Practice 13

Practice name	Using Subqueries
Academic Program	Software Engineering
Subject name	Laboratory of Database Systems II
Unit	I. SQL.
Professor	Aldonso Becerra Sánchez
Due date	November 10, 2022
Due date with penalty	November 11, 2022
Elaboration date	November 8, 2022

Practice objective	Use SQL SELECT statements for retrieving data from several sources using different operations.
Estimated time of completion	5 hours
Introduction	SQL language allows the realization of projection and selection of data from several tables to satisfy the needs of reports that may be required for a programmer, developer or end user.

Reference 1:

1. Oracle Database 11g: SQL Fundamentals.

Reference 2:

2. Oracle Database SQL Language Reference 11g.

Reference 3:

Initial Activity:

Read the whole practice before start it.

Write the corresponding report, starting with the **Introduction** section.

Activity 1:

Write the section that describes the **work developed** in the following activities.

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

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.

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.

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.

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.

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.

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.

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. 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.

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);

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.

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?

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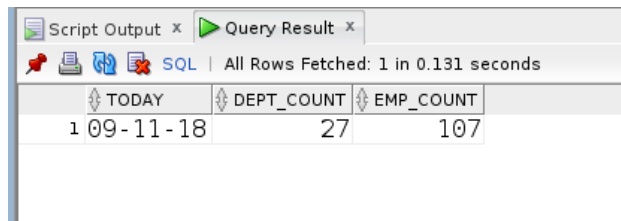
NOTE (on the JOB): There is an instance initialization parameter, STAR_TRANSFORMATION_ENABLED, which (if set to true) will permit the Oracle query optimizer to re-write code into star queries.

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?

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:

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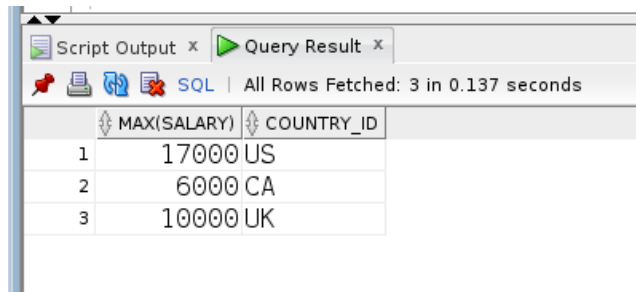
	LAST_NAME
1	Cambrault
2	De Haan
3	Errazuriz
4	Fripp
5	Greenberg
6	Hartstein
7	Higgins
8	Hunold
9	Kaufling
10	King
11	Kochhar
12	Mourgos
13	Partners
14	Raphaely
15	Russell
16	Vollman
17	Weiss
18	Zlotkey

c) Write a query to identify the highest salary paid in each country. This will require using a subquery in the FROM clause:
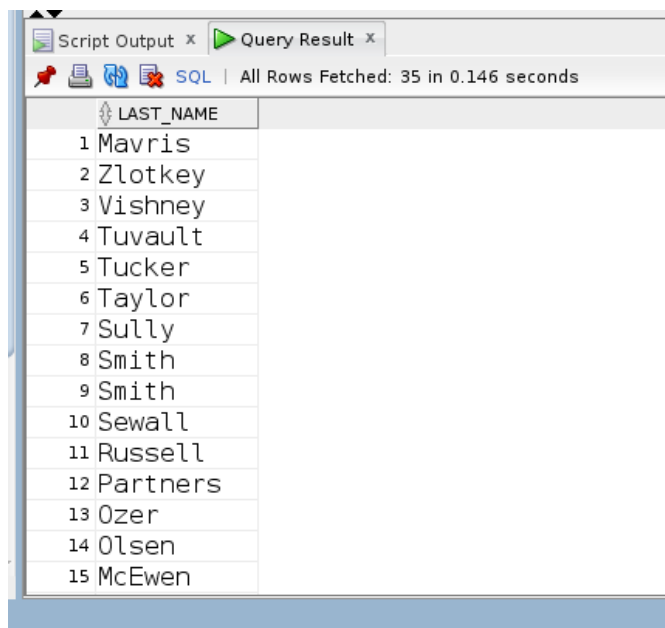
	MAX(SALARY)	COUNTRY_ID
1	17000	US
2	6000	CA
3	10000	UK

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:
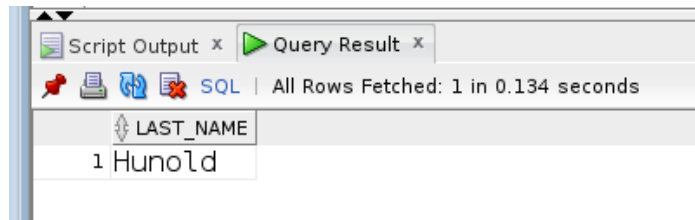
	LAST_NAME
1	Mavris
2	Zlotkey
3	Vishney
4	Tuvault
5	Tucker
6	Taylor
7	Sully
8	Smith
9	Smith
10	Sewall
11	Russell
12	Partners
13	Ozer
14	Olsen
15	McEwen

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:

Script Output × Query Result ×
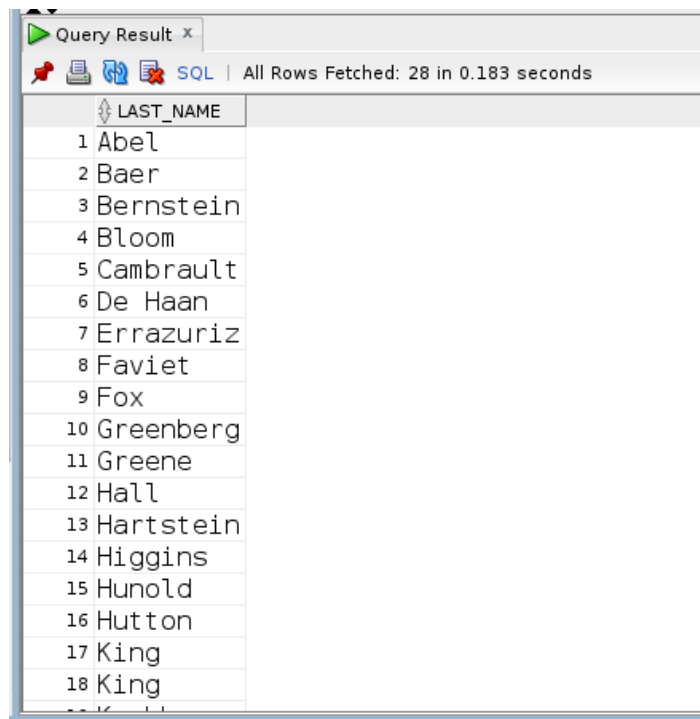
SQL | All Rows Fetched: 1 in 0.134 seconds

	LAST_NAME
1	Hunold

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

The second solution:

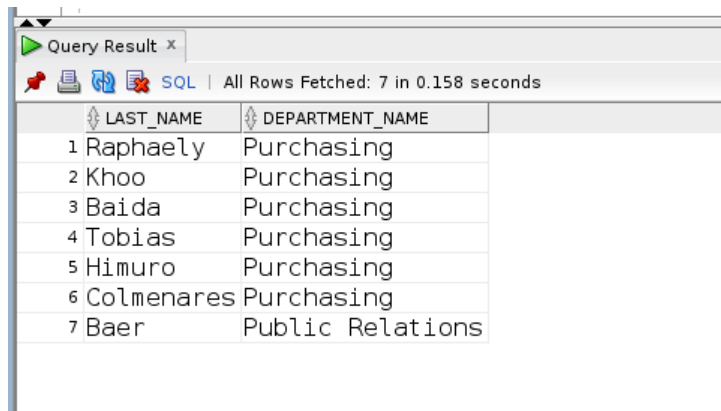Query Result ×

SQL | All Rows Fetched: 28 in 0.183 seconds

	LAST_NAME
1	Abel
2	Baer
3	Bernstein
4	Bloom
5	Cambrault
6	De Haan
7	Errazuriz
8	Faviet
9	Fox
10	Greenberg
11	Greene
12	Hall
13	Hartstein
14	Higgins
15	Hunold
16	Hutton
17	King
18	King

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?

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…":
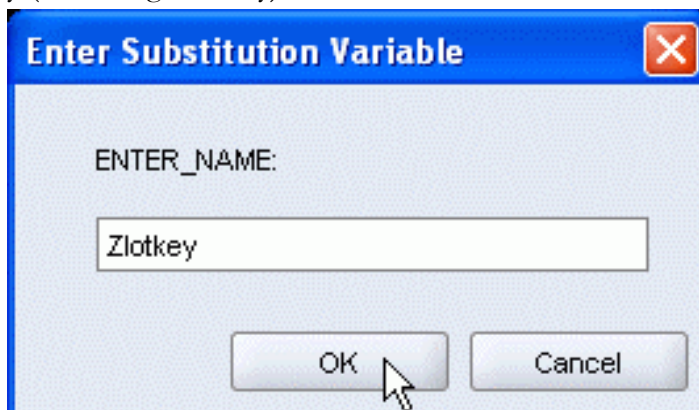

The NOTE: Capture an image for each statement output.

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).

	LAST_NAME	HIRE_DATE
1	Abel	11-MAY-96
2	Taylor	24-MAR-98

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.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12000
5	201	Hartstein	13000
6	101	Kochhar	17000
7	102	De Haan	17000
8	100	King	24000

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.

	EMPLOYEE_ID	LAST_NAME
1	124	Mourgos
2	141	Rajs
3	142	Davies
4	143	Matos
5	144	Vargas
6	103	Hunold
7	104	Ernst
8	107	Lorentz

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.

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen	10	AD_ASST
2	King	90	AD_PRES
3	Kochhar	90	AD_VP
4	De Haan	90	AD_VP
5	Higgins	110	AC_MGR
6	Gietz	110	AC_ACCOUNT

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

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

	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Mourgos	5800
4	Zlotkey	10500
5	Hartstein	13000

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

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

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`.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000

Activity 5:

Pre-assessment section.

Final activity:

Conclusion section.

Attached file that is required for this task (optional):

e-mail: a7donso@gmail.com