

제 4 장

조인(JOIN)과 부속질의어(subQuery)





테이블 생성

■ CREATE TABLE : 테이블을 생성한다.

CREATE TABLE 테이블이름

(

필드 정보,

필드 정보,

....

);

■ 필드는 다음 형식으로 정의한다.

- 필드이름 타입 [제약]

■ tCity 테이블 생성 스크립트

CREATE TABLE tCity

(

name CHAR(10) PRIMARY KEY,

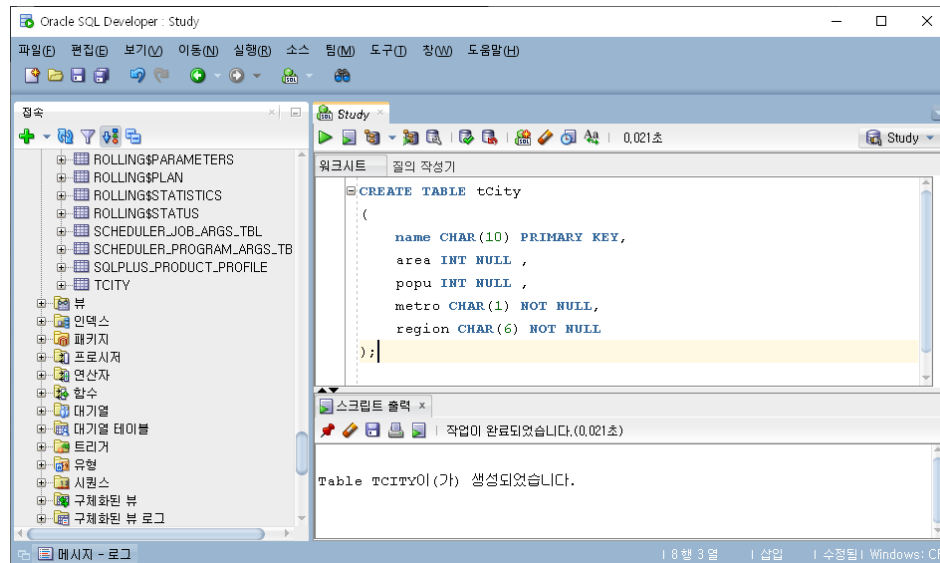
area INT NULL ,

popu INT NULL ,

metro CHAR(1) NOT NULL,

region CHAR(6) NOT NULL

);



COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 NAME	CHAR(10 BYTE)	No	(null)	1 (null)	
2 AREA	NUMBER(38,0)	Yes	(null)	2 (null)	
3 POPU	NUMBER(38,0)	Yes	(null)	3 (null)	
4 METRO	CHAR(1 BYTE)	No	(null)	4 (null)	
5 REGION	CHAR(6 BYTE)	No	(null)	5 (null)	



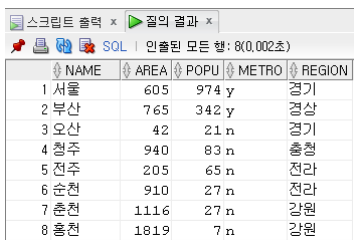
데이터 입력

- 레코드를 추가할 때는 INSERT 명령을 사용하며 VALUES 절에 레코드의 각 필드값을 콤마로 구분하여 나열한다.

```
INSERT INTO tCity VALUES ('서울',605,974,'y','경기');
INSERT INTO tCity VALUES ('부산',765,342,'y','경상');
INSERT INTO tCity VALUES ('오산',42,21,'n','경기');
INSERT INTO tCity VALUES ('청주',940,83,'n','충청');
INSERT INTO tCity VALUES ('전주',205,65,'n','전라');
INSERT INTO tCity VALUES ('순천',910,27,'n','전라');
INSERT INTO tCity VALUES ('춘천',1116,27,'n','강원');
INSERT INTO tCity VALUES ('홍천',1819,7,'n','강원');
```

- 잘 삽입되었는지 확인한다.

```
SELECT * FROM tCity;
```



스크립트 출력 x | 실행 결과 x

SQL | 인출된 모든 행: 8(0.002초)

	NAME	AREA	POPULATION	METRO	REGION
1	서울	605	974 y		경기
2	부산	765	342 y		경상
3	오산	42	21 n		경기
4	청주	940	83 n		충청
5	전주	205	65 n		전라
6	순천	910	27 n		전라
7	춘천	1116	27 n		강원
8	홍천	1819	7 n		강원

- 오라클은 임시영역에 변경 사항을 관리한다.
- COMMIT 명령으로 삽입을 확정한다.



SQL 삭제

- 삭제 명령은 **DROP**이며 삭제할 개체 이름만 밝히면 된다.

```
DROP TABLE tCity;
```

- 확인 과정 없이 즉시 삭제한다.

- 언제든지 새로 만들 수 있다.

```
CREATE TABLE tCity
```

```
....
```

```
INSERT INTO tCity VALUES ('서울',605,974,'y','경기');
```

```
....
```

```
COMMIT;
```

- 실무 환경에서 **DROP** 명령은 위험한 명령이다.



SQL 쿼리를 알아야 하는 이유

- 스크립트는 스스로 문서화를 겸한다.
- 한줄 주석은 --, 블록 주석은 /* */로 작성한다.

/* 대한민국의 도시 정보를 저장하는 테이블이다.

작성자 : 김미남 */

CREATE TABLE tCity

(

name CHAR(10) PRIMARY KEY,

-- 도시의 이름

area INT NULL ,

-- 면적. 제곱킬로미터 단위

popu INT NULL ,

-- 인구수. 만명 단위

metro CHAR(1) NOT NULL,

-- 광역시 여부. Y/N

region CHAR(6) NOT NULL

-- 소속 지역

);

- 스크립트는 개발자끼리 대화하는 가장 명확한 수단이다.
- 데이터 관리에 대한 모든 일을 다 처리할 수 있고 표준화되어 있다.



SQL tStaff

- 실습을 위해 여러 형태의 예제 테이블이 필요하다.
- tStaff은 직원의 신상 명세를 저장한다.

```
CREATE TABLE tStaff
(
    name CHAR (15) PRIMARY KEY,
    depart CHAR (10) NOT NULL,
    gender CHAR(3) NOT NULL,
    joindate DATE NOT NULL,
    grade CHAR(10) NOT NULL,
    salary INT NOT NULL,
    score DECIMAL(5,2) NULL
);

INSERT INTO tStaff VALUES ('김유신','총무부','남','2000-2-3','이사',420,88.8);
INSERT INTO tStaff VALUES ('유관순','영업부','여','2009-3-1','과장',380,NULL);
INSERT INTO tStaff VALUES ('안중근','인사과','남','2012-5-5','대리',256,76.5);
INSERT INTO tStaff VALUES ('윤봉길','영업부','남','2015-8-15','과장',350,71.25);
INSERT INTO tStaff VALUES ('강감찬','영업부','남','2018-10-9','사원',320,56.0);
INSERT INTO tStaff VALUES ('정몽주','총무부','남','2010-9-16','대리',370,89.5);
INSERT INTO tStaff VALUES ('허난설헌','인사과','여','2020-1-5','사원',285,44.5);
INSERT INTO tStaff VALUES ('신사임당','영업부','여','2013-6-19','부장',400,92.0);
INSERT INTO tStaff VALUES ('성삼문','영업부','남','2014-6-8','대리',285,87.75);
INSERT INTO tStaff VALUES ('논개','인사과','여','2010-9-16','대리',340,46.2);
INSERT INTO tStaff VALUES ('황진이','인사과','여','2012-5-5','사원',275,52.5);
INSERT INTO tStaff VALUES ('이율곡','총무부','남','2016-3-8','과장',385,65.4);
INSERT INTO tStaff VALUES ('이사부','총무부','남','2000-2-3','대리',375,50);
INSERT INTO tStaff VALUES ('안창호','영업부','남','2015-8-15','사원',370,74.2);
INSERT INTO tStaff VALUES ('을지문덕','영업부','남','2019-6-29','사원',330,NULL);
INSERT INTO tStaff VALUES ('정약용','총무부','남','2020-3-14','과장',380,69.8);
INSERT INTO tStaff VALUES ('홍길동','인사과','남','2019-8-8','차장',380,77.7);
INSERT INTO tStaff VALUES ('대조영','총무부','남','2020-7-7','차장',290,49.9);
INSERT INTO tStaff VALUES ('장보고','인사과','남','2005-4-1','부장',440,58.3);
INSERT INTO tStaff VALUES ('선덕여왕','인사과','여','2017-8-3','사원',315,45.1);
```

NAME	DEPART	GENDER	JOINDATE	GRADE	SALARY	SCORE
1 김유신	총무부	남	00/02/03	이사	420	88.8
2 유관순	영업부	여	09/03/01	과장	380	(null)
3 안중근	인사과	남	12/05/05	대리	256	76.5
4 윤봉길	영업부	남	15/08/15	과장	350	71.25
5 강감찬	영업부	남	18/10/09	사원	320	56
6 정몽주	총무부	남	10/09/16	대리	370	89.5
7 허난설헌	인사과	여	20/01/05	사원	285	44.5

목차

4.1 조인(Join)

4.2 부질의(SUBQuery)

4.3 집합(SET)연산자

4.1 조인(Join)

●조인(Join)

- 상호 특성 관련성을 갖는 두 개 이상의 테이블로부터 새로운 테이블을 생성하는데 사용되는 연산
- 두 개 이상의 테이블을 “조인(join)” 할 수 있는 기능은 관계시스템(RDBMS)이 갖는 가장 강력한 특징.
- 정규화를 거친 데이터베이스의 각 테이블은 분리되어 관련된 다른 테이블과의 결합이 필요하며 이런 경우 관련 열의 관계성을 유도하여 결합됨

●오라클에서 조인의 종류

- Cross 조인 : 서로관련성이 없는 테이블
- Equi 조인
- Non-Equi 조인
- Self 조인
- Outer 조인

4.1 조인(Join)

●CROSS 조인

- 크로스 조인(cross join)은 관계시스템의 관계 대수 8가지 연산 중 카티션 프로덕트 (**Cartesian Product**)를 구현함
- 2개 이상의 테이블을 조건 없이 실행되는 조인 연산

```
SQL> select student.*, enrol.*  
2  from student cross join enrol;
```

Student X enroll
10 12 → 120

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT	SUB_NO	STU_NO_1	ENR_GRADE
20153075	옥한빛	기계	1	C	M	177	80	101	20131001	80
20153075	옥한빛	기계	1	C	M	177	80	104	20131001	56
20153075	옥한빛	기계	1	C	M	177	80	106	20132003	72
20153075	옥한빛	기계	1	C	M	177	80	103	20152088	45
20153075	옥한빛	기계	1	C	M	177	80	101	20131025	65
20153075	옥한빛	기계	1	C	M	177	80	104	20131025	65
20153075	옥한빛	기계	1	C	M	177	80	108	20151062	81
20153075	옥한빛	기계	1	C	M	177	80	107	20143054	41
20153075	옥한빛	기계	1	C	M	177	80	102	20153075	66
20153075	옥한빛	기계	1	C	M	177	80	105	20153075	56
20153075	옥한빛	기계	1	C	M	177	80	102	20153088	61
20153075	옥한빛	기계	1	C	M	177	80	105	20153088	78
20153088	이태연	기계	1	C	F	162	50	101	20131001	80
20153088	이태연	기계	1	C	F	162	50	104	20131001	56
.....

```
select student.*, enrol.*, subject.*  
from student cross join enrol cross join subject;
```

4.1 조인(Join)

●EQUI 조인

- 2개 이상의 테이블에 **관련 있는 공통 열**의 값을 이용하여 논리적으로 결합하는 연산이 수행되는 조인

```
SQL> select student.stu_no, stu_name, stu_dept, enr_grade  
2   from student, enrol  
3   where student.stu_no = enrol.stu_no;
```

STU_NO	STU_NAME	STU_DEPT	ENR_GRADE
20131001	김종헌	컴퓨터정보	80
20131001	김종헌	컴퓨터정보	56
20132003	박희철	전기전자	72
20152088	조민우	전기전자	45
20131025	육성우	컴퓨터정보	65
20131025	육성우	컴퓨터정보	65
20151062	김인중	컴퓨터정보	81
20143054	유가인	기계	41
.....

4.1 조인(Join)

- NATURAL JOIN

```
SQL> select stu_no, stu_name, stu_dept, enr_grade  
2   from student natural join enrol;
```

➤ 두 테이블에 같은 열의 이름이 존재하면 사용.

- 해결방법1] JOIN ~ USING

```
SQL> select stu_no, stu_name, stu_dept, enr_grade  
2   from student join enrol using(stu_no) ;
```

- 해결방법2] JOIN ~ ON

```
SQL> select student.stu_no, stu_name, stu_dept, enr_grade  
2   from student join enrol on student.stu_no = enrol.stu_no ;
```

National join test

```
create table CCC (  
  stu_no number(9),  
  stu_name varchar2(12),  
  stu_gender char(1),  
  stu_hakjum char(1),  
  stu_kg number(3));
```

```
drop table CCC;
```

```
insert into CCC values(20153075, '옥한빛', 'M','A',78);  
insert into CCC values(20153088, '이태연', 'F','B',68);  
insert into CCC values(20143054, '유가인', 'F','C',58);  
insert into CCC values(20152088, '조민우', 'M','A',88);  
insert into CCC values(20142021, '심수정', 'F','B',68);  
insert into CCC values(20132003, '박희철', 'M','B',68);  
insert into CCC values(20151062, '김인중', 'M','C',58);  
insert into CCC values(20141007, '진현무', 'M','A',88);  
insert into CCC values(20131001, '김종헌', 'M','B',68);  
insert into CCC values(20131025, '옥성우', 'F','B',68);  
insert into CCC values(20131026, '옥성우2', 'F','A',64);
```

Natural join test

commit;

```
select stu_no, stu_name, stu_gender, stu_dept, stu_kg  
      from student natural join CCC;
```

```
select stu_no, student.stu_name, student.stu_gender, stu_dept, stu_kg  
      from student join CCC using(stu_no);
```

```
select stu_no, stu_name, stu_gender, stu_dept, stu_kg  
      from student join CCC using(stu_no, stu_name, stu_gender);
```

```
select student.stu_no, student.stu_name, student.stu_gender, stu_dept, CCC.stu_kg  
      from student join CCC on student.stu_no = CCC.stu_no;--결과 출력됨.
```

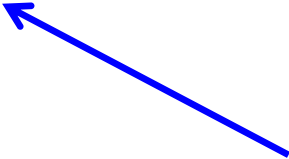
* select를 적용할 때, 테이블 이름을 적어도 2개는 적용해야 합니다.

* stu_kg처럼 CCC.stu_kg 또는 stu_kg라고 해도 됩니다.

4.1 조인(Join)

- 101번 과목을 수강하는 학생들의 학번과 이름 검색

```
SQL> select student.stu_no, stu_name  
2   from student, enrol  
3   where student.stu_no = enrol.stu_no  
      and sub_no = 101;
```



STU_NO	STU_NAME
20131001	김종현
20131025	육성우

sub_no = 101 or sub_no = 102

```
select distinct s.stu_no, s.stu_name, e.sub_no, s2.sub_name  
from student s, enrol e, subject s2  
where s.stu_no = e.stu_no  
and e.sub_no = s2.sub_no  
and (e.sub_no = 101  
or e.sub_no = 104)  
order by 1;
```

4.1 조인(Join)

- 과목번호 101 또는 102를 수강하는 학생의 학번과 이름 검색

```
SQL> select student.stu_no, stu_name  
2   from student, enrol  
3   where student.stu_no = enrol.stu_no  
4   and sub_no = 101 or sub_no = 102;
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20152088	조민우
20142021	심수정
20132003	박희철
.....
20131025	옥성우
20131001	김종현
20131025	옥성우

그러나, **and**논리연산자가 먼저 실행되어 잘못된 결과가 나오는 오류가 발생합니다.

4.1 조인(Join)

- 논리 연산자의 우선순위(문제해결)

```
SQL> select student.stu_no, stu_name  
2   from student, enrol  
3   where student.stu_no = enrol.stu_no  
4   and (sub_no = 101 or sub_no = 102);
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20131001	김종현
20131025	옥성우

4.1 조인(Join)

* 추가 조건이 여러 개인 경우에는 일반적인 EQUI조인보다는 EQUI조인의 다른 표현 방법을 사용하는 것이 좋습니다.

```
select stu_no, stu_name, sub_no
  from student natural join enrol
 where sub_no = 101 or sub_no = 102;
```

```
select stu_no, stu_name, sub_no
  from student join enrol using(stu_no)
 where sub_no = 101 or sub_no = 102;
```

```
select student.stu_no, stu_name, sub_no
  from student join enrol on student.stu_no = enrol.stu_no
 where sub_no = 101 or sub_no = 102;
```

4.1 조인(Join)

● 테이블 조인

➢ '컴퓨터개론' 과목을 수강하는 학생들의 학번, 이름, 과목이름 검색

```
SQL> select student.stu_no, stu_name, sub_name, subject.sub_no
2   from student, enrol, subject
3   where student.stu_no = enrol.stu_no
4   and enrol.sub_no = subject.sub_no
5   and (enrol.sub_no = 101 or enrol.sub_no = 102);
```

	STU_NO	STU_NAME	SUB_NO	SUB_NAME
1	20131001	김송헌	101	컴퓨터개론
2	20131025	옥성우	101	컴퓨터개론
3	20153075	옥한빛	102	기계공학법
4	20153088	이태연	102	기계공학법

4.1 조인(Join)

●Non-Equi 조인

- WHERE 절에서 사용하는 '='이 아닌 연산자를 사용
- 기본 키와 외래 키 관계가 아닌 열 값들의 의미 있는 관계 조인

```
select empno, ename, sal, grade
from emp, salgrade
where sal between losal and hisal;
```

EMPNO	ENAME	SAL	GRADE
7900	JAMES	950	1
7369	SMITH	800	1
7876	ADAMS	1100	1
7521	WARD	1250	2
7654	MARTIN	1250	2
7934	MILLER	1300	2
7499	ALLEN	1600	3
7844	TURNER	1500	3
.....

sal >= losal and sal <= hisal

4.1 조인(Join)

●SELF 조인

- 같은 테이블 간의 조인, 테이블의 **별칭**을 사용함
- 다음은 자신의 상급자를 구하는 질의문

```
SQL> select a.empno as 사원번호, a.ename as 사원이름,  
2    b.empno as 상급자사원번호, b.ename as 상급자이름  
3    from emp a, emp b  
4    where a.mgr = b.empno;
```

사원번호	사원이름	상급자사원번호	상급자이름
7369	SMITH	7902	FORD
7499	ALLEN	7698	BLAKE
7521	WARD	7698	BLAKE
7566	JONES	7839	KING
.....

➔ JOIN ~ ON

```
select a.empno as 사원번호, a.ename as 사원  
이름,  
b.empno as 상급자사원번호, b.ename as 상급  
자이름  
from emp a JOIN emp b  
ON a.mgr = b.empno
```

4.1 조인(Join)

●Outer 조인

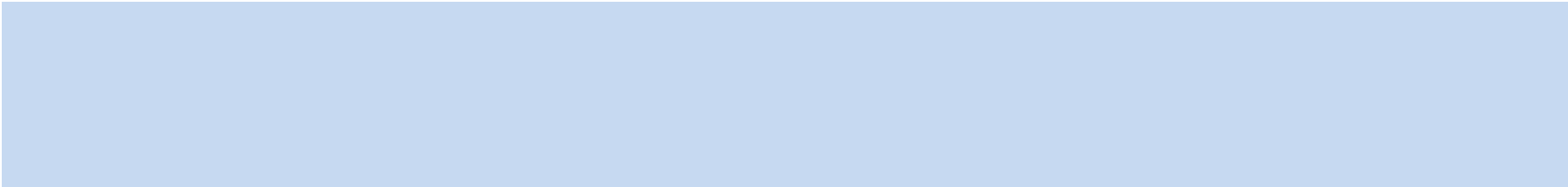
- OUTER JOIN은 **조인 조건을 만족하지 않는 행들도 질의 결과에 포함함**
- 수강(enrol) 테이블을 기준으로 과목이름 검색

```
SQL> select e.*, sub_name
2   from enrol e, subject s
3   where e.sub_no = s.sub_no
4   order by s.sub_no;
```

***과목번호 109 ~ 111은 수강생이
없어 조인에서 제외됨**

SUB_NO	SUB_NAME
109	자동화설계
110	자동제어
111	데이터베이스

SUB_NO	STU_NO	ENR_GRADE	SUB_NAME
101	20131001	80	컴퓨터개론
101	20131025	65	컴퓨터개론
102	20153075	66	기계공작법
102	20153088	61	기계공작법
103	20152088	45	기초전자실험
104	20131001	56	시스템분석설계
104	20131025	65	시스템분석설계
105	20153088	78	기계요소설계
105	20153075	56	기계요소설계
106	20132003	72	전자회로실험
107	20143054	41	CAD응용실습
108	20151062	81	소프트웨어공학



```
select e.sub_no, e.stu_no, e.enr_grade, sub_name
      from enrol e, subject s
      where e.sub_no = s.sub_no
      order by 1; -- sub_no [o]
```

```
select e.*, sub_name
      from enrol e, subject s
      where e.sub_no = s.sub_no
      order by 1; -- [O], s.sub_no [O]
```

4.1 조인(Join)

```
select a.*, sub_name  
  from enrol a right outer join subject b  
    on a.sub_no = b.sub_no  
 order by 1;
```

SUB_NO	STU_NO	ENR_GRADE	SUB_NAME
101	20131001	80	컴퓨터개론
101	20131025	65	컴퓨터개론
102	20153088	61	기계공학법
102	20153075	66	기계공학법
103	20152088	45	기초전자실험
104	20131025	65	시스템분석설계
104	20131001	56	시스템분석설계
105	20153075	56	기계요소설계
105	20153088	78	기계요소설계
106	20132003	72	전자회로실험
107	20143054	41	CAD응용실습
108	20151062	81	소프트웨어공학
			데이터베이스
			자동화설계
			자동제어

- ✓ RIGTH OUTER JOIN
- ✓ LEFT OUTER JOIN
- ✓ FULL OUTER JOIN

4.1 조인(Join)

●SELF 조인과 OUTER 조인의 결합

select a.empno as 사원번호, a.ename as 사원이름,
b.empno as 상급자사원번호, b.ename as 상급자이름
from emp a left outer join emp b on a.mgr = b.empno;

사원번호	사원이름	상급자사원번호	상급자이름
7782	CLARK	7839	KING
7698	BLAKE	7839	KING
7566	JONES	7839	KING
.....
7499	ALLEN	7698	BLAKE
7934	MILLER	7782	CLARK
7876	ADAMS	7788	SCOTT
7369	SMITH	7902	FORD
7839	KING		

full outer join

select a.empno as 사원번호, a.ename as 사원이름,
b.empno as 상급자사원번호, b.ename as 상급자이름
from emp a full outer join emp b on a.mgr = b.empno;

SQL | 인출된 모든 행: 24(0.003초)

	사원번호	사원이름	상급자사원번호	상급자이름
9	7521	WARD	7698	BLAKE
10	7499	ALLEN	7698	BLAKE
11	7934	MILLER	7782	CLARK
12	8000	ADAMS2	7788	SCOTT
13	7876	ADAMS	7788	SCOTT
14	7369	SMITH	7902	FORD
15	(null)	(null)	7499	ALLEN
16	(null)	(null)	7521	WARD
17	(null)	(null)	7654	MARTIN
18	(null)	(null)	7844	TURNER
19	(null)	(null)	7900	JAMES
20	(null)	(null)	7934	MILLER
21	(null)	(null)	7369	SMITH
22	(null)	(null)	7876	ADAMS
23	(null)	(null)	8000	ADAMS2
24	7839	KING	(null)	(null)

4.2 부질의(SUBQuery)

●부질의(SUBQuery)

- SELECT문내에 **또 다른 SELECT문을 포함함(부질의)**
- '옥성우' 학생보다 신장이 큰 학생들의 학번, 이름, 신장 검색

```
SQL>select stu_height  
2  from student  
3  where stu_name = '옥성우';
```

STU_HEIGHT
172

```
SQL>select stu_no, stu_height, stu_name  
2  from student  
3  where stu_height > 172;
```

STU_HEIGHT
177
188
174

4.2 부질의(SubQuery)

- 하나의 질의문으로 처리함

```
select stu_no, stu_name, stu_height
from student
where stu_height >
      (select stu_height
       from student
       where stu_name = '옥성우');
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20152088	조민우	188
20141007	진현무	174

- 부질의는 WHERE절, HAVING절, FROM절을 사용가능 한데, 하나의 열의 값을 반환하는 단일열부질의와 복수개의 값을 반환하는 복수열부질의로 나뉜다.

4.2 부질의(SubQuery)

- SELF 조인 이용하여 재 작성함

```
select a.stu_no, a.stu_name, a.stu_height  
from student a, student b  
where a.stu_height > b.stu_height  
and b.stu_name = '옥성우';
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20152088	조민우	188
20141007	진현무	174

4.2 부질의(SubQuery)

● 단일열 부질의

- 열의 값 한 개를 반환하는 부질의
- 박희철 학생과 같은 체중을 가지고 있는 학생의 정보 검색

SQL> select *

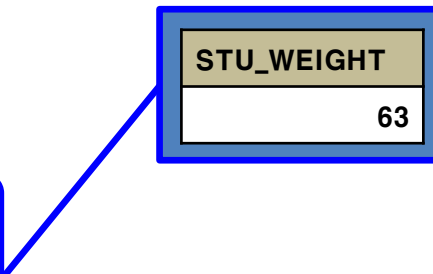
2 from student

3 where stu_weight =

4 (select stu_weight

5 from student

6 where stu_name = '박희철');



STU_WEIGHT
63

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20131025	옥성우	컴퓨터정보	3	A	F	172	63

4.2 부질의(SubQuery)

```
select *  
  from student  
 where stu_weight =  
        (select stu_weight  
          from student  
         where stu_name = '박희철')  
       and stu_name <> '박희철';
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20131025	육성우	컴퓨터정보	3	A	F	172	63

4.2 부질의(SubQuery)

- 부질의 결과가 다중일 경우(IN 연산자 사용)
- '컴퓨터정보'과 학생과 같은 반을 가진 학생의 정보 검색

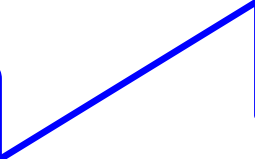
select *

from student

where stu_class in

**(select distinct stu_class
from student
where stu_dept = '컴퓨터정보')**

and stu_dept <> '컴퓨터정보';



STU_CLASS
B
A
C

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

4.2 부질의(SubQuery)

➤ 전체 학생들의 평균신장 보다 큰 학생의 정보 검색

select *

from student

where stu_height >

**(select avg(stu_height)
from student);**

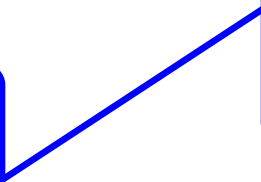
STU_HEIGHT
170.125

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20152088	조민우	전기전자	1	C	M	188	90
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131025	옥성우	컴퓨터정보	3	A	F	172	63

4.2 부질의(SubQuery)

➤ 신장이 모든 학과들의 평균 신장보다 큰 학생의 정보 검색

```
SQL> select *  
2   from student  
3   where stu_height > all  
4     (select avg(stu_height)  
5        from student  
6        group by stu_dept);
```



STU_HEIGHT
178
164.3333333
170.6666667

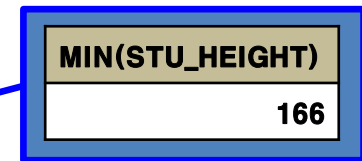
STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90

4.2 부질의(SubQuery)

➤ 컴퓨터정보과의 최소 신장보다 최소 신장이 더 큰 학과의 학과명과 최소 신장 검색

```
select stu_dept, min(stu_height)
from student
group by stu_dept having min(stu_height) >
(select min(stu_height)
 from student
 where stu_dept = '컴퓨터정보');
```

STU_DEPT	MIN(STU_HEIGHT)
전기전자	168



MIN(STU_HEIGHT)
166

4.2 부질의(SubQuery)

- 일반적으로 **부질의**의 질의문을 **조인** 질의로 표현 가능

➢ 101번 과목을 수강한 학생들의 정보 검색

```
SQL> select *  
2  from student  
3  where stu_no in  
4      (select stu_no  
5         from enrol  
6         where sub_no = 101);
```

```
SQL> select *  
2  from student a, enrol b  
3  where a.stu_no = b.stu_no and b.sub_no = 101;
```

4.2 부질의(SubQuery)

- 101번 과목을 수강한 학생들의 학번, 이름, 점수를 검색

```
SQL> select a.stu_no, a.stu_name, b.enr_grade (equi join)
2   from student a, enrol b
3   where a.stu_no = b.stu_no and b.sub_no = 101;
```

STU_NO	STU_NAME	ENR_GRADE
20131001	김종현	80
20131025	육성우	65

- 부질의로 표현 불가

select 내용중에서 enr_grade가 학생 테이블이 아닌 수강테이블이므로

4.2 부질의(SubQuery)

- 복수열 부질의

- 복수열 값을 반환하는 부질의("=" 사용)

- 실습을 위한 사원(emp) 테이블과 구조가 같은 test 테이블 생성

```
create table test(empno, ename, sal, comm, deptno)
as
select empno, ename, sal, comm, deptno
from emp
where deptno = 1;
```

```
insert into test values(11, 'apple', 1000, null, 30);
insert into test values(12, 'banana', 2000, 100, 30);
insert into test values(13, 'cheese', 1000, 0, 10);
insert into test values(14, 'dog', 2000, null, 20);
insert into test values(15, 'egg', 1000, 100, 20);
```

```
select * from test;
```

4.2 부질의(SubQuery)

●test 테이블 데이터 검색

SQL> select * from test;

EMPNO	ENAME	SAL	COMM	DEPTNO
11	apple	1000		30
12	banana	2000	100	30
13	cheese	1000	0	10
14	dog	2000		20
15	egg	1000	100	20

4.2 부질의(SubQuery)

➤ 부질의 결과값이 복수열

```
select *  
  from test  
 where (sal, nvl(comm, -1)) =  
       (select sal, nvl(comm, -1)  
        from test  
        where empno = 11);
```

SAL	NVL(COMM, -1)
1000	-1

EMPNO	ENAME	SAL	COMM	DEPTNO
11	apple	1000		30

***커미션의 경우에는 null 값인데, null값은 비교 자체가 불가능하므로 nvl함수를 사용하여 처리합니다.**

4.2 부질의(SubQuery)


* 시스템 계정을 변경해야 하므로 생략 합니다.

●FROM 절의 부질의(In-Line 뷰)

➢ 학생들의 학과별 평균 신장보다 큰 신장의 학생들 정보 검색

```
SQL> select stu_dept, round(avg(stu_height),2) as avg_height  
2   from student  
3   group by stu_dept; //각 학과의 평균신장 출력
```

```
SQL> select stu_no, stu_name, a.stu_dept, stu_height, avg_height  
2   from student a, (  
3  
4  
5   ) b  
5 where a.stu_dept = b.stu_dept and stu_height > avg_height;
```



➢ 위와 같이 인라인 뷰를 사용하는 경우에는 테이블이름대신 별칭을 사용합니다.

<문제풀이>

--HR 계정의 jobs 테이블 복사하여 이용합니다.

```
Create table jobs (  
    JOB_ID VARCHAR2(10 BYTE)    primary key,  
    JOB_TITLE      VARCHAR2(35 BYTE)    not null,  
    MIN_SALARY     NUMBER(6,0),  
    MAX_SALARY     NUMBER(6,0)  
);
```

* 참고사항

```
create table jobs  
as select *  
from subject;
```

<문제풀이>

```
insert into jobs values('AD_PRES', 'President', 20080, 40000);
insert into jobs values('AD_VP', 'Administration Vice President', 15000, 30000);
insert into jobs values('AD_ASST', 'Administration Assistant', 3000, 6000);
insert into jobs values('FI_MGR', 'Finance Manager', 8200, 16000);
insert into jobs values('FI_ACCOUNT', 'Accountant', 4200, 9000);
insert into jobs values('AC_MGR', 'Accounting Manager', 8200, 16000);
insert into jobs values('AC_ACCOUNT', 'Public Accountant', 4200, 9000);
insert into jobs values('SA_MAN', 'Sales Manager', 10000, 20080);
insert into jobs values('SA_REP', 'Sales Representative', 6000, 12008);
insert into jobs values('PU_MAN', 'Purchasing Manager', 8000, 15000);
insert into jobs values('PU_CLERK', 'Purchasing Clerk', 2500, 5500);
insert into jobs values('ST_MAN', 'Stock Manager', 5500, 8500);
insert into jobs values('ST_CLERK', 'Stock Clerk', 2008, 5000);
insert into jobs values('SH_CLERK', 'Shipping Clerk', 2500, 5500);
insert into jobs values('IT_PROG', 'Programmer', 4000, 10000);
insert into jobs values('MK_MAN', 'Marketing Manager', 9000, 15000);
insert into jobs values('MK_REP', 'Marketing Representative', 4000, 9000);
insert into jobs values('HR_REP', 'Human Resources Representative', 4000, 9000);
insert into jobs values('PR_REP', 'Public Relations Representative', 4500, 10500);
```

<문제풀이>

--1. 직책(Job Title)이 Programmer인 직원들의 입사년도와 입사년도(hire_date)별 평균 급여를 출력하시오.

--- 출력 시 년도를 기준으로 오름차순 정렬하시오.

```
select to_char(hire_date, 'YYYY') 입사년도, avg(salary) 평균급여
from employees
where job_id = (select job_id
                from jobs
                where job_title = 'Programmer')
group by to_char(hire_date, 'YYYY')
order by 1;
```

<문제풀이>

--HR 계정의 location 테이블 복사하여 이용합니다.

- 2. 각 도시(city)에 있는 모든 부서 직원들의 평균급여를 조회하고자 한다.
- 평균급여가 가장 낮은 도시부터 도시명(city)과 평균연봉, 해당 도시의 직원수를 출력 하시오.
- 단, 도시에 근무하는 직원이 10명 이상인 곳은 제외하고 조회하시오.

```
create table locations(  
  LOCATION_ID  NUMBER(4,0)  primary key,  
  STREET_ADDRESS  VARCHAR2(40),  
  POSTAL_CODE  VARCHAR2(12),  
  CITY  VARCHAR2(30) not null,  
  STATE_PROVINCE  VARCHAR2(25) not null,  
  COUNTRY_ID  CHAR(2)  
);
```

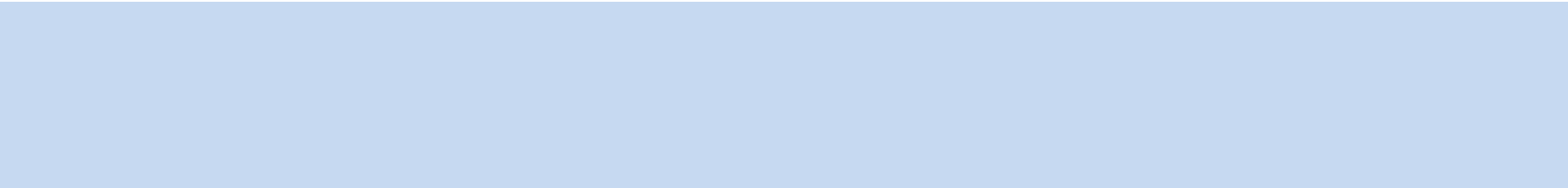
<문제풀이>

```

insert into locations values(1000, '1297 Via Cola di Rie', '00989', 'Roma', null, 'IT');
insert into locations values(1100, '93091 Calle della Testa', '10934', 'Venice', null, 'IT');
insert into locations values(1200, '2017 Shinjuku-ku', '1689', 'Tokyo', 'Tokyo Prefecture', 'JP');
insert into locations values(1300, '9450 Kamiya-cho', '6823', 'Hiroshima', 'null', 'JP');
insert into locations values(1400, '2014 Jabberwocky Rd', '26192', 'Southlake', 'Texas', 'US');
insert into locations values(1500, '2011 Interiors Blvd', '99236', 'South San Francisco', 'California', 'US');
insert into locations values(1600, '2007 Zagora St', '50090', 'South Brunswick', 'New Jersey', 'US');
insert into locations values(1700, '2004 Charade Rd', '98199', 'Seattle', 'Washington', 'US');
insert into locations values(1800, '147 Spadina Ave', 'M5V 2L7', 'Toronto', 'Ontario', 'CA');
insert into locations values(1900, '6092 Boxwood St', 'YSW 9T2', 'Whitehorse', 'Yukon', 'CA');
insert into locations values(2000, '40-5-12 Laogianggen', '190518', 'Beijing', 'null', 'CN');
insert into locations values(2100, '1298 Vileparle (E)', '490231', 'Bombay', 'Maharashtra', 'IN');
insert into locations values(2200, '12-98 Victoria Street', '2901', 'Sydney', 'New South Wales', 'AU');
insert into locations values(2300, '198 Clementi North', '540198', 'Singapore', 'null', 'SG');
insert into locations values(2400, '8204 Arthur St', 'null', 'London', 'null', 'UK');
insert into locations values(2500, 'Magdalen Centre, The Oxford Science Park', 'OX9 9ZB', 'Oxford', 'Oxford', 'UK');
insert into locations values(2600, '9702 Chester Road', '09629850293', 'Stretford', 'Manchester', 'UK');
insert into locations values(2700, 'Schwanthalerstr. 7031', '80925', 'Munich', 'Bavaria', 'DE');
insert into locations values(2800, 'Rua Frei Caneca 1360', '01307-002', 'Sao Paulo', 'Sao Paulo', 'BR');
insert into locations values(2900, '20 Rue des Corps-Saints', '1730', 'Geneva', 'Geneve', 'CH');
insert into locations values(3000, 'Murtenstrasse 921', '3095', 'Bern', 'BE', 'CH');
insert into locations values(3100, 'Pieter Breughelstraat 837', '3029SK', 'Utrecht', 'Utrecht', 'NL');
insert into locations values(3200, 'Mariano Escobedo 9991', '11932', 'Mexico City', 'Distrito Federal', 'MX');

```

select * from locations order by LOCATION_ID;--21, null포함하면 23개.



```
create table DEPARTMENTS(  
    DEPARTMENT_ID    NUMBER(4,0) primary key,  -- unique + not null  
    DEPARTMENT_NAME  VARCHAR2(30 BYTE) not null,  
    MANAGER_ID       NUMBER(6,0),  
    LOCATION_ID      NUMBER(4,0)  
);  
select * from departments;
```

```
insert into departments values(10, 'Administration', 200, 1700);
insert into departments values(20, 'Marketing', 201, 1800);
insert into departments values(30, 'Purchasing', 114, 1700);
insert into departments values(40, 'Human Resources', 203, 2400);
insert into departments values(50, 'Shipping', 121, 1500);
insert into departments values(60, 'IT', 103, 1400);
insert into departments values(70, 'Public Relations', 204, 2700);
insert into departments values(80, 'Sales', 145, 2500);
insert into departments values(90, 'Executive', 100, 1700);
insert into departments values(100, 'Finance', 108, 1700);
insert into departments values(110, 'Accounting', 205, 1700);
insert into departments values(120, 'Treasury', null, 1700);
insert into departments values(130, 'Corporate Tax', null, 1700);
insert into departments values(140, 'Control And Credit', null, 1700);
insert into departments values(150, 'Shareholder Services', null, 1700);
insert into departments values(160, 'Benefits', null, 1700);
insert into departments values(170, 'Manufacturing', null, 1700);
insert into departments values(180, 'Construction', null, 1700);
insert into departments values(190, 'Contracting', null, 1700);
insert into departments values(200, 'Operations', null, 1700);
insert into departments values(210, 'IT Support', null, 1700);
insert into departments values(220, 'NOC', null, 1700);
insert into departments values(230, 'IT Helpdesk', null, 1700);
insert into departments values(240, 'Government Sales', null, 1700);
insert into departments values(250, 'Retail Sales', null, 1700);
insert into departments values(260, 'Recruiting', null, 1700);
insert into departments values(270, 'Payroll', null, 1700);
```

```
commit;
```

<문제풀이>

```
select l.city, avg(salary), count(*)  
from employees e, departments d, locations l  
where e.department_id = d.department_id  
and d.location_id = l.location_id  
group by l.city  
having count(*) < 10  
order by avg(salary);
```

	↕ CITY	↕ AVG(SALARY)	↕ COUNT(*)
1	Oxford	4200	1
2	Southlake	4800	1
3	Munich	4800	1
4	South San Francisco	6000	1
5	London	9000	1
6	Seattle	15502	4
7	Toronto	17000	1

<문제풀이>

-- HR 계정에서 복사하여 JOB_HISTORY 테이블 생성

--3. 'Programmer'의 직책(job_title)으로 과거에 근무한 적이 있는 모든 사원의 사번과 이름을 출력하시오.

-- (현재 'Programmer'의 직책(job_title)으로 근무하는 사원은 고려하지 않는다.)

```
create table JOB_HISTORY (  
    EMPLOYEE_ID      NUMBER(6,0),  
    START_DATE DATE   not null,  
    END_DATE   DATE   not null,  
    JOB_ID       VARCHAR2(10) not null,  
    DEPARTMENT_ID  NUMBER(4,0)  
);
```

<문제풀이>

```
insert into JOB_HISTORY values(102, '01/01/13', '06/07/24', 'IT_PROG', 60);
insert into JOB_HISTORY values(101, '97/09/21', '01/10/27', 'AC_ACCOUNT', 110);
insert into JOB_HISTORY values(101, '01/10/28', '05/03/15', 'AC_MGR', 110);
insert into JOB_HISTORY values(201, '04/02/17', '07/12/19', 'MK_REP', 20);
insert into JOB_HISTORY values(114, '06/03/24', '07/12/31', 'ST_CLERK', 50);
insert into JOB_HISTORY values(122, '07/01/01', '07/12/31', 'ST_CLERK', 50);
insert into JOB_HISTORY values(200, '95/09/17', '01/06/17', 'AD_ASST', 90);
insert into JOB_HISTORY values(176, '06/03/24', '06/12/31', 'SA_REP', 80);
insert into JOB_HISTORY values(176, '07/01/01', '07/12/31', 'SA_MAN', 80);
insert into JOB_HISTORY values(200, '02/07/01', '06/12/31', 'AC_ACCOUNT', 90);
```

```
select e.EMPLOYEE_ID || ' ' || e.EMPLOYEE_NAME as 사번이름
from employees e, job_history h
where e.employee_id = h.employee_id
and h.job_id = (select job_id
                from jobs
                where job_title = 'Programmer');-- 102 Lex
```

--4. 자신의 매니저보다 연봉(salary)를 많이 받는 직원들의 성(last_name)과 연봉(salary)를 출력하시오.

```
select a.EMPLOYEE_NAME "직원 이름", a.salary "직원 연봉", b.salary "매니저 연봉"
from employees a, employees b
where a.manager_id = b.employee_id
and a.salary > b.salary;
```

	직원 이름	직원 연봉	매니저 연봉
1	StevenKing	24000	9000

--5. 2007년에 입사(hire_date)한 직원들의 사번(employee_id), 이름(EMPLOYEE_NAME), 부서명(department_name)을 조회합니다.

-- 이때, 부서에 배치되지 않은 직원의 경우, '<Not Assigned>'로 출력하십시오.

```
select e.employee_id, EMPLOYEE_NAME, nvl(d.department_name, 'Not Assigned')
```

```
from employees e, departments d
```

```
where e.department_id = d.department_id
```

```
and to_char(e.hire_date, 'yyyy') = '2007';
```

	EMPLOYEE_ID	EMPLOYEE_NAME	NVL(D,DEPARTMENT_NAME,'NOTASSIGNED')
1	104	Bruce	Shipping
2	107	Diana	Sales

--6. 업무명(job_title)이 'Programmer'인 직원 중에서 연봉(salary)이 6,000 이상, 12,000 이하인 직원들의 이름(EMPLOYEE_NAME)과 연봉(salary)를 출력하시오.

```
select EMPLOYEE_NAME, e.salary
from employees e, jobs j
where e.job_id = j.job_id
and j.job_title = 'Programmer'
and salary between 6000 and 12000;
```

	EMPLOYEE_NAME	SALARY
1	Alexander	9000
2	Bruce	6000

--7. 부서별로 가장 적은 급여를 받고 있는 직원의 이름, 부서이름, 급여를 출력하시오.

```
select e.employee_name, d.department_name, e.salary
from employees e, departments d
where e.department_id = d.department_id
and e.salary in (select min(salary)
                  from employees e, departments d
                  where e.department_id = d.department_id
                  group by d.department_id);
```

	EMPLOYEE_NAME	DEPARTMENT_NAME	SALARY
1	StevenKing	Administration	24000
2	NeenaKochhar	Marketing	17000
3	Lex	Purchasing	17000
4	Alexander	Human Resources	9000
5	Bruce	Shipping	6000
6	David	IT	4800
7	Valli	Public Relations	4800

--8. EMPLOYEES 테이블에서 급여를 많이 받는 순서대로 조회했을 때 결과처럼 6번째부터 10 번째까지 5명의 이름과 salary를 조회 하시오.

```
select EMPLOYEE_NAME, salary
from (select rownum no, EMPLOYEE_NAME, salary
      from employees
      order by salary desc)
where no between 6 and 10;
```

	EMPLOYEE_NAME	SALARY
1	Nancy	12008
2	Daniel	9000
3	David	4800
4	Valli	4800
5	Diana	4200

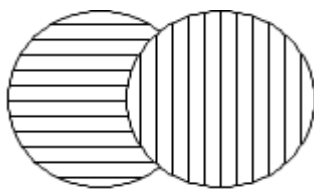
9. 각 업무(job) 별로 연봉(salary)의 총합을 구하고자 한다. 연봉 총합이 가장 높은 업무부터 업무명(job_title)과 연봉 총합을 조회 하시오.
단, 연봉 총합이 10,000보 다 큰 업무만 출력 하시오.

```
select job_title, sum(e.salary)
from jobs j, employees e
where e.job_id = j.job_id
group by j.job_title
having sum(e.salary) >= 10000
order by sum(e.salary) desc;
```

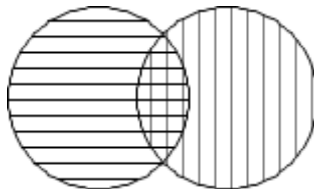
	JOB_TITLE	SUM(E, SALARY)
1	Programmer	28800
2	Finance Manager	12008

4.3 집합(SET)

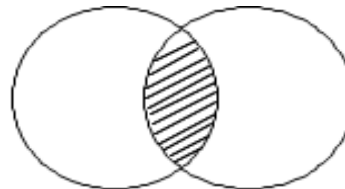
연산자	설 명
UNION	두 질의 결과값의 합으로 중복을 제거됨
UNION ALL	두 질의 결과값의 합으로 중복을 포함됨
INTERSECT	두 질의 결과값의 공통되는 값
MINUS	첫 번째 질의 결과에서 두 번째 질의 결과에 있는 행을 제거한 값



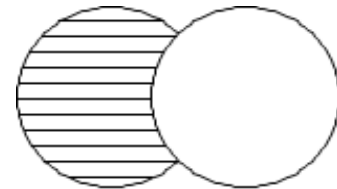
UNION



UNION ALL



INTERSECT



MINUS

4.3 집합(SET)

- 집합 연산자 사용 규칙은 다음과 같다.

- 두 개 이상의 SELECT문장의 열의 수와 데이터 타입 일치
- 열의 이름 일치하지 않을 경우 첫 번째 SELECT문의 열의 이름
- 정렬을 위해서는 마지막 SELECT문에 ORDER BY절 표현
- 집합 연산자 부질의에 사용가능

4.3 집합(SET)

- 실습을 위해 a_student, b_student 테이블 생성

```
create table a_student  
as select *  
from student  
where stu_dept in ('기계', '전기전자');  
desc a_student;
```

```
create table b_student  
as select *  
from student  
where stu_dept in ('전기전자', '컴퓨터정보');  
desc b_student;
```

4.3 집합(SET)

➤ 생성된 a_student, b_student의 내용 확인

SQL> select * from a_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63

SQL> select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

4.3 집합(SET)

●UNION

SQL> select * from a_student

2 **union** (중복데이터 제거 출력)

3 select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	육한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	육성우	컴퓨터정보	3	A	F	172	63

4.3 집합(SET)

●UNION

```
SQL> select * from a_student  
2 union all (중복데이터 허용 출력)  
3 select * from b_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	육한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	육성우	컴퓨터정보	3	A	F	172	63

4.3 집합(SET)

●INTERSECT

```
SQL> select * from a_student  
2 intersect (중복데이터만 출력)  
3 select * from b_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63

4.3 집합(SET)

●MINUS

```
SQL> select * from a_student  
2 minus  
3 select * from b_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47

*minus

(첫번째 질의 결과에서 두번째 질의 결과와 중복된 전기전자과 학생들의 정보만 제외되고 **기계과 학생정보만 출력됩니다.**)