7

Using Subqueries to Solve Queries

Objectives

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

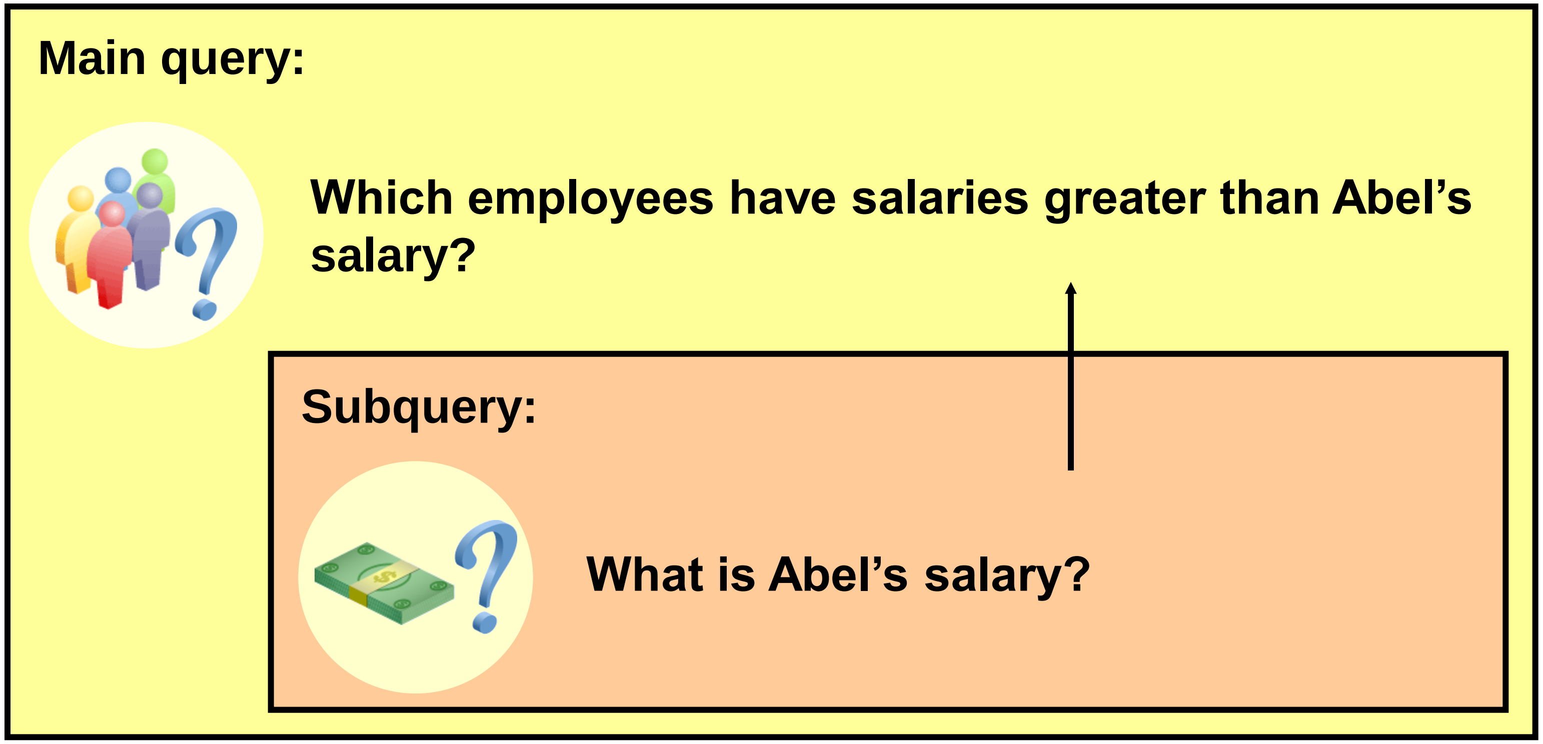
- Define subqueries
- Describe the types of problems that the subqueries can solve
- List the types of subqueries
- Write single-row and multiple-row subqueries

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
  - Group functions in a subquery
  - `HAVING` clause with subqueries
- Multiple-row subqueries
  - Use `ALL` or `ANY` operator
- Null values in a subquery

Using a Subquery to Solve a Problem

Who has a salary greater than Abel’s?



Main query:

Which employees have salaries greater than Abel’s salary?

Subquery:

What is Abel’s salary?

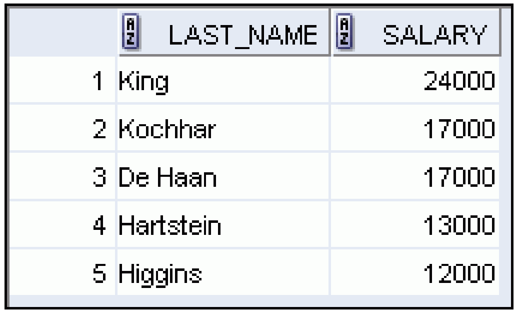
Subquery Syntax

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

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

Using a Subquery

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



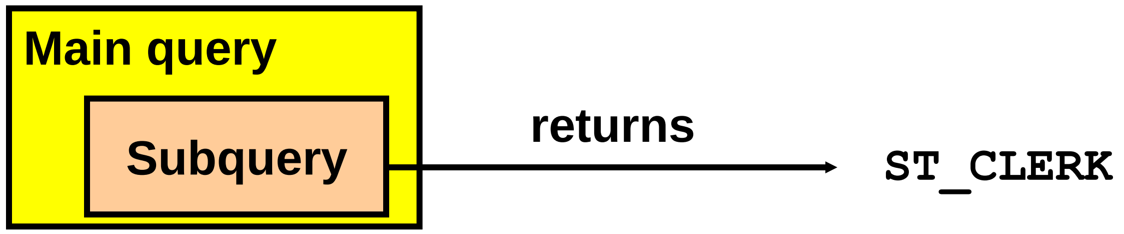
	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12000

Guidelines for Using Subqueries

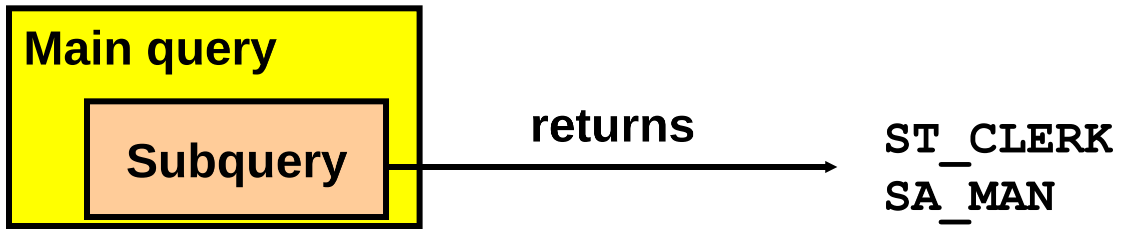
- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition for readability (However, the subquery can appear on either side of the comparison operator.).
- Use single-row operators with single-row subqueries and multiple-row operators with multiple-row subqueries.

Types of Subqueries

- Single-row subquery



- Multiple-row subquery



Lesson Agenda

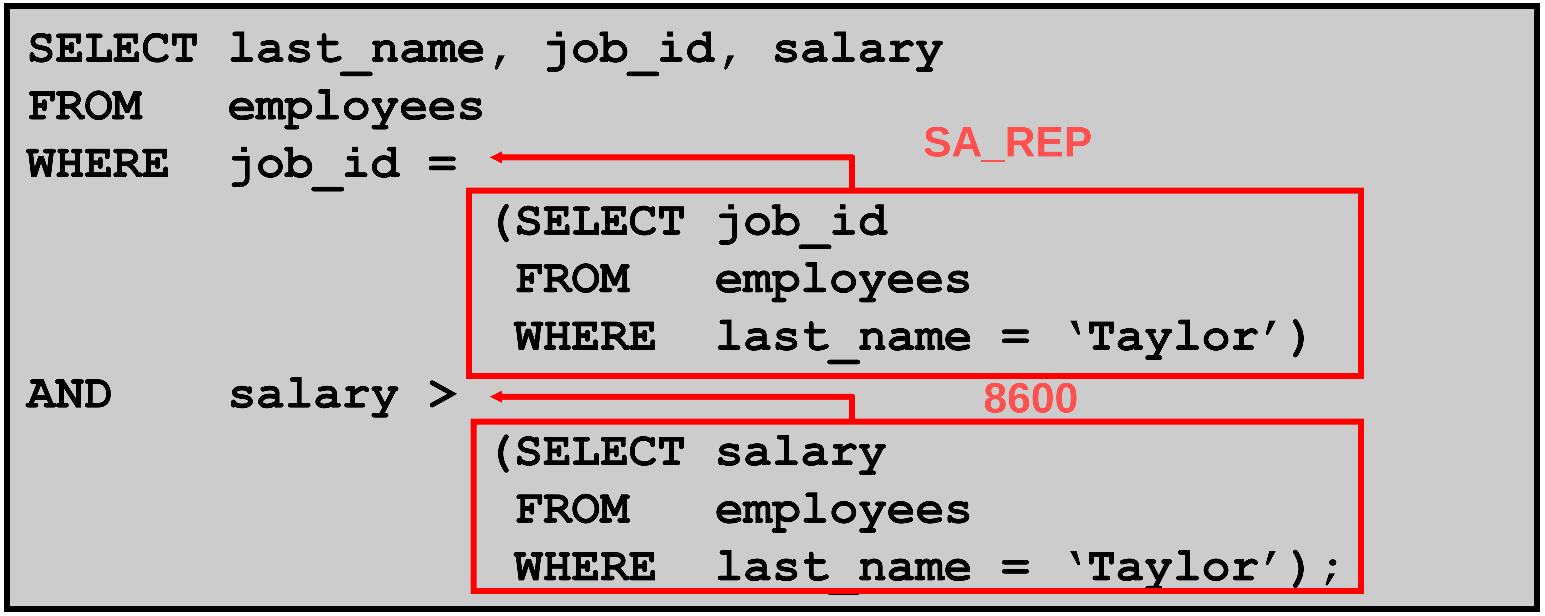
- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
  - Group functions in a subquery
  - `HAVING` clause with subqueries
- Multiple-row subqueries
  - Use `ALL` or `ANY` operator
- Null values in a subquery

Single-Row Subqueries

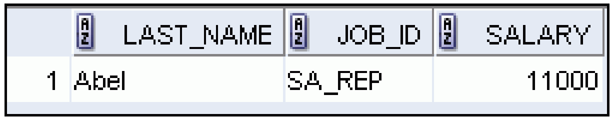
- Return only one row
- Use single-row comparison operators

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

Executing Single-Row Subqueries

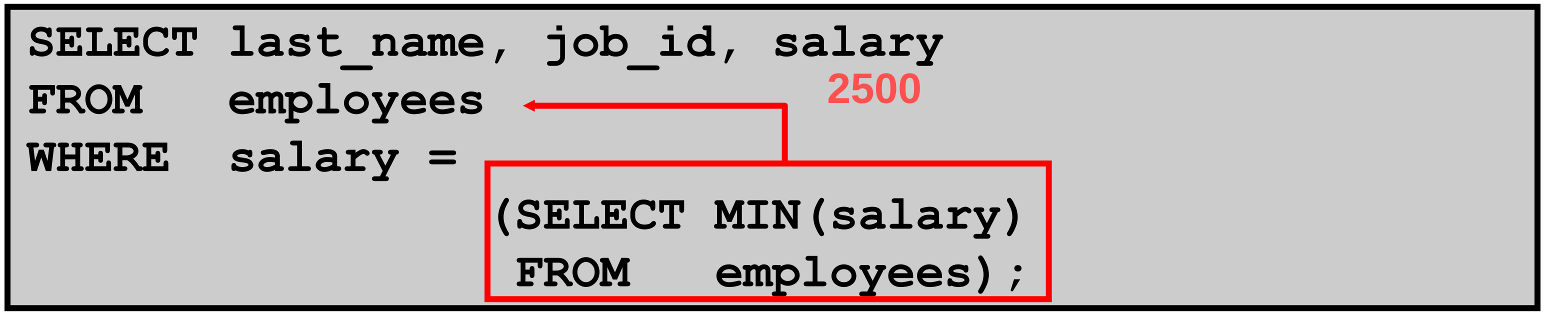


```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Taylor')
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Taylor');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

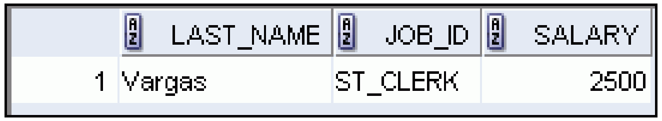


	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000

Using Group Functions in a Subquery



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

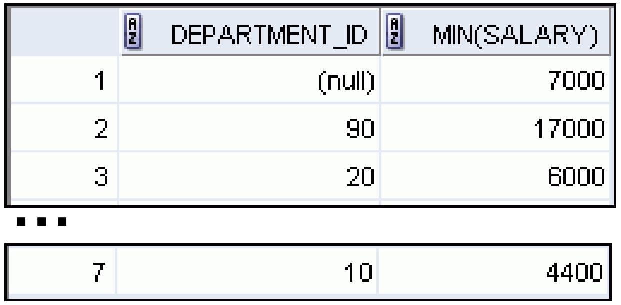


	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

The HAVING Clause with Subqueries

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

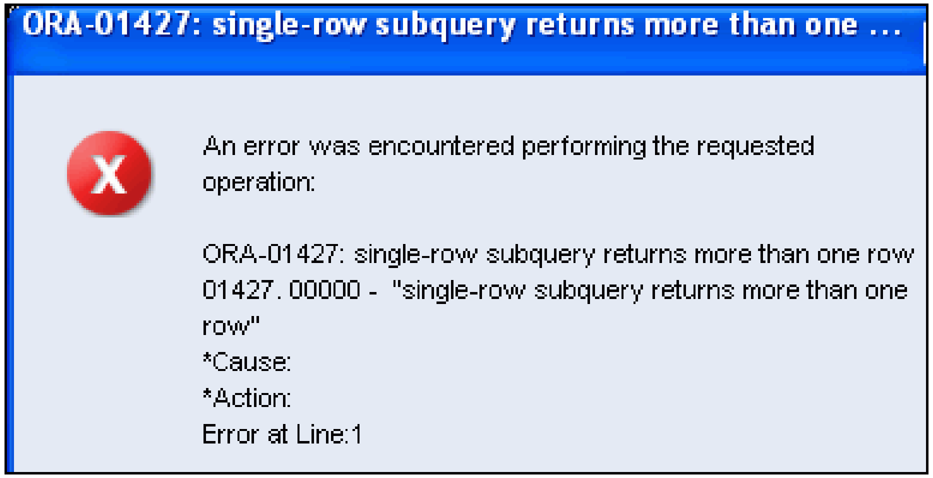
```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY department_id                          2500
HAVING    MIN(salary) >
                       (SELECT MIN(salary)
                        FROM   employees
                        WHERE  department_id = 50);
```



	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	90	17000
3	20	6000
...		
7	10	4400

What Is Wrong with This Statement?

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



ORA-01427: single-row subquery returns more than one ...

An error was encountered performing the requested operation:

ORA-01427: single-row subquery returns more than one row
01427. 00000 - "single-row subquery returns more than one row"
*Cause:
*Action:
Error at Line:1

Single-row operator with multiple-row subquery

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');
```

```
0 rows selected
```

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

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
  - Group functions in a subquery
  - `HAVING` clause with subqueries
- Multiple-row subqueries
  - Use `ALL` or `ANY` operator
- Null values in a subquery

Multiple-Row Subqueries

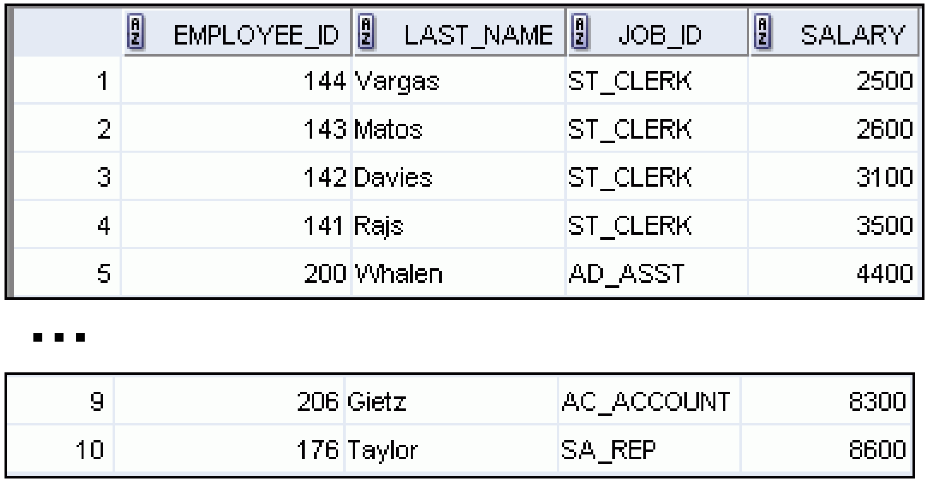
- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
`IN`	Equal to any member in the list
`ANY`	Must be preceded by `=`, `!=`, `>`, `<`, `<=`, `>=`. Compares a value to each value in a list or returned by a query. Evaluates to `FALSE` if the query returns no rows.
`ALL`	Must be preceded by `=`, `!=`, `>`, `<`, `<=`, `>=`. Compares a value to every value in a list or returned by a query. Evaluates to `TRUE` if the query returns no rows.

Using the ANY Operator in Multiple-Row Subqueries

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

9000, 6000, 4200



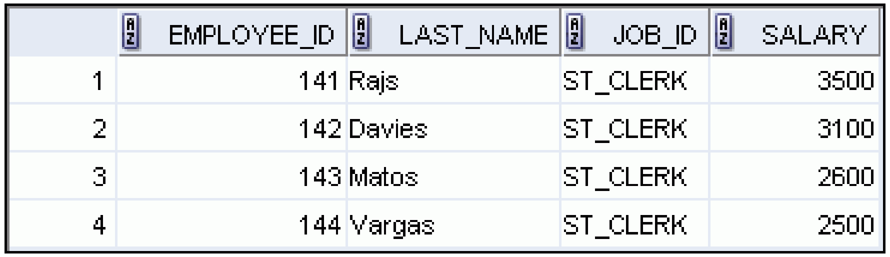
	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400

…

9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

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
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

Lesson Agenda

- Subquery: Types, syntax, and guidelines
- Single-row subqueries:
  - Group functions in a subquery
  - `HAVING` clause with subqueries
- Multiple-row subqueries
  - Use `ALL` or `ANY` operator
- Null values in a subquery

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);
```

```
0 rows selected
```

Summary

In this lesson, you should have learned how to:

- Identify when a subquery can help solve a problem
- Write subqueries when a query is based on unknown values

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