

Subqueries

Objectives

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

- Define what subqueries are
- Describe the types of problems that subqueries can solve
- Identify and list the different types of subqueries
- Write both single-row and multiple-row subqueries

Using a Subquery to Solve a Problem

Who has a salary greater than Ozel's?

Main query:



Which employees have salaries greater than Ozel's salary?

Subquery:

What is Ozel's salary?



Subquery Syntax

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

- The subquery executes *before* the main query.
- The result of the subquery is used by the main query.

Using a Subquery

The screenshot displays a database management interface. On the left, a tree view shows the database structure, including tables like EMP1, EMPLOYEES, FURNITORI, JOB_HISTORY, JOBS, LOCATIONS, PJESE, PROJEKTI, PUN, REGIONS, and SHITJET. The EMPLOYEES table is expanded, showing columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, and DEPARTMENT_ID.

The SQL editor on the right contains the following query:

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

Below the SQL editor, a "Query Result" window shows the results of the query. It indicates that all 9 rows were fetched in 0.006 seconds. The results are displayed in a table with two columns: LAST_NAME and SALARY.

	LAST_NAME	SALARY
1	King	26000
2	Kochhar	17000
3	De Haan	17000
4	Greenberg	12008
5	Russell	14000
6	Partners	13500
7	Errazuriz	12000
8	Hartstein	13000
9	Higgins	12008

Guidelines for Using Subqueries

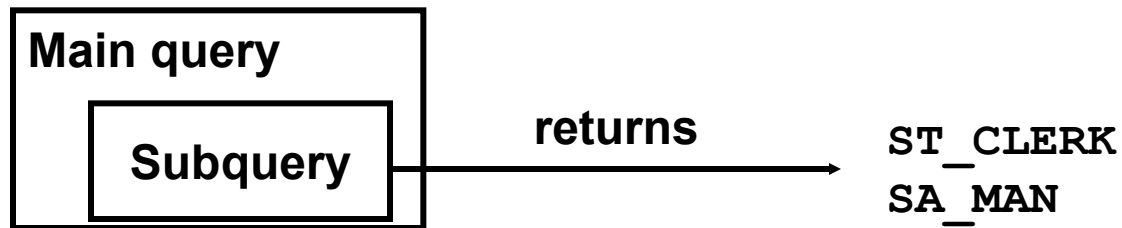
- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition for better readability (although a subquery can appear on either side of the comparison operator).
- Use single-row operators with single-row subqueries, and multiple-row operators with multiple-row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

Types of Subqueries

- Single-row subquery



- Multiple-row subquery

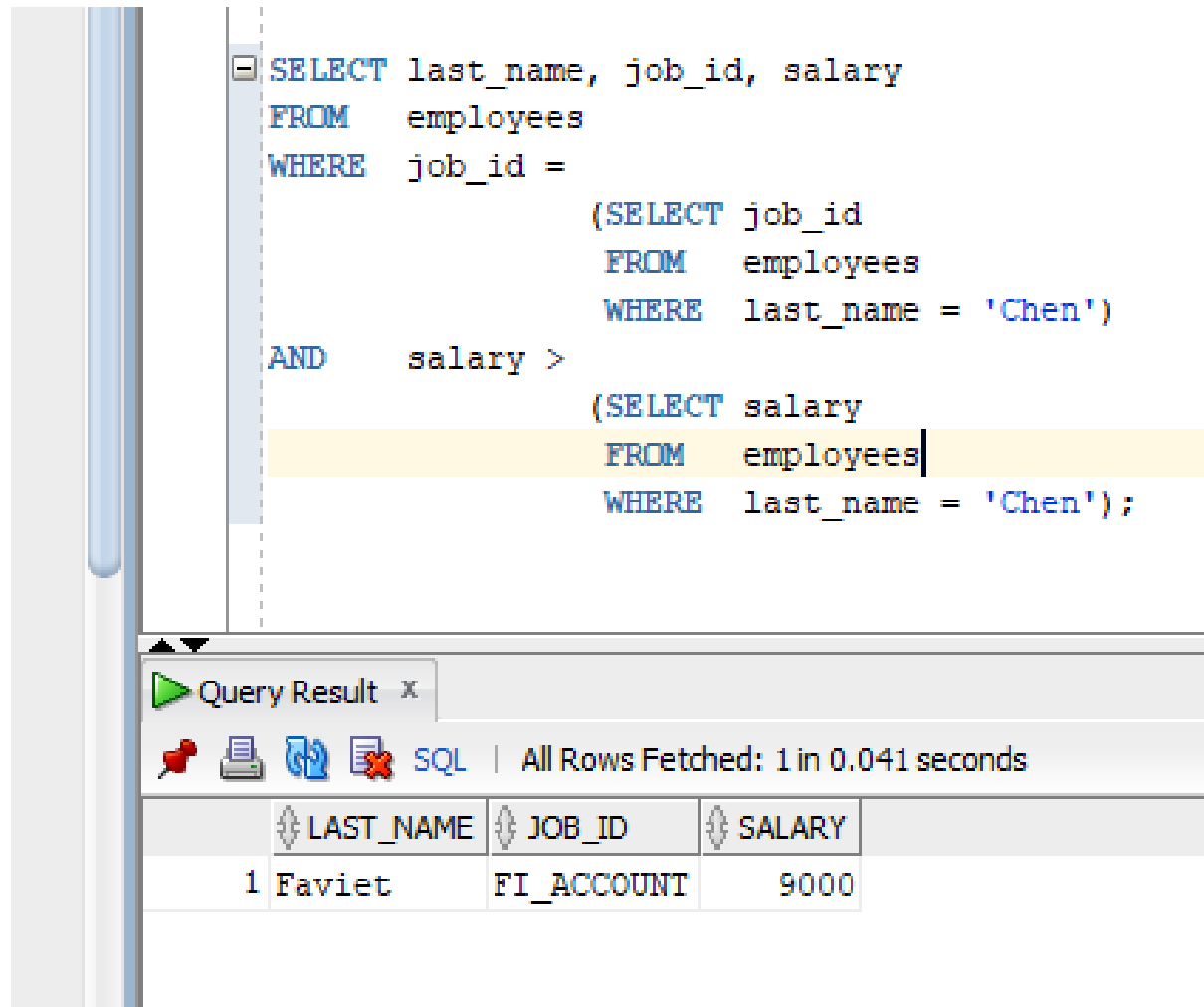


Single-Row Subqueries

- Return only one row and 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



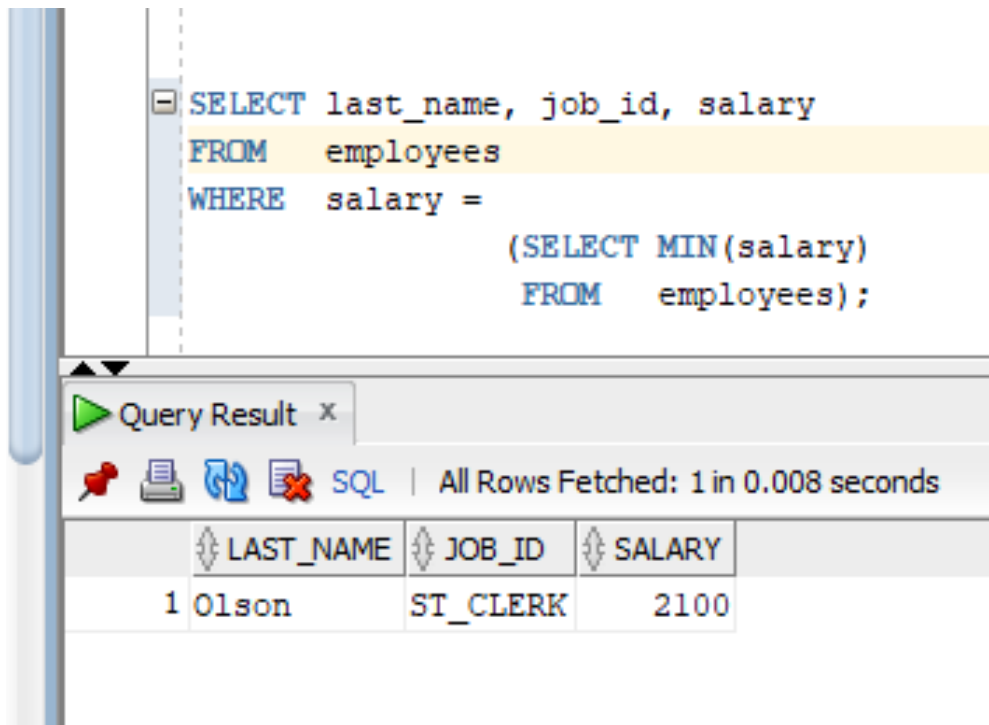
The screenshot displays a SQL query in a text editor, which is then executed in a query window. The query selects employee details based on two subqueries: one for job_id and one for salary, both filtered by last_name = 'Chen'.

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

The query result window shows the following data:

	LAST_NAME	JOB_ID	SALARY
1	Faviet	FI_ACCOUNT	9000

Using Group Functions in a Subquery



The screenshot shows a SQL query editor window with a query that finds the employee with the lowest salary. The query is as follows:

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

Below the query editor, the 'Query Result' window is open, displaying the results of the query. It shows that one row was fetched in 0.008 seconds. The results are presented in a table with three columns: LAST_NAME, JOB_ID, and SALARY.

	LAST_NAME	JOB_ID	SALARY
1	Olson	ST_CLERK	2100

The use of HAVING with Subqueries

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

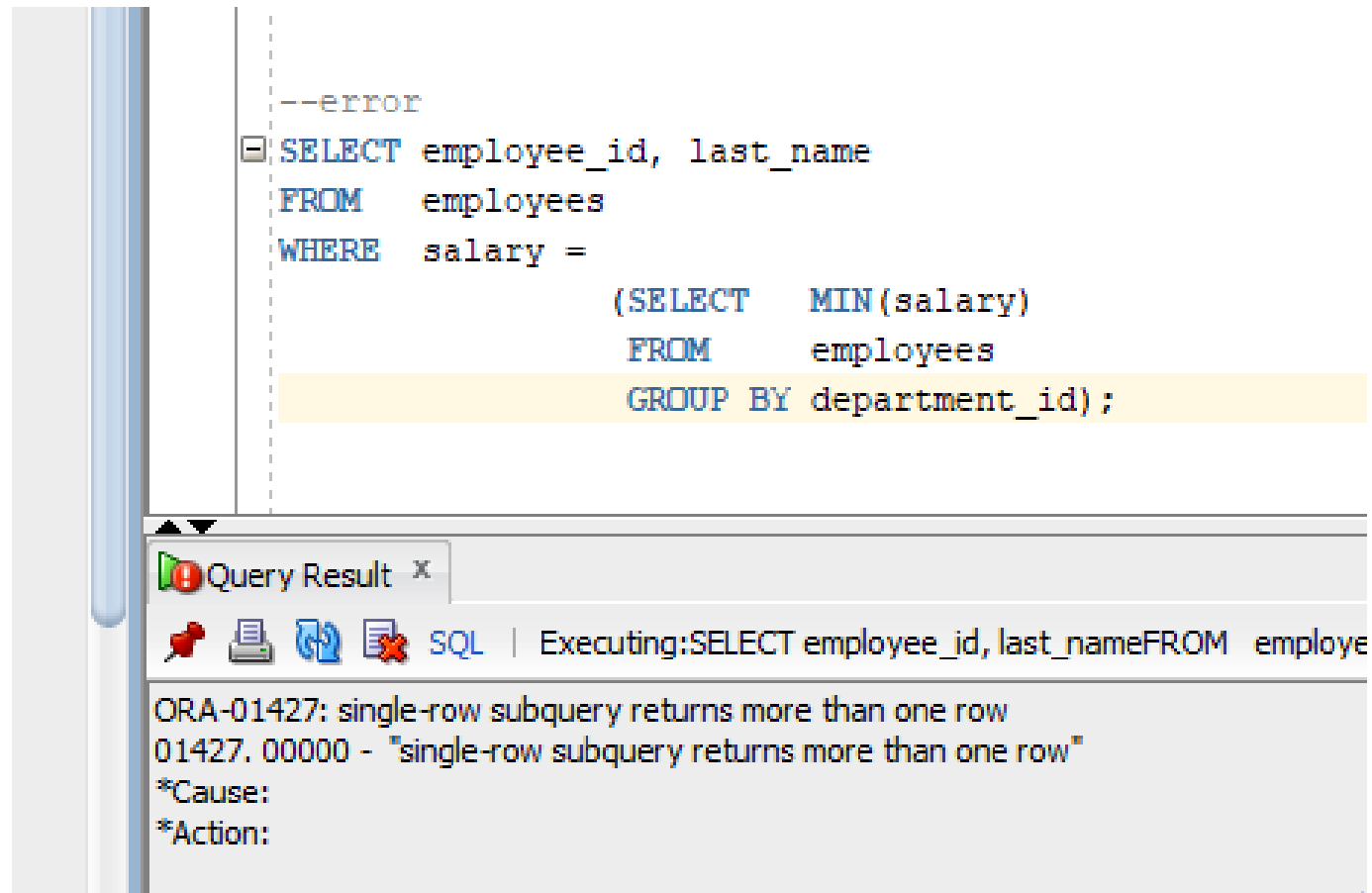
```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) >
        (SELECT MIN(salary)
         FROM employees
         WHERE department_id = 60);
```

Query Result x

SQL | All Rows Fetched: 9 in 0.008 seconds

	DEPARTMENT_ID	MIN(SALARY)
1	40	6500
2	110	8300
3	90	17000
4	70	10000
5	(null)	7000
6	10	4400
7	20	6000
8	100	6900
9	80	6100

What Is Wrong with This Statement?



The screenshot shows an SQL IDE with a query editor and a results pane. The query in the editor is:

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

The results pane, titled "Query Result", shows an error message:

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

The error occurs because the subquery `(SELECT MIN(salary) FROM employees GROUP BY department_id)` returns multiple rows (one for each department), while the main query expects a single value for the `salary` column.

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');
```

Query Result x

SQL | All Rows Fetched: 0 in 0.005 seconds

LAST_NAME	JOB_ID
-----------	--------

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

Multiple-Row Subqueries

- Return more than one row and 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 <code>FALSE</code> 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 <code>TRUE</code> 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';
```

Query Result x

SQL | Fetched 50 rows in 0.008 seconds

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	132	Olson	ST_CLERK	2100
2	136	Philtanker	ST_CLERK	2200
3	128	Markle	ST_CLERK	2200
4	135	Gee	ST_CLERK	2400
5	127	Landry	ST_CLERK	2400
6	191	Perkins	SH_CLERK	2500
7	182	Sullivan	SH_CLERK	2500
8	144	Vargas	ST_CLERK	2500
9	140	Patel	ST_CLERK	2500

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

Using the ALL Operator in Multiple-Row Subqueries

The screenshot shows an SQL Worksheet interface with a query editor and a results pane. The query is as follows:

```
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';
```

The results pane shows the following data:

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	185	Bull	SH_CLERK	4100
2	192	Bell	SH_CLERK	4000
3	193	Everett	SH_CLERK	3900
4	188	Chung	SH_CLERK	3800
5	137	Ladwig	ST_CLERK	3600
6	189	Dilly	SH_CLERK	3600
7	141	Rajs	ST_CLERK	3500
8	186	Dellinger	SH_CLERK	3400
9	133	Mallin	ST_CLERK	3300
10	129	Bissot	ST_CLERK	3300
11	180	Toussaint	SH_CLERK	3200

The screenshot shows an SQL Worksheet interface with a query editor. The query is as follows:

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


Null Values in a Subquery

The screenshot shows the SQL Developer interface. On the left, the schema browser displays the 'EMPLOYEES' table with columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, and DEPARTMENT_ID. The main query editor contains the following SQL:

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

The 'Query Result' window at the bottom shows 'All Rows Fetched: 0 in 0.033 seconds', indicating that no results were returned due to the lack of null handling in the subquery.

The screenshot shows the same SQL Developer interface, but the query has been modified to include a null check in the subquery:

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

The 'Query Result' window now shows 'Fetched 50 rows in 0.124 seconds'. The results table below the query shows the following data:

	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida
7	Banda

Summary

In this lesson, you should have learned how to:

- Identify situations where a subquery is useful for solving a problem
- Understand how to write subqueries when query conditions depend on unknown values
- Apply subqueries to break complex queries into clearer, logical steps

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