



**Autonomous University of Zacatecas**

ACADEMIC UNIT OF ELECTRICAL ENGINEERING

Software Engineering Academic Program

*Group: 5B - Semester: 2022-5<sup>o</sup>*

**Practice Number: 13**

**Using Subqueries**

DATE: 08/NOVEMBER/2022

**Professor:**

Aldonso Becerra Sánchez.

**Student:**

Cristian Omar Alvarado Rodríguez.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Practice objective</b>	<b>4</b>
<b>3</b>	<b>Developing</b>	<b>5</b>
<b>4</b>	<b>Pre-assessment</b>	<b>21</b>
<b>5</b>	<b>Conclusion</b>	<b>21</b>

# Using Subqueries

November 8th, 2022

## 1 Introduction

The use of subqueries is a technique that allows the result of a SELECT table to be used in another SELECT query. It allows solving complex queries by using previous results obtained through another query.

The SELECT that is placed inside another SELECT is known as the SUBSELECT. That SUBSELECT can be placed inside the WHERE, HAVING, FROM, or JOIN clauses.

### **Simple subqueries:**

Simple subqueries are those that return a single row. If they also return a single column, they are called scalar subqueries, since they return a single value.

A subquery that uses the values  $>$ ,  $<$ ,  $>=$ , ... has to return a single value, otherwise an error occurs. They also have to return the same type and number of data to relate the subquery to the query that uses it (it cannot happen that the subquery has two columns and that result is compared using a single column in the general query).

### **multi-row subqueries:**

In the previous section it was mentioned that subqueries can only return one row. But sometimes queries of the type are needed: show the salary and name of employees whose salary exceeds that of

any employee in the sales department.

The subquery required for that result would return all salaries in the sales department. But we won't be able to use a comparison operator directly since that subquery returns more than one row. The solution to this is to use special instructions between the operator and the query, which allow the use of multi-row subqueries.

Those instructions are:

**ANY:** Compares with any record in the subquery. The statement is valid if there is a record in the subquery that allows the comparison to be true.

**ALL:** Compares with all the records of the query. The statement is true if all comparisons with the records of the subquery are true.

**IN:** It does not use a comparator, since it is used to check if a value is found in the result of the subquery.

**NOT IN:** Checks if a value is not found in a subquery.

### **Logical operator EXISTS:**

It is used when the selection condition consists exclusively of checking that the subquery returns some selected row based on the condition included in the subquery itself. The EXISTS operator does not require the subquery to return any columns because it does not use no comparison expression, thus justifying the acceptance of the \* in its format.

## **2 Practice objective**

Use SQL SELECT statements for retrieving data from several sources using different operations

### 3 Developing

**Activity 1:** Read all the choices carefully because there might be more than one correct answer. Choose all the correct answers for each question.

**Explain the reason for your answer.**

**DEFINE SUBQUERIES.**

1. Consider this generic description of a **SELECT** statement:

**SELECT** select\_list **FROM** table **WHERE** condition **GROUP BY** expression\_1 **HAVING** expression\_2 **ORDER BY** expression\_3;

Where could subqueries be used? (Choose all correct answers.))

- A) select\_list.
- B) expression\_2.
- C) condition.
- D) expression\_1.
- E) table.
- F) expression\_3.

**Explanation:** A, B, C, D, E. Subqueries can be used in all of these options. Option F is wrong since a subquery cannot be used in the **ORDER BY** clause of a query.

2. A query can have a subquery embedded within it. Under what circumstances could there be more than one subquery? (Choose the best answer.)

- A) Subqueries can be embedded within each other with no practical limitations on depth.
- B) It is possible to embed a single-row subquery inside a multiple-row subquery, but not the other way around.
- C) The outer query can have multiple inner queries, but they must not be embedded within each

other.

D) The outer query can include an inner query. It is not possible to have another query within the inner query.

**Explanation: A.** Nesting of subqueries can be done at various levels.

**3. Consider this statement:**

```
select employee_id, last_name from employees where salary > (select avg(salary) from employees);
```

**When will the subquery be executed? (Choose the best answer.)**

A) It will be executed once for every row in the EMPLOYEES table.

B) It will be executed after the outer query.

C) It will be executed concurrently with the outer query.

D) It will be executed before the outer query.

**Explanation: D.** The result set of the inner query is needed before the outer query can be executed.

**4. Consider this statement:**

```
select o.employee_id, o.last_name from employees o where o.salary > (select avg(i.salary) from employees i where i.department_id = o.department_id);
```

**When will the subquery be executed? (Choose the best answer.)**

A) It will be executed once for every row in the EMPLOYEES table.

B) It will be executed after the outer query.

C) It will be executed concurrently with the outer query.

D) It will be executed before the outer query.

**Explanation: A.** This is a correlated subquery that must be executed for each of the rows in the table.

**DESCRIBE THE TYPES OF PROBLEMS THAT THE SUBQUERIES CAN SOLVE.**

5. Consider the following statement:

```
select last_name from employees join departments on employees.department_id = departments.department_id where department_name = 'Executive';
```

and this statement:

```
select last_name from employees where department_id in (select department_id from departments where department_name = 'Executive');
```

What can be said about the two statements? (Choose two correct answers.)

- A) Both statements will always run successfully, even if there are two departments with DEPARTMENT\_NAME 'Executive.'
- B) The two statements could generate different results.
- C) The first statement will always run successfully; the second statement will error if there are two departments with DEPARTMENT\_NAME 'Executive.'
- D) The two statements should generate the same result.

**Explanation: A and D.** The statements return the same result and neither will indicate an error if the name is duplicated.

**LIST THE TYPES OF SUBQUERIES.**

6. What are the distinguishing characteristics of a scalar subquery? (Choose two correct answers.)

- A) A scalar subquery returns one row.
- B) A scalar subquery cannot be used as a correlated subquery.
- C) A scalar subquery cannot be used in the SELECT LIST of the parent query.
- D) A scalar subquery returns one column.

**Explanation: A and D.** A Single-row subquery is defined as a query that returns a single value.

7. Which comparison operator can be used with multiple-row subqueries? (Choose the best answer.)

- A) ALL
- B) ANY
- C) IN
- D) NOT IN
- E) All the above can be used.

**Explanation: E.** ALL, ANY, IN, and NOT IN are the multiple-row comparison operators.

#### WRITE SINGLE-ROW AND MULTIPLE-ROW SUBQUERIES.

8. Consider this statement:

```
select last_name, (select count(*) from departments) from employees where salary =  
(select salary from employees);
```

What is wrong with it? (Choose the best answer.)

- A) NoThe statement will run but is extremely inefficient because of the need to run the second subquery once for every row in EMPLOYEES.
- B) The statement will fail if the second query returns more than one row.
- C) The statement will fail because the subquery in the SELECT list references a table that is not listed in the FROM clause.
- D) Nothing is wrong-the statement should run without error.

**Explanation: B.** The equals operator requires a single row subquery and the second subquery could return multiple rows so this will raise an error.



9. Which of the following statements are equivalent? (Choose two answers.)

A) `select employee_id from employees where salary < all (select salary from employees where department_id=10);`

B) `select employee_id from employees e join departments d on e.department_id = d.department_id where e.salary < (select min(salary) from employees) and d.department_id = 10;`

C) `select employee_id from employees where salary not >= any (select salary from employees where department_id = 10);`

D) `select employee_id from employees where salary < (select min(salary) from employees where department_id = 10);`

**Explanation: A and D.** These two queries are the same as they return the same result.

10. Consider this statement, which is intended to prompt for an employee's name and then find all employees who have the same job as the first employee:

```
select last_name, employee_id from employees where job_id = (select job_id from employees where last_name = '&Name');
```

What would happen if a value were given for &Name that did not match with any row in EMPLOYEES? (Choose the best answer.)

A) The statement would return every row in the table.

B) The statement would fail with an error.

C) The statement would return all rows where JOB\_ID is NULL.

D) The statement would return no rows.

**Explanation: D.** If a subquery returns NULL the comparison will also return NULL meaning no rows will be retrieved.

**Activity 2:** Propose an answer to the following issues:

- How can you best design subqueries such that they will not fail with “ORA01427: single-row subquery returns more than one row” errors?

**R =** You will have to use multiple-row comparison operators (ALL, ANY, IN, etc.)

• **Star Transformation.** An extension of the use of subqueries as an alternative to a join is to enable the star transformation often needed in data warehouse applications. Consider a large table recording sales. Each sale is marked as being of a particular product to a particular buyer through a particular channel. These attributes are identified by codes, used as foreign keys to dimension tables with rows that describe each product, buyer, and channel. To identify all sales of books to buyers in Germany through Internet orders, one could run a query like this:

```
select ... from sales s, products p, buyers b, channels c where s.prod_code = p.prod_code and  
s.buy_code=b.buy_code and s.chan_code=c.chan_code and p.product = 'Books' and b.country =  
'Germany' and c.channel = 'Internet';
```

This query uses the WHERE clause to join the tables and then to filter the results. The following is an alternative query that will yield the same result:

```
select ... from sales where prod_code in (select prod_code from products where product = 'Books')  
and buy_code in (select buy_code from buyers where country = 'Germany') and chan_code in (select  
chan_code from channels where channel = 'Internet');
```

The rewrite of the first statement to the second is the star transformation. Apart from being an inherently more elegant structure (most SQL developers with any sense of aesthetics will agree with that), there are technical reasons why the database may be able to execute it more efficiently than the original query. Also, star queries are easier to maintain; it is very simple to add more dimensions to the query or to replace the single literals ('Books,' 'Germany,' and 'Internet') with lists of values.

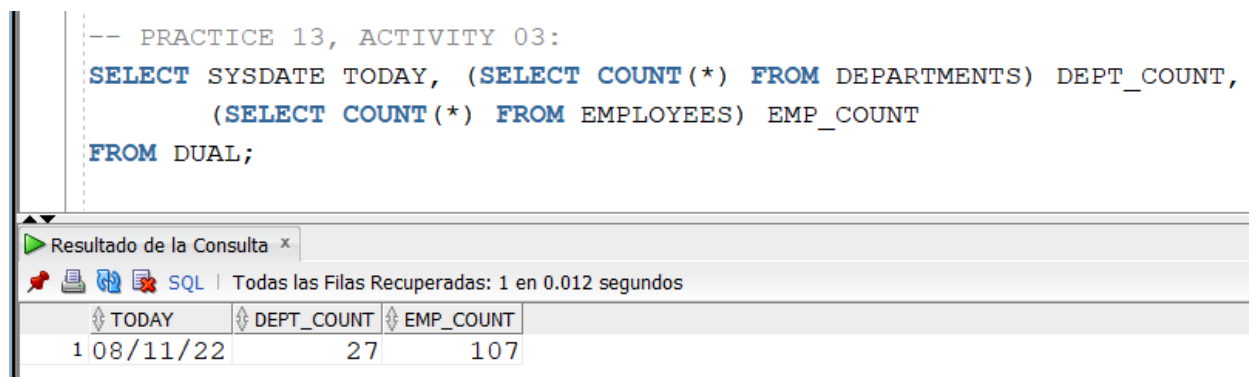
**NOTE (on the JOB):** There is an instance initialization parameter, `STAR_TRANSFORMATION_ENABLED`, which (if set to true) will permit the Oracle query optimizer to re-write code into star queries.

Sometimes there is a choice between using a subquery or using some other technique: the star transformation is a case in point. Which is better?

**R =** Depends on the circumstances. It is not uncommon for different methods to trigger different execution methods on the database. Depending on how your instance is configured, the database, and the data structures within it, one instance can be much more efficient than the other. Whenever such a choice is available, the operator should be subject to a fine-tuning analysis.

**Activity 3:** This exercise must be performed using HR schema.

a) Write a query that uses subqueries in the column projection list. The query will report on the current (date of today) numbers of departments and staff, see figure 1.



```
-- PRACTICE 13, ACTIVITY 03:
SELECT SYSDATE TODAY, (SELECT COUNT(*) FROM DEPARTMENTS) DEPT_COUNT,
      (SELECT COUNT(*) FROM EMPLOYEES) EMP_COUNT
FROM DUAL;
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0.012 segundos

	TODAY	DEPT_COUNT	EMP_COUNT
1	08/11/22	27	107

Figure 1: *Using subqueries: HR Schema.*

b) Write a query to identify all the employees who are managers. This will require using a subquery in the WHERE clause to select all the employees whose `EMPLOYEE_ID` appears as a `MANAGER_ID`, see figure 2.

```
SELECT LAST_NAME FROM EMPLOYEES
WHERE EMPLOYEE_ID IN (SELECT MANAGER_ID FROM EMPLOYEES);
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 18 en 0.022 segundos

	LAST_NAME
1	Cambrault
2	De Haan
3	Errazuriz
4	Fripp
5	Greenberg
6	Hartstein
7	Higgins
8	Hunold
9	Kaufling
10	King
11	Kochhar

Figure 2: *Using subqueries: HR Schema.*

C) Write a query to identify the highest salary paid in each country. This will require using a subquery in the FROM clause, see figure 3.

```
SELECT MAX(SALARY), COUNTRY_ID FROM (SELECT SALARY, DEPARTMENT_ID
LOCATION_ID, COUNTRY_ID FROM EMPLOYEES NATURAL JOIN DEPARTMENTS
NATURAL JOIN LOCATIONS)
GROUP BY COUNTRY_ID;
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 3 en 0.011 segundos

	MAX(SALARY)	COUNTRY_ID
1	17000	US
2	6000	CA
3	10000	UK

Figure 3: *Using subqueries: HR Schema.*

d) Write a query that will identify all employees who work in departments located in the United Kingdom. This will require three levels of nested subqueries in the WHERE clause, see figure 4.

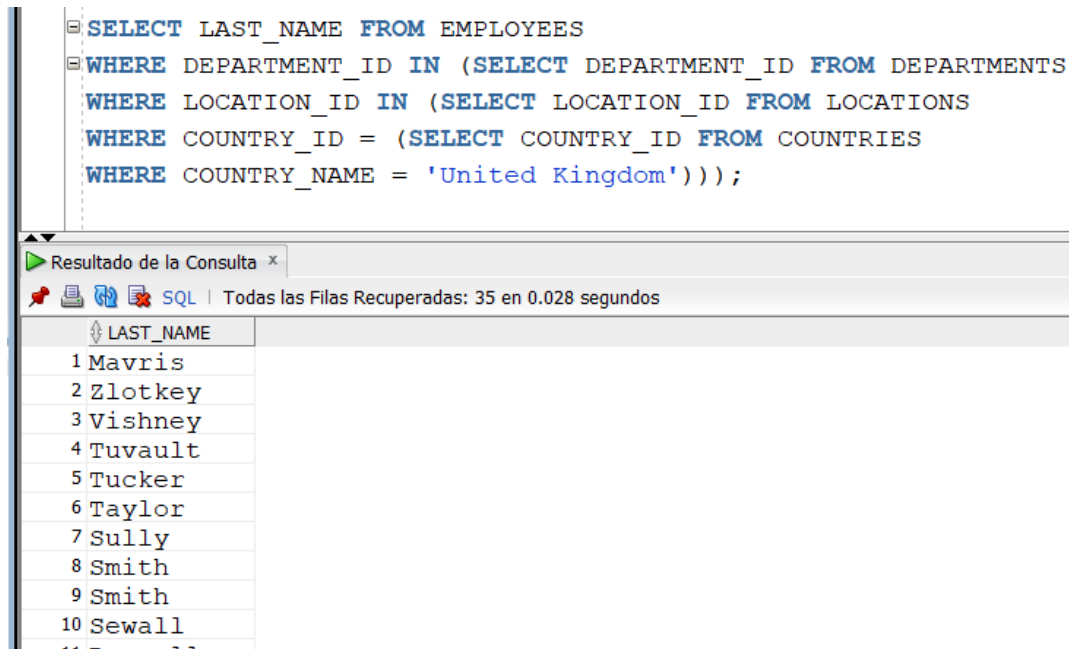


Figure 4: *Using subqueries: HR Schema.*

e) Write a query to identify all the employees who earn more than the average and who work in any of the IT departments. This will require two subqueries in the WHERE clause, not nested, see figure 5.

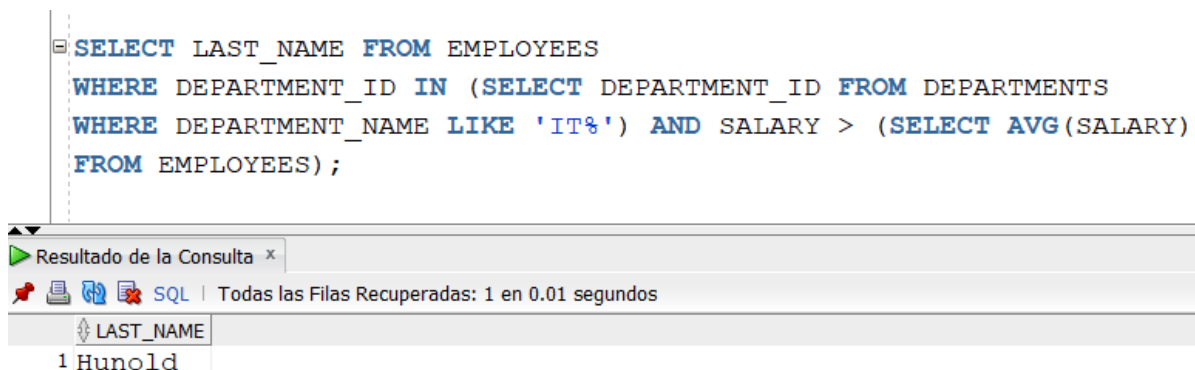


Figure 5: *Using subqueries: HR Schema.*

f) Write a query to determine who earns more than Mr. Tobias, see figure 6.

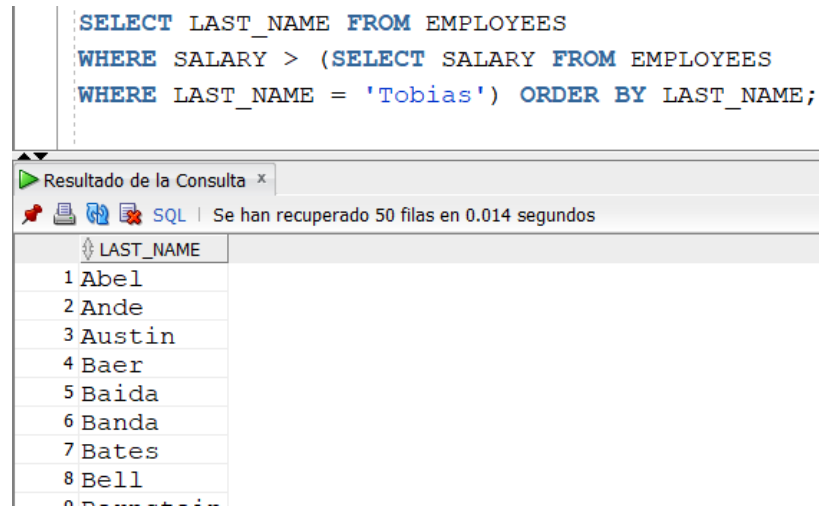


Figure 6: Using subqueries: HR Schema.

Write a query to determine who earns more than Mr. Taylor. Write the sentence to be useful no matter the number of rows returned by the subquery in the WHERE clause (use > operator). There can be several solutions (show two):

**First solution:**

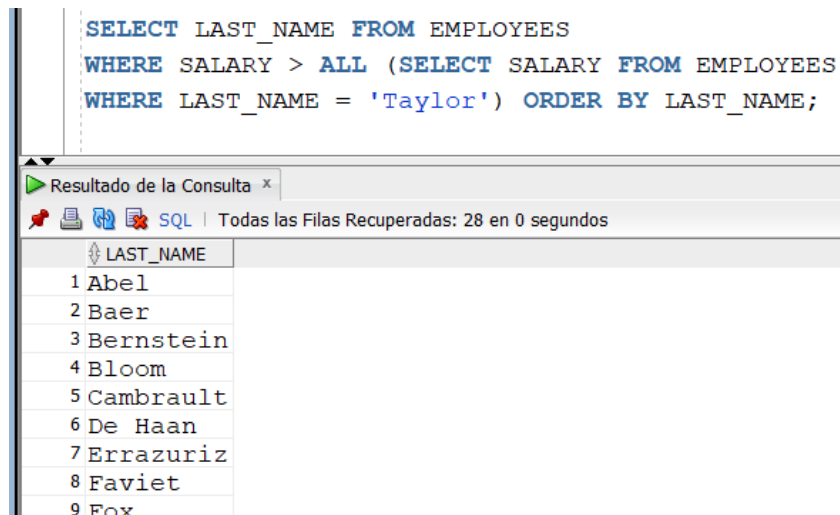


Figure 7: Using subqueries: HR Schema.

The second solution:

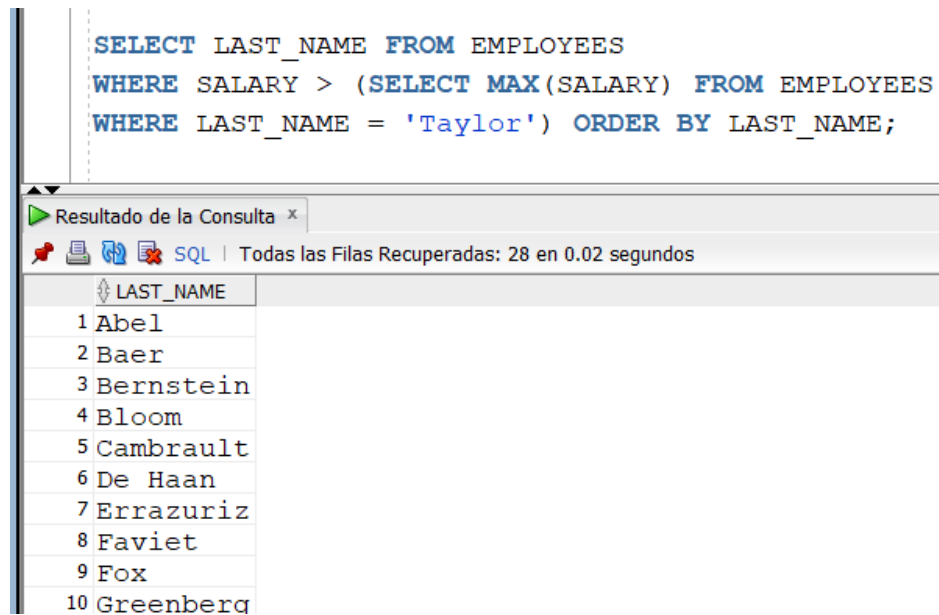
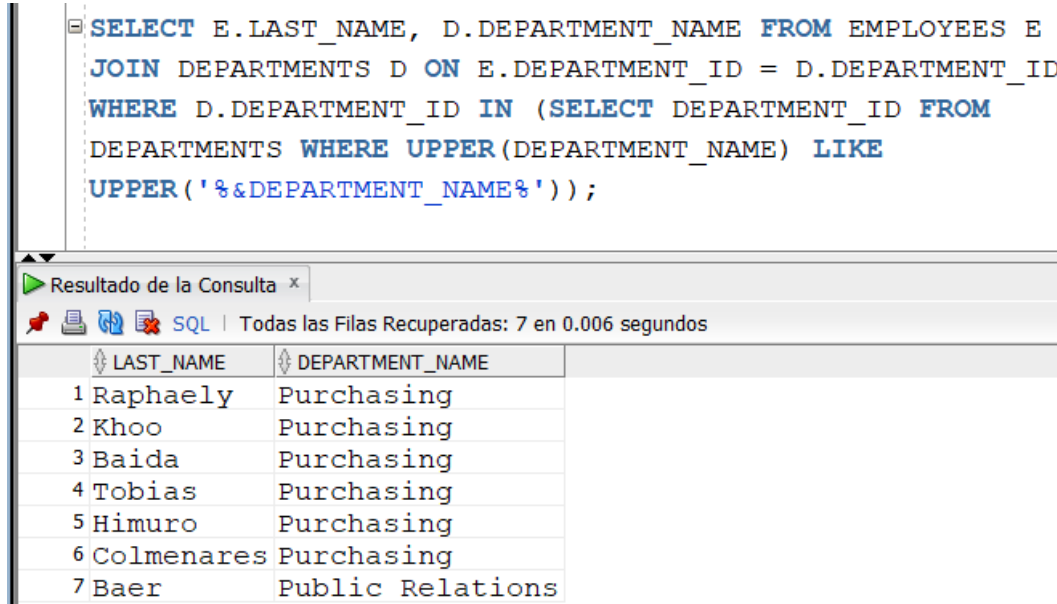


Figure 8: *Using subqueries: HR Schema.*

g) Later exercise included a query that attempted to find all employees whose salary is higher than that of a nominated employee. There are other queries that will run successfully; construct two other solutions, one using the ANY comparison operator, the other using the MIN aggregation function. Now that you have several solutions, do they all give the same result? All these “solutions” are in fact just ways of avoiding error. They do not necessarily give the result the user wants, and they may not be consistent. What change needs to be made to give a consistent, unambiguous, result?

**R =** Make good use of operators for subqueries, whether they are single-row operators or multiple-row operators, this will depend on the result you want to obtain in your subquery.

h) Design a query that will prompt for a department name (no matter if the input is lower or upper case) and list the last name of every employee in that department, use a subquery in the WHERE clause. For instance, if the input is a department name with the string “Pu...”, see **figure 9**.



```
SELECT E.LAST_NAME, D.DEPARTMENT_NAME FROM EMPLOYEES E
JOIN DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE D.DEPARTMENT_ID IN (SELECT DEPARTMENT_ID FROM
DEPARTMENTS WHERE UPPER(DEPARTMENT_NAME) LIKE
UPPER('%&DEPARTMENT_NAME%')) ;
```

Resultado de la Consulta x

Todas las Filas Recuperadas: 7 en 0.006 segundos

	LAST_NAME	DEPARTMENT_NAME
1	Raphaely	Purchasing
2	Khoo	Purchasing
3	Baida	Purchasing
4	Tobias	Purchasing
5	Himuro	Purchasing
6	Colmenares	Purchasing
7	Baer	Public Relations

Figure 9: *Using subqueries: HR Schema.*

**Activity 4:** You will write complex queries using nested SELECT statements.

For practice questions, you may want to create the inner query first. Make sure that it runs and produces the data that you anticipate before you code the outer query.

1. The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey), see **figure 10**.



```

SELECT LAST_NAME, HIRE_DATE
FROM EMPLOYEES WHERE DEPARTMENT_ID = (SELECT
DEPARTMENT_ID FROM EMPLOYEES
WHERE UPPER(LAST_NAME) = UPPER('&EMPLOYEE_NAME'))
AND LAST_NAME <> '&EMPLOYEE_NAME';

```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 33 en 0.014 segundos

	LAST_NAME	HIRE_DATE
1	Russell	01/10/04
2	Partners	05/01/05
3	Errazuriz	10/03/05
4	Cambrault	15/10/07
5	Tucker	30/01/05
6	Bernstein	24/03/05
7	Hall	20/08/05
8	Olsen	30/03/06
9	Cambrault	09/12/06
10	Tuvault	23/11/07
11	King	30/01/04
12	Sully	04/03/04

Figure 10: Using subqueries: HR Schema.

2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary, see figure 11.

```

SELECT EMPLOYEE_ID, LAST_NAME, SALARY FROM EMPLOYEES
WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES)
ORDER BY SALARY;

```

Resultado de la Consulta x

SQL | Se han recuperado 50 filas en 0.019 segundos

	EMPLOYEE_ID	LAST_NAME	SALARY
1	203	Mavris	6500
2	123	Vollman	6500
3	165	Lee	6800
4	113	Popp	6900
5	155	Tuvault	7000
6	161	Sewall	7000
7	178	Grant	7000
8	164	Marvins	7200

Figure 11: Using subqueries: HR Schema.

3. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains the letter “u.” Save your SQL statement as lab\_13\_03.sql. Run your query, see figure 12.

```
SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID
FROM EMPLOYEES WHERE LAST_NAME LIKE '%u%');
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 90 en 0.017 segundos

	EMPLOYEE_ID	LAST_NAME
1	107	Lorentz
2	106	Pataballa
3	105	Austin
4	104	Ernst
5	103	Hunold
6	199	Grant
7	198	OConnell
8	197	Feeney
9	100	De Haan

Figure 12: Using subqueries: HR Schema.

4. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700, see figure 13.

```
SELECT LAST_NAME, DEPARTMENT_ID, JOB_ID FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID FROM DEPARTMENTS
WHERE LOCATION_ID = 1700);
```

Resultado de la Co... x

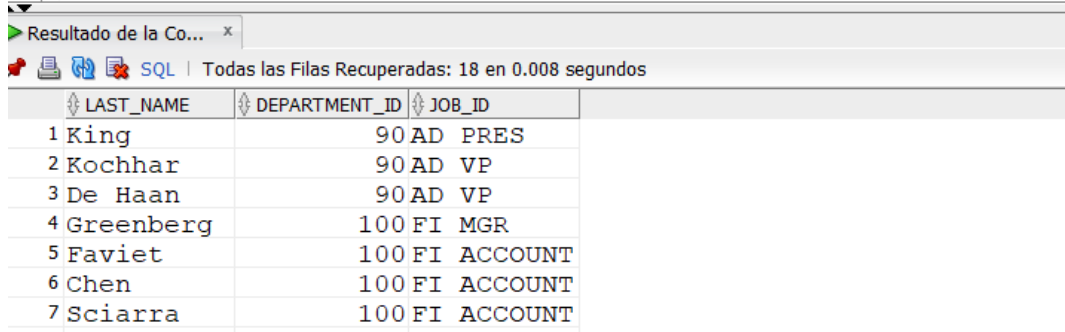
SQL | Todas las Filas Recuperadas: 18 en 0.244 segundos

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	King	90 AD	PRES
2	Kochhar	90 AD	VP
3	De Haan	90 AD	VP
4	Greenberg	100 FI	MGR
5	Faviet	100 FI	ACCOUNT
6	Chen	100 FI	ACCOUNT
7	Sciarra	100 FI	ACCOUNT
8	Ullrich	100 FI	ACCOUNT

Figure 13: Using subqueries: HR Schema.

Modify the query so that the user is prompted for a location ID. Save this to a file named lab\_13\_04.sql, see figure 14.

```
SELECT LAST_NAME, DEPARTMENT_ID, JOB_ID FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID FROM DEPARTMENTS
WHERE LOCATION_ID = &LOCATION_ID);
```



Resultado de la Co... x

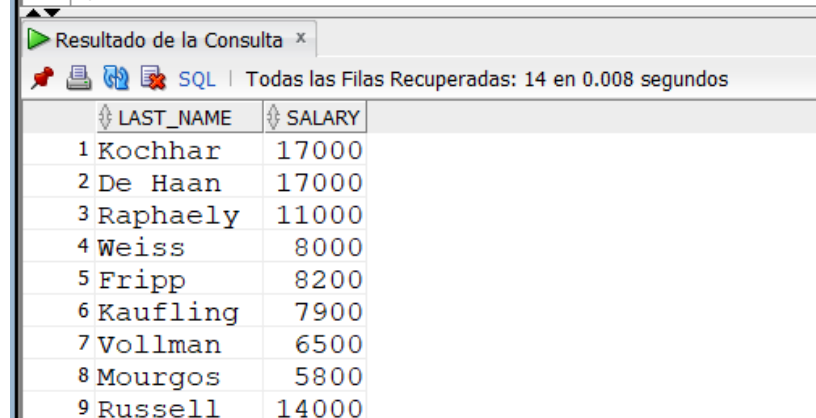
Todas las Filas Recuperadas: 18 en 0.008 segundos

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	King	90 AD	PRES
2	Kochhar	90 AD	VP
3	De Haan	90 AD	VP
4	Greenberg	100 FI	MGR
5	Faviet	100 FI	ACCOUNT
6	Chen	100 FI	ACCOUNT
7	Sciarra	100 FI	ACCOUNT

Figure 14: *Using subqueries: HR Schema.*

5. Create a report for HR that displays the last name and salary of every employee who reports to King, see figure 15.

```
SELECT LAST_NAME, SALARY FROM EMPLOYEES
WHERE MANAGER_ID IN (SELECT EMPLOYEE_ID
FROM EMPLOYEES WHERE LAST_NAME = 'King');
```



Resultado de la Consulta x

Todas las Filas Recuperadas: 14 en 0.008 segundos

	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Raphaely	11000
4	Weiss	8000
5	Fripp	8200
6	Kaufling	7900
7	Vollman	6500
8	Mourgos	5800
9	Russell	14000

Figure 15: *Using subqueries: HR Schema.*

6. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department, see figure 16.

```
SELECT DEPARTMENT_ID, LAST_NAME, JOB_ID FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID FROM DEPARTMENTS
WHERE DEPARTMENT_NAME = 'Executive');
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 3 en 0.016 segundos

	DEPARTMENT_ID	LAST_NAME	JOB_ID
1	90	King	AD PRES
2	90	Kochhar	AD VP
3	90	De Haan	AD VP

Figure 16: Using subqueries: HR Schema.

7. Modify the query in lab\_13.03.sql to display the employee number, last name, and salary of all employees who earn more than the average salary, and who work in a department with any employee whose last name contains a “u.” Resave lab\_13.03.sql as lab\_13.07.sql. Run the statement in lab\_13.07.sql, see figure 17.

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID FROM EMPLOYEES
WHERE LAST_NAME LIKE '%u%') AND SALARY > (SELECT AVG(SALARY)
FROM EMPLOYEES);
```

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 36 en 0.01 segundos

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	123	Vollman	6500
3	122	Kaufling	7900
4	121	Fripp	8200
5	120	Weiss	8000
6	177	Livingston	8400
7	176	Taylor	8600
8	175	Hutton	8800

Figure 17: Using subqueries: HR Schema.

## 4 Pre-assessment

In this section you will find the Pre-assessment.

Criteria to be evaluate	Does it comply?	(%)
COMPLIES WITH THE REQUESTED FUNCTIONALITY	YES	
HAS THE CORRECT INDENTATION	YES	
HAS AN EASY WAY TO ACCESS THE PROVIDED FILES	YES	
HAS A REPORT WITH IDC FORMAT	YES	
REPORT INFORMATION IS FREE OF SPELLING ERRORS	YES	
DELIVERED IN TIME AND FORM	YES	
IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)	YES	100%

## 5 Conclusion

The projection of the data within a relational database is stored in the table in the form of rows and columns. Projections are the first items identified during query execution. They are the selected columns within a table for which a query has been designed. Projections are mentioned in the first part of the SQL query, that is, the SELECT statement. After identifying the projections within the query frame, the next step would be to identify the rows that are relevant to the query. Filters are mentioned within the WHERE clause of the query and will identify the rows to be included in the results, the latter is called a selection.

This practice number 13 helped me practice the uses of the SELECT statement for data retrieval and projection. Finally, something important to mention is that the SQL language allows the projection and selection of data to meet the reporting needs that a programmer, developer or end user may need.