Practice 13: Using Subqueries

Jesús Manuel Juárez Pasillas

November 11, 2021

1 Introduction

Subqueries are very important when making complex queries, since many times you will want data from a query within another query to make a comparison, or simply to manipulate the data together with the data that the query itself consults. External queries can have more than one subquery, which can be in various parts of the external query.

2 Development

2.1 Activity 1:

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

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?

   A. **select_list**

   B. **expression_2**

   C. **condition**

   D. expression_1

   E. **table**

   F. **expression_3**

   As long as the rules for each of these parts of the query are not violated, subqueries can be carried out, in addition to knowing if the subqueries will return a value or several to use an appropriate operator.

2. A query can have a subquery embedded within it. Under what circumstances could there be more than one subquery?

   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.

   The queries can contain the subqueries that they require, in addition also the subqueries can have subqueries, although this is not highly recommended due to the cost it requires.

3. Consider this statement:
   select employee_id, last_name from employees where
   salary > (select avg(salary) from employees);
   When will the subquery be executed?

   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.**

   Subqueries always run first than outer queries because subqueries return important values so the outer query runs smoothly.

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?

   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.

   It is executed once for each record due to the comparison that is made within the subquery which compares a value from the subquery with a value from the outer query.

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?

   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.**

   The two queries will always be correct, even if there is more than one value with 'Executive' since in the second query it uses the in operator that verifies that it is in the range of values returned by the subquery, in addition to always returning the same thing due to its structure that what it does is do the same with different methods.

6. What are the distinguishing characteristics of a scalar subquery?

   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.**

   Scalar subqueries only return a single value for both rows and columns.

7. Which comparison operator can be used with multiple-row subqueries?

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

All of these comparison operators are used to perform comparisons with multiple items so they can be compared with subqueries that return multiple rows.

8. Consider this statement:
   select last_name, (select count(*) from departments) from employees where salary = (select salary from employees);
   What is wrong with it?

   A. The 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.

   As to compare the salary value with the second subquery it uses a single line operator, if this subquery returns more than one value, the comparison will fail.

9. Which of the following statements are equivalent?

   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);**

   The first query gets all the employees who have a lower salary than all the employees with department 10 and the last sentence gets the employees who have a salary less than the minimum salary of the employees with department 10. The first query does the same than the second, but making more comparisons.

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?

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. **<u>The statement would return no rows.</u>**

Since the subquery will not return any value, the outer query will not return any value either as it requires the condition stated where the job_id is equal to the job_id of the employee with the entered name to be met.

2.2 Activity 2:

Propose an answer to the following issues:

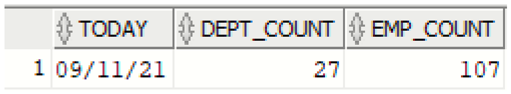
- How can you best design subqueries such that they will not fail with "ORA-01427: single-row subquery returns more than one row" errors?
  - **To perform a subquery that only returns a value, it is necessary to use conditions where data that cannot be repeated by the records, such as identifiers, are compared.**
- 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?
  - **The star transformation is much better since the subqueries do not have to be performed more than once, since the returned values are much smaller, in addition they already make a comparison, instead in the first query it has to be constantly comparing at least the two data from each table, one with the main table and the other with the given value.**

2.3 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:

- **Select sysdate Today, (Select count(*) from Departments) Dept_Count, (Select count(*) from Employees) Emp_Count from dual;**

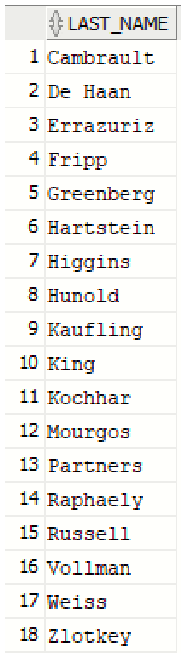


	TODAY	DEPT_COUNT	EMP_COUNT
1	09/11/21	27	107

Figure 1: Date of today and numbers of departments and staff.

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:

- **Select last_name from Employees where employee_id in (Select manager_id from Employees);**

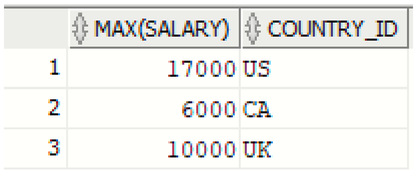


	LAST_NAME
1	Cambrault
2	De Haan
3	Errazuriz
4	Fripp
5	Greenberg
6	Hartstein
7	Higgins
8	Hunold
9	Kaufling
10	King
11	Kochhar
12	Mourgos
13	Partners
14	Raphaely
15	Russell
16	Vollman
17	Weiss
18	Zlotkey

Figure 2: Employees who are managers.

c) Write a query to identify the highest salary paid in each country. This will require using a subquery in the FROM clause:

- **Select * from (Select max(salary),country_id from Employees natural join Departments natural join Locations where department_id is not null group by country_id);**

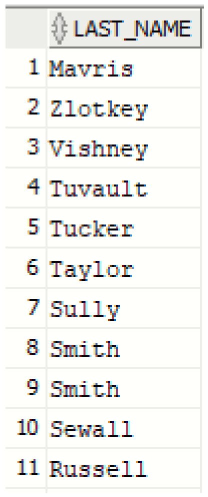


	MAX(SALARY)	COUNTRY_ID
1	17000	US
2	6000	CA
3	10000	UK

Figure 3: Highest salary paid in each country.

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:

- **Select last_name from Employees where department_id in (Select department_id from Departments where location_id in (Select location_id from Locations where country_id = (Select country_id from Countries where country_name = 'United Kingdom')));**

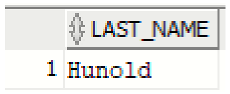


```
   LAST_NAME
1  Mavris
2  Zlotkey
3  Vishney
4  Tuvault
5  Tucker
6  Taylor
7  Sully
8  Smith
9  Smith
10 Sewall
11 Russell
```

Figure 4: Employees who work in the United Kingdom.

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:

- **Select last_name from Employees where salary > (Select avg(salary) from Employees) and department_id in (Select department_id from Departments where department_name like 'IT%');**



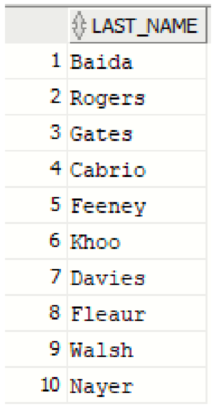
```
   LAST_NAME
1  Hunold
```

Figure 5: Employees who work in any of the IT departments.

f) Write a query to determine who earns more than Mr. Tobias.
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):

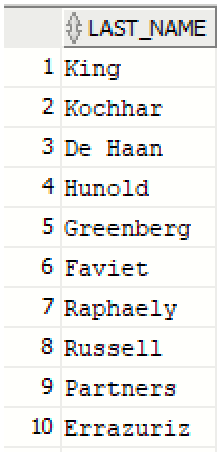
- **First solution:**
Select last_name from Employees where salary > all
(Select salary from Employees where last_name = 'Tobias');



	LAST_NAME
1	Baida
2	Rogers
3	Gates
4	Cabrio
5	Feeney
6	Khoo
7	Davies
8	Fleaur
9	Walsh
10	Nayer

Figure 6: First solution.

- **The second solution:**
Select last_name from Employees where salary >
(Select max(salary) from Employees
where last_name = 'Taylor');

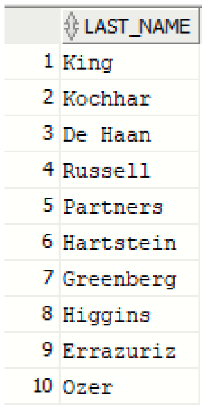


	LAST_NAME
1	King
2	Kochhar
3	De Haan
4	Hunold
5	Greenberg
6	Faviet
7	Raphaely
8	Russell
9	Partners
10	Errazuriz

Figure 7: Second solution.

g) Construct two other solutions, one using the ANY comparison operator, the other using the MIN aggregation function.

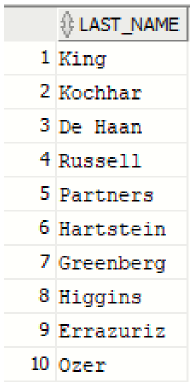
- **First solution using any:**
  Select last_name from Employees where salary > any
  (Select salary from Employees where last_name = 'Tobias');



	LAST_NAME
1	King
2	Kochhar
3	De Haan
4	Russell
5	Partners
6	Hartstein
7	Greenberg
8	Higgins
9	Errazuriz
10	Ozer

Figure 8: First solution using any.

- **The second solution using min:**
  Select last_name from Employees where salary >
  (Select min(salary) from Employees
  where last_name = 'Taylor');



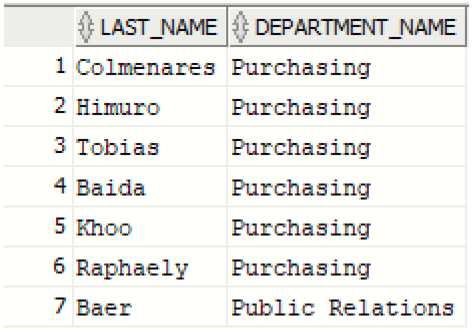
	LAST_NAME
1	King
2	Kochhar
3	De Haan
4	Russell
5	Partners
6	Hartstein
7	Greenberg
8	Higgins
9	Errazuriz
10	Ozer

Figure 9: Second solution using min.

- **These two queries give erroneous results for the tools it uses, as in any any employee with more salary than at least one of the employees will return and with the min function it does the same.**

h) Design a query that will prompt for a department name and list the last name of every employee in that department:

- **Select e.last_name, d.department_name from Employees e.departments d where (e.department_id = d.department_id) and e.department_id in (Select department_id from Departments where department_name like '%&Department_name%');**



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

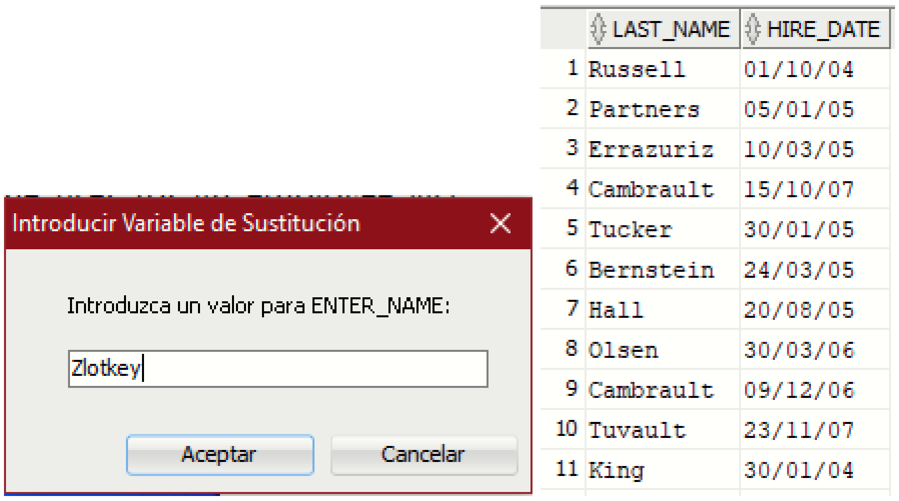
Figure 10: Prompt for a department name.

2.4 Activity 4:

You will write complex queries using nested SELECT statements.

1. 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).

   - **Select e.last_name, e.hire_date from Employees e join (Select employee_id,department_id from Employees where last_name = '&ENTER_NAME') d on (e.department_id = d.department_id) where e.employee_id != d.employee_id;**


Introducir Variable de Sustitución

Introduzca un valor para ENTER_NAME:

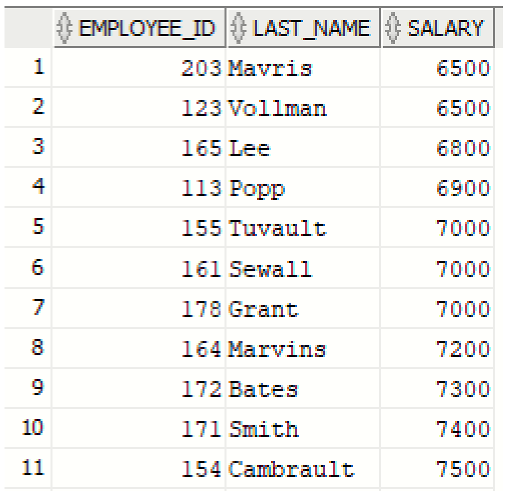
Zlotkey

	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

Figure 11: Employees in the same department.

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.

   - **Select employee_id, last_name, salary from Employees where salary > (Select avg(salary) from Employees) order by salary;**

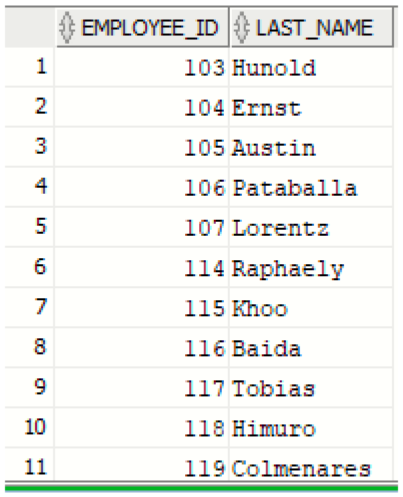


	EMPLOYEE_ID	LAST_NAME	SALARY
1	203	Mavris	6500
2	123	Vollman	6500
3	165	Lee	6800
4	113	Popp	6900
5	155	Tuvault	7000
6	161	Sewall	7000
7	178	Grant	7000
8	164	Marvins	7200
9	172	Bates	7300
10	171	Smith	7400
11	154	Cambrault	7500

Figure 12: Employees who earn more than the average.

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.

    - **The file is located inside the file called ”Plugins.zip”.**

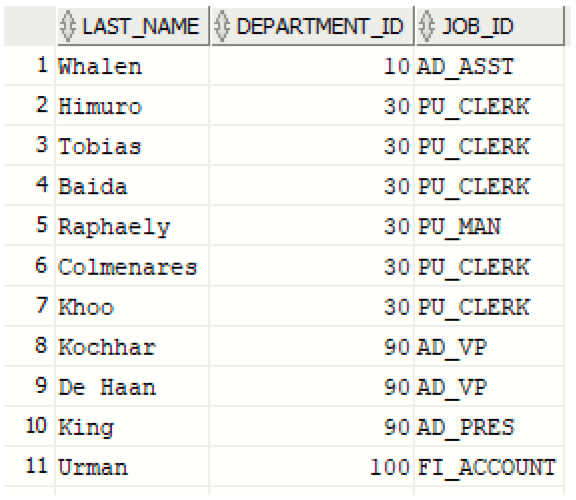


	EMPLOYEE_ID	LAST_NAME
1	103	Hunold
2	104	Ernst
3	105	Austin
4	106	Pataballa
5	107	Lorentz
6	114	Raphaely
7	115	Khoo
8	116	Baida
9	117	Tobias
10	118	Himuro
11	119	Colmenares

Figure 13: Employee whose last name contains the letter “u”.

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.
   Modify the query so that the user is prompted for a location ID. Save this to a file named lab_13_04.sql.

    - **Select last_name, department_id, job_id from Employees where department_id in (Select department_id from Departments where location_id = 1700) order by department_id;**
    - **The file is located inside the file called ”Plugins.zip”.**

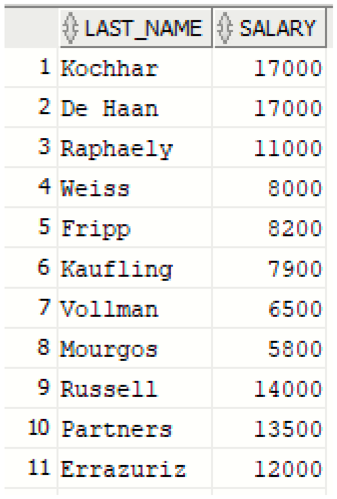


	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen	10	AD_ASST
2	Himuro	30	PU_CLERK
3	Tobias	30	PU_CLERK
4	Baida	30	PU_CLERK
5	Raphaely	30	PU_MAN
6	Colmenares	30	PU_CLERK
7	Khoo	30	PU_CLERK
8	Kochhar	90	AD_VP
9	De Haan	90	AD_VP
10	King	90	AD_PRES
11	Urman	100	FI_ACCOUNT

Figure 14: Employees whose department location ID is 1700.

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

   - **Select last_name, salary from Employees where manager_id = (Select employee_id from Employees where last_name = 'King' and manager_id is null);**

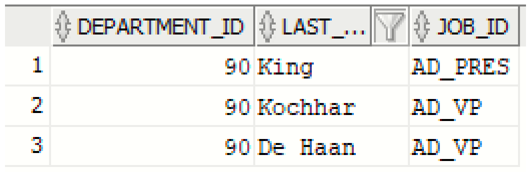


	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Raphaely	11000
4	Weiss	8000
5	Fripp	8200
6	Kaufling	7900
7	Vollman	6500
8	Mourgos	5800
9	Russell	14000
10	Partners	13500
11	Errazuriz	12000

Figure 15: Employee who reports to King.

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

   - **Select department_id, last_name, job_id from Employees where department_id = (Select department_id from Departments where department_name = 'Executive');**

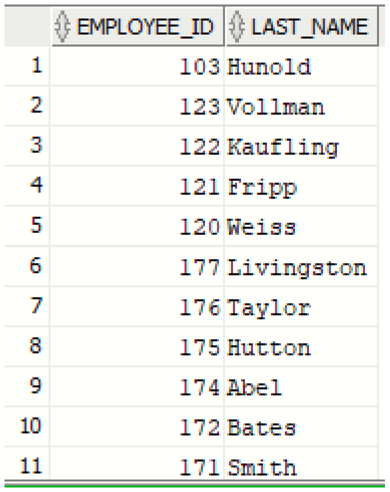


	DEPARTMENT_ID	LAST_...	JOB_ID
1	90	King	AD_PRES
2	90	Kochhar	AD_VP
3	90	De Haan	AD_VP

Figure 16: Employee in the Executive department.

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.

   - **The file is located inside the file called "Plugins.zip".**



	EMPLOYEE_ID	LAST_NAME
1	103	Hunold
2	123	Vollman
3	122	Kaufling
4	121	Fripp
5	120	Weiss
6	177	Livingston
7	176	Taylor
8	175	Hutton
9	174	Abel
10	172	Bates
11	171	Smith

Figure 17: Employees who earn more than the average salary, and who work in a department with any employee whose last name contains a "u".

3 Pre-Assessment:

- Practices pre-Assessment for Database Systems Laboratory II

Practice	Pre-Assessment
COMPLIES WITH THE REQUESTED FUNCTIONALITY	X
HAS THE CORRECT INDENTATION	X
HAS AN EASY WAY TO ACCESS THE PROVIDED FILES	X
HAS A REPORT WITH IDC FORMAT	X
REPORT INFORMATION IS FREE OF SPELLING ERRORS	X
DELIVERED IN TIME AND FORM	X
IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)	100%

Table 1: Pre-Assessment.

4 Conclusion:

Queries that contain subqueries are very useful when querying data that cannot be known only using joins or some function, but rather that they need a subquery to make use of these or only return some particular data to be able to compare data or perform them. some function for the external query to comply with the requested information.