Database Programming with SQL  
10-1: Fundamentals of Subqueries

* **Correlated Subquery -** It accepts a value from the inner query to complete its SELECT statement.
* **Multi-row Subquery -** An inner query that returns one or more rows to the outer query
* **Subquery -** An inner query that is nested within an outer query
* **Multiple-column Subquery -** An inner query that compares multiple columns at the same time
* **Single-row Subquery -** An inner query that returns only one row to the outer query
* **Multiple-column Subquery in Sequence -** An inner query that compares the multiple columns one at a time in different subqueries
* **Inner Query -** Another name for a subquery

1. What is the purpose of using a subquery?

**To nest one query within another so that the outer query can use the results from the inner query to filter, compare, and analyze data. They make complex data retrieval easier by breaking down the query logic into separate steps.**

2. What is a subquery?

**A query nested within another SQL statement, typically within the WHERE clause, FROM clause, or SELECT clause, to provide a result that is used by the outer query.**

3. What DJs on Demand d\_play\_list\_items song\_id’s have the same event\_id as song\_id 45?

**SELECT song\_id**

**FROM d\_play\_list\_items**

**WHERE event\_id = (SELECT event\_id FROM d\_play\_list\_items WHERE song\_id = 45);**

4. Which events in the DJs on Demand database cost more than event\_id = 100?

**SELECT event\_id, event\_cost**

**FROM events**

**WHERE event\_cost > (SELECT event\_cost FROM events WHERE event\_id = 100);**5. Find the track number of the song that has the same CD number as “Party Music for All  
Occasions.”

**SELECT track\_number**

**FROM songs**

**WHERE cd\_number = (SELECT cd\_number FROM songs WHERE song\_title = 'Party Music for All Occasions');**

6. List the DJs on Demand events whose theme code is the same as the code for “Tropical.”

**SELECT event\_id, event\_name**

**FROM events**

**WHERE theme\_code = (SELECT theme\_code FROM themes WHERE theme\_name = 'Tropical');**

7. What are the names of the Global Fast Foods staff members whose salaries are greater than the staff member whose ID is 12?

**SELECT staff\_name**

**FROM staff**

**WHERE salary > (SELECT salary FROM staff WHERE staff\_id = 12);**8. What are the names of the Global Fast Foods staff members whose staff types are not the same as Bob Miller’s?

**SELECT staff\_name**

**FROM staff**

**WHERE staff\_type <> (SELECT staff\_type FROM staff WHERE staff\_name = 'Bob Miller');**

9. Which Oracle employees have the same department ID as the IT department?

**SELECT employee\_name**

**FROM employees**

**WHERE department\_id = (SELECT department\_id FROM departments WHERE department\_name = 'IT');**

10. What are the department names of the Oracle departments that have the same location ID as Seattle?

**SELECT department\_name**

**FROM departments**

**WHERE location\_id = (SELECT location\_id FROM locations WHERE city = 'Seattle');**

11. Indicate whether the statement regarding subqueries is True or False.  
a. It is good programming practice to place a subquery on the right side of the comparison  
operator. **TRUE**  
b. A subquery can reference a table that is not included in the outer query’s FROM clause. **TRUE**  
c. Single-row subqueries can return multiple values to the outer query. **FALSE**

10-2: Single-Row Subqueries  
  
1. Write a query to return all those employees who have a salary greater than that of Lorentz and are in the same department as Abel.

**SELECT employee\_name**

**FROM employees**

**WHERE salary > (SELECT salary FROM employees WHERE employee\_name = 'Lorentz')**

**AND department\_id = (SELECT department\_id FROM employees WHERE employee\_name = 'Abel');**

2. Write a query to return all those employees who have the same job id as Rajs and were hired after Davies.

**SELECT employee\_name**

**FROM employees**

**WHERE job\_id = (SELECT job\_id FROM employees WHERE employee\_name = 'Rajs')**

**AND hire\_date > (SELECT hire\_date FROM employees WHERE employee\_name = 'Davies');**

3. What DJs on Demand events have the same theme code as event ID = 100?

**SELECT event\_name**

**FROM events**

**WHERE theme\_code = (SELECT theme\_code FROM events WHERE event\_id = 100);**

4. What is the staff type for those Global Fast Foods jobs that have a salary less than those of any Cook staff-type jobs?

**SELECT staff\_type**

**FROM jobs**

**WHERE salary < (SELECT MIN(salary) FROM jobs WHERE staff\_type = 'Cook');**

5. Write a query to return a list of department id’s and average salaries where the department’s average salary is greater than Ernst’s salary.

**SELECT department\_id, AVG(salary) AS avg\_salary**

**FROM employees**

**GROUP BY department\_id**

**HAVING AVG(salary) > (SELECT salary FROM employees WHERE employee\_name = 'Ernst');**

6. Return the department ID and minimum salary of all employees, grouped by department ID, having a minimum salary greater than the minimum salary of those employees whose department ID is not equal to 50.

**SELECT department\_id, MIN(salary) AS min\_salary**

**FROM employees**

**GROUP BY department\_id**

**HAVING MIN(salary) > (SELECT MIN(salary) FROM employees WHERE department\_id <> 50);**

10-3: Multiple-Row Subqueries

1. What will be returned by a query if it has a subquery that returns a null ?

**The result will be NULL or no rows.**

2. Write a query that returns jazz and pop songs. Write a multi-row subquery and use the d\_songs and d\_types tables. Include the id, title, duration, and the artist name.

**SELECT d\_songs.id, d\_songs.title, d\_songs.duration, d\_songs.artist\_name**

**FROM d\_songs**

**WHERE type\_id IN (SELECT type\_id FROM d\_types WHERE type\_name IN ('Jazz', 'Pop'));**

3. Find the last names of all employees whose salaries are the same as the minimum salary for any department.

**SELECT last\_name**

**FROM employees**

**WHERE salary = ANY (SELECT MIN(salary) FROM employees GROUP BY department\_id);**

4. Which Global Fast Foods employee earns the lowest salary? Hint: You can use either a single- row or a multiple-row subquery.

**SELECT employee\_name**

**FROM employees**

**WHERE salary = (SELECT MIN(salary) FROM employees);**

5. Place the correct multiple-row comparison operators in the outer query WHERE clause of each of the following:  
a. Which CDs in our d\_cds collection were produced before “Carpe Diem” was produced?

**WHERE year < (SELECT year FROM d\_cds WHERE title = 'Carpe Diem');**  
b. Which employees have salaries lower than any one of the programmers in the IT department?

**WHERE salary < ANY (SELECT salary FROM employees WHERE job\_title = 'Programmer' AND department\_name = 'IT');**

c. What CD titles were produced in the same year as “Party Music for All Occasions” or “Carpe Diem”?

**WHERE year IN (SELECT year FROM d\_cds WHERE title IN ('Party Music for All Occasions',** 'Carpe Diem'));

d. What song title has a duration longer than every type code 77 title?  
WHERE duration \_\_\_\_\_\_\_\_\_(SELECT duration ...

**WHERE duration > ALL (SELECT duration FROM d\_songs WHERE type\_code = 77);**

6. If each WHERE clause is from the outer query, which of the following are true?  
\_\_**T**\_\_a. WHERE size > ANY -- If the inner query returns sizes ranging from 8 to 12, the value 9 could be returned in the outer query.  
\_\_**F**\_\_b. WHERE book\_number IN -- If the inner query returns books numbered 102, 105, 437, and 225 then 325 could be returned in the outer query.  
\_\_**F**\_\_c. WHERE score <= ALL -- If the inner query returns the scores 89, 98, 65, and 72, then 82 could be returned in the outer query.  
\_\_**T**\_\_d. WHERE color NOT IN -- If the inner query returns red, green, blue, black, and then the outer query could return white.  
\_\_**T**\_\_e. WHERE game\_date = ANY -- If the inner query returns 05-Jun-1997, 10-Dec-2002, and 2-Jan-2004, then the outer query could return 10-Sep-2002.

7. The goal of the following query is to display the minimum salary for each department whose minimum salary is less than the lowest salary of the employees in department 50. However, the subquery does not execute because it has five errors. Find them, correct them, and run the query.  
SELECT department\_id  
FROM employees  
WHERE MIN(salary)  
HAVING MIN(salary) >  
GROUP BY department\_id  
SELECT MIN(salary)  
WHERE department\_id < 50;

**SELECT department\_id, MIN(salary)**

**FROM employees**

**GROUP BY department\_id**

**HAVING MIN(salary) < (SELECT MIN(salary) FROM employees WHERE department\_id = 50);**

8. Which statements are true about the subquery below?  
SELECT employee\_id, last\_name  
FROM employees  
WHERE salary =  
(SELECT MIN(salary)  
FROM employees  
GROUP BY department\_id);  
\_\_\_**F**\_\_\_ a. The inner query could be eliminated simply by changing the WHERE clause to  
WHERE MIN(salary).  
\_\_\_**T**\_\_\_ b. The query wants the names of employees who make the same salary as the smallest  
salary in any department.  
\_\_\_**F**\_\_\_ c. The query first selects the employee ID and last name, and then compares that to the  
salaries in every department.  
\_\_\_**T**\_\_\_ d. This query will not execute.

9. Write a pair-wise subquery listing the last\_name, first\_name, department\_id, and manager\_id for all employees that have the same department\_ id and manager\_id as employee 141. Exclude employee 141 from the result set.

**SELECT last\_name, first\_name, department\_id, manager\_id**

**FROM employees**

**WHERE (department\_id, manager\_id) = (SELECT department\_id, manager\_id FROM employees WHERE employee\_id = 141)**

**AND employee\_id <> 141;**

10. Write a non-pair-wise subquery listing the last\_name, first\_name, department\_id, and manager\_id for all employees that have the same department\_ id and manager\_id as employee 141

**SELECT last\_name, first\_name, department\_id, manager\_id**

**FROM employees**

**WHERE department\_id = (SELECT department\_id FROM employees WHERE employee\_id = 141)**

**AND manager\_id = (SELECT manager\_id FROM employees WHERE employee\_id = 141)**

**AND employee\_id <> 141;**

10-4: Correlated Subqueries

1. Explain the main difference between correlated and non-correlated subqueries?

**A correlated subquery depends on the outer query for its values and is executed once for each row processed by the outer query. A non-correlated subquery can be executed independently of the outer query.**

2. Write a query that lists the highest earners for each department. Include the last\_name,  
department\_id, and the salary for each employee.

**SELECT last\_name, department\_id, salary**

**FROM employees e**

**WHERE salary = (SELECT MAX(salary) FROM employees WHERE department\_id = e.department\_id);**

3. Examine the following select statement and finish it so that it will return the last\_name,  
department\_id, and salary of employees who have at least one person reporting to them. So we are effectively looking for managers only. In the partially written SELECT statement, the WHERE clause will work as it is. It is simply testing for the existence of a row in the subquery.  
SELECT (enter columns here)  
FROM (enter table name here) outer  
WHERE 'x' IN (SELECT 'x'  
FROM (enter table name here) inner  
WHERE inner(enter column name here) = inner(enter column name here)  
Finish off the statement by sorting the rows on the department\_id column.

**SELECT last\_name, department\_id, salary**

**FROM employees outer**

**WHERE 'x' IN (SELECT 'x'**

**FROM employees inner**

**WHERE inner.manager\_id = outer.employee\_id)**

**ORDER BY department\_id;**

4. Using a WITH clause, write a SELECT statement to list the job\_title of those jobs whose maximum salary is more than half the maximum salary of the entire company. Name your subquery MAX\_CALC\_SAL. Name the columns in the result JOB\_TITLE and JOB\_TOTAL, and sort the result on JOB\_TOTAL in descending order. Hint: Examine the jobs table. You will need to join JOBS and EMPLOYEES to display the job\_title.

**WITH MAX\_CALC\_SAL AS (**

**SELECT job\_id, MAX(salary) AS job\_total**

**FROM employees**

**GROUP BY job\_id**

**HAVING MAX(salary) > (SELECT MAX(salary) / 2 FROM employees)**

**)**

**SELECT j.job\_title AS job\_title, m.job\_total AS job\_total**

**FROM jobs j**

**JOIN MAX\_CALC\_SAL m ON j.job\_id = m.job\_id**

**ORDER BY job\_total DESC;**

11-1: Ensuring Quality Query Results

**Went over how to create and modify a query to produce specified data using different examples, focusing on the problem and the tables used**.