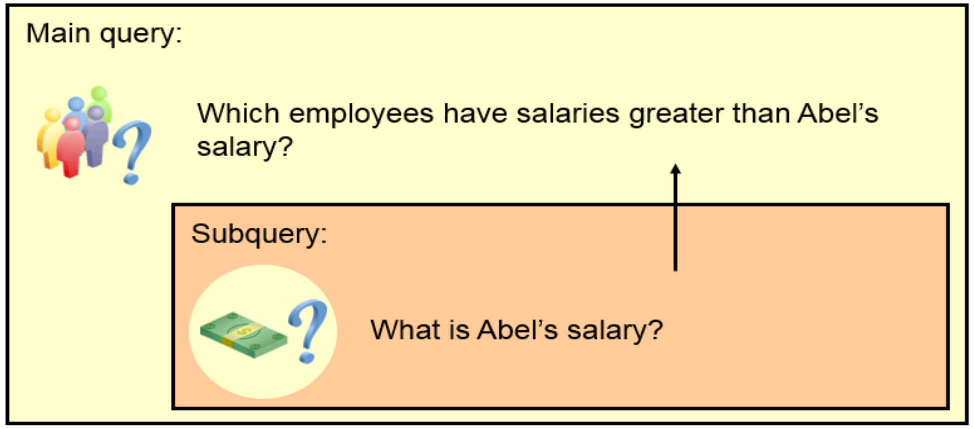
Using Subqueries to Solve Queries

Using a Subquery to Solve a Problem

Who has a salary greater than Abel's?



Subquery Syntax

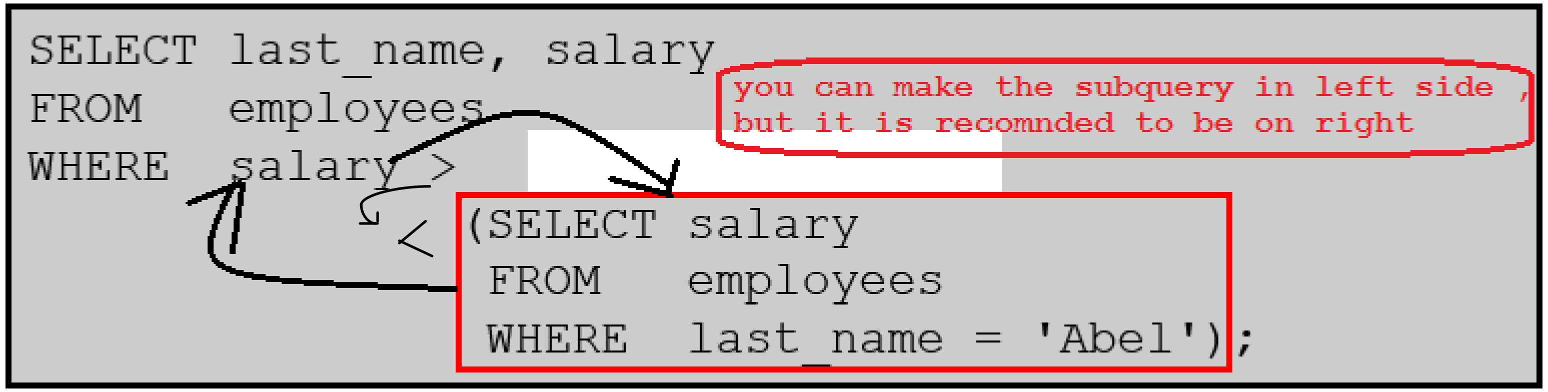
- The subquery (inner query) executes *before* the main query (outer query).
- The result of the subquery is used by the main query.

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

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

Rules for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition for readability. (However, the subquery can appear on either side of the comparison operator.)
- Use single-row operators with single-row subqueries and multiple-row operators with multiple-row subqueries.



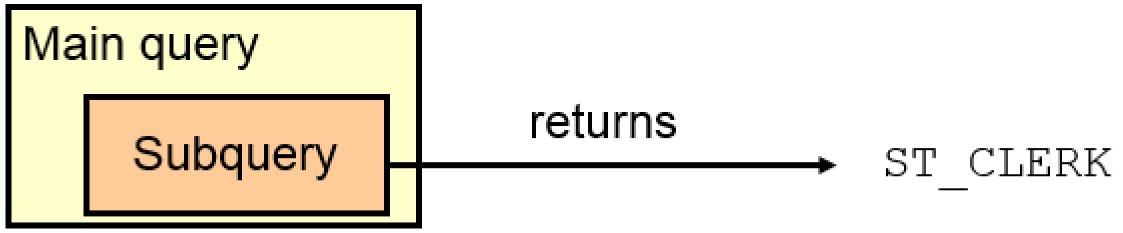
```
SELECT last_name, salary
FROM   employees
WHERE  salary >
                 (SELECT salary
                  FROM   employees
                  WHERE  last_name = 'Abel');
```

you can make the subquery in left side , but it is recomnded to be on right
<

```
select EMPLOYEE_ID,first_name, last_name, salary
FROM
EMPLOYEES
WHERE  ( SELECT SALARY FROM EMPLOYEES WHERE LAST_NAME='Abel' )<SALARY
```

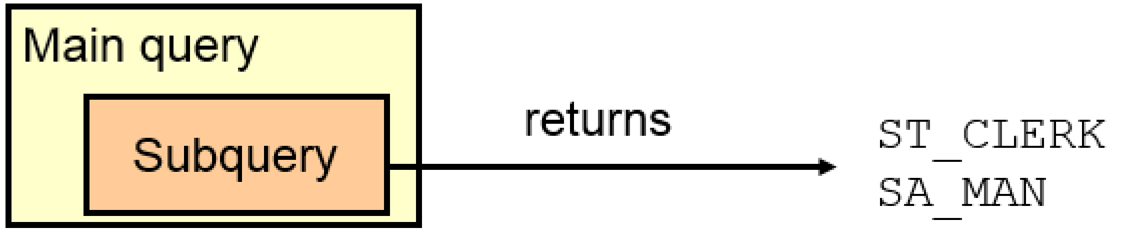
Types of Subqueries

- Single-row subquery



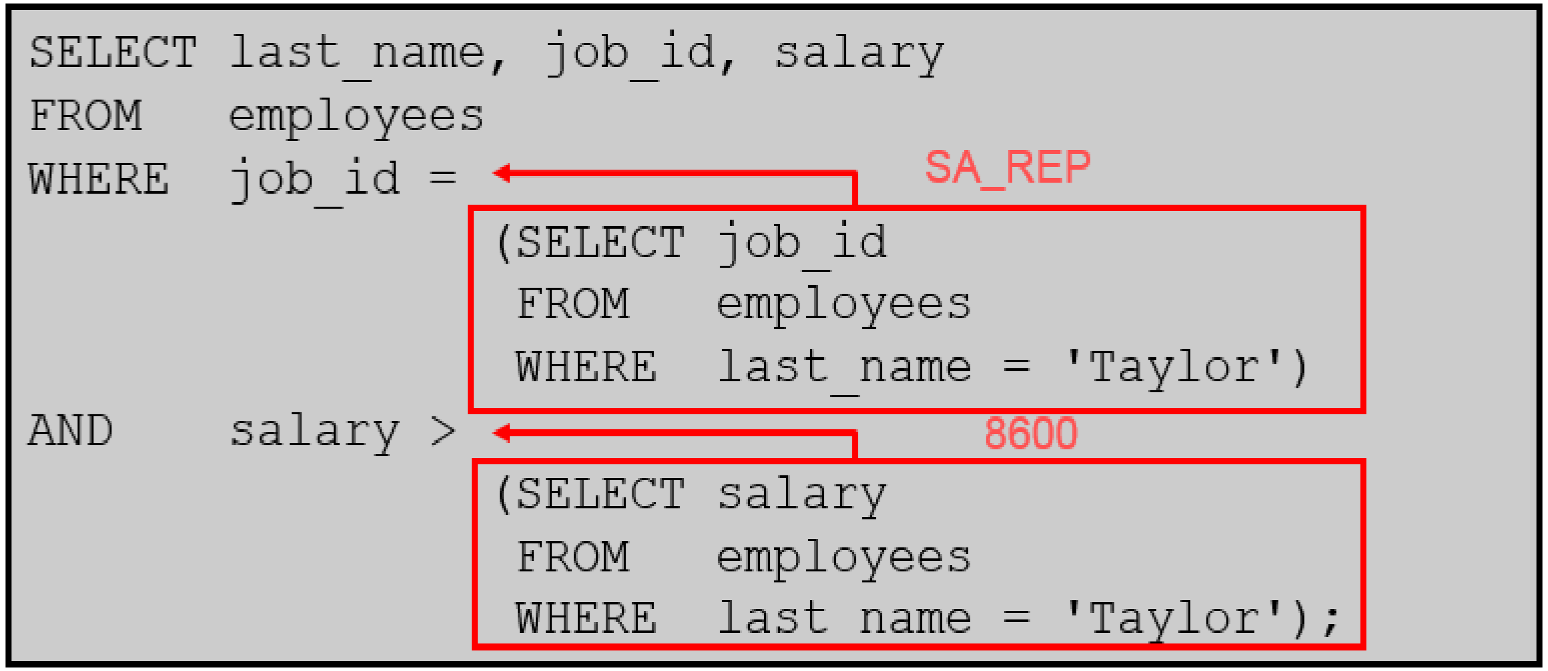
Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

- Multiple-row subquery



Use IN, ALL, or ANY

Executing Single-Row Subqueries

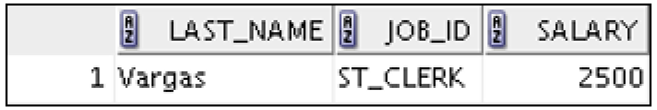


```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                 (SELECT job_id
                  FROM   employees
                  WHERE  last_name = 'Taylor')
AND    salary >
                 (SELECT salary
                  FROM   employees
                  WHERE  last name = 'Taylor');
```

Using Group Functions in a Subquery

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

2500



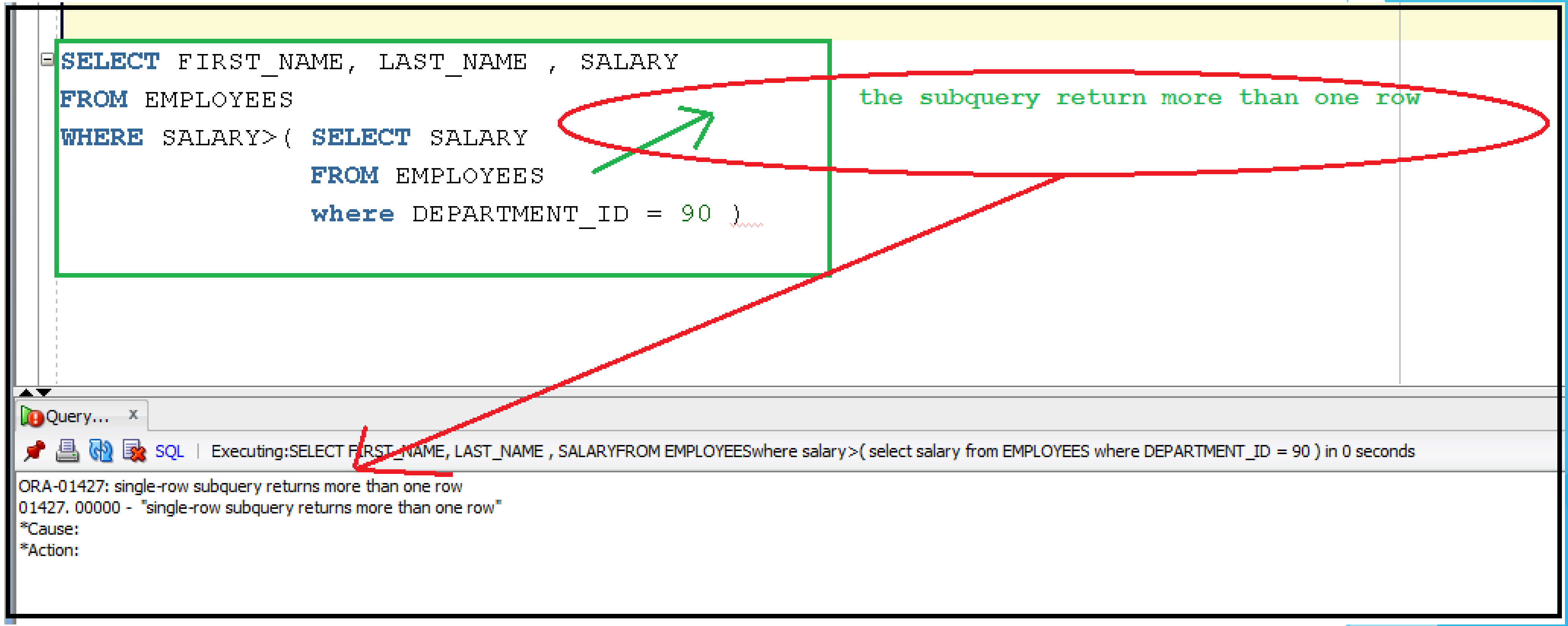
	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

HAVING Clause with Subqueries

- The Oracle server executes the 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                    2500
HAVING    MIN(salary)  >
                        (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50);
```

What Is Wrong with This Statement?

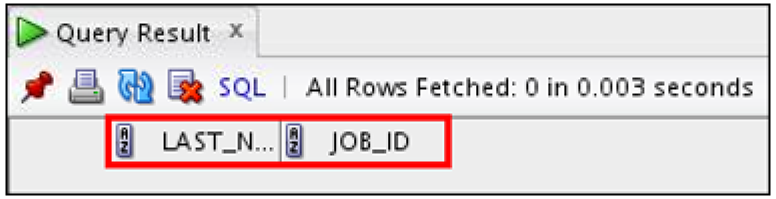


```
SELECT FIRST_NAME, LAST_NAME , SALARY
FROM EMPLOYEES
WHERE SALARY>( SELECT SALARY
               FROM EMPLOYEES
               where DEPARTMENT_ID = 90 )
```

```
Executing:SELECT FIRST_NAME, LAST_NAME , SALARYFROM EMPLOYEESwhere salary>( select salary from EMPLOYEES where DEPARTMENT_ID = 90 ) in 0 seconds
ORA-01427: single-row subquery returns more than one row
01427. 00000 -  "single-row subquery returns more than one row"
*Cause:
*Action:
```

No Rows Returned by the Inner Query

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



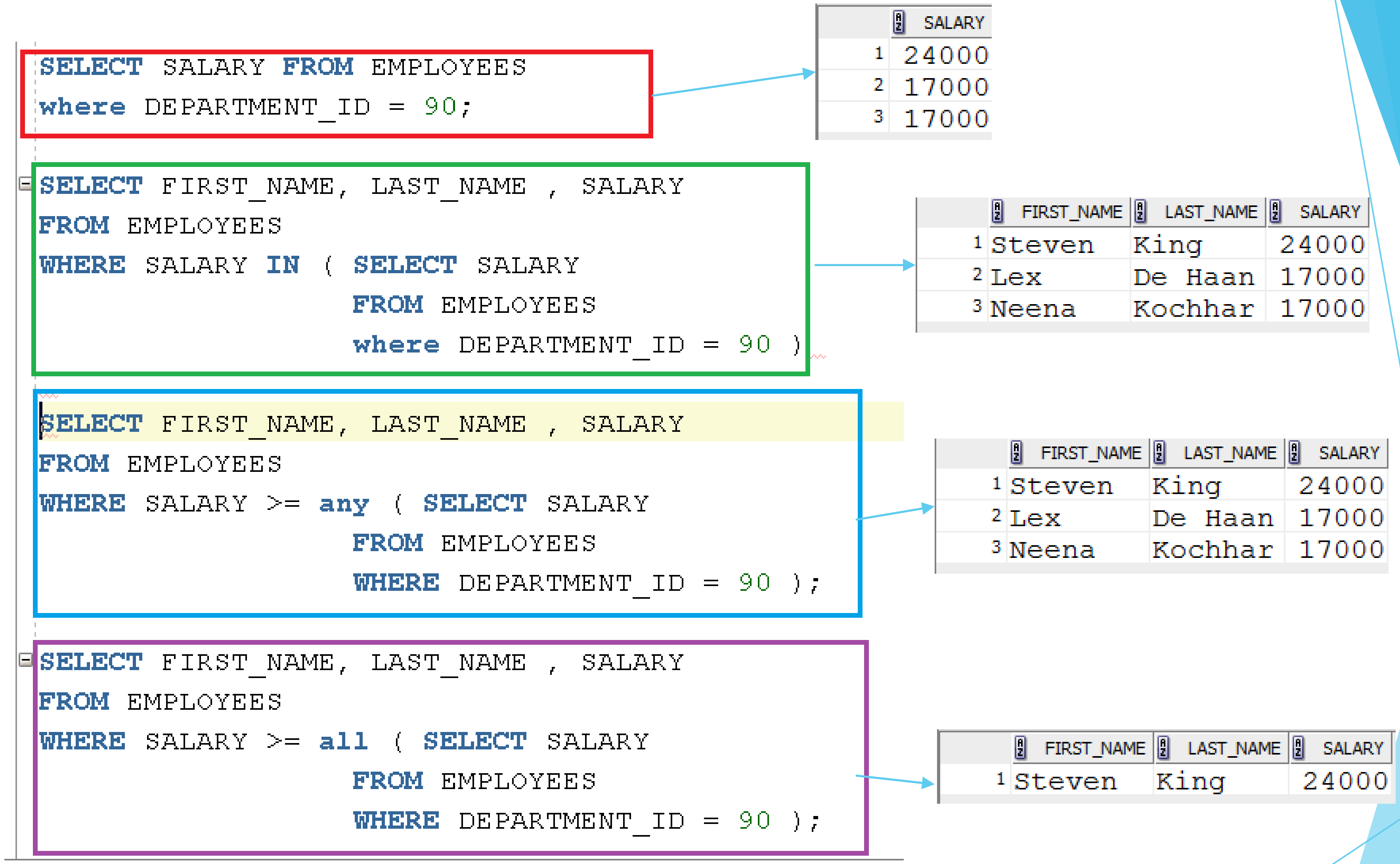
```
All Rows Fetched: 0 in 0.003 seconds
LAST_N...  JOB_ID
```

Subquery returns no rows because there is no employee named “Haas.”

Multiple-Row Subqueries

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

Operator	Meaning
`IN`	Equal to any member in the list
`ANY`	Must be preceded by =, !=, >, <, <=, >=. Returns `TRUE` if at least one element exists in the result-set of the Subquery for which the relation is `TRUE`.
`ALL`	Must be preceded by =, !=, >, <, <=, >=. Returns `TRUE` if the relation is `TRUE` for all elements in the result set of the Subquery.



```sql
SELECT SALARY FROM EMPLOYEES
where DEPARTMENT_ID = 90;
```

	SALARY
1	24000
2	17000
3	17000

```sql
SELECT FIRST_NAME, LAST_NAME , SALARY
FROM EMPLOYEES
WHERE SALARY IN ( SELECT SALARY
                  FROM EMPLOYEES
                  where DEPARTMENT_ID = 90 )
```

	FIRST_NAME	LAST_NAME	SALARY
1	Steven	King	24000
2	Lex	De Haan	17000
3	Neena	Kochhar	17000

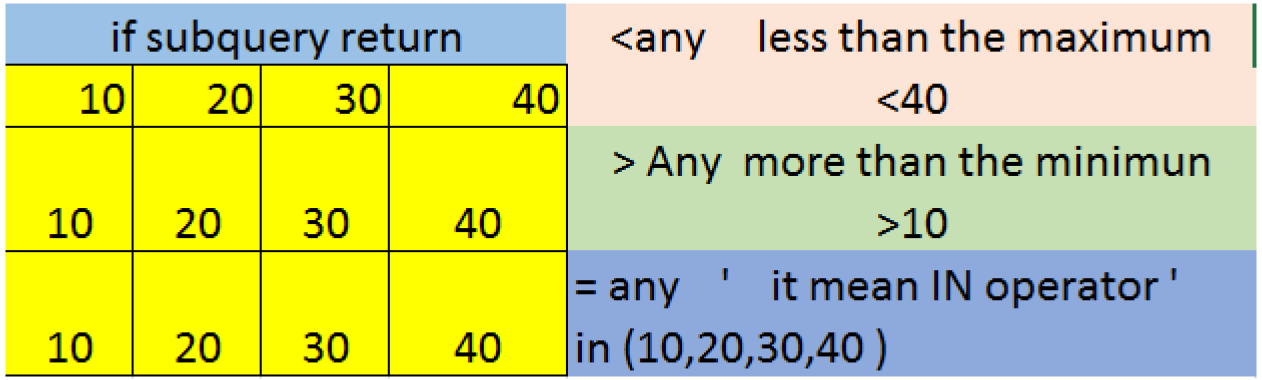
```sql
SELECT FIRST_NAME, LAST_NAME , SALARY
FROM EMPLOYEES
WHERE SALARY >= any ( SELECT SALARY
                  FROM EMPLOYEES
                  WHERE DEPARTMENT_ID = 90 );
```

	FIRST_NAME	LAST_NAME	SALARY
1	Steven	King	24000
2	Lex	De Haan	17000
3	Neena	Kochhar	17000

```sql
SELECT FIRST_NAME, LAST_NAME , SALARY
FROM EMPLOYEES
WHERE SALARY >= all ( SELECT SALARY
                  FROM EMPLOYEES
                  WHERE DEPARTMENT_ID = 90 );
```

	FIRST_NAME	LAST_NAME	SALARY
1	Steven	King	24000

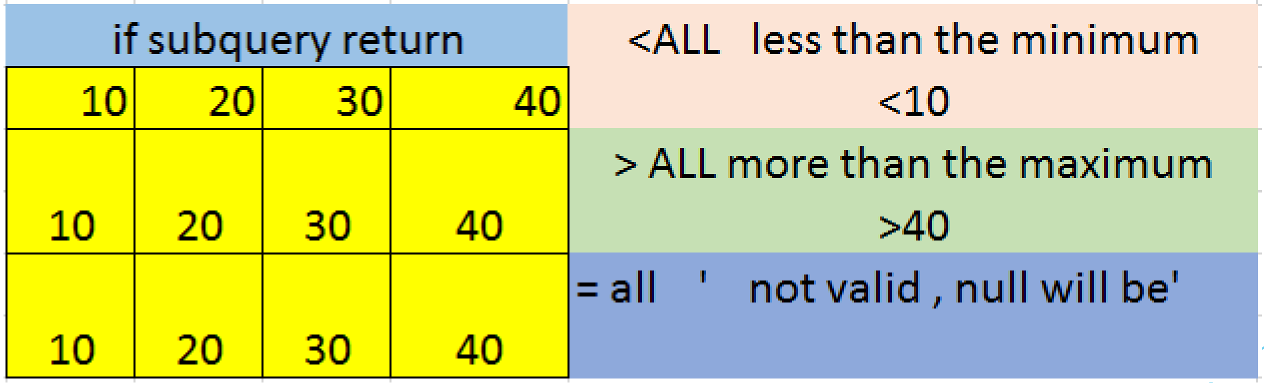
- <ANY means less than the maximum.
- >ANY means more than the minimum.
- =ANY is equivalent to IN.



if subquery return				
				<any less than the maximum
10	20	30	40	<40
				> Any more than the minimun
10	20	30	40	>10
				= any ' it mean IN operator '
10	20	30	40	in (10,20,30,40)

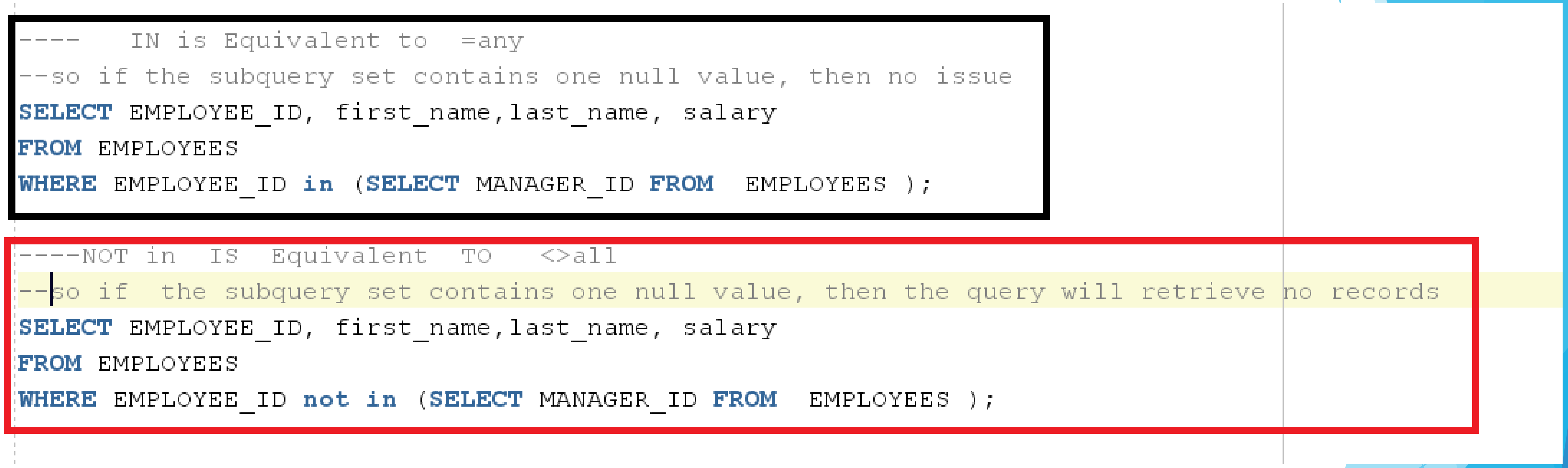
>ALL means more than the maximum and <ALL means less than the minimum.

The NOT operator can be used with IN, ANY, and ALL operators.



if subquery return				
				<ALL less than the minimum
10	20	30	40	<10
				> ALL more than the maximum
10	20	30	40	>40
				= all ' not valid , null will be'
10	20	30	40	

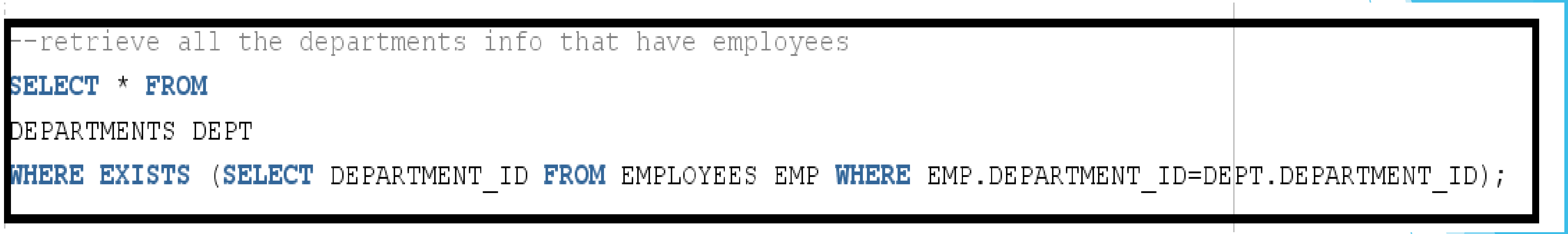
Do not use NOT IN when the subquery return some null values



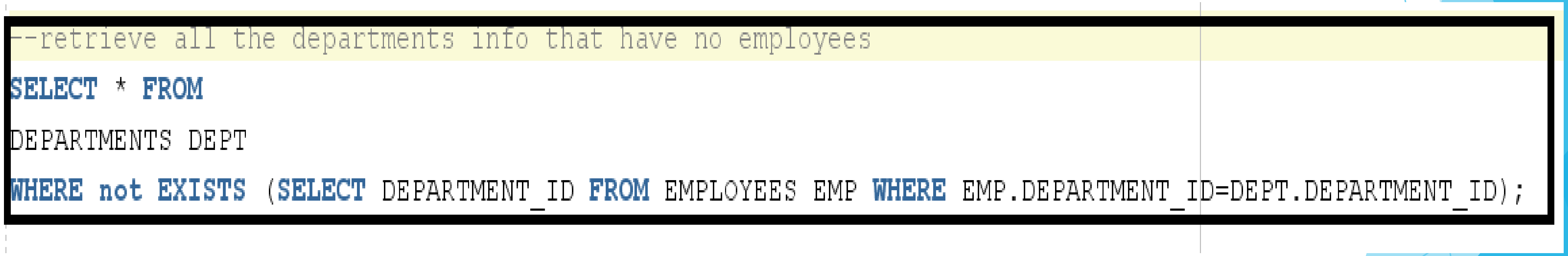
```
----   IN is Equivalent to  =any
--so if the subquery set contains one null value, then no issue
SELECT EMPLOYEE_ID, first_name,last_name, salary
FROM EMPLOYEES
WHERE EMPLOYEE_ID in (SELECT MANAGER_ID FROM  EMPLOYEES );
```

```
----NOT in  IS  Equivalent  TO   <>all
--so if  the subquery set contains one null value, then the query will retrieve no records
SELECT EMPLOYEE_ID, first_name,last_name, salary
FROM EMPLOYEES
WHERE EMPLOYEE_ID not in (SELECT MANAGER_ID FROM  EMPLOYEES );
```

Exists / not Exists



```
--retrieve all the departments info that have employees
SELECT * FROM
DEPARTMENTS DEPT
WHERE EXISTS (SELECT DEPARTMENT_ID FROM EMPLOYEES EMP WHERE EMP.DEPARTMENT_ID=DEPT.DEPARTMENT_ID);
```



```
--retrieve all the departments info that have no employees
SELECT * FROM
DEPARTMENTS DEPT
WHERE not EXISTS (SELECT DEPARTMENT_ID FROM EMPLOYEES EMP WHERE EMP.DEPARTMENT_ID=DEPT.DEPARTMENT_ID);
```

Note: always use table alias in exists/ not exists

Thank You