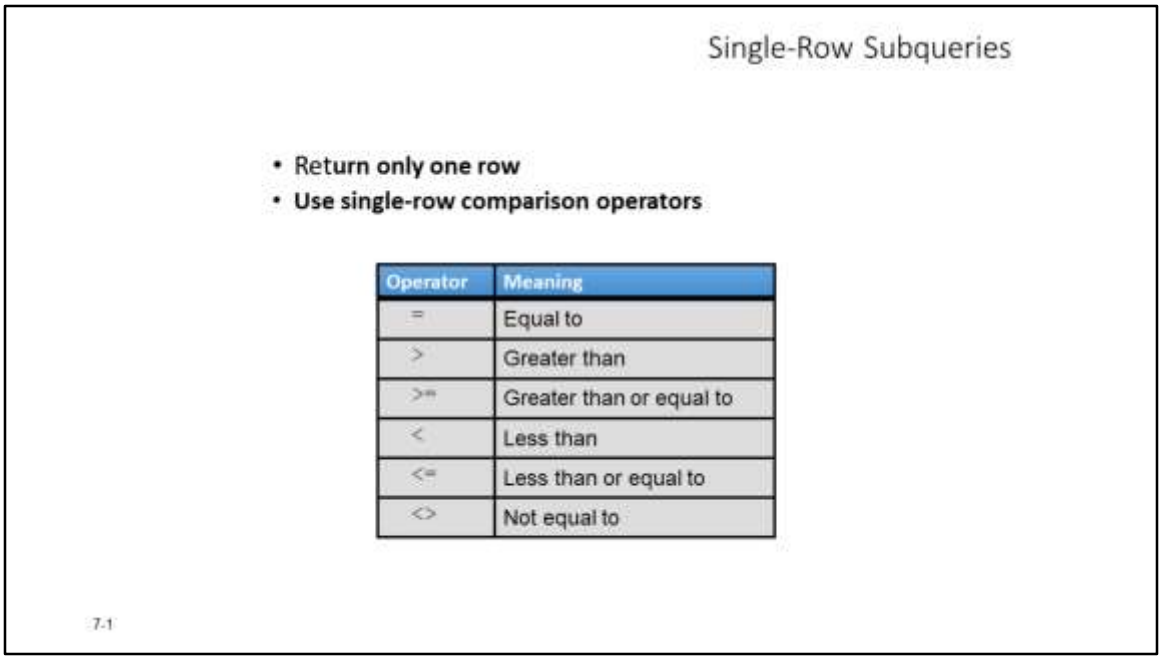


Single-Row Subqueries

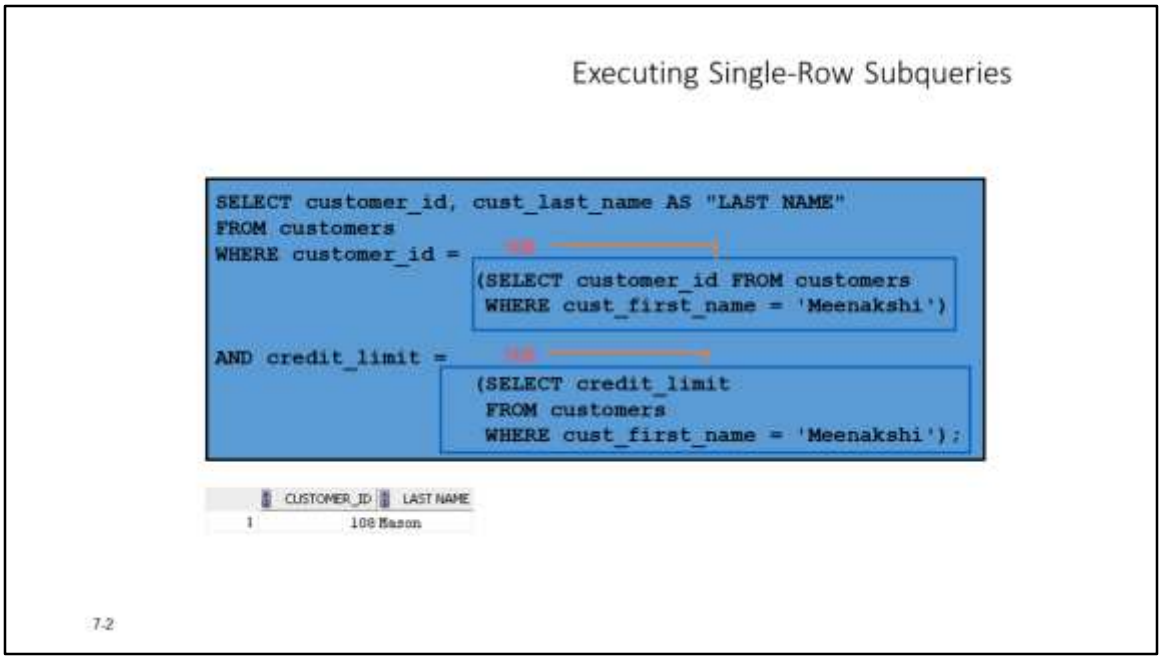
A single-row subquery is one that returns one row from the inner `SELECT` statement. This type of subquery uses a single-row operator. The slide gives a list of single-row operators.

Example:

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

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

	LAST_NAME	JOB_ID
1	Rajs	ST_CLERK
2	Davies	ST_CLERK
3	Matos	ST_CLERK
4	Vargas	ST_CLERK

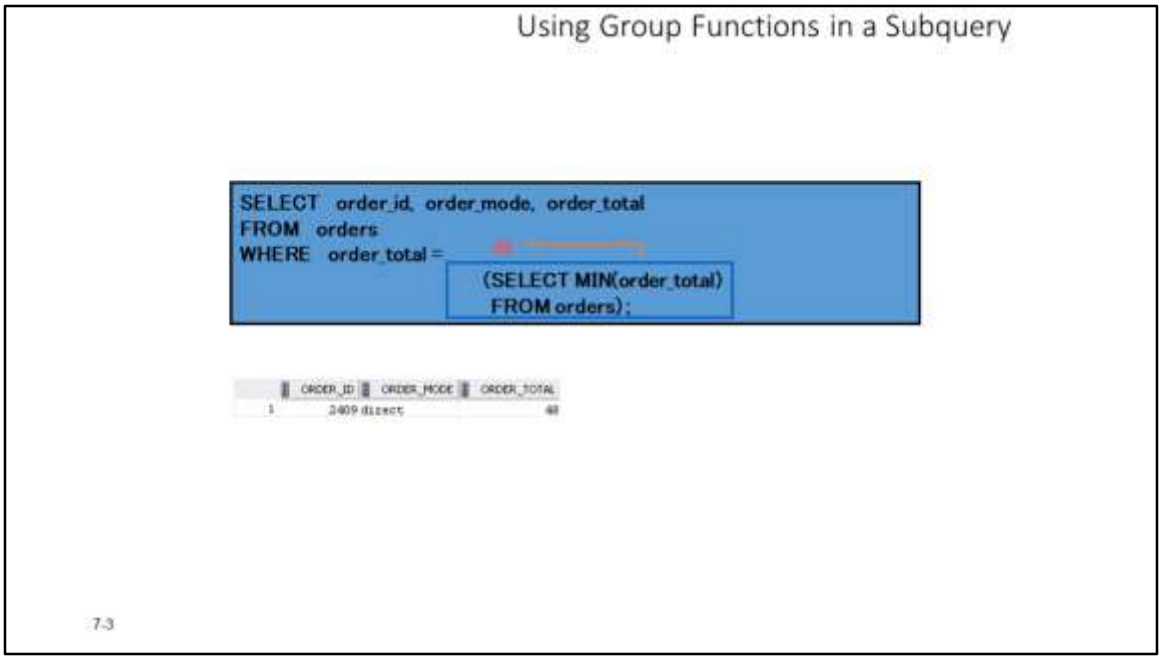


Executing Single-Row Subqueries

A `SELECT` statement can be considered as a query block. The example in the slide displays customers who have the same ID as “Meenakshi,” and have a credit limit as that of.

The example consists of three query blocks: the outer query and two inner queries. The inner query blocks are executed first, producing the query results 108 and `700`, respectively. The outer query block is then processed and uses the values that were returned by the inner queries to complete its search conditions. Both inner queries return single values (`108` and `700`, respectively), so this SQL statement is called a single-row subquery.

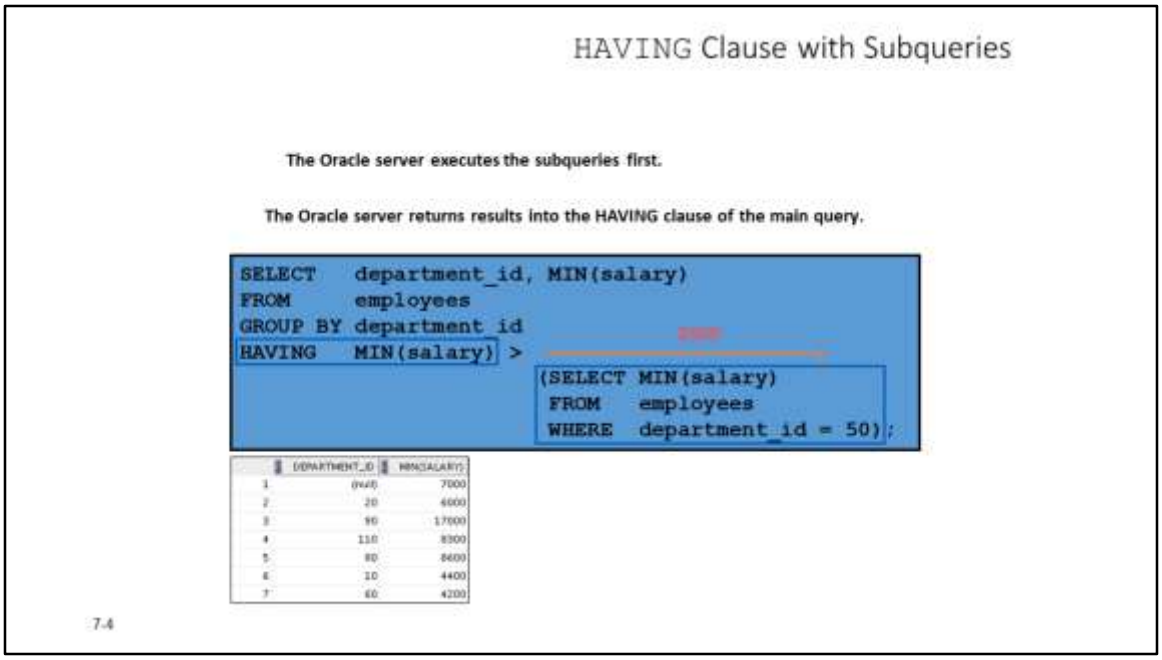
Note: The outer and inner queries can get data from different tables.



Using Group Functions in a Subquery

You can display data from a main query by using a group function in a subquery to return a single row. The subquery is in parentheses and is placed after the comparison condition.

The example in the slide displays the order ID, order mode, and total order of all orders whose total order is equal to the minimum total order. The `MIN` group function returns a single value (`48`) to the outer query.



HAVING Clause with Subqueries

You can use subqueries not only in the WHERE clause, but also in the HAVING clause. The Oracle server executes the subquery and the results are returned into the HAVING clause of the main query.

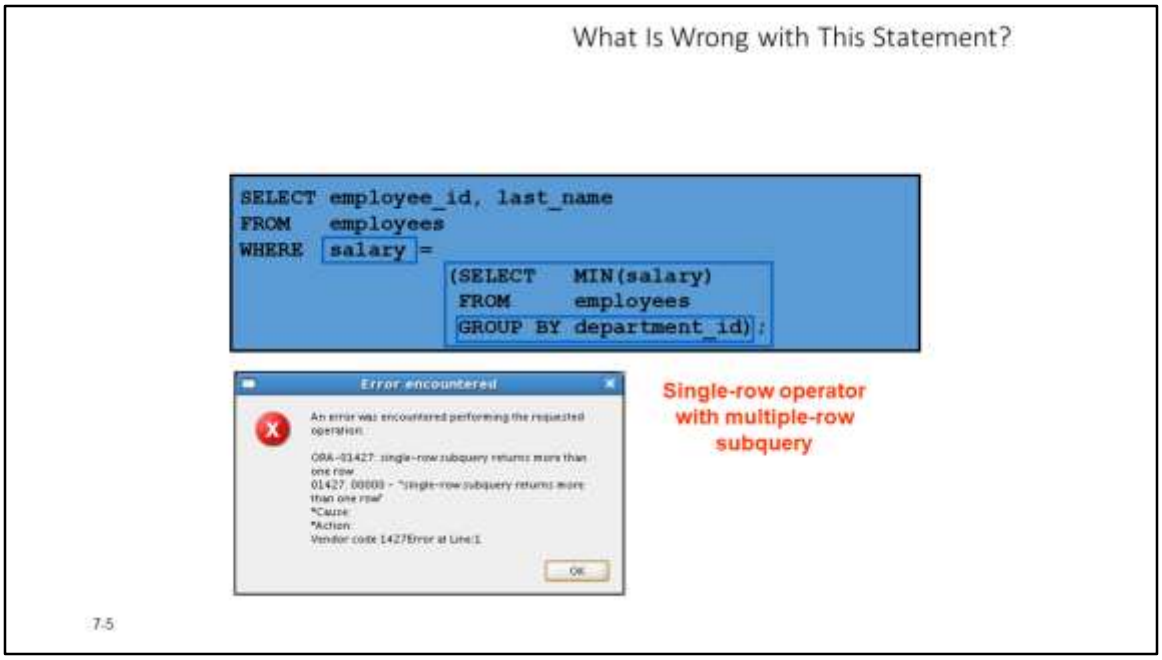
The SQL statement in the slide displays all the departments that have a minimum salary greater than that of department 50.

Example:

Find the job with the lowest 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);
```

	JOB_ID	AVG(SALARY)
1	ST_CLERK	2925



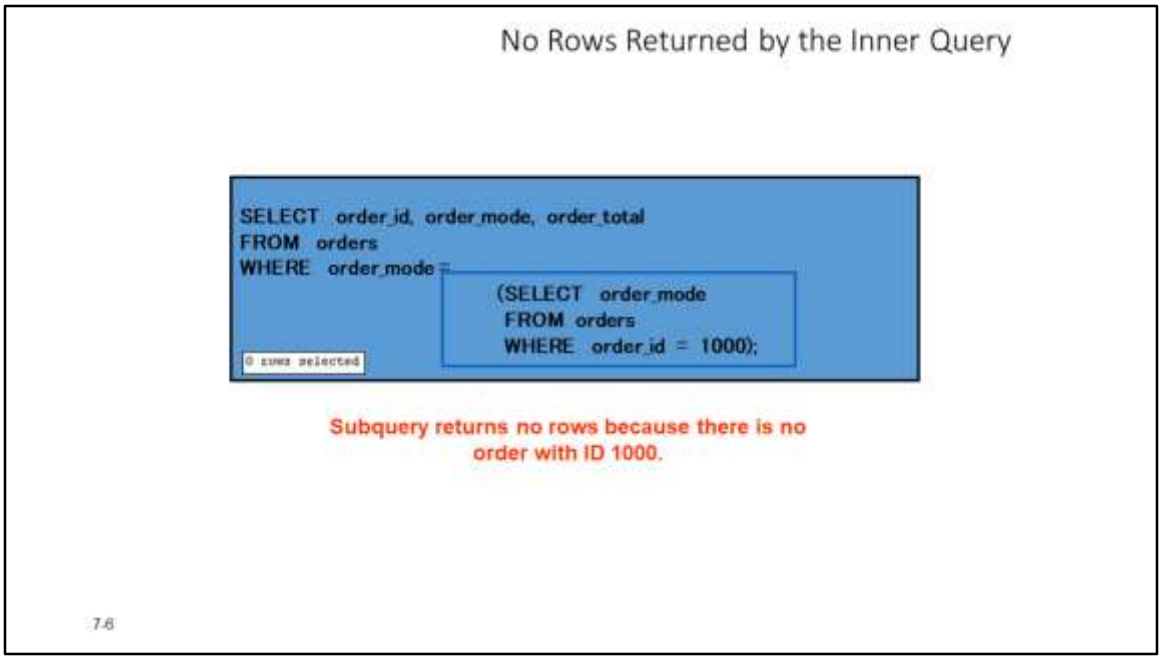
What Is Wrong with This Statement?

A common error with subqueries occurs when more than one row is returned for a single-row subquery.

In the SQL statement in the slide, the subquery contains a `GROUP BY` clause, which implies that the subquery will return multiple rows, one for each group that it finds. In this case, the results of the subquery are `4400`, `6000`, `2500`, `4200`, `7000`, `17000`, and `8300`.

The outer query takes those results and uses them in its `WHERE` clause. The `WHERE` clause contains an equal (=) operator, a single-row comparison operator that expects only one value. The = operator cannot accept more than one value from the subquery and, therefore, generates the error.

To correct this error, change the = operator to `IN`.



No Rows Returned by the Inner Query

Another common problem with subqueries occurs when no rows are returned by the inner query.

In the SQL statement in the slide, the subquery contains a `WHERE` clause. Presumably, the intention is to find the order whose ID is 1000. The statement is correct, but selects no rows when executed because there is no order with ID 1000. Therefore, the subquery returns no rows.

The outer query takes the results of the subquery (null) and uses these results in its `WHERE` clause. The outer query finds no order with an order mode equal to null, and so returns no rows. If a job existed with a value of null, the row is not returned because comparison of two null values yields a null; therefore, the `WHERE` condition is not true.