## Lab Experiment 7

### To demonstrate the usage of Subqueries in SQL

### 

### Objectives

* Demonstration of various types of subqueries including correlated, scalar & multi-row subqueries

### Introduction

In SQL a Subquery can be simply defined as a query within another query. In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query. Important rules for Subqueries:

* You can place the Subquery in a number of SQL clauses: WHERE clause, HAVING clause, FROM clause. Subqueries can be used with SELECT, UPDATE, INSERT, DELETE statements along with expression operator. It could be equality operator or comparison operator such as =, >, =, <= and Like operator.
* A subquery is a query within another query. The outer query is called as **main query** and inner query is called as**subquery**.
* The subquery generally executes first when the subquery doesn’t have any**co-relation** with the **main query**, when there is a co-relation the parser takes the decision **on the fly**on which query to execute on **precedence** and uses the output of the subquery accordingly.
* Subquery must be enclosed in parentheses.
* Subqueries are on the right side of the comparison operator.
* Use single-row operators with single row Subqueries. Use multiple-row operators with multiple-row Subqueries.

##### Syntax

**Syntax:** There is not any general syntax for Subqueries. However, Subqueries are seen to be used most frequently with SELECT statement as shown below:

SELECT column\_name  
FROM table\_name  
WHERE column\_name expression operator   
 (SELECT column\_name FROM table\_name WHERE ...);

Employee table has employee data as below:

E\_ID Name Salary Dept\_ID

1 A 10000 1

2 B 20000 1

3 C 30000 1

4 D 40000 2

5 E 50000 2

6 F 60000 1

7 G 70000 1

8 H 80000 2

9 I 90000 3

10 J 25000 1

11 A 30000 4

12 K 35000 4

13 L 40000 4

14 E 45000 4

Department table has details of the departments as below:

Dept\_ID name

1 HR

2 QA

3 IT

4 Sales

5 Marketing

6 Production

### Example:

**List the details of the employee who has the highest salary?**

Select \* from employee where salary = (select max(salary) from employee)

Here the inner query is also called scalar query as it returns a single value. Inner query is executed first and its result is then used to execute the main query.

Here’s another example.

**List the details of the employee who has the second highest salary**

select \* from employee where

salary =(select max(salary) from employee where

salary< (select max(salary) from employee))

**What if you had been asked to show the details of the employee who had 50th highest salary or n-th highest salary?** The above nested subquery would not be as scalable. Consider the following query:

SELECT \* FROM emp e1

WHERE N-1 = (

SELECT COUNT(DISTINCT salary) FROM emp e2

WHERE e2.salary > e1.salary)

The above subquery is called **correlated query** where the inner query uses part of the outer query. Every row of outer query is compared to the inner query to output the result. Execute it and understand the operation of this query to get details of employees with 3rd highest and 4th highest salary.

### Task 1:

**Give the names of all employees who are being underpaid with respect to average salaries of the employees**

### Task 2:

**Give the names of the employees who work in department with 2 or more employees.**

Let’s take a look at how to approach such problems. Consider below query:

Select dept\_ID from employee

Group by dept\_ID

Having count(\*)>=2

This query will give the department IDs of those departments where 2 or more employees are working. Test this query and observe the result generated using your dataset.

Now we need those employees who work in these departments. So, we will write the statement as:

Select name from employee

where dept\_ID IN

( Select dept\_ID from employee

Group by dept\_ID

Having count(\*)>=2 )

Now, **what if we had to show the names of the departments in the above example as well**? Then, we had to join the employees table with department table to get the desired data. See below:

Select e.name,e.dept\_ID,d.name from employee e

join department d on e.dept\_ID=d.d\_ID

where e.dept\_ID IN

( Select e1.dept\_ID from employee e1

Group by e1.dept\_ID

Having count(\*)>=2 )

order by d.name desc

### Task 3:

**Retrieve the details of those employees who have salary higher than the average salary of employees in the respective department**

How to proceed??First, we need to check if salary of an employee is higher than the average salary in his department and for that average salary of that department is needed which can be fetched using following statement:

Select avg(salary) from employee where department\_ID = 1;

Using the above query, we generalize it for all departments as follows using correlated query:

Select \* from employee e1

Where salary >

(

Select avg(salary) from employee e2

Where e1.dept\_ID = e2.dept\_ID

)

### Task 4:

**Fetch details of all employees whose salary is equal to any salary in QA department**

Select e.salary from employee e

join department d

on e.dept\_id = d.dept\_id

Where name=’QA’

Above query will return the salaries of those employees who are in QA department

What now then? We need the details of those employees whose salary is equal to any salary retrieved by above query.

Select \* from employee emp

Where emp.salary = any (Select e.salary from employee e

join department d

on e.dept\_id = d.dept\_id

Where name=’QA’

)

**ANY** operator is used in combination with comparison operators to compare a value with a set of values result by a subquery (multi-row subquery). It returns true if comparison is true for at least one value in the set. This operator is typically used with where and having operators

Select column\_name(s)

From table\_name

Where column\_name *operator* Any (Subquery);

*Operator* refers to standard comparison operators

### Task 5:

**Fetch details of all employees whose salary is greater than the salary of all employees in ‘QA’ Department.**

*Hint: Use ALL operator instead of ANY*

### Task 6:

**Fetch details of all departments which have employees working in them.**

To approach this query, we are going to use another operator called EXISTS.

Step 1: Write a subquery to fetch all the dept\_IDs from employees table.

Select dept\_ID from employee

Step 2: Check if the dept\_ID’s in department table exist in employee’s table.

Select name from department where

exists (select \* from employee where employee.dept\_ID=department.d\_ID)

### Task 7:

**Retrieve the same data from task 6 using IN operator**

### Task 8:

**Retrieve the same data from task 6 using Joins**

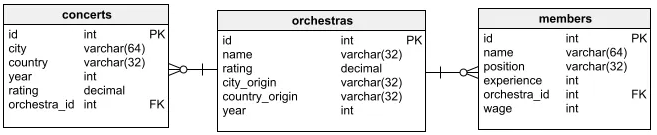
### Task 9:

**Retrieve the details of those departments who don’t have any employees in them using NOT EXISTS operator**

### Task 10:

**Retrieve the details of those departments who don’t have any employees in them using NOT IN operator**

Consider ERD in fig 1. Recall that we have already discussed this dataset in class.



**Fig. 1.**

Based on ERD in Fig. 1., complete the following tasks.

### Task 11:

**Select the names of all orchestras that have the same city of origin as any city in which any orchestra performed in 2013**

### Task 12:

**Select the names and positions (i.e. instrument played) of all orchestra members that have above 10 years of experience and do not belong to orchestras with a rating below 8.0**

### Task 13:

**Show the name and position of orchestra members who earn more than the highest paid violinist**

### Task 14:

**Show the name and number of members for each orchestra that has more members than the average membership of all orchestras in the table**

### Task 15:

**Show the name of orchestras who have 3 or fewer members**

**Rubric for Lab Assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| **The student performance for the assigned task during the lab session was:** | | | |
| Excellent | The student completed assigned tasks without any help from the instructor and showed the results appropriately. | 4 |  |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 |  |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 |  |
| Worst | The student did not complete assigned tasks. | 1 |  |

**Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**