

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 subqueries can solve
 - List the types of subqueries
 - Write single-row and multiple-row subqueries

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



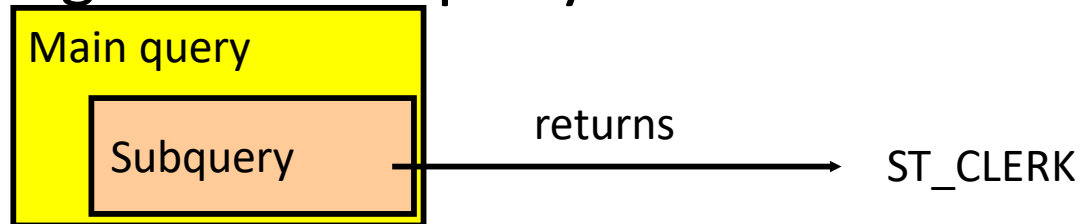
LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hartstein	13000
Higgins	12000

Guidelines for Using Subqueries

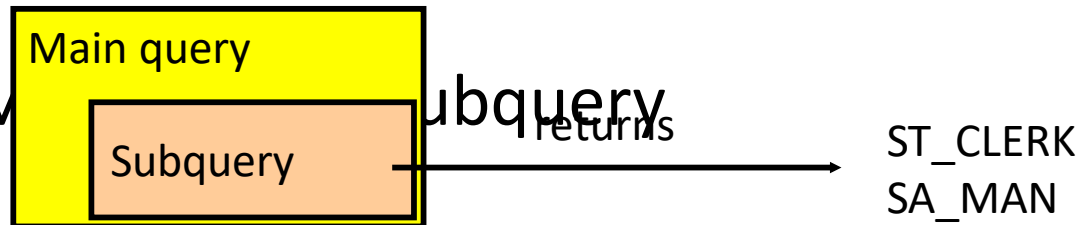
- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition.
- The `ORDER BY` clause in the subquery is not needed unless you are performing Top-N analysis.
- Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row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

Types of Subqueries

- Single-row subquery



- Multiple-row 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

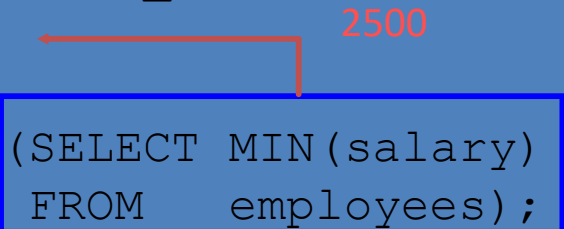
Executing Single-Row Subqueries

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =  ST_CLERK
AND    salary >  2600
      (SELECT job_id
       FROM   employees
       WHERE  employee_id = 141)
      (SELECT salary
       FROM   employees
       WHERE  employee_id = 143);
```

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

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
Vargas	ST_CLERK	2500

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);
```

The diagram illustrates the execution of the SQL query. A red arrow points from the subquery result, which is the value 2500, to the comparison operator > in the HAVING clause. This indicates that the Oracle server first evaluates the subquery to find the minimum salary for department_id 50, and then uses this result to filter the main query's results.

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
```

Single-row operator with multiple-row subquery

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.

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

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

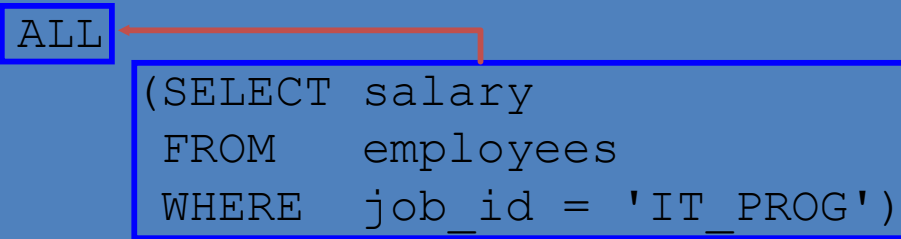
EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

10 rows selected.

Using the ALL Operator in Multiple-Row Subqueries

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

9000, 6000, 4200



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

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

Summary

- In this lesson, you should have learned how to:

- Identify when a subquery can help solve a question

- Write subqueries when a query is based on

```
SELECT select_list
FROM table
WHERE expr operator
```

unknown values

```
(SELECT select_list
FROM table);
```

Practice 6: Overview

- This practice covers the following topics:
 - Creating subqueries to query values based on unknown criteria
 - Using subqueries to find out which values exist in one set of data and not in another