SubQueries

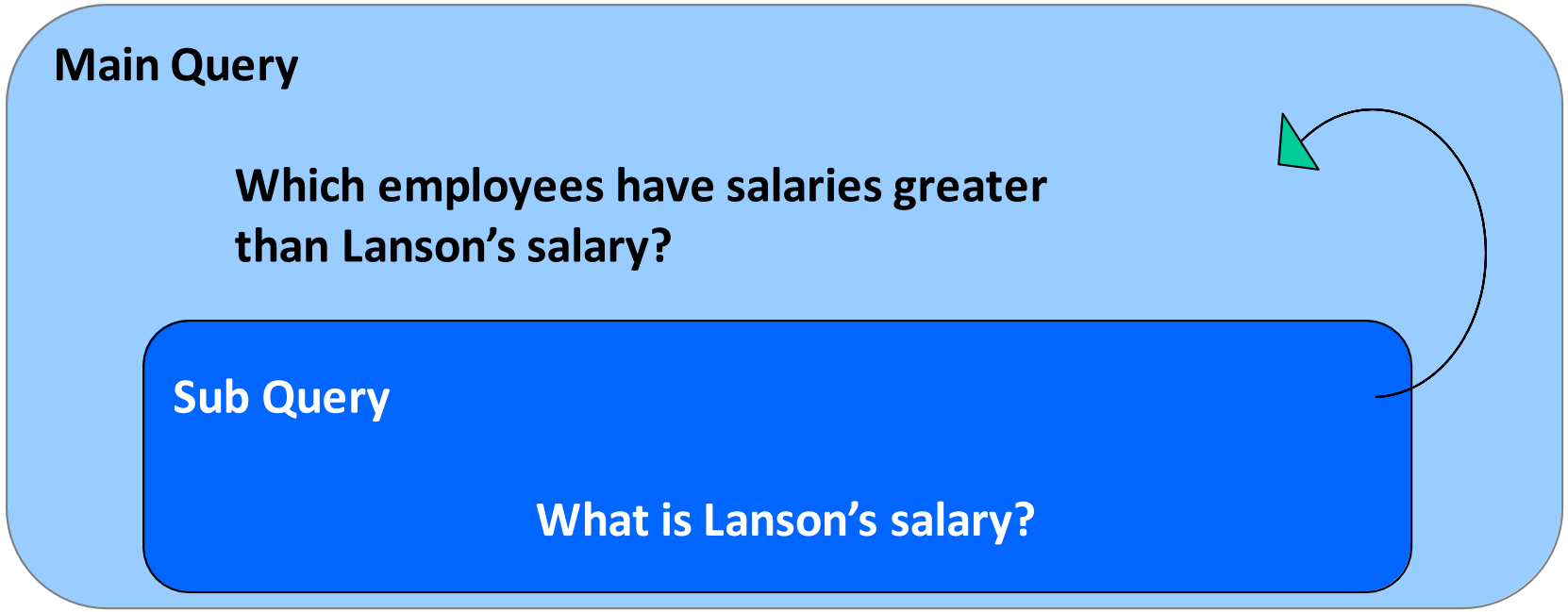
Objective

After completing this module, the participant will be able to:

- **Describe the types of problem that sub queries can solve**
- **Define sub queries**
- **List the types of sub queries**
- **Write single-row and multiple-row sub queries**

Using a Sub query to Solve a Problem

Who has Salary greater than Lanson's salary?



Sub query

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

Syntax:

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

Example:

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

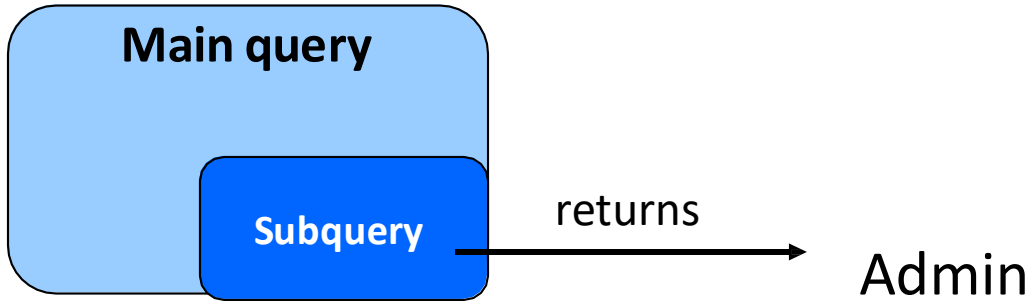

Guidelines

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

Types of Sub queries

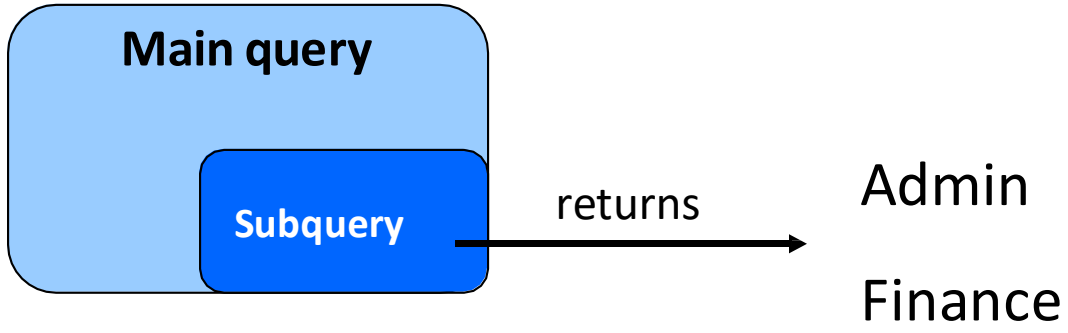
Single-row subqueries:

Queries that return only one row from the inner SELECT statement



Multiple-row subqueries:

Queries that return more than one row from the inner SELECT statement



Single Row Sub query

- Returns onle one row
- Use single row comparision operators
- sub queries and use multiple-row operators with multiple-row sub queries.

= , > , >= ,

< , <= , <>

Example 1:

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

Example 2:

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

Group Functions in a Sub query

Example:

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


The HAVING Clause with Subqueries

Example:

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


Multiple-Row Subqueries

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

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

Multiple-Row Subqueries-IN Operator

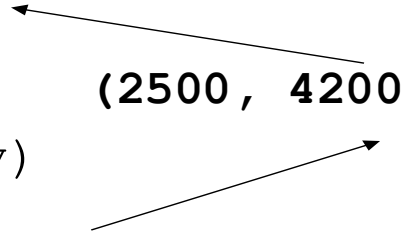
Example:

```
SELECT last_name, salary, department_id  FROM

employees

WHERE salary IN          (2500, 4200, 4400, 6000, 7000, 8300, 8600, 17000);
(SELECT MIN(salary)
FROM employees

GROUP BY department_id);
```



Multiple-Row Subqueries-ANY Operator

Example:

```
SELECT employee_id,
last_name,job_id, salary
FROM employees  WHERE salary
< ANY
(SELECT salary ------>  (9000,6000,4200)
FROM employees
WHERE job_id = 'IT_PROG')  AND
job_id <> 'IT_PROG';
```

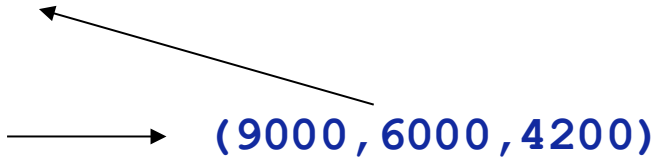
<ANY means less than the maximum.

>ANY means more than the minimum.

Multiple-Row Subqueries-ALL Operator

Example:

```
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';
```



<ALL means less than the minimum.

>ALL means more than the máximum.

Set Operators

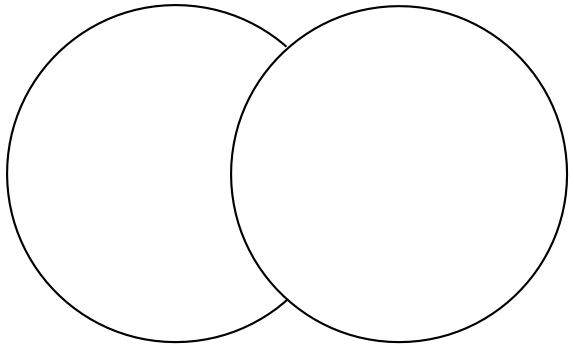
Objective

After completing this module, the participant will be able to:

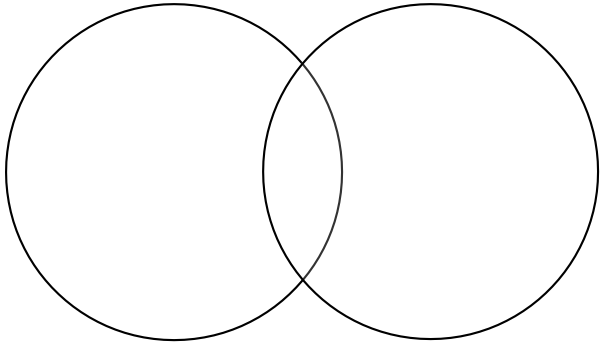
- **Describe SET operators**
- **Use a SET operator to combine multiple queries into a single**
- **query**
- **Control the order of rows returned**

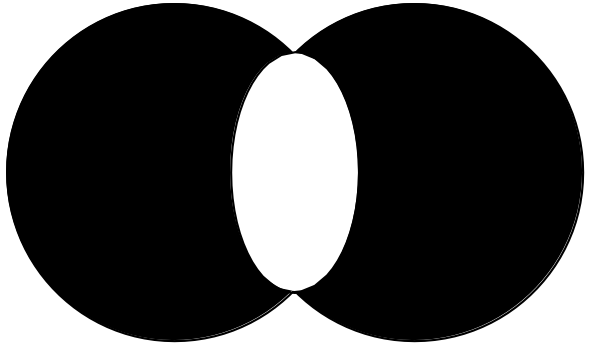
Set Operators

UNION

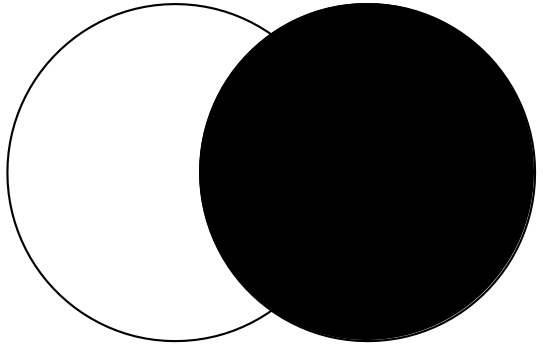


UNION ALL





INTERSECT



MINUS

Union Operator

The UNION operator returns results from both queries after eliminating duplications.

Example:

```
SELECT employee_id, job_id  FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

Union Operator

- The UNION operator returns results from both queries after eliminating duplications.
- The number of columns and the datatypes of the columns being selected must be identical in all the SELECT statements used in the query.
  - The names of the columns need not be identical.
  - By default, the output is sorted in ascending order of the first column of the SELECT clause.

UNION ALL operator

The UNION ALL operator returns results from both queries, including all duplications.

Example:

```
SELECT employee_id, job_id, department_id  FROM employees
UNION ALL
SELECT employee_id, job_id, department_id  FROM job_history
ORDER BY employee_id;
```


The INTERSECT Operator

Use the INTERSECT operator to return all rows common to multiple queries.

Example:

```
SELECT employee_id, job_id  FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```


The MINUS Operator

Use the MINUS operator to return rows returned by the first query that are not present in the second query (the first SELECT statement MINUS the second SELECT statement)

Example:

```
SELECT employee_id,job_id
FROM employees  MINUS
SELECT employee_id,job_id  FROM job_history;
```

The Oracle Server and SET Operators

- Duplicate rows are automatically eliminated except in UNION ALL.
- Column names from the first query appear in the result.
- The output is sorted in ascending order by default except in UNION ALL.

Matching the SELECT Statements

Example 1:

```
SELECT department_id, TO_NUMBER(null) location, hire_date
FROM employees
UNION
SELECT department_id, location_id, TO_DATE(null)
FROM departments;
```

Example 2:

```
SELECT employee_id, job_id,salary
FROM employees
UNION
SELECT employee_id, job_id,0
FROM job_history;
```

Thank you

Innovative Services

Passionate Employees

Delighted Customers