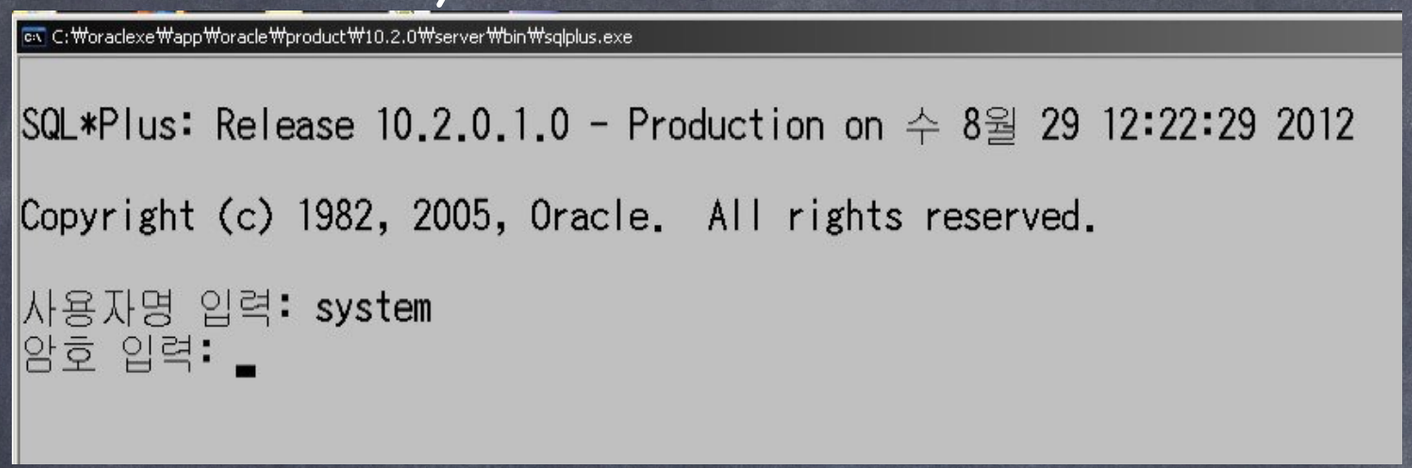
Oracle Database
설치 및 활용

설치 및 사용자 계정 만들기

1. 오라클 설치 http://www.oracle.com 에서 다운 설치

설치시 암호가 system 계정의 암호로 설정된다.



2. 연습용 계정 만들기

--계정은 system 계정으로 들어가서 만든다
--커맨트 창에서 sqlplus system/oracle

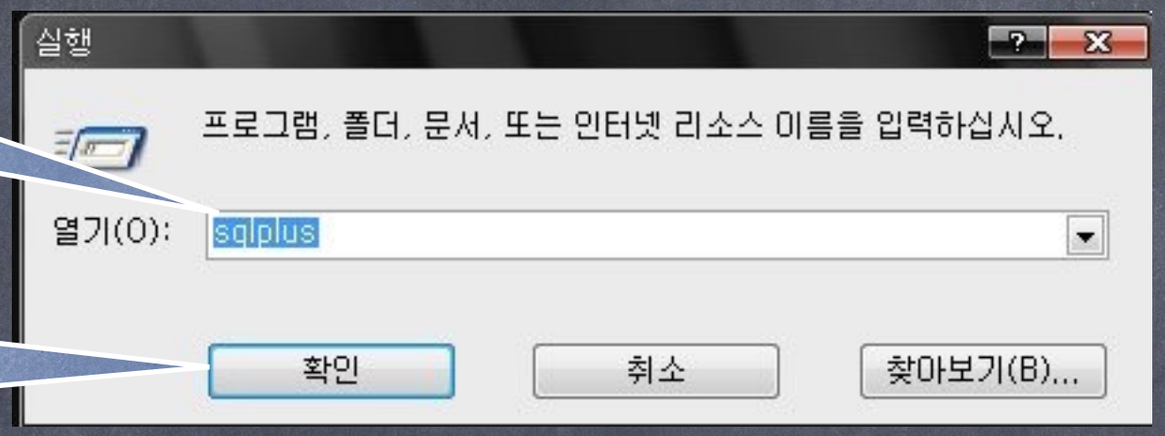
--사용자 생성하기
SQL> CREATE USER 계정 IDENTIFIED BY 비밀번호 ;

--권한 주기
SQL> GRANT RESOURCE,CONNECT TO scott ;

관리자(system) 계정으로 접속하기

1. 실행 => sqlplus

2. click !




```
SQL*Plus: Release 10.2.0.1.0 - Production on 수 4월 3 10:35:35 2013

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

사용자명 입력: system
암호 입력:

다음에 접속됨:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

SQL>
```

사용자 계정 만들기

CREATE USER 계정 IDENTIFIED BY 비밀번호

새로운 사용자계정(ID)

비밀번호

```
SQL> CREATE USER scott IDENTIFIED BY tiger;

사용자가 생성되었습니다.

SQL> _
```

권한 부여하기

GRANT 권한의 종류 TO 계정

```
SQL> GRANT RESOURCE, CONNECT TO scott;

권한이 부여되었습니다.

SQL>
```

RESOURCE => 자원
CONNECT => 접속권한

새로 만든 사용자 계정으로 접속하기

```
SQL*Plus: Release 10.2.0.1.0 - Production on 수 4월 3 10:39:47 2013

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

사용자명 입력: scott
암호 입력:

다음에 접속됨:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

SQL>
```

회원정보를 저장할 테이블 만들기

CREATE TABLE 테이블명 (칼럼명 TYPE 제약조건, ...);

테이블명

```
SQL> CREATE TABLE member
  2  (num NUMBER PRIMARY KEY, name VARCHAR2(50), addr VARCHAR2(100));

테이블이 생성되었습니다.

SQL>
```

칼럼명

회원정보를 저장할 테이블 만들기

CREATE TABLE 테이블명 (칼럼명 TYPE 제약조건, ...);

테이블 만들때 사용하는 예약어

```
SQL> CREATE TABLE member
  2  (num NUMBER PRIMARY KEY, name VARCHAR2(50), addr VARCHAR2(100));

테이블이 생성되었습니다.

SQL>
```

칼럼의 데이터 Type

NUMEBER => 숫자 Type

VARCHAR2(50) => 가변 문자열 최대 영문자 50 (한글:25) 글자

VARCHAR2(100) => 가변 문자열 최대 영문자 100 (한글:50) 글자

회원정보를 저장할 테이블 만들기

CREATE TABLE 테이블명 (칼럼명 TYPE 제약조건, ...);

```
SQL> CREATE TABLE member
  2  (num NUMBER PRIMARY KEY, name VARCHAR2(50), addr VARCHAR2(100));

테이블이 생성되었습니다.

SQL>
```

NOT NULL + UNIQUE 조건
1. 반듯이 값을 넣어주어야한다.
2. 중복된 값을 허용하지 않는다.

생성한 테이블의 구조 보기

DESC 테이블명

```
SQL> DESC member
 이름                                      널?       유형
 ----------------------------------------- -------- ----------------

 NUM                                       NOT NULL NUMBER
 NAME                                               VARCHAR2(50)
 ADDR                                               VARCHAR2(100)

SQL> _
```

테이블에 데이터 저장하기

INSERT INTO 테이블명 (칼럼명, ..) VALUES (값1, ..);

```
SQL> INSERT INTO member(num, name, addr) VALUES(1, '김구라', '노량진');

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO member(num, name) VALUES(2, '해골');

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO member VALUES(3, '원숭이', '상도동');

1 개의 행이 만들어졌습니다.

SQL>
```

특정 칼럼에만 값을 넣을수도 있다.

모든 칼럼의 값을 순서대로 넣어줄때는
칼럼명 생략 가능하다.

데이터를 실제로 테이블에 반영하기

수정 내용을 반영하기 위해서는 COMMIT 해야한다.

```
SQL> COMMIT;

커밋이 완료되었습니다.

SQL>
```

테이블에 저장된 데이터 출력해보기

SELECT 칼럼명1, .. FROM 테이블명

보고 싶은 칼럼명을 나열한다.

```
SQL> SELECT num, name, addr FROM member;

       NUM NAME                 ADDR
---------- -------------------- ------------------------------
         1 김구라               노량진
         2 해골
         3 원숭이               상도동

SQL>
```

테이블에 저장된 데이터 출력해보기

보고 싶은 칼럼명을 나열한다.

```
SQL> SELECT num, name FROM member;

       NUM NAME
---------- --------------------
         1 김구라
         2 해골
         3 원숭이

SQL> _
```

테이블에 저장된 데이터 출력해보기

모든 칼럼의 내용을 다 보고 싶을때는 * 로 대치

```
SQL> SELECT * FROM member;

       NUM NAME                 ADDR
---------- -------------------- ------------------------------
         1 김구라               노량진
         2 해골
         3 원숭이               상도동

SQL>
```

테이블에서 특정 ROW 를 삭제

DELETE FROM 테이블명 WHERE 조건절

```
SQL> DELETE FROM member WHERE num=2;

1 행이 삭제되었습니다.

SQL> SELECT * FROM member;

       NUM NAME                 ADDR
---------- -------------------- ------------------------------
         1 김구라               노량진
         3 원숭이               상도동

SQL> COMMIT;

커밋이 완료되었습니다.

SQL> _
```

primary key 값으로 지정된 칼럼을 조건절에서 이용한다.

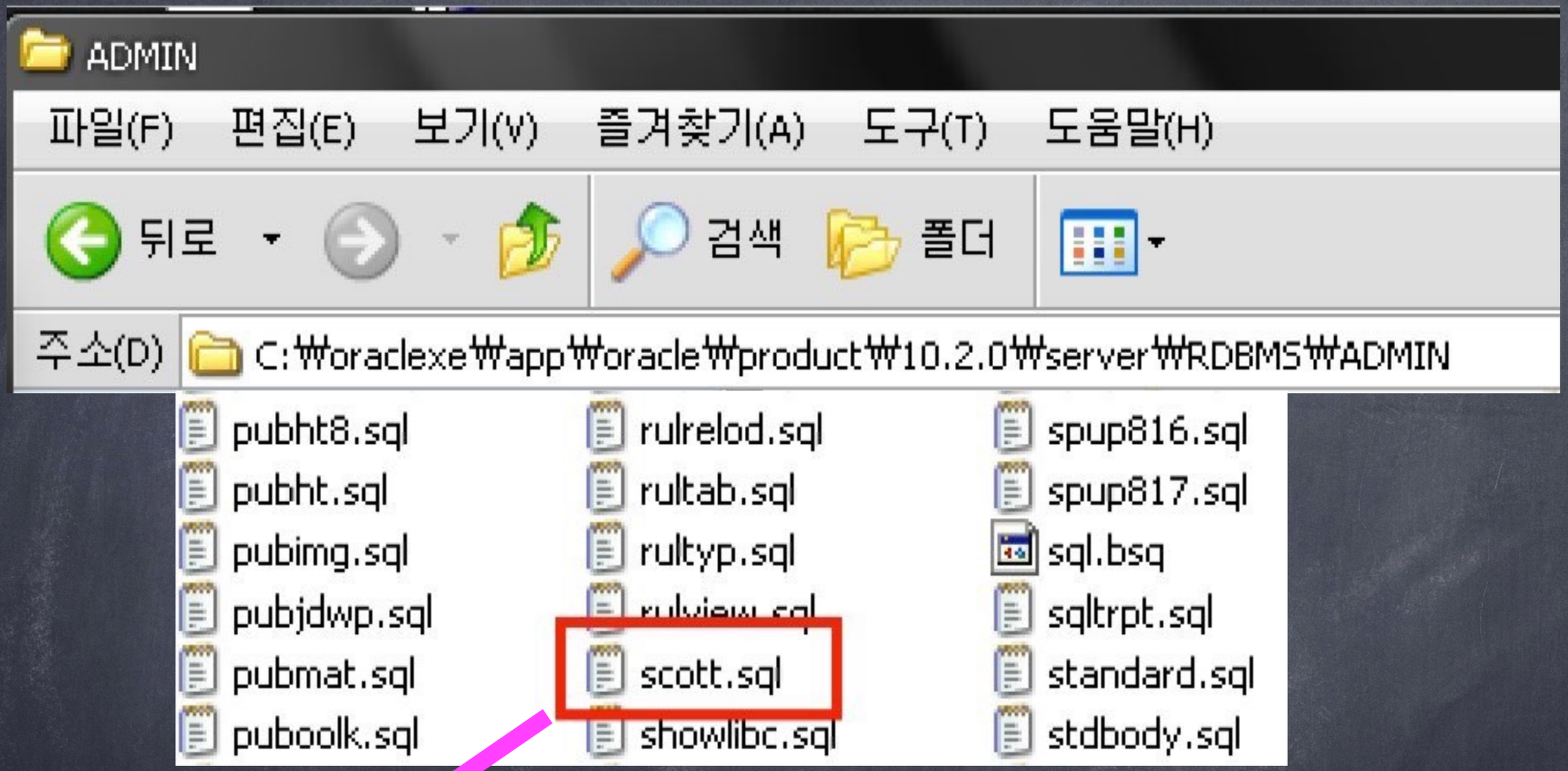
특정 ROW 의 데이터 수정하기

UPDATE 테이블명 SET 칼럼명1=값1, .. WHERE 조건절

```
SQL> UPDATE member SET addr='동물원' WHERE num=3;

1 행이 갱신되었습니다.

SQL> SELECT * FROM member;

       NUM NAME                 ADDR
---------- -------------------- ------------------------------
         1 김구라               노량진
         3 원숭이               동물원

SQL> COMMIT;

커밋이 완료되었습니다.

SQL>
```

primary key 값으로 지정된 칼럼을 조건절에서 이용한다.

연습용 사원(EMP) 테이블 만들기



drag and drop !

```
SQL> @C:₩oraclexe₩app₩oracle₩product₩10.2.0₩server₩RDBMS₩ADMIN₩scott.sql
```

[1] DQL (Data Query Language)

1. SELECT

[기본형식]

```
SELECT 칼럼명1,칼럼명2,....        << 실행순서 >>   .5
FROM 테이블명                      .1
WHERE 조건절                       .2
GROUP BY 칼럼명                    .3
HAVING 조건절                      .4
ORDER BY 칼럼명 [ASC|DESC]  =>오름 차순 혹은 내림차순.     .6
```

(1) SELECT 문 사용하기

- emp 테이블에서 사원번호, 사원이름, 직업을 출력해보세요.

```
SQL> SELECT empno, ename, job FROM emp;
```

- emp 테이블에서 사원번호,급여,부서번호를 출력하세요.
단, 급여가 많은 순서대로 출력

```
SQL>SELECT empno, sal, deptno
FROM emp
ORDER BY  sal DESC ;
```

- emp 테이블 에서 사원번호,급여,입사일을 출력해보세요
단,급여가 적은 순서대로.

```
SQL>SELECT empno, sal, hiredate
FROM emp
ORDER BY sal ASC ;
```

-emp 테이블에서 직업,급여를 출력 해보세요
단,직업명으로 오름차순, 급여로 내림차순 정렬해서

```
SQL>SELECT job, sal
FROM emp
ORDER BY job ASC, sal DESC ;
```

(2) Where 절 사용하기

-급여가 2000 이상인 사원의 사원번호,사원이름,급여 출력하기

```
SQL>SELECT empno, ename, sal
FROM emp
WHERE sal >= 2000 ;
```

-emp 테이블에서 부서번호가 10번인 사원들의 모든 정보를 출력하세요.

```
SQL>SELECT *
FROM emp
WHERE deptno=10;
```

(2) Where 절 사용하기

-emp 테이블에서 입사일이 '81/02/20' 인 사원의 사원번호,이름,입사일을 출력해보세요

```
SQL>SELECT empno, ename, hiredate
FROM emp
WHERE hiredate='81/02/20'
```

-emp 테이블에서 직업이 'SALESMAN' 인 사람들의 이름,직업,급여를 출력해 보세요 ,단 급여가 높은 순서대로

```
SQL>SELECT ename, job, sal
FROM emp
WHERE job='SALESMAN'
ORDER BY sal DESC ;
```

(3) ALIAS 사용하기 (칼럼에 별칭 붙이기)

```
SQL>SELECT empno AS "사원번호" , ename AS "사원이름"
FROM emp ;
```

-AS 와 큰따옴표 " 는 생략가능 하다(공백문자가 있으면 안됨 => 사원 번호)

```
SQL>SELECT empno 사원번호 , ename 사원이름
FROM emp ;
```

(4) 연산자

1) 산술 연산자 (+ , - , * , /)

-부서번호가 10 번인 사원들의 급여를 출력하되 10% 인상된 금액으로 출력해 보세요.

```
SQL>SELECT sal, sal*1.1
FROM emp
WHERE deptno = 10 ;
```

(4) 연산자

2) 비교 연산자 (= , != , > , < , >= , <=)

-급여가 3000 이상인 사원들의 모든 정보를 출력하세요.

```
SQL>SELECT *
FROM emp
WHERE sal >= 3000 ;
```

-부서번호가 30번이 아닌 사람들의 이름과 부서번호를 출력해보세요.

```
SQL>SELECT ename, deptno
FROM emp
WHERE deptno != 30 ;
```

(4) 연산자

3) 논리 연산자 (AND , OR , NOT)

- 부서번호가 10번이고 급여가 3000 이상인 사원들의 이름과 급여를 출력하세요.

```
SQL>SELECT ename, sal
FROM emp
WHERE deptno = 10 AND sal >= 3000 ;
```

- 직업이 SALESMAN 이거나 MANAGER 인 사원의 사원번호와 부서번호를 출력하세요

```
SQL>SELECT empno, deptno
FROM emp
WHERE job = 'SALESMAN' OR job = 'MANAGER' ;
```

(4) 연산자

4)SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

<1> IN 연산자 (OR 연산자와 비슷한 역활)

- 부서번호가 10번이거나 20번인 사원의 사원번호와 이름, 부서번호 출력하기

```
SQL>SELECT empno, ename, deptno
FROM emp
WHERE deptno=10 OR deptno=20 ;
```

IN 연산자를 사용한다면 ?

```
SQL>SELECT empno, ename, deptno
FROM emp
WHERE deptno IN(10,20) ;
```

(4) 연산자

4) SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

<2>ANY 연산자 (조건을 비교할때 어느 하나라도 맞으면 true)

```
SQL>SELECT empno, sal
FROM emp
WHERE sal > ANY(1000, 2000, 3000) ;
```

==> 결과적으로 는 급여가 1000 이상인 로우를 SELECT 하게 된다.

<3>ALL 연산자(조건을 비교할때 조건이 모두 맞느면 true)

```
SQL>SELECT empno, sal
FROM emp
WHERE sal > ALL(1000, 2000, 3000) ;
```

==>결과적으로는 급여가 3000 이상인 로우를 SELECT 하게 된다.

(4) 연산자

4)SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

<4>BETWEEN A AND B (A와 B 사이의 데이타를 얻어온다)

-급여가 1000 과 2000 사이인 사원들의 사원번호,이름,급여를 출력하세요.

```
SQL>SELECT empno, ename, sal
FROM emp
WHERE  sal BETWEEN 1000 AND 2000 ;
```

-사원이름이 'FORD' 와 'SCOTT' 사이의 사원들의 사원번호,이름을 출력해보세요

```
SQL>SELECT empno, ename
FROM emp
WHERE ename BETWEEN 'FORD' AND 'SCOTT' ;
```

(4) 연산자

4)SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

<5> IS NULL (NULL 인경우 TRUE) , IS NOT NULL (NULL 이 아닌경우 TRUE)

-커미션이 NULL 인 사원의 사원이름과 커미션을 출력해보세요

```
SQL>SELECT ename, comm
FROM emp
WHERE comm IS NULL ;
```

-커미션이 NULL 이 아닌 사원의 사원이름과 커미션을 출력해보세요

```
SQL>SELECT ename, comm
FROM emp
WHERE comm IS NOT NULL ;
```

(4) 연산자

4)SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

<6> EXISTS (데이터가 존재하면 TRUE)

- 사원이름이 'FORD' 인 사원이 존재하면 사원의 이름과 커미션을 출력하기

```
SQL>SELECT ename, comm
FROM emp
WHERE EXISTS (SELECT ename FROM emp WHERE ename='FORD');
```

<7>LIKE 연산자 (문자열 비교) 중요 ! ! **********

-사원이름이 'J' 로 시작하는 사원의 사원이름과 부서번호를 출력하기

```
SQL>SELECT ename, deptno
FROM emp
WHERE ename LIKE 'J%' ;
```

(4) 연산자

4)SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

-사원이름에 'J' 가 포함된는 사원의 이름과 부서번호를 출력하기

```
SQL>SELECT ename, deptno
FROM emp
WHERE ename LIKE '%J%' ;
```

-사원이름의 두번째 글자가 'A' 인 사원의 이름,급여,입사일을 출력하기

```
SQL>SELECT ename, sal, hiredate
FROM emp
WHERE ename LIKE '_A%' ;
```

(4) 연산자

4)SQL 연산자 (IN , ANY , ALL , BETWEEN , LIKE , IS NULL , IS NOT NULL)

-사원 이름이 'ES' 로 끝나는 사원의 이름,급여,입사일을 출력해 보세요.

```
SQL>SELECT ename, sal, hiredate
FROM emp
WHERE ename LIKE '%ES' ;
```

-입사년도가 81년 인 사원들의 입사일과 사원번호를 출력해 보세요.

```
SQL>SELECT hiredate, empno
FROM emp
WHERE hiredate LIKE '81%' ;
```

<5> 결합 연산자 (||) =>단순히 문자열을 연결해서 하나의 데이타로 리턴한다.

```
SQL>SELECT ename || '의 직업은' || job || ' 입니다.' FROM emp ;
```

2. 함수 (Function)

- 어떠한 일을 수행하는 기능으로써 주어진 인수를 재료로 처리를 하여 그 결과를반환하는 일을 수행한다.

- 함수의 종류 -

1)단일행 함수
하나의 row 당 하나의 결과값을 반환하는 함수.

2)복수행 함수
여러개의 row 당 하나의 결과값을 반환하는 함수.

(1) 단일행 함수 => 문자함수

<1> CHR(아스키 코드)

```
SQL>SELECT CHR(65) FROM DUAL ;
```

<2>CONCAT(칼럼명, '붙일문자') =>문자열 연결함수

```
SQL>SELECT CONCAT(ename, ' 님' ) name FROM emp ;
```

<3>INITCAP('문자열') => 시작문자를 대문자로 바꿔준다.

```
SQL>SELECT INITCAP( 'hello world') FROM DUAL;
```

<4>LOWER('문자열') =>문자열을 소문자로 바꿔준다.

```
SQL>SELECT LOWER( 'HELLO!' ) FROM DUAL;
```

<5>UPPER('문자열') =>문자열을 대문자로 바꿔준다.

```
SQL>SELECT UPPER( 'hello!' ) FROM DUAL;
```

<6>LPAD('문자열' , 전체 자리수 , '남는자리를 체울 문자') =>왼쪽에 체운다.

```
SQL>SELECT LPAD( 'HI', 10 , '*' ) FROM DUAL;
```

(1) 단일행 함수 => 문자함수

<7>RPAD('문자열' , 전체 자리수 , '남는자리를 체울 문자') =>오른쪽에 체운다.

```
SQL>SELECT RPAD( 'HELLO', 15 , '^' ) FROM DUAL;
```

<8>LTRIM('문자열' , '제거할문자')

```
SQL>SELECT LTRIM( 'ABCD' , 'A' ) FROM DUAL;
SQL>SELECT LTRIM( ' ABCD', ' ' ) FROM DUAL;
SQL>SELECT LTRIM( ' AAAABBACC', 'A' ) FROM DUAL;
SQL>SELECT LTRIM( 'ACACBCD' , 'AC' ) FROM DUAL;
```

<9>RTRIM('문자열' , '제거할문자')

```
SQL>SELECT RTRIM( 'ACACBCD', 'CD') FROM DUAL;
```

<10>REPLACE('문자열1' , '문자열2' , '문자열3')

=> 문자열 1에 있는 문자열중 문자열2를 찾아서 문자열3 으로 바꿔준다.

```
SQL>SELECT REPLACE( 'Hello mimi' , 'mimi', 'mama' )  FROM DUAL;
```

<11>SUBSTR('문자열' , N1, N2)

=>문자열의 N1 번째 위치에서 N2 개만큼 문자열 빼오기

```
SQL>SELECT SUBSTR( 'ABCDEFGHIJ' , 3 , 5) FROM DUAL;
```

(1) 단일행 함수 => 문자함수

EX) EMP 테이블에서 ename(사원이름) 의 두번째 문자가 'A' 인 사원의 이름을 출력한다면?

```
SQL>SELECT ename
FROM EMP
WHERE SUBSTR( ename, 2, 1) = 'A' ;
```

<12> ASCII('문자') =>문자에 해당하는 ASCII 코드값을 반환한다.

```
SQL>SELECT ASCII( 'A' ) FROM DUAL;
```

<13> LENGTH('문자열') =>문자열의 길이를 반환한다.

```
SQL>SELECT LENGTH( 'ABCDE' ) FROM DUAL;
```

EX) EMP 테이블에서 사원이름이 5 자 이상인 사원들의 사번과 이름을 출력하기.

```
SQL>SELECT empno,ename
FROM EMP
WHERE LENGTH(ename) >= 5 ;
```

(1) 단일행 함수 => 문자함수

<14>LEAST('문자열1', '문자열2' , '문자열3')
=>문자열 중에서 가장 앞의 값을 리턴한다.

SQL>SELECT LEAST('AB','ABC','D') FROM DUAL;

<15>NVL(칼럼명 , 값) => 해당 칼럼이 NULL 인경우 정해진 값을 반환한다.
SQL>SELECT ename,NVL(comm, 0) FROM emp ;

(1) 단일행 함수 => 숫자함수

<1>ABS(숫자) => 숫자의 절대값을 반환한다.
SQL>SELECT ABS(-10) FROM DUAL;

<2>CEIL(소수점이 있는 수) => 파라미터 값보다 같거나 가장 큰 정수를 반환(올림)
SQL>SELECT CEIL(3.1234) FROM DUAL;
SQL>SELECT CEIL(5.9999) FROM DUAL;

<3>FLOOR(소수점이 있는 수) =>파라미터 값보다 같거나 가장 작은 정수반환(내림)
SQL>SELECT FLOOR(3.2241) FROM DUAL;
SQL>SELECT FLOOR(2.888829) FROM DUAL;

<4>ROUND(숫자,자리수) =>숫자를 자리수+1 번째 위치에서 반올림한다.
SQL>SELECT ROUND(3.22645, 2) FROM DUAL;
SQL>SELECT ROUND(5.2345, 3) FROM DUAL;

<5>MOD(숫자1 , 숫자2) =>숫자1을 숫자2로 나눈 나머지를 리턴한다.
SQL>SELECT MOD(10,3) FROM DUAL;

(1) 단일행 함수 => 숫자함수

<6>TRUNC(숫자1, 자리수)
=> 숫자1의 값을 소주점이하 자리수까지만 나타낸다. 나머지는 잘라낸다.

```
SQL>SELECT TRUNC(12.23532576 , 2) FROM DUAL;
SQL>SELECT TRUNC(34.1234) FROM DUAL;
```

(1) 단일행 함수 => 날짜함수

<1> SYSDATE = > 현재 시간을 리턴한다.

```
SQL>SELECT SYSDATE FROM DUAL;
```

<2>ADD_MONTHS(날짜, 더해질월)

```
SQL>SELECT ADD_MONTHS(SYSDATE, 10) FROM DUAL;
```

<3>LAST_DAY(날짜) => 해당날짜에 해당하는 달의 마지막 날짜을 반환한다.

```
SQL>SELECT LAST_DAY(SYSDATE) FROM DUAL;
```

<4>MONTHS_BETWEEN(날짜1, 날짜2) => 두 날짜 사이의 월의 수

```
SQL>SELECT empno,MONTHS_BETWEEN(SYSDATE, HIREDATE) 근무개월
FROM emp ;
```

(1) 단일행 함수 => 문자 변환함수

```
SQL>SELECT TO_CHAR(SYSDATE , 'YYYY-MM-DD') FROM DUAL;
SQL>SELECT TO_CHAR(SYSDATE , 'YYYY:MM:DD') FROM DUAL;
SQL>SELECT TO_CHAR(SYSDATE , 'YYYY.MM.DD') FROM DUAL;
SQL>SELECT TO_CHAR(SYSDATE , 'YY.MM.DD') FROM DUAL;
SQL>SELECT TO_CHAR(SYSDATE , 'YY" 년 "MM" 월 "DD" 일 "') FROM DUAL;

SQL>SELECT TO_CHAR(SYSDATE , 'HH:MI:SS' ) FROM DUAL;
SQL>SELECT TO_CHAR(SYSDATE , 'AM HH:MI:SS' ) FROM DUAL;
SQL>SELECT TO_CHAR(SYSDATE , 'PM HH:MI:SS' ) FROM DUAL;
 SQL>SELECT TO_CHAR(SYSDATE , 'HH24:MI:SS' ) FROM DUAL;
 SQL>SELECT TO_CHAR(SYSDATE , 'HH24" 시 "MI" 분 "SS" 초 "' ) FROM DUAL;
 SQL>SELECT TO_CHAR(SYSDATE , 'YY" 년 "MM" 월 "DD" 일 " HH24" 시 "MI" 분
"SS" 초 "') D FROM DUAL;

sql>SELECT TO_CHAR(SYSDATE, 'MM.DD day') FROM DUAL;
sql>SELECT TO_CHAR(SYSDATE, 'MM.DD dy') FROM DUAL;
sql>SELECT TO_CHAR(SYSDATE, 'MM.DD d') FROM DUAL;

ex) 현재 날짜는 2016년 11월 3일 목요일 오후 12:10 입니다.
sql> SELECT '현재 날짜는 ' || TO_CHAR(SYSDATE, 'YYYY"년 "MM"월 "DD"일" day
AM HH:MI') || ' 입니다' FROM DUAL;
```

(1) 단일행 함수 => 숫자변환함수, 날짜변환함수

- TO_NUMBER('숫자에 대응되는 문자');

```
SQL>SELECT TO_NUMBER('999') +1 FROM DUAL;
```

- TO_DATE('날짜에 대응되는 문자')

```
SQL>SELECT TO_DATE('2012-12-12') FROM DUAL;
```

(1) 단일행 함수 => 날짜변환함수

- TO_DATE(문자열, 형식)

```
CREATE TABLE message(
   num NUMBER PRIMARY KEY,
   msg VARCHAR2(20),
   regdate DATE );
```

- 데이터 저장하기

```
sql>INSERT INTO message (num,msg,regdate)
VALUES(1, 'hi', SYSDATE);

sql>INSERT INTO message
VALUES(2, 'hello', '2016/10/20');

sql>INSERT INTO message
VALUES(2, 'gura!', TO_DATE('20161020123020','YYYYMMDDHHMISS'));

sql>INSERT INTO message
VALUES(2, 'gura!', TO_DATE('20161020 오전 12:30','YYYYMMDD AM HH:MI'));
```

(2) 복수행 (그룹) 함수

1) COUNT(칼럼명) =>해당 칼럼이 존재하는 ROW 의 갯수를 반환한다.
단, 저장된 데이터가 NULL 인 칼럼은 세지 않는다.

SQL>SELECT COUNT(ename) FROM emp;

SQL>SELECT COUNT(comm) FROM emp;

SQL>SELECT COUNT(*) FROM emp ; => 모든 행(row)의 갯수를 얻어온다.

2)SUM(칼럼명) => 해당 칼럼의 값을 모두 더한 값을 리턴한다.

SQL>SELECT SUM(sal) FROM emp;

복수행 (그룹) 함수

3)AVG(칼럼명) => 해당 칼럼의 값을 모든값을 더한후 ROW 의 갯수로 나눈 평균값을 리턴한다. 단 NULL 칼럼은 제외된다.

```
SQL>SELECT AVG(sal) FROM emp;

SQL>SELECT AVG(comm) FROM emp;
```

ex) comm 이 NULL 인 사원도 평균에 포함 시켜서 출력을 하려면?

hint : NVL() 함수를 이용한다.

```
SQL>SELECT AVG( NVL( comm , 0 ) ) FROM emp;
```

4) MAX(칼럼명) => 최대값을 리턴한다.

```
SQL>SELECT MAX(sal) FROM emp;
```

5) MIN(칼럼명) =>최소값을 리턴한다.

```
SQL>SELECT MIN(sal) FROM emp;
```

GROUP BY 절

-그룹으로 묶을 때 사용한다.

- 부서별 급여의 총합을 출력하라.

```
SQL>SELECT deptno,SUM(sal)
FROM emp
GROUP BY deptno ;
```

- 부서별 평균 급여를 구해보세요.

```
SQL>SELECT deptno, AVG(sal)
FROM emp
GROUP BY deptno;
```

- 부서별 평균 급여를 구해보세요 (반올림해서 소수 첫째 자리 까지만)

```
SQL>SELECT deptno, ROUND( AVG(sal) , 1 )
FROM emp
GROUP BY deptno ;
```

GROUP BY 절

- 직업별 최대 급여를 구해보세요.

```
SQL>SELECT job, MAX(sal)
FROM emp
GROUP BY job ;
```

-급여가 1000 이상인 사원들의 부서별 평균 급여의 반올림 값을 부서번호로 내림차순 정렬해서 출력해 보세요.

```
SQL>SELECT deptno, ROUND( AVG(sal) )
FROM emp
WHERE sal >= 1000
GROUP BY deptno
ORDER BY deptno DESC ;
```

GROUP BY 절

- 급여가 2000 이상인 사원들의 부서별 평균 급여의 반올림 값을 평균 급여의 반올림 값으로 오름차순 정렬해서 출력해 보세요.

```
SQL>SELECT deptno, ROUND( AVG(sal) )
FROM emp
WHERE sal >= 2000
GROUP BY deptno
ORDER BY ROUND( AVG(sal) ) ASC ;
```

-각 부서별 같은 업무(job)를 하는 사람의 인원수를 구해서 부서번호, 업무(job), 인원수를 부서번호에 대해서 오름차순 정렬해서 출력해 보세요.

```
SQL>SELECT deptno, job, count(*)
FROM emp
GROUP BY deptno, job
ORDER BY deptno ASC;
```

GROUP BY 절

- 급여가 1000 이상인 사원들의 부서별 평균 급여를 출력해보세요
단, 부서별 평균 급여가 2000 이상인 부서만 출력하세요.

```
SQL>SELECT deptno, AVG(sal)
FROM emp
WHERE sal >= 1000
GROUP BY deptno
HAVING AVG(sal) >= 2000 ;
```

JOIN

- 하나의 테이블로 원하는 칼럼정보를 참조 할수 없는 경우 관련된 테이블을 논리적으로 결합하여 원하는 칼럼 정보를 참조하는 방법을 JOIN 이라고 한다.

[형식]

SELECT 칼럼명1,칼럼명2...
FROM 테이블명1, 테이블명2...
WHERE JOIN 조건 AND 다른 조건 ...

JOIN

-EMP 테이블의 모든 사원들의 이름,부서번호,부서명을 출력해 보세요.

```
SQL>SELECT  ename, emp.deptno, dname
FROM emp,dept
WHERE emp.deptno = dept.deptno ;
```

- 급여가 3000 에서 5000 사이의 사원이름과 부서명을 출력해보세요.

```
SQL>SELECT ename, dname
FROM emp,dept
WHERE emp.deptno = dept.deptno AND
sal BETWEEN 3000 AND 5000 ;

SQL>SELECT ename, dname
FROM emp,dept
WHERE emp.deptno = dept.deptno AND
( sal >= 3000 AND sal <= 5000 ) ;
```

JOIN

- 부서명이 'ACCOUNTNG' 인 사원의 이름,입사일,부서번호,부서명을 출력해보세요.

```
SQL>SELECT ename, hiredate, emp.deptno, dname
FROM emp , dept
WHERE emp.deptno = dept.deptno
AND dname = 'ACCOUNTING' ;
```

*테이블에 별칭(alias) 를 붙인다면

```
SQL>SELECT ename, hiredate, e.deptno, dname
FROM emp e, dept d
WHERE e.deptno = d.deptno
AND dname = 'ACCOUNTING' ;

SQL>SELECT e.ename, e.hiredate, e.deptno, d.dname
FROM emp e, dept d
WHERE e.deptno = d.deptno
AND d.dname = 'ACCOUNTING' ;
```

ANSI JOIN 표현식을 사용한다면

- 부서명이 'ACCOUNTNG' 인 사원의 이름,입사일,부서번호,부서명을 출력해보세요.

```
SQL>SELECT ename, hiredate, emp.deptno, dname
FROM emp
INNER JOIN dept ON emp.deptno = dept.deptno
WHERE dname = 'ACCOUNTING' ;
```

*테이블에 별칭(alias) 를 붙인다면

```
SQL>SELECT ename, hiredate, e.deptno, dname
FROM emp e
INNER JOIN dept d ON e.deptno=d.deptno
WHERE d.dname = 'ACCOUNTING' ;

SQL>SELECT e.ename, e.hiredate, e.deptno, d.dname
FROM emp e
INNER JOIN dept d ON e.deptno = d.deptno
WHERE d.dname = 'ACCOUNTING' ;
```

ANSI JOIN 표현식(조인 조건의 칼럼명이 같다면)

- 부서명이 'ACCOUNTNG' 인 사원의 이름,입사일,부서번호,부서명을 출력해보세요.

```
SQL>SELECT ename, hiredate, deptno, dname
FROM emp
INNER JOIN dept USING(deptno)
WHERE dname = 'ACCOUNTING' ;
```

*테이블에 별칭(alias) 를 붙인다면

```
SQL>SELECT ename, hiredate, e.deptno, dname
FROM emp e
INNER JOIN dept d USING(deptno)
WHERE d.dname = 'ACCOUNTING' ;

SQL>SELECT e.ename, e.hiredate, e.deptno, d.dname
FROM emp e
INNER JOIN dept d USING(deptno)
WHERE d.dname = 'ACCOUNTING' ;
```

JOIN

- 커미션이 null 이 아닌 사원의 이름, 입사일, 부서명을 출력해보세요

```
SQL>SELECT e.ename, e.hiredate, d.dname
FROM emp e, dept d
WHERE e.deptno = d.deptno
AND e.comm IS NOT NULL ;
```

2) SELF 조인

- 참조해야할 칼럼이 자신의 테이블에 있는 경우에 사용하는 JOIN 방법

- 각 사원의 이름과 매니져 이름을 출력하세요.

```
SQL>SELECT e1.ename, e2.ename
FROM emp e1, emp e2
WHERE e1.mgr = e2.empno ;
```

JOIN

3)OUTER JOIN 조인

- 한쪽 테이블에는 해당하는 데이터가 존재하는데 다른 테이블에는 데니터가 존재하지 않을때에도 모든 데이터를 추출하도록 하는 JOIN 방법

- 사원번호,부서번호,부서명을 출력하세요
단, 사원이 근무하지 않는 부서명도 같이 출력해보세요.

```
SQL>SELECT e.empno, d.deptno, d.dname
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno ;
```

QUIZ

1. emp 테이블과 dept 테이블을 조인하여 부서번호,부서명,이름,급여를 출력해 보세요!

```
SQL>SELECT e.deptno, d.dname, e.ename, e.sal
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

2. 사원의 이름이 'ALLEN' 인 사원의 부서명을 출력해보세요.

```
SQL>SELECT e.ename, d.dname
FROM emp e, dept d
WHERE e.deptno = d.deptno
AND e.ename = 'ALLEN' ;
```

3. 모든 사원의 이름, 부서번호, 부서명, 급여를 출력하세요.
단, emp 테이블에 없는 부서도 출력해보세요.

```
SQL>SELECT e.ename, e.deptno, d.dname, e.sal
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno ;
```

4. 다음과 같이 모든 사원의 매니저를 출력해보세요.

```
SQL>SELECT e1.ename || ' 의 매니저는 ' || e2.ename || ' 입니다'
FROM emp e1,emp e2
WHERE e1.mgr = e2.empno ;
```

SMITH 의 매니저는 FORD 입니다.
??? 의 매니저는 ??? 입니다.
.

5. 사원의 이름과 급여, 급여의 등급을 출력해 보세요

```
SQL>SELECT ename, sal, grade
FROM emp, salgrade
WHERE sal BETWEEN losal AND hisal;
```

6.사원의 이름과, 부서명, 급여의 등급을 출력해 보세요

```
SQL>SELECT e.ename, d.dname, s.grade
FROM emp e, dept d, salgrade s
WHERE e.deptno = d.deptno
AND e.sal BETWEEN s.losal AND s.hisal;
```

서브 쿼리

- 하나의 SQL 문장절에 포함된 또다른 SELECT 문장, 따라서 두번 질의를 해야 얻을수 있는 결과를 한번의 질의로 해결이 가능하게 하는 쿼리

-용어: Main-Query 또는 Outer-Query
Sub-Query 또는 Inner-Query 두쌍이 같은 의미이다.

-특징: 괄호를 반듯이 묶어야한다.
서브쿼리는 메인 쿼리의 다음 부분에 위치할수 있다
1)SELECT / DELETE / UPDATE 문의 FROM 절과 WHERE 절
2)SELECT 문의 HAVING 절
3)INSERT 문의 INTO 절
4)UPDATE 문의 SET 절

-종류

<1> 단일행 서브쿼리

-서브쿼리의 실행결과가 하나의 칼럼과 하나의 행만을 리턴해주는 쿼리
(하나의 데이터만 리턴해주는 쿼리)

<2> 복수행 서브쿼리

-서브쿼리의 실행결과가 하나의 칼럼과 여러개의 행을 리턴해주는 쿼리
(여러개의 데이터만 리턴해주는 쿼리)

*** 단일행 서브 쿼리 테스트 ***

1. 'SMITH' 가 근무하는 부서명을 서브쿼리를 이용해서 출력해 보세요.

```
>SELECT dname
FROM dept
WHERE deptno = (SELECT deptno FROM emp WHERE ename='SMITH');
```

2. 'ALLEN' 과 같은 부서에서 근무하는 사원의 이름과 부서의 번호를 출력해 보세요.

```
>SELECT ename,deptno
FROM emp
WHERE deptno = (SELECT deptno FROM emp WHERE ename='ALLEN');
```

3. 'ALLEN' 과 동일한 직책(job) 을 가진 사원의 사번과 이름, 직책을 출력해 보세요.

```
>SELECT empno, ename, job
FROM emp
WHERE job = (SELECT job FROM emp WHERE ename='ALLEN');
```

*** 단일행 서브 쿼리 테스트 ***

4. 'ALLEN' 의 급여와 동일하거나 더 많이 받는 사원의 이름과 급여를 출력해 보세요.

```
>SELECT ename, sal
FROM emp
WHERE sal >= (SELECT sal FROM emp WHERE ename='ALLEN');
```

5. 'DALLAS' 에서 근무하는 사원의 이름, 부서번호를 출력해보세요.

```
>SELECT ename, deptno
FROM emp
WHERE deptno = (SELECT deptno FROM dept WHERE loc='DALLAS');
```

6. 'SALES' 부서에서 근무하는 모든 사원의 이름과 급여를 출력해보세요.

```
>SELECT ename, sal
FROM emp
WHERE deptno = (SELECT deptno FROM dept WHERE dname='SALES');
```

*** 단일행 서브 쿼리 테스트 ***

7. 자신의 직속 상관이 'KING' 인 사원의 이름과 급여를 출력해 보세요.

```
>SELECT ename,sal
FROM emp
WHERE mgr = (SELECT empno FROM emp WHERE ename='KING');
```

*** 다중행 서브 쿼리 테스트 ***

- 다중행 서브쿼리의 결과값을 조건절에서 사용할때는 반드시 다중행 연산자와 함께 사용해야한다. (IN, ALL, ANY, EXIST)

1. 급여를 3000 이상받는 사원이 소속된 부서와 동일한 부서에서 근무하는 사원들의 이름과 급여, 부서번호를 출력해 보세요.

```
>SELECT ename,sal,deptno
FROM emp
WHERE deptno IN(SELECT deptno FROM emp WHERE sal>=3000);
```

*** 다중행 서브 쿼리 테스트 ***

2. IN 연산자를 이용하여 부서별로 가장 급여를 많이 받는 사원의 사원번호, 급여, 부서번호를 출력해보세요.

```
>SELECT empno, sal, deptno
FROM emp
WHERE sal IN(SELECT MAX(sal) FROM emp GROUP BY deptno);
```

3. 직책이 MANAGER 인 사원이 속한 부서의 부서번호와 부서명과 부서의 위치를 출력해보세요.

```
>SELECT deptno, dname, loc
FROM dept
WHERE deptno IN(SELECT deptno FROM emp WHERE job='MANAGER');
```

*** 다중행 서브 쿼리 테스트 ***

4. 30번 부서의 사원중에서 급여를 가장 많이 받는 사원보다 더 많은 급여를 받는 사원의 이름과 급여를 출력해보세요.

- 단일 행 서브쿼리 -

```
>SELECT ename,sal
FROM emp
WHERE sal > (SELECT MAX(sal) FROM emp GROUP BY deptno HAVING
deptno=30);
```

- 다중 행 서브 쿼리 -

```
>SELECT ename,sal
FROM emp
WHERE sal > ALL(SELECT sal FROM emp WHERE deptno=30);
```

5. 직책이 'SALESMAN' 보다 급여를 많이 받는 사원들의 이름과 급여를 출력하라.
   (ANY 연산자 이용)

```
>SELECT ename,sal
FROM emp
WHERE sal > ANY(SELECT sal FROM emp WHERE job='SALESMAN');
```

*** 다중행 서브 쿼리 테스트 ***

6. 부서번호가 30번인 사원들의 급여중 최저 급여보다 높은 급여를 받는 사원의 이름, 급여를 출력해보세요.

- 단일행 서브 쿼리 -

```
>SELECT ename,sal
FROM emp
WHERE sal > (SELECT MIN(sal) FROM emp GROUP BY deptno HAVING
deptno=30);
```

- 다중행 서브 쿼리 -
(ANY 연산자 사용)

```
>SELECT ename,sal
FROM emp
WHERE sal > ANY(SELECT sal FROM emp WHERE deptno=30);
```

*** 다중행 서브 쿼리 테스트 ***

7. 직책이 'SALESMAN' 인 사원의 최소 급여보다 많이 받는 사원들의 이름과 급여, 직책을 출력하되 'SALESMAN' 은 출력하지 않습니다.
   (ANY 연산자를 사용하세요)

```
>SELECT ename, sal, job
FROM emp
WHERE sal > ANY(SELECT sal FROM emp WHERE job='SALESMAN')
AND job != 'SALESMAN' ;
```

8. SMITH 와 동일한 직책을 가진 사원의 이름과 직책을 출력하세요

```
>SELECT ename, job
FROM emp
WHERE job = (SELECT job FROM emp WHERE ename='SMITH');
```

*** 다중행 서브 쿼리 테스트 ***

9. 직책이 'SALESMAN' 인 사원이 받는 급여들의 최대 급여보다 많이 받는 사원들의 이름과 급여를 출력하되 부서번호가 20번인 사원은 제외한다.
(ALL 연산자 이용)

```
>SELECT ename,sal
FROM emp
WHERE sal > ALL(SELECT sal FROM emp WHERE job='SALESMAN')
AND deptno != 20;
```

10. 직책이 'SALESMAN' 인 사원이 받는 급여들의 최소 급여보다 많이 받는 사원들의 이름과 급여를 출력하되 부서번호가 20번인 사원은 제외한다.
(ANY 연산자 이용)

```
>SELECT ename,sal
FROM emp
WHERE sal > ANY (SELECT sal FROM emp WHERE job='SALESMAN')
AND deptno != 20;
```

[2] DML (Data Multipulation Language)

-테이블 내의 데이터를 입력, 수정, 삭제

1) INSERT : 테이블에 데이터를 저장할때 사용

(1) 형식

```
INSERT INTO 테이블명( 칼럼명1, 칼럼명,2 ….)
VALUES (값1, 값2, ….)
```

(2) 입력시 제약 사항

-한번에 하나의 행만 입력할수 있다

-INSERT 절에 명시되는 칼럼의 갯수와 VALUES 절의 갯수는 일치해야한다.

-모든 칼럼의 내용을 다 저장할때는 칼럼명은 생략 가능하다.

예) EMP 테이블에 아래와 같은 사원을 추가 해보세요.

EMPNO : 8000
ENAME : 최수만
JOB : 방장
MGR : 7900
HIREDATE : 오늘
SAL : 2000
COMM : 100
DEPTNO : 40

```
SQL>INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (8000, '최수만', '방장', 7900, SYSDATE, 18, 100, 40 );
```

-연습용 테이블 만들기

```
SQL>CREATE TABLE member( num NUMBER  PRIMARY KEY,
            name VARCHAR2(30),
            addr VARCHAR2(50) ) ;
```

2) UPDATE 문

-데이터를 수정할때 사용하는 문장
-[형식]
UPDATE 테이블명 SET 칼럼명1 = 수정값 , 칼럼명2 = 수정값
WHERE 조건절

ex) member 테이블의 내용중 num 칼럼이 3 인 회원의 주소를 인천 으로 수정해 보세요.

```
SQL>UPDATE member SET addr = '인천'  WHERE num = 3 ;
```

ex) member 테이블의 내용중 num 칼럼이 2인 회원의 이름과 주소를 '한빛' , '강남' 으로 바꿔보세요.

```
SQL>UPDATE member SET name = '한빛' , addr = '강남'
WHERE num = 2 ;
```

3)DELETE 문

-DATA 를 삭제할때 사용하는 문장
-[형식]
DELETE FROM 테이블명 WHERE 조건절

ex) member 테이블에서 주소가 '강남' 인 회원의 정보를 삭제 해보세요

```
SQL>DELETE FROM member WHERE addr = '강남' ;
```

[3] TCL (Transaction Control Language)

- DML 문이 실행되어 DBMS 에 저장되거나 되돌리기 위해 실행되는 SQL 문

1) 트랜젝션
-분리되어서는 안되는 논리적 작업단위

ex) 자신의 통장에서 타인에게 송금한다고 가정을 한다면

<-- 트랜젝션의 시작 -->
-내통장에서 금액이 빠저 나간다.
-수취인 통장에 돈이 입금된다.
<-- 트랜젝션의 끝 -->

(1) 트랜젝션의 시작
- DBMS 에 처음 접속 하였을때
-DML 작업을 한후 COMMIT 혹은 ROLLBACK 을 실행했을때

(2) 끝
-COMMIT 이나 ROLLBACK 이 실행되는 순간
-DB 가 정상/ 비정상 종료 될때
-작업 (세션을 종료할때)

(3) TCL 의 종류
-COMMIT : SQL 문의 결과를 영구적으로 DB 에 반영
-ROLLBACK : SQL 문의 실행결과를 취소 할때
-SAVEPOINT : 트랜젝션의 한지점에 표시하는 임시 저장점

예)

```
SQL> INSERT INTO member VALUES( 4, 'AAA','BBB');
SQL>SAVEPOINT myPoint;
SQL> INSERT INTO member VALUES( 5, 'bbb','BBB');
SQL> INSERT INTO member VALUES( 6, 'ccc','BBB');
SQL>ROLLBACK TO myPoint;
SQL>COMMIT ;
```

[4] DDL (Data Definition Language)

-데이터 베이스 내의 객체(테이블, 시퀀스 , ...) 등을 생성하고 변경하고 삭제하기 위해서 사용되는 SQL 문

1)DDL 의 종류

(1)CREATE : 객체를 생성할때

ex) 일반적인 테이블 생성

```
SQL>CREATE TABLE test(num NUMBER, name VARCHAR2(20));  --테이블 생성
```

-참고-

```
SQL>DESC test ;  - - 테이블 구조 보기
```

(2)ALTER : 객체를 변경할때

ex)

SQL>CREATE TABLE simple(num NUMBER) ; - -테이블 생성

(3)DROP : 객체를 삭제 할때

SQL>DROP TABLE dept2;

SQL>DROP TABLE dept3;

SQL>DROP TABLE dept4;

SQL>DROP TABLE simple;

[5] 제약조건 (Constraint)

- 테이블의 해당 칼럼에 원하지 않는 데이터를 입력/수정/삭제 되는 것을 방지 하기 위해 테이블 생성 또는 변경시 설정하는 조건이다.(저장된 데이터의 신뢰성을 높이기 위해)

1) 종류

(1)NOT NULL : NULL 로 입력이 되어서는 안되는 칼럼에 부여하는 조건으로 칼럼 레벨에서 만 부여할수 있는 제약조건이다.

(2)UNIQUE KEY (유일키) : 저장된 값이 중복되지 않고 오직 유일하게 유지되어야 할때 사용하는 제약조건이다. (NULL 은 허용된다)

(3)PRIMARY KEY (대표키) : NOT NULL 조건과 UNIQUE KEY 를 합친 조건이다.

(4)CHECK : 조건에 맞는 데이터만 입력되도록 조건을 부여하는 제약 조건

(5)FOREIGN KEY (외래키) : 부모 테이블의 PRIMARY KEY 를 참조하는 칼럼에 붙이는 제약조건 이다(예 emp 테이블의 deptno 칼럼)

2) 제약조건 (예제)

```
SQL>CREATE TABLE dept2
(deptno NUMBER(2) CONSTRAINT dept2_deptno_pk PRIMARY KEY,
dname VARCHAR2(15) DEFAULT '영업부',
loc CHAR(1) CONSTRAINT dept2_loc_ck CHECK( loc IN( '1' , '2') ) );
```

*** CONSTRAINT 와 제약 조건명은 생략이 가능하다(칼럼 레벨일때)

```
SQL>CREATE TABLE dept3
(deptno NUMBER(2) PRIMARY KEY,
dname VARCHAR2(15) DEFAULT '영업부',
loc CHAR(1)  CHECK( loc IN( '1' , '2') ) );
```

*** 외래키를 만들기 위해서는 부모 테이블을 먼저 만들어야한다.

```
SQL> CREATE TABLE dept2(
deptno NUMBER(2) PRIMARY KEY,
dname VARCHAR2(15) NOT NULL);

SQL> CREATE TABLE emp2(
empno NUMBER(4) PRIMARY KEY,
ename VARCHAR2(15) NOT NULL,
deptno NUMBER(2) REFERENCES dept2(deptno) );
```

4)제약조건 알아보기

-제약조건 이름 검색하기

SQL>SELECT CONSTRAINT_NAME FROM USER_CONSTRAINTS ;

-제약조건은 수정은 불가능하고 삭제만 가능하다

SQL>ALTER TABLE dept2 DROP CONSTRAINT 제약조건명 ;

-제약조건 추가하기

SQL>ALTER TABLE dept2 ADD(CONSTRAINT 제약조건명 PRIMARY KEY(deptno)) ;

Table 칼럼 수정

- 테이블 만들기

```
sql>CREATE TABLE test(num NUMBER);
```

- 칼럼 추가

```
sql>ALTER TABLE test ADD(name VARCHAR2(10));
```

- 칼럼 수정

```
sql>ALTER TABLE test MODIFY(name VARCHAR2(20));
```

- 칼럼의 이름 바꾸기

```
sql>ALTER TABLE test RENAME COLUMN name TO myname;
```

- 칼럼 삭제

```
sql>ALTER TABLE test DROP(myname);
```

Table CONSTRAINT 추가, 삭제

- 테이블 만들기

```
sql>CREATE TABLE dept3(deptno NUMBER(2), dname VARCHAR2(15), loc
CHAR(1) );
```

- 일반 제약조건 추가하기

```
sql>ALTER TABLE dept3
ADD(CONSTRAINT dept3_deptno_pk PRIMARY KEY(deptno));
```

- NOT NULL 제약 조건 추가하기

```
sql>ALTER TABLE dept3
MODIFY dname CONSTRAINT dept3_dname_nn NOT NULL;
```

- 제약조건 삭제하기

```
sql>ALTER TABLE dept3 DROP CONSTRAINT dept3_deptno_pk;
```

[6] 시퀀스 (Sequence)

- 연속적인 숫자 값을 자동으로 증감 시켜서 값을 리턴하는 객체

[형식]

SQL>CREATE SEQUENCE 시퀀스명
INCREMENT BY 한번에 증감할 양(DEFAULT : +1)
START WITH 시작값 (DEFAULT: 0)
cache 여부

ex)테이블 복사하기

```
SQL>CREATE  TABLE dept2 (deptno NUMBER(2),
          dname VARCHAR2(14),
          loc VARCHAR2(13) );

SQL>INSERT INTO dept2 SELECT * FROM dept;
```

ex) 테이블 복사2(CTAS 기법) =>제약조건은 복사가 안된다.

```
SQL>CREATE TABLE dept3 AS SELECT * FROM dept ;
```

ex) 테이블의 구조만 복사하려면
-조건이 항상 거짓이 되는 편법사용

```
SQL>CREATE  TABLE dept4 AS SELECT * FROM dept WHERE 1=2;
```

1)함수

(1) NEXTVAL : 다음값을 얻어온다.

```
SQL>SELECT MY_SEQ.NEXTVAL FROM DUAL;
```

(2) CURRVAL : 현재값을 얻어온다.

```
SQL>SELECT MY_SEQ.CURRVAL FROM DUAL;
```

- 시퀀스 삭제

```
SQL>DROP SEQUENCE 시퀀스명;
```

-사용하고 있는 시퀀스명 조회하기

```
SQL>SELECT SEQUENCE_NAME  FROM USER_SEQUENCES;
```

[7] ROWID 와 ROWNUM

-오라클에서 테이블을 생성하면 기본적으로 제공되는 칼럼
- ROWID : ROW 고유의 아이디 (ROW 를 수정해도 변하지 않음)
- ROWNUM : 행의 INDEX (ROW 삭제시 변경될수 있다)

ex)

```
SQL>SELECT ROWID,ROWNUM FROM  member ;
```

ex)ROW 의 갯수를 알고 싶다면?

```
SQL>SELECT COUNT(*) FROM EMP;
SQL>SELECT MAX(ROWNUM) FROM EMP;      - - 더 빠르다
```

[8] 계정관리하기

1) 생성

-관리자 권한이 있는 관리자 계정으로 접속

SQL> CONN SYSTEM / MASTER ;

-계정 생성하기

SQL>CREATE USER 아이디 IDENTIFIED BY 비밀번호 ;

- 권한주기

SQL>GRANT RESOURCE,CONNECT TO 생성한 아이디;

-생성된 계정으로 접속하기

SQL>CONN 아이디 / 비밀번호;

2)삭제

-계정 삭제 권한이 있는 관리자 계정으로 접속

SQL> DROP USER 아이디 ; - - 생성된 객체가 있으면 삭제가 안된다.

SQL> DROP USER 아이디 CASCADE; - - 무시하고 삭제 가능

3)비밀번호 수정

-ALTER USER 아이디 IDENTIFIED BY 수정할비밀번호

[9] 스칼라 타입(오라클 데이터형)

- CHAR : 고정 길이의 문자, 최대 2000 BYTE

- VARCHAR2 : 가변 길이의 문자, 최대 4000 BYTE

- NUMBER 숫자값을 -38 자리수 부터 +38 자리수를 저장가능
  ex) NUMBER(10) 정수 10자리
  ex) NUMBER(10 , 2) 전체 자리수 10자리 소수점이하 2자리
  ex) NUMBER(P , S) P 는 전체 자리수 , S 는 소수점 이하 자리수

- CLOB : 문자 데이터 최대 4GB 까지 저장 가능하다.
JDBC 에서 읽어올때 getString() 으로 읽어올수 없다(10g 버전 부터 가능) .
getClob() 으로 읽어와야한다.

- DATE : 날짜(시간) 저장 JDBC 에서 getDate() 로 불러 올수도 있지만
TO_CHAR 함수를 이용해서 문자열로 바꿔서 읽어 와야한다.
문자열로 바꾸었다면 getString() 으로 읽어올수 있다.

- BLOB : 2진 데이터, 즉 바이너리 데이터를 저장할때 사용한다.

[[커맨트(dos) 창에서 명령어 사용하기]]

dos>LSNRCTL STOP : 오라클 리스너 정지하기
dos>LSNRCTL START : 오라클 리스너 시작하기
dos>LSNRCTL STATUS : 현재 리스너 상태 보기
dos>TNSPING DBNAME 숫자(확인 횟수) : 접속할 DB 와의 통신상태 확인