

삼성 청년 SW 아카데미

MySQL

<알림>

본 강의는 삼성 청년 SW아카데미의 콘텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

목차

1. JOIN
2. INNER JOIN
3. OUTER JOIN
4. SELF, Non-Equi JOIN
5. SUBQUERY
6. NESTED SUBQUERY
7. INLINE VIEW
8. SCALAR SUBQUERY
9. SUBQUERY 활용

JOIN

삼성 청년 SW 아카데미

✓ JOIN ?

- 둘 이상의 테이블에서 데이터가 필요한 경우 테이블 조인이 필요.
- 일반적으로 조인 조건을 포함하는 WHERE 절을 작성해야 한다.
- 조인 조건은 일반적으로 각 테이블의 PK 및 FK로 구성됩니다.

✓ JOIN의 종류.

- INNER JOIN
- OUTER JOIN
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN

✓ JOIN 조건의 명시에 따른 구분.

- NATURAL JOIN
- CROSS JOIN(FULL JOIN, CARTESIAN JOIN)

✓ JOIN시 주의.

- 조인의 처리는 어느 테이블을 먼저 읽을지를 결정하는 것이 중요(처리할 작업량이 상당히 달라진다.)
- INNER JOIN : 어느 테이블을 먼저 읽어도 결과가 달라지지 않아 MySQL 옵티마이저가 조인의 순서를 조절해서 다양한 방법으로 최적화를 수행할 수 있다.
- OUTER JOIN : 반드시 OUTER가 되는 테이블을 먼저 읽어야 하므로 옵티마이저가 조인 순서를 선택할 수 없다.

✓ JOIN의 필요성.

- 사번이 100인 사원의 사번, 이름, 급여, 부서이름을 출력.

```
select employee_id, first_name, salary, department_name
from employees
where employee_id = 100;
```

Error

- 문제 : 사번, 이름, 급여는 employees table에 있고, department_name은 departments table에 있다.

```
select employee_id, first_name, salary, department_id
from employees
where employee_id = 100;
```

employee_id	first_name	salary	department_id
100	Steven	24000.00	90

```
select department_name
from departments
where department_id = 90;
```

department_name
Executive

✓ JOIN의 필요성.

- 해결 : 두 table(employees와 departments)을 join.

```
select employee_id, first_name, salary, department_name
from employees, departments
where employees.department_id = departments.department_id
and employee_id = 100;
```

employee_id	first_name	salary	department_name
100	Steven	24000.00	Executive

이어서..

삼성 청년 SW 아카데미

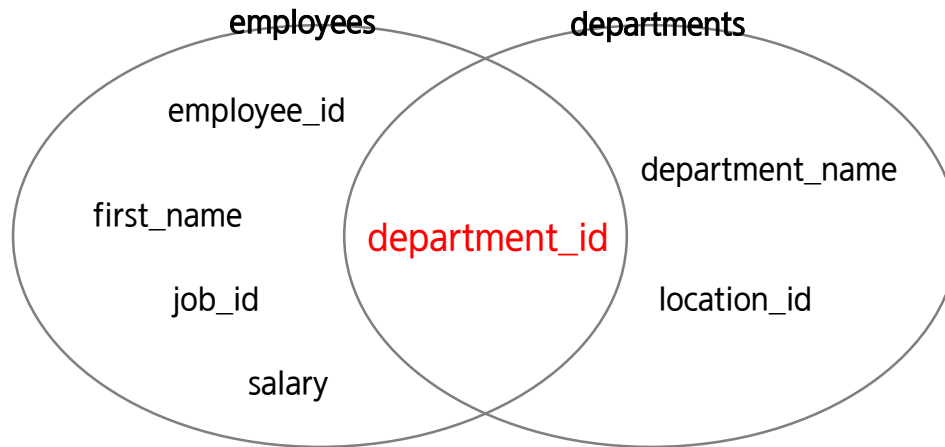
INNER JOIN

삼성 청년 SW 아카데미

✓ JOIN의 종류.

✓ INNER JOIN.

- 가장 일반적인 JOIN의 종류이며 교집합이다.



- 동등 조인(Equi-Join)이라고도 하며, N개의 테이블 조인 시 **N-1개의 조인 조건**이 필요 함.

✓ JOIN의 종류.

✓ INNER JOIN.

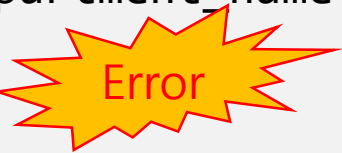
- 형식.
 - > `SELECT COL1, COL2, ..., COLN`
`FROM table1 INNER JOIN table2`
`ON table1.column = table2.column;`
- table에 alias 사용.
 - > `SELECT alias1 .COL1, alias1 .COL2, ..., alias2 .COLN`
`FROM table1 AS alias1 INNER JOIN table2 AS alias2`
`ON alias1.column = alias2.column;`

✓ JOIN의 종류.

✓ INNER JOIN.

- 사번이 100인 사원의 사번, 이름, 급여, 부서번호, 부서이름을 출력.

```
select employee_id, first_name, salary, department_id, department_name
from employees, departments
where employees.department_id = departments.department_id
and employee_id = 100;
```



```
select employee_id, first_name, salary, employees.department_id, department_name
from employees, departments
where employees.department_id = departments.department_id
and employee_id = 100;
```

employee_id	first_name	salary	department_id	department_name
100	Steven	24000.00	90	Executive

join 조건과 일반 조건이 혼합.
두 테이블내 동일 컬럼명 존재
=> table의 이름을 참조.

✓ JOIN의 종류.

✓ INNER JOIN - ON을 이용한 join 조건 지정.

- 사번이 100인 사원의 사번, 이름, 급여, 부서번호, 부서이름을 출력.

```
select e.employee_id, e.first_name, e.salary, e.department_id, d.department_name
from employees e, departments d
where e.department_id = d.department_id
and e.employee_id = 100;
```

alias 사용.

```
select e.employee_id, e.first_name, e.salary, e.department_id, d.department_name
from employees e inner join departments d
on e.department_id = d.department_id
where e.employee_id = 100;
```

join 조건 >> on
일반 조건 >> where.

employee_id	first_name	salary	department_id	department_name
100	Steven	24000.00	90	Executive

✓ JOIN의 종류.

✓ INNER JOIN - USING을 이용한 join 조건 지정.

- 형식.

- > `SELECT COL1, COL2, ..., COLN`

- `FROM table1 JOIN table2`

- `USING (공통column);`

- 사번이 100인 사원의 사번, 이름, 급여, 부서번호, 부서이름을 출력.

```
select e.employee_id, e.first_name, e.salary, e.department_id, d.department_name
from employees e join departments d
using (department_id)
where e.employee_id = 100;
```

using절에서는 table이름이나 alias를 명시하면 error.

employee_id	first_name	salary	department_id	department_name
100	Steven	24000.00	90	Executive

✓ JOIN의 종류.

✓ NATURAL JOIN.

- 형식.

- > `SELECT COL1, COL2, ..., COLN`
`FROM table1 NATURAL JOIN table2;`

- 사번이 100인 사원의 사번, 이름, 급여, 부서번호, 부서이름을 출력.

```
select e.employee_id, e.first_name, e.salary, e.department_id, d.department_name
from employees e natural join departments d
where e.employee_id = 100;
```

employee_id	first_name	salary	department_id	department_name

result row가 없다. 왜 그럴까?

✓ JOIN의 종류.

✓ NATURAL JOIN.

- 형식.

> `SELECT COL1, COL2, ..., COLN`
`FROM table1 NATURAL JOIN table2;`

- 부서번호가 10인 부서의 부서번호, 부서이름, 도시

```
select d.department_id, d.department_name, l.city
from departments d natural join locations l
where d.department_id = 10;
```

department_id	department_name	city
10	Administration	Seattle

이어서..

삼성 청년 SW 아카데미

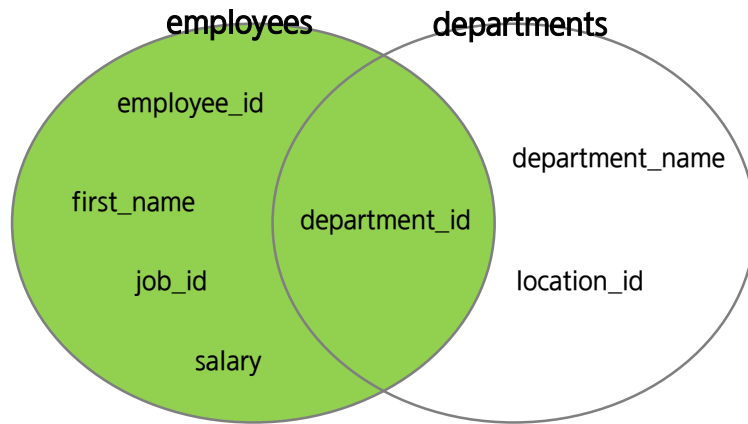
OUTER JOIN

삼성 청년 SW 아카데미

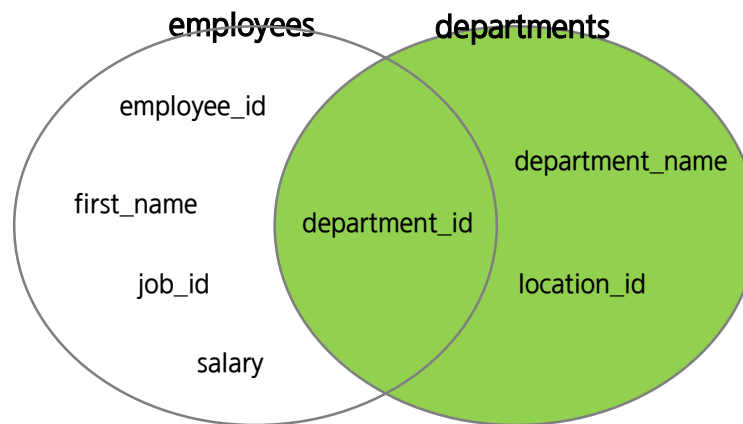
✓ JOIN의 종류.

✓ OUTER JOIN.

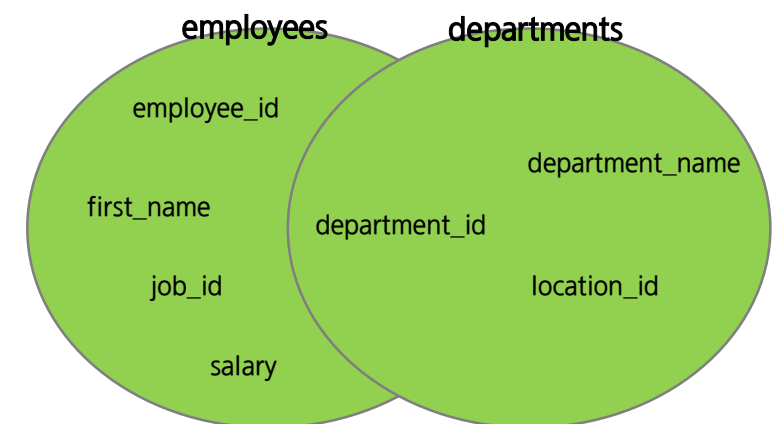
- LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN으로 구분 됨.
- 어느 한쪽 테이블에는 해당하는 데이터가 존재하는데 다른 쪽 테이블에는 데이터가 존재하지 않을 경우 그 데이터가 검색되지 않는 문제점을 해결하기 위해 사용.



LEFT OUTER JOIN



RIGHT OUTER JOIN



FULL OUTER JOIN

✓ JOIN의 종류.

✓ LEFT OUTER JOIN.

- 왼쪽 테이블을 기준으로 JOIN 조건에 일치 하지 않는 데이터까지 출력.

- 형식.

```
> SELECT COL1, COL2, ..., COLN  
FROM table1 LEFT OUTER JOIN table2  
ON or USING;
```

- 회사에 근무하는 **모든 사원**의 사번, 이름, 부서이름

```
select count(employee_id)  
from employees;
```

count(employee_id)
107

회사의 총 사원수

✓ JOIN의 종류.

✓ LEFT OUTER JOIN 예 - 1.

- 회사에 근무하는 모든 사원의 사번, 이름, 부서이름

```
select e.employee_id, e.first_name, d.department_name
from employees e join departments d
using (department_id);
```

employee_id	first_name	department_name
200	Jennifer	Administration
201	Michael	Marketing
202	Pat	Marketing
114	Den	Purchasing
115	Alexander	Purchasing
116	Shelli	Purchasing
117	Sigal	Purchasing
118	Guy	Purchasing
119	Karen	Purchasing

106 row(s) returned

회사에 근무하는 사원은 107명인데 위 join 결과는 106명이 검색 되었다. >> 문제발생.

✓ JOIN의 종류.

✓ LEFT OUTER JOIN 예 - 2.

- 부서가 없는(부서번호가 null) 사원 검색.

```
select employee_id, first_name, department_id
from employees
where department_id is null;
```

employee_id	first_name	department_id
178	Kimberely	NULL

부서가 지정 되지 않은 사원이 1명 존재.
결국 join 조건에 만족하지 않는 데이터가 1건 존재.

- 해결

```
select e.employee_id, e.first_name, d.department_name
from employees e left outer join departments d
using (department_id);
```

107 row(s) returned

employee_id	first_name	department_name
175	Alyssa	Sales
176	Jonathon	Sales
177	Jack	Sales
178	Kimberely	NULL
179	Charles	Sales
180	Winston	Shipping

✓ JOIN의 종류.

✓ RIGHT OUTER JOIN.

- 오른쪽 테이블을 기준으로 JOIN 조건에 일치 하지 않는 데이터까지 출력.

- 형식.

```
> SELECT COL1, COL2, ..., COLN  
FROM table1 RIGHT OUTER JOIN table2  
ON or USING;
```

- 회사에 존재하는 **모든 부서**의 부서이름과 부서에서 근무하는 사원의 사번, 이름.

```
select count(department_id)  
from departments;
```

count(department_id)
27

회사의 총 부서개수

✓ JOIN의 종류.

✓ RIGHT OUTER JOIN.

- 사원이 근무하는 부서수.

```
select count(distinct department_id)
from employees;
```

count(distinct department_id)
11

회사의 부서수는 27개이고 사원이 근무하는 부서는 11개
이므로 16개의 사원이 없는 부서가 존재.

- 사원이 없는 부서의 정보는 출력이 안됨.

```
select d.department_name, e.employee_id, e.first_name
from employees e join departments d
using (department_id);
```

106 row(s) returned

department_name	employee_id	first_name
Administration	200	Jennifer
Marketing	201	Michael
Marketing	202	Pat
Purchasing	114	Den
Purchasing	115	Alexander
Purchasing	116	Shelli
Purchasing	117	Sigal
Purchasing	118	Guy

✓ JOIN의 종류.

✓ RIGHT OUTER JOIN.

- 회사에 존재하는 **모든 부서**의 부서이름과 부서에서 근무하는 사원의 사번, 이름.
- 해결.

```
select d.department_name, e.employee_id, e.first_name
from employees e right outer join departments d
using (department_id);
```

department_name	employee_id	first_name
Finance	112	Jose Man...
Finance	113	Luis
Accounting	205	Shelley
Accounting	206	William
Treasury	NULL	NULL
Corporate Tax	NULL	NULL
Control And Credit	NULL	NULL
Shareholder Serv...	NULL	NULL
Benefits	NULL	NULL
Manufacturing	NULL	NULL
Construction	NULL	NULL

122 row(s) returned

✓ JOIN의 종류.

✓ FULL OUTER JOIN.

- 양쪽 테이블을 기준으로 JOIN 조건에 일치 하지 않는 데이터까지 출력.

- 형식.

```
> SELECT COL1, COL2, ..., COLN  
FROM table1 FULL OUTER JOIN table2  
ON or USING;
```

- MySQL은 지원하지 않음.

이어서..

삼성 청년 SW 아카데미

SELF, Non- Equi JOIN

삼성 청년 SW 아카데미

✓ JOIN의 종류.

✓ SELF JOIN.

- 같은 테이블끼리 JOIN.
- 모든 사원의 사번, 이름, 매니저사번, 매니저이름.

```
select e.employee_id, e.first_name, m.employee_id, m.first_name
from employees e inner join employees m
on e.manager_id = m.employee_id;
```

employee_id	first_name	employee_id	first_name
101	Neena	100	Steven
102	Lex	100	Steven
103	Alexander	102	Lex
104	Bruce	103	Alexander
105	David	103	Alexander
106	Valli	103	Alexander
107	Diana	103	Alexander
108	Nancy	101	Neena

106 row(s) returned

사원의 수는 107명인데 결과는 107명?
그 이유와 모든 사원의 정보를 출력하기 위해서는?

✓ JOIN의 종류.

✓ None-Equi JOIN.

- table의 PK, FK가 아닌 일반 column을 join 조건으로 지정.
- 모든 사원의 사번, 이름, 급여, 급여등급.

```
select e.employee_id, e.first_name, e.salary, s.grade
from employees e join salgrades s
where e.salary >= s.losal
and e.salary <= s.hisal;
```

```
select e.employee_id, e.first_name, e.salary, s.grade
from employees e join salgrades s
where e.salary between s.losal and s.hisal;
```

107 row(s) returned

employee_id	first_name	salary	grade
100	Steven	24000.00	5
101	Neena	17000.00	5
102	Lex	17000.00	5
103	Alexander	9000.00	3
104	Bruce	6000.00	2
105	David	4800.00	2
106	Valli	4800.00	2
...

✓ Quiz.

- 사번이 101인 사원의 근무 이력을 검색.
- 근무 당시의 정보를 아래를 참고하여 출력.
 - 출력 컬럼명 : 사번, 이름, 부서이름, 직급이름, 시작일, 종료일
 - 날짜 형식 : 00.00.00

사번	이름	부서이름	직급이름	시작일	종료일
101	Neena	Accounting	Public Accountant	89.10.21	93.10.27
101	Neena	Accounting	Accounting Manager	93.10.23	97.05.15

- 위의 정보를 출력 하였다면 위의 정보에 현재의 정보를 추가하여 출력.
 - 현재 근무 이력의 시작일은 직전 근무 이력의 종료일 + 1일로 설정.
 - 종료일은 null로 설정.

사번	이름	부서이름	직급이름	시작일	종료일
101	Neena	Accounting	Public Accountant	89.10.21	93.10.27
101	Neena	Accounting	Accounting Manager	93.10.23	97.05.15
101	Neena	Executive	Administration Vice President	97.05.16	NULL

이어서..

삼성 청년 SW 아카데미

SUBQUERY

삼성 청년 SW 아카데미

✓ 서브 쿼리 (Subquery)

- 서브 쿼리(subquery)란 다른 쿼리 내부에 포함되어 있는 SELECT 문을 의미한다.
- 서브 쿼리를 포함하고 있는 쿼리를 외부 쿼리(outer query) 또는 메인 쿼리라고 부르며, 서브 쿼리는 내부 쿼리(inner query)라고도 부른다.
- 서브 쿼리는 비교 연산자의 오른쪽에 기술해야 하고 반드시 괄호(())로 감싸져 있어야만 한다.
- 서브 쿼리의 종류.
 - 중첩 서브 쿼리(Nested Subquery) WHERE 문에 작성하는 서브 쿼리.
 - ① 단일 행
 - ② 복수(다중) 행
 - ③ 다중 컬럼
 - 인라인 뷰 (Inline View) - FROM 문에 작성하는 서브 쿼리.
 - 스칼라 서브 쿼리(Scalar Subquery) - SELECT 문에 작성하는 서브 쿼리.

✓ 서브 쿼리 (Subquery)

- 주의 사항.
 - 서브 쿼리는 반드시 ()로 감싸야 한다.
 - 서브 쿼리는 단일 행 또는 다중 행 비교 연산자와 함께 사용된다.
- 서브 쿼리가 사용이 가능한 곳
 - SELECT
 - FROM
 - WHERE
 - HAVING
 - ORDER BY
 - INSERT문의 VALUES
 - UPDATE문의 SET

✓ 서브 쿼리 (Subquery)

- 사용이유.
- 사번이 100인 사원의 부서이름은?

```
select department_name  
from employees e inner join departments d  
on e.department_id = d.department_id  
where e.employee_id = 100;
```

department_name
Executive

출력해야 할 결과인 department_name은 departments table에 존재 하지만 지문상 알 수 있는 data인 employee_id는 employees table에 존재한다.

subquery를 모른다면 어쩔 수 없이 join을 이용해야 한다.

join의 경우 쿼리가 복잡해 지거나 카테시안곱으로 인한 속도 저하가 올 수 있다. (case by case)

Subquery

✓ 서브 쿼리 (Subquery)

- 해결.

```
select department_id  
from employees  
where employee_id = 100;
```

department_id
90

```
select department_name  
from departments  
where department_id = 90;
```

department_name
Executive

Subquery

✓ 서브 쿼리 (Subquery)

- 해결.

```
select department_id  
from employees  
where employee_id = 100;
```

```
select department_name  
from departments  
where department_id = 90;
```

```
select department_name  
from departments  
where department_id = (  
    select department_id  
    from employees  
    where employee_id = 100  
);
```

department_name
Executive

이어서..

삼성 청년 SW 아카데미

NESTED SUBQUERY

삼성 청년 SW 아카데미

✓ 서브 쿼리 종류.

- Nested Subquery - 단일 행.
 - 서브 쿼리의 결과가 단일행을 리턴.
- 부서가 'seattle'(대소문자 구분)에 있는 부서의 부서 번호, 부서 이름.

```
select department_id, department_name
from departments
where location_id = (
    select location_id
    from locations
    where binary upper(city) = upper('seattle')
);
```

department_id	department_name
10	Administration
30	Purchasing
90	Executive
100	Finance
110	Accounting
120	Treasury

21 row(s) returned

✓ 서브 쿼리 종류.

- Nested Subquery - 단일 행.
 - 서브 쿼리의 결과가 단일행을 리턴.
- 'adam'과 같은 부서에 근무하는 사원의 사번, 이름, 부서번호.

```
select employee_id, first_name, department_id
from employees
where department_id = (
    select department_id
    from employees
    where first_name = 'adam'
);
```

employee_id	first_name	department_id
120	Matthew	50
121	Adam	50
122	Payam	50
123	Shanta	50
124	Kevin	50
125	Julia	50
126	Irene	50

45 row(s) returned

✓ 서브 쿼리 종류.

- Nested Subquery - 단일 행.
 - 서브 쿼리의 결과가 단일행을 리턴.
- 전체 사원의 평균 급여보다 많이 받는 사원의 사번, 이름, 급여.
- 급여 순 정렬

```
select employee_id, first_name, salary
from employees
where salary > (select avg(salary) from employees)
order by salary desc;
```

employee_id	first_name	salary
100	Steven	24000.00
101	Neena	17000.00
102	Lex	17000.00
145	John	14000.00
146	Karen	13500.00
201	Michael	13000.00
108	Nancy	12000.00

51 row(s) returned

✓ 서브 쿼리 종류.

- Nested Subquery - 다중 행.
 - 서브 쿼리의 결과가 다중행을 리턴 : **IN**, ANY, ALL
- 근무 도시가 'seattle'(대소문자 구분)인 사원의 사번, 이름.

```
select employee_id, first_name
from employees
where department_id in (
    select department_id
    from departments
    where location_id = (
        select location_id
        from locations
        where binary upper(city) = upper('seattle')
    )
);
```

employee_id	first_name
200	Jennifer
114	Den
115	Alexander
116	Shelli
117	Sigal
118	Guy
119	Karen
100	...

18 row(s) returned

✓ 서브 쿼리 종류.

- Nested Subquery - 다중 행.
 - 서브 쿼리의 결과가 다중행을 리턴 : IN, **ANY**, ALL
 - 적어도 하나만 만족하면 true.
- 모든 사원 중 적어도(최소급여자보다) 30번 부서에서 근무하는 사원의 급여보다 많이 받는 사원의 사번, 이름, 급여, 부서번호

```
select employee_id, first_name, salary, department_id
from employees
where salary > any (
    select salary
    from employees
    where department_id = 30
)
order by salary;
```

96 row(s) returned

employee_id	first_name	salary	department_id
118	Guy	2600.00	30
143	Randall	2600.00	50
198	Donald	2600.00	50
199	Douglas	2600.00	50
139	John	2700.00	50
126	Irene	2700.00	50
130	Mozhe	2800.00	50

✓ 서브 쿼리 종류.

- Nested Subquery - 다중 행.
 - 서브 쿼리의 결과가 다중행을 리턴 : IN, ANY, **ALL**
 - 모두 만족하면 true.
- 30번 부서에서 근무하는 모든(최대급여자보다) 사원들보다 급여를 많이 받는 사원의 사번, 이름, 급여, 부서번호.

```
select employee_id, first_name, salary, department_id
from employees
where salary > all (
    select salary
    from employees
    where department_id = 30
)
order by salary;
```

10 row(s) returned

employee_id	first_name	salary	department_id
168	Lisa	11500.00	80
108	Nancy	12000.00	100
147	Alberto	12000.00	80
205	Shelley	12000.00	110
201	Michael	13000.00	20
146	Karen	13500.00	80
145	John	14000.00	80
104	N	17000.00	80

✓ 서브 쿼리 종류.

- Nested Subquery - 다중 열.
 - 서브 쿼리의 결과가 다중열을 리턴.
- 커미션을 받는 직원중 매니저 사번이 148인 직원의 급여와 부서번호가 일치하는 직원의 사번, 이름

```
select employee_id, first_name
from employees
where (salary, department_id) in (
    select salary, department_id
    from employees
    where commission_pct is not null
    and manager_id = 148
);
```

employee_id	first_name
168	Lisa
150	Peter
156	Janette
169	Harrison
170	Tayler
171	William
172	Elizabeth
173	Sundita

8 row(s) returned

이어서..

삼성 청년 SW 아카데미

INLINE VIEW

삼성 청년 SW 아카데미

✓ 서브 쿼리 종류.

- 인라인 뷰(Inline View).
 - FROM절에 사용되는 서브 쿼리를 인라인 뷰(Inline View)라 한다.
 - 서브 쿼리가 FROM절에 사용되면 뷰(View)처럼 결과가 동적으로 생성된 테이블로 사용 가능.
 - 임시적인 뷰이기 때문에 데이터베이스에는 저장되지 않는다.
 - 동적으로 생성된 테이블이기 때문에 column을 자유롭게 참조 가능.

✓ 서브 쿼리 종류.

- 인라인 뷰(Inline View).
- 모든 사원의 평균 급여보다 적게 받는 사원들과 같은 부서에서 근무하는 사원의 사번, 이름, 급여, 부서번호

```
select e.employee_id, e.first_name, e.salary, e.department_id
from (
    select distinct department_id
    from employees
    where salary < (select avg(salary) from employees)
) d join employees e
on d.department_id = e.department_id;
```

employee_id	first_name	salary	department_id
200	Jennifer	4400.00	10
201	Michael	13000.00	20
202	Pat	6000.00	20
114	Den	11000.00	30
115	Alexander	3100.00	30
116	Shelli	2900.00	30
117	Sigal	2800.00	30
118	G	3600.00	30

93 row(s) returned

✓ 서브 쿼리 종류.

- 인라인 뷰(Inline View) - TopN 질의.
- 모든 사원의 사번, 이름, 급여를 출력.(단 아래의 조건 참조)
 - 사원 정보를 급여순으로 정렬.
 - 한 페이지당 5명이 출력.
 - 현재페이지가 3페이지라고 가정. (급여 순 11등 ~ 15등까지 출력)

```
set @pageno = 3; -- 변수 설정.
```

```
select b.rn, b.employee_id, b.first_name, b.salary
from (
    select @rownum := @rownum + 1 as rn, a.*
    from (
        select employee_id, first_name, salary
        from employees
        order by salary desc
    ) a, (select @rownum := 0) tmp
    ) b
where b.rn > (@pageno * 5 - 5) and b.rn <= (@pageno * 5);
```

rn	employee_id	first_name	salary
11	114	Den	11000.00
12	148	Gerald	11000.00
13	174	Ellen	11000.00
14	149	Eleni	10500.00
15	162	Clara	10500.00

✓ 서브 쿼리 종류.

- 인라인 뷰(Inline View) - limit 활용 (MySQL).
- 모든 사원의 사번, 이름, 급여를 출력.(단 아래의 조건 참조)
 - 사원 정보를 급여순으로 정렬.
 - 한 페이지당 5명이 출력.
 - 현재페이지가 3페이지라고 가정. (급여 순 11등 ~ 15등까지 출력)

```
select employee_id, first_name, salary
from employees
order by salary desc limit 10, 5;
```

```
select a.*
from (
    select @rownum := @rownum + 1 as rn, employee_id, first_name, salary
    from employees e, (select @rownum := 0) tmp
    order by salary desc
) a limit 10, 5;
```

rn	employee_id	first_name	salary
11	114	Den	11000.00
12	148	Gerald	11000.00
13	174	Ellen	11000.00
14	149	Eleni	10500.00
15	162	Clara	10500.00

이어서..

삼성 청년 SW 아카데미

SCALAR SUBQUERY

삼성 청년 SW 아카데미

✓ 서브 쿼리 종류.

- 스칼라 서브 쿼리 (Scalar Subquery).
 - SELECT 절에 있는 서브 쿼리.
 - 한 개의 행만 반환.
- 직급 아이디가 IT_PROG인 사원의 사번, 이름, 직급 아이디, 부서이름

```
select e.employee_id, e.first_name, job_id,  
       (select department_name from departments d  
        where e.department_id = d.department_id) as department_name  
from employees e  
where job_id = 'IT_PROG';
```

employee_id	first_name	department_name
103	Alexander	IT
104	Bruce	IT
105	David	IT
106	Valli	IT
107	Diana	IT

5 row(s) returned

✓ 서브 쿼리 종류.

- 스칼라 서브 쿼리 (Scalar Subquery).
- 60번 부서에 근무하는 사원의 사번, 이름, 급여, 부서번호, 60번부서의 평균급여

```
select e.employee_id, e.first_name, salary, department_id,  
       (select avg(salary) from employees where department_id = 60) as avg60  
from employees e  
where department_id = 60;
```

employee_id	first_name	salary	department_id	avg60
103	Alexander	9000.00	60	5760.000000
104	Bruce	6000.00	60	5760.000000
105	David	4800.00	60	5760.000000
106	Valli	4800.00	60	5760.000000
107	Diana	4200.00	60	5760.000000

5 row(s) returned

✓ 서브 쿼리 종류.

- 스칼라 서브 쿼리 (Scalar Subquery).
- 부서번호가 50인 부서의 총급여, 60인 부서의 평균급여, 90인 부서의 최고급여, 90인 부서의 최저급여

```
select
    (select sum(salary) from employees where department_id = 50) sum50,
    (select avg(salary) from employees where department_id = 60) avg60,
    (select max(salary) from employees where department_id = 90) max90,
    (select min(salary) from employees where department_id = 90) min90
from dual;
```

sum50	avg60	max90	min90
156400.00	5760.000000	24000.00	17000.00

이어서..

삼성 청년 SW 아카데미

SUBQUERY 활용

삼성 청년 SW 아카데미

✓ 서브 쿼리 (Subquery)

- 서브 쿼리를 이용한 **CREATE**, INSERT, UPDATE, DELETE.
- employees table을 emp_copy라는 이름으로 복사(컬럼 이름 동일).

```
create table emp_copy  
select * from employees;
```

- employees table의 구조만 emp_blank라는 이름으로 생성(컬럼 이름 동일).

```
create table emp_blank  
select * from employees  
where 1 = 0;
```

- 50번 부서의 사번(eid), 이름(name), 급여(sal), 부서번호(did)만 emp50이라는 이름으로 생성.

```
create table emp50  
select employee_id eid, first_name name, salary sal, department_id did  
from employees  
where department_id = 50;
```

eid	name	sal	did
120	Matthew	8000.00	50
121	Adam	8200.00	50
122	Jayson	7000.00	50

✓ 서브 쿼리 (Subquery)

- 서브 쿼리를 이용한 CREATE, INSERT, UPDATE, DELETE.
- employees table에서 부서번호가 80인 사원의 모든 정보를 emp_blank에 insert.

```
insert into emp_blank  
select * from employees  
where department_id = 80;
```

employee_id	first_name	last_name	email	phone_number	hire_date	job	34 row(s) returned
145	John	Russell	JRUSSEL	011.44.1344.429268	1996-10-10	SA	
146	Karen	Partners	KPARTNER	011.44.1344.467268	1997-04-05	SA	
147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	1997-05-10	SA	
148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	1999-10-15	SA	
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	2000-12-12	SA	
150	Peter	Tucker	PTUCKER	011.44.1344.129268	1997-11-21	SA	
151	David	Bernstein	DBERNSTE	011.44.1344.245268	1997-12-20	SA	

✓ 서브 쿼리 (Subquery)

- 서브 쿼리를 이용한 CREATE, INSERT, **UPDATE**, DELETE.
- employees table의 모든 사원의 평균 급여보다 적게 받는 emp50 table의 사원의 급여를 500 인상.

```
update emp50  
set sal = sal + 500  
where sal < (select avg(salary) from employees);
```

```
41 row(s) affected Rows matched: 41 Changed: 41 Warnings: 0
```


✓ 서브 쿼리 (Subquery)

- 서브 쿼리를 이용한 CREATE, INSERT, UPDATE, DELETE.
- employees table의 모든 사원의 평균 급여보다 적게 받는 emp50 table의 사원은 퇴사.

```
delete from emp50  
where sal < (select avg(salary) from employees);
```

41 row(s) affected

내일 방송에서 만나요!

삼성 청년 SW 아카데미