Lesson 6 – Using Subqueries to Solve Queries

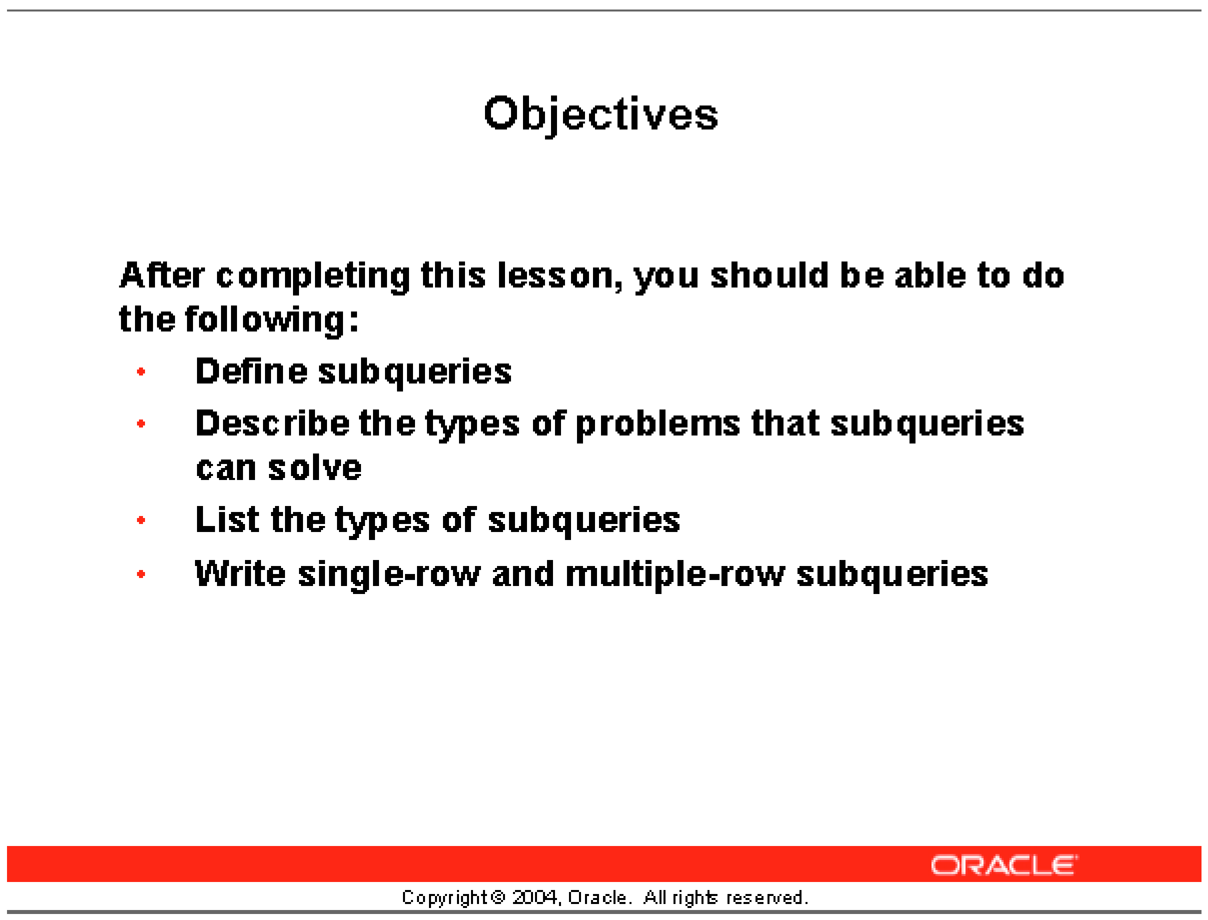
Lesson 7 in Oracle notes

Before starting ask the question

➔ Give me a list of all employees that earn more than Haas

What is the SQL logic?

OBJECTIVES



Objectives

After completing this lesson, you should be able to do the following:

- Define subqueries
- Describe the types of problems that subqueries can solve
- List the types of subqueries
- Write single-row and multiple-row subqueries

After completing this lesson, you should be able to do the following:

→ Define subqueries
→ Describe the type of problems that subqueries can solve
→ List the types of subqueries
→ Write single-row and multiple- row subqueries

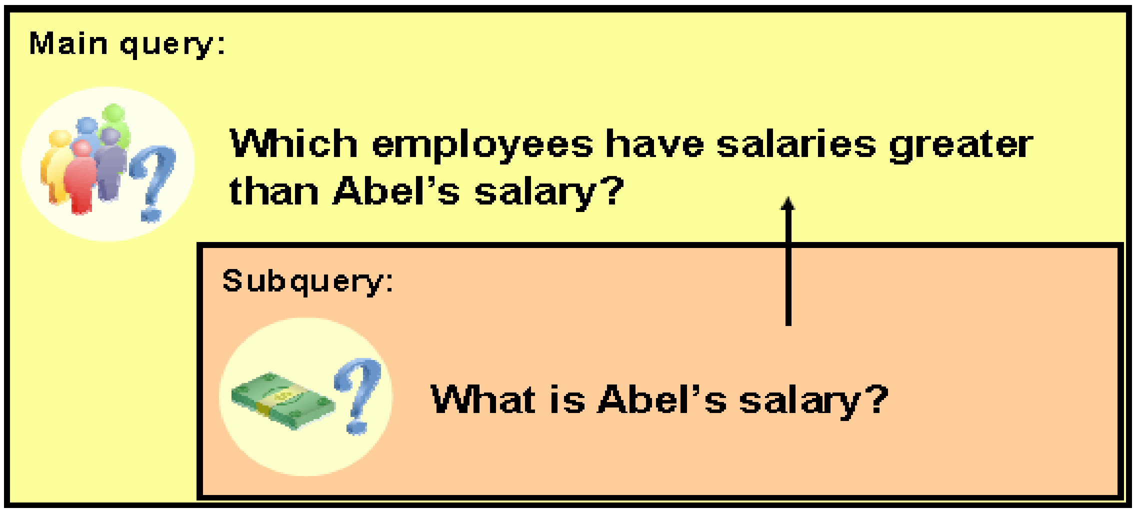
This chapter covers the more advanced features of the SELECT statement.

You can write Subqueries in the where clause of another SQL statement to obtain values based on an unknown conditional value.

This chapter covers single row subqueries and multiple row subqueries.

Using a Subquery to Solve a Problem

Who has a salary greater than Abel's?



Using a Subquery to solve a problem

Problem:

Who has a salary greater than Abel's salary?

Solution:

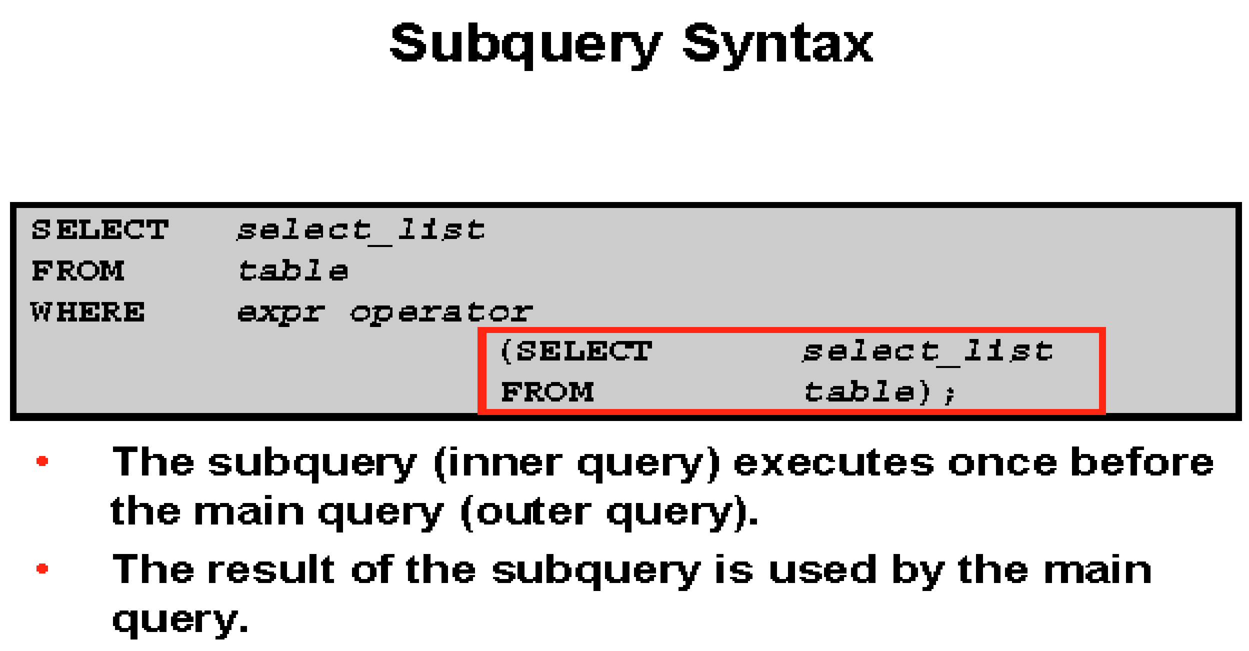
2 steps

→ Find out how much Abel earns

→ Find out who earns more than that amount

That requires two queries. We need to pass information from the first query into the second query. Writing two separate queries does not do that.
We need a Subquery to define Abel's salary and pass it to the main query that produces the results.

7-5 (279)



Subquery Syntax

```
SELECT   select_list
FROM     table
WHERE    expr operator
                  (SELECT       select_list
                   FROM         table);
```

- **The subquery (inner query) executes once before the main query (outer query).**
- **The result of the subquery is used by the main query.**

Subquery Syntax

A Subquery is a SELECT statement that is imbedded in a clause of another SELECT statement.

Useful when you need to select rows from a table with a condition that depend so on data from the same table or other tables.

Where used

On the following clauses:

→ WHERE clause
→ HAVING clause
→ FROM clause

NOTE: operator means

Single-row operator	< > = etc.
Multiple-row operators	IN, ANY, ALL, EXISTS

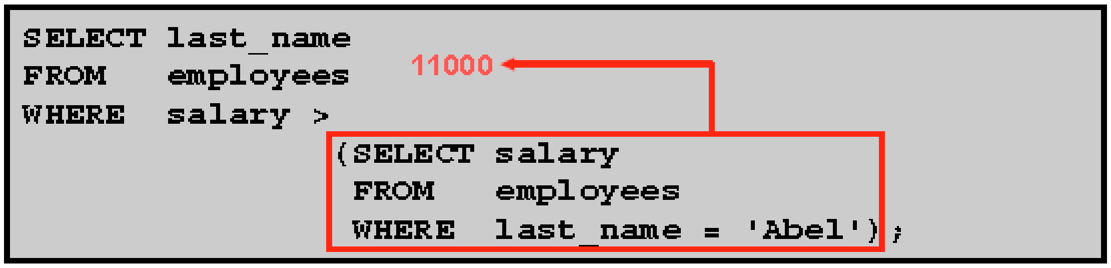
OTHER TERMS USED

Nested SELECT
Sub-SELECT
Inner SELECT

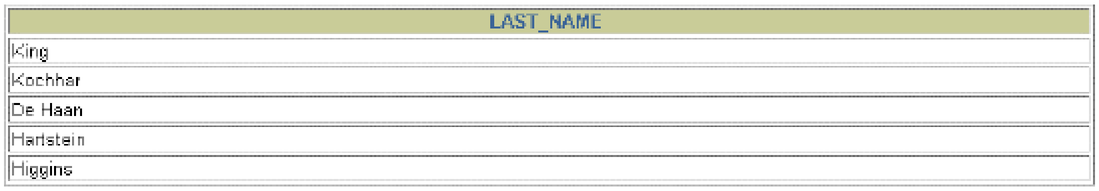
ORDER of OPERATION

The Subquery generally executes first and its output is then the fed to the main or OUTER query.

Using a Subquery



```
SELECT last_name
FROM   employees          11000
WHERE  salary >
                 (SELECT salary
                  FROM   employees
                  WHERE  last_name = 'Abel');
```



LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

The above slide shows how we solve the problem who earns more money than Abel.

Note that the Subquery executes first and returns the value 11,000.

The outer or main query then executes with the 11,000.
It will supply all employees with a salary greater than 11,000.

ASIDE:
A better example would be to show the salary in the output.

Guidelines for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition.
- The `ORDER BY` clause in the subquery is not needed unless you are performing Top-N analysis.
- Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row subqueries.

Guidelines for using Subqueries:

→ A Subquery must be enclosed in parenthesis.

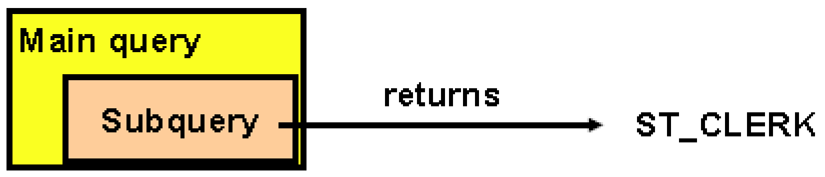
→ Place the Subquery on the right side of the comparison operator for readability
You can do it the other way
SELECT * from employees
WHERE (select salary from employees where last_name = 'Abel') < salary

→ ORDER BY clause in the Subquery is only needed when performing TOP-N analysis
- Normally the order by clause is only found at the end of the SQL statement.
- TOP-N analysis refers two finding the top number of rows.
  - Example top seven salaries

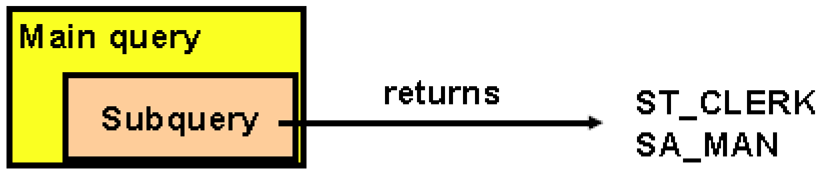
→ 2 types of Subqueries are used:
Single-row operators
Multiple-row operators

Types of Subqueries

- Single-row subquery



- Multiple-row subquery



Types of Subqueries:

This slide shows the two types of Subqueries.

- Single-row Subqueries that return only one row from the inner SELECT statement

- Multiple-row Subqueries return more than one row from the inner SELECT statement

Special note:
There are Subqueries that return multiple columns. These are covered later.

Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Just like other programming languages, the same operators exist

Single-Row Subqueries:
For single row Subqueries that return only one row from the inner SELECT statement, single row operators are used.

NOTE: you cannot use an equal to operator when you are comparing something to multiple rows.

PROBLEM:
Display the employees whose job ID is the same as that of employee 141

SOLUTION:
First find the job ID for employee 141
Use that job ID in the where clause to filter out the employees with the same job ID in the main SELECT statement.

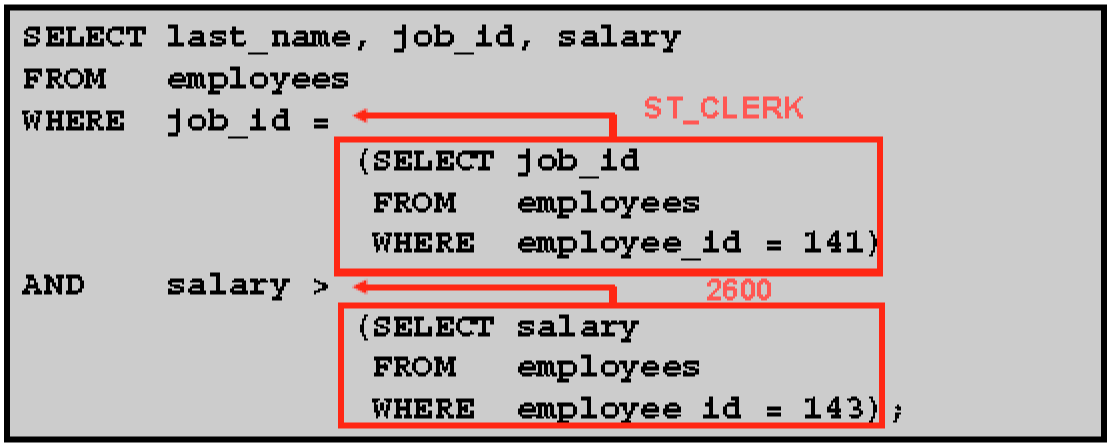
WRITE THE CODE TO DO THIS SOLUTION

Demonstrate by writing INNER query first

```
SELECT      last_name, job_id
FROM        employees
WHERE       job_id =      (SELECT     job_id
                           FROM       employees
                           WHERE      employee_id = 141);
```

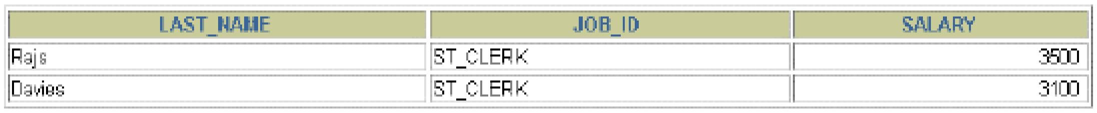
Note: I often write the inner or Subquery first to find what it returns, then I write the main query.

Executing Single-Row Subqueries



```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

Many subqueries can be used



LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

QUERY BLOCKS

A SELECT statement is often called a query block.

→ In the above example there are 3 query blocks.

The inner query block executes first bringing back the results ST_CLERK and 2600

The outer query block is then processed as if the WHERE clause was hard coded with those values that were returned from the inner query.

NOTE:
The Subquery can get information from different tables.

7-12

PROBLEM:

Display the last name, job ID, and salary
of all employees whose salary is equal to the minimum salary of all employees.

SOLUTION:

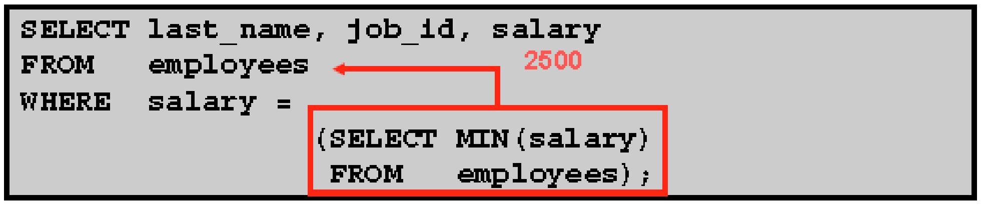
To solve this problem

First get the minimum salary of all employees from the inner SELECT or subquery.
Secondly, use the result in the main query

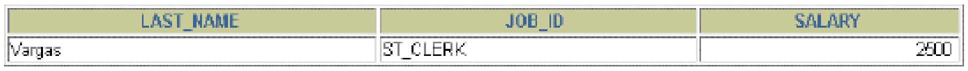
→ The inner query will result in a minimum salary of 2500.

→ The 2500 replaces the right side of the WHERE clause

Using Group Functions in a Subquery



```
SELECT last_name, job_id, salary
FROM   employees          2500
WHERE  salary =
                 (SELECT MIN(salary)
                  FROM   employees);
```



LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

```
SELECT        LAST_NAME, JOB_ID, SALARY
FROM          EMPLOYEES
WHERE          SALARY = (SELECT MIN (SALARY)
                         FROM EMPLOYEES);
```

Group functions in a Subquery:

This example demonstrates that you can get information from the Subquery when the Subquery has a group function in it.

NOTE:

SELECT LAST_NAME, JOB_ID, SALARY
FROM EMPLOYEES
WHERE SALARY = MIN (SALARY); 🡐 can't use group function here

7-13
PROBLEM 1: Display all the departments WITH minimum salary greater DEPARTMENT 50s minimum salary

Another way of saying it
Looking for all minimum salaries in each department that is greater than the minimum in department 50

Step 1 – Find the minimum salary of department 50
- that will require a group function

```
SELECT      min(salary)
FROM        employees
WHERE       department_id = 50;
```

Step 2-Since you want to find the minimum salary in other departments you need the group function in the main query.

Step 3-But you want to limit which groups are displayed. That requires a HAVING statement

Therefore the inner query is attached to the HAVING statement.

```
SELECT      department_id, min(salary)
FROM        employees
GROUP BY    department_id
```

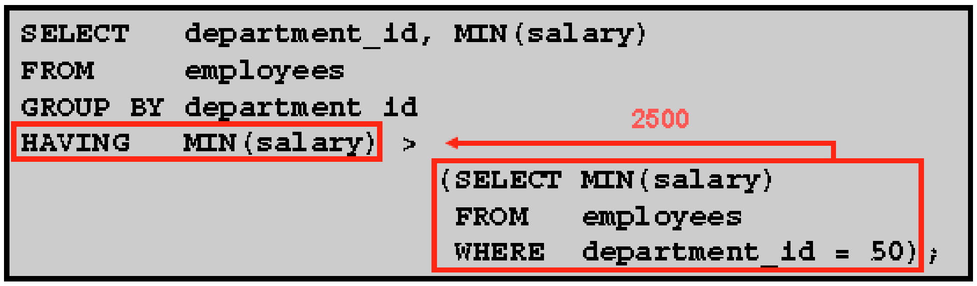
BUT … you do not want all of them. You want the ones that have a minimum greater than department 50
… lead to HAVING

SOLUTION

```
SELECT      department_id, min(salary)
FROM        employees
GROUP BY    department_id
HAVING min(salary) > ( SELECT min(salary)
                       FROM        employees
                       WHERE       department_id = 50;)
```

The HAVING Clause with Subqueries

- The Oracle server executes subqueries first.
- The Oracle server returns results into the HAVING clause of the main query.



```
SELECT   department_id, MIN(salary)
FROM     employees
GROUP BY department_id
HAVING   MIN(salary) >
                        (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50);
```

Using Subqueries with the HAVING clause

PROBLEM 2:
Find the job with the lowest average salary. Display the job ID and that average salary.

SOLUTION: **#1** Find the lowest average salary for a job ID
#2 Display that job ID and that average salary

```
SELECT      job_id, AVG (salary)
FROM        employees
GROUP BY    job_id
HAVING      AVG (salary) =      (SELECT     MIN ( AVG (salary) )
                                 FROM       employees
                                 GROUP BY   job_id );
```

What Is Wrong with This Statement?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                   (SELECT   MIN(salary)
                    FROM     employees
                    GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

Error:
More than one row is returned – you cannot be equal to more than one value

When you use a GROUP BY there is an implication that there will be multiple rows returned. In this case the result of the Subquery is 7 rows returned. Each department ID in the employees table generated a minimum salary.

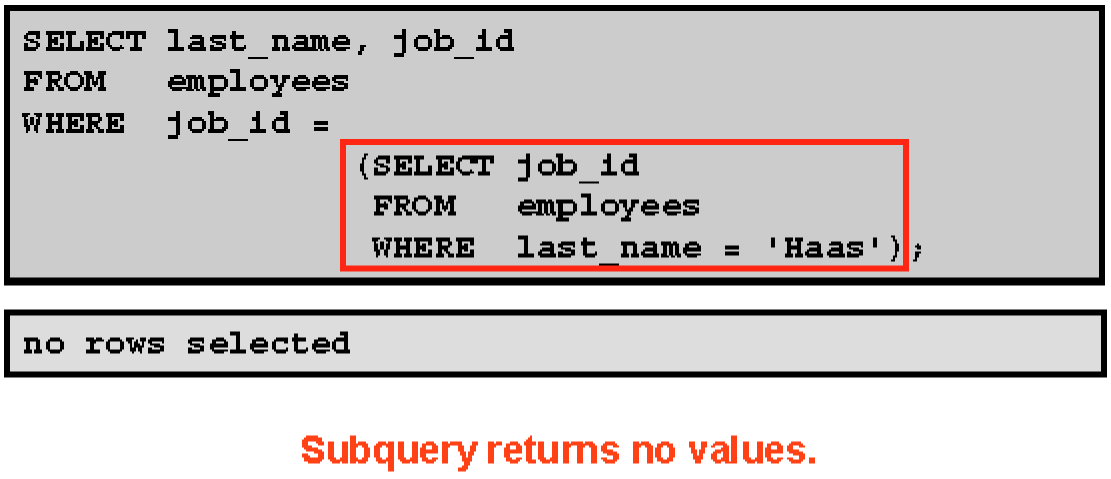
The outer query cannot be equal to seven different values.

Change to IN

```
SELECT      department_id, employee_id, last_name, salary
FROM        employees
WHERE       salary IN    (SELECT     min (salary)
                          FROM       employees
                          GROUP BY   department_id)
```

DEPARTMENT_ID	EMPLOYEE_ID	LAST_NAME	SALARY
90	101	Kochhar	17000
90	102	De Haan	17000
60	104	Ernst	6000
60	107	Lorentz	4200
50	144	Vargas	2500
80	176	Taylor	8600
	178	Grant	7000
10	200	Whalen	4400
20	202	Fay	6000
110	206	Gietz	8300

Will This Statement Return Rows?



```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');
```

```
no rows selected
```

Subquery returns no values.

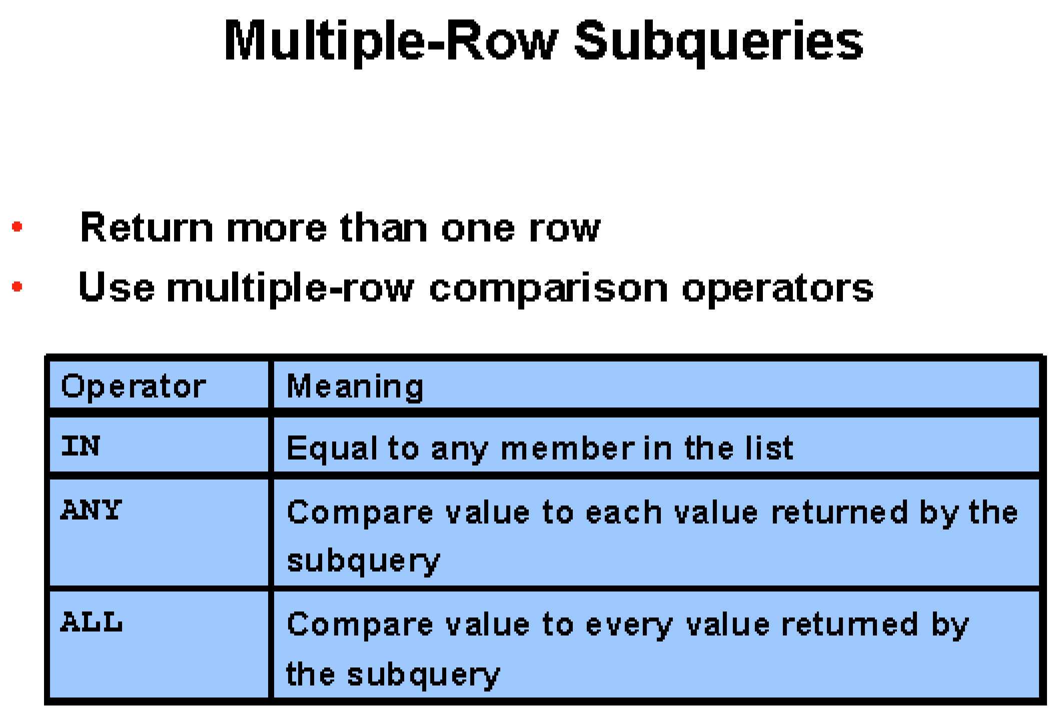
COMMON PROBLEM:

The above statement is correct. It didn't return any rows from the Subquery. (no Haas exists)
The query passes a **null** value back to the right hand condition on the WHERE clause.
There is no job ID that is equal to NULL.
Therefore, no rows are selected

SPECIAL NOTE:
If there was a job ID with a NULL value then the left side value would be NULL, and the right side value would be NULL. This means that NULL would be equal to NULL and the row would be displayed.
For the row to be displayed, the WHERE clause must evaluate to TRUE
Because a comparison of two NULL values results in a NULL (instead of a 1 or 0) the WHERE condition is not true

Getting a little harder …

7-17



Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

Multiple-Row Subqueries:

To use a Subquery that returns more than one row you need to use a Multiple-row operator

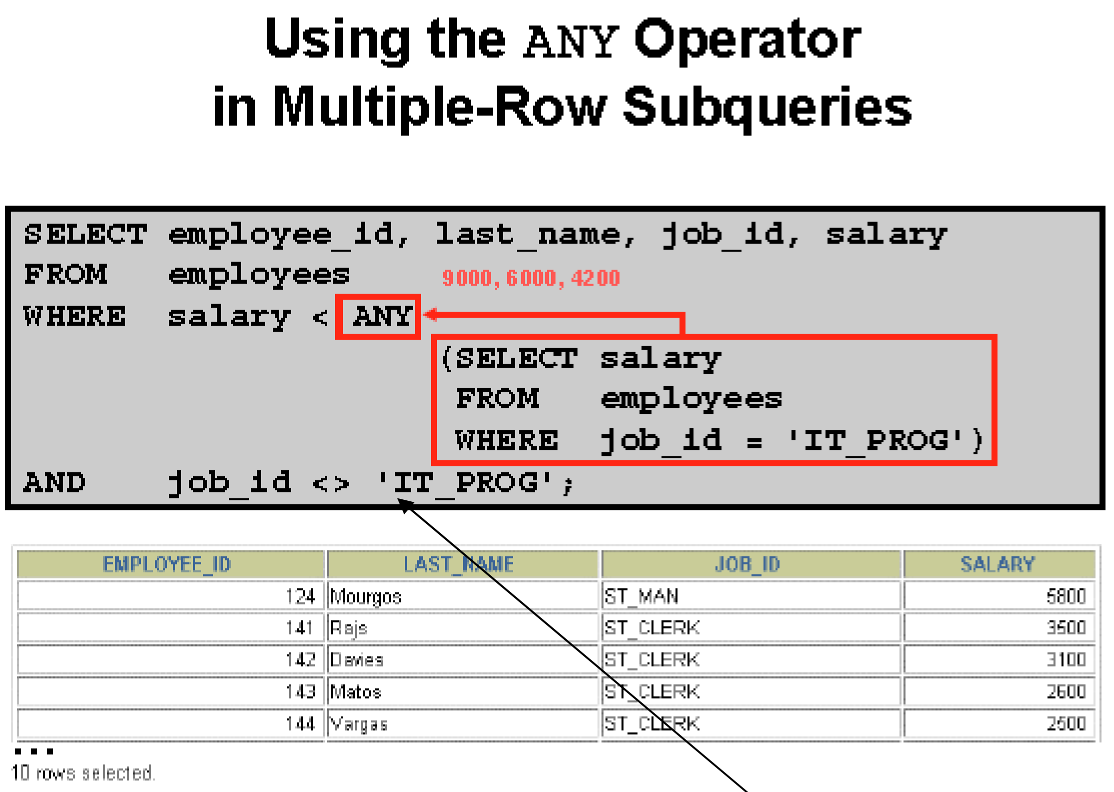
We did this before when we had a problem with the query.

We used the IN operator

```
SELECT      department_id, employee_id, last_name, salary
FROM        employees
WHERE       salary IN       (SELECT       MIN (salary)
                             FROM         employees
                             GROUP BY     department_id)
```

7-18

Problem: Display employees with a salary less than people with job_id IT_PROG



Using the ANY Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees          9000, 6000, 4200
WHERE  salary < ANY
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

...

10 rows selected.

Multiple-Row Subqueries:
ANY clause

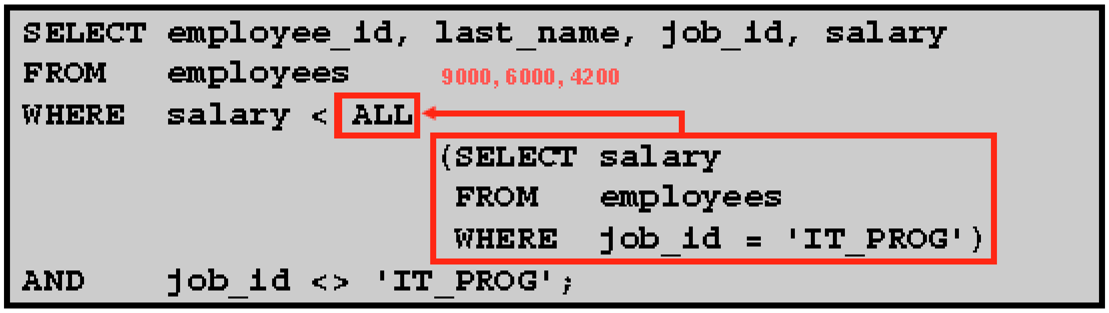
Looking at the outer query, the slide displays employees who are not IT programmers
And
whose salary is less than ANY salary that is returned by the inner Subquery

The inner Subquery sends back all the salaries for job ID equal to IT programmer.
The inner Subquery returns 3 salaries with values 9000, 6000 and 4200.

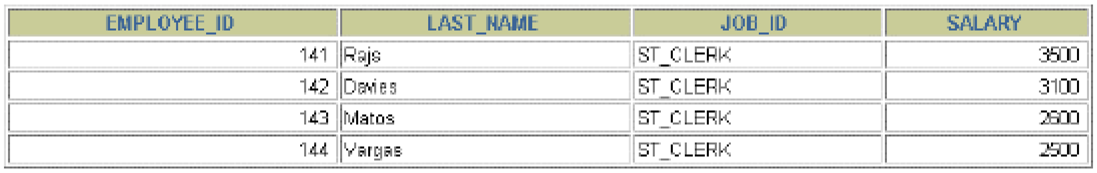
Since the outer query is looking for a salary **less than ANY** of the IT programmer salaries then it is looking for a value that is less than 4200 and less than 6000 and less than 9000. In other words, it is looking for a value less than the maximum value returned by the inner Subquery. The maximum value is $9000.
This will then return IT_PROG also unless the final line is added to the query

NOTE:
< ANY -- less than any will mean less than the maximum return
> ANY -- greater than any means more than the minimum value returned
= ANY -- equal to any is the equivalent of the **IN** operator

Using the ALL Operator in Multiple-Row Subqueries



```
SELECT employee_id, last_name, job_id, salary
FROM   employees       9000, 6000, 4200
WHERE  salary < ALL
                   (SELECT salary
                    FROM   employees
                    WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

ALL operator

The all operator compares a value to every value returned by a Subquery.

The example on the slide displays employees whose salary is less than the salaries of all the employees that have a job_id of IT_PROG
AND
whose job is not the IT_PROG

Again there are three values being returned. They are 9000, 6000 and 4200.
➔ To be less than ALL means you have to be less than 4200

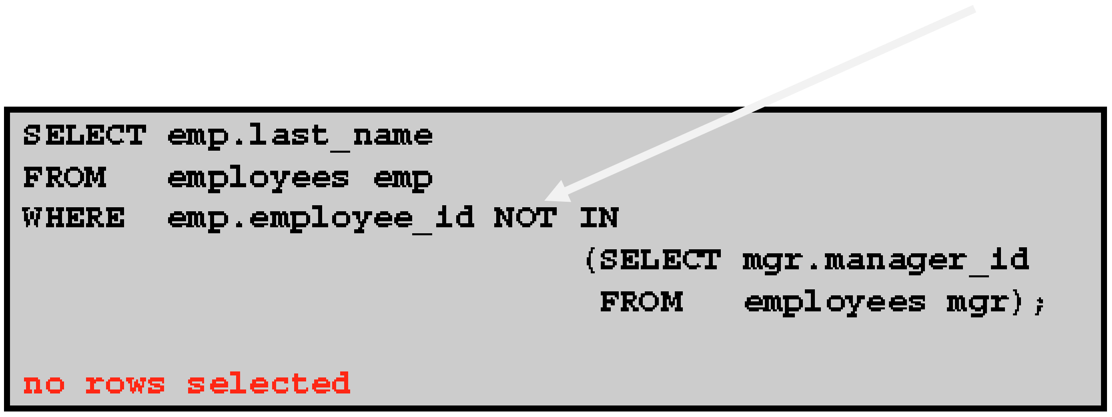
NOTE:
> ALL -- greater than all means more than the maximum
< ALL -- less than all means less than the minimum

NOTE:

The **NOT** operator can be used with any of these. Caution is recommended the use of the not operator just as it was in other programming languages.

6-17
PROBLEM: Display employees who do not havce anyone working for them. (No subordinates)

Null Values in a Subquery



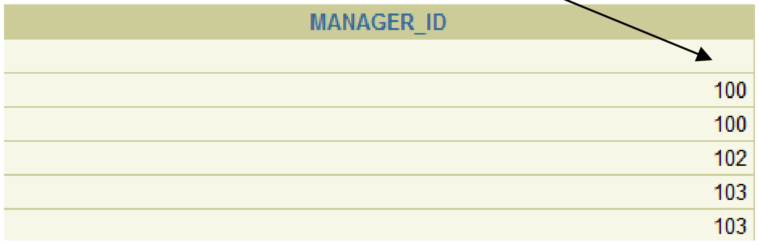
```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                            (SELECT mgr.manager_id
                             FROM   employees mgr);

no rows selected
```

SUBQUERY RETURNS NULL VALUES

The subquery
SELECT mgr.manager_id
FROM employees mgr

Will return 20 rows, but one of the rows is NULL



MANAGER_ID
100
100
102
103
103

NOTE: On next page

NOTE: NOT IN

One of the condition is a NULL value. The entire query returns no rows.
The problem is the NOT IN. The NOT IN is equivalent to <>ALL

All conditions that compare a NULL value returns a NULL

NOTE: ***IN works with NULLS***

```
SELECT last_name
FROM        employees emp
WHERE       emp.employee_id   IN
                                        (SELECT     mgr.manager_id
                                        FROM        employees mgr);
```

IN is equivalent to =ANY

NOTE:
Could have added a WHERE clause in the Subquery → WHERE manager_id is NOT NULL

```
SELECT last_name
FROM        employees emp
WHERE       emp.employee_id   IN
                                        (SELECT     mgr.manager_id
                                        FROM        employees mgr);
                                        WHERE       manager_id is NOT NULL)
```

ASIDE:
Did we need the ALIAS table names?
No, it was done for readability

PRACTICE QUESTION

Prompt the user for the employee last name. The query will return last name and hire date of any employee in the same department as the name supplied. Do not include the employee supplied.

What is the INNER query?

```
SELECT      department_id
FROM        employees
WHERE        last_name = ‘&Name’
```

Enter ZLOTKEY and it will find nothing. Should use function UPPER

```
SELECT      department_id
FROM        employees
WHERE       UPPER(last_name) = UPPER('&Name')
```

Now do the outer query

```
SELECT      last_name, department_id
FROM        employees
WHERE       department_id = (   SELECT      department_id
                                FROM        employees
                                WHERE       UPPER(last_name) = UPPER('&Name') )
```

Now eliminate the name entered

```
SELECT      last_name, department_id
FROM        employees
WHERE       department_id = (   SELECT      department_id
                         FROM        employees
                         WHERE       UPPER(last_name) = UPPER('&&Name') )
AND         UPPER (last_name) < > UPPER ('&Name');
```

UNDEFINE NAME;

Multiple Column Sub Query

A multiple-column subquery returns more than one column to the outer query and can be listed in the outer query's FROM, WHERE, or HAVING clause. For example, the below query shows the employee or employees in each department whose current salary is the lowest (or minimum) salary in the department.

```
SELECT      last_name, department_id, salary
FROM        employees
WHERE       (department_id, salary) IN (SELECT      department_id, min(salary)
                                        FROM        employees
                                        GROUP BY    department_id)
ORDER BY  department_id
```

The sub query returns the following:

```
DEPARTMENT_ID MIN(SALARY)
------------- -----------
           10        4400
           20        6000
           50        2500
           60        4200
           80        8600
           90       17000
          110        8300
                     7000
 8 rows selected
```

The full query returns

```
LAST_NAME                 DEPARTMENT_ID     SALARY
------------------------- ------------- ----------
Whalen                               10       4400
Fay                                  20       6000
Vargas                               50       2500
Lorentz                              60       4200
Taylor                               80       8600
Kochhar                              90      17000
De Haan                              90      17000
Gietz                               110       8300

 8 rows selected
```

NOTE: In department 90 is 2 people with the same minimum. Since both the sub and the full query returned 8 rows, then there must be a row missing in the full query.

➔ The NULL department did not show.

How would you fix this? Assuming the user wants to show the results where there is no department