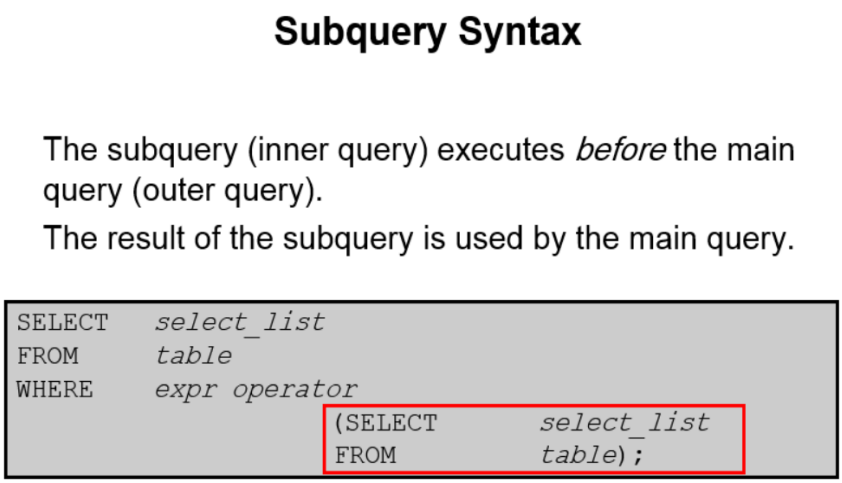
Using Subqueries to Solve Queries

Single row Subqueries



Find a person who has salary > Abel's salary. Assume that Abel is the last name.

We need to know Abel's salary

```
SELECT SALARY FROM EMPLOYEES WHERE LAST_NAME = 'Abel';
```

SALARY
11000

Then make the query like this

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
FROM
EMPLOYEES
WHERE SALARY > (SELECT SALARY FROM EMPLOYEES WHERE LAST_NAME = 'Abel');
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000

101	Neena	Kochhar	17000

Note: you can make the subquery on the left side, but this is no recommended.

When we use single-row operator (=, >, < , >=, <=, <>, !=), then the subquery should return a single row.
This SELECT statement will give an error.

```
SELECT * FROM
EMPLOYEES
WHERE SALARY > (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID = 30);
```

ORA-01427: single-row subquery returns more than one row

```
SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID = 30;
```

SALARY
11000
3100
2900
2800
2600
2500

Find the highest salary

```
SELECT * FROM
EMPLOYEES
WHERE SALARY = (SELECT MAX(SALARY) FROM EMPLOYEES);
```

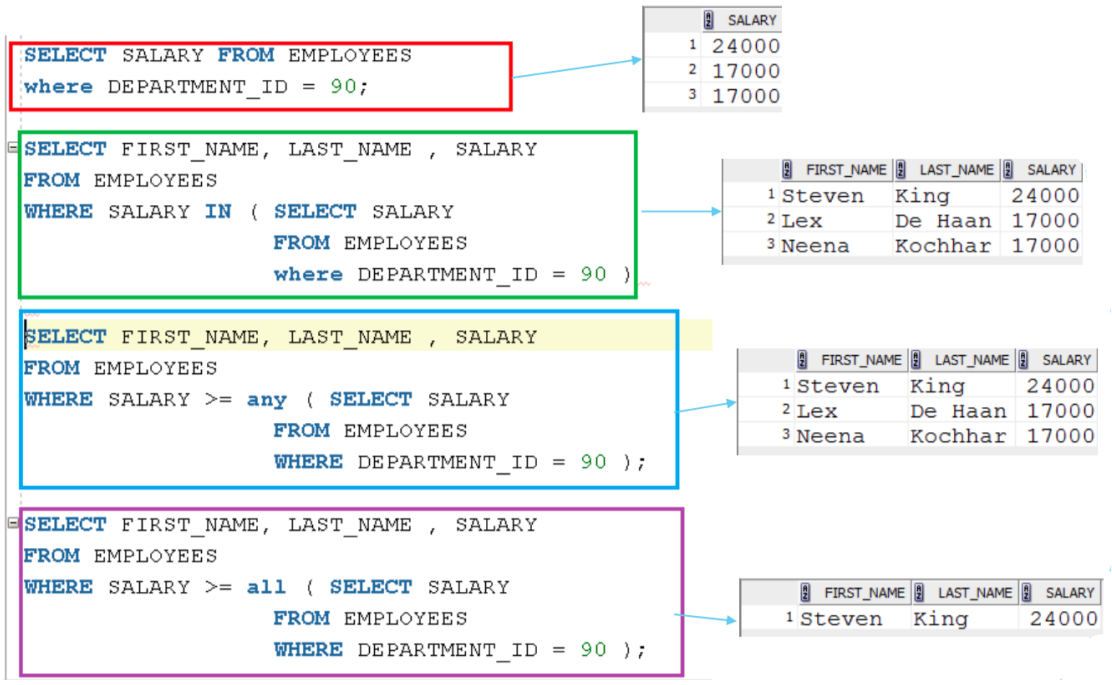
```
SELECT DEPARTMENT_ID, COUNT(EMPLOYEE_ID)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING COUNT (EMPLOYEE_ID) > (SELECT COUNT(1) FROM EMPLOYEES WHERE DEPARTMNENT_ID = 90);
```

Multiple rows Subqueries

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 =, !=, >, <, <=, >=. Returns TRUE if at least one element exists in the result-set of the Subquery for which the relation is TRUE.
ALL	Must be preceded by =, !=, >, <, <=, >=. Returns TRUE if the relation is TRUE for all elements in the result set of the Subquery.



- < ANY means less than the maximum
- >ANY means more than the minimum
- =ANY is equivalent to IN

- >ALL means more than the maximum and <ALL means less than the minimum
- The NOT operator can be used with IN, ANY, and ALL operators

NULL values and Subqueries

When you want to retrieve the employee who has no manager, we use IS NULL

```
SELECT * FROM EMPLOYEES
WHERE MANAGER_ID IS NULL;
```

Note: we cannot use =NULL

If the subquery returns null with operator IN, this is ok

```
SELECT * FROM EMPLOYEES
WHERE MANGER_ID IN (100, 101, NULL);
```

Note: IN is equivalent to =ANY

If the subquery returns null with operator NOT IN, then no records will be retrieved

```
SELECT * FROM EMPLOYEES
WHERE MANAGER_ID NOT IN (100, 101, NULL);
```

Note: NOT IN is equivalent to <>ALL

Exists and not Exists

```
SELECT * FROM
DEPARTMENTS DEPT
WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID FROM EMPLOYEES EMP);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resource	203	2400

50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700

```
SELECT * FROM
DEPARTMENTS DEPT
WHERE EXISTS (SELECT DEPARTMENT_ID FROM EMPLOYEES EMP WHERE EMP.DEPARTMENT_ID =
DEPT.DEPARTMENT_ID);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resource	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700

```
SELECT * FROM
DEPARTMENTS DEPT
WHERE NOT EXISTS (SELECT DEPARTMENT_ID FROM EMPLOYEES EMP WHERE EMP.DEPARTMENT_ID =
DEPT.DEPARTMENT_ID);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
120	Treasury	(null)	1700
130	Corporate Tax	(null)	1700
140	Control And Credit	(null)	1700