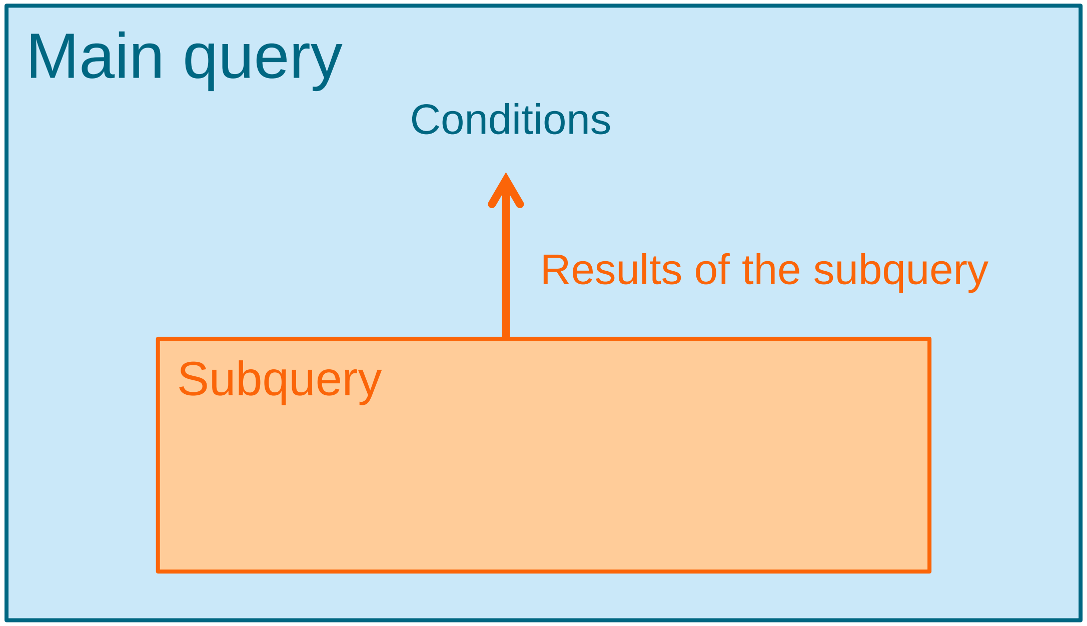
Module 5
Subqueries

Table of Content

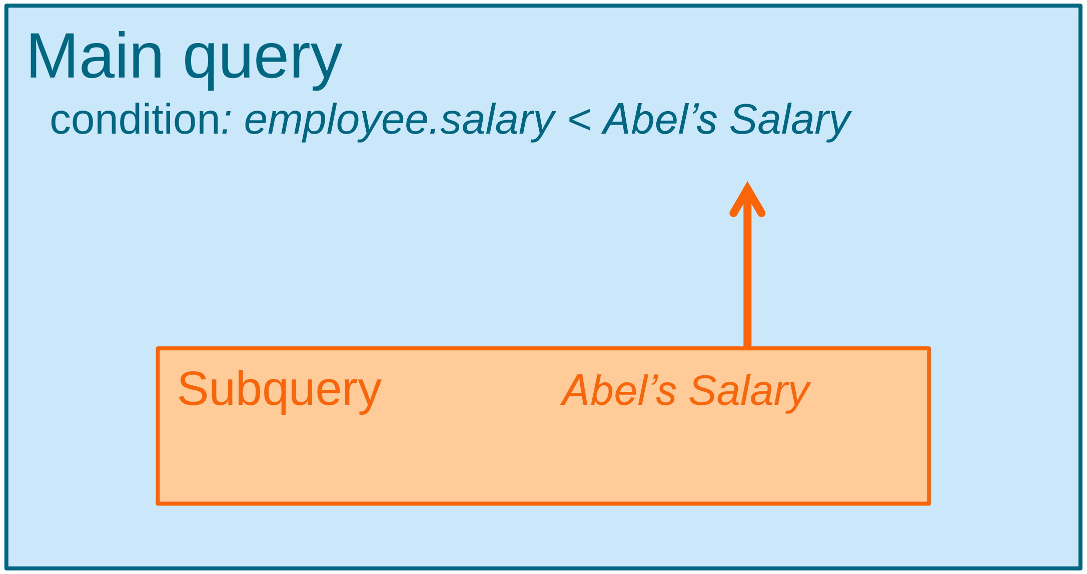
Using Subquery in Main Query

- The subquery is executed before the main query
- The result of the subquery is used in the conditions of the main query



Using Subquery in Main Query

- E.g, *who earns less than Abel ?*



Subquery Syntax

```
SELECT   selectlist
FROM     table1
WHERE    expression1    operator
                ( SELECT expression2
                  FROM  table2 ) ;
```

- E.g.

```
select    last_name, salary
from      employee
where     salary <                         11000
          (select     salary
           from       employee
           where      employee_id  = 174) ;
```

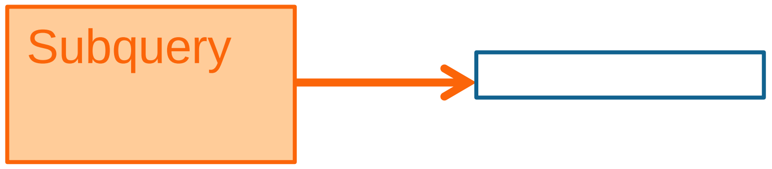
Subquery Syntax

- Subqueries enclosed in parentheses
- Subqueries on the right side of the comparison condition for readability

Types of Subqueries

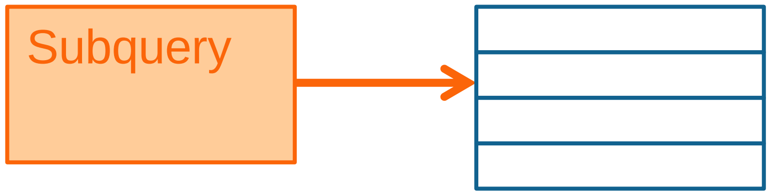
- Single-row subquery
  - Returns only one row

    ↳Use of **single-row operators** in condition of the main query



- Multiple-row subquery
  - Returns multiple rows

    ↳Use of **multiple-row operators** in condition of the main query



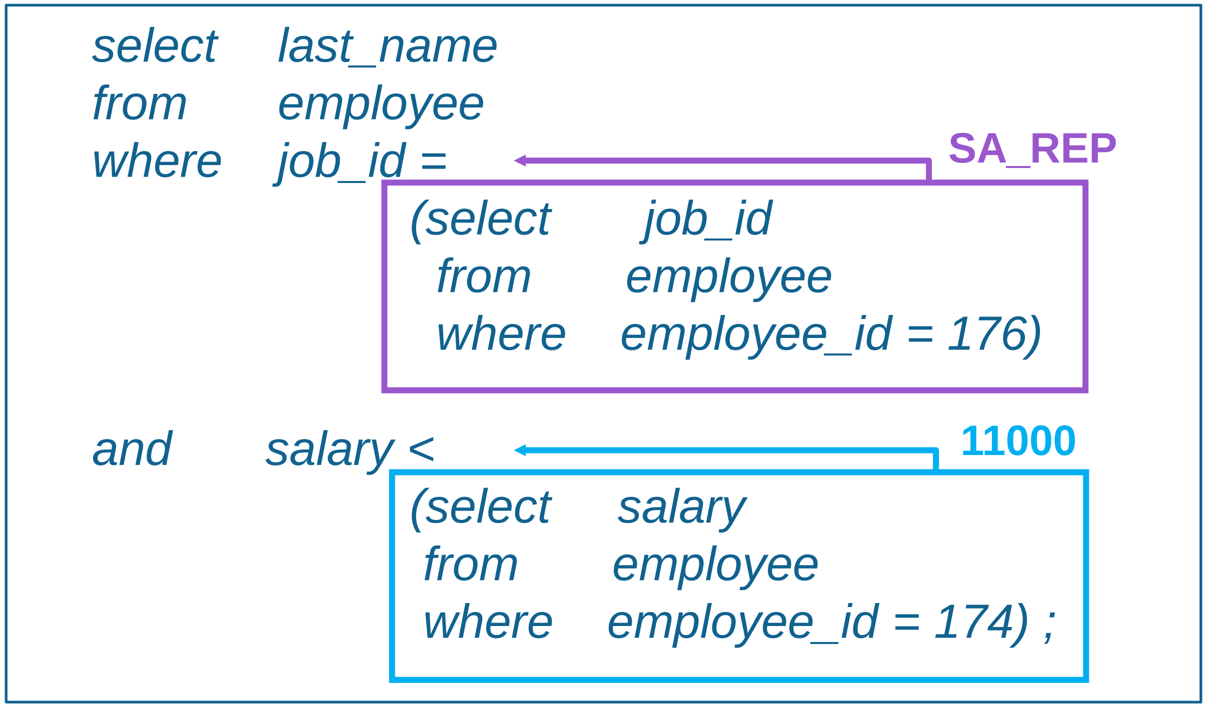
Single-Row Subqueries

- 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

Single-Row Subqueries

- E.g.



```
select    last_name
from      employee
where     job_id =
               (select      job_id
                 from       employee
                 where      employee_id = 176)

and       salary <
               (select      salary
                from        employee
                where       employee_id = 174) ;
```

Single-Row Subqueries

```
select   last_name
from     employee
where    salary =
                 ( select   min(salary)
                   from     employee ) ;
```

```
select    last_name
from      employee
where     job_id =
                   ( select    job_id
                     from      employee
                     where     employee_id = 1099 ) ;
```

↳The main query returns no row
because the subquery returns no row (no employee 1099)

Single-Row Subqueries

- Subquery is allowed in the HAVING clause
  - First, the subquery is executed
  - Then, the having clause of the main query is executed
  - E.g.

```
select      department_id, min(salary)
from        employee
group by    department_id
having      min(salary) >  <----------------------  2500
                    ( select   min(salary)
                      from     employee
                      where    department_id = 50) ;
```

Multiple-Row Subqueries

- If the subquery returns multiple rows
  ↳ Multiple-row operators have to be used in condition of the main query

- Wrong example

```
select   last_name
from     employee
where    salary  =
                  ( select  salary
                    from    employee
                    where   last_name in ( 'Abel',  'Taylor', 'King' ) ) ;
```

Returns more than one row

↳Error : Single-row operator with multiple-row subquery

Multiple-Row Subqueries

- 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.

Multiple-Row Subqueries

- E.g.

```
select    last_name
from      employee
where     salary <  any          9000, 6000, 4200
              (select    salary
               from      employee
               where     job_id = 'IT_PROG')
and       job_id <> 'IT_PROG' ;
```

↳ *Select employees whose salary is lower than at least one of the list i.e. employees who earn less than 9000 !*

Multiple-Row Subqueries

- E.g.

```
select      last_name
from        employee
where       salary <  all          9000, 6000, 4200
                 (select    salary
                  from      employee
                  where     job_id = 'IT_PROG')
and         job_id <> 'IT_PROG' ;
```

↳ *Select employees whose salary is lower than all salaries of the list i.e. employees who earn less than 4200 !*

Multiple-Row Subqueries

- **NOT IN**
  - If a multiple-row subquery returns a list containing a null value

    ⇘ The main query returns no row

  - E.g.

```
select      last_name
from        employee
where       employee_id   not in
                 ( select    manager_id
                   from      employee );
```

⇘ *Some employees have no manager,*

 ⇨ *the list returned by the subquery contains the null value*

⇨ *The main query returns no row*

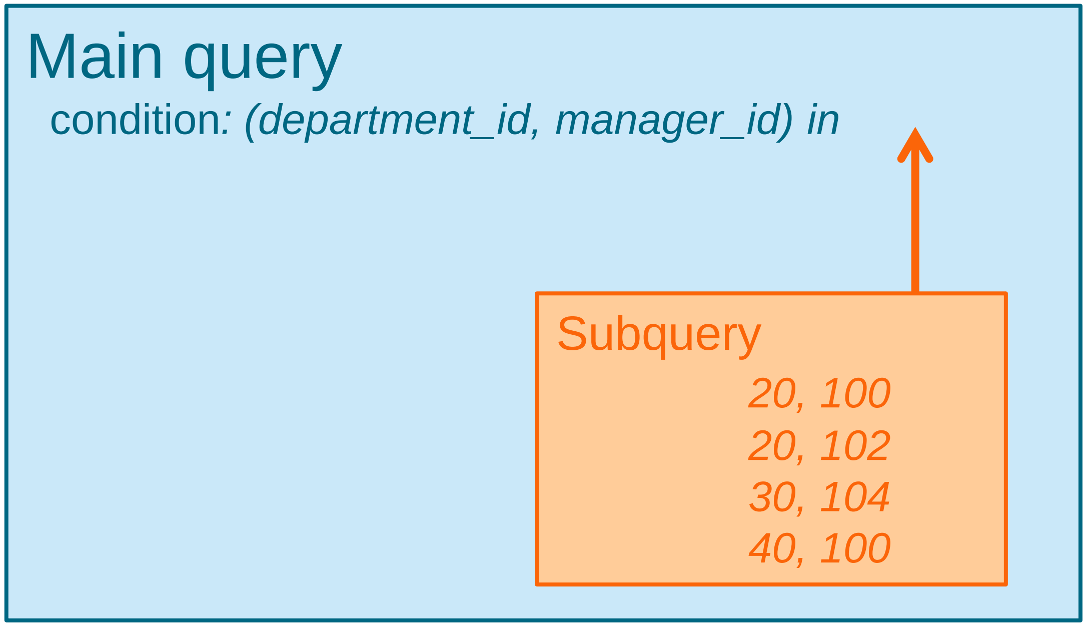
Multiple-Row Subqueries

⇨ *To return the last names of employees who are not managers :*

```
select      last_name
from        employee
where       employee_id   not in
                 ( select    manager_id
                   from      employee
                   where     manager_id  is not null );
```

Multiple-Column Subqueries

- E.g,



Multiple-Column Subqueries

```
SELECT      selectlist
FROM        table1
WHERE       ( { expression1, … } )  operator
                     ( SELECT    { expression2, … }
                       FROM        table2 ) ;
```

- Multiple-column comparisons involving subqueries can be
  - Nonpairwise comparisons
  - Pairwise comparisons **(only in Oracle)**

Pairwise/Nonpairwise Comparison

- Pairwise comparison **(only in Oracle)**

```
select   *
from     employee
where    (department_id, manager_id) in
                 ( select   department_id, manager_id
                   from     employee
                   where    last_name in ( 'Abel',  'Vargas' ) ) ;
```

	Department_id	Manager_id
Abel	80	149
Vargas	50	124

↳Allowed pairs : (80,149) and (50,124)

Pairwise/Nonpairwise Comparison

- Nonpairwise comparison

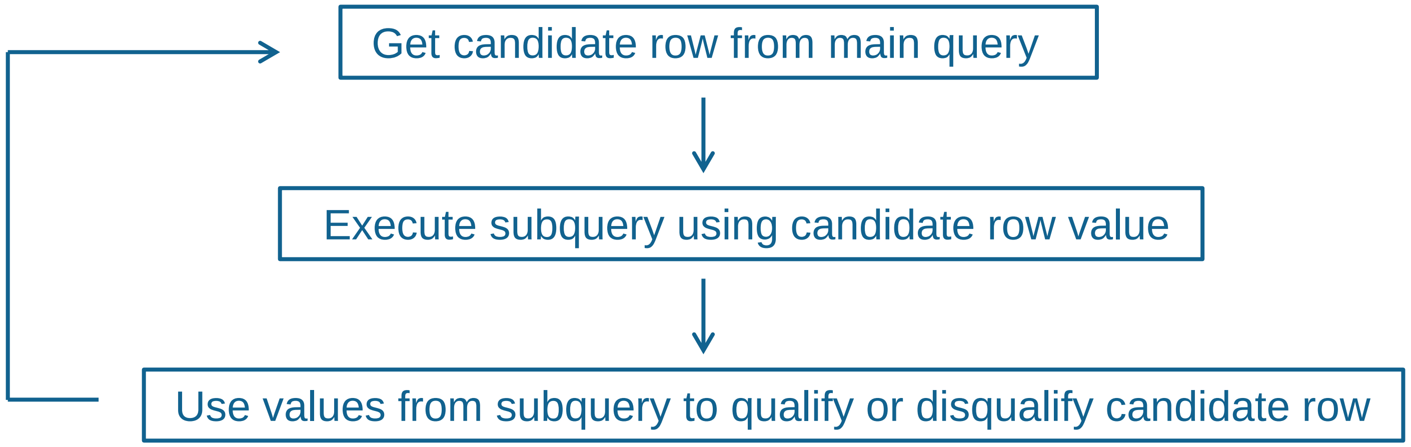
```
select      *
from       employee
where     department_id  in
                  ( select   department_id
                    from      employee
                    where   last_name in ( 'Abel',  'Vargas' ))
and       manager_id  in
                  ( select   manager_id
                    from      employee
                    where   last_name in ( 'Abel',  'Vargas’)) ;
```

↳Allowed pairs: (80,149), (50,124), (80,124) and (50,149)

- Pairwise comparison is different from nonpairwise comparison

Correlated Subqueries

- Row-by-row processing
  - Each subquery is executed once for every row of the main query


Get candidate row from main query
Execute subquery using candidate row value
Use values from subquery to qualify or disqualify candidate row

Correlated Subqueries

- The subquery references a column from a table of the main query

```
SELECT   selectlist1
FROM     table1    alias1
WHERE    expression1  operator
                  ( SELECT    expression2
                    FROM      table2
                    WHERE     column2  = alias1.column1 ) ;
```

- E.g.

```
select      *
from        employee   empl
where      salary >
          ( select   avg(salary)
            from      employee
            where   department_id = empl.department_id) ;
```

Average salary of the employee department
↳ Calculated for each employee

Exists Operator

- Tests for existence of rows in the results set of the subquery
  - Returns true if at least one row is returned by the subquery
  - Returns false if no row is returned by the subquery

```
SELECT   select_list1
FROM     table1    alias1
WHERE    [NOT] EXISTS
                ( SELECT    select_list2
                  FROM      table2
                  WHERE     expr2  = alias1.expr1) ;
```

Exists Operator

- E.g,
  - *Employees who are managers*

```
select    *
from      employee   empl
where     exists
     ( select   *
       from     employee
       where    manager_id = empl.employee_id ) ;
```

  - *Departments without employee*

```
select    *
from      department   dept
where     not exists
     ( select   *
       from     employee
       where    department_id = dept.department_id ) ;
```

Equivalence of Queries

- Different ways to write a same query
- E.g, *names of departments having employees*
  - *Join*

```
 select  distinct  d.department_name   from employee  e inner join department  d
on e.department_id = d.department_id ;
```

  - *Subquery*

```
 select   department_name   from department
 where   department_id  in
                (select  distinct  department_id   from  employee) ;
```

  - *Exists*

```
 select  department_name   from   department  d
 where   exists ( select   *     from  employee  e
                           where    e.department_id = d.department_id) ;
```

Summary

```
SELECT      select_list
FROM        table1   [ alias1 ]
WHERE       ( { expression1, … } )  operator
                        ( SELECT    { expression2, … }
                          FROM        table2
                          [ WHERE    conditions ] ) ;
```