FROM book where bookname like '\_구%';--2번째 문자에 '구' 가 들어가는 북네임

- \_ : 1개의 문자와 일치(

- %구% : 앞뒤에 0개 이상의 문자열과 일치

별칭 부여하기(as)

Select stu\_no as Id, stu\_name as name from student;

|| 두 개 의 속성을 붙여서 출력한다.

SELECT stu\_dept || stu\_name FROM student

SELECT stu\_dept||'과 '||stu\_name||' 입니다' FROM student;

속성 사이에 문자열 포함.

Order by 정렬

오름차순으로 정렬할 경우 생략가능(ASC)

내림차순으로 정렬할 경우 DESC를 뒤에 붙인다.

SELECT \* FROM Book ORDER BY price DESC, publisher ASC;

SELECT

stu\_no,stu\_weight as target

FROM student

order by target;

SELECT

stu\_no,stu\_weight as target

FROM student

order by 2; 타겟 대신 타겟의 열 번호를 써도 됨.

select UPPER('korea') from dual;

select initcap('korea') from dual;--첫글자만 대문자.

select LOWER('KOREA')from dual;

select CONCAT('korea','japan')from dual;

select SUBSTR('korea',2,3)from dual;

select sysdate from dual; 오늘 날짜 반환

alter session set nls\_date\_format='mm-dd-yyyy' 날짜 반환 형식 변환

select next\_day(sysdate,2) from dual ->다음주 n요일(1=일 2=월 3=화…)

select round(sysdate,'MON') from dual

====변환 함수=====

To\_number() = 문자를 숫자로

To\_date() = 문자를 날짜로

To\_char() = 숫자와 날짜를 문자로.

NVL(인수1 ,인수2)

:인수값이 NULL인지 체크하여 연산이 가능한 값으로 변환하여 반환하는 함수

Ex)인수1이 널이면 인수2

인수2가 널이면 인수1을 반환한다.

Select nvl(stu\_height,0) from student

Stu\_height가 널이면 0을 반환.

SELECT count(stu\_class) FROM student;

SELECT count(\*) FROM student;

Count(\*) : 전체 레코드(튜플 )의 수

SELECT

stu\_gender,avg(stu\_weight)

FROM student

group by stu\_gender;

그룹핑 젠더(F와 M)을 기준으로 묶는다.

SELECT stu\_grade,

avg(stu\_height)

FROM student

where stu\_dept='기계'

group by stu\_grade

having avg(stu\_height)>=120;

그룹후 조건문은 where이 아니라 having

SELECT

rpad(stu\_name,10,'&')

FROM student

Ex)옥성우&&&&

조인

SELECT \* FROM customer,orders

where orders.custid = customer.custid;

커스터머와 오더의 id를 조인

SELECT

student.stu\_no,stu\_name,enr\_grade

FROM student,enrol

where student.stu\_no=enrol.stu\_no;

다른 테이블의 같은 속성을 합친다.

\*\*이퀴조인(equi join)

where절에서 ‘=’ 연산자 사용

\*\*natural join

Select stu\_no, stu\_name, enr\_grade

From student natural join enroll

=가 필요 없음 알아서 조인 시켜줌

<Join~using>

SELECT

student.stu\_name

FROM enrol join student using (stu\_no)

where enrol.enr\_grade >=60;

테이블1 조인 테이블2 유징(공통 속성)

부속 질의 (옥성우 보다 키가 큰 학생 출력)

SELECT \* FROM student

where stu\_height>

(SELECT stu\_height from student where stu\_name ='옥성우');

(박희철과 같은 몸무게를 가진 학생)

SELECT \* FROM student

where stu\_weight =(select stu\_weight from student where stu\_name = '박희철')

and stu\_name<>'박희철';

김인중이 듣는 강의이름과 교수 이름 출력

SELECT

subject.sub\_name,sub\_prof,stu\_name

FROM student,enrol,subject

where student.stu\_no = enrol.stu\_no and subject.sub\_no = enrol.sub\_no

and student.stu\_name='김인중';

SELECT

subject.sub\_name,sub\_prof,stu\_name

FROM student natural join enrol natural join subject

where student.stu\_name='김인중';

create table test1(u\_id varchar2(10),

u\_data date

);

varchar2(n) : 가변길이 문자 데이터에 대한 정의

number (n,m) : 숫자 데이터에 대한 정의로 n은 자릿수,m은 소수이하 자리수

char(n) : 문자 데이터에 대한 정의

date: 날짜 데이터에 대한 정의

long:2GB의 가변길이 문자 데이터에 대한 정의

alter table t\_student

add(army\_char(1));

기존 테이블에 새로운 열을 추가하는 작업

<기본 테이블의 열 구조 변경>

(modify)

alter table t\_student  
modify(army number) army\_char 🡪 army number로

<테이블 이름 변경>

Rename t\_student to test\_student;

<테이블내의 데이터 삭제>

Truncate table test\_student;

위 테이블내의 모든 데이터를 삭제함..

Delete의 경우 트랜잭션을 종료하여야만 최종 데이터가 삭제되지만,

Truncate의 경우 auto commit이 발생하여 트랜잭션을 종료할 필요가 없다.