

DQL(SELECT)

DQL - SELECT

데이터를 검색(추출)하기 위해 사용되는 언어

DQL은 DML에 속한 언어이기도 하고 데이터를 조회한 결과를 Result Set 이라고 함

SELECT 구문에 의해 반환된 행들의 집합을 의미함

Result Set은 0개 이상의 행이 포함될 수 있음

Result Set은 특정한 기준에 의해 정렬 될 수 있음

특정 컬럼이나 특정 행을 조회할 수 있으며, 여러 테이블에서 특정행 / 컬럼을 조회할 수 있음

SELECT 기본 작성법

SELECT 컬럼명 [, 컬럼명, ...]

FROM 테이블명

WHERE 조건식;

- SELECT : 조회하고자 하는 **컬럼명**을 기술함.
여러 컬럼을 조회하는 경우 컬럼은 쉼표로 구분함.
모든 컬럼을 조회시 컬럼명 대신 '*' 기호를 사용 가능함.
조회 결과는 기술한 컬럼명 순으로 표시함.
- FROM : 조회 대상 컬럼이 포함된 **테이블명**을 기술함.
- WHERE : 행을 선택하는 조건을 기술함.
여러 개의 제한조건을 포함할 수 있으며,
각각의 제한 조건은 논리 연산자로 연결함.
제한조건을 만족시키는 행들만 Result Set에 포함됨

SELECT 사용 예시 - 기본

직원들의 사번, 이름 연봉을 조회하는 SELECT 구문

```
SELECT EMP_ID,  
       EMP_NAME  
       SALARY  
FROM EMPLOYEE;
```

EMP_ID	EMP_NAME	SALARY
200	선동일	8000000
201	송종기	6000000
202	노용철	3700000
203	송은희	2800000
204	유재식	3400000
205	정중하	3900000
206	박나라	1800000
207	하이유	2200000
208	김해술	2500000
209	심봉선	3500000
210	윤은해	2000000

Oracle - DQL

SELECT 사용 예시 - 기본 (모든 정보 조회)

```
SELECT EMP_ID, EMP_NAME, EMP_NO, EMAIL, PHONE, DEPT_CODE,  
       JOB_CODE, SAL_LEVEL, SALARY, BONUS, MANAGER_ID,  
       HIRE_DATE, ENT_DATE, ENT_YN
```

```
FROM EMPLOYEE;
```

또는

```
SELECT * FROM EMPLOYEE;
```

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	DEPT_CODE	JOB_CODE	SAL_LEVEL	SALARY	BONUS	MANAGER_ID	HIRE_DATE	ENT_DATE	ENT_YN
200	선동일	621235-1985634	sun_di@kh.or.kr	01099546325	D9	J1	S1	8000000	0.3 (null)		90/02/06	(null)	N
201	송중기	631156-1548654	song_jk@kh.or.kr	01045686656	D9	J2	S1	6000000	(null) 200		01/09/01	(null)	N
202	노웅철	861015-1356452	no_hc@kh.or.kr	01066656263	D9	J2	S4	3700000	(null) 201		01/01/01	(null)	N
203	송은희	631010-2653546	song_eh@kh.or.kr	01077607879	D6	J4	S5	2800000	(null) 204		96/05/03	(null)	N
204	유재식	660508-1342154	yoo_js@kh.or.kr	01099999129	D6	J3	S4	3400000	0.2 200		00/12/29	(null)	N
205	정중하	770102-1357951	jung_jh@kh.or.kr	01036654875	D6	J3	S4	3900000	(null) 204		99/09/09	(null)	N
206	박나라	630709-2054321	pack_nr@kh.or.kr	01096935222	D5	J7	S6	1800000	(null) 207		08/04/02	(null)	N
207	하미유	690402-2040612	ha_iy@kh.or.kr	01036654488	D5	J5	S5	2200000	0.1 200		94/07/07	(null)	N
208	김해술	870927-1313564	kim_hs@kh.or.kr	01078634444	D5	J5	S5	2500000	(null) 207		04/04/30	(null)	N
209	심봉선	750206-1325546	sim_bs@kh.or.kr	0113654485	D5	J3	S4	3500000	0.15 207		11/11/11	(null)	N
210	윤은혜	650505-2356985	youn_eh@kh.or.kr	0179964233	D5	J7	S5	2000000	(null) 207		01/02/03	(null)	N
211	전형돈	830807-1121321	jun_hd@kh.or.kr	01044432222	D8	J6	S5	2000000	(null) 200		12/12/12	(null)	N
212	장프위	780923-2234542	jang_zw@kh.or.kr	01066682224	D8	J6	S5	2550000	0.25 211		15/06/17	(null)	N
213	하동운	621111-1785463	ha_dh@kh.or.kr	01158456632	(null)	J6	S5	2320000	0.1 (null)		99/12/31	(null)	N
214	방명수	856795-1313513	bang_ms@kh.or.kr	01074127545	D1	J7	S6	1380000	(null) 200		10/04/04	(null)	N
215	대복존	881130-1050911	dae_bh@kh.or.kr	01088808584	D5	J5	S4	3760000	(null) (null)		17/06/19	(null)	N

SELECT 사용 예시 - 컬럼 값 산술연산

```
SELECT  
  EMP_NAME,  
  SALARY * 12,  
  SALARY + (SALARY * BONUS_PCT) * 12  
FROM  
  EMPLOYEE;
```

EMP_NAME	SALARY*12	(SALARY+(SALARY*BONUS))*12
전통일	96000000	124800000
송종기	72000000	(null)
노용철	44400000	(null)
송은희	33600000	(null)
유재식	40800000	48960000
정중하	46800000	(null)
박나라	21600000	(null)
하미유	26400000	29040000
김해솔	30000000	(null)
심봉선	42000000	48300000
윤은해	24000000	(null)
전형돈	24000000	(null)
장프위	30600000	38250000
히두우	33600000	38250000

SELECT 사용 예시 - 컬럼 별칭

```
SELECT EMP_NAME AS 이름,  
       SALARY * 12 "1년 급여(원)",  
       (SALARY + (SALARY * BONUS_PCT)) * 12 AS "총소득(원)"  
FROM EMPLOYEE;
```

숫자 혹은 특수문자가
포함되는 경우에는
" "를 사용해야 한다.

AS는 생략 가능하다.
(공백으로 구분함)

이름	1년 급여	총소득(원)
선동일	96000000	124800000
송종기	72000000	(null)
노용철	44400000	(null)
송은희	33600000	(null)
유재식	40800000	48960000
정종하	46800000	(null)
박나라	21600000	(null)
하미유	26400000	29040000
김해술	30000000	(null)
심봉선	42000000	48300000

SELECT 사용 예시 - 리터럴

임의로 지정한 문자열을 SELECT 절에 사용하면, 테이블에 존재하는 데이터처럼 사용할 수 있다.

```
SELECT EMP_ID,  
       SALARY,  
       '원' AS 단위  
FROM EMPLOYEE;
```

문자 혹은 날짜 리터럴은
' '기호를 사용해야 한다.

리터럴은 Result Set의
모든 행에 반복 표시된다.

EMP_ID	SALARY	단위
200	8000000	원
201	6000000	원
202	3700000	원
203	2800000	원
204	3400000	원
205	3900000	원
206	1800000	원
207	2200000	원
208	2500000	원
209	3500000	원
210	2000000	원

리터럴 - 실습문제

1. EMPLOYEE 테이블에서 이름, 연봉, 총수령액(보너스포함), 실수령액(총수령액-(월급*세금 3%))가 출력되도록 하시오.
2. EMPLOYEE 테이블에서 이름, 근무 일수를 출력해보시오.
(SYSDATE를 사용하면 현재 시간 출력)
3. EMPLOYEE 테이블에서 20년 이상 근속자의 이름, 월급, 보너스율 출력하시오

SELECT 사용 예시 - DISTINCT

```
SELECT EMP_NAME AS 이름,  
       SALARY * 12 "1년 급여(원)",  
       (SALARY + (SALARY * BONUS_PCT)) * 12 AS "총소득(원)"  
FROM EMPLOYEE;
```

숫자 혹은 특수문자가
포함되는 경우에는
" "를 사용해야 한다.

AS는 생략 가능하다.
(공백으로 구분함)

이름	1년 급여	총소득(원)
선동일	96000000	124800000
송종기	72000000	(null)
노웅철	44400000	(null)
송은희	33600000	(null)
유재식	40800000	48960000
정종하	46800000	(null)
박나라	21600000	(null)
하미유	26400000	29040000
김해술	30000000	(null)
심봉선	42000000	48300000

SELECT 사용 예시 - 컬럼 별칭

컬럼에 포함된 중복 값을 한번씩만 표시하고자 할 때 사용한다.

SELECT JOB_CODE
FROM EMPLOYEE;

JOB_CODE
J1
J2
J2
J4
J3
J3
J7
J5
J5
J3
J7
J6
J6
J6
J7
J5
J6
J6

SELECT DISTINCT JOB_CODE
FROM EMPLOYEE;

JOB_CODE
J2
J7
J3
J6
J5
J1
J4

SELECT절에 1회만 기술 가능하다.

SELECT 사용 예시 - WHERE절

검색할 컬럼의 조건을 설정하여 행을 결정한다.

[부서코드가 'D9'인 직원의 이름, 부서코드 조회]

```
SELECT EMP_NAME,  
       DEPT_CODE  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D9';
```

EMP_NAME	DEPT_CODE
선동일	D9
송종기	D9
노용철	D9

DEPT_CODE 값이 'D9'인
행만 Result Set에 포함

[급여가 4000000 보다 많은 직원 이름과 급여 조회]

```
SELECT EMP_NAME,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY > 4000000;
```

EMP_NAME	SALARY
선동일	8000000
송종기	6000000

SALARY 값이 4000000
보다 큰 행만 Result Set
에 포함

SELECT 사용 예시 - WHERE절

여러 개의 조건 작성 시 AND / OR 를 사용할 수 있다.

[부서코드가 D6이고 급여를 2000000보다 많이 받는 직원의 이름, 부서코드, 급여 조회]

```
SELECT EMP_NAME,  
       SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D6'  
AND SALARY > 2000000;
```

	EMP_NAME	SALARY
1	송은희	2800000
2	유재식	3400000
3	정중하	3900000

[부서코드가 D6이거나 급여를 2000000보다 많이 받는 직원의 이름, 부서코드, 급여 조회]

```
SELECT EMP_NAME,  
       SALARY  
FROM EMPLOYEE  
WHERE DEPT_ID = 'D6'  
OR SALARY > 2000000;
```

	EMP_NAME	SALARY
	선동일	8000000
	송중기	6000000
	노웅철	3700000
	송은희	2800000
	유재식	3400000
	정중하	3900000
	하미유	2200000
	김해솔	2500000
	심봉선	3500000
	장프위	2550000
	하동욱	2320000

SELECT 사용 예시 - ORDER BY

SELECT한 컬럼에 대해 정렬을 할 때 사용하는 구문

SELECT 구문의 가장 마지막에 작성하며, 실행순서도 가장 마지막에 수행됨

[표현식]

SELECT 컬럼1, 컬럼2, 컬럼3... FROM 테이블명 WHERE 조건절
ORDER BY 컬럼명|별칭|컬럼순서 정렬방식 [NULLS FIRST | LAST]

[데이터 정렬 방법]

	NUMBER	CHARACTER	DATE	NULL
ASC	작은수 ⇨ 큰수	사전순	빠른날 ⇨ 늦은날	맨 아래행 (NULL값이 맨 아래로 감)
DESC	큰수 ⇨ 작은수	사전역순	늦은날 ⇨ 빠른날	맨 위행 (NULL값이 맨 위로 감)

SELECT 사용 예시 - 정렬 방법

ASC : 오름차순 정렬 (DEFAULT)

지정한 컬럼을 기준으로 오름차순으로 정렬함, 기본적(DEFAULT)으로 ASC 정렬임

ex) 모든 정보 출력시 이름 차순으로 정렬하고 싶을때! (ASC는 사전순)

```
SELECT * FROM EMPLOYEE ORDER BY EMP_NAME;
```

또는

```
SELECT * FROM EMPLOYEE ORDER BY EMP_NAME ASC;
```

DESC : 내림차순 정렬

지정한 컬럼을 기준으로 오름차순으로 정렬함

ex) 모든 정보 출력시 이름 차순으로 정렬하고 싶을때! (DESC는 사전역순)

```
SELECT * FROM EMPLOYEE ORDER BY EMP_NAME DESC;
```

Oracle - DQL

연결 연산자

연결 연산자인 '||'를 사용하여 여러 컬럼을 하나의 컬럼인 것처럼 연결하거나, 컬럼과 리터럴을 연결할 수 있다.

[컬럼과 컬럼을 연결한 경우]

```
SELECT EMP_ID || EMP_NAME || SALARY  
FROM EMPLOYEE;
```

	EMP_ID	EMP_NAME	SALARY
1	200	선동일	8000000
2	201	송종기	6000000
3	202	노용철	3700000
4	203	송은희	2800000
5	204	유재식	3400000
6	205	정중하	3900000
7	206	박나라	1800000
8	207	하이유	2200000
9	208	김해숙	2500000

[컬럼과 리터럴을 연결한 경우]

```
SELECT EMP_NAME || '의 월급은' || SALARY || '원 입니다.'  
FROM EMPLOYEE;
```

	EMP_NAME	'의 월급은'	SALARY	'원 입니다.'
1	선동일	의 월급은	8000000	원 입니다.
2	송종기	의 월급은	6000000	원 입니다.
3	노용철	의 월급은	3700000	원 입니다.
4	송은희	의 월급은	2800000	원 입니다.
5	유재식	의 월급은	3400000	원 입니다.
6	정중하	의 월급은	3900000	원 입니다.
7	박나라	의 월급은	1800000	원 입니다.
8	하이유	의 월급은	2200000	원 입니다.
9	김해숙	의 월급은	2500000	원 입니다.

논리 연산자

여러 개의 제한 조건 결과를 하나의 논리결과로 만들어준다.

연산자	설명
AND	여러 조건이 동시에 TRUE일 경우에만 TRUE값 반환
OR	여러 조건들 중에 어느 하나의 조건만 TRUE이면 TRUE값 반환
NOT	조건에 대한 반대값으로 반환(NULL은 예외)

[AND 연산 결과]

	TRUE	FALSE	NULL
TRUE	T	F	N
FALSE	F	F	F
NULL	N	F	N

[OR 연산 결과]

	TRUE	FALSE	NULL
TRUE	T	T	T
FALSE	T	F	N
NULL	T	N	N

비교 연산자 - 주요 비교 연산자

표현식 사이의 관계를 비교하기 위해 사용하고,
비교 결과는 논리 결과중에 하나(TRUE/FALSE/NULL)가 된다.
단, 비교하는 두 컬럼 값/표현식은 서로 동일한 데이터 타입이어야 한다.

[주요 비교 연산자]

연산자	설명
=	같다
>, <	크다 / 작다
>=, <=	크거나 같다 / 작거나 같다
<>, !=, ^=	같지 않다
BETWEEN AND	특정 범위에 포함되는지 비교
LIKE / NOT LIKE	문자 패턴 비교
IS NULL / IS NOT NULL	NULL 여부 비교
IN / NOT IN	비교 값 목록에 포함/미포함 되는지 여부 비교

비교 연산자 - BETWEEN~AND

비교하려는 값이 지정한 범위(상한 값과 하한 값의 경계도 포함됨)에 포함되면 TRUE를 리턴하는 연산자이다.

[급여를 3500000원보다 많이 받고 6000000보다 적게 받는 직원 이름과 급여 조회]

```
SELECT EMP_NAME,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY BETWEEN 3500000 AND 6000000;
```

또는

```
SELECT EMP_NAME,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY >= 3500000  
AND SALARY <= 6000000;
```

	EMP_NAME	SALARY
1	송종기	6000000
2	노용철	3700000
3	정중하	3900000
4	심봉선	3500000
5	대북훈	3760000
6	전지연	3660000

비교 연산자 - LIKE

비교하려는 값이 지정한 특정 패턴을 만족시키면 TRUE를 리턴하는 연산자로 '%'와 '_'를 와일드카드로 사용할 수 있다.

[‘전’씨 성을 가진 직원 이름과 급여 조회]

```
SELECT EMP_NAME,  
       SALARY  
FROM EMPLOYEE  
WHERE EMP_NAME LIKE '전%';
```

	EMP_NAME	SALARY
1	전철돈	2000000
2	전지연	3660000

[7000번 대 4자리 국번의 전화번호를 사용하는 직원 전화번호 조회]

```
SELECT EMP_NAME,  
       PHONE  
FROM EMPLOYEE  
WHERE PHONE LIKE '__7____';
```

	EMP_NAME	PHONE
1	송은희	01077607879
2	김해술	01078634444
3	방명수	01074127545

Oracle - DML

비교 연산자 - NOT LIKE

[‘이’씨 성이 아닌 직원 사번, 이름, 이메일 조회]

```
SELECT EMP_ID,  
       EMP_NAME,  
       EMAIL  
FROM EMPLOYEE  
WHERE EMP_NAME NOT LIKE ‘이%’;
```

또는

```
SELECT EMP_ID,  
       EMP_NAME,  
       EMAIL  
FROM EMPLOYEE  
WHERE NOT EMP_NAME LIKE ‘이%’;
```

	EMP_ID	EMP_NAME	EMAIL
1	200	선동일	sun_di@kh.or.kr
2	201	송종기	song_jk@kh.or.kr
3	202	노용철	no_hc@kh.or.kr
4	203	송은희	song_eh@kh.or.kr
5	204	유재식	yoo_js@kh.or.kr
6	205	정중하	jung_jh@kh.or.kr
7	206	박나라	pack_nr@kh.or.kr
8	207	하미유	ha_iy@kh.or.kr
9	208	김해슬	kim_hs@kh.or.kr
10	209	심봉선	sim_bs@kh.or.kr
11	210	윤은혜	youn_eh@kh.or.kr
12	211	전형돈	jun_hd@kh.or.kr
13	212	장조위	jang_zw@kh.or.kr
14	213	하동운	ha_dh@kh.or.kr
15	214	방영수	bang_ms@kh.or.kr
16	215	대복훈	dae_bh@kh.or.kr
17	216	차태연	cha_ty@kh.or.kr
18	217	전지연	jun_jy@kh.or.kr
19	219	임시환	im_sw@kh.or.kr
20	221	유하진	yoo_hj@kh.or.kr

비교 연산자 - 간단실습

1. EMPLOYEE 테이블에서 이름 끝이 연으로 끝나는 사원의 이름을 출력하시오
2. EMPLOYEE 테이블에서 전화번호 처음 3자리가 010이 아닌 사원의 이름, 전화번호를 출력하시오
3. EMPLOYEE 테이블에서 메일주소의 's'가 들어가면서, DEPT_CODE가 D9 또는 D6이고 고용일이 90/01/01 ~ 00/12/01이면서, 월급이 270만원이상인 사원의 전체 정보를 출력하시오

비교 연산자 - IS NULL / IS NOT NULL

데이터 값이 null인 경우를 조회할 수 있는 연산자.

[관리자도 없고 부서 배치도 받지 않은 직원 이름 조회]

```
SELECT EMP_NAME,  
       MANAGER_ID,  
       DEPT_CODE  
FROM EMPLOYEE  
WHERE MANAGER_ID IS NULL  
AND DEPT_CODE IS NULL;
```

	EMP_NAME	MANAGER_ID	DEPT_CODE
1	하동운	(null)	(null)
2	미오리	(null)	(null)

[부서 배치를 받지 않았지만 보너스를 지급받는 직원 조회]

```
SELECT EMP_NAME, BONUS, DEPT_CODE  
FROM EMPLOYEE  
WHERE DEPT_CODE IS NULL  
AND BONUS IS NOT NULL;
```

	EMP_NAME	BONUS	DEPT_CODE
1	하동운	0.1	(null)

비교 연산자 - IN

비교하려는 값 목록에 일치하는 값이 있으면 TRUE를 반환하는 연산자.

[60번 부서와 90번 부서원들의 이름, 부서코드, 급여 조회]

```
SELECT EMP_NAME, DEPT_CODE, SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE IN ('D6', 'D8');
```

또는

```
SELECT EMP_NAME, DEPT_CODE, SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D6'  
OR DEPT_CODE = 'D8';
```

	EMP_NAME	DEPT_CODE	SALARY
1	송은희	D6	2800000
2	유재식	D6	3400000
3	정중하	D6	3900000
4	전형돈	D8	2000000
5	장프위	D8	2550000
6	이태림	D8	2436240

연산자 우선순위

여러 연산자를 사용하는 경우 우선순위를 고려해서 사용해야 한다.

우선순위	연산자
1	산술 연산자
2	연결 연산자
3	비교 연산자
4	IS NULL / IS NOT NULL , LIKE , IN / NOT IN
5	BETWEEN AND / NOT BETWEEN AND
6	논리 연산자 - NOT
7	논리 연산자 - AND
8	논리연산자 - OR

연산자 우선순위

[20번 또는 90번 부서원 중 급여를 3000000원 보다 많이 받는 직원의 이름, 급여, 부서코드 조회]

```
SELECT EMP_NAME, SALARY, JOB_CODE  
FROM EMPLOYEE  
WHERE (JOB_CODE = 'J7'  
OR JOB_CODE = 'J2')  
AND SALARY > 3000000;
```

	EMP_N...	SALARY	JOB_CODE
1	송종기	6000000	J2
2	노웅철	3700000	J2
3	미오리	2890000	J7

우선순위를 고려하여 OR가 먼저 처리되도록 () 를 이용하여 우선순위 변경함

```
WHERE (JOB_CODE = 'J7' OR JOB_CODE = 'J2') AND SALARY > 2000000;
```

J7직급이거나 J2직급인 직원들 중 급여 2000000원 이상 받는 직원이라는 의미

연산자 우선순위

[20번 또는 90번 부서원 중 급여를 3000000원 보다 많이 받는 직원의 이름, 급여, 부서코드 조회]

```
SELECT EMP_NAME, SALARY, JOB_CODE  
FROM EMPLOYEE  
WHERE (JOB_CODE = 'J7'  
OR JOB_CODE = 'J2' )  
AND SALARY > 3000000;
```

	EMP_N...	SALARY	JOB_CODE
1	송종기	6000000	J2
2	노웅철	3700000	J2
3	미오리	2890000	J7

우선순위를 고려하여 OR가 먼저 처리되도록 () 를 이용하여 우선순위 변경함

```
WHERE (JOB_CODE = 'J7' OR JOB_CODE = 'J2') AND SALARY > 2000000;
```

J7직급이거나 J2직급인 직원들 중 급여 2000000원 이상 받는 직원이라는 의미

최종 실습 문제

1. 입사일이 5년 이상, 10년 이하인 직원의 이름, 주민번호, 급여, 입사일을 검색하여라
2. 재직중이 아닌 직원의 이름,부서코드를 검색하여라 (퇴사 여부 : ENT_YN)
3. 근속년수가 10년 이상인 직원들을 검색하여 출력 결과는 이름,급여,근속년수 (소수점X)를 근속년수가 오름차순으로 정렬하여 출력하여라.
단, 급여는 50% 인상된 급여로 출력되도록 하여라.

최종 실습 문제

4. 입사일이 99/01/01 ~ 10/01/01 인 사람 중에서 급여가 2000000 원 이하인 사람의 이름, 주민번호, 이메일, 폰번호, 급여를 검색 하시오

5. 급여가 2000000원 ~ 3000000원 인 여직원 중에서 4월 생일자를 검색하여 이름, 주민번호, 급여, 부서코드를 주민번호 순으로(내림차순) 출력하여라.
단, 부서코드가 null인 사람은 부서코드가 '없음' 으로 출력 하여라.

6. 남자 사원 중 보너스가 없는 사원의 오늘까지 근무일을 측정하여 1000일 마다(소수점 제외) 급여의 10% 보너스를 계산하여 이름, 특별 보너스 (계산 금액) 결과를 출력하여라.

단, 이름 순으로 오름 차순 정렬하여 출력하여라.

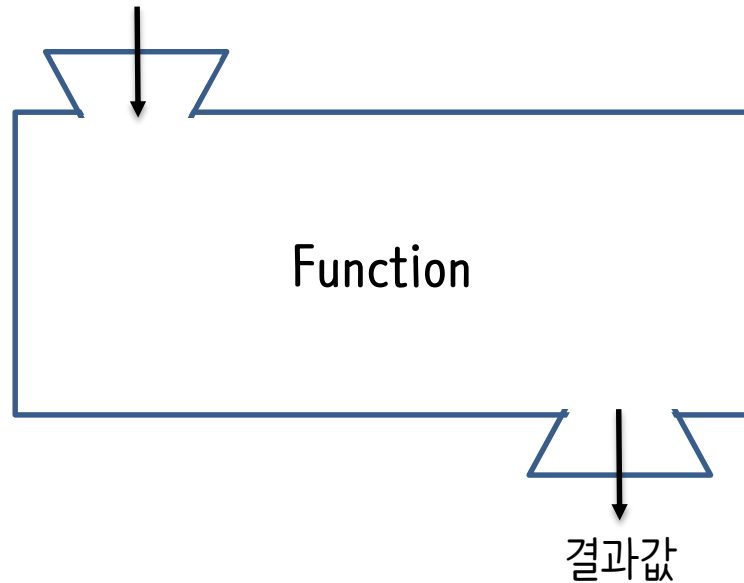
Oracle 함수

Oracle - Function

함수

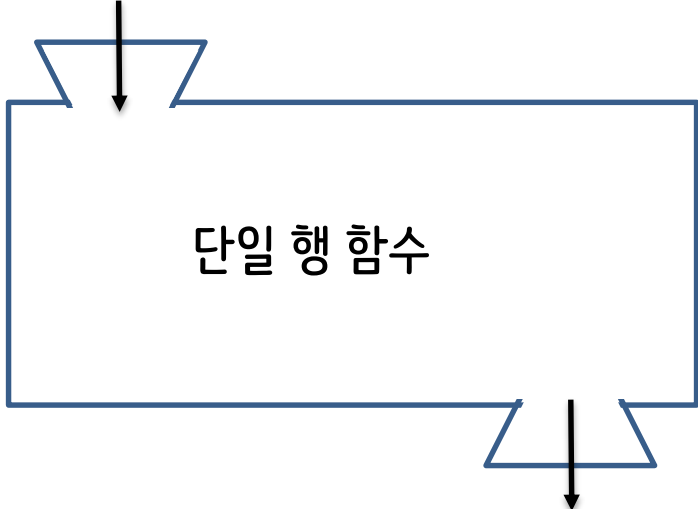
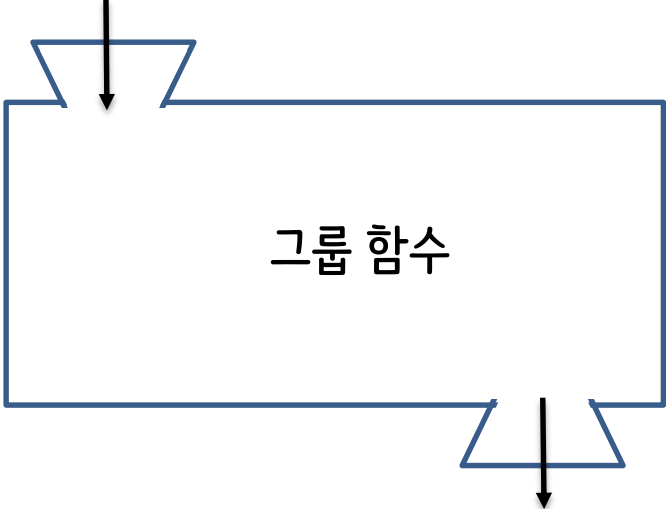
하나의 큰 프로그램에서 반복적으로 사용되는 부분들을 분리하여 작성해 놓은 작은 서브 프로그램으로, 호출하며 값을 전달하면 수행 결과를 리턴하는 방식으로 사용된다.

값 전달하여 호출



Oracle - Function

함수의 유형

<p>여러 개 값 전달</p>  <p>단일 행 함수</p> <p>결과 값 여러개</p>	<p>여러 개 값 전달</p>  <p>그룹 함수</p> <p>결과 값 1개</p>
단일 행 함수	그룹 함수
각 행마다 반복적으로 적용되어 입력 받은 행의 개수만큼 결과를 반환	특정한 행들의 집합으로 그룹이 형성되어 적용됨 그룹당 1개의 결과를 반환한다.

Oracle - Function

문자 처리 함수

구분	입력 값 타입	리턴 값 타입	설명
LENGTH	CHARACTER	NUMBER	문자열의 길이를 반환한다.
LENGTHB			문자열의 바이트 크기를 반환한다.
INSTR			특정 문자의 위치를 반환한다.
INSTRB			특정 문자의 위치 바이트 크기를 반환한다.

문자 처리 함수

구분	입력 값 타입	리턴 값 타입	설명
LPAD	CHARACTER	NUMBER	문자열을 지정된 숫자만큼의 크기로 설정하고, 지정한 문자를 왼쪽부터 채워서 생성된 문자열을 리턴 한다.
RPAD			문자열을 지정된 숫자만큼의 크기로 설정하고, 지정한 문자를 오른쪽부터 채워서 생성된 문자열을 리턴 한다.
RTRIM			왼쪽부터 지정한 문자를 잘라내고 남은 문자를 리턴한다.
LTRIM			오른쪽부터 지정한 문자를 잘라내고 남은 문자를 리턴한다.
TRIM			왼쪽/오른쪽/양쪽부터 지정한 문자를 잘라내고 남은 문자를 리턴한다.
SUBSTR			지정한 위치에서 지정한 길이만큼 문자를 잘라내어 리턴한다.
SUBSTRB			지정한 위치에서 지정한 바이트만큼 문자를 잘라내어 리턴한다.
LOWER			전달받은 문자/문자열을 소문자로 변환하여 리턴한다.
UPPER			전달받은 문자/문자열을 대문자로 변환하여 리턴한다.
INITCAP			전달받은 문자/문자열의 첫 글자를 대문자로, 나머지 글자는 소문자로 변환하여 리턴한다.
CONCAT			인자로 전달받은 두 개의 문자/문자열을 합쳐서 리턴한다.
REPLACE			전달받은 문자열중에 지정한 문자를 인자로 전달받은 문자로 변환하여 리턴한다.

Oracle - Function

문자 처리 함수 - LENGTH

주어진 컬럼 값/문자열의 길이(문자 개수)를 반환하는 함수

작성법	리턴 값 타입	파라미터
LENGTH(CHAR STRING)	NUMBER	CHARACTER 타입의 컬럼 또는 임의의 문자열

```
SELECT EMP_NAME,  
       LENGTH(EMP_NAME),  
       EMAIL,  
       LENGTH(EMAIL)  
FROM EMPLOYEE;
```

EMP_NAME	LENGTH(EMP_NAME)	EMAIL	LENGTH(EMAIL)
1 선동일	3	sun_di@kh.or.kr	15
2 송종기	3	song_jk@kh.or.kr	16
3 노웅철	3	no_hc@kh.or.kr	14
4 송은희	3	song_eh@kh.or.kr	16
5 유재식	3	yoo_js@kh.or.kr	15
6 정중하	3	jung_jh@kh.or.kr	16
7 박나라	3	pack_nr@kh.or.kr	16
8 하미유	3	ha_iy@kh.or.kr	14
9 김해솔	3	kim_hs@kh.or.kr	15
10 심봉선	3	sim_bs@kh.or.kr	15
11 윤은혜	3	youn_eh@kh.or.kr	16
12 저형도	3	jun_hd@kh.or.kr	15

Oracle - Function

문자 처리 함수 - LENGTHB

주어진 컬럼 값/문자열의 길이(BYTE)를 반환하는 함수

작성법	리턴 값 타입	파라미터
LENGTHB(CHAR STRING)	NUMBER	CHARACTER 타입의 컬럼 또는 임의의 문자열

```
SELECT EMP_NAME,  
       LENGTHB(EMP_NAME),  
       EMAIL,  
       LENGTHB(EMAIL)  
FROM EMPLOYEE;
```

EMP_NAME	LENGTHB(EMP_NAME)	EMAIL	LENGTHB(EMAIL)
1 선동일		9sun_di@kh.or.kr	15
2 송중기		9song_jk@kh.or.kr	16
3 노웅철		9no_hc@kh.or.kr	14
4 송은희		9song_eh@kh.or.kr	16
5 유재식		9yoo_js@kh.or.kr	15
6 정중하		9jung_jh@kh.or.kr	16
7 박나라		9pack_nr@kh.or.kr	16
8 하미유		9ha_iy@kh.or.kr	14
9 김해솔		9kim_hs@kh.or.kr	15
10 심봉선		9sim_bs@kh.or.kr	15
11 윤은혜		9youn_eh@kh.or.kr	16
12 전형돈		9jun_hd@kh.or.kr	15
13 장쯔위		9zhang_zw@kh.or.kr	16

문자 처리 함수 - INSTR

찾는 문자(열)이 지정한 위치부터 지정한 회수만큼 나타난 시작 위치를 반환

작성법	리턴 값 타입
<code>INSTR(String, STR, [POSITION],[OCCURRENCE]])</code>	NUMBER

파라미터	설명
STRING	문자 타입 컬럼 또는 문자열
STR	찾으려는 문자(열)
POSITION	찾을 위치 시작 값(기본값 1) POSITION > 0 : STRING의 시작부터 끝 방향으로 찾음 POSITION < 0 : STRING의 끝부터 시작 방향으로 찾음
OCCURRENCE	SUBSTRING이 반복될 때의 지정하는 빈도(기본값 1), 음수 사용 불가

문자 처리 함수 - INSTR

찾는 문자(열)이 지정한 위치부터 지정한 회수만큼 나타난 시작 위치를 반환

작성법	리턴 값 타입
<code>INSTR(String, STR, [POSITION],[OCCURRENCE])</code>	NUMBER

파라미터	설명
STRING	문자 타입 컬럼 또는 문자열 (비교할 대상)
STR	찾으려는 문자(열) (비교하고자하는 값)
POSITION	찾을 위치 시작 값(기본값 1) (비교를 시작할 위치) POSITION > 0 : STRING의 시작부터 끝 방향으로 찾음 POSITION < 0 : STRING의 끝부터 시작 방향으로 찾음
OCCURRENCE	SUBSTRING이 반복될 때의 지정하는 빈도(기본값 1), 음수 사용 불가 (검색된 결과의 순번)

Oracle - Function

문자 처리 함수 - INSTR

[EMAIL 컬럼의 문자열 중 “@”의 위치를 구하시오]

```
SELECT EMAIL,  
       INSTR( EMAIL, '@' , -1, 1 ) 위치  
FROM EMPLOYEE;
```

	EMAIL	←	위치
1	sun_di@kh.or.kr		7
2	song_jk@kh.or.kr		8
3	no_hc@kh.or.kr		6
4	song_eh@kh.or.kr		8
5	yoo_js@kh.or.kr		7
6	jung_jh@kh.or.kr		8
7	pack_nr@kh.or.kr		8
8	ha_iy@kh.or.kr		6
9	kim_hs@kh.or.kr		7
10	sim_bs@kh.or.kr		7
11	youn_eh@kh.or.kr		8
12	jun_hd@kh.or.kr		7

문자 처리 함수 - LPAD/RPAD

주어진 컬럼 문자열에 임의의 문자열을 왼쪽/오른쪽에 덧붙여 길이 N의 문자열을 반환하는 함수

작성법	리턴 값 타입
LPAD(String, N, [STR]) / RPAD(String, N, [STR])	CHARACTER

파라미터	설명
STRING	문자 타입 컬럼 또는 문자열
N	반환할 문자(열)의 길이(바이트) 원래 STRING 길이보다 작다면 N만큼 잘라서 표시한다.
STR	덧붙이려는 문자(열), 생략 시 공백문자임

Oracle - Function

문자 처리 함수 - LPAD/RPAD

```
SELECT LPAD(EMAIL, 20, '#')  
FROM EMPLOYEE;
```

	LPAD(EMAIL,20,'#')
1	#####sun_di@kh.or.kr
2	#####song_jk@kh.or.kr
3	#####no_hc@kh.or.kr
4	#####song_eh@kh.or.kr
5	#####yoo_js@kh.or.kr
6	#####jung_jh@kh.or.kr
7	#####pack_nr@kh.or.kr
8	#####ha_iy@kh.or.kr
9	#####kim_hs@kh.or.kr
10	#####sim_bs@kh.or.kr
11	#####youn_eh@kh.or.kr
12	#####jun_hd@kh.or.kr
13	#####jang_zw@kh.or.kr
14	#####ha_dh@kh.or.kr
15	#####bang_ms@kh.or.kr

```
SELECT RPAD(EMAIL, 20, '#')  
FROM EMPLOYEE;
```

	RPAD(EMAIL,20,'#')
1	sun_di@kh.or.kr#####
2	song_jk@kh.or.kr#####
3	no_hc@kh.or.kr#####
4	song_eh@kh.or.kr#####
5	yoo_js@kh.or.kr#####
6	jung_jh@kh.or.kr#####
7	pack_nr@kh.or.kr#####
8	ha_iy@kh.or.kr#####
9	kim_hs@kh.or.kr#####
10	sim_bs@kh.or.kr#####
11	youn_eh@kh.or.kr#####
12	jun_hd@kh.or.kr#####
13	jang_zw@kh.or.kr#####
14	ha_dh@kh.or.kr#####
15	bang_ms@kh.or.kr#####

문자 처리 함수 - LTRIM/RTRIM

주어진 컬럼이나 문자열의 왼쪽 혹은 오른쪽에서 지정한 STR에 포함된 모든 문자를 제거한 나머지를 반환한다.

작성법	리턴 값 타입
LTRIM(String, STR) / RTRIM(String, STR)	CHARACTER

파라미터	설명
STRING	문자 타입 컬럼 또는 문자열
STR	제거하려는 문자(열), 생략하면 공백문자

문자 처리 함수 - LTRIM/RTRIM

[문자 처리 함수 - LTRIM]

수행 문장	결과
SELECT LTRIM(' KH') FROM DUAL;	KH
SELECT LTRIM(' KH', ' ') FROM DUAL;	KH
SELECT LTRIM('000123456', '0') FROM DUAL;	123456
SELECT LTRIM('123123KH', '123') FROM DUAL;	KH
SELECT LTRIM('123123KH123', '123') FROM DUAL;	KH123
SELECT LTRIM('ACABACCKH', 'ABC') FROM DUAL;	KH
SELECT LTRIM('5782KH', '0123456789') FROM DUAL;	KH

[문자 처리 함수 - RTRIM]

수행 문장	결과
SELECT RTRIM('KH ') FROM DUAL;	KH
SELECT RTRIM('KH ', ' ') FROM DUAL;	KH
SELECT RTRIM('123456000', '0') FROM DUAL;	123456
SELECT RTRIM('KH123123', '123') FROM DUAL;	KH
SELECT RTRIM('123KH123123', '123') FROM DUAL;	123KH
SELECT RTRIM('KHACABACC', 'ABC') FROM DUAL;	KH
SELECT RTRIM('KH5782', '0123456789') FROM DUAL;	KH

Oracle - Function

문자 처리 함수 - TRIM

주어진 컬럼이나 문자열의 앞/뒤/양쪽에 있는 지정한 문자를 제거한 나머지를 반환한다.

작성법	리턴 값 타입
TRIM(STRING) TRIM(CHAR FROM STRING) TRIM(LEADING TRAILING BOTH [CHAR] FROM STRING)	CHARACTER

파라미터	설명
STRING	문자 타입 컬럼 또는 문자열
CHAR	제거하려는 문자, 생략하면 공백문자
LEADING	TRIM할 CHAR의 위치를 지정한다. 앞(LEADING) / 뒤(TRAILING) / 양쪽(BOTH) 지정 가능 (기본값 양쪽)

Oracle - Function

문자 처리 함수 - TRIM

수행 문장	결과
SELECT TRIM(' KH ') FROM DUAL;	KH
SELECT TRIM('Z' FROM 'ZZZKHZZZ') FROM DUAL;	KH
SELECT TRIM(LEADING 'Z' FROM 'ZZZ123456') FROM DUAL;	123456
SELECT TRIM(TRAILING '123' FROM 'KH123123') FROM DUAL;	KH
SELECT TRIM(BOTH '123' FROM '123KH123123') FROM DUAL;	KH
SELECT TRIM(LEADING '123' FROM '123KH123123') FROM DUAL;	KH123123

문자 처리 함수 - SUBSTR

컬럼이나 문자열에서 지정한 위치부터 지정한 개수의 문자열을 잘라내어 리턴하는 함수이다.

작성법	리턴 값 타입
SUBSTR(STRING, POSITION, [LENGTH])	CHARACTER

파라미터	설명
STRING	문자 타입 컬럼 또는 문자열
POSITION	문자열을 잘라낼 위치이다. 양수이면 시작방향에서 지정한 수 만큼 위치를 의미하고, 음수이면 끝 방향에서 지정한 수 만큼의 위치를 의미한다.
LENGTH	반환할 문자의 개수를 의미한다. (생략시 문자열의 끝까지를 의미하고, 음수이면 NULL을 리턴함)

문자 처리 함수 - SUBSTR

수행 문장	결과
SELECT SUBSTR('SHOWMETHEMONEY', 5, 2) FROM DUAL;	ME
SELECT SUBSTR('SHOWMETHEMONEY', 7) FROM DUAL;	THEMONEY
SELECT SUBSTR('SHOWMETHEMONEY', 1, 6) FROM DUAL;	SHOWME
SELECT SUBSTR('SHOWMETHEMONEY', -8, 3) FROM DUAL;	THE
SELECT SUBSTR('SHOWMETHEMONEY', -10, 2) FROM DUAL;	ME
SELECT SUBSTR('쇼우 미 더 머니', 2, 5) FROM DUAL;	우 미 더

Oracle - Function

문자 처리 함수 - LOWER/UPPER/INITCAP

컬럼의 문자 혹은 문자열을 소문자/대문자/첫 글자만 대문자로 변환하여 리턴하는 함수이다.

작성법	리턴 값 타입
LOWER(String) / UPPER(String) / INITCAP(String)	CHARACTER

파라미터	설명
String	문자 타입 컬럼 또는 문자열

수행 문장	결과
SELECT LOWER('Welcome To My World') from dual;	welcome to my world
SELECT UPPER('Welcome To My World') from dual;	WELCOME TO MY WORLD
SELECT INITCAP('welcome to my world')from dual;	Welcome To My World

Oracle - Function

문자 처리 함수 - CONCAT

컬럼의 문자 혹은 문자열을 두 개 전달 받아 하나로 합친 후 리턴하는 함수이다.

작성법	리턴 값 타입
CONCAT(String, String)	CHARACTER

파라미터	설명
String	문자 타입 컬럼 또는 문자열

수행 문장	결과
SELECT CONCAT('가나다라', 'ABCD') FROM DUAL;	가나다라ABCD
SELECT '가나다라' 'ABCD' FROM DUAL	가나다라ABCD

Oracle - Function

문자 처리 함수 - REPLACE

문자 타입 컬럼 혹은 문자열 중 특정 문자(열)를 다른 문자(열)로 치환하는 함수.

작성법	리턴 값 타입
REPLACE(String, STR1, STR2)	CHARACTER

파라미터	설명
String	문자 타입 컬럼 또는 문자열
STR1	변경하려고 하는 문자 혹은 문자열
STR2	변경하고자 하는 문자 혹은 문자열

수행 문장	결과
SELECT REPLACE('서울시 강남구 역삼동', '역삼동', '삼성동') FROM DUAL;	서울시 강남구 삼성동
SELECT REPLACE('sun_di@kh.or.kr' '@kh.or.kr', '@gmail.com') FROM DUAL;	sun_di@gmail.com

Oracle - Function

숫자 처리 함수

구분	입력 값 타입	리턴 값 타입	설명
ABS	NUMBER	NUMBER	절대값을 구하여 리턴한다.
MOD			입력받은 수를 나눈 나머지 값을 반환한다.
ROUND			특정 자릿수에서 반올림한다.
FLOOR			소수점 아래를 잘라내고 리턴한다.(버림)
TRUNC			특정 자릿수에서 잘라내고 리턴한다.(버림)
CEIL			지정한 자릿수에서 올림하여 리턴한다.

ABS - 인자로 전달받은 숫자의 절대값을 구하는 함수이다.

작성법	리턴 값 타입
ABS(NUMBER)	NUMBER

파라미터	설명
NUMBER	숫자 혹은 숫자 데이터 컬럼

Oracle - Function

숫자 처리 함수 - MOD

인자로 전달받은 숫자를 나누어 나머지를 구하는 함수이다.

작성법	리턴 값 타입
MOD(NUMBER, DIVISION)	NUMBER

파라미터	설명
NUMBER	숫자 혹은 숫자 데이터 컬럼
DIVISION	나눌 수 혹은 나눌 숫자 데이터 컬럼

숫자 처리 함수 - ROUND

인자로 전달 받은 숫자 혹은 컬럼에서 지정한 위치부터 반올림하여 값을 리턴하는 함수이다.

작성법	리턴 값 타입
ROUND(NUMBER) ROUND(NUMBER, POSITION)	NUMBER

파라미터	설명
NUMBER	숫자 혹은 숫자 데이터 컬럼
POSITION	반올림 할 위치 자리

Oracle - Function

숫자 처리 함수 - FLOOR

인자로 전달받은 숫자 혹은 컬럼의 소수점 자리의 수를 버리는 함수이다.

작성법	리턴 값 타입
FLOOR(NUMBER)	NUMBER

파라미터	설명
NUMBER	숫자 혹은 숫자 데이터 컬럼

Oracle - Function

숫자 처리 함수 - TRUNC

인자로 전달받은 숫자 혹은 컬럼의 지정한 위치부터 소수점 자리의 수를 버리는 함수이다.

작성법	리턴 값 타입
TRUNC(NUMBER, POSITION)	NUMBER

파라미터	설명
NUMBER	숫자 혹은 숫자 데이터 컬럼
POSITION	버림 할 위치 자리

Oracle - Function

숫자 처리 함수 - CEIL

인자로 전달받은 숫자 혹은 컬럼을 올림 계산 하여 나온 값을 리턴하는 함수이다.

작성법	리턴 값 타입
CEIL(NUMBER)	NUMBER

파라미터	설명
NUMBER	숫자 혹은 숫자 데이터 컬럼

Oracle - Function

숫자 처리 함수 - ROUND/FLOOR/TRUNC/CEIL

수행 문장	결과
SELECT ROUND(123.456) FROM DUAL;	123
SELECT ROUND(123.456, 1) FROM DUAL;	123.5
SELECT ROUND(123.456, 2) FROM DUAL;	123.46
SELECT ROUND(123.456, -1) FROM DUAL;	120
SELECT FLOOR(123.456) FROM DUAL;	123
SELECT TRUNC(123.456) FROM DUAL;	123
SELECT TRUNC(123.456, 1) FROM DUAL;	123.4
SELECT TRUNC(123.456, 2) FROM DUAL;	123.45
SELECT TRUNC(123.456, -1) FROM DUAL;	120
SELECT CEIL(123.456) FROM DUAL;	124

날짜 처리 함수

구분	입력 값 타입	리턴 값 타입	설명
SYSDATE		DATE	시스템에 저장된 현재 날짜를 반환한다.
MONTHS_BETWEEN	DATE	NUMBER	두 날짜를 전달받아 몇 개월 차이인지 계산하여 반환한다.
ADD_MONTHS	DATE	DATE	특정 날짜에 개월 수를 더한다.
NEXT_DAY			특정 날짜에서 최초로 다가오는 인자로 받은 요일의 날짜를 반환한다.
LAST_DAY			해당 달의 마지막 날짜를 반환한다.
EXTRACT			년, 월, 일 정보를 추출하여 반환한다.

SYSDATE - 시스템에 저장되어 있는 현재 날짜를 반환하는 함수이다.

작성법	리턴 값 타입
SYSDATE	DATE

날짜 처리 함수 - MONTHS_BETWEEN

인자로 날짜 두 개를 전달받아, 개월 수의 차이를 숫자 데이터형으로 리턴하는 함수이다.

작성법	리턴 값 타입
MONTHS_BETWEEN(DATE1, DATE2)	DATE

파라미터	설명
DATE1	기준이 되는 날짜를 입력한다.
DATE2	개월 수를 구하려는 날짜를 입력한다.

KH KH정보교육원

[illegible]

Oracle - Function

날짜 처리 함수 - ADD_MONTHS

인자로 전달받은 날짜에 인자로 전달받은 숫자만큼 개월 수를 더하여 특정 날짜를 리턴하는 함수이다.

작성법	리턴 값 타입
ADD_MONTHS(DATE, NUMBER)	DATE

파라미터	설명
DATE	기준이 되는 날짜를 입력한다.
NUMBER	더하려는 개월 수를 입력한다.

Oracle - Function

날짜 처리 함수 - ADD_MONTHS

[EMPLOYEE 테이블에서 사원의 이름, 입사일, 입사 후 6개월이 된 날짜를 조회하세요.]

```
SELECT EMP_NAME,  
       HIRE_DATE,  
       ADD_MONTHS(HIRE_DATE, 6)  
FROM EMPLOYEE;
```

	EMP_NAME	HIRE_DATE	ADD_MONTHS(HIRE_DATE,6)
1	선동일	90/02/06	90/08/06
2	송종기	01/09/01	02/03/01
3	노웅철	01/01/01	01/07/01
4	송은희	96/05/03	96/11/03
5	유재식	00/12/29	01/06/29
6	정중하	99/09/09	00/03/09
7	박나라	08/04/02	08/10/02
8	하미유	94/07/07	95/01/07
9	김해술	04/04/30	04/10/31
10	심봉선	11/11/11	12/05/11
11	윤은해	01/02/03	01/08/03
12	전형돈	12/12/12	13/06/12
13	장프위	15/06/17	15/12/17
14	하동운	99/12/31	00/06/30
15	방명수	10/04/04	10/10/04
16	대북혼	17/06/19	17/12/19
17	차태연	13/03/01	13/09/01

Oracle - Function

날짜 처리 함수 - NEXT_DAY

인자로 전달받은 날짜에 인자로 전달받은 요일의 가장 가까운 날짜를 구하여 리턴하는 함수이다.

작성법	리턴 값 타입
NEXT_DAY (DATE, STRING [OR NUMBER])	DATE

파라미터	설명
DATE	기준이 되는 날짜를 입력한다.
STRING [OR NUMBER]	구하려는 요일을 입력한다.(숫자의 경우 1 = 일요일, ... , 7 = 토요일)

Oracle - Function

날짜 처리 함수 - NEXT_DAY

```
SELECT SYSDATE,  
       NEXT_DAY(SYSDATE, '월요일')  
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE, '월요일')
1	17/09/11	17/09/18

```
SELECT SYSDATE,  
       NEXT_DAY(SYSDATE, 2)  
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE, '월요일')
1	17/09/11	17/09/18

```
SELECT SYSDATE,  
       NEXT_DAY(SYSDATE, '월')  
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE, '월요일')
1	17/09/11	17/09/18

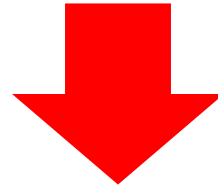
```
SELECT SYSDATE,  
       NEXT_DAY(SYSDATE, 'MONDAY')  
FROM DUAL;
```

ORA-01846: not a valid day of the week
01846, 00000 - "not a valid day of the week"
*Cause:
*Action:

Oracle - Function

날짜 처리 함수 - NEXT_DAY

```
ALTER SESSION SET NLS_LANGUAGE = AMERICAN;
```



```
SELECT SYSDATE,  
       NEXT_DAY(SYSDATE, 'MONDAY')  
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE,'MONDAY')
1	17/09/11	17/09/18

Oracle - Function

날짜 처리 함수 - LAST_DAY

인자로 전달받은 날짜가 속한 달의 마지막 날짜를 구하여 리턴한다.

작성법	리턴 값 타입
LAST_DAY (DATE)	DATE

파라미터	설명
DATE	기준이 되는 날짜를 입력한다.

[EMPLOYEE 테이블에서 사원의 이름, 입사일, 입사일의 마지막날을 조회하세요.]

```
SELECT EMP_NAME,  
       HIRE_DATE,  
       LAST_DAY(HIRE_DATE)  
FROM EMPLOYEE;
```

	EMP_NAME	HIRE_DATE	LAST_DAY(HIRE_DATE)
1	선동일	90/02/06	90/02/28
2	송종기	01/09/01	01/09/30
3	노웅철	01/01/01	01/01/31
4	송은희	96/05/03	96/05/31
5	유재식	00/12/29	00/12/31
6	정중하	99/09/09	99/09/30
7	박나라	08/04/02	08/04/30
8	하미유	94/07/07	94/07/31
9	김해술	04/04/30	04/04/30

Oracle - Function

날짜 처리 함수 - EXTRACT

년, 월, 일 정보를 추출하여 리턴 한다.

작성법	리턴 값 타입
EXTRACT(YEAR FROM <u>DATE</u>) EXTRACT(MONTH FROM <u>DATE</u>) EXTRACT(DAY FROM <u>DATE</u>)	DATE

파라미터	설명
DATE	기준이 되는 날짜를 입력한다.

[EMPLOYEE테이블에서 사원 이름, 입사 년, 입사 월, 입사 일을 조회하세요]

```
SELECT EMP_NAME,  
       EXTRACT(YEAR FROM HIRE_DATE),  
       EXTRACT(MONTH FROM HIRE_DATE),  
       EXTRACT(DAY FROM HIRE_DATE)  
FROM EMPLOYEE
```

EMP_NAME	EXTRACT(YEARFROMHIRE_DATE)	EXTRACT(MONTHFROMHIRE_DATE)	EXTRACT(DAYFROMHIRE_DATE)
1 선동일	1990	2	6
2 송종기	2001	9	1
3 노홍철	2001	1	1
4 송은희	1996	5	3
5 유재석	2000	12	29
6 정종하	1999	9	9
7 박나라	2008	4	2
8 하미유	1994	7	7
9 김해솔	2004	4	30
10 심봉선	2011	11	11
11 윤은혜	2001	2	3
12 전형돈	2012	12	12
13 장프위	2015	6	17

Oracle - Function

형변환 함수

구분	입력 값 타입	리턴 값 타입	설명
TO_CHAR	DATE NUMBER	CHARACTER	날짜형 혹은 숫자형을 문자형으로 변환한다.
TO_DATE	CHARACTER	DATE	문자형을 날짜형으로 변환한다.
TO_NUMBER	CHARACTER	NUMBER	문자형을 숫자형으로 변환한다.



Oracle - Function

형변환 함수 - TO_CHAR

```
SELECT EMP_NAME,  
       TO_CHAR(HIRE_DATE, 'YYYY-MM-DD'),  
       TO_CHAR(HIRE_DATE, 'YY/MON, DAY, DY')  
FROM EMPLOYEE;
```

	EMP_NAME	TO_CHAR(HIRE_DATE, 'YYYY-MM-DD')	TO_CHAR(HIRE_DATE, 'YY/MON, DAY, DY')
1	선동일	1990-02-06	90/2월 , 화요일, 화
2	송종기	2001-09-01	01/9월 , 토요일, 토
3	노웅철	2001-01-01	01/1월 , 월요일, 월
4	송은희	1996-05-03	96/5월 , 금요일, 금
5	유재식	2000-12-29	00/12월, 금요일, 금
6	정중하	1999-09-09	99/9월 , 목요일, 목
7	박나라	2008-04-02	08/4월 , 수요일, 수
8	하미유	1994-07-07	94/7월 , 목요일, 목
9	김해술	2004-04-30	04/4월 , 금요일, 금
10	심봉선	2011-11-11	11/11월, 금요일, 금
11	윤은해	2001-02-03	01/2월 , 토요일, 토
12	전형돈	2012-12-12	12/12월, 수요일, 수
13	장프위	2015-06-17	15/6월 , 수요일, 수
14	하동운	1999-12-31	99/12월, 금요일, 금
15	방명수	2010-04-04	10/4월 , 일요일, 일
16	대북훈	2017-06-19	17/6월 , 월요일, 월

Oracle - Function

형변환 함수 - TO_CHAR

```
SELECT EMP_NAME,  
       TO_CHAR(SALARY, 'L999,999,999'),  
       TO_CHAR(SALARY, '000,000,000')  
FROM EMPLOYEE;
```

	EMP_NAME	TO_CHAR(SALARY, 'L999,999,999')	TO_CHAR(SALARY, '000,000,000')
1	선동일	₩8,000,000	008,000,000
2	송종기	₩6,000,000	006,000,000
3	노웅철	₩3,700,000	003,700,000
4	송은희	₩2,800,000	002,800,000
5	유재식	₩3,400,000	003,400,000
6	정중하	₩3,900,000	003,900,000
7	박나라	₩1,800,000	001,800,000
8	하미유	₩2,200,000	002,200,000
9	김해술	₩2,500,000	002,500,000
10	심봉선	₩3,500,000	003,500,000
11	윤은해	₩2,000,000	002,000,000
12	전형돈	₩2,000,000	002,000,000
13	장쯔위	₩2,550,000	002,550,000
14	하동운	₩2,320,000	002,320,000
15	반면스	₩1,380,000	001,380,000

Oracle - Function

형변환 함수 - TO_DATE

숫자 혹은 문자형 데이터를 날짜형 데이터로 변환하여 리턴 한다.

작성법	리턴 값 타입
TO_DATE(Character, [Format]) TO_DATE(Number, [Format])	DATE NUMBER

파라미터	설명
Character	날짜형으로 변환하려는 문자형 데이터
Number	날짜형으로 변환하려는 숫자형 데이터
Format	날짜형으로 변환 시 입력 포맷 지정

Oracle - Function

형변환 함수 - TO_DATE

[EMPLOYEE테이블에서 2000년도 이후에 입사한 사원의 사번, 이름, 입사일을 조회하세요]

```
SELECT EMP_NO,  
       EMP_NAME,  
       HIRE_DATE  
FROM EMPLOYEE  
WHERE HIRE_DATE > TO_DATE(20000101, 'YYYYMMDD');
```

	EMP_NO	EMP_NAME	HIRE_DATE
1	631156-1548654	송종기	01/09/01
2	861015-1356452	노용철	01/01/01
3	660508-1342154	유재식	00/12/29
4	630709-2054321	박나라	08/04/02
5	870927-1313564	김해술	04/04/30
6	750206-1325546	심봉선	11/11/11
7	650505-2356985	윤은해	01/02/03
8	830807-1121321	전형돈	12/12/12
9	780923-2234542	장프위	15/06/17
10	856795-1313513	방명수	10/04/04
11	881130-1050911	대북혼	17/06/19
12	770808-1364897	차태연	13/03/01
13	770808-2665412	전지연	07/03/20
14	870427-2232123	미오리	16/11/28
15	770823-1113111	이중석	14/09/18

Oracle - Function

형변환 함수 - TO_NUMBER

날짜 혹은 문자형 데이터를 숫자형 데이터로 변환하여 리턴 한다.

작성법	리턴 값 타입
TO_NUMBER (CHARACTER, [FORMAT])	DATE NUMBER

파라미터	설명
CHARACTER	숫자형으로 변환하려는 문자형 데이터
FORMAT	숫자형으로 변환 시 입력 포맷 지정

```
SELECT TO_NUMBER('1,000,000', '99,999,999') -  
       TO_NUMBER('550,000', '999,999')  
FROM DUAL;
```

	TO_NUMBER('1,000,000', '99,999,999')-TO_NUMBER('550,000', '999,999')
1	450000

Oracle - Function

NULL 처리 함수 - NVL

NULL로 되어 있는 컬럼의 값을 지정한 숫자 혹은 문자로 변경하여 리턴한다.

작성법	리턴 값 타입
NVL(P1, P2)	NUMBER CHARACTER

파라미터	설명
P1	NULL데이터를 처리할 컬럼명 혹은 값
P2	NULL값을 변경하고자 하는 값

Oracle - Function

NULL 처리 함수 - NVL

```
SELECT EMP_NO,  
       EMP_NAME,  
       SALARY,  
       NVL(BONUS, 0),  
       (SALARY * 12 + (SALARY * 12 * NVL(BONUS, 0)))  
FROM EMPLOYEE;
```

	EMP_NO	EMP_NAME	SALARY	NVL(BONUS,0)	((SALARY*12)+(SALARY*12)*NVL(BONUS,0))
1	621235-1985634	선동일	8000000	0.3	124800000
2	631156-1548654	송종기	6000000	0	72000000
3	861015-1356452	노용철	3700000	0	44400000
4	631010-2653546	송은희	2800000	0	33600000
5	660508-1342154	유재식	3400000	0.2	48960000
6	770102-1357951	정중하	3900000	0	46800000
7	630709-2054321	박나라	1800000	0	21600000
8	690402-2040612	하미유	2200000	0.1	29040000
9	870927-1313564	김해솔	2500000	0	30000000
10	750206-1325546	심봉선	3500000	0.15	48300000
11	650505-2356985	윤은해	2000000	0	24000000
12	830807-1121321	전형돈	2000000	0	24000000
13	780923-2234542	장프위	2550000	0.25	38250000
14	621111-1785463	하동운	2320000	0.1	30624000
15	856795-1313513	방명수	1380000	0	16560000
16	881130-1050911	대복훈	3760000	0	45120000

선택 함수 - DECODE

CASE 식 또는 IF-THEN-ELSE문의 작업을 수행하여 조건부 조회를 편리하게 수행함.

작성법	리턴 값 타입
DECODE(컬럼 표현식, 조건1, 결과1, 조건2, 결과2, 조건3, 결과3, ...)	결과

파라미터	설명
표현식	값에 따라 선택을 다르게 할 컬럼 혹은 값 입력
조건	해당 값이 참인지 거짓인지 여부를 판단한다.
결과	해당 조건과 일치하는 경우 결과를 리턴한다.
DEFAULT	모든 조건이 불일치 시 리턴할 값

Oracle - Function

선택 함수 - DECODE

```
SELECT EMP_ID,  
       EMP_NAME,  
       EMP_NO,  
       DECODE(SUBSTR(EMP_NO, 8, 1), '1', '남', '2', '여') AS 성별  
FROM EMPLOYEE;
```

	EMP_ID	EMP_NAME	EMP_NO	성별
1	200	선동일	621235-1985634	남
2	201	송종기	631156-1548654	남
3	202	노용철	861015-1356452	남
4	203	송은희	631010-2653546	여
5	204	유재식	660508-1342154	남
6	205	정중하	770102-1357951	남
7	206	박나라	630709-2054321	여
8	207	하미유	690402-2040612	여
9	208	김해솔	870927-1313564	남
10	209	심봉선	750206-1325546	남
11	210	윤은혜	650505-2356985	여
12	211	전형돈	830807-1121321	남
13	212	장프위	780923-2234542	여
14	213	하동운	621111-1785463	남
15	214	방명수	856795-1313513	남

Oracle - Function

선택 함수 - CASE

여러 가지 경우에 선택을 할 수 있는 기능을 제공한다.(범위값 가능)

작성법	리턴 값 타입
<pre>CASE WHEN 조건1 THEN 결과1 WHEN 조건2 THEN 결과2 WHEN 조건3 THEN 결과3 ELSE 결과N END</pre>	결과

파라미터	설명
조건	해당 값이 참인지 거짓인지 여부를 판단한다.
결과	해당 조건과 일치하는 경우 결과를 리턴한다.
DEFAULT	모든 조건이 불일치 시 리턴할 값

Oracle - Function

선택 함수 - CASE

```
SELECT EMP_ID,  
       EMP_NAME,  
       EMP_NO,  
       CASE WHEN SUBSTR(EMP_NO, 8, 1) = 1 THEN '남'  
            ELSE '여'  
       END AS 성별  
FROM EMPLOYEE;
```

	EMP_ID	EMP_NAME	EMP_NO	성별
1	200	선동일	621235-1985634	남
2	201	송종기	631156-1548654	남
3	202	노용철	861015-1356452	남
4	203	송은희	631010-2653546	여
5	204	유재식	660508-1342154	남
6	205	정종하	770102-1357951	남
7	206	박나라	630709-2054321	여
8	207	하미유	690402-2040612	여
9	208	김해술	870927-1313564	남
10	209	심봉선	750206-1325546	남
11	210	윤은해	650505-2356985	여
12	211	전형돈	830807-1121321	남
13	212	장프위	780923-2234542	여
14	213	하동운	621111-1785463	남

Oracle - Function

그룹함수 - SUM

해당 컬럼 값들의 총 합을 구하는 함수이다.

[EMPLOYEE테이블에서 남자 사원의 급여 총 합을 계산하세요]

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE SUBSTR(EMP_NO, 8, 1) = 1;
```

	SUM(SALARY)
1	49760000

[EMPLOYEE테이블에서 부서코드가 D5인 직원의 보너스 포함 연봉을 계산하세요]

```
SELECT SUM((SALARY + (SALARY * NVL(BONUS, 0)) * 12))
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5';
```

	SUM((SALARY+(SALARY*NVL(BONUS,0))*12))
1	24700000

Oracle - Function

그룹함수 - AVG

해당 컬럼 값들의 평균을 구하는 함수이다.

[EMPLOYEE테이블에서 전 사원의 보너스 평균을 소수 둘째자리에서 반올림하여 구하세요]

```
SELECT ROUND(AVG(BONUS), 2)
FROM EMPLOYEE;
```

ROUND(AVG(BONUS),2)	
1	0.22

** NULL값을 가진 행을 평균 계산에서 제외시킴



[EMPLOYEE테이블에서 전 사원의 보너스 평균을 소수 둘째자리에서 반올림하여 구하세요]

```
SELECT ROUND(AVG(NVL(BONUS, 0)), 2)
FROM EMPLOYEE;
```

ROUND(AVG(NVL(BONUS,0)),2)	
1	0.08

** NULL값을 가진 행을 평균 계산에서 포함시킴

Oracle - Function

그룹함수 - COUNT

해당 컬럼 값들의 총 합을 구하는 함수이다.

[EMPLOYEE테이블에서 남자 사원의 급여 총 합을 계산하세요]

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE SUBSTR(EMP_NO, 8, 1) = 1;
```

	SUM(SALARY)
1	49760000

[EMPLOYEE테이블에서 부서코드가 D5인 직원의 보너스 포함 연봉을 계산하세요]

```
SELECT SUM((SALARY + (SALARY * NVL(BONUS, 0)) * 12))
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5';
```

	SUM((SALARY+(SALARY*NVL(BONUS,0))*12))
1	24700000

Oracle - Function

그룹함수 - MAX/MIN

그룹의 최대값과 최소값을 구하여 리턴하는 함수이다.

[EMPLOYEE테이블에서 사원 중 가장 높은 급여와 가장 낮은 급여를 조회하세요]

```
SELECT MAX(SALARY),  
       MIN(SALARY)  
FROM EMPLOYEE;
```

	MAX(SALARY)	MIN(SALARY)
1	8000000	1380000

[EMPLOYEE테이블에서 가장 오래된 입사일과 가장 최근 입사일을 조회하세요]

```
SELECT MAX(SALARY),  
       MIN(SALARY)  
FROM EMPLOYEE;
```

	MAX(HIRE_DATE)	MIN(HIRE_DATE)
1	17/06/19	90/02/06

Oracle GROUP BY & HAVING

Oracle SQL – GROUP BY & HAVING

GROUP BY절

그룹함수는 단 한 개의 결과값만 산출하기 때문에, 그룹함수를 이용하여 여러 개의 결과값을 산출하기 위해서는 그룹함수가 적용될 그룹의 기준을 GROUP BY절에 기술하여 사용해야 한다.

```
SELECT DEPT_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE;
```

**** 에러 발생**



DEPT_CODE	SUM_SALARY
D1	SUM(SALARY)
D2	×
D3	×

```
SELECT DEPT_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE  
GROUP BY DEPT_CODE;
```



DEPT_CODE	SUM_SALARY
D1	SUM(SALARY)
D2	SUM(SALARY)
D3	SUM(SALARY)

Oracle SQL – GROUP BY & HAVING

GROUP BY절

[EMPLOYEE 테이블에서 부서코드, 그룹별 급여의 합계, 그룹별 급여의 평균(정수처리), 인원수를 조회하고, 부서코드 순으로 정렬하세요]

```
SELECT DEPT_CODE,  
       SUM(SALARY) AS 합계,  
       AVG(SALARY) AS 평균,  
       COUNT(*) AS 인원수  
FROM EMPLOYEE  
GROUP BY DEPT_CODE  
ORDER BY DEPT_CODE ASC;
```

	DEPT_CODE	합계	평균	인원수
1	D1	7820000	2606666	3
2	D2	6520000	2173333	3
3	D5	15760000	2626666	6
4	D6	10100000	3366666	3
5	D8	6986240	2328746	3
6	D9	17700000	5900000	3
7	(null)	5210000	2605000	2

[EMPLOYEE 테이블에서 부서코드, 보너스를 지급받는 사원 수를 조회하고 부서코드 순으로 정렬하세요]

```
SELECT DEPT_CODE,  
       COUNT(BONUS)  
FROM EMPLOYEE  
WHERE BONUS IS NOT NULL  
GROUP BY DEPT_CODE  
ORDER BY DEPT_CODE ASC;
```

	DEPT_CODE	COUNT(BONUS)
1	D1	2
2	D5	2
3	D6	1
4	D8	2
5	D9	1
6	(null)	1

Oracle SQL – GROUP BY & HAVING

GROUP BY절

[EMPLOYEE 테이블에서 EMP_NO의 8번째 자리가 1이면 ‘남’, 2이면 ‘여’로 결과를 조회하고, 성별별 급여의 평균(정수처리), 급여의 합계, 인원수를 조회한 뒤, 인원수로 내림차순 정렬하세요]

```
SELECT DECODE(SUBSTR(EMP_NO, 8, 1), 1, '남', 2, '여') AS 성별,  
       FLOOR(AVG(SALARY)) AS 평균,  
       SUM(SALARY) AS 합계  
       COUNT(*) AS 인원수  
FROM EMPLOYEE  
GROUP BY DECODE(SUBSTR(EMP_NO, 8, 1), 1, '남', 2, '여')  
ORDER BY COUNT(*) DESC;
```

	♣ 성별	♣ 평균	♣ 합계	♣ 인원수
1	남	3317333	49760000	15
2	여	2542030	20336240	8

Oracle SQL – GROUP BY & HAVING

HAVING절

그룹함수로 값을 구해올 그룹에 대해 조건을 설정할 때는 HAVING절에 기술한다.
(WHERE절은 SELECT에 대한 조건)

```
SELECT DEPT_CODE,  
       FLOOR(AVG(SALARY)) 평균  
FROM EMPLOYEE  
WHERE SALARY > 3000000  
GROUP BY DEPT_CODE  
ORDER BY 1;
```

[급여 3000000원 이상인 직원의 그룹별 평균]

	DEPT_CODE	평균
1	D1	3660000
2	D5	3630000
3	D6	3650000
4	D9	5900000

```
SELECT DEPT_CODE,  
       FLOOR(AVG(SALARY)) 평균  
FROM EMPLOYEE  
GROUP BY DEPT_CODE  
HAVING FLOOR(AVG(SALARY)) > 3000000  
ORDER BY 1;
```

[급여 평균이 3000000원 이상인 그룹에 대한 평균]

	DEPT_CODE	평균
1	D6	3366666
2	D9	5900000

Oracle SQL – GROUP BY & HAVING

ROLLUP과 CUBE

그룹별 산출한 결과값의 집계를 계산하는 함수이다.

```
SELECT JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY ROLLUP(JOB_CODE)
ORDER BY 1;
```

```
SELECT JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY CUBE(JOB_CODE)
ORDER BY 1;
```

	⚡ JOB_CODE	⚡ SUM(SALARY)
1	J1	8000000
2	J2	9700000
3	J3	10800000
4	J4	9320000
5	J5	8460000
6	J6	15746240
7	J7	8070000
8	(null)	70096240

Oracle SQL – GROUP BY & HAVING

ROLLUP

인자로 전달받은 그룹 중에 가장 먼저 지정한 그룹별 합계와 총 합계를 구한다.

```
SELECT DEPT_CODE,  
       JOB_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE  
GROUP BY ROLLUP(DEPT_CODE, JOB_CODE)  
ORDER BY 1;
```

	DEPT_CODE	JOB_CODE	SUM(SALARY)
1	D1	J6	6440000
2	D1	J7	1380000
3	D1	(null)	7820000
4	D2	J4	6520000
5	D2	(null)	6520000
6	D5	J3	3500000
7	D5	J5	8460000
8	D5	J7	3800000
9	D5	(null)	15760000
10	D6	J3	7300000
11	D6	J4	2800000
12	D6	(null)	10100000
13	D8	J6	6986240
14	D8	(null)	6986240
15	D9	J1	8000000
16	D9	J2	9700000
17	D9	(null)	17700000
18	(null)	J6	2320000
19	(null)	J7	2890000
20	(null)	(null)	5210000

Oracle SQL – GROUP BY & HAVING

CUBE

그룹으로 지정된 모든 그룹에 대한 합계와 총 합계를 구한다.

```
SELECT DEPT_CODE,  
       JOB_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE  
GROUP BY CUBE(DEPT_CODE, JOB_CODE)  
ORDER BY 1;
```

	DEPT_CODE	JOB_CODE	SUM(SALARY)
1	D1	J6	6440000
2	D1	J7	1380000
3	D1	(null)	7820000
4	D2	J4	6520000
5	D2	(null)	6520000
6	D5	J3	3500000
7	D5	J5	8460000
8	D5	J7	3800000
9	D5	(null)	15760000
10	D6	J3	7300000
11	D6	J4	2800000
12	D6	(null)	10100000
13	D8	J6	6986240
14	D8	(null)	6986240
15	D9	J1	8000000
16	D9	J2	9700000
17	D9	(null)	17700000
18	(null)	J1	8000000
19	(null)	J2	9700000
20	(null)	J3	10800000

Oracle SQL – GROUP BY & HAVING

GROUPING

ROLLUP이나 CUBE에 의한 집계 산출물이 인자로 전달받은 컬럼 집합의 산출물이면 0을 반환하고, 아니면 1을 반환하는 함수이다.

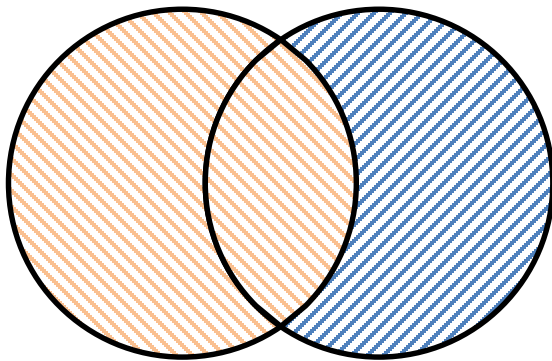
```
SELECT DEPT_CODE,  
       JOB_CODE,  
       SUM(SALARY),  
       CASE WHEN GROUPING(DEPT_CODE) = 0  
             AND GROUPING(JOB_CODE) = 1  
             THEN '부서별합계'  
             WHEN GROUPING(DEPT_CODE) = 1  
             AND GROUPING(JOB_CODE) = 0  
             THEN '직급별합계'  
             WHEN GROUPING(DEPT_CODE) = 1  
             AND GROUPING(JOB_CODE) = 1  
             THEN '총합계'  
             ELSE '그룹별합계'  
       END AS '구분'  
FROM EMPLOYEE  
GROUP BY CUBE(DEPT_CODE, JOB_CODE)  
ORDER BY 1;
```

DEPT_CODE	JOB_CODE	합계	구분
1 D1	J6	6440000	그룹별합계
2 D1	J7	1380000	그룹별합계
3 D2	J4	6520000	그룹별합계
4 D5	J3	3500000	그룹별합계
5 D5	J5	8460000	그룹별합계
6 D5	J7	3800000	그룹별합계
7 D6	J3	7300000	그룹별합계
8 D6	J4	2800000	그룹별합계
9 D8	J6	6986240	그룹별합계
10 D9	J1	8000000	그룹별합계
11 D9	J2	9700000	그룹별합계
12 (null)	J6	2320000	그룹별합계
13 (null)	J7	2890000	그룹별합계
14 D1	(null)	7820000	부서합계
15 D2	(null)	6520000	부서합계
16 D5	(null)	15760000	부서합계
17 D6	(null)	10100000	부서합계
18 D8	(null)	6986240	부서합계
19 D9	(null)	17700000	부서합계
20 (null)	(null)	5210000	부서합계
21 (null)	J1	8000000	직급합계
22 (null)	J2	9700000	직급합계
23 (null)	J3	10800000	직급합계
24 (null)	J4	9320000	직급합계
25 (null)	J5	8460000	직급합계

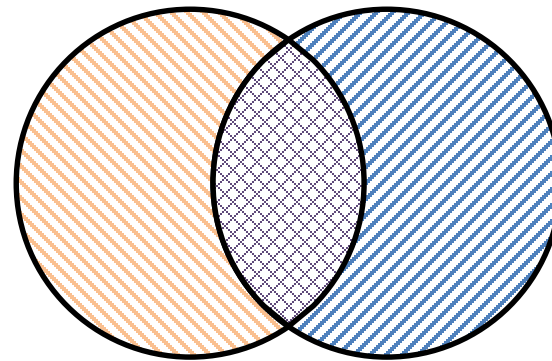
Oracle SQL – GROUP BY & HAVING

SET OPERATION

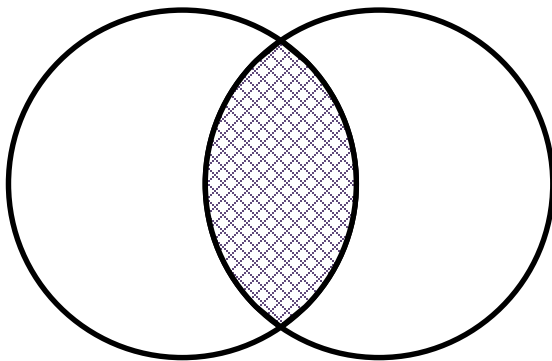
여러 개의 SELECT 결과물을 하나의 쿼리로 만드는 연산자이다.



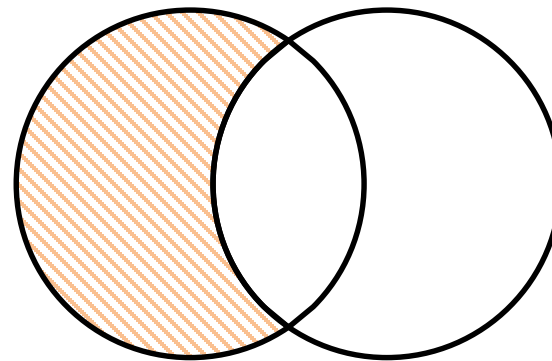
UNION



UNION ALL



INTERSECT



MINUS

Oracle SQL – GROUP BY & HAVING

UNION

UNION과 UNION ALL은 여러 개의 쿼리 결과를 하나로 합치는 연산자이다.

그 중 UNION은 중복된 영역을 제외하여 하나로 합치는 연산자이다.

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D5'
```

UNION

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	200	선동일	D9	8000000
2	201	송종기	D9	6000000
3	202	노웅철	D9	3700000
4	204	유재식	D6	3400000
5	205	정중하	D6	3900000
6	206	박나라	D5	1800000
7	207	하미유	D5	2200000
8	208	김해술	D5	2500000
9	209	심봉선	D5	3500000
10	210	윤은해	D5	2000000
11	215	대북혼	D5	3760000
12	217	전지연	D1	3660000

Oracle SQL – GROUP BY & HAVING

UNION ALL

UNION ALL은 UNION과 같은 여러 쿼리 결과물에 대한 합집합을 의미하며,
UNION과의 차이점은 중복된 영역을 모두 포함시키는 연산자이다.

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D5'
```

UNION ALL

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	206	박나라	D5	1800000
2	207	하미유	D5	2200000
3	208	김해술	D5	2500000
4	209	심봉선	D5	3500000
5	210	윤은해	D5	2000000
6	215	대북혼	D5	3760000
7	200	선동일	D9	8000000
8	201	송종기	D9	6000000
9	202	노웅철	D9	3700000
10	204	유재식	D6	3400000
11	205	정중하	D6	3900000
12	209	심봉선	D5	3500000
13	215	대북혼	D5	3760000
14	217	전지연	D1	3660000

Oracle SQL – GROUP BY & HAVING

INTERSECT

여러 개의 SELECT 결과에서 공통된 부분만 결과로 추출한다. 즉, 수행 결과에 대한 교집합이라고 볼 수 있다.

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D5'
```

INTERSECT

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	209	심봉선	D5	3500000
2	215	대북혼	D5	3760000

Oracle SQL – GROUP BY & HAVING

MINUS

선행 SELECT 결과에서 다음 SELECT 결과와 겹치는 부분을 제외한 나머지 부분만 추출한다.
즉, 두 쿼리 결과물의 차집합이라고 볼 수 있다.

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE DEPT_CODE = 'D5'
```

MINUS

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       SALARY  
FROM EMPLOYEE  
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	206	박나라	D5	1800000
2	207	하미유	D5	2200000
3	208	김해술	D5	2500000
4	210	윤은해	D5	2000000

Oracle SQL – GROUP BY & HAVING

GROUPING SETS

그룹별로 처리된 여러 개의 SELECT문을 하나로 합친 결과를 원할 때 사용한다. SET OPERATOR(집합연산자) 사용한 결과와 동일한 결과를 얻을 수 있다.

```
SELECT DEPT_CODE,  
       JOB_CODE,  
       MANAGER_ID,  
       FLOOR(AVG(SALARY))  
FROM EMPLOYEE  
GROUP BY GROUPING SETS(  
    (DEPT_CODE, JOB_CODE, MANAGER_ID),  
    (DEPT_CODE, MANAGER_ID),  
    (JOB_CODE, MANAGER_ID)  
);
```

SQL | 인출된 모든 행: 53(0.016초)

	DEPT_CODE	JOB_CODE	MANAGER_ID	FLOOR(AVG(SALARY))
1	D5	J5	207	2500000
2	D6	J4	204	2800000
3	D5	J3	207	3500000
4	D9	J2	200	6000000
5	D6	J3	200	3400000
6	D8	J6	211	2550000
7	(null)	J7	(null)	2890000
8	D8	J6	100	2436240
9	(null)	J6	(null)	2320000
10	D1	J6	214	3220000
11	D6	J3	204	3900000
12	D5	J7	207	1900000
13	D5	J5	200	2200000
14	D1	J7	200	1380000
15	D2	J4	(null)	2173333

•
•
•

51	(null)	J5	207	2500000
52	(null)	J5	(null)	3760000
53	(null)	J6	214	3220000

Oracle SQL – GROUP BY & HAVING

ROLLUP과 CUBE

```
SELECT DEPT_CODE,  
       JOB_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE  
GROUP BY ROLLUP(DEPT_CODE, JOB_CODE)  
  
UNION  
  
SELECT ‘’,  
       JOB_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE  
GROUP BY ROLLUP(DEPT_CODE, JOB_CODE)  
ORDER BY 1;
```

```
SELECT DEPT_CODE,  
       JOB_CODE,  
       SUM(SALARY)  
FROM EMPLOYEE  
GROUP BY CUBE(DEPT_CODE, JOB_CODE)  
ORDER BY 1;
```

Oracle - JOIN

Oracle SQL - JOIN

JOIN이란?

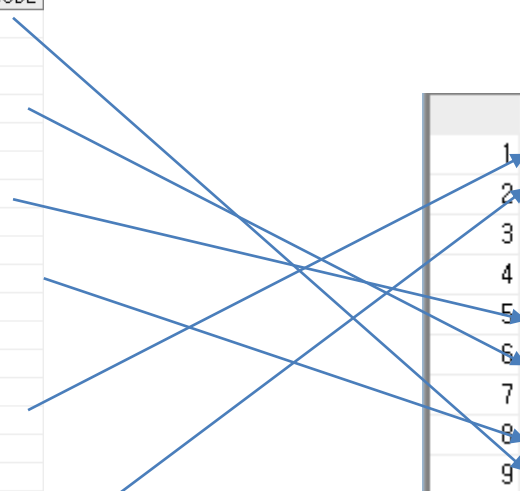
두 개 이상의 테이블에서 연관성을 가지고 있는 데이터들을 따로 분류하여 새로운 가상의 테이블을 이용하여 출력함. -> 여러 테이블의 레코드를 조합하여 하나의 열로 표현 한 것

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE  
FROM EMPLOYEE;
```

```
SELECT DEPT_ID  
       DEPT_TITLE  
FROM DEPARTMENT;
```

EMP_ID	EMP_NAME	DEPT_CODE
1 200	선동일	D9
2 201	송종기	D9
3 202	노웅철	D9
4 203	송은희	D6
5 204	유재식	D6
6 205	정종하	D6
7 206	박나라	D5
8 207	하미유	D5
9 208	김해솔	D5
10 209	심봉선	D5
11 210	윤은혜	D5
12 211	전형돈	D8
13 212	장프위	D8
14 213	하동운	(null)
15 214	방명수	D1
16 215	대북훈	D5
17 216	차태연	D1
18 217	전지연	D1
19 218	미오리	(null)
20 219	임시환	D2

DEPT_ID	DEPT_TITLE
1 D1	인사관리부
2 D2	회계관리부
3 D3	마케팅부
4 D4	국내영업부
5 D5	해외영업1부
6 D6	해외영업2부
7 D7	해외영업3부
8 D8	기술지원부
9 D9	총무부



JOIN의 종류

- INNER JOIN(내부 조인) : 교집합
(일반적으로 사용하는 JOIN)
- OUTER JOIN(외부 조인) : 합집합
 - => LEFT OUTER JOIN (왼쪽 외부 조인)
 - => RIGHT OUTER JOIN (오른쪽 외부 조인)
 - => FULL OUTER JOIN (완전 외부 조인)
- CROSS JOIN(상호 조인)
 - > 카테이션곱(Cartensian Product) 라고도 함
- NON_EQUI JOIN(비등가 조인)
 - > 지정한 컬럼 값이 일치하는 경우가 아닌, 값의 범위에 포함되는 행들을 연결하는 방식

JOIN의 사용법

※ JOIN 시 컬럼명이 같을경우와 다를경우 사용법_2가지

- Oracle 전용 구문

- ANSI 표준 구문 (DBMS에 상관없이 공통적으로 사용하는 표준 SQL)

○ 연결에 사용할 두 테이블의 컬럼명이 서로 다른 경우

- Oracle 전용 구문

```
SELECT EMP_ID,EMP_NAME,DEPT_CODE,DEPT_TITLE  
      FROM EMPLOYEE,DEPARTMENT WHERE DEPT_CODE = DEPT_ID;
```

Oracle에서는 where 절을 이용하여 '=' 를 사용하면됨

- ANSI 표준 구문

```
SELECT EMP_ID,EMP_NAME,DEPT_CODE,DEPT_TITLE  
      FROM EMPLOYEE JOIN DEPARTMENT ON(DEPT_CODE=DEPT_ID);
```

ANSI 표준에서는 JOIN 과 ON 을 사용함

JOIN의 사용법

○ 연결에 사용할 두 테이블의 컬럼명이 서로 같은 경우

- Oracle 전용 구문

```
SELECT EMP_ID, EMP_NAME, EMPLOYEE.JOB_CODE, JOB_NAME  
FROM EMPLOYEE, JOB WHERE EMPLOYEE.JOB_CODE = JOB.JOB_CODE;
```

Oracle에서는 테이블명.컬럼명 을 이용해서 작성해주어야 함

- ANSI 표준 구문

```
SELECT EMP_ID, EMP_NAME, JOB_CODE, JOB_NAME  
FROM EMPLOYEE JOIN JOB USING(JOB_CODE);
```

연결하려는 테이블의 컬럼명이 같을경우엔 USING() 을 사용함

USING 안에 있는 컬럼명은 JOIN된 테이블의 컬럼명

Oracle SQL - JOIN

JOIN의 종류 - INNER JOIN

테이블 A 와 테이블 B 모두 조건 구문에 일치하는 데이터만 반환

SELECT <컬럼> FROM <테이블 A> INNER JOIN <테이블 B> ON <조건 구문>;

```
SELECT
    EMP_NAME,
    DEPT_TITLE
FROM EMPLOYEE
INNER JOIN DEPARTMENT
ON (EMPLOYEE.DEPT_CODE = DEPARTMENT.DEPT_ID);
```

	EMP_NAME	DEPT_TITLE
1	선동일	총무부
2	송종기	총무부
3	노웅철	총무부
4	송은희	해외영업2부
5	유재식	해외영업2부
6	정중하	해외영업2부
7	박나라	해외영업1부
8	하미유	해외영업1부
9	김해솔	해외영업1부
10	심봉선	해외영업1부
11	윤은해	해외영업1부
12	전형돈	기술지원부
13	장쯔위	기술지원부
14	방명수	인사관리부
15	대북훈	해외영업1부
16	차태연	인사관리부
17	전지연	인사관리부
18	임시환	회계관리부

Oracle SQL - JOIN

JOIN의 종류 - LEFT OUTER JOIN

테이블 A 모두 반환, 테이블 B 조건 구문에 일치하는 데이터만 반환

SELECT <컬럼> FROM <테이블 A> LEFT OUTER JOIN <테이블 B> ON <조건 구문>;

```
SELECT
    EMP_NAME,
    DEPT_TITLE
FROM EMPLOYEE
LEFT OUTER JOIN DEPARTMENT
ON (EMPLOYEE.DEPT_CODE = DEPARTMENT.DEPT_ID);
```

EMP_NAME	DEPT_TITLE
1 전지연	인사관리부
2 차태연	인사관리부
3 방명수	인사관리부
4 유하진	회계관리부
5 이종석	회계관리부
6 임시환	회계관리부
7 대복훈	해외영업1부
8 윤은해	해외영업1부
9 심봉선	해외영업1부
10 김해솔	해외영업1부
11 하이유	해외영업1부
12 박나라	해외영업1부
13 정중하	해외영업2부
14 유재식	해외영업2부
15 송은희	해외영업2부
16 이태림	기술지원부
17 장프위	기술지원부
18 전형돈	기술지원부
19 노용철	총무부
20 송종기	총무부
21 선동일	총무부
22 미오리	(null)
23 하동운	(null)

Oracle SQL - JOIN

JOIN의 종류 - RIGHT OUTER JOIN

테이블 B 모두 반환, 테이블 A조건 구문에 일치하는 데이터만 반환

SELECT <컬럼> FROM <테이블 A> RIGHT OUTER JOIN <테이블 B> ON <조건 구문>;

```
SELECT
    EMP_NAME,
    DEPT_TITLE
FROM EMPLOYEE
RIGHT OUTER JOIN DEPARTMENT
ON (EMPLOYEE.DEPT_CODE = DEPARTMENT.DEPT_ID);
```

EMP_NAME	DEPT_TITLE
1 선동일	총무부
2 송중기	총무부
3 노웅철	총무부
4 송은희	해외영업2부
5 유재식	해외영업2부
6 정중하	해외영업2부
7 박나라	해외영업1부
8 하미유	해외영업1부
9 김해솔	해외영업1부
10 심봉선	해외영업1부
11 윤은혜	해외영업1부
12 전형돈	기술지원부
13 장프위	기술지원부
14 방영수	인사관리부
15 대복훈	해외영업1부
16 차태연	인사관리부
17 전지연	인사관리부
18 임시환	회계관리부
19 이종석	회계관리부
20 유하진	회계관리부
21 이태림	기술지원부
22 (null)	해외영업3부
23 (null)	마케팅부
24 (null)	국내영업부

Oracle SQL - JOIN

JOIN의 종류 - FULL OUTER JOIN

테이블 A, 테이블 B 조건 구문에 일치하는 모든 데이터 반환

SELECT <컬럼> FROM <테이블 A> FULL OUTER JOIN <테이블 B> ON <조건 구문>;

```
SELECT
    EMP_NAME,
    DEPT_TITLE
FROM EMPLOYEE
FULL OUTER JOIN DEPARTMENT
ON (EMPLOYEE.DEPT_CODE = DEPARTMENT.DEPT_ID);
```

EMP_NAME	DEPT_TITLE
1 선동일	총무부
2 송종기	총무부
3 노웅철	총무부
4 송은희	해외영업2부
5 유재식	해외영업2부
6 정중하	해외영업2부
7 박나라	해외영업1부
8 하미유	해외영업1부
9 김해솔	해외영업1부
10 심봉선	해외영업1부
11 윤은해	해외영업1부
12 전형돈	기술지원부
13 장쯔위	기술지원부
14 하동운	(null)
15 방영수	인사관리부
16 대북훈	해외영업1부
17 차태연	인사관리부
18 전지연	인사관리부
19 이오리	(null)
20 임시환	회계관리부
21 이종석	회계관리부
22 유하진	회계관리부
23 이태림	기술지원부
24 (null)	해외영업3부
25 (null)	마케팅부
26 (null)	국내영업부

Oracle SQL - JOIN

오라클 전용 구문 - OUTER JOIN

```
SELECT EMP_NAME, DEPT_TITLE  
FROM EMPLOYEE  
WHERE DEPT_CODE(+) = DEPT_ID;
```

```
SELECT EMP_NAME, DEPT_TITLE  
FROM EMPLOYEE  
WHERE DEPT_CODE(+) = DEPT_ID(+);
```

** 예러남, (+)는 한 쪽에만 사용할 수 있음

EMP_NAME	DEPT_TITLE
1 선동일	총무부
2 송종기	총무부
3 노웅철	총무부
4 송은희	해외영업2부
5 유재식	해외영업2부
6 정중하	해외영업2부
7 박나라	해외영업1부
8 하미유	해외영업1부
9 김해술	해외영업1부
10 심봉선	해외영업1부
11 윤은해	해외영업1부
12 전형돈	기술지원부
13 장프위	기술지원부
14 방명수	인사관리부
15 대북훈	해외영업1부
16 차태연	인사관리부
17 전지연	인사관리부
18 임시환	회계관리부
19 이종석	회계관리부
20 유하진	회계관리부
21 이태림	기술지원부
22 (null)	해외영업3부
23 (null)	마케팅부
24 (null)	국내영업부

Oracle SQL - JOIN

JOIN의 종류 - CROSS JOIN

카테이션곱(Cartesian product)라고도 한다. 조인 되는 테이블의 각 행들이 모두 매핑된 데이터가 검색 되는 조인 방법으로, 검색되는 데이터 수는 행의 컬럼수 x 행의 컬럼수 반환

SELECT <컬럼> FROM <테이블 A> CROSS JOIN <테이블 B>;

```
SELECT EMP_NAME, DEPT_TITLE  
FROM EMPLOYEE  
CROSS JOIN DEPARTMENT;
```

	EMP_N...	DEPT_TITLE
1	선동일	인사관리부
2	송종기	인사관리부
3	노용철	인사관리부
4	송은희	인사관리부
5	유재식	인사관리부
6	정중하	인사관리부
7	박나라	인사관리부
8	하미유	인사관리부
9	김해술	인사관리부
10	심봉선	인사관리부
11	윤은혜	인사관리부
205	미중식	총무부
206	유하진	총무부
207	이태림	총무부

Oracle SQL - JOIN

JOIN의 종류 - NON_EQUI JOIN

지정한 컬럼 값이 일치하는 경우가 아닌, 값의 범위에 포함되는 행들을 연결하는 방식
"=" 등가 연산자가 아닌 다른 (Between, >, <, >=, <=, <> 등) 연산자들을 사용함

```
SELECT EMP_ID,  
       EMP_NAME,  
       FLOOR(MONTHS_BETWEEN(SYSDATE, HIRE_DATE)/12) AS 근속년수,  
       YEAR_LEVEL  
JOIN YEAR_GRADE ON ((FLOOR(MONTHS_BETWEEN(SYSDATE, HIRE_DATE)/12))  
                   BETWEEN MIN_YEAR AND MAX_YEAR)  
ORDER BY 3 DESC;
```

SQL | 인출된 모든 행: 23(0초)

EMP_ID	EMP_NAME	근속년수	YEAR_LEVEL
1 200	선동일	27 A	
2 221	유하진	23 A	
3 207	하이유	23 A	
4 203	송은희	21 A	
5 222	이태림	20 A	
6 219	임시환	18 A	
7 205	정중하	18 A	
22 215	대복훈	0 D	
23 218	미오리	0 D	

Oracle SQL - JOIN

JOIN의 종류 - SELF JOIN

조인은 두 개 이상의 서로 다른 테이블을 연결하지만, 같은 테이블을 조인하는 것으로 자기 자신과 조인을 맺는 것이라 하여 SELF JOIN라 함

```
SELECT E.EMP_ID,  
       E.EMP_NAME 사원이름,  
       E.DEPT_CODE  
       E.MANAGER_ID,  
       M.EMP_NAME 관리자이름  
FROM EMPLOYEE E, EMPLOYEE M  
WHERE E.MANAGER_ID = M.EMP_ID;
```

EMP_ID	사원이름	DEPT_CODE	MANAGER_ID	관리자이름
1 214	방명수	D1	200	선동일
2 211	전형돈	D8	200	선동일
3 207	하미유	D5	200	선동일
4 204	유재식	D6	200	선동일
5 201	송종기	D9	200	선동일
6 202	노용철	D9	201	송종기
7 205	정중하	D6	204	유재식
8 203	송은희	D6	204	유재식
9 210	윤은혜	D5	207	하미유
10 209	심봉선	D5	207	하미유
11 208	김해솔	D5	207	하미유
12 206	박나라	D5	207	하미유
13 212	장프위	D8	211	전형돈
14 217	전지연	D1	214	방명수
15 216	차태연	D1	214	방명수

Oracle SQL - JOIN

오라클 전용 구문

```
SELECT EMP_ID,  
       EMP_NAME,  
       EMPLOYEE.JOB_CODE,  
       JOB_NAME  
FROM EMPLOYEE, JOB  
WHERE EMPLOYEE.JOB_CODE = JOB.JOB_CODE;
```

** 연결에 사용할 두 컬럼명이 같은 경우
테이블명.컬럼명 으로 작성해서 구분해 주어야 한다.

```
SELECT EMP_ID,  
       EMP_NAME,  
       E.JOB_CODE,  
       JOB_NAME  
FROM EMPLOYEE E, JOB J  
WHERE E.JOB_CODE = J.JOB_CODE;
```

** FROM절에 사용한 테이블 별칭을 이용해도 된다.

	EMP_ID	EMP_NAME	JOB_CODE	JOB_NAME
1	200	선동일	J1	대표
2	201	송중기	J2	부사장
3	202	노웅철	J2	부사장
4	203	송은희	J4	차장
5	204	유재식	J3	부장
6	205	정중하	J3	부장
7	206	박나라	J7	사원
8	207	하미유	J5	과장
9	208	김해술	J5	과장
10	209	심봉선	J3	부장
11	210	윤은혜	J7	사원
12	211	전형돈	J6	대리
13	212	장프위	J6	대리
14	213	하동운	J6	대리
15	214	방명수	J7	사원
16	215	대북혼	J5	과장
17	216	차태연	J6	대리
18	217	전지연	J6	대리
19	218	미오리	J7	사원
20	219	임시환	J4	차장
21	220	이중석	J4	차장
22	221	유하진	J4	차장
23	222	이태림	J6	대리

Oracle SQL - JOIN

JOIN의 종류 - 다중 JOIN

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       DEPT_TITLE,  
       LOCAL_NAME  
FROM EMPLOYEE  
JOIN DEPARTMENT ON(DEPT_CODE = DEPT_ID)  
JOIN LOCATION ON(LOCATION_ID = LOCAL_CODE);
```

```
SELECT EMP_ID,  
       EMP_NAME,  
       DEPT_CODE,  
       DEPT_TITLE,  
       LOCAL_NAME  
FROM EMPLOYEE  
JOIN LOCATION ON(LOCATION_ID = LOCAL_CODE)  
JOIN DEPARTMENT ON(DEPT_CODE = DEPT_ID);
```

**** 에러 발생**

다중 조인의 경우 조인의 순서가 중요하다.

EMP_ID	EMP_NAME	DEPT_CODE	DEPT_TITLE	LOCAL_NAME
1 200	선동일	D9	총무부	ASIA1
2 201	송중기	D9	총무부	ASIA1
3 202	노웅철	D9	총무부	ASIA1
4 203	송은희	D6	해외영업2부	ASIA3
5 204	유재식	D6	해외영업2부	ASIA3
6 205	정중하	D6	해외영업2부	ASIA3
7 206	박나라	D5	해외영업1부	ASIA2
8 207	하미유	D5	해외영업1부	ASIA2
9 208	김해솔	D5	해외영업1부	ASIA2
10 209	심봉선	D5	해외영업1부	ASIA2
11 210	윤은해	D5	해외영업1부	ASIA2
12 211	전형돈	D8	기술지원부	EU
13 212	장프워	D8	기술지원부	EU
14 214	방명수	D1	인사관리부	ASIA1
15 215	대복존	D5	해외영업1부	ASIA2
16 216	차태연	D1	인사관리부	ASIA1
17 217	전지연	D1	인사관리부	ASIA1
18 219	임시환	D2	회계관리부	ASIA1
19 220	이중석	D2	회계관리부	ASIA1
20 221	유하진	D2	회계관리부	ASIA1
21 222	이태림	D8	기술지원부	EU

Oracle SQL - JOIN

ANSI 표준 구문

연결에 사용하려는 컬럼명이 같은 경우 USING()을 사용하고, 다른 경우 ON()을 사용한다.

```
SELECT EMP_ID,  
       EMP_NAME,  
       JOB_CODE,  
       JOB_NAME  
FROM EMPLOYEE  
JOIN JOB USING(JOB_CODE);
```

	EMP_ID	EMP_NAME	JOB_CODE	JOB_NAME
1	200	선동일	J1	대표
2	201	송종기	J2	부사장
3	202	노웅철	J2	부사장
4	203	송은희	J4	차장
5	204	유재식	J3	부장
6	205	정중하	J3	부장
7	206	박나라	J7	사원
8	207	하미유	J5	과장
9	208	김해솔	J5	과장
10	209	심봉선	J3	부장
11	210	윤은해	J7	사원
12	211	전형돈	J6	대리
13	212	장프위	J6	대리
14	213	하동운	J6	대리
15	214	방명수	J7	사원
16	215	대복훈	J5	과장
17	216	차태연	J6	대리
18	217	전지연	J6	대리
19	218	미오리	J7	사원
20	219	임시환	J4	차장
21	220	이중석	J4	차장
22	221	유하진	J4	차장
23	222	이태림	J6	대리

Oracle SQL - JOIN

ANSI 표준 구문

```
SELECT
  EMP_ID,
  EMP_NAME,
  DEPT_CODE,
  DEPT_TITLE
FROM EMPLOYEE
JOIN DEPARTMENT
ON(DEPT_CODE = DEPT_ID);
```

	EMP_ID	EMP_NAME	DEPT_CODE	DEPT_TITLE
1	200	선동일	D9	총무부
2	201	송중기	D9	총무부
3	202	노용철	D9	총무부
4	203	송은희	D6	해외영업2부
5	204	유재식	D6	해외영업2부
6	205	정중하	D6	해외영업2부
7	206	박나라	D5	해외영업1부
8	207	하미유	D5	해외영업1부
9	208	김해솔	D5	해외영업1부
10	209	심봉선	D5	해외영업1부
11	210	윤은해	D5	해외영업1부
12	211	전형돈	D8	기술지원부
13	212	장프위	D8	기술지원부
14	214	방명수	D1	인사관리부
15	215	대복존	D5	해외영업1부
16	216	차태연	D1	인사관리부
17	217	전지연	D1	인사관리부
18	219	임시환	D2	회계관리부
19	220	이중석	D2	회계관리부
20	221	유하진	D2	회계관리부
21	222	이태림	D8	기술지원부