데이터베이스
(Oracle)

1. 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.
```

```
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;
```

2. 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	김영명	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 관련 SQL

```
desc 테이블명  /*테이블구조*/
select table_name from  user_tables;  /*사용자 테이블 검색*/
drop table 테이블명  /*테이블 삭제*/
```

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('C301','95454003','98/03/05',83);
insert into enrollments(lcode, scode, edate, grade) values('E221','95454003','98/03/05',75);
```

- **데이터베이스 조회(1)**

교수 테이블의 모든 데이터를 검색하시오.

```
select * from professors;
```

교수 테이블에서 모든 교수의 이름, 소속학과, 직급을 검색하시오.

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

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

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

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

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

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

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

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

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

수강신청 테이블에서 성적이 80점 이상인 과목번호, 학생번호를 검색하시오.

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

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

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

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';
```

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;
```

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

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

수강 신청테이블에서 성적이 입력된 과목들의 강좌번호, 학생번호를 모두 검색하시오.

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

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

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

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');
```

'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)**

학생들의 학과, 학생이름, 지도교수이름을 검색하시오.

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

강좌번호, 강좌명, 교수이름을 검색하시오.

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

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

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

학번, 학생들이 수강신청 한 강좌번호, 강좌명, 성적을 검색하시오.

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

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

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

'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';
```

'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';
```

'자료구조'를 강의하는 교수의 학과명, 교수 명을 검색하시오.

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

'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';
```

'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';
```

'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';
```

'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';
```

'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';
```

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;
```

- **데이터베이스 조회(3)**

교수들의 총 급여액과 평균 급여액을 구하시오.

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

전산과 교수들의 총 급여액과 평균 급여액을 구하시오.

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

수강신청 한 과목들 중에서 최고 점수와 최저 점수를 구하시오.

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

전산과 학생들은 모두 몇 명인지 구하시오.

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

수강신청 한 학생들은 모두 몇 명인지 구하시오.

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

각 학과별 학생들의 수를 구하시오.

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

교수들을 소속학과별, 직급별로 분류하여 각 직급별 평균 급여액수를 구하시오.

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

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;
```

각 학생들에 대해 수강신청 과목들의 평균 점수를 구하시오.

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

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;
```

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

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

강좌별 평균점수가 80점 이상인 강좌들의 강좌번호와 평균점수를 출력하시오.

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

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

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

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;
```

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)**

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

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

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');
```

'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');
```

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);
```

'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');
```

'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');
```

'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'));
```

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);
```

'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')
```

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

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

'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'));
```

'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'));
```

'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');
```

전체 학생의 30% 이상이 수강신청한 강좌의 번호를 검색하시오.

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

- **데이터베이스 갱신(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');
```

노진순의 수강신청 내용을 수강신청(Enrollments)테이블에 삽입하시오.

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

'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';
```

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;
```

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';
```

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

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

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

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

수강신청 데이터(oldenrollments)를 모두 삭제하시오.

```
delete from oldenrollments;
```

학생테이블에서 전산과 3학년 데이터를 4학년으로 변경하시오.

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

'오문환' 교수의 직급을 '조교수'에서 '부교수'로 변경하시오.

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

'건축과' 학생이 신청한 모든 수강신청 데이터를 삭제하시오.

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

'전산'과 교수들의 급여를 10% 증가 시키시오.

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

모든 교수들의 급여를 10% 증가 시키시오.

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

'전자'과 학생들이 신청한 수강신청 데이터를 모두 삭제하시오.

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

'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');
```

- **데이터베이스 갱신(2)**

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

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

'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');
```

'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');
```

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

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

'논리회로'를 강의하는 교수의 급여를 100000원 인상하시오.

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

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);
```

서연우' 학생의 모든 과목 점수를 5점씩 감소 시키시오.

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

'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);
```

'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');
```

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);
```

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);
```

'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);
```

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);
```

'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);
```

'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);
```

- **View (가상테이블)**

뷰 생성 및 삭제

```
create view 뷰명 as (select문);  /* 뷰 생성 */
select * from user_views;  /*정의된 뷰 확인*/
drop view 뷰명;  /*뷰 삭제*/
```

학생테이블에 지도교수명을 추가하는 view를 생성하시오.

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

강좌테이블에 담당교수명을 추가하는 view를 생성하시오.

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

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);
```

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);
```

'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);
```

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);
```

'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);
```

'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);
```

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);
```

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 */
```

'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;
```

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;
```

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;
```

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;
```

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();
```

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;
```

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;
```

교수 테이블에서 특정학과 교수들을 출력하고 인원수를 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;
```

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;
```

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;
```

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;
```

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

```
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;
}
```

Controller.java

```
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);
```

insert.jsp

```
<body>
    <input type="text" id="txtScode">
    <input type="text" id="txtSname">
    <button id="btnSave">저장</button>
</body>
<script>
    $("#btnSave").on("click", function(){
        $.ajax({
            type:"post",
            url:"/insert",
            dataType:"json",
            data:{scode: $("#txtScode").val(), sname: $("#txtSname").val()},
            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';
```

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

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

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

```
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;
```

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

```
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;
```

- **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;
```

- 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;
```

- 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;
```

- 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;
```

---

3. 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
);
```

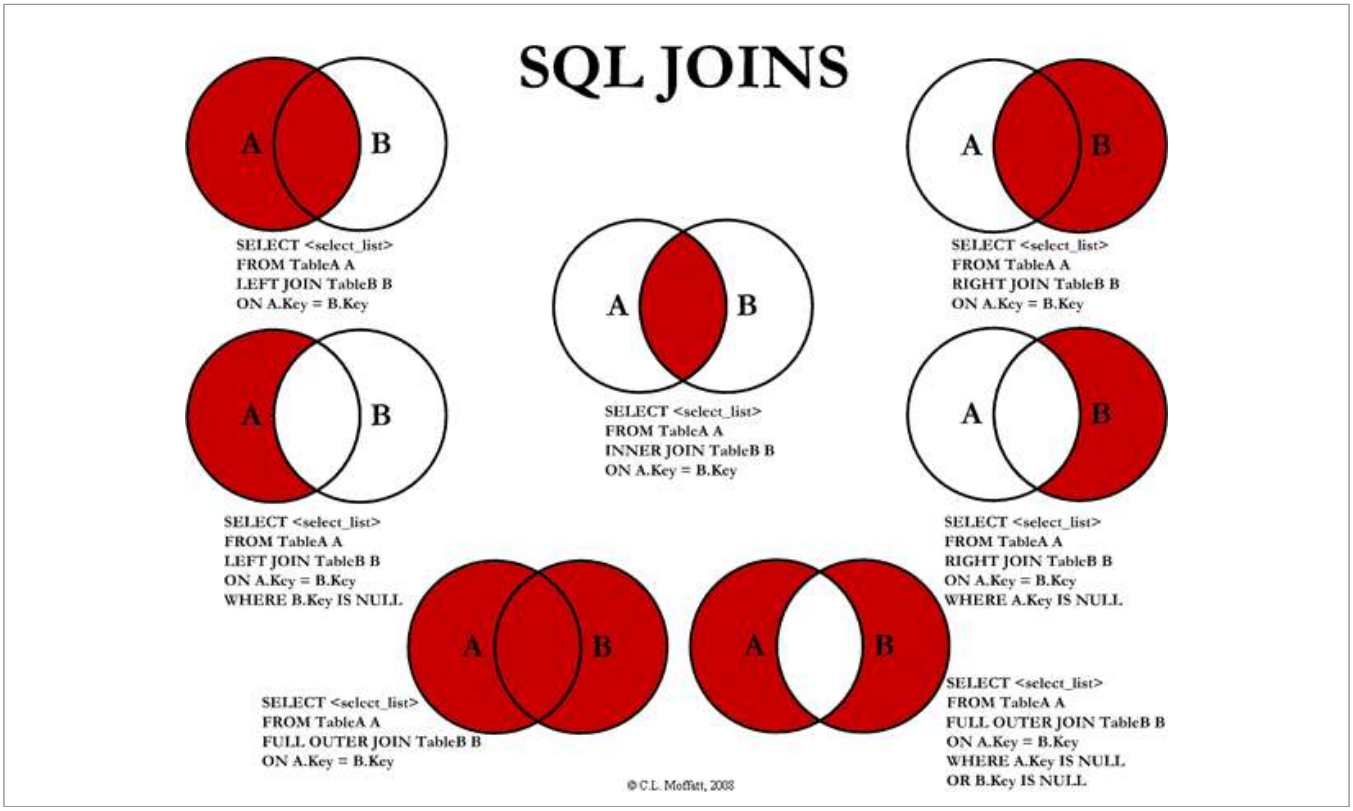
특정 지역과 특정 기간을 입력하면 view_camp_list 테이블의 모든 필드와 특정 기간에 예약된 예약건수를 출력하는 sql문 작성

```
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;
```

- 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;

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;

/*----------Oracle(outer join)--------------*/
select pcode, pname, scode, sname, advisor
from professors full outer join students
on pcode=advisor;

/*----------Mysql(outer join)---------------*/
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
```

4. 데이터베이스 서버 외부에서 접속하기 (+IPTIME 설정)

- **고정 아이피 설정**

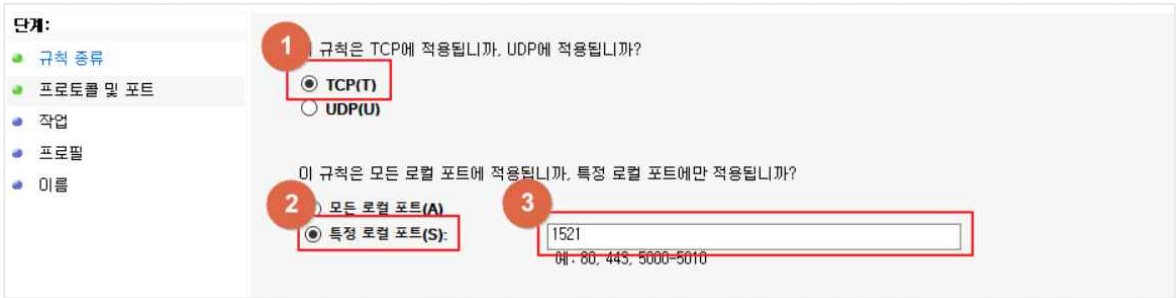
- **내부포트 열기 (방화벽 해제)**

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

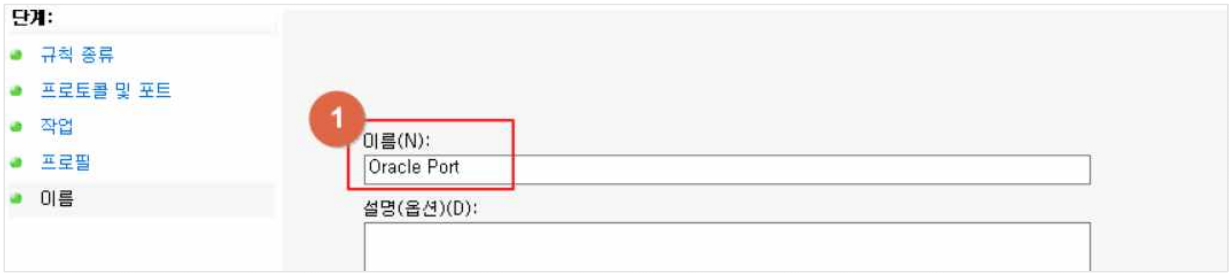
2. [인바운드 규칙]-[새 규칙]

3. [포트 선택] - [다음]

4. [TCP] -[특정 로컬 포트] - [다음]



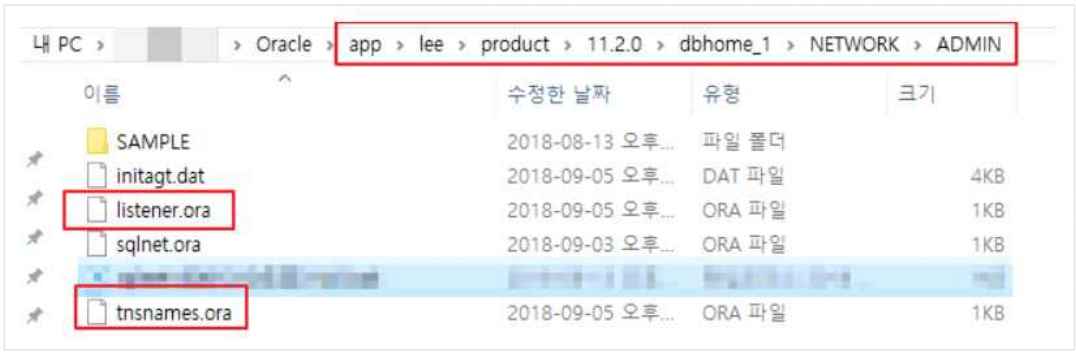
5. [연결 허용 체크] - [다음] - [도메인, 개인, 공용 체크] - [다음] -[이름입력] - [마침]



- **Oracle 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)
    )
  )
```

- **MySql 외부 컴퓨터에서 접근 허용**

localhost에만 접속 허용

```
GRANT ALL PRIVILEGES ON webdb.* TO web@'localhost' identified by 'pass';
```

192.169.x.x로 시작되는 모든 IP에서 접속 허용

```
GRANT ALL PRIVILEGES ON webdb.* TO web@'192.168.%' identified by 'pass';
```

외부 컴퓨터에서 접속 허용

```
GRANT ALL PRIVILEGES ON webdb.* TO web@'%' identified by 'pass';
```

접속허용 확인

```
select host, user from mysql.user;
```

접속허용 제거

```
delete from mysql.user where host='192.168.%' and user='web';
```

- **외부포트 허용하기 (포트 포워딩:iptime 설정)**

1. [크롬]-[192.168.0.1]-[로그인]-[관리도구] [cmd]-[ipconfig/all] 기본게이트

2. [고급설정]-[NAT/라우터 설정]-[포트포워드 설정]

3. 포트 포워드 설정하기