


sql 독학 강의 # mysql join (정의 및 종류) 11편 -sTricky

sTricky 2020. 4. 16. 15:56

sql 독학 강의 # mysql join (정의 및 종류) 11편 -sTricky



SQL 독학 강의
JOIN 의
정의 및 종류
알아보기

[컨텐츠 index](#)

0. join 수업용 데이터 생성

1. join 이란 무엇인가?
2. join의 종류는 무엇이 있을까?

안녕하세요.

오늘부터 mysql에서 join 하는 방법에 대해서 강의를 진행하겠습니다.

사실 join 부분을 sql을 작성할 때 어려워하시는 분들이 많이 계신데, 조금만 마음을 편안하게 먹고, 차근차근 보시다 보면 이해 하 실수 있도록 설명하겠습니다.

어렵지 않으니! 아래 내용을 하나하나 차근차근 읽어 보시면서 따라오시기 바랍니다.

#이전 강의 보러 가기#

[2020/04/13 - \[Database/sql 강의\] - sql 독학 강의 # 복수 행\(window\) 함수 잘 사용 하기\(group by\) 10편 -sTricky](#)

sql 독학 강의 # 복수 행(window) 함수 잘 사용 하기(group by) ...

stricky.tistory.com

0. join 수업용 데이터 생성

join과 관련된 강의를 진행하기에 앞서서 강의 도중 사용할 데이터를 미리 생성하겠습니다. 중간에 추가가 될 수 있겠지만 우선 아래와 같이 데이터를 생성합니다.

우선 테이블을 생성합니다.

```
create table class.student (  
  student_id int(10) comment '학생번호',
```

```

major_id int(10) comment '학과ID',
bl_prfs_id int(10) comment '담당교수ID',
name varchar(20) comment '학생이름',
tel varchar(15) comment '학생연락처'
);

create table class.professor (
prfs_id int(10) comment '교수ID',
bl_major_id int(10) comment '소속학과ID',
name varchar(20) comment '교수이름',
tel varchar(15) comment '교수연락처'
);

create table class.major (
major_id int(10) comment '학과ID',
major_title varchar(30) comment '학과명',
major_prfs_cnt int(5) comment '학과소속교수수',
major_student_cnt int(5) comment '학과소속학생수',
tel varchar(15) comment '학과사무실연락처'
);

```

데이터를 입력합니다.

```

INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1001, 9901, 7029901,
'한지호', '01098447362');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1002, 9902, 7029902,
'김은숙', '01023456787');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1003, 9903, 7039903,
'강경호', '01092938476');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1004, 9904, 7049904,
'민현민', '01088786623');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1005, 9905, 7059905,
'조승우', '01092877795');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1006, 9901, 7069901,
'이남철', '01045671234');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1007, 9902, 7079902,
'이강철', '01021213434');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1008, 9903, 7089903,
'조민수', '01098937262');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1009, 9904, 7099904,
'박찬경', '01029884432');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1010, 9905, 7109905,
'이도경', '01029385647');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1011, 9901, 7019901,
'이만호', '01099996453');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1012, 9902, 7029902,
'김효민', '01092887666');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1013, 9903, 7039903,
'최효성', '01098999933');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1014, 9904, 7049904,
'우민국', '01087651112');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1015, 9905, 7059905,
'지대한', '01093934848');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1016, 9901, 7069901,
'한나름', '01023329882');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1017, 9902, 7079902,
'유육경', '01099881111');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1018, 9903, 7089903,
'조민경', '01023311120');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1019, 9904, 7099904,
'경지수', '01029100293');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1020, 9905, 7109905,
'오종환', '01098882226');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1021, 9901, 7019901,
'조형민', '01098909876');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1022, 9902, 7029902,
'이수강', '01099992222');

```

```

INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1023, 9903, 7039903,
'서민호', '01092997654');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1024, 9904, 7049904,
'박효숙', '01022293332');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1025, 9905, 7059905,
'남궁옥경', '01099938475');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1026, 9901, 7069901,
'피경남', '01029222233');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1027, 9902, 7079902,
'고주경', '01099226655');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1028, 9903, 7089903,
'하지만', '01022228965');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1029, 9904, 7099904,
'기지호', '01012090912');
INSERT INTO class.student (student_id, major_id, bl_prfs_id, name, tel) VALUES (1030, 9905, 7109905,
'박민호', '01074746363');

INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7019901, 9901, '김보경', '0234
45678');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7029902, 9902, '조숙', '023446
789');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7039903, 9903, '이호', '023449
584');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7049904, 9904, '박철남', '0234
49588');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7059905, 9905, '이만기', '0234
43443');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7069901, 9901, '강조교', '0234
49994');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7079902, 9902, '이희숙', '0234
43321');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7089903, 9903, '소머리', '0234
40123');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7099904, 9904, '두수위', '0234
43327');
INSERT INTO class.professor (prfs_id, bl_major_id, name, tel) VALUES (7109905, 9905, '지만래', '0234
49995');

INSERT INTO class.major (major_id, major_title, major_prfs_cnt, major_student_cnt, tel) VALUES (9901
, '컴퓨터공학과', 7, 123, '023454321');
INSERT INTO class.major (major_id, major_title, major_prfs_cnt, major_student_cnt, tel) VALUES (9902
, '아동보육학과', 8, 345, '023456676');
INSERT INTO class.major (major_id, major_title, major_prfs_cnt, major_student_cnt, tel) VALUES (9903
, '국문학과', 6, 213, '023456567');
INSERT INTO class.major (major_id, major_title, major_prfs_cnt, major_student_cnt, tel) VALUES (9904
, '경제학과', 5, 432, '023456987');
INSERT INTO class.major (major_id, major_title, major_prfs_cnt, major_student_cnt, tel) VALUES (9905
, '사회복지학과', 9, 312, '023454534');

```

이젠 이 데이터를 이용해서 강의를 진행하겠습니다.

1. join이란 무엇인가?

우리가 흔히 아는 **oracle, mysql, mariadb, ms-sql, postgres** 등등은 모두 **RDBMS**입니다. DBMS라는 말은 많이 들어보셨을 겁니다.

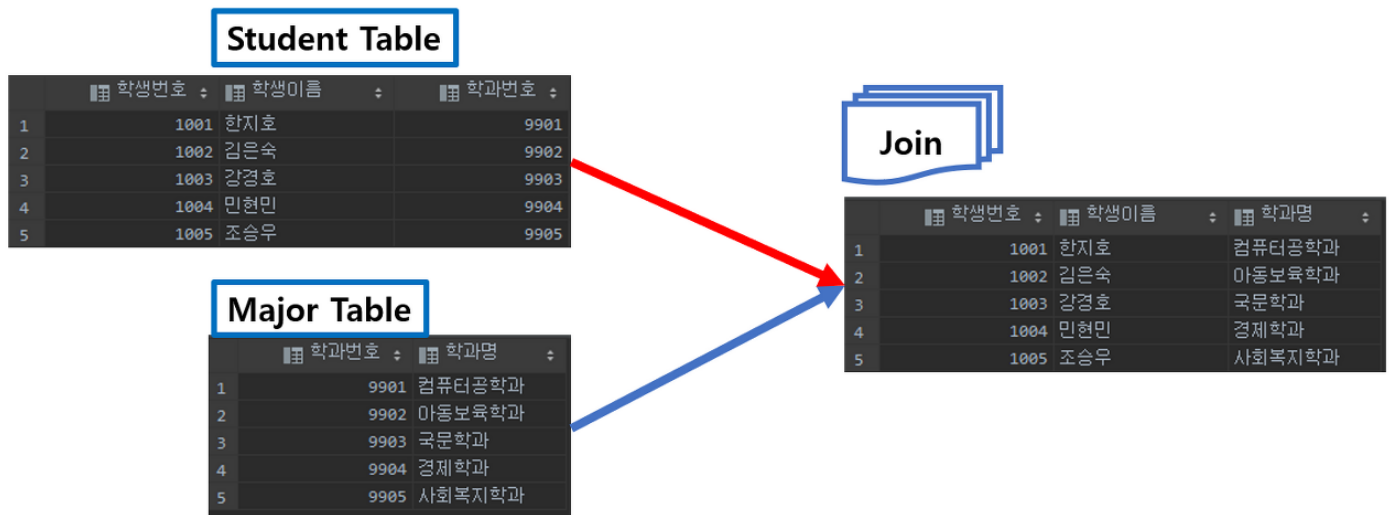
맞습니다! 바로 **DataBase Management System**의 약자로서 데이터베이스를 관리하는 시스템이라는 뜻입니다. 그렇다면 **RDBMS**는요? **Relational DataBase Management System**의 약어로서 관계형 데이터베이스

스를 관리하는 시스템이라는 말이 되겠죠.

여기서 관계형이란, 말 그대로 데이터베이스 내에 있는 테이블이나 스키마들이 서로 관계를 가지고 있다는 뜻입니다. 그렇다면 이러한 관계를 이용해서 우리는 SQL을 작성하기도 해야 할 텐데요, 이럴 때 사용하는 게 바로 join이 됩니다.

join을 사용해서 여러 테이블이나 스키마에 분산되어 있는 데이터를 하나의 view로 출력하게 하는 것입니다.

아래 그림을 참고하겠습니다.



join 설명

위 그림과 같이 student 테이블과 major 테이블에 각각 따로 들어가 있는 데이터를 오른쪽에 join 로직을 거치면 하나의 테이블처럼 데이터를 볼 수 있게 됩니다.

위 그림에서는 2개의 테이블만 join을 했지만 3,4개 그 이상 join이 가능합니다. 추후 포스팅에서 2개를 초과하는 테이블의 join을 쉽게 하는 방법도 알려 드리도록 하겠습니다.

2. join의 종류는 무엇이 있을까?

DBMS에서 join을 하는 데 있어서 몇 가지 방법이 존재합니다. 각 join들의 특징을 잘 알고 계셔야지 자신이 원하는 결과를 출력하는데 유리합니다. 아래, 각 join의 종류별로 간단한 설명을 드리겠습니다.

카티션곱 join

카티션곱 join이란 테이블들을 join 할 때 join 조건을 기술하지 않고 하는 join을 말합니다. 카티션곱 join의 결과는 **두 테이블의 row 건수를 서로 곱한 것만큼의 결과를 출력**합니다. 흔히 업무에서는 잘 사용되지 않으나, 데이터를 많이 불러야 하거나, 특정한 조건 안에서 필요할 때가 있습니다.

학생 테이블(16건)

STUDNO	NAME	USERID	DEPTNO
10101	전인하	jun123	101
20101	이동훈	Dals	201
.....
10203	윤진욱	Samba7	102
10107	이광훈	huriky	101

부서 테이블(7건)

DEPTNO	DNAME	LOC
101	컴퓨터공학과	1호관
102	멀티미디어학과	2호관
201	전자공학과	3호관
202	기계공학과	4호관
100	정보미디어학부	
200	메카트로닉스학부	
10	공과대학	

카티션 곱

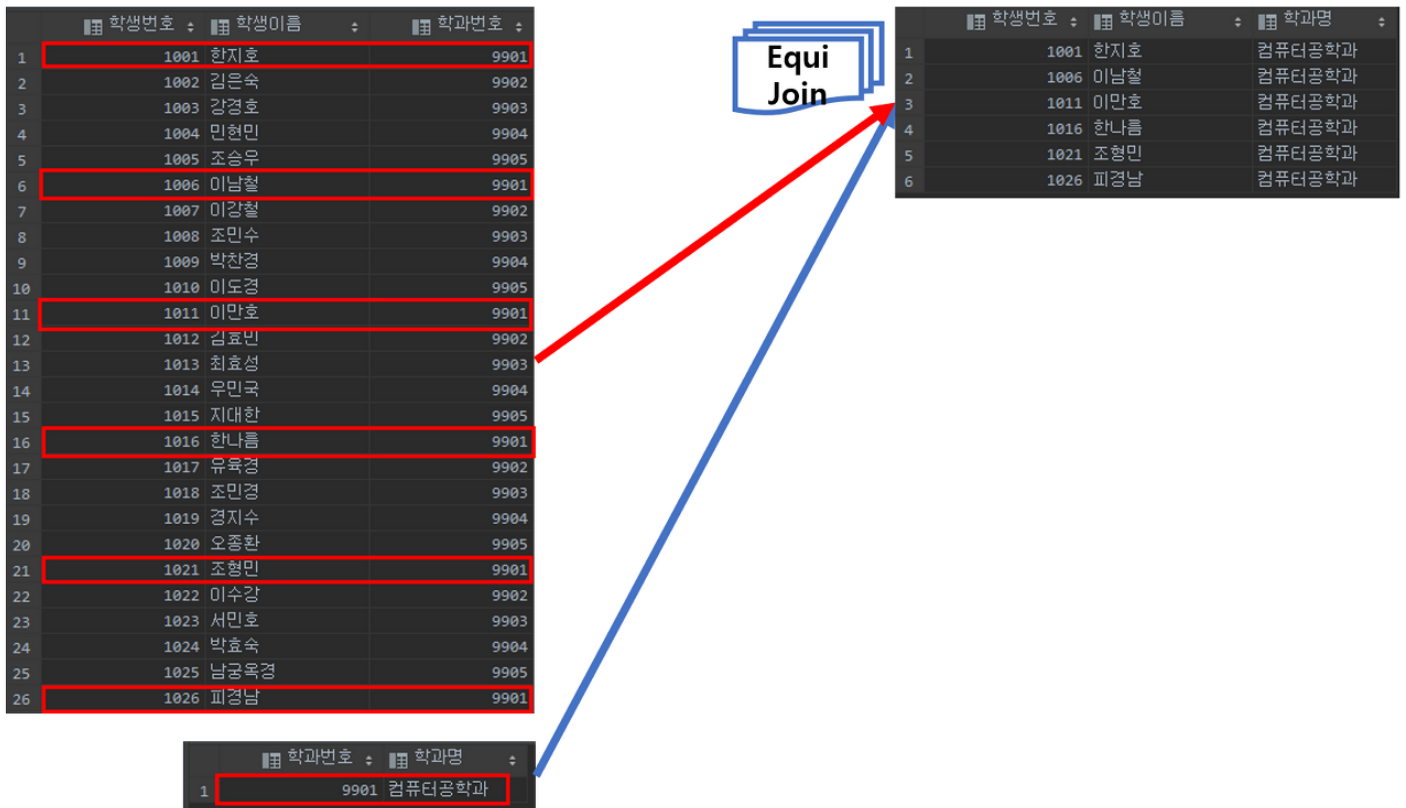
```
SELECT studno, name, s.deptno, d.deptno, dname
FROM student s, department d
```

카티션 곱 결과 (112건) 371

STUDNO	NAME	S.DEPTNO	D.DEPTNO	DNAME
10101	전인하	101	101	컴퓨터공학과
20101	이동훈	201	101	컴퓨터공학과
10102	박미경	101	101	컴퓨터공학과
10103	김영균	102	101	컴퓨터공학과
20102	박동진	201	101	컴퓨터공학과
10201	김진영	102	101	컴퓨터공학과
10104	지은경	101	101	컴퓨터공학과
10202	오유석	102	101	컴퓨터공학과
10203	하나리	102	101	컴퓨터공학과
10105	임유진	101	101	컴퓨터공학과
10106	서재진	101	101	컴퓨터공학과
10204	윤진욱	102	101	컴퓨터공학과
10107	이광훈	101	101	컴퓨터공학과
20103	김진경	201	101	컴퓨터공학과
20104	조명훈	201	101	컴퓨터공학과
10108	류민정	101	101	컴퓨터공학과
10101	전인하	101	102	멀티미디어학과
20101	이동훈	201	102	멀티미디어학과
.....
10101	전인하	101	201	전자공학과
20101	이동훈	201	201	전자공학과
.....

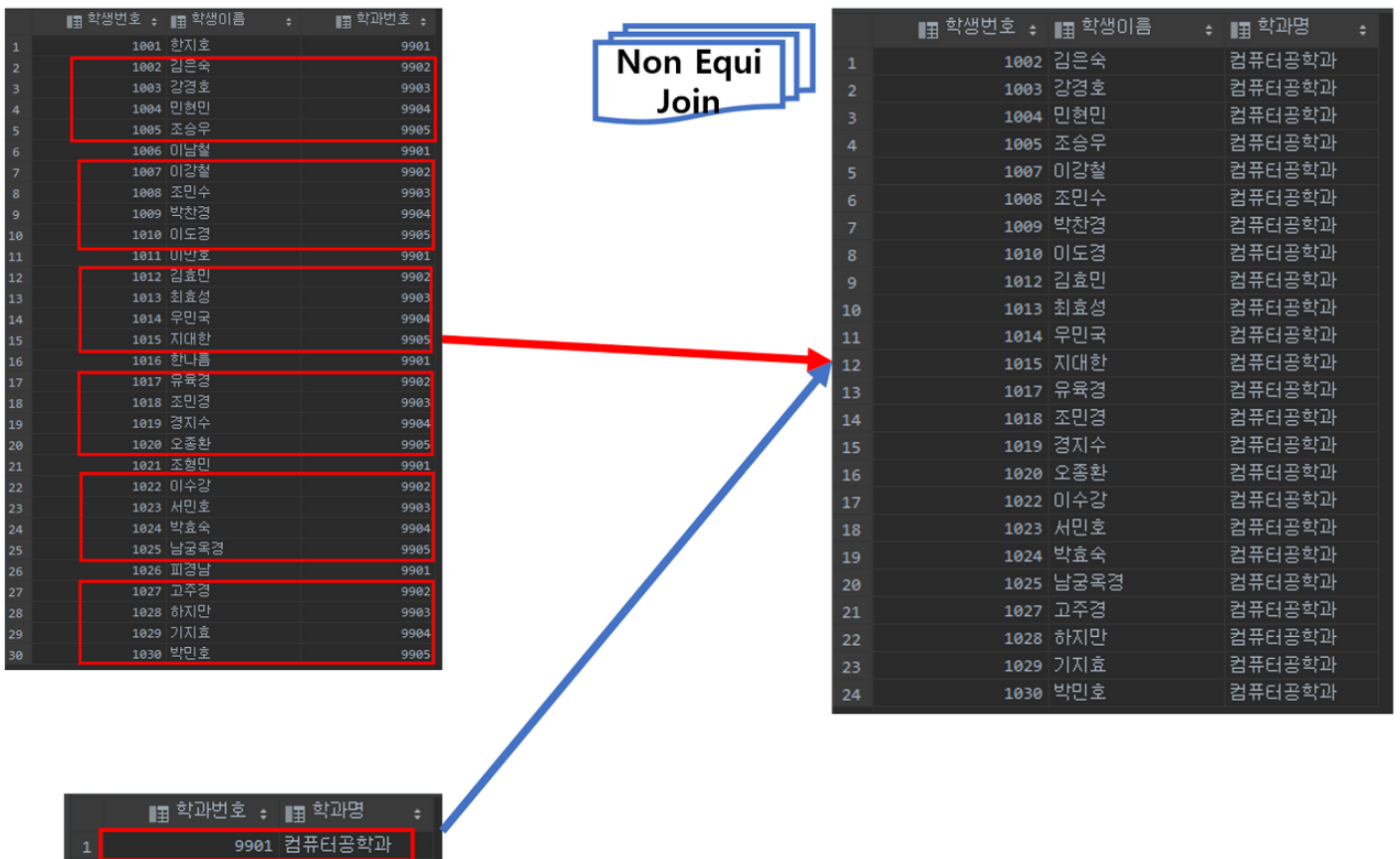
Equi join

두 테이블을 서로 join 한다고 하면, 양쪽 테이블의 어떤 칼럼에 같은 값이 존재할 때 이것을 **Equal 연산자(=)**를 이용하여 **양쪽에 다 존재하는 값만 결과로 출력하는 join**입니다. 가장 보편적인 join 방법입니다. **inner join(이너 조인)** 이라고도 불립니다.



Non Equi join

Equi join과 반대 개념입니다. 두 테이블을 서로 join 할 때, 서로 다른 값을 가지거나, 한쪽 데이터가 다른 쪽 테이블의 데이터 범위 내에 있는 것만 출력을 원할 때 쓰는 join 방법입니다. Non Equi join 역시 **inner join(이너 조인)**에 속합니다.



Outer join

Outer join은 **left outer join**, **right outer join**, **full outer join**으로 구분됩니다. left outer join, right outer join의 경우 어느 한쪽의 데이터를 모두 출력 한 뒤에 조건에 맞는 데이터만 다른 쪽에 출력을 하는 것을 말합니다. 조건에 맞지 않은 데이터 옆에는 null이 표시됩니다.

Left outer Join

	학생번호	학생이름	학과번호
1	1001	한지호	9901
2	1002	김은숙	9902
3	1003	강경호	9903
4	1004	민현민	9904
5	1005	조승우	9905
6	1006	이남철	9901
7	1007	이강철	9902
8	1008	조민수	9903
9	1009	박찬경	9904
10	1010	이도경	9905
11	1011	이만호	9901
12	1012	김효민	9902
13	1013	최효성	9903
14	1014	우민국	9904
15	1015	지대환	9905
16	1016	한나름	9901
17	1017	유육경	9902
18	1018	조민경	9903
19	1019	경지수	9904
20	1020	오종환	9905
21	1021	조형민	9901
22	1022	이수강	9902
23	1023	서민호	9903
24	1024	박효숙	9904
25	1025	남궁옥경	9905
26	1026	피경남	9901

	학과번호	학과명
1	9901	컴퓨터공학과

	학생번호	학생이름	학과명
1	1001	한지호	컴퓨터공학과
2	1002	김은숙	<null>
3	1003	강경호	<null>
4	1004	민현민	<null>
5	1005	조승우	<null>
6	1006	이남철	컴퓨터공학과
7	1007	이강철	<null>
8	1008	조민수	<null>
9	1009	박찬경	<null>
10	1010	이도경	<null>
11	1011	이만호	컴퓨터공학과
12	1012	김효민	<null>
13	1013	최효성	<null>
14	1014	우민국	<null>
15	1015	지대환	<null>
16	1016	한나름	컴퓨터공학과
17	1017	유육경	<null>
18	1018	조민경	<null>
19	1019	경지수	<null>
20	1020	오종환	<null>
21	1021	조형민	컴퓨터공학과
22	1022	이수강	<null>
23	1023	서민호	<null>
24	1024	박효숙	<null>
25	1025	남궁옥경	<null>
26	1026	피경남	컴퓨터공학과
27	1027	고주경	<null>
28	1028	하지만	<null>
29	1029	기지호	<null>
30	1030	박민호	<null>

Self join

Self join은 한 테이블이 자기 자신과 join을 다시 하는 경우를 말합니다. 아주 일반적인 경우는 아니지만 꼭 필요한 경우가 있습니다. 일반적인 사용법은 다른 join과 같습니다.