/YYYY')

Captura de pantalla de computadora

Descripción generada automáticamente AND e.enroll\_date < TO\_DATE('01/31/2003','MM/DD/YYYY');

rem [Rischert, 2004,341]

SELECT s.student\_id, i.instructor\_id,

s.zip, i.zip

FROM student s, instructor i

WHERE s.zip = i.zip

ORDER BY s.student\_id, i.instructor\_id;

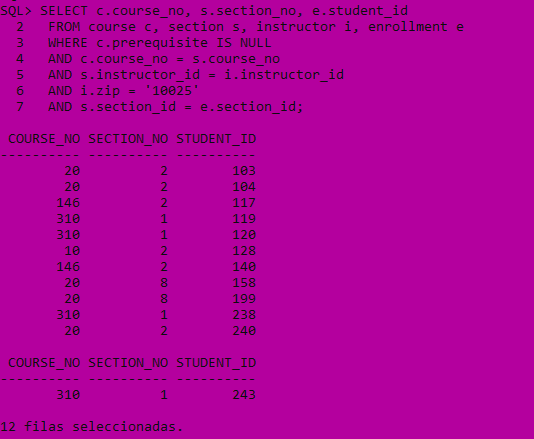
Captura de pantalla de computadora

Descripción generada automáticamente

# Lab 6.2 Exercises

## 6.2.1. Join Three or More Tables

Display the student ID, course number, and section number of enrolled students where the instructor of the section lives in zip code 10025. Additionally, the course should not have any prerequisites.



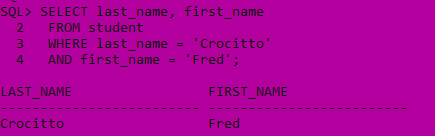
rem [Rischert, 2004,358]

SELECT last\_name, first\_name

FROM student

WHERE last\_name = 'Crocitto'

AND first\_name = 'Fred';



# Chapter 7. Subqueries

## Lab 7.1 Exercises

rem [Rischert, 2004,376]

SELECT first\_name, last\_name

FROM student

WHERE registration\_date =

(SELECT MIN(registration\_date)

FROM student);



rem [Rischert, 2004,376]

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 <=

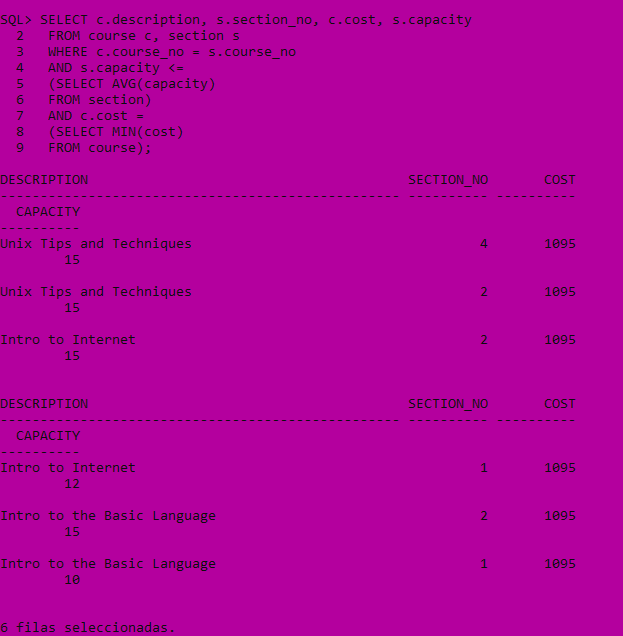
(SELECT AVG(capacity)

FROM section)

AND c.cost =

(SELECT MIN(cost)

FROM course);



rem [Rischert, 2004,376]

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);

rem [Rischert, 2004,381]

SELECT instructor\_id, last\_name, first\_name, zip

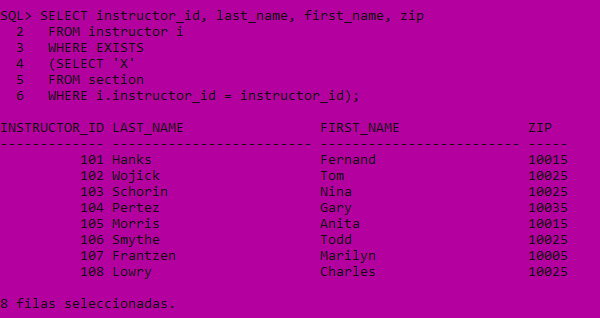
FROM instructor i

WHERE EXISTS

(SELECT 'X'

FROM section

WHERE i.instructor\_id = instructor\_id);

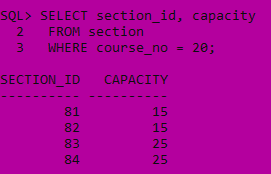


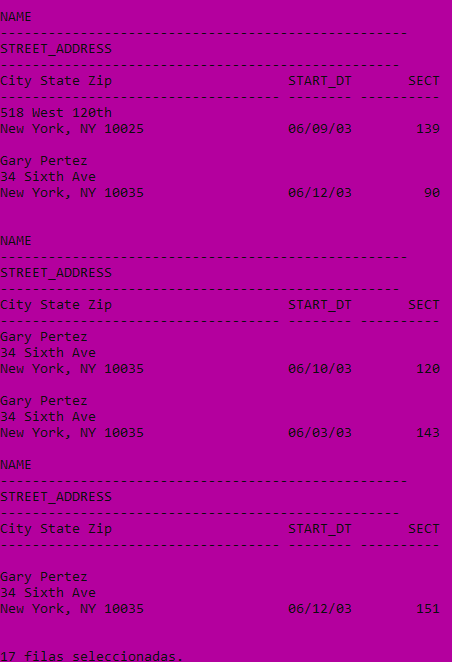
rem [Rischert, 2004,388]

SELECT section\_id, capacity

FROM section

WHERE course\_no = 20;





rem [Rischert, 2004,377]

SELECT course\_no, SUM(capacity)

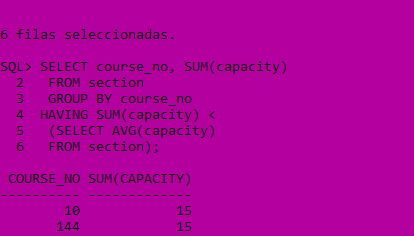
FROM section

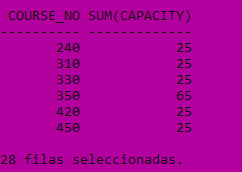
GROUP BY course\_no

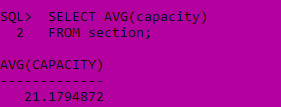
HAVING SUM(capacity) <

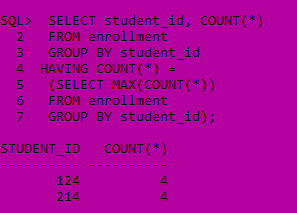
(SELECT AVG(capacity)

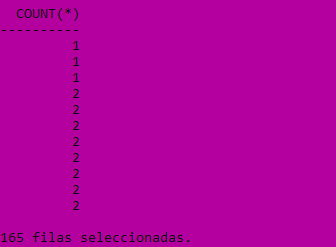
FROM section);

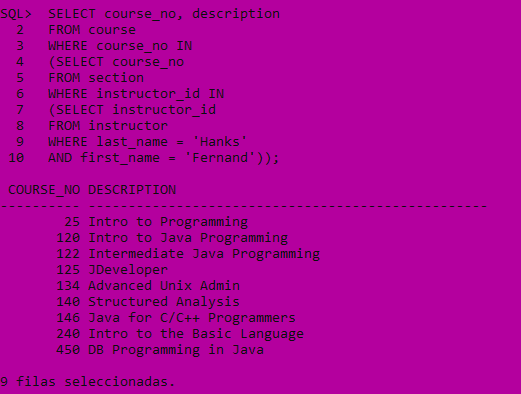


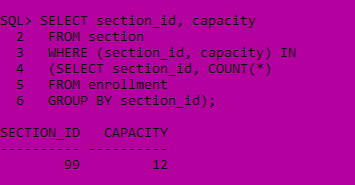












rem [Rischert, 2004,397]

SELECT course\_no, section\_id

FROM section s

WHERE NOT EXISTS

(SELECT NULL

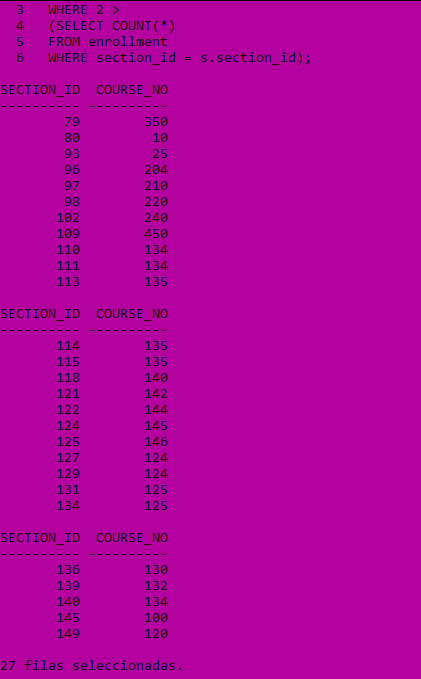
FROM enrollment

WHERE s.section\_id = section\_id)

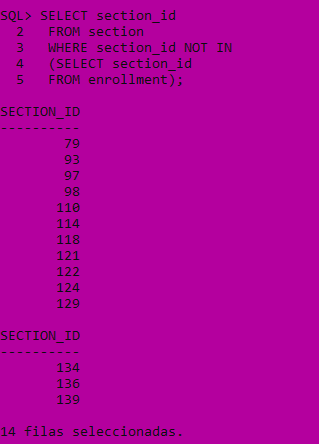
ORDER BY course\_no;

## 7.2 exercises

Explain what the following correlated subquery accomplishes.



Then write a second query to show the sections without any enrollments (i.e., the SECTION\_ID does not exist in the ENROLLMENT table). To determine these sections, you can use the NOT IN operator because the SECTION\_ID in the ENROLLMENT table is defined as NOT NULL.



rem [Rischert, 2004,393]

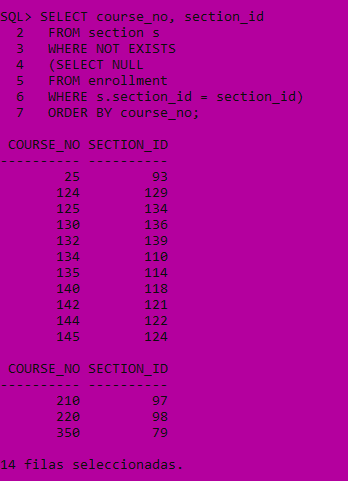
SELECT section\_id

FROM section

WHERE section\_id NOT IN

(SELECT section\_id

FROM enrollment);



## Lab 7.4 Exercise

rem [Rischert, 2004,417]

SELECT student\_id, section\_id, numeric\_grade

FROM grade g

WHERE grade\_type\_code = 'FI'

AND numeric\_grade > ALL

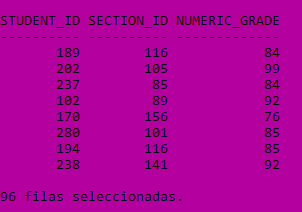
(SELECT numeric\_grade

FROM grade

WHERE grade\_type\_code = 'HM'

AND g.section\_id = section\_id

AND g.student\_id = student\_id);



rem [Rischert, 2004,418]

SELECT student\_id, section\_id, grade\_type\_code,

MAX(numeric\_grade) max, MIN(numeric\_grade) min

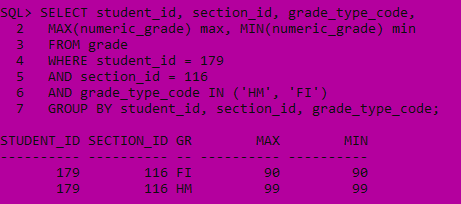
FROM grade

WHERE student\_id = 179

AND section\_id = 116

AND grade\_type\_code IN ('HM', 'FI')

GROUP BY student\_id, section\_id, grade\_type\_code;



rem [Rischert, 2004,420]

SELECT student\_id, section\_id, grade\_type\_code,

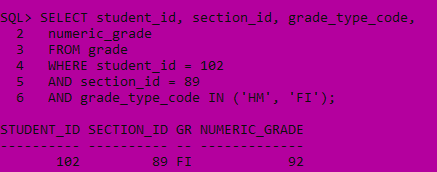
numeric\_grade

FROM grade

WHERE student\_id = 102

AND section\_id = 89

AND grade\_type\_code IN ('HM', 'FI')



# Chapter 8. Set Operators

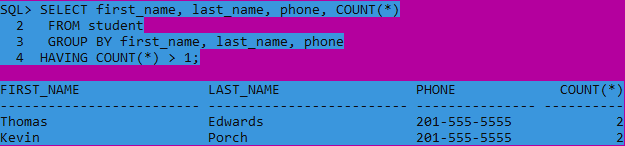
rem [Rischert, 2004,425]

SELECT first\_name, last\_name, phone, COUNT(\*)

FROM student

GROUP BY first\_name, last\_name, phone

HAVING COUNT(\*) > 1;



rem [Rischert, 2004,426]

SELECT instructor\_id id, first\_name, last\_name, phone

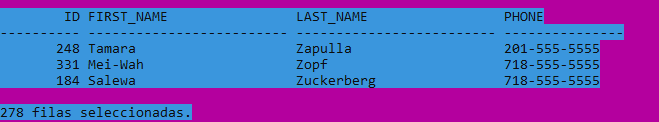
FROM instructor

UNION

SELECT student\_id, first\_name, last\_name, phone

FROM student

ORDER BY 3;



## Lab 8.1 Exercises

rem [Rischert, 2004,427]

SELECT first\_name, last\_name,

'Instructor' "Type"

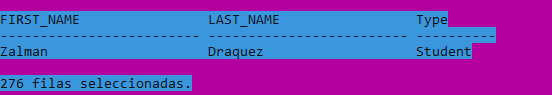
FROM instructor

UNION

SELECT first\_name, last\_name,

'Student'

FROM student;



rem [Rischert, 2004,429]

SELECT first\_name, last\_name,

'Instructor' "Type"

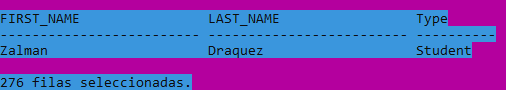
FROM instructor

UNION

SELECT first\_name, last\_name,

'Student'

FROM student;



rem [Rischert, 2004,430]

SELECT created\_by

FROM enrollment

UNION

SELECT created\_by

FROM grade

UNION

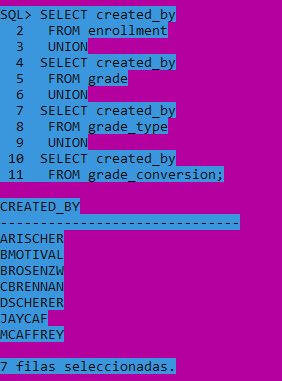
SELECT created\_by

FROM grade\_type

UNION

SELECT created\_by

FROM grade\_conversion;



rem [Rischert, 2004,432]

SELECT DISTINCT salutation, CAST(NULL AS NUMBER),

state, z.created\_date

FROM instructor i, zipcode z

WHERE i.zip = z.zip

UNION ALL

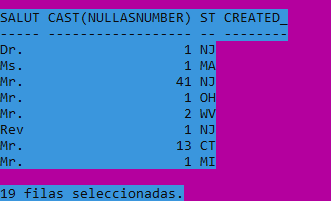
SELECT salutation, COUNT(\*),

state, TO\_DATE(NULL)

FROM student s, zipcode z

WHERE s.zip = z.zip

GROUP BY salutation, state;



## Lab 8.2 Exercises

rem [Rischert, 2004,440]

SELECT course\_no, description

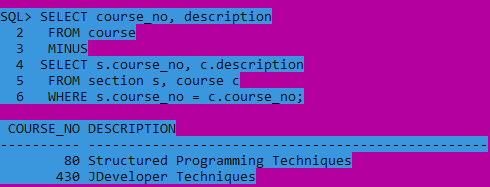
FROM course

MINUS

SELECT s.course\_no, c.description

FROM section s, course c

WHERE s.course\_no = c.course\_no;



rem [Rischert, 2004,441]

SELECT course\_no, description

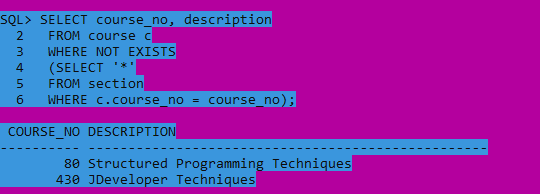
FROM course c

WHERE NOT EXISTS

(SELECT '\*'

FROM section

WHERE c.course\_no = course\_no);



rem [Rischert, 2004,442]

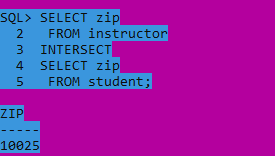
SELECT zip

FROM instructor

INTERSECT

SELECT zip

FROM student;



# Chapter 9. Complex Joins

rem [Rischert, 2004,447]

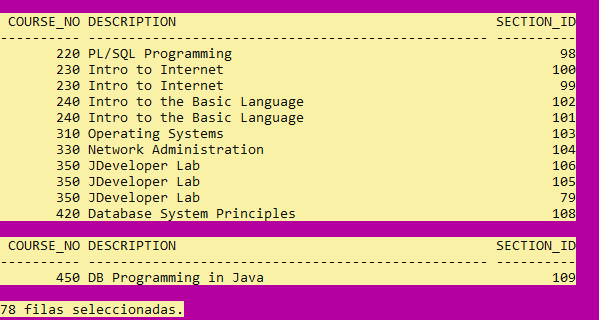
SELECT course\_no, description,

section\_id

FROM course JOIN section

USING (course\_no)

ORDER BY course\_no;



rem [Rischert, 2004,448]

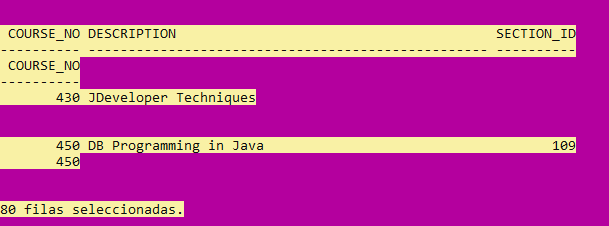
SELECT c.course\_no, c.description,

s.section\_id, s.course\_no

FROM course c LEFT OUTER JOIN section s

ON c.course\_no = s.course\_no

ORDER BY c.course\_no;



rem [Rischert, 2004,452]

SELECT col1, col2

FROM t1, t2

WHERE t1.col1 = t2.col2(+)

UNION

SELECT col1, col2

FROM t1, t2

WHERE t1.col1(+) = t2.col2;

# Lab 9.1 Exercises

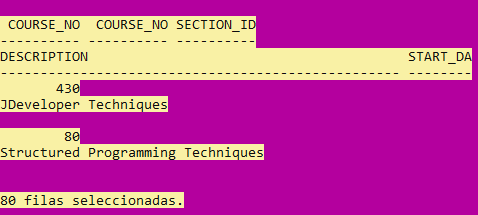
rem [Rischert, 2004,455]

SELECT c.course\_no, s.course\_no, s.section\_id,

c.description, s.start\_date\_time

FROM course c FULL OUTER JOIN section s

ON c.course\_no = s.course\_no;

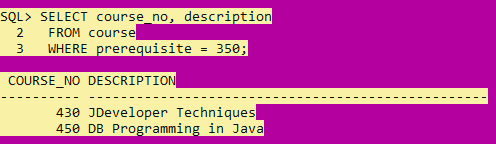


rem [Rischert, 2004,456]

SELECT course\_no, description

FROM course

WHERE prerequisite = 350;



rem [Rischert, 2004,457]

SELECT course\_no cno,

description,

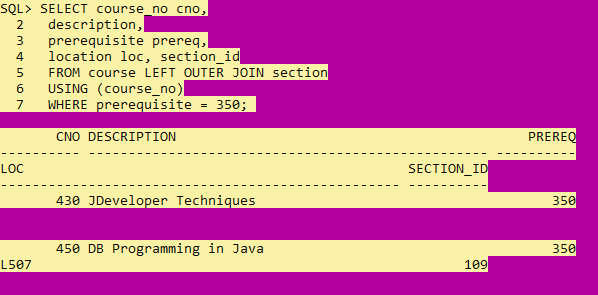
prerequisite prereq,

location loc, section\_id

FROM course LEFT OUTER JOIN section

USING (course\_no)

WHERE prerequisite = 350;



rem [Rischert, 2004,460]

SELECT c.course\_no cno, s.course\_no sno,

c.description,

c.prerequisite prereq,

s.location loc, s.section\_id

FROM (SELECT \*

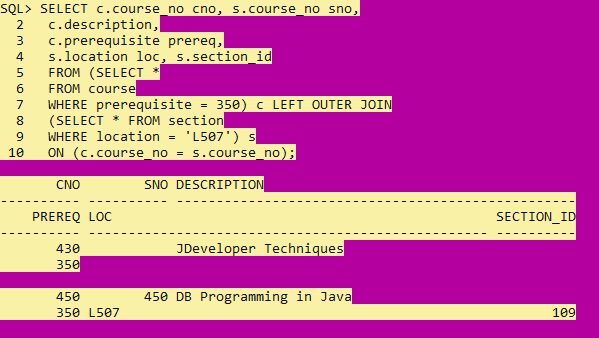
FROM course

WHERE prerequisite = 350) c LEFT OUTER JOIN

(SELECT \* FROM section

WHERE location = 'L507') s

ON (c.course\_no = s.course\_no);



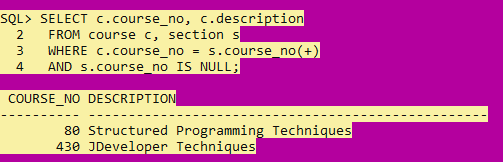
rem [Rischert, 2004,461]

SELECT c.course\_no, c.description

FROM course c, section s

WHERE c.course\_no = s.course\_no(+)

AND s.course\_no IS NULL;



rem [Rischert, 2004,462]

SELECT city, state, zip,

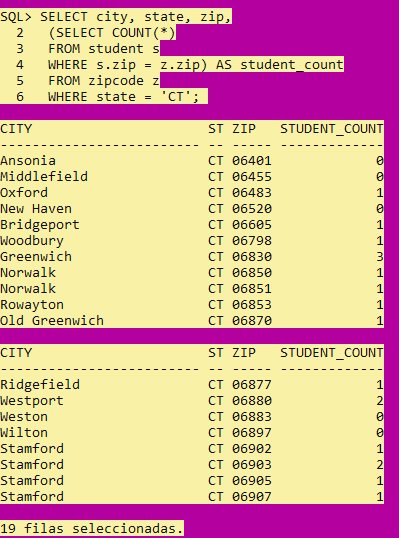
(SELECT COUNT(\*)

FROM student s

WHERE s.zip = z.zip) AS student\_count

FROM zipcode z

WHERE state = 'CT';



rem [Rischert, 2004,471]

SELECT student\_id, section\_id, numeric\_grade

FROM grade g

WHERE grade\_type\_code = 'FI'

AND section\_id = 86

AND numeric\_grade < ANY

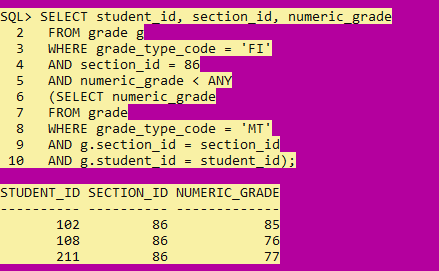
(SELECT numeric\_grade

FROM grade

WHERE grade\_type\_code = 'MT'

AND g.section\_id = section\_id

AND g.student\_id = student\_id);



rem [Rischert, 2004,472]

SELECT DISTINCT a.student\_id, a.last\_name,

a.street\_address

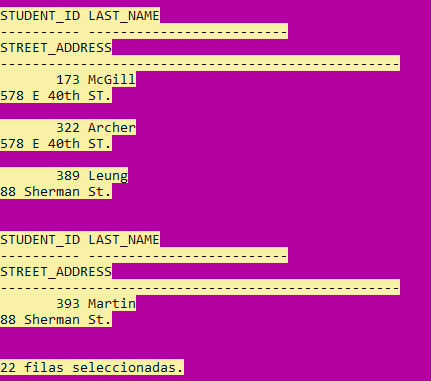
FROM student a, student b

WHERE a.street\_address = b.street\_address

AND a.zip = b.zip

AND a.student\_id <> b.student\_id

ORDER BY a.street\_address;



rem [Rischert, 2004,474]

SELECT c1.course\_no,

SUBSTR(c1.description, 1,15) course\_descr,

C1.prerequisite,

SUBSTR(c2.description,1,15) pre\_req\_descr

FROM course c1, course c2

WHERE c1.prerequisite = c2.course\_no(+)

ORDER BY 1;

