



# Database Programming with SQL

10-4

Correlated Subqueries



# Objectives

This lesson covers the following objectives:

- Identify when correlated subqueries are needed.
- Construct and execute correlated subqueries.
- Construct and execute named subqueries using the WITH clause.

# Purpose

- Sometimes you have to answer more than one question in one sentence.
- Your friend might ask you if you have enough money for a cinema ticket, popcorn, and a drink.
- Before you can answer your friend, you need to know the prices of the ticket, the popcorn, and the drink.
- You also need to see how much money you have in your pocket.
- So actually, what seemed like an easy question, turns into four questions that you need answers to before you can say "Yes" or "No."

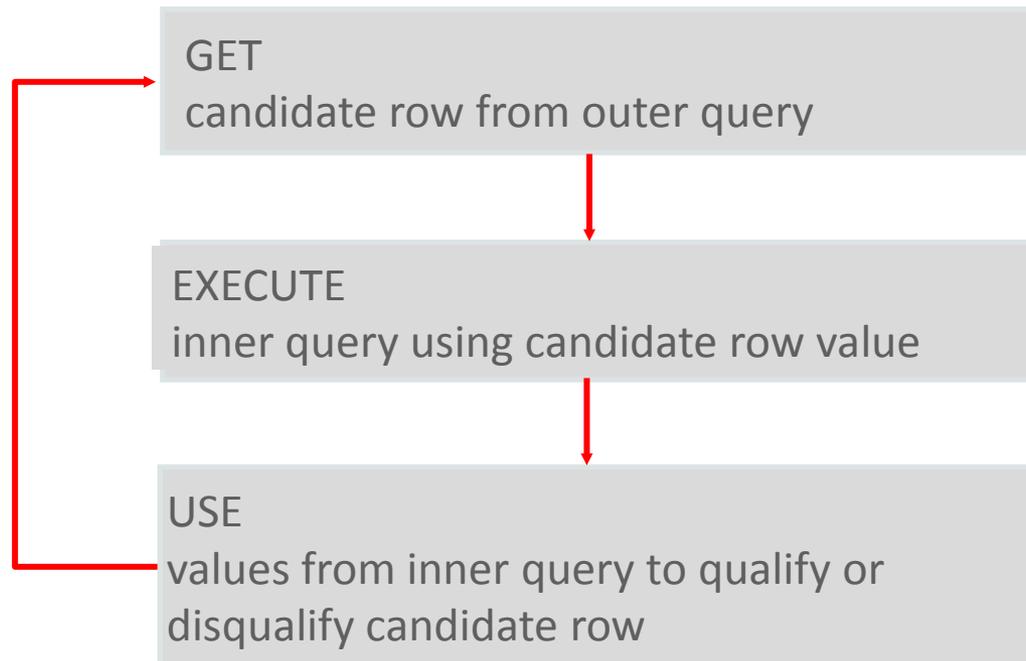
# Purpose

- In business, you might get asked to produce a report of all employees earning more than the average salary for their departments.
- So here you first have to calculate the average salary per department, and then compare the salary for each employee to the average salary of that employee's department.



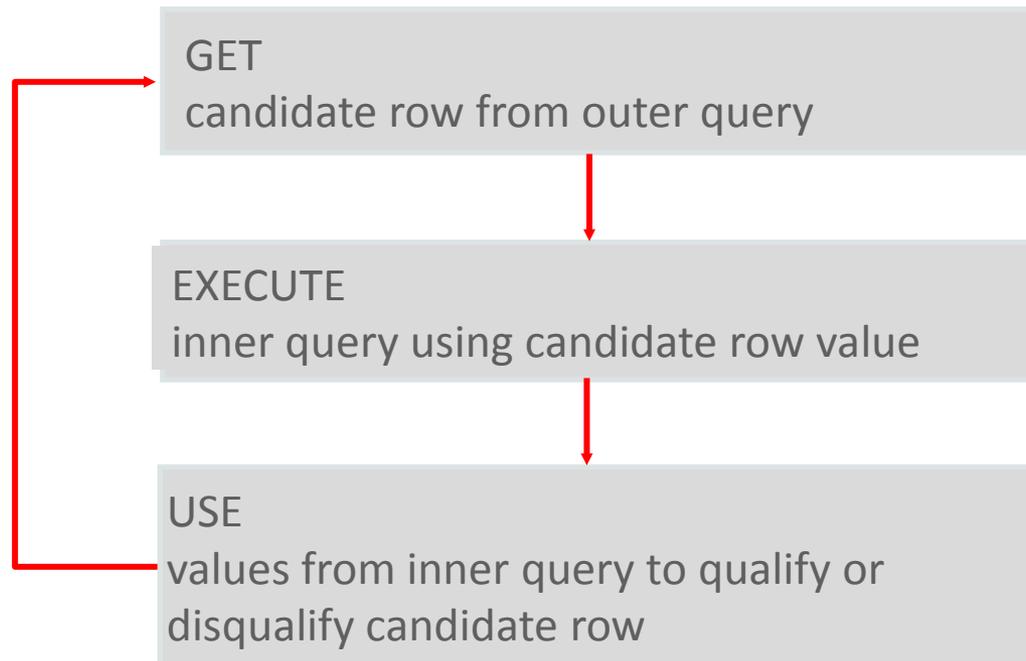
# Correlated Subqueries

- The Oracle server performs a correlated subquery when the subquery references a column from a table referred to in the parent statement.



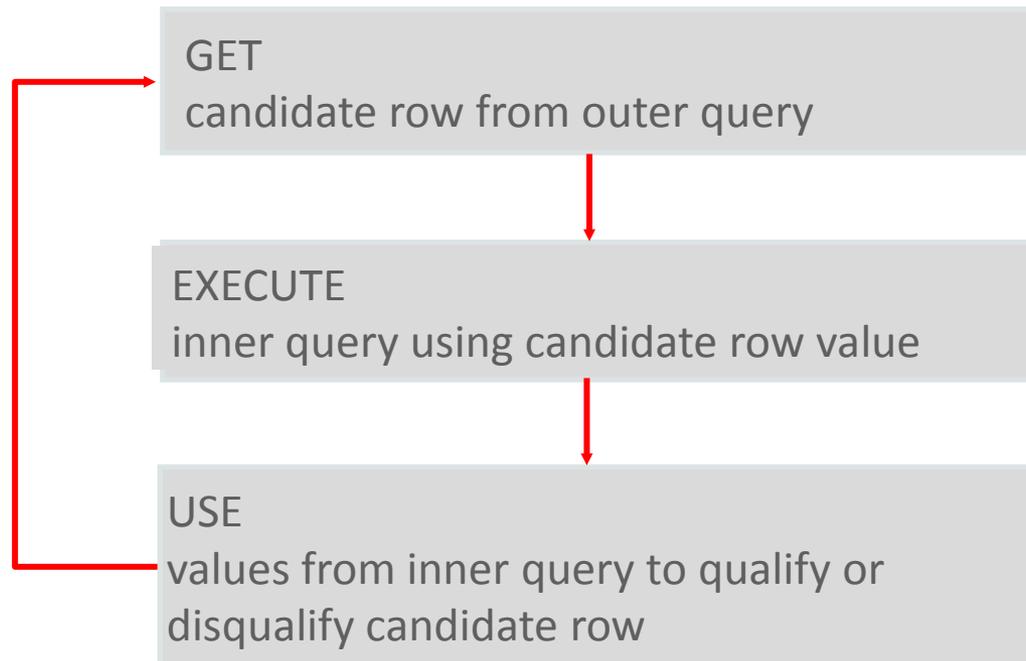
# Correlated Subqueries

- A correlated subquery is evaluated once for each row processed by the parent statement.



# Correlated Subqueries

- The parent statement can be a SELECT, UPDATE, or DELETE statement.



# Correlated Subquery Example

- Whose salary is higher than the average salary of their department?
- To answer that question, we need to write a correlated subquery.
- Correlated subqueries are used for row-by-row processing.

```
SELECT  o.first_name,  
        o.last_name,  
        o.salary  
FROM employees o  
WHERE o.salary >  
      (SELECT AVG(i.salary)  
       FROM employees i  
       WHERE i.department_id =  
             o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

# Correlated Subquery Example

- Each subquery is executed once for every row of the outer query.
- With a normal subquery, the inner SELECT query runs first and executes once, returning values to be used by the outer query.

```
SELECT  o.first_name,  
        o.last_name,  
        o.salary  
FROM employees o  
WHERE o.salary >  
      (SELECT AVG(i.salary)  
       FROM employees i  
       WHERE i.department_id =  
             o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

# Correlated Subquery Example

- A correlated subquery, however, executes once for each row considered by the outer query.
- In other words, the inner query is driven by the outer query.
- The correlated subquery in this example is marked in red.

```
SELECT  o.first_name,  
        o.last_name,  
        o.salary  
FROM employees o  
WHERE o.salary >  
      (SELECT AVG(i.salary)  
       FROM employees i  
       WHERE i.department_id =  
             o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

# WITH Clause

- If you have to write a very complex query with joins and aggregations used many times, you can write the different parts of the statement as query blocks and then use those same query blocks in a SELECT statement.
- Oracle allows you to write named subqueries in one single statement, as long as you start your statement with the keyword WITH.
- The WITH clause retrieves the results of one or more query blocks and stores those results for the user who runs the query.

# WITH Clause

- The WITH clause improves performance.
- The WITH clause makes the query easier to read.
- The syntax for the WITH clause is as follows:

```
WITH subquery-name AS (subquery),  
     subquery-name AS (subquery)  
SELECT column-list  
FROM   {table | subquery-name | view}  
WHERE  condition is true;
```

# WITH Clause

- Write the query for the following requirement:
  - Display a list of employee last names that are not managers.
- To construct this query, you will first need to get a list of `manager_ids` from the `employee` table, then return the names of the employees whose employee id is not on the managers list.
- We can create a named subquery using the `WITH` clause to retrieve the `manager_id` from the `employees` table, then the outer query will return the employees that do not appear on that list.

# WITH Clause

```
WITH managers AS
  (SELECT DISTINCT manager_id
   FROM employees
   WHERE manager_id IS NOT NULL)

SELECT last_name AS "Not a manager"
FROM employees
WHERE employee_id NOT IN
  (SELECT *
   FROM managers);
```

Not a manager
Abel
Davies
Ernst
Fay
Gietz
Grant
Lorentz
Matos
Rajs
Taylor
Vargas
Whalen

# Summary

In this lesson, you should have learned how to:

- Identify when correlated subqueries are needed.
- Construct and execute correlated subqueries.
- Construct and execute named subqueries using the WITH clause.

