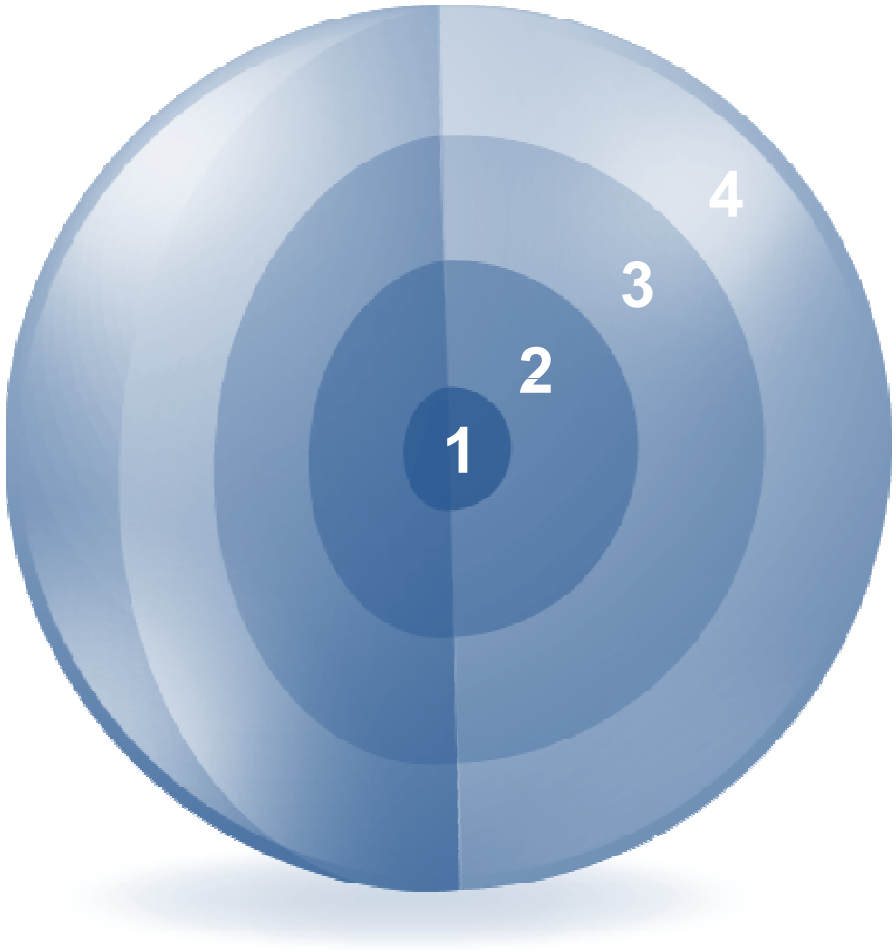
Lesson 7

Using Sub queries to Solve Queries

What You will learn at the end of this Session?



1. Define subqueries

2. Describe the types of problems that the subqueries can solve

3. List the types of subqueries

4. Write single-row and multiple-row subqueries

Using a Subquery to Solve a Problem

Who has a credit limit than Charlie Pacino's?

Main query:

Which customers have credit limit greater than Charlie Pacino's credit limit?

Subquery:

What is Charlie Pacino's credit limit?

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

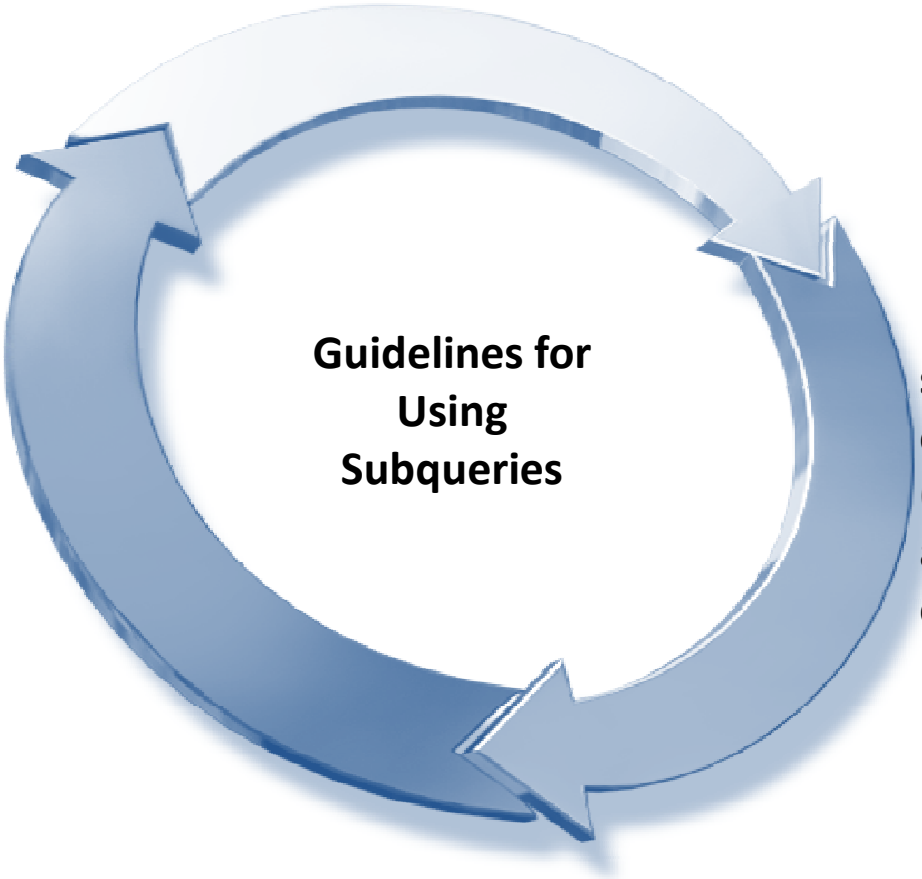
```
SELECT cust_first_name || cust_last_name
FROM customers
WHERE credit_limit <  200
                      (Select credit_limit
                       FROM customers
                       WHERE cust_first_name = 'Charlie'
                       AND cust_last_name = 'Pacino');
```

	CUSTOMER_ID	UNIT_PRICE	WAREHOUSE_ID
1	105	199.1	9
2	105	199.1	2
3	105	199.1	4
4	105	199.1	6
5	105	199.1	8
6	105	226.6	9
7	105	226.6	2

▪ ▪ ▪

Guidelines for Using Subqueries

Enclose subqueries in parentheses.



Use single-row operators with single-row subqueries and multiple-row operators with multiple-row subqueries.

Place subqueries on the right side of the comparison condition for readability. (However, the subquery can appear on either side of the comparison operator.)

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 customer_id, cust_last_name AS "LAST NAME"
FROM customers
WHERE customer_id =     108
                    (SELECT customer_id FROM customers
                     WHERE cust_first_name = 'Meenakshi')

AND credit_limit =      700
                    (SELECT credit_limit
                     FROM customers
                     WHERE cust_first_name = 'Meenakshi');
```

	CUSTOMER_ID	LAST NAME
1	108	Mason

Using Group Functions in a Subquery

```
SELECT   order_id,  order_mode,  order_total
FROM   orders
WHERE   order_total =      48
                          (SELECT MIN(order_total)
                           FROM orders) ;
```

	ORDER_ID	ORDER_MODE	ORDER_TOTAL
1	2409	direct	48

HAVING Clause with Subqueries

The Oracle server executes the subqueries first.

The Oracle server returns results into the HAVING clause of the main query.

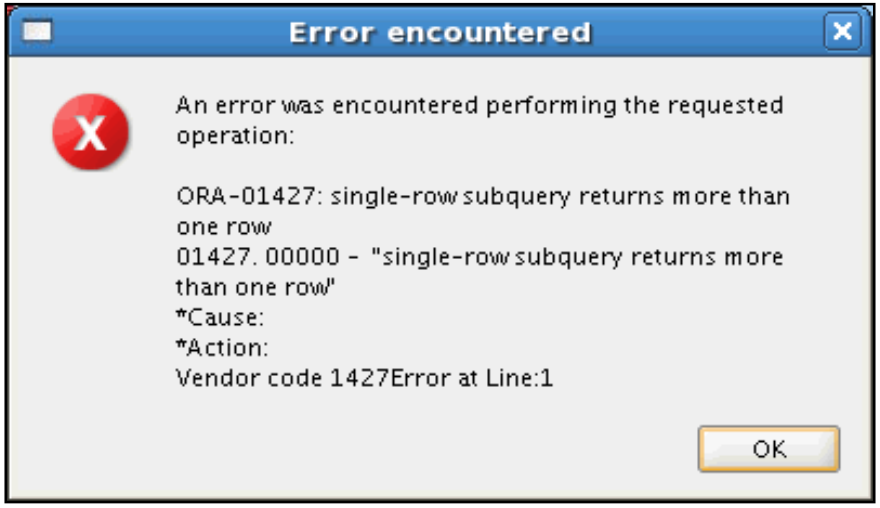
```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY  department_id
HAVING    MIN(salary) > (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50);
```

2500

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	20	6000
3	90	17000
4	110	8300
5	80	8600
6	10	4400
7	60	4200

What Is Wrong with This Statement?

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



Single-row operator with multiple-row subquery

No Rows Returned by the Inner Query

```
SELECT   order_id,  order_mode,  order_total
FROM   orders
WHERE   order_mode =
                     (SELECT   order_mode
                      FROM  orders
                      WHERE   order_id  =  1000);
```

```
0 rows selected
```

Subquery returns no rows because there is no order with ID 1000.

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

Quiz

Using a subquery is equivalent to performing two sequential queries and using the result of the first query as the search values in the second query.

1. True
2. False