데이터베이스
(Oracle)

Database 구성

• Oracle 설치

1) http://www.oracle.com 오라클 홈페이지에서 회원가입 한다.

2) Oracle Database 11g Express를 선택하여 다운로드 후 설치한다.

3)설치 시 데이터베이스 암호를 manager 로 입력한다.

• 서비스 수동으로 변경

1) 시작 -> 설정-> 제어판-> 관리도구 -> 서비스에서 Oracle과 관련된Service를 찾아본다.

2) OracleServiceXE의 속성에서 시작유형을 자동에서 수동으로 변경한다.

3) OracleXETNSLinstener의 속성에서 시작유형을 자동에서 수동으로 변경한다.

• Oracle Database 서버접속

방법1: 시작메뉴 -> Oracle Database 11g Express Edition -> Run SQL Command Line을 실행한다.

```
SQL> connect;
Enter user-name : system
Enter password : manager
SQL> disconnect;
```

방법2: Oracle 11g에서는 isqlplus가 지원되지 않으므로 sqldeveloper를 사용한다.

```
http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html 에서 다운로드한다.
```

sqldeveloper.exe를 실행한 후 새 접속을 등록한다.

```
1)접속이름: oracle
2)사용자 이름: system
3)비밀번호: manager
4)호스트 이름: localhost
5)포트: 1521
6)SID : xe
```

• 사용자(User)관련 명령어

```
사용자 생성: CREATE USER  haksa(사용자)  IDENTIFIED  BY  pass(비밀번호);
사용자 권한 지정: GRANT CONNECT, RESOURCE, DBA  TO  haksa(사용자);
사용자 비밀번호 변경: ALTER  USER  haksa(사용자)  IDENTIFIED BY  newpass(새 비밀번호);
사용자 검색: SELECT Username FROM dba_users;
사용자 삭제: drop user haksa;
```

• system 비밀번호 분실 시

```
1) connect
2) Enter user-name: -sys as sysdba
3) pass: 그냥 엔터
4) ALTER  USER  system IDENTIFIED BY  newpass(새 비밀번호);
```

• 오라클 포트 확인 및 변경

```
확인: select dbms_xdb.gethttpport() from dual;
변경: exec dbms_xdb.sethttpport(변경할 포트번호);
```

• 오라클 SID(Session Identification)확인

```
SQL> select name from v$database;
```

학사관리 데이터베이스

- **교수(professors) 테이블의 구조**

열이름	데이터형	정보내용
pcode	char(3)	교수번호
pname	varchar(15)	교수이름
dept	varchar(30)	소속 학과
hiredate	date	임용일자
title	varchar(15)	직급
salary	int	급여

- **학생(students) 테이블의 구조**

열이름	데이터형	정보내용
scode	char(8)	학생번호(학번)
sname	varchar(15)	학생이름
dept	varchar(30)	소속 학과
year	char(1)	학년
birthday	date	생년월일
advisor	char(3)	지도교수

- **강좌(courses) 테이블의 구조**

열이름	데이터형	정보내용
lcode	char(4)	강좌번호
lname	varchar(50)	강좌이름
hours	int	강의 시간 수
room	char(3)	강의실
instructor	char(3)	담당교수(교수번호)
capacity	int	최대수강인원수
persons	int	수강신청인원수

- **수강신청(enrollments)테이블의 구조**

열이름	데이터형	정보내용
lcode	char(4)	강좌번호
scode	char(8)	학생번호
edate	date	수강신청일
grade	int	성적

ERD 다이어그램 (Oracle)

```
[보기] -[Data Modeler] - [브라우저]-[제목없음_1]-[관계형 모델] -[새 관계형 모델] 선택 후 테이블을 넣어준다.
```

ERD 다이어그램 (MySQL)

```
[Database]-[Reverse Engineer Database]
```

- **교수(professors)테이블의 데이터**

pcode(교수번호)	pname(교수이름)	dept(소속 학과)	hiredate(임용일)	title(직급)	salary(급여)
221	이병렬	전산	75/04/03	정교수	3,000,000
228	이재광	전산	91/09/19	부교수	2,500,000
311	강승일	전자	94/06/09	부교수	2,300,000
509	오문환	건축	92/10/14	조교수	2,000,000

- **학생(Students)테이블의 데이터**

scode(학생번호)	sname(학생이름)	dept(소속학과)	year(학년)	birthday(생년월일)	advisor(지도교수 번호)
92414029	서연우	전산	3	73/10/06	228
92414033	김창덕	전산	4	73/10/26	221
92514009	이지행	전자	4	73/11/16	311
92514023	김형명	전자	4	73/08/29	311
92454018	이원구	건축	3	74/09/30	509
95454003	이재영	건축	4	76/02/06	509
95414058	박혜경	전산	4	76/03/12	221
96414404	김수정	전산	3	77/12/22	228

- **강좌(courses)테이블의 데이터**

lcode (강좌번호)	lname (강좌이름)	hours (강의시간수)	room (강의실)	instructor (담당교수번호)	capacity (최대수강인원수)	persons (수강신청인원수)
C301	파일처리론	3	506	221	100	80
C401	데이터베이스	3	414	221	80	80
C421	알고리즘	3	510	228	80	72
C312	자료구조	2	510	228	100	60
E221	논리회로	3	304	311	100	80
A109	한국의건축문화	2	101	509	120	36

- **수강신청(enrollments)테이블의 데이터**

lcode(강좌번호)	scode(학생번호)	edate(수강신청일)	grade(성적)
C401	92414033	98/03/02	85
C301	92414033	98/03/02	80
C421	92414033	98/03/02	0
C401	95414058	98/03/03	90
C312	95414058	98/03/03	80
C401	92514023	98/03/03	70
C301	92414029	98/03/03	90
C421	92414029	98/03/03	0
C312	92414029	98/03/03	70

날짜 포맷 변경

```
select * from nls_session_parameter;
alter session set nls_date_format='YYYY-MM-DD HH24:MI:SS'
```

자동증감

```
create sequence seq_product increment by 1 start with 1 minvalue 1 maxvalue 1000 nocycle cache;
insert into tbl_product(pcode) values(seq_product.nextval);
```

테이블 구조

```
desc 테이블명
```

사용자 테이블 검색

```
select table_name from  user_tables;
```

테이블 삭제문

```
drop table 테이블명
```

교수테이블 생성

```
create table professors(
    pcode char(3) not null,
    pname varchar(15)  not null,
    dept varchar(30),
    hiredate date,
    title varchar(15),
    salary int default 0,
    primary key(pcode)
);
```

학생테이블 생성

```
create table students(
    scode char(8) not null,
    sname varchar(15) not null,
    dept varchar(30),
    year int default 1,
    birthday date,
    advisor char(3),
    primary key(scode),
    foreign key(advisor) references professors(pcode)  /* on delete cascade */
);
```

강좌테이블생성

```
create table courses(
    lcode char(4) not null,
    lname varchar(50) not null,
    hours int,
    room char(3),
    instructor char(3),
    capacity int default 0,
    persons int default 0,
    primary key(lcode),  /* constraint child_pk foreign key(instructor) references professors(pcode) */
    foreign key(instructor) references professors(pcode)
);
```

수강신청테이블 생성

```
create table enrollments(
     lcode char(4) not null,
     scode char(8) not null,
     edate date,
     grade int default 0,
     primary key(lcode, scode),
     foreign key(lcode) references courses(lcode),
     foreign key(scode) references students(scode)
);
```

교수테이블 데이터입력

```
insert into professors(pcode,pname,dept,hiredate,title,salary) values('221','이병렬','전산','75/04/03','정교수',3000000);
insert into professors(pcode,pname,dept,hiredate,title,salary) values('228','이재광','전산','91/09/19','부교수',2500000);
insert into professors(pcode,pname,dept,hiredate,title,salary) values('311','강승일','전자','94/06/09','부교수',2300000);
insert into professors(pcode,pname,dept,hiredate,title,salary) values('509','오문환','건축','92/10/14','조교수',2000000);
```

학생테이블 데이터 입력

```
insert into students(scode,sname,dept,year,birthday,advisor) values('92414029','서연우','전산',3,'73/10/06','228');
insert into students(scode,sname,dept,year,birthday,advisor) values('92414033','김창덕','전산',4,'73/10/26','221');
insert into students(scode,sname,dept,year,birthday,advisor) values('92514009','이지행','전자',4,'73/11/16','311');
insert into students(scode,sname,dept,year,birthday,advisor) values('92514023','김형명','전자',4,'73/08/29','311');
insert into students(scode,sname,dept,year,birthday,advisor) values('92454018','이원구','건축',3,'74/09/30','509');
insert into students(scode,sname,dept,year,birthday,advisor) values('95454003','이재영','건축',4,'76/02/06','509');
insert into students(scode,sname,dept,year,birthday,advisor) values('95414058','박혜경','전산',4,'76/03/12','221');
insert into students(scode,sname,dept,year,birthday,advisor) values('96414404','김수정','전산',3,'77/12/22','228');
```

강좌테이블 데이터 입력

```
insert into courses(lcode,lname,hours,room,instructor,capacity,persons) values('C301','파일처리론', 3 ,'506','221',100,80);
insert into courses(lcode,lname,hours,room,instructor,capacity,persons) values('C401','데이터베이스',3,'414','221',80,80);
insert into courses(lcode,lname,hours,room,instructor,capacity,persons) values('C421','알고리즘',3,'510','228',80,72);
insert into courses(lcode,lname,hours,room,instructor,capacity,persons) values('C312','자료구조',2,'510','228',100,60);
insert into courses(lcode,lname,hours,room,instructor,capacity,persons) values('E221','논리회로',3,'304','311',100,80);
insert into courses(lcode,lname,hours,room,instructor,capacity,persons) values('A109','한국의건축문화',2,'101','509',120,36);
```

수강신청 테이블 데이터 입력

```
insert into enrollments(lcode, scode, edate, grade) values('C401','92414033','98/03/02',85);
insert into enrollments(lcode, scode, edate, grade) values('C301','92414033','98/03/02',80);
insert into enrollments(lcode, scode, edate, grade) values('C421','92414033','98/03/02', 0);
insert into enrollments(lcode, scode, edate, grade) values('C401','95414058','98/03/03',90);
insert into enrollments(lcode, scode, edate, grade) values('C301','95414058','98/03/03',80);
insert into enrollments(lcode, scode, edate, grade) values('C312','95414058','98/03/03',80);
insert into enrollments(lcode, scode, edate, grade) values('C401','92514023','98/03/03',70);
insert into enrollments(lcode, scode, edate, grade) values('C301','92514023','98/03/03',70);
insert into enrollments(lcode, scode, edate, grade) values('C421','92514023','98/03/03',70);
insert into enrollments(lcode, scode, edate, grade) values('C301','92414029','98/03/03',90);
insert into enrollments(lcode, scode, edate, grade) values('C421','92414029','98/03/03',0);
insert into enrollments(lcode, scode, edate, grade) values('C312','92414029','98/03/03',70);
insert into enrollments(lcode, scode, edate, grade) values('E221','92414029','98/03/03',75);
insert into enrollments(lcode, scode, edate, grade) values('A109','92414029','98/03/03',90);
insert into enrollments(lcode, scode, edate, grade) values('C301','92514009','98/03/03',70);
insert into enrollments(lcode, scode, edate, grade) values('C401','92514009','98/03/03',85);
insert into enrollments(lcode, scode, edate, grade) values('E221','92514009','98/03/03',85);
insert into enrollments(lcode, scode, edate, grade) values('C301','96414404','98/03/04',75);
insert into enrollments(lcode, scode, edate, grade) values('C401','96414404','98/03/04',75);
insert into enrollments(lcode, scode, edate, grade) values('C421','96414404','98/03/04',75);
insert into enrollments(lcode, scode, edate, grade) values('C312','92454018','98/03/04',90);
insert into enrollments(lcode, scode, edate, grade) values('E221','92454018','98/03/04',90);
insert into enrollments(lcode, scode, edate, grade) values('A109','95454003','98/03/05',85);
insert into enrollments(lcode, scode, edate, grade) values('E221','95454003','98/03/05',85);
```

데이터베이스 조회(1)

1) 교수 테이블의 모든 데이터를 검색하시오.

```
select * from professors;
```

2) 교수 테이블에서 모든 교수의 이름, 소속학과, 직급을 검색하시오.

```
select  pname, dept, title from professors;
```

3) 교수 테이블에서 교수들이 근무하는 소속학과 이름을 검색하시오(단 , 중복 값은 제거하시오).

```
select distinct(dept) from professors;
```

4) 학생 테이블에서 '전산'이면서 '3'학년 학생들의 이름, 학번, 생년월일을 검색하시오.

```
select sname, scode, birthday from students where dept='전산' and year=3;
```

5) 교수 테이블에서 '1993년3월1일' 이전에 임용된 교수들의 이름 소속학과를 검색하시오.

```
select pname, dept from professors where hiredate < '1993/03/01';
```

6) 학생 테이블에서 교수번호가 '221'인 교수가 지도하지 않는 학생들을 검색하시오.

```
select * from students where advisor <> '221'; ( where advisor != '221' )
```

7) 수강신청 테이블에서 성적이 80점 이상인 과목번호, 학생번호를 검색하시오.

```
select lcode, scode from enrollments where grade >= 80;
```

8) 성이 '이'씨인 학생들의 이름, 학생번호, 학년, 지도교수 번호를 검색하시오.

```
select sname, scode, year, advisor from students where sname like '이%';
```

9) 강좌 테이블에서 강좌이름이 '건축'이라는 단어를 포함하는 강좌의 강좌번호, 강좌이름, 담당교수, 강의시간수를 검색하시오.

```
select lcode, lname, instructor, hours from courses where lname like '%건축%';
```

10) 수강신청 테이블에서 1998년 3월 1일에서 3월3일 사이에 수강신청 한 강좌번호, 학생번호, 수강신청일을 검색하시오.

```
select lcode, scode, edate from enrollments where edate between '98/03/01' and '98/03/03';
```

11) 학생 테이블에서 학년이 2학년과 4학년 사이에 학생들의 학번, 학생명, 학과, 학년을 검색하시오.

```
select scode, sname, dept, year from students where year between 1 and 4;
```

11) 수강 신청테이블에서 성적이 입력되지 않은 과목들의 강좌번호, 학생번호를 모두 검색하시오.

```
select lcode, scode from enrollments where grade is null;
```

12) 수강 신청테이블에서 성적이 입력된 과목들의 강좌번호, 학생번호를 모두 검색하시오.

```
select lcode, scode from enrollments where grade is not null;
```

13) 교수 테이블에서 직급이 '정교수' 이거나 '부교수'인 교수들의 교수번호, 교수명, 직급을 검색하시오.

```
select pcode, pname, title from professors where title = '정교수' or title = '부교수'
```

14) 학생 테이블에서 '전산'과 또는 '건축'과 또는 '전자'과 학생들의 이름, 소속학과, 학년을 검색하시오.

```
select sname, dept, year from students where dept in ('전산', '건축', '전자');
```

15) '전산'과 학생들의 이름, 소속학과, 생년월일을 검색하는데 소속학과, 학생이름 순으로 정렬하시오.

```
select sname, dept, birthday from students where dept='전산'
order by dept, sname;
```

16) '전산'과 학생들의 학번, 이름, 생년월일을 이름을 기준으로 오름차순, 생년월일을 기준으로 내림차순 정렬을 하시오.

```
select dept, scode, sname, birthday from students where dept='전산'
order by dept asc, birthday desc;
```

데이터베이스 조회(2)

1) 학생들의 학과, 학생이름, 지도교수이름을 검색하시오.

```
select s.dept, sname, pname from students s, professors p where advisor=pcode;
```

2) 강좌번호, 강좌명, 교수이름을 검색하시오.

```
select lcode, lname, pname from courses, professors where instructor=pcode;
```

3) 학번, 학생이름, 학생들이 수강신청 한 강좌번호, 수강신청일을 검색하시오.

```
select s.scode, sname, lcode, edate from students s, enrollments e where s.scode=e.scode;
```

4) 학번, 학생들이 수강신청 한 강좌번호, 강좌명, 성적을 검색하시오.

```
select scode, e.lcode, lname, grade from enrollments e, courses c where e.lcode=c.lcode;
```

5) '이병렬'과 교수가 지도하는 학생들의 이름, 학년, 생년월일을 검색하시오.

```
select sname, year, birthday from students, professors where pcode=advisor and pname='이병렬';
```

6) '98/03/03'에 수강신청 한 학생들의 학번, 학생이름, 강좌번호를 검색하시오.

```
select s.scode, sname, lcode from students s, enrollments e where s.scode=e.scode and edate='98/03/03';
```

7) '전산'과 교수들이 지도하는 학생들의 이름, 학년, 생년월일을 검색하시오.

```
select sname, year, birthday from students s, professors p where pcode=advisor and p.dept='전산';
```

8) '자료구조'를 강의하는 교수의 학과명, 교수 명을 검색하시오.

```
select dept, pname from professors, courses where pcode=instructor and lname='자료구조';
```

9) '파일처리론'을 수강신청 한 학생들의 학번, 수강신청일, 점수를 검색하시오.

```
select scode, edate, grade from courses c, enrollments e where c.lcode=e.lcode and lname='파일처리론';
```

10) '자료구조' 과목을 수강신청 한 학생들의 학과, 학생이름, 성적을 검색하시오.

```
select dept, sname, grade from students s, courses c, enrollments e
where s.scode=e.scode and c.lcode=e.lcode and lname='자료구조';
```

11) '전자'과 학생들의 학번, 학생이름, 수강신청 한 강좌번호, 강좌 명, 성적을 검색하시오.

```
select s.scode, sname, c.lcode, lname, grade
from students s, enrollments e, courses c
where s.scode=e.scode and c.lcode=e.lcode and dept='전자';
```

12) '파일처리론'을 강의하는 교수의 이름, 이 교수가 지도하는 학생들의 학번, 학생명을 검색하시오.

```
select pname, scode, sname
from courses c, professors p, students s
where instructor=pcode and advisor=pcode and lname='파일처리론';
```

13) '데이터베이스'를 강의하는 교수명, 이 과목을 수강신청 한 학생들의 학과, 이름, 성적을 검색하시오.

```
select s.dept, sname, grade, pname
from courses c, students s, enrollments e, professors
where c.lcode=e.lcode and e.scode=s.scode and instructor=pcode and lname='데이터베이스';
```

14) 성적이 80점 이상인 학생들의 학번, 학생이름, 강좌번호, 강좌명, 담당교수 명을 검색하시오.

```
select s.scode, sname, e.lcode, lname, pname
from studentss, enrollments e, courses c, professors p
where e.scode=s.scode and e.lcode=c.lcode and pcode=instructor and grade >= 80;
```

15)'이병렬' 교수가 강의하는 강좌번호, 강좌명, 이 강좌를 수강신청 한 학생들의 학과, 학생이름, 성적을 검색하시오.

```
select c.lcode, lname, s.dept, sname, grade
from professors, enrollments e, courses c, students s
where pcode=instructor and c.lcode=e.lcode and e.scode=s.scode and pname='이병렬';
```

데이터베이스 조회(3)

1) 교수들의 총 급여액과 평균 급여액을 구하시오.

```
select sum(salary), avg(salary) from professors;
```

2) 전산과 교수들의 총 급여액과 평균 급여액을 구하시오.

```
select sum(salary), avg(salary) from professors where dept='전산';
```

3) 수강신청 한 과목들 중에서 최고 점수와 최저 점수를 구하시오.

```
select max(grade), min(grade) from enrollments;
```

4) 전산과 학생들은 모두 몇 명인지 구하시오.

```
select count(scode) from students where dept='전산';
```

5) 수강신청 한 학생들은 모두 몇 명인지 구하시오.

```
select count(distinct(scode)) from enrollments;
```

6) 각 학과별 학생들의 수를 구하시오.

```
select dept, count(*) from students group by dept;
```

7) 교수들을 소속학과별, 직급별로 분류하여 각 직급별 평균 급여액수를 구하시오.

```
select dept, title, avg(salary) from professors group by dept, title;
```

8) 각 학생들에 대해 학번, 학생이름, 수강신청 과목들의 평균 점수를 구하시오.

```
select s.scode, sname, avg(grade) from students s, enrollments e where s.scode=e.scode
group by s.scode, sname;
```

9) 각 학생들에 대해 수강신청 과목들의 평균 점수를 구하시오.

```
select scode, avg(grade) from enrollments group by scode;
```

10) 수강신청 한 과목들을 학생별로 그룹핑하여 평균 점수를 구한 다음, 학생번호, 평균 점수를 성적이 높은 순으로 정렬하여 출력하시오.

```
select scode, avg(grade) from enrollments group by scode order by avg(grade) desc;
```

11) 수강신청 과목들의 평균 점수가 85점 이상인 학생들의 학생번호, 평균 점수를 구하시오.

```
select scode, avg(grade) from enrollments group by scode having avg(grade) >= 85;
```

12) 강좌별 평균점수가 80점 이상인 강좌들의 강좌번호와 평균점수를 출력하시오.

```
select lcode, avg(grade) from enrollments group by lcode having avg(grade) >= 80;
```

13) 학생수가 3명 이상인 학과 구한 다음, 학과명, 학생수를 출력하시오.

```
select dept, count(scode) from students group by dept having count(scode) >= 3;
```

14) 수강신청 평균점수가 85점 이상인 학생들의 학생번호, 학생이름, 평균 점수를 평균점수가 높은 순으로 출력하시오.

```
select s.scode, sname, avg(grade) from students s, enrollments e where s.scode=e.scode
group by s.scode, sname
having avg(grade) >= 85
order by avg(grade) desc;
```

15) 강좌별 평균점수가 80점 이상인 강좌들의 강좌번호, 강좌명, 평균점수를 평균점수가 높은 순으로 출력하시오.

```
select s.scode, sname, avg(grade) from students s, enrollments e where s.scode=e.scode
group by s.scode, sname
having avg(grade) >= 85
order by avg(grade) desc;
```

데이터베이스 조회(4)

1) '알고리즘'을 강의하는 교수의 교수번호, 교수이름, 소속학과를 검색하시오.

```
select pcode, pname, dept from professors
where pcode in (select instructor from courses where lname='알고리즘');
```

2) 강의실 '510'호에서 강의하는 교수의 교수번호, 교수이름, 소속학과를 검색하시오.

```
select pcode, pname, dept from professors
where pcode in (select instructor from courses where room='510');
```

3) '김창덕' 학생이 소속된 학과에 재직하는 교수들의 이름, 직급, 임용일자를 검색하시오.

```
select pname, title, hiredate from professors
where dept in (select dept from students where sname='김창덕');
```

4) 수강신청 과목의 점수가 90점 이상인 학생들의 이름, 학생번호, 소속학과, 학년을 검색하시오.

```
select sname, scode, dept, year from students
where scode in (select scode from enrollments where grade >= 90);
```

5) '전산'과 교수들이 담당하는 강좌의 이름, 강의시간수, 강의실을 검색하시오.

```
select lname, hour, room from courses
where pcode in (select pcode from professors where dept='전산');
```

6) '98/03/02'에 수강신청 한 학생들의 학과, 학번, 학생이름, 학년을 검색하시오.

```
select dept, scode, sname, year from students
where scode in (select scode from enrollment where edate='98/03/02');
```

7) '509'호에서 강의를 듣는 학생들의 학과, 학번, 학생이름을 검색하시오.

```
select dept, scode, sname from students
where scode in (select scode from enrollment where lcode in (select lcode from courses where room='509'));
```

8) 수강신청 과목의 평균점수가 80점 이상인 학생들의 이름, 학생번호, 소속학과, 학년을 검색하시오.

```
select sname, scode, dept, year
where scode in (select scode from enrollments group by scode having avg(grade) >=80);
```

9)'건축'과 학생들을 지도하는 교수의 이름, 교수번호, 소속학과, 직급을 검색하시오.

```
select pname, pcode, dept, title from professors
where pcode in (select advisor from students where dept='건축')
```

10) 학생수가 '3'명 이상인 학과에 근무하는 교수들의 이름, 소속학과, 직급을 검색하시오.

```
select pname, dept, title from professors
where dept in (select dept from students group by dept having count(scode) >= 3);
```

11)'이원구'가 수강신청한 과목의 번호, 과목명, 점수를 검색하시오.

```
select lcode, lname, grade
where lcode in (select lcode from enrollments where scode in (select scode from students where sname='이원구'));
```

12) '알고리즘'을 수강신청한 학생들의 학번, 학생이름, 학과를 검색하시오.

```
select scode, sname, dept from students
where scode in (select scode from enrollments where lcode in (select lcode from courses where lname='알고리즘'));
```

13)'1973'년생 학생들을 지도하는 교수들의 이름, 소속학과, 직급을 검색하시오.

```
select pname, dept, title from professors
where pcode in (select advisor from students where to_char(birthday,YYYY)= '1973');
```

14)전체 학생의 30% 이상이 수강신청한 강좌의 번호를 검색하시오.

```
select lcode from enrollments group by lcode
having count(*) >= (select count(*) * 0.3 from students);
```

15)'이병렬' 교수가 강의하는 강좌를 수강신청한 학생들의 학과, 학번, 학생이름을 검색하시오.

```
select dept, scode, sname from students where scode in (select scode from enrollments
where lcode in (select lcode from professors where instructor in (select pcode from professors where pname='이병렬')));
```

데이터베이스 갱신(1)

1) '98414022', '노진순', '75-05-10', '전산' 값을 학생테이블에 삽입하시오.

```
insert into students(scode, sname, birthday, dept, year) values('98414022', '노진순', '79-05-10', '전산', '1');
```

2) 노진순의 수강신청 내용을 수강신청(Enrollments)테이블에 삽입하시오.

```
insert into enrollments(lcode, scode, edate) values('C301', '98414022', sysdate);
```

3) '1998년 1월 1일' 이전에 발생한 모든 수강신청 데이터를 oldEnrollments테이블로 복사하시오.

```
insert into oldEnrollments(lcode, scode, edate, grade)
select lcode, scode, edate, grade from enrollments where edate < '98-03-03';
```

4) 4학년 학생들의 모든 학생 데이터를 oldstudents 테이블로 복사하시오.

```
insert into oldStudents(scode, sname, dept, year, birthday, advisor)
select scode, sname, dept, year, birthday, advisor from students where year=4;
```

5) 학생테이블에서 '노진순'의 데이터를 삭제하시오.

```
delete from enrollments
where scode in (select e.scode from enrollments e, students s where e.scode=s.scode and sname='노진순');

delete from students where sname='노진순';
```

6) '1998년 1월 1일' 이전에 신청한 모든 수강신청 데이터를 삭제하시오.

```
delete from enrollments where edate < '1998-1-1';
```

7) 수강신청 한 과목에 대해 성적을 아직 받지 못한 수강신청 데이터를 삭제하시오.

```
delete from enrollments where grade is null;
```

8) 수강신청 데이터(oldenrollments)를 모두 삭제하시오.

```
delete from oldenrollments;
```

9) 학생테이블에서 전산과 3학년 데이터를 4학년으로 변경하시오.

```
update students set year=4 where dept='전산' and year=3;
```

10) '오문환' 교수의 직급을 '조교수'에서 '부교수'로 변경하시오.

```
update professors set title='부교수' where pname='오문환';
```

11) '건축과' 학생이 신청한 모든 수강신청 데이터를 삭제하시오.

```
delete from enrollments where scode in (select scode from students where dept='건축');
```

12) '전산'과 교수들의 급여를 10% 증가 시키시오.

```
update professors set salary = salary *1.1 where dept='전산';
```

13) 모든 교수들의 급여를 10% 증가 시키시오.

```
update professors set salary = salary * 1.1;
```

14)'전자'과 학생들이 신청한 수강신청 데이터를 모두 삭제하시오.

```
delete from enrollments
where scode in (select scode from students where dept='전산');
```

15)'전산'과 교수가 담당하는 강좌의 강의실을 모두 '123'호실로 변경하시오.

```
update courses set room= '123'
where instructor in (select pcode from professors where dept='전산');
```

16) 학생테이블에서 4학년 학생들을 모두 삭제하고 나머지 학생들의 학년을 1씩 증가 시키시오.

```
delete from enrollments
where scode in (select scode from students where year=4);

delete from students where year=4;

update students set year = year + 1;
```

데이터베이스 갱신(2)

1) '이재광' 교수가 지도하는 학생들의 지도교수를 교수번호 '221'로 변경하시오.

```
update students set advisor='221' where advisor in (select pcode from professors where pname='이재광');
```

2) '파일처리론' 과목을 수강신청 한 학생들의 점수를 5점씩 증가 시키시오.

```
update enrollments set grade=grade+5 where lcode in (select lcode from courses where lname='파일처리론');
```

3) '전자'과 학생들이 수강신청 한 수강신청 데이터의 점수를 0점 처리 하시오.

```
update enrollments set grade=0 where scode in (select scode from students where dept='전자');
```

4) '서연우' 학생의 지도교수가 강의하는 강좌의 강의실을 '304'호로 변경하시오.

```
update courses set room='304' where instructor in (select advisor from students where sname='서연우');
```

5) '논리회로'를 강의하는 교수의 급여를 100000원 인상하시오.

```
update professors set salary=salary+1000000 where pcode in (select instructor from courses where lname='논리회로');
```

6) 수강신청인원수가 80명 이상인 강좌를 강의하는 교수들의 급여를 100000원 인상하시오.

```
update professors set salary=salary+100000
where pcode in (select instructor from courses where persons >= 80);
```

7) '서연우' 학생의 모든 과목 점수를 5점씩 감소 시키시오.

```
update enrollments set grade=grade-5
where scode in(select scode from students where sname='서연우');
```

8) '전산'과 3학년 학생들이 수강신청 한 과목들의 성적을 5점씩 증가 시키시오.

```
update enrollments set grade=grade+5
where scode in(select scode from students where dept='전산' and year=3);
```

9) '전산'과 '부교수'가 강의하는 강의시간수를 1씩 증가 시키시오.

```
update courses set hours=hours+1 where instructor in (select pcode from professors where dept='전산' and title='부교수');
```

10) 수강신청 한 과목이3과목 이상인 학생들의 학년을 1씩 증가 시키시오.

```
update students set year=year+1
where code in(select scode from enrollments group by scode having count(lcode) > =3);
```

11) 수강신청 평균점수가 80점 미만인 학생들의 학년을 1씩 감소시키시오.

```
update students set year=year-1
where scode in (select scode from enrollments group by scode having avg(grade) < 80);
```

12) '파일처리론'을 수강신청 한 학생들의 학과를 '건축'으로 수정하시오.

```
update students set dept='건축'
where scode in (select scode from enrollments e, courses c where lname='파일처리론' and c.lcode=e.lcode);
```

13) 강좌별 평균점수가 80점 이상인 과목들의 강의실을 '509'호로 변경하시오.

```
update courses set room='509'
where lcode in (select lcode from enrollments group by lcode having avg(grade) >= 80);
```

14) '오문환' 교수가 강의하는 강좌를 신청한 수강신청 데이터를 삭제하시오.

```
delete from enrollments
where lcode in (select lcode from courses, professors where pname='오문환' and pcode=instructor);
```

15) '자료구조'를 수강신청 한 학생의 학년을 1씩 증가 시키시오.

```
update students set year=year + 1
where code in (select e.scode from courses c, enrollments e where lname='자료구조' and c.lcode=e.lcode);
```

16) '이병렬' 교수가 강의하는 강좌를 수강신청 한 학생들의 학년을 1씩 감소시키시오.

```
update students set year=year-1
where scode in (select e.scode from professors p, courses c, enrollments e
where pname='이병렬' and p.pcode=instructor andc.lcode=e.lcode);
```

View (가상테이블)

view생성

create view 뷰명 as (select문);
create view 뷰명(열이름1, 열이름2, …) as select 열이름1, 열이름2,…;

정의된 view 확인

select * from user_views;

정의된 view 삭제

drop view 뷰명;

1) 학생테이블에 지도교수명을 추가하는 view를 생성하시오.

```
create view view01 as
(select student.*, pname from students, professors where pcode=advisor);
```

2) 강좌테이블에 담당교수명을 추가하는 view를 생성하시오.

```
create view view02 as
(select courses.*, pname from courses, professors where pcode=instructor);
```

3) 수강신청 테이블에 강좌명과, 학생명을 추가하는 view를 생성하시오.

```
create view view03 as
(select e.*, sname, lname from enrollments e, courses c, students s where e.lcode=c.lcode and e.scode=s.scode);
```

4) 강좌테이블에서 강좌번호, 강좌이름, 담당교수, 최대수강인원수에서 현재수강인원수를 뺀 값을 나타내는 뷰를 생성하시오.

```
create view view04(lcode, lname, instructor, diff) as
(select lcode, lname, instructor, (capacity-persons) from courses);
```

5) '전산'과 학생들의 학번, 이름, 이 학생들이 수강신청 한 강좌들의 평균을 구하는 view를 생성하시오.

```
create view view05 as
(select e.scode, sname, avg(grade) from students s, enrollments e where s.scode=s.scode and dept='전산'
group by e.scode, sname);
```

6) 수강신청 테이블에서 강좌명과 담당교수명, 이 강좌들의 평균을 구하는 view를 생성하시오.

```
create view view06 as
(select lname, pname, avg(grade) from professors p, courses c, enrollments e where instructor=pcode and c.lcode=e.lcode
group by lname, pname);
```

7) '전산'과 학생 중 평균점수가 80점 이상인 학생들의 학번, 학생명, 평균 점수를 구하는 view를 생성하시오.

```
create view view07 as
(select s.scode, sname, avg(grade) from students s, enrollments e where e.scode=s.scode and s.dept='전산'
group by s.scode, sname
having avg(grade)>=80);
```

8) '전산'과 교수들이 강의하는 강좌의 강좌명과 이 강좌들의 평균을 구하는 view를 생성하시오.

```
create view view08 as
(select lname, avg(grade) from professors p, enrollments e, courses c
where pcode=instructor and c.lcode=e.lcode and dept='전산'
group by lname);
```

9) 평균점수가 80점 이상인 강좌의 강좌명들과 평균을 구하는 view를 생성하시오.

```
create view view09 as
(select lname, avg(grade) from courses c, enrollments e where c.lcode=e.lcode
group by lname
having avg(grade)>=80);
```

10) 평균점수가 80점 이상인 학생들의 학생명, 강좌명, 평균을 구하는 view를 생성하시오.

```
create view view10 as
(select sname, lname, avg(grade) from students s, courses c, enrollments e where e.scode=s.scode and e.lcode=c.lcode
group by sname, lname
having avg(grade)>=80);
```

PL/SQL(저장 프로시저)

저장프로시저 생성

```
CREATE OR REPLACE PROCEDURE 저장프로시저명 IS
BEGIN
...
END;
```

저장프로시저 관련 SQL문

```
EXEC 저장프로시저명;  /*저장 프로시저 실행*/
select * from user_objects where object_type='PROCEDURE';  /*저장 프로시저 확인*/
select text from user_source where name='저장프로시저명';  /*저장프로시저 내용확인 */
```

1)'HELLO WORLD!'를 출력하는 저장 프로시저 작성.

exec hello_word;

```
SET SERVEROUTPUT ON; /* 결과를 화면에 출력하기 위한 설정 */
CREATE OR REPLACE PROCEDURE HELLO_WORLD
AS
    o_message varchar2(100):='HELLO WORLD';
BEGIN
    DBMS_OUTPUT.PUT_LINE(o_message);
END HELLO_WORLD;
```

2) 파라미터 변수를 사용하여 파라미터 변수 값을 출력하는 저장 프로시저 작성

exec hello_word1('안녕하세요!');

```
CREATE OR REPLACE PROCEDURE HELLO_WORLD1(i_message in varchar)
AS
BEGIN
    DBMS_OUTPUT.PUT_LINE(i_message);
END HELLO_WORLD1;
```

3) 1~100까지의 합을 구하는 저장 프로시저 작성 (LOOP문 이용)

exec_out_sum;

```
CREATE OR REPLACE PROCEDURE OUT_SUM
AS
     n int := 0;
     tot int := 0;
BEGIN
     LOOP
         n := n + 1;
         tot := tot + n;
         EXIT WHEN n = 100;
     END LOOP;
     DBMS_OUTPUT.PUT_LINE('1~100까지의 합:' || tot);
END OUT_SUM;
```

4) 1~100까지의 합을 구하는 저장 프로시저 작성 (FOR문 이용)

exec out_sum1;

```
CREATE OR REPLACE PROCEDURE OUT_SUM1
AS
    tot int := 0;
BEGIN
    FOR i IN 1..100 LOOP
        tot := tot + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('1~100까지의 합:' || tot);
END OUT_SUM1;
```

5) 교수 테이블에 교수 한명을 추가하는 저장 프로시저 작성

```
exec add_professors('101', '홍길동', '전산', sysdate, '부교수', 3200000);
```

```
CREATE OR REPLACE PROCEDURE ADD_PROFESSORS(
    i_pcode in professors.pcode%TYPE,
    i_pname in professors.pname%TYPE,
    i_dept in professors.dept%TYPE,
    i_hiredate in professors.hiredate%TYPE,
    i_title in professors.title%TYPE,
    i_salary in professors.salary%TYPE)
AS
BEGIN
    insert into professors(pcode,pname,dept,hiredate,title,salary) values(i_pcode,i_pname,i_dept,i_hiredate,i_title,i_salary);
    commit;
END ADD_PROFESSORS;
```

DAO.java insert(PVO vo)

```
String sql="{call add_professors(?,?,?,?,?,?)}";
CallableStatement cs=con().prepareCall(sql);
...
cs.execute();
```

6) 교수번호를 입력하면 교수이름을 출력하는 저장 프로시저 작성

```
exec out_pname('512');
```

```
CREATE OR REPLACE PROCEDURE  OUT_PNAME(i_pcode in professors.pcode%TYPE)
AS
    o_pname professors.pname%TYPE;
BEGIN
    select pname into o_pname from professors where pcode=i_pcode;
    DBMS_OUTPUT.PUT_LINE('학번:' || i_pcode);
    DBMS_OUTPUT.PUT_LINE('이름:' || o_pname);
END OUT_PNAME;
```

7) 교수 학과명을 입력하면 평균 급여를 출력하는 저장 프로시저 작성

```
exec out_salary('전산');
```

```
CREATE OR REPLACE PROCEDURE OUT_SALARY(p_dept in professors.dept%TYPE)
AS
    i_salary int;
BEGIN
    select avg(salary) into i_salary from professors where dept=p_dept;
    DBMS_OUTPUT.PUT_LINE(p_dept || ' 평균 연봉:' || i_salary);
END OUT_SALARY;
```

8) 교수 테이블에서 특정학과 교수들을 출력하고 인원수룰 출력하는 저장 프로시저를 작성하시오.

```
exec out_dept('전산');
```

```
CREATE OR REPLACE PROCEDURE OUT_DEPT(i_dept in professors.dept%TYPE)
AS
    CURSOR cur_professors IS select pcode,pname,dept,title from professors where dept=i_dept;
    o_pcode professors.pcode%TYPE;
    o_pname professors.pname%TYPE;
    o_dept professors.dept%TYPE;
    o_title professors.title%TYPE;
BEGIN
    OPEN cur_professors;
    LOOP
        FETCH cur_professors INTO o_pcode, o_pname, o_dept, o_title;
        EXIT WHEN cur_professors%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(o_pcode || o_pname || o_dept || o_title);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('전체인원:' || cur_professors%ROWCOUNT);
    CLOSE cur_professors;
END OUT_DEPT;
```

9) 교수 테이블에서 특정학과 교수들의 급여를 정교수인 경우 500, 부교수인 경우 100을 인상하는 저장 프로시저를 작성하시오.

```
exec update_salary('전산');
```

```
CREATE OR REPLACE PROCEDURE UPDATE_SALARY(i_dept in professors.dept%TYPE)
AS
    CURSOR cur_professors IS select pcode, salary, title from professors where dept='전산';
    o_pcode professors.pcode%TYPE;
    o_salary professors.salary%TYPE;
    o_title professors.title%TYPE;
BEGIN
    OPEN cur_professors;
    LOOP
        FETCH cur_professors INTO o_pcode, o_salary, o_title;
        EXIT WHEN cur_professors%NOTFOUND;
        IF(o_title='정교수')THEN
            o_salary := o_salary + 500;
        ELSE
            o_salary := o_salary + 100;
        END IF;
        update professors set salary=o_salary where pcode=o_pcode;
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('수정한 교수: ' || cur_professors%ROWCOUNT);
END UPDATE_SALARY;
```

10)수강신청 테이블에서 두 값 사이에 점수가 존재하면 출력하는 저장 프로시저를 작성하시오.

```
exec out_grade(90, 95);
```

```
CREATE OR REPLACE PROCEDURE OUT_GRADE(i_min in int, i_max in int)
    AS
        CURSOR cur_enrollments IS select lcode,scode,grade from enrollments where grade between i_min and i_max;
        o_lcode enrollments.lcode%TYPE;
        o_scode enrollments.scode%TYPE;
        o_grade enrollments.grade%TYPE;
        chkrange_err EXCEPTION;
    BEGIN
        IF(i_max > 100) OR (i_min < 0) THEN
            RAISE chkrange_err;
        END IF;
        OPEN cur_enrollments;
        LOOP
            FETCH cur_enrollments INTO o_lcode, o_scode, o_grade;
            EXIT WHEN cur_enrollments%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(o_lcode || ':' || o_scode || ':' || o_grade);
        END LOOP;
        CLOSE cur_enrollments;
    EXCEPTION
        WHEN  chkrange_err THEN DBMS_OUTPUT.PUT_LINE('입력한 두 점수를 확인하세요!');
        WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('관리자에게 문의 하세요');
END OUT_GRADE;
```

10) 학과를 입력 받아 해당 교수들 데이터를 CURSOR에 저장한 후 리턴 하는 저장 프로시저를 작성하시오.

```
variable cur refcursor;  exec pro_list(:cur, '전산');  print cur;
```

```
create or replace PROCEDURE PRO_LIST(
    cur out SYS_REFCURSOR,
    i_dept in students.dept%TYPE)
AS
BEGIN
  OPEN cur FOR
    select * from students where dept=i_dept;
END PRO_LIST;
```

DAO.java list()

```
String sql="{call pro_list(?, ?)}";
CallableStatement cs=con().prepareCall(sql);
cs.registerOutParameter(1, oracle.jdbc.OracleTypes.CURSOR);
cs.setString(2, "전자");
cs.execute();
ResultSet rs=(ResultSet)cs.getObject(1);
```

11) 학생테이블에 새로운 학생을 추가할 경우 학생이 존재하지 않을 경우만 입력되는 저장 프로시저를 작성하시오.

variable cnt number; exec add_students('92414029', '심청이', :cnt); print cnt;

```
CREATE OR REPLACE PROCEDURE ADD_STUDENTS (
    i_scode in students.scode%TYPE,
    i_sname in students.sname%TYPE,
    o_count out int)
AS
BEGIN
    select count(*) into o_count from students where scode=i_scode;
    IF(o_count = 0) THEN
        insert into students(scode, sname) values(i_scode, i_sname);
    END IF;
    commit;
END ADD_STUDENTS;
```

DAO.java

```
public int insert(String scode, String sname, String year)throws Exception{
    String sql="{call add_students(?, ?, ?)}";
    CallableStatement cs=con().prepareCall(sql);
    cs.setString(1, scode);
    cs.setString(2, sname);
    cs.registerOutParameter(3, oracle.jdbc.OracleTypes.INTEGER);
    cs.execute();
    int cnt=(int)cs.getObject(4);
    return cnt;
}
```

insert.jsp

```
<%@ page import="ex0504.*, org.json.simple.*"%>
<%
    String scode=request.getParameter("scode");
    String sname=request.getParameter("sname");

    StudentsDAO dao=new StudentsDAO();
    int cnt=dao.insert(scode, sname, year);
    JSONObject jObject = new JSONObject();
    jObject.put("count", cnt);
    out.print(jObject);
%>
```

DAO.java

```
<body>
    <div>
        학번: <input type="text" id="txtScode">
        학생명: <input type="text" id="txtSname">
        <input type="button" id="btnSave" value="저장">
    </div>
</body>
<script>
    $("#btnSave").on("click", function(){
        var scode=$("#txtScode").val();
        var sname=$("#txtSname").val();
        $.ajax({
            type:"post",
            url:"insert.jsp",
            dataType:"json",
            data:{"scode":scode, "sname":sname},
            success:function(data){
                if(data.count==1){
                    alert("이미 존재하는 학번입니다.");
                }else{
                     alert("학생이 입력되었습니다.");
                }
            }
        });
    });
</script>
```

Trigger (트리거)

트리거 내용 확인

```
select * from user_objects where object_type='TRIGGER';
```

트리거 확인

```
select * from user_objects where object_type='TRIGGER';
```

1) 학생 테이블에 학생 한명을 추가하면 '정상적으로 추가되었습니다!' 메시지를 출력하는 트리거를 작성하시오.

```
insert into students(scode, sname, birthday, dept, year) values('98414022', '노진순', '79-05-10', '전산', '1');
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER OUT_INSERT
BEFORE INSERT ON students
BEGIN
  DBMS_OUTPUT.PUT_LINE('정상적으로 추가되었습니다.');
END;
```

2) 학생 테이블의 학년이 1 ~ 4사이를 벗어나면 추가 내지 수정이 발생하지 않도록 하는 트리거를 작성하시오.

```
update students set year=5;
```

```
CREATE OR REPLACE TRIGGER RANGE_YEAR
BEFORE INSERT OR UPDATE ON students
FOR EACH ROW
BEGIN
    IF :new.year NOT BETWEEN 1 AND 4 THEN
        RAISE_APPLICATION_ERROR(-20001,'학년이 범위를 벗어났습니다.');
    END IF;
END;
```

3) 교수들의 급여가 기존 교수들의 최대, 최소 연봉을 벗어나면 추가 내지 수정이 발생하지 않도록 하는 트리거를 작성하시오.

```
insert into professors(pcode, pname, dept, salary) values('229','홍길동','전산',30000000);
```

```
CREATE OR REPLACE TRIGGER RANGE_SALARY
BEFORE INSERT OR UPDATE ON professors
FOR EACH ROW
DECLARE
    max_salary professors.salary%TYPE;
    min_salary professors.salary%TYPE;
BEGIN
    select max(salary), min(salary) into max_salary, min_salary from professors;
    IF :new.salary NOT BETWEEN min_salary AND max_salary THEN
        RAISE_APPLICATION_ERROR(-20001,'급여의 범위를 확인하세요!');
    END IF;
END;
```

4) 교수 테이블에 수정 작업이 발생한 경우에 변경 전후의 값을 'oldProfessors' 테이블에 기록하는 트리거를 작성하시오.

oldProfessors 테이블 생성

```
create table oldProfessors(
    / *변경전 데이터를 저장하는 컬럼은 시작을 old의 약자인 o로 정의 */
    opcode char(3),opname char(15), odept varchar(30), ohiredate date, otitle char(15), osalary int,

    /* 변경 후 추가된 데이터를 저장하는 컬럼은 시작을 new의 약자은 n으로 정의 */
    npcode char(3), npname char(15), ndept  varchar(30), nhiredate date, ntitle char(15), nsalary int,

    /* 어떤 사용자가 어떤 작업을 언제 작업을 했는지 정보를 저장 */
    wuser varchar(30),
    wwhen date,
    chk char(1)
);
```

update professors set dept='전산' where pcode='221';

```
CREATE OR REPLACE TRIGGER HISTORY_PROFESSORS
AFTER UPDATE ON professors
FOR EACH ROW
BEGIN
    IF UPDATING THEN
        insert into oldProfessors values(
            :OLD.pcode, :OLD.pname, :OLD.dept, :OLD.hireDate, :OLD.title, :OLD.salary,
            :NEW.pcode, :NEW.pname, :NEW.dept, :NEW.hireDate, :NEW.title, :NEW.salary,
            user, sysdate, 'U');
    END IF;
END;
```

5)수강신청 테이블에 데이터를 삭제하면 'delEnrollments' 테이블에 기록하는 트리거를 작성하시오.

delEnrollments 테이블 생성

```
create table delEnrollments(
    lcode char(4),
    scode char(8),
    edate date,
    grade int,
    wuser varchar(30),
    wwhen date,
    chk char(1)
);
```

delete from enrollments where lcode='C401';

```
CREATE OR REPLACE TRIGGER DEL_PROFESSORS
AFTER DELETE ON enrollments
FOR EACH ROW
BEGIN
    IF DELETING THEN
        insert into delEnrollments values(:OLD.lcode, :OLD.scode, :OLD.edate, :OLD.grade, user, sysdate, 'D');
    END IF;
END;
```

6)강좌 테이블의 강좌를 삭제하면 수강신청 테이블에 해당 강좌를 삭제하는 트리거를 작성하시오.

delete from courses where lcode='C501';

```
CREATE OR REPLACE TRIGGER Del_COURSES
BEFORE DELETE ON courses
FOR EACH ROW
BEGIN
    IF DELETING THEN
        delete from enrollments where lcode=:OLD.lcode;
    END IF;
END;
```

7)학생 테이블의 학생을 삭제하면 수강신청 테이블에 해당 학생을 삭제하는 트리거를 작성하시오.

delete form students where scode='92414031';

```
CREATE OR REPLACE TRIGGER DEL_STUDENTS
BEFORE DELETE ON students
FOR EACH ROW
BEGIN
    IF DELETING THEN
        delete from enrollments where scode=:OLD.scode;
    END IF;
END;
```

쇼핑몰 데이터베이스

- **업체(mall) 테이블의 구조**

열이름	데이터형	정보내용
mall_id	char(4)	업체코드
mall_name	varchar(50)	업체명
manager	varchar(15)	담당자
address	varchar(100)	회사위치
tel	varchar(15)	전화
email	varchar(20)	이메일
detail	varchar(100)	참조사항

- **상품(product)테이블의 구조**

열이름	데이터형	정보내용
prod_id	char(4)	상품코드
prod_name	varchar(50)	상품명
mall_id	char(4)	업체코드
company	varchar(50)	제조원/수입원
price1	int	판매가격
price2	int	일반가격
detail	varchar(100)	상품설명

- **주문(order)테이블의 구조**

열이름	데이터형	정보내용
order_id	char(4)	주문번호
prod_id	char(4)	구매 상품코드
price	int	상품 구매가격
quantity	int	상품 구매수량

- **주문자(purchase)테이블의 구조**

열이름	데이터형	정보내용
order_id	char(4)	주문번호
name	char(10)	주문자명
address	varchar(100)	주문자 주소
email	varchar(20)	주문자 이메일
tel	char(15)	주문자 연락전화번호
pdate	date	구매날짜
payType	char(1)	결제방법 (0:무통장 1:카드)
status	char(1)	처리상태 (0:처리중 1:완료)

- **업체(mall) 테이블의 데이터**

mall_id	mall_name	manager	address	tel	email	detail
M101	제로드 컴퓨터	이병렬	서울 강서구	02-522-6898	Bllee@hanmail.net	
M102	인터넷 컴퓨터	이재광	서울 강북구	02-897-6690	Jglee@netian.com	
M103	넷나라	서연우	인천 부평구	032-502-9599	Ywshu@netian.com	
M104	컴퓨터 뱅크	강승일	서울 강남구	02-602-9876	Sigang@netian.com	
M105	프린터 나라	이원구	인천 남동구	032-503-9021	Wglee@hanmail.net	

- **상품(product) 테이블의 데이터**

prod_id	prod_name	mall_id	company	price1	price2	detail
P101	드림시스템인지업	M101	삼보	2200000	2300000	
P102	넥스티어	M101	삼보	2100000	2200000	
P103	싱크마스터 700	M102	삼성	3000000	3200000	
P104	매직스테이션	M102	삼성	1500000	1600000	
P105	멀티넷 760	M101	삼보	1600000	1650000	
P106	센스 720	M102	삼성	2500000	2600000	
P107	데스크젯 695C	M101	삼보	300000	320000	
P108	LG-17인치 LCD	M103	LG	400000	420000	
P109	삼성완전평면 17인치	M104	삼성	520000	550000	
P110	알파스캔	M105	LG	100000	120000	
P111	공간도우미	M102	삼성	120000	130000	
P112	삼성전자 플래쉬 메모리	M102	삼성	150000	167000	

- **구매자(purchase) 테이블의 구조**

order_id	name	address	email	tel	pdate	payType	status
R101	김수정	서울 강남구 대치1동	Sjkim@netian.com	02-503-9090	2014/08/10	1	1
R102	박혜경	인천 부평구 부평5동	Hkpark@hanmail.net	032-609-9987	2014/08/10	1	1
R103	임미숙	인천 서구 청라2동	mslim@hanmail.net	032-256-9987	2014/08/13	1	1
R104	김연호	서울 종로구 관훈동 관훈빌딩	Yhkim@yahoo.com	02-688-7703	2014/08/13	1	0
R105	한보연	서울 종로구 인사동	byhan@yahoo.com	02-230-2044	2014/08/13	1	1
R106	김수정	서울 강남구 대치1동	Sjkim@netian.com	02-503-9090	2014/08/13	0	0
R107	이준호	인천 계양구 효성동	Jhlee@hanmil.net	032-522-5678	2014/08/14	0	1
R108	김연호	서울 종로구 관훈동 관훈빌딩	Yhkim@yahoo.com	02-688-7703	2014/08/15	1	1
R109	김수정	서울 강남구 대치1동	Sjkim@netian.com	02-503-9090	2014/08/15	1	1
R110	이준호	인천 계양구 효성동	Jhlee@hanmil.net	032-522-5678	2014/08/15	0	0

- **주문(orders) 테이블의 데이터**

order_id	prod_id	price	quantity
R101	P109	520000	1
R102	P106	2500000	1
R102	P112	150000	1
R103	P107	300000	2
R104	P109	520000	2
R104	P107	300000	1
R104	P111	120000	1

order_id	prod_id	price	quantity
R105	P110	100000	1
R106	P112	150000	2
R107	P112	150000	4
R108	P104	1500000	2
R108	P108	400000	1
R109	P103	3000000	1
R110	P109	520000	1

쇼핑몰 user 생성

```
create user shop identified by pass;
grant connect, resource, dba to shop;
```

업체 테이블 생성

```
create table mall(
    mall_id char(4) not null,
    mall_name varchar(50) not null,
    manager varchar(15),
    address varchar(100),
    tel varchar(15),
    email varchar(20),
    detail varchar(100),
    primary key(mall_id)
);
```

상품 테이블 생성

```
create table product(
    prod_id char(4) not null,
    prod_name varchar(50) not null,
    mall_id char(4),
    company varchar(50),
    price1 int,
    price2 int,
    detail int,
    primary key(prod_id),
    foreign key(mall_id) references mall(mall_id)
);
```

구매자 테이블 생성

```
create table purchase(
    order_id char(4) not null,
    name varchar(15) not null,
    address varchar(100),
    email varchar(20),
    tel varchar(15),
    pdate date,
    payType char(1),
    status char(1),
    primary key(order_id)
);
```

주문상품 테이블 생성

```
create table orders(
    order_id char(4) not null,
    prod_id char(4) not null,
    price int,
    quantity int,
    primary key(order_id, prod_id),
    foreign key (order_id) references purchase(order_id),
    foreign key (prod_id) references product(prod_id)
);
```

업체 테이블 데이터입력

```
insert into mall(mall_id, mall_name, manager, address, tel, email)
    values('M101', '제로드 컴퓨터', '이병력', '서울 강서구', '02-522-6898','Bllee@hanmail.net');
insert into mall(mall_id, mall_name, manager, address, tel, email)
    values('M102', '인터넷 컴퓨터', '이재광', '서울 강북구', '02-897-6690','Jglee@netian.com');
insert into mall(mall_id, mall_name, manager, address, tel, email)
    values('M103', '넷나라', '서연우', '인천 부평구', '032-502-9599','Ywshu@netian.com');
insert into mall(mall_id, mall_name, manager, address, tel, email)
   values('M104', '컴퓨터 뱅크', '강승일', '서울 강남구', '02-602-9876','Sigang@netian.com');
insert into mall(mall_id, mall_name, manager, address, tel, email)
    values('M105', '프린터 나라', '이원구', '인천 남동구', '032-503-9021','Wglee@hanmail.net');
```

상품 테이블 데이터입력

```
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
values('P101', '드림시스템인지업', 'M101', '삼보', 2200000, 2300000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P102', '넥스티어', 'M101', '삼보', 2100000, 2200000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P103', '싱크마스터 700', 'M102', '삼성', 3000000, 3200000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P104', '싱크마스터 700', 'M102', '삼성', 1500000, 1600000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P105', '멀티넷 760', 'M101', '삼보', 1600000, 1650000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P106', '센스 720', 'M102', '삼성', 2500000, 2600000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P107', '데스크젯 695C', 'M101', '삼보', 300000, 320000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P108', 'LG-17인치 LCD', 'M103', 'LG', 400000, 420000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P109', '삼성완전평면 17인치', 'M104', '삼성', 520000, 550000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P110', '삼성완전평면 17인치', 'M105', 'LG', 100000, 120000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P111', '삼성완전평면 17인치', 'M102', '삼성', 120000, 130000);
insert into product(prod_id, prod_name, mall_id, company, price1, price2)
    values('P112', '삼성완전평면 17인치', 'M102', '삼성', 150000, 167000);
```

구매자 테이블 데이터입력

```
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R101', '김수정', '서울 강남구 대치1동', 'Sjkim@netian.com', '02-503-9090','2014/08/10', '1', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R102', '박혜경', '인천 부평구 부평5동', 'Hkpark@hanmail.net', '032-609-9987','2014/08/10', '1', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R103', '임미숙', '인천 서구 청라2동', 'mslim@hanmail.net', '02-688-7703','2014/08/13', '1', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R104', '김연호', '서울 종로구 관훈동 관훈빌딩', 'Yhkim@yahoo.com', '02-688-7703','2014/08/13', '1', '0');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R105', '한보연', '서울 종로구 인사동', 'byhan@yahoo.com', '02-230-2044','2014/08/13', '1', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R106', '김수정', '서울 강남구 대치1동', 'Sjkim@netian.com', '02-503-9090','2014/08/13', '0', '0');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R107', '이준호', '인천 계양구 효성동', 'Jhlee@hanmil.net', '032-522-5678','2014/08/14', '0', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R108', '김연호', '서울 종로구 관훈동 관훈빌딩', 'Yhkim@yahoo.com', '02-688-7703','2014/08/15', '1', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R109', '김수정', '서울 강남구 대치1동', 'Sjkim@netian.com', '02-503-9090','2014/08/15', '1', '1');
insert into purchase(order_id, name, address, email, tel, pdate, payType, status)
    values('R110', '이준호', '인천 계양구 효성동', 'Jhlee@hanmil.net', '032-522-5678','2014/08/15', '0', '0');
```

주문 테이블 데이터입력

```
insert into orders(order_id, prod_id, price, quantity) values('R101', 'P109', 520000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R102', 'P106', 2500000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R102', 'P112', 150000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R103', 'P107', 300000, 2);
insert into orders(order_id, prod_id, price, quantity) values('R104', 'P109', 520000, 2);
insert into orders(order_id, prod_id, price, quantity) values('R104', 'P107', 300000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R104', 'P111', 120000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R105', 'P110', 100000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R106', 'P112', 150000, 2);
insert into orders(order_id, prod_id, price, quantity) values('R107', 'P112', 150000, 4);
insert into orders(order_id, prod_id, price, quantity) values('R108', 'P104', 1500000, 2);
insert into orders(order_id, prod_id, price, quantity) values('R109', 'P103', 3000000, 1);
insert into orders(order_id, prod_id, price, quantity) values('R110', 'P109', 520000, 1);
```

데이터베이스 조회

1) 회사위치가 '서울'인 업체들의 업체코드, 업체명, 회사위치, 전화번호를 출력하시오.

2) 제조원이 '삼성'인 상품들의 상품코드, 상품명, 업체명, 판매가격을 출력하시오.

3) 판매가격이 200000 에서 500000사이인 상품들의 상품코드, 상품명, 판매가격을 출력하시오.

4) 제조원이 '삼성'이거나 '삼보'인 상품들의 상품코드, 상품명, 제조원을 출력하시오.

5) '제로드 컴퓨터' 업체에서 납품하는 상품들의 상품코드, 상품명, 판매가격을 출력하시오.

6) '센스 720' 상품을 납품하는 업체의 업체코드, 업체명, 담당자, 전화번호를 출력하시오.

7) '김수정'이 주문한 상품의 상품코드, 상품 구매가격, 상품 구매수량을 출력하시오.

8) 'P112'상품을 주문한 사람들의 주문자명, 주문일, 주문자 주소를 출력하시오.

9) '서울'에서 구매한 상품들의 상품코드, 상품 구매가격, 상품 구매수량, 주문일을 출력하시오.

10) 상품구매수량이 2개 이상인 상품들의 상품코드, 상품명, 판매가격을 출력하시오.

11) '삼성완전평면 17인치' 상품을 주문한 주문자명, 주문일, 상품 구매가격을 출력하시오.

12) '김연호'가 주문한 상품들의 상품코드, 상품명, 주문일, 상품 구매가격을 출력하시오.

13) 상태가 '1'인 주문 상품들의 상품코드, 상품명, 상품 구매수량을 출력하시오.

14) '2003-04-23'에 주문한 상품들의 상품코드, 상품명, 주문자명을 출력하시오.

15) '센스 720'을 주문한 주문자명, 주문자 우편번호, 주문자 주소, 주문자 이메일을 출력하시오.

16) '넷나라' 컴퓨터에서 납품한 상품들을 주문한 주문자의 주문자명, 상품명, 주문일을 출력하시오.

17) '김수정'이 주문한 상품의 주문합계를 출력하시오.

18) 상품별 상품코드, 상품명, 판매액합계를 출력하시오.

19) 날짜별 판매액합계를 출력하시오.

20) 날짜별, 상품별, 날짜, 상품코드, 상품명, 판매액합계를 출력하시오.

21) 주문번호, 상품코드, 상품명, 상품구매가격, 상품 구매수량, 총 구매가격, 구매자명을 출력하시오.

22) 업체코드가 'M101'인 업체의 담당자명을 '김병렬'로 수정하시오.

23) 제조회사가 '삼성'인 모든 제품의 가격을 5%인하하시오.

24) 주문자 우편번호가 '110'으로 시작하는 것을 '113'으로 수정하시오.

25) 업체 테이블에 업체코드 'M105'을 삽입하시오. (만약 에러가 나면 'M106'을 삽입하시오.)

26) 업체 테이블에서 업체코드가 'M105'를 삭제하시오. (만약 에러가 나면 'M106'을 삭제하시오.)

비디오관리 데이터베이스

- **영화정보(movies) 테이블의 구조**

열이름	데이터형	정보내용
mcode	char(5)	영화코드 (PK)
mname	varchar(100)	영화제목
actor	varchar(30)	주연
director	varchar(30)	감독
odate	date	개봉일자
genre	varchar(30)	장르
grade	int	등급

- **비디오정보(videos) 테이블의 구조**

열이름	데이터형	정보내용
vcode	char(5)	비디오코드 (PK)
pdate	date	구입일
price	int	구입금액
lstate	char(1)	대여상태 (0:대여불가능 1:대여가능)
vstate	char(1)	비디오상태 (0:불량 1:양호)
mcode	char(5)	영화코드 (FK)

- **회원정보(users) 테이블의 구조**

열이름	데이터형	정보내용
ucode	char(5)	회원코드 (PK)
uname	varchar(15)	회원이름
birthday	date	생년월일
tel	varchar(15)	전화번호
address	varchar(100)	주소
point	int	포인트

- **대여정보(lend) 테이블의 구조**

열이름	데이터형	정보내용
lcode	char10)	대여코드 (PK)
vcode	char(5)	비디오코드 (FK)
ucode	char(5)	회원코드 (FK)
ldate	date	대여일자
lprice	int	대여금액
ddate	date	반납예정일
rdate	date	반납일

- **영화정보(movies) 테이블의 데이터**

mcode	mname	actor	director	odate	genre	grade
M0001	반지의 제왕	일라이저 우드	피터 잭슨	2002/01/01	판타지	12
M0002	너는 내 운명	전도연	박진표	2005/09/23	드라마	19
M0003	스파이더맨3	토비 맥과이어	샘 레이미	2007/05/01	액션	12
M0004	그놈 목소리	설경구	박진표	2007/07/01	스릴러	12
M0005	짱구는 못 말려(극장판)	짱구	하시모토 모사카즈	2014/04/03	애니메이션	0

- **비디오정보(videos) 테이블의 데이터**

vcode	pdate	price	lstate	vstate	mcode
V0001	2002/03/01	5000	0	1	M0001
V0002	2002/03/01	5000	0	1	M0001
V0003	2005/11/23	4000	0	0	M0002
V0004	2005/11/25	4000	1	1	M0002
V0005	2007/07/01	5000	0	1	M0003
V0006	2007/07/01	5000	0	1	M0003
V0007	2007/09/01	4000	1	0	M0004
V0008	2007/09/01	4000	0	1	M0004
V0009	2014/06/03	3000	0	1	M0005
V0010	2014/06/03	3000	1	1	M0005

- **회원정보(users) 테이블의 데이터**

ucode	uname	birthday	tel	address	point
U0001	김광석	1969/05/08	010-2230-6780	인천광역시 부평구 계산동 133번지	500
U0002	김동기	1971/11/23	010-9600-8288	인천광역시 서구 청라2동 130번지	1450
U0003	배은진	1980/10/05	010-9626-7020	인천광역시 서구 신현동 561번지	500
U0004	김경아	1979/02/01	010-8290-0009	인천광역시 서구 청라1동 131번지	2300
U0005	조재일	1973/04/05	010-7200-6022	인천광역시 서구 신현동 562번지	1900

- **대여정보(lend) 테이블의 데이터**

lcode	vcode	ucode	ldate	lprice	ddate	rdate
L0001	V0004	U0005	2014/08/01	3000	2014/08/08	2014/08/05
L0002	V0008	U0002	2014/08/01	3000	2014/08/08	2014/08/05
L0003	V0009	U0002	2014/08/02	2500	2014/08/09	2014/08/07
L0004	V0002	U0004	2014/08/15	3500	2014/08/22	
L0005	V0006	U0005	2014/08/15	3500	2014/08/22	2014/08/20
L0006	V0004	U0001	2014/08/15	3000	2014/08/22	2014/08/20
L0007	V0009	U0003	2014/08/17	3500	2014/08/24	
L0008	V0010	U0005	2014/08/17	2000	2014/08/24	2014/08/20
L0009	V0003	U0002	2014/08/18	3500	2014/08/25	
L0010	V0006	U0004	2014/08/18	2500	2014/08/25	
L0011	V0001	U0001	2014/08/18	3500	2014/08/25	
L0012	V0005	U0005	2014/08/19	3500	2014/08/26	2014/08/20
L0013	V0008	U0003	2014/08/20	3000	2014/08/27	
L0014	V0002	U0002	2014/08/20	3500	2014/08/27	2014/08/20
L0015	V0005	U0001	2014/08/21	3500	2014/08/28	

video user 생성

```
create user video  identified by pass;
grant connect, resource, dba to video;
```

영화정보(movies) 테이블의 생성

```
create table movies(
    mcode char(5) not null,
    mname varchar(100) not null,
    actor varchar(30),
    director varchar(30),
    odate date,
    genre varchar(30),
    grade int,
    primary key (mcode)
);
```

비디오정보(videos) 테이블의 생성

```
create table videos(
    vcode char(5) not null,
    pdate date,
    price int,
    lstate char(1),
    vstate char(1),
    mcode char(5),
    primary key (vcode);
    foreign key (mcode) references movies(mcode)
);
```

회원정보(users) 테이블의 구조

```
create table users(
    ucode char(5) not null,
    uname varchar(15),
    birthday date,
    tel varchar(15),
    address varchar(100),
    point int,
    primary key(ucode)
);
```

대여정보(lend) 테이블의 생성

```
create table lend(
    lcode  char(10) not null,
    vcode char(5),
    ucode char(5),
    ldate date,
    lprice int,
    ddate date,
    rdate date,
    primary key (lcode),
    foreign key (vcode) references videos(vcode),
    foreign key (ucode) references users(ucode)
);
```

영화정보(movies) 테이블의 데이터 입력

```
insert into movies(mcode, mname, actor, director, odate, genre, grade)
    values('M0001','반지의 제왕','일라이저 우드', '피터 잭슨', '2002/01/01', '판타지', 12);
insert into movies(mcode, mname, actor, director, odate, genre, grade)
    values('M0002','너는 내 운명','전도연', '박진표', '2005/09/23', '드라마', 19);
insert into movies(mcode, mname, actor, director, odate, genre, grade)
    values('M0003','스파이더맨3','토비 맥과이어', '샘 레이미', '2007/05/01', '액션', 12);
insert into movies(mcode, mname, actor, director, odate, genre, grade)
    values('M0004','그놈 목소리','설경구', '박진표', '2007/07/01', '스릴러', 12);
insert into movies(mcode, mname, actor, director, odate, genre, grade)
    values('M0005','짱구는 못 말려(극장판)','짱구', '하시모토 모사카즈', '2014/04/03', '애니메이션', 0);
```

비디오정보(videos) 테이블의 데이터 입력

```
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0001','2002/03/01','5000', '0', '1', 'M0001');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0002','2002/03/01','5000', '0', '1', 'M0001');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0003','2005/11/23','4000', '0', '0', 'M0002');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0004','2005/11/25','4000', '1', '1', 'M0002');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0005','2007/07/01','5000', '0', '1', 'M0003');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0006','2007/07/01','5000', '0', '1', 'M0003');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0007','2007/09/01','4000', '1', '0', 'M0004');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0008','2007/09/01','4000', '0', '1', 'M0004');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0009','2014/06/03','3000', '0', '1', 'M0005');
insert into videos(vcode, pdate, price, lstate, vstate, mcode) values('V0010','2014/06/03','3000', '1', '1', 'M0005');
```

회원정보(users) 테이블의 데이터 입력

```
insert into users(ucode, uname, birthday, tel, address, point)
    values('U0001','김광석','1969/05/08', '010-2230-6780', '인천광역시 부평구 계산동 133번지', 500);
insert into users(ucode, uname, birthday, tel, address, point)
    values('U0002','김동기','1971/11/23', '010-9600-8288', '인천광역시 서구 청라2동 130번지', 1450);
insert into users(ucode, uname, birthday, tel, address, point)
    values('U0003','배은진','1980/10/05', '010-9626-7020', '인천광역시 서구 신현동 561번지', 500);
insert into users(ucode, uname, birthday, tel, address, point)
    values('U0004','김경아','1979/02/01', '010-8290-0009', '인천광역시 서구 청라1동 131번지', 2300);
insert into users(ucode, uname, birthday, tel, address, point)
    values('U0005','조재일','1973/04/05', '010-7200-6022', '인천광역시 서구 신현동 562번지', 1900);
```

대여정보(lend) 테이블의 데이터 입력

```
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0001','V0004','U0005','2014/08/01',3000,'2014/08/08', '2014/08/05');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0002','V0008','U0002', '2014/08/01', 3000, '2014/08/08', '2014/08/05');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0003','V0009','U0002', '2014/08/02', 2500, '2014/08/09', '2014/08/07');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0004','V0002','U0004', '2014/08/15', 3500, '2014/08/22', '');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0005','V0006','U0005', '2014/08/15', 3500, '2014/08/22', '2014/08/20');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0006','V0004','U0001', '2014/08/15', 3000, '2014/08/22', '2014/08/20');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0007','V0009','U0003', '2014/08/17', 3500, '2014/08/24', '');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0008','V0010','U0005', '2014/08/17', 2000, '2014/08/24', '2014/08/20');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0009','V0003','U0002', '2014/08/18', 3500, '2014/08/25', '');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0010','V0006','U0004', '2014/08/18', 2500, '2014/08/25', '');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0011','V0001','U0001', '2014/08/18', 3500, '2014/08/25', '');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0012','V0005','U0005', '2014/08/19', 3500, '2014/08/26', '2014/08/20');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0013','V0008','U0003', '2014/08/20', 3000, '2014/08/27', '');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0014','V0002','U0002', '2014/08/20', 3500, '2014/08/27', '2014/08/20');
insert into lend(lcode,vcode,ucode,ldate,lprice,ddate,rdate)
    values('L0015','V0005','U0001', '2014/08/21', 3500, '2014/08/28', '');
```

캠핑장 예약 데이터베이스

- 캠핑장 정보 테이블 (tbl_camp_list)

열이름	데이터형	정보내용
camp_id	char(5)	캠핑장 코드
camp_name	varchar(100)	캠핑장 이름
camp_image	varchar(200)	캠핑장 이미지
camp_address	varchar(200)	캠핑장 주소
camp_tel	varchar(20)	캠핑장 전화번호

- 캠핑장 스타일 테이블 (tbl_camp_style)

열이름	데이터형	정보내용
camp_id	char(5)	캠핑장 코드
style_no	char(2)	스타일 코드
style_qty	int	스타일 갯수
style_price	varchar(20)	스타일 가격

- 캠핑장 스타일 이름 테이블 (tbl_style_name)

열이름	데이터형	정보내용
style_no	char(2)	스타일 코드
style_name	varchar(50)	스타일 이름
style_icon	varchar(200)	스타일 아이콘

- 캠핑장 시설 테이블 (tbl_camp_facility)

열이름	데이터형	정보내용
camp_id	char(5)	캠핑장 코드
facility_no	char(2)	시설 코드

- 캠핑장 시설 이름 테이블 (tbl_facility_name)

열이름	데이터형	정보내용
facility_no	char(5)	캠핑장 코드
facility_name	char(2)	시설 코드
facility_icon	varchar(200)	시설 이미지

- 캠핑장 리뷰 테이블 (tbl_camp_review)

열이름	데이터형	정보내용
review_id	int	리뷰 코드
camp_id	char(5)	캠프 코드
review_content	varchar(1000)	리뷰 내용
review_date	date	리뷰 작성날짜
review_writer	varchar(20)	리뷰 작성자

- **캠핑장 예약 테이블 (tbl_camp_reserve)**

열이름	데이터형	정보내용
reser_no	int	예약번호
camp_id	char(5)	캠프 코드
style_no	char(2)	캠프장 스타일 코드
reser_checkin	date	체크인 날짜
reser_checkout	date	체크아웃 날짜
reser_name	varchar(20)	예약자 이름
reser_tel	varchar(20)	예약자 전화번호
reser_status	char(1)	0:미결제, 1:카드결제, 2:현금결제

- **캠핑장별 캠핑장 코드, 이름, 주소, 이미지, 스타일 총 개수, 최저가 최고가를 출력하는 view생성 (view_camp_list)**

```
create view view_camp_list as
    (select c.camp_id, camp_name, camp_address, camp_image,
            sum(style_qty) sum_cnt, min(style_price) min_price, max(style_price) max_price
    from tbl_camp_list c left join tbl_camp_style s
    on c.camp_id=s.camp_id
    group by c.camp_id
);
```

- **특정 지역과 특정 기간을 입력하면 view_camp_list 테이블의 모든 필드와 특정 기간에 예약된 예약건수를 출력하는 sql문 작성**

```
select c.*, reser_cnt from
    (select * from view_camp_sql where camp_address like '%전남%') c
left join
    (select camp_id, count(*) reser_cnt
    from tbl_camp_reserve
    where (reser_checkin between '2021-11-01' and '2021-12-31')
    or (reser_checkout between  '2021-11-01' and '2021-12-31')
    or (reser_checkin < '2021-11-01' and reser_checkout > '2021-12-31')
    group by camp_id) r
on c.camp_id= r.camp_id;
```

- **특정지역, 특정기간, 시설물, 스타일로 캠핑장 목록을 검색하는 sql문 작성**

```
select * from
    (select c.*, reserve_cnt
    from
        (select * from view_camp_sql where camp_addr like '%%') c  /*지역검색*/
    left join
        (select camp_id, count(*) reserve_cnt
        from tbl_camp_reserve
        where (reser_checkin between '2021-11-01' and '2021-12-31')  /*특정 기간검색*/
        or (reser_checkout between '2021-11-01' and '2021-12-31')
        or (reser_checkin < '2021-11-01' and reser_checkout > '2021-12-31')
        group by camp_id) r
    on c.camp_id= r.camp_id) tbl
where camp_id in  /*시설검색*/
    (select camp_id
    from tbl_camp_facility
    where facility_no in ('1', '2')
    group by camp_id
    having count(*) = 2)
and camp_id in  /*스타일검색*/
    (select camp_id from tbl_camp_style where style_no = '1')
order by camp_id;
```

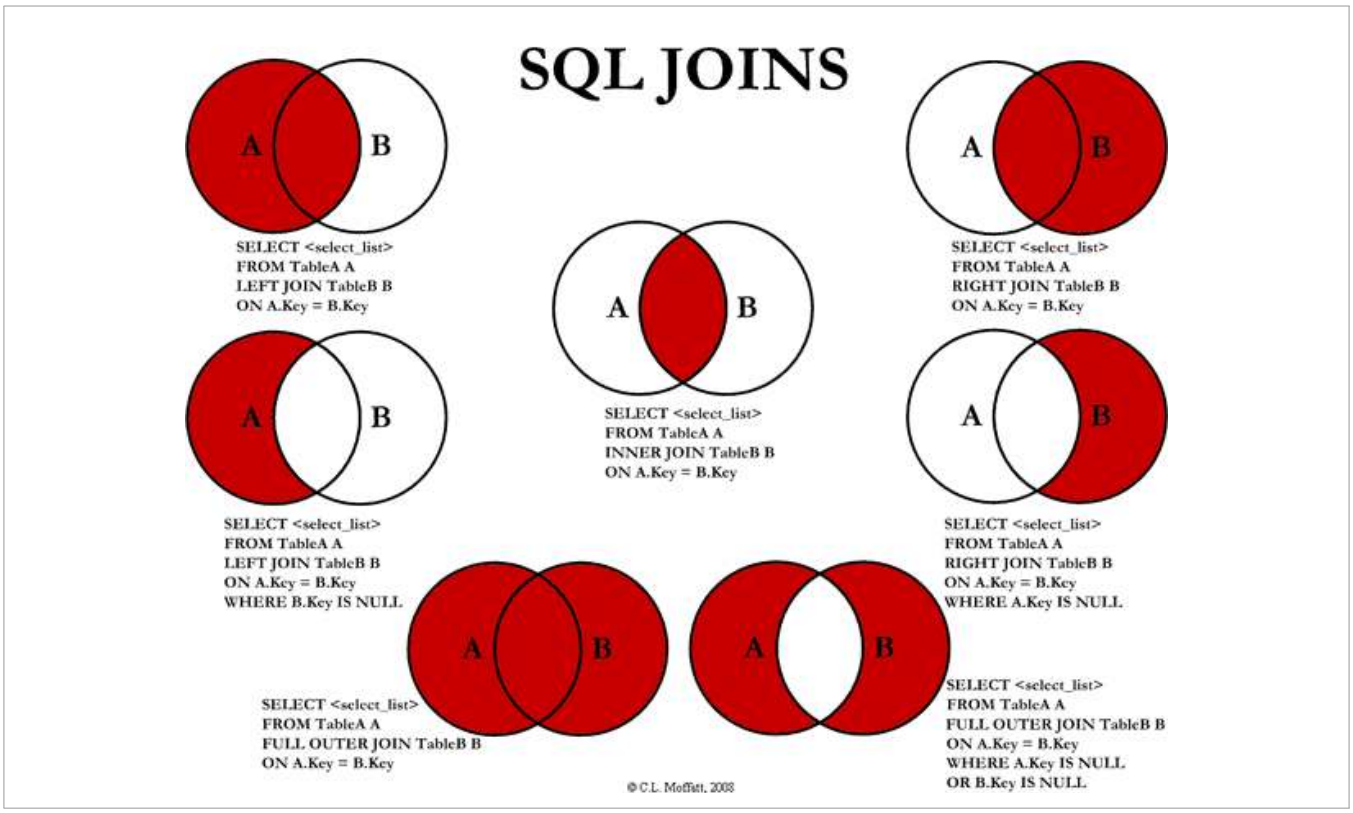
- MySql 외부 컴퓨터에서 접근 허용

```
1) root/1234 접속

2) 내부 ip로 외부 컴퓨터에서 접속 설정
-GRANT ALL PRIVILEGES ON webdb.* TO 'web'@'192.168.%' IDENTIFIED BY 'pass'; //대역 ip 허용
-GRANT ALL PRIVILEGES ON webdb.* TO 'web'@'%' IDENTIFIED BY 'pass'; //모든 ip 허용
-select host, user from mysql.user; //ip 허용 확인
-delete from mysql.user where host='192.168.%' and user='web'; //대역 ip 허용 삭제

3) 외부 ip로 접근할 경우 방화벽 설정 / iptime 포트 포워드 설정
```

- SQL Joins



jon SQL문 연습

```
insert into professors(pcode, pname, dept, hiredate) values('100', '심청이', '전산', now());
insert into students(scode, sname,dept, birthday) values('92010001','홍길동','전산', '1969/05/08');
insert into courses(lcode, lname) value('1011', '자바스크립트');

select pcode, pname, scode, sname, advisor from professors inner join students on  pcode=advisor order by pcode;

select pcode, pname, scode, sname, advisor from professors left join students on  pcode=advisor;

select pcode, pname, scode, sname, advisor from professors left join students on  pcode=advisor where scode is null;

select pcode, pname, scode, sname, advisor from professors right join students on  pcode=advisor;

select pcode, pname, scode, sname, advisor from professors right join students on  pcode=advisor where pcode is null;

select pcode, pname, scode, sname, advisor from professors full outer join students on  pcode=advisor;

/* ----------Mysql인 경우---------------- */
select pcode, pname, scode, sname
from professors left join students on pcode=advisor
union
select pcode, pname, scode, sname
from professors right join students on pcode=advisor
order by pcode;
```

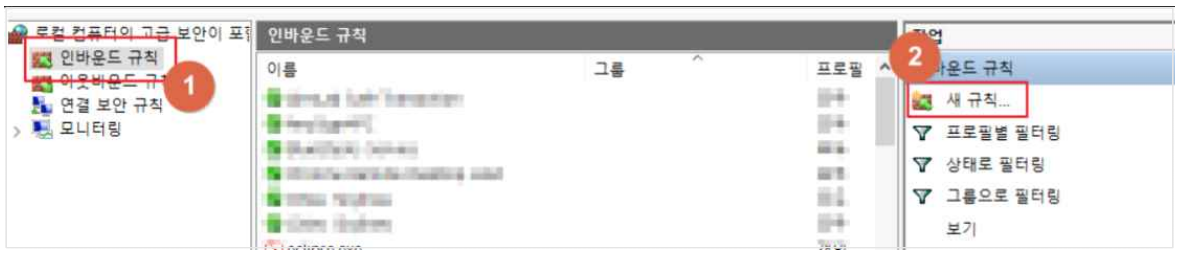
Oracle 서버 외부에서 접속하기 (+IPTIME 설정)

- **고정 아이피 설정**

- **방화벽 설정**

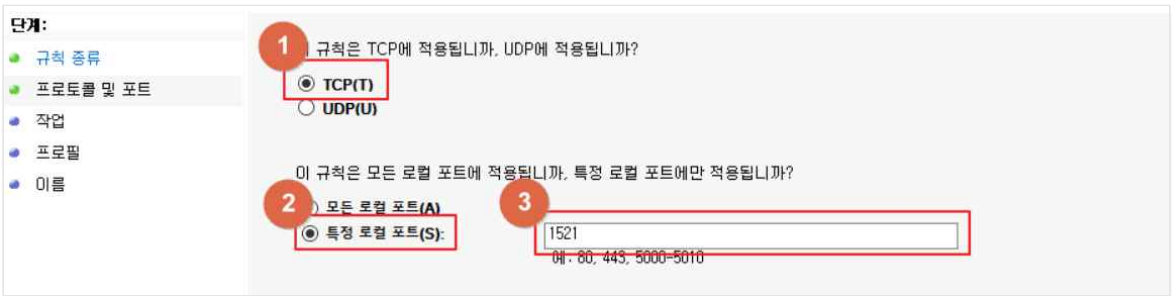
1. **[제어판]-[모든 제어판 항목]-[Windows Defender 방화벽]-[고급설정]**

2. **[인바운드 규칙]-[새 규칙]**



3. **[포트 선택] - [다음]**

4. **[TCP] -[특정 로컬 포트] - [다음]**

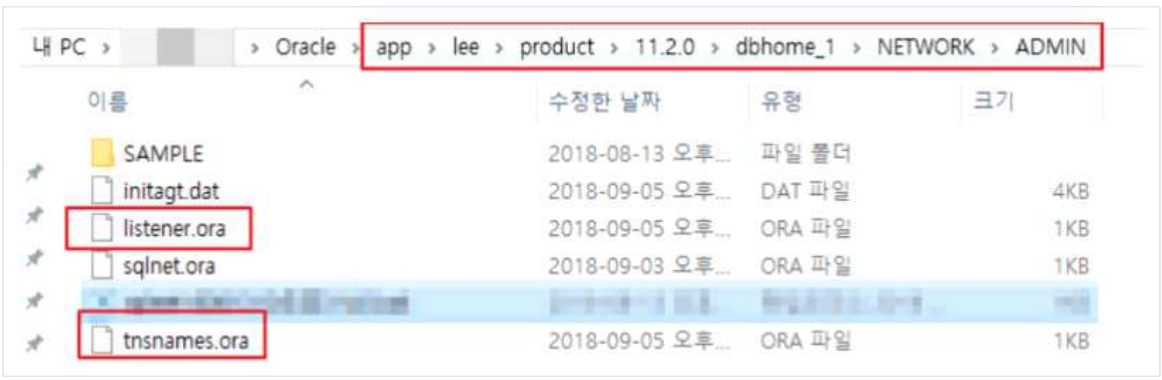


5. [연결 허용 체크] - [다음] - [도메인, 개인, 공용 체크] - [다음] -[이름입력] - [마침]

- 서버 PC의 listener.ora와 tnsnames.ora 설정

1. [바탕화면] - [내PC] -[속성] - [컴퓨터 이름]

2. Oracle이 설치된 위치로 이동 [설치장소]-[app]-[oracle]-[product]-[11.2.0]-[server]-[newtwork]-[ADMIN]



3. listener.org 파일 HOST= localhost라 되어 있는 것을 컴퓨터 이름으로 수정한 후 저장

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = Lee-PC)(PORT = 1521))
    )
  )
```

4. tnsnames.ora 파일도 마찬가지로 HOST=localhost라 쓰여져 있는 것을 컴퓨터 이름으로 수정한 후 저장

```
# tnsnames.ora Network Configuration File: [redacted]product\11.2.0\dbhome_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

LISTENER_MYORACLE =
  (ADDRESS = (PROTOCOL = TCP)(HOST = Lee-PC)(PORT = 1521))
```

```
MYORACLE =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = Lee-PC)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = myoracle)
    )
  )
```

- **iptime 설정**

1. [크롬]-[192.168.0.1]-[로그인]-[관리도구]

2. [고급설정]-[NAT/라우터 설정]-[포트포워드 설정]

3. 포트 포워드 설정하기

- 규칙이름 : Oracle
- 내부 IP 주소: 현재 접속된 IP 주소 체크박스 체크
- 프로토콜: TCP
- 외부포트: 1521 ~ 1521
- 내부코트: 1521 ~ 1521

4. 상단의 [저장] 버튼 클릭

- **내 IP 조회 (네이버에서 내IP 검색)**