



**DML
(SELECT)**

▶ 주요 용어

EMP_ID	EMP_NAME	EMP_NO	EMAIL	PHONE	DEPT_CODE	JOB_CODE	SAL_LEVEL	SALARY	BONUS	M
200	선동일	621235-1985634	sun_di@kh.or.kr	01099546325	D9	J1	S1	8000000	0.3 (nul	
201	송중기	631156-1548654	song_jk@kh.or.kr	01045686656	D9	J2	S1	6000000	(null) 5	
202	노웅철	861015-1356452	no_hc@kh.or.kr	01066656263	D9	J2	S4	3700000	(null) 201	
203	송은희	631010-2653546	song_eh@kh.or.kr	01077607879	D6	J4	S5	2800000	(null) 204	
204	유재식	660508-1342154	yoo_js@kh.or.kr	01099999129	D6	J3	S4	3400000	0.2 200	
205	정중하	770102-1357951	ung_jh@kh.or.kr	01036654875	D6	J3	S4	3900000	(null) 204	
206	박나라	630709-2054321	pack_nr@kh.or.kr	01096935222	D5	J7	S6	1800000	(null) 207	
207	하미유	690402-2040612	ha_iy@kh.or.kr	01036654488	D5	J5	S5	2200000	0.1 200	
208	김해솔	870927-1313564	kim_hs@kh.or.kr	01078634444	D5	J5	S5	2500000	(null) 207	
209	심봉선	750206-1325546	sim_bs@kh.or.kr	0113654485	D5	J3	S4	3500000	0.15 207	
210	윤은혜	650505-2356985	youn_eh@kh.or.kr	0179964233	D5	J7	S5	2000000	(null) 207	
211	전형돈	830807-1121321	jun_hd@kh.or.kr	01044432222	D8	J6	S5	2000000	(null) 200	
212	장쯔위	780923-2234542	jang_zw@kh.or.kr	01066682224	D8	J6	S5	2550000	0.25 211	
213	하동운	621111-1785463	ha_dh@kh.or.kr	01158456632	(null)	J6	S5	2320000	0.1 (nul	
214	방명수	856795-1313513	bang_ms@kh.or.kr	01074127545	D1	J7	S6	1380000	(null) 200	
215	대북혼	881130-1050911	dae_bh@kh.or.kr	01088808584	D5	J5	S4	3760000	(null) (nul	
216	차태연	770808-1364897	cha_ty@kh.or.kr	01064643212	D1	J6	S5	2780000	0.2 214	
217	전지연	770808-2665412	jun_jy@kh.or.kr	01033624442	D1	J6	S4	3660000	0.3 214	
218	미오리	870427-2232123	loo_or@kh.or.kr	01022306545	(null)	J7	S5	2890000	(null) (nul	
219	임시환	660712-1212123	im_sw@kh.or.kr	(null)	D2	J4	S6	1550000	(null) (nul	

SAL_LEVEL	MIN_SAL	MAX_SAL
S1	6000000	10000000
S2	5000000	5999999
S3	4000000	4999999
S4	3000000	3999999
S5	2000000	2999999
S6	1000000	1999999

① 행(Row), 튜플

② 컬럼, 도메인

③ 기본키(Primary Key)

④ 외래키(Foreign Key)

⑤ Null

⑥ 컬럼값, 속성값

▶ SQL(Structured Query Language)

관계형 데이터베이스에서 데이터를 조회하거나 조작하기 위해 사용하는 표준 검색 언어로 원하는 데이터를 찾는 방법이나 절차를 기술하는 게 아닌 조건을 기술하여 작성

분류	용도	명령어
DQL (Data Query Language)	데이터 검색	SELECT
DML (Data Manipulation Language)	데이터 조작	INSERT, UPDATE, DELETE
DDL (Data Definition Language)	데이터 정의	CREATE, DROP, ALTER
TCL (Transaction Control Language)	트랜잭션 제어	COMMIT, ROLLBACK

▶ 주요 데이터 타입

데이터 타입	하위 데이터 타입	설명
NUMBER		숫자
CHARACTER	CHAR	고정길이 문자(최대 2000바이트)
	VARCHAR2	가변길이 문자(최대 4000바이트)
	LONG	가변길이 문자(최대 2기가 바이트)
DATE		날짜
LOB	CLOB	가변길이 문자(최대 4기가 바이트)
	BLOB	Binary Data

▶ NUMBER

NUMBER[(P [, S])]

* P : 표현할 수 있는 전체 숫자 자리 수 (1 ~ 38)

* S : 소수점 이하 자리 수 (-84 ~ 127)

실제 값	데이터 타입	저장 되는 값	설명
12345.678	NUMBER	12345.678	
	NUMBER(7)	12346	7자리이지만 정수는 5자리이므로 5개만 표현, 첫 번째 소수로 인해 반올림 되어 저장
	NUMBER(7, 1)	12345.7	7자리이지만 정수 5자리와 소수 1자리만 표현, 두 번째 소수로 인해 반올림 되어 저장
	NUMBER(7, 3)	(오류)	7자리, 소수점 이하 3자리로 정수는 총 4자리인데 실제 값의 정수는 5자리이므로 오류
	NUMBER(5, -2)	12300	s가 -2여서 소수점 왼쪽 두 번째 자리 4가 반올림되어 저장
0.1234	NUMBER(4, 5)	(오류)	유효숫자는 4개가 맞지만 소수점 아래가 5자리인데 4자리이므로 오류
0.01234	NUMBER(4, 5)	0.01234	
0.0001234	NUMBER(3, 7)	(오류)	소수점 이하 일곱 째 자리까지 유효숫자는 4개인데 p가 3이므로 오류
0.00001234	NUMBER(3, 7)	0.0000123	소수점 이하 일곱 째 자리까지 유효숫자는 3개이기 때문에 마지막 4 제외

▶ CHARACTER

✓ CHAR

CHAR(SIZE [(byte | char)]

- * SIZE : 포함될 문자(열)의 크기
- * 지정한 크기보다 작은 문자(열)가 입력되면 남은 공간은 공백으로 채움
- * 데이터는 "를 사용하여 표기하고 대·소문자를 구분함

실제 값	데이터 타입	저장 되는 값	설명
KIMCHI	CHAR(6)	KIMCHI	
	CHAR(9)	KIMCHI***	공백 3칸(3byte)
	CHAR(3)	(오류)	저장되는 글자는 6글자인데 공간은 3자리이기 때문에 오류
김치	CHAR(6)	김치	한글은 한 글자 당 3byte이므로 공간에 딱 맞음
	CHAR(9)	김치***	공백 3byte
	CHAR(3)	(오류)	저장되는 글자는 총 6byte인데 공간은 3byte이므로 오류

▶ CHARACTER

✓ VARCHAR2

VARCHAR2(SIZE [(byte | char)]

- * SIZE : 포함될 문자(열)의 크기
- * 크기가 0인 값은 NULL로 인식
- * 데이터는 "를 사용하여 표기하고 대·소문자를 구분함

실제 값	데이터 타입	저장 되는 값	설명
KIMCHI	VARCHAR(6)	KIMCHI	
	VARCHAR(10)	KIMCHI	
	VARCHAR(3)	(오류)	저장되는 글자는 6글자인데 공간은 3자리이기 때문에 오류
김치	VARCHAR(6)	김치	한글은 한 글자 당 3byte이므로 공간에 딱 맞음
	VARCHAR(10)	김치	
	VARCHAR(3)	(오류)	저장되는 글자는 총 6byte인데 공간은 3byte이므로 오류

▶ DATE

DATE

- * 일자(세기/년/월/일) 및 시간(시/분/초) 정보 관리
- * 기본적으로 화면에 년/월/일 정보만 표기
- * 날짜 연산 및 비교 가능

연산	결과 타입	설명
날짜 + 숫자	DATE	날짜에서 숫자만큼 며칠 후
날짜 - 숫자	DATE	날짜에서 숫자만큼 며칠 전
날짜 - 날짜	NUMBER	두 날짜의 일수 차
날짜 + 숫자/24	DATE	날짜 + 시간

▶ SELECT

데이터를 조회한 결과를 Result Set이라고 하는데 SELECT구문에 의해 반환된 행들의 집합을 의미함

Result Set은 0개 이상의 행이 포함될 수 있고 특정 기준에 의해 정렬 가능한 테이블의 특정 컬럼, 행, 행/컬럼 또는 여러 테이블의 특정 행/컬럼 조회 가능

▶ SELECT

✓ 작성법

```
SELECT 컬럼 명 [, 컬럼명, ...]  
FROM 테이블 명  
[WHERE 조건식];
```

* SELECT

조회하고자 하는 컬럼명 기술

여러 컬럼을 조회하는 경우 컬럼은 쉼표로 구분하고, 마지막 컬럼 다음은 쉼표를 사용하지 않음
모든 컬럼 조회 시 컬럼 명 대신 '*' 기호 사용 가능하며 조회 결과는 기술한 컬럼 명 순으로 표시 됨

* FROM

조회 대상 컬럼이 포함된 테이블 명 기술

* WHERE

행을 선택하는 조건 기술

여러 개의 제한 조건을 포함할 수 있으며, 각각의 제한 조건은 논리 연산자로 연결
제한 조건을 만족시키는 행들만 Result Set에 포함

▶ SELECT 예시

✓ 기본

직원 전부의 사번과 이름, 월급을 조회하는 구문

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

EMP_ID	EMP_NAME	SALARY
200	선동일	8000000
201	송종기	6000000
202	노웅철	3700000
203	송은희	2800000
204	유재식	3400000
205	정중하	3900000
206	박나라	1800000
207	하미유	2200000
208	김해술	2500000
209	심봉선	3500000
210	윤은해	2000000
211	전형돈	2000000
212	장프위	2550000
213	하동운	2320000
214	방명수	1380000
215	대복존	3760000
216	차태연	2780000
217	전지연	3660000
218	이오리	2890000
219	임시환	1550000
220	이중석	2490000
221	유하진	2480000
222	이태림	2436240

▶ SELECT 예시

✓ 기본

직원 전부의 모든 정보를 조회하는 구문

```
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	송종기	631156-1548654	song_jk@kh.or.kr	01045686656	D9	J2	S1	6000000	(null) 200		01/09/01	(null)	N
202	노용철	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	김해솔	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	대복훈	881130-1050911	dae_bh@kh.or.kr	01088808584	D5	J5	S4	3760000	(null) (null)		17/06/19	(null)	N
216	차태연	770808-1364897	cha_ty@kh.or.kr	01064643212	D1	J6	S5	2780000	0.2 214		13/03/01	(null)	N
217	전지연	770808-2665412	jun_jy@kh.or.kr	01033624442	D1	J6	S4	3660000	0.3 214		07/03/20	(null)	N
218	미오리	870427-2232123	loo_or@kh.or.kr	01022306545 (null)		J7	S5	2890000	(null) (null)		16/11/28	(null)	N
219	임시환	660712-1212123	im_sw@kh.or.kr	(null)	D2	J4	S6	1550000	(null) (null)		99/09/09	(null)	N
220	이종석	770823-1113111	lee_js@kh.or.kr	(null)	D2	J4	S5	2490000	(null) (null)		14/09/18	(null)	N
221	유하진	800808-1123341	yoo_hj@kh.or.kr	(null)	D2	J4	S5	2480000	(null) (null)		94/01/20	(null)	N
222	이태림	760918-2854697	lee_tr@kh.or.kr	01033000002	D8	J6	S5	2436240	0.35 100		97/09/12	17/09/12	Y

▶ SELECT 예시

✓ 컬럼 값 산술 연산

컬럼 값에 대해 산술 연산한 결과 조회 가능

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

EMP_NAME	SALARY*12	(SALARY+(SALARY*BONUS))*12
선동일	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
최도우	37800000	47520000

▶ SELECT 예시

✓ 컬럼 별칭

'AS 별칭'이나 "별칭", 또는 'AS "별칭"'을 기술하여
컬럼 별칭을 지을 수 있음

```
SELECT EMP_NAME AS 이름, SALARY*12 "연봉(원)",  
       (SALARY + (SALARY*BONUS))*12 AS "총 소득(원)"  
FROM EMPLOYEE;
```

- * 숫자 혹은 특수문자가 포함되는 경우에 " " 사용
- * AS 생략 가능(공백으로 구분)

	이름	연봉(원)	총 소득(원)
1	선동일	96000000	124800000
2	송종기	72000000	(null)
3	노용철	44400000	(null)
4	송은희	33600000	(null)
5	유재식	40800000	48960000
6	정중하	46800000	(null)
7	박나라	21600000	(null)
8	하미유	26400000	29040000
9	김해솔	30000000	(null)
10	심봉선	42000000	48300000
11	윤은해	24000000	(null)
12	전형돈	24000000	(null)
13	장프워	30600000	38250000
14	하동우	27840000	30624000

▶ SELECT 예시

✓ 리터럴

임의로 지정한 문자열을 SELECT절에 사용하면 테이블에 존재하는 데이터처럼 활용 가능

SELECT EMP_ID, SALARY, '원' AS 단위

FROM EMPLOYEE;

- * 문자나 날짜 리터럴은 '' 기호 사용
- * 리터럴은 Result Set의 모든 행에 반복 표시 됨

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
211	2000000	원

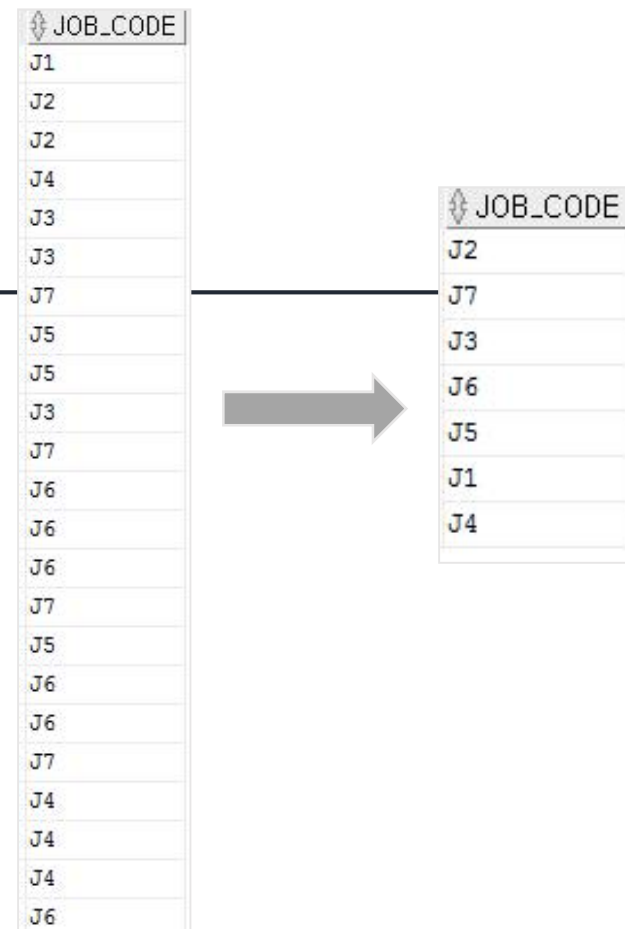
▶ SELECT 예시

✓ DISTINCT

컬럼에 포함된 데이터 중 중복 값을 제외하고 한 번씩만
표시하고자 할 때 사용

```
SELECT DISTINCT JOB_CODE  
FROM EMPLOYEE;
```

* SELECT절에 1회만 기술 가능



The diagram illustrates the effect of the DISTINCT clause. On the left, a table named 'JOB_CODE' contains a list of values with many duplicates. An arrow points to the right, where a second 'JOB_CODE' table shows only the unique values from the first table, demonstrating how DISTINCT removes duplicates.

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

JOB_CODE
J2
J7
J3
J6
J5
J1
J4

▶ SELECT 예시

✓ WHERE절

검색할 컬럼의 조건을 설정하여 행 결정

- 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
노웅철	D9

- 급여가 4000000보다 많은 직원 이름과 급여 조회

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

EMP_NAME	SALARY
선동일	8000000
송종기	6000000

▶ SELECT 예시

✓ WHERE절

여러 개 조건 작성 시 AND/OR 사용

- 부서코드가 'D6'이고 급여를 2000000보다 많이 받는 직원의 이름, 부서코드, 급여 조회

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

	EMP_NAME	DEPT_CODE	SALARY
1	송은희	D6	2800000
2	유재식	D6	3400000
3	정중하	D6	3900000

- 부서코드가 'D6'이거나 급여를 2000000보다 많이 받는 직원의 이름, 부서코드, 급여 조회

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

	EMP_NAME	DEPT_CODE	SALARY
1	길성훈	D5	3000000
2	선동일	D9	8000000
3	송중기	D9	6000000
4	노용철	D9	3700000
5	송은희	D6	2800000

▶ 연결 연산자

'||'를 사용하여 여러 컬럼을 하나의 컬럼인 것처럼 연결하거나 컬럼과 리터럴을 연결함

✓ 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	하미유	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	선동일의	월급은	8000000	원 입니다.
2	송종기의	월급은	6000000	원 입니다.
3	노용철의	월급은	3700000	원 입니다.
4	송은희의	월급은	2800000	원 입니다.
5	유재식의	월급은	3400000	원 입니다.
6	정중하의	월급은	3900000	원 입니다.
7	박나라의	월급은	1800000	원 입니다.
8	하미유의	월급은	2200000	원 입니다.
9	김해숙의	월급은	2500000	원 입니다.

▶ 논리 연산자

여러 개의 제한 조건 결과를 하나의 논리 결과로 만들어줌

연산자	설명
AND	여러 조건이 동시에 TRUE일 경우에만 TRUE값 반환
OR	여러 조건들 중에 어느 하나의 조건만 TRUE이면 TRUE값 반환
NOT	조건에 대한 반대 값으로 반환(NULL 제외)

[AND 연산 결과]

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

[OR 연산 결과]

	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

▶ 비교 연산자

표현식 사이의 관계를 비교하기 위해 사용하고 비교 결과는 논리 결과(TRUE/FALSE/NULL) 중 하나가 됨
단 비교하는 두 컬럼 값/표현식은 서로 동일한 데이터 타입이어야 함

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

[주요 비교 연산자]

▶ 비교 연산자

✓ BETWEEN AND

비교하려는 값이 지정한 범위에 포함되면 TRUE를 리턴하는 연산자
상한 값과 하한 값의 경계도 포함됨

- 급여를 3500000보다 많이 받고 6000000보다 적게 받는 직원
이름과 급여 조회

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

또는

```
SELECT EMP_NAME, SALARY
FROM EMPLOYEE
WHERE SALARY BETWEEN 3500000 AND 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 만족하면 TRUE를 리턴하는 연산자로 '%'와 '_'를 와일드카드로 사용

- '전'씨 성을 가진 직원 이름과 급여 조회

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

	EMP_NAME	SALARY
1	전형돈	2000000
2	전지연	3660000

- 핸드폰의 앞 네 자리 중 첫 번호가 7인 직원 이름과 전화번호 조회

```
SELECT EMP_NAME, PHONE  
FROM EMPLOYEE  
WHERE PHONE LIKE '_ _ _ 7%';
```

	EMP_NAME	PHONE
1	송은희	01077607879
2	김해술	01078634444
3	방명수	01074127545

▶ 비교 연산자

✓ LIKE

와일드 카드 문자와 패턴의 특수문자가 동일한 경우 어떤 것을 패턴으로 결정하는지 구분하지 못하기 때문에 데이터로 처리할 와일드 카드 문자 패턴 기호 앞에 임의의 특수문자를 사용하고 ESCAPE OPTION으로 등록하여 처리

- EMAIL ID 중 '_'의 앞이 3자리인 직원 이름, 이메일 조회

```
SELECT EMP_NAME, EMAIL
```

```
FROM EMPLOYEE
```

```
WHERE EMAIL LIKE '___#_%' ESCAPE '#';
```

	EMP_NAME	EMAIL
1	선동일	sun_di@kh.or.kr
2	유재식	yoo_js@kh.or.kr
3	김해술	kim_hs@kh.or.kr
4	심봉선	sim_bs@kh.or.kr
5	전형돈	jun_hd@kh.or.kr
6	대복훈	dae_bh@kh.or.kr
7	차태연	cha_ty@kh.or.kr
8	전지연	jun_jy@kh.or.kr
9	미오리	loo_or@kh.or.kr
10	이중석	lee_js@kh.or.kr
11	유하진	yoo_hj@kh.or.kr
12	이태림	lee_tr@kh.or.kr

▶ 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	김해솔	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	장프위	jang_zw@kh.or.kr
14 213	하동은	ha_dh@kh.or.kr
15 214	방명수	bang_ms@kh.or.kr
16 215	대북혼	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

✓ IS NULL과 IS NOT NULL

NULL 여부를 비교하는 연산자

- 관리자도 없고 부서 배치도 받지 않은 직원 조회

```
SELECT EMP_NAME, MANAGER_ID, DEPT_CODE  
FROM EMPLOYEE
```

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

```
WHERE MANAGER_ID IS NULL AND DEPT_CODE IS NULL;
```

- 부서 배치를 받지 않았지만 보너스를 지급받는 직원 조회

```
SELECT EMP_NAME, BONUS, DEPT_CODE  
FROM EMPLOYEE
```

EMP_NAME	BONUS	DEPT_CODE
1 하동운	0.1	(null)

```
WHERE DEPT_CODE IS NULL AND BONUS IS NOT NULL;
```

▶ 비교 연산자

✓ IN

비교하려는 값 목록에 일치하는 값이 있으면
TRUE를 반환하는 연산자

- D6 부서와 D8 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

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

- 'J2' 또는 'J7' 직급 코드 중 급여를 2000000보다 많이 받는 직원의 이름, 급여, 직급코드 조회

```
SELECT EMP_NAME, SALARY, JOB_CODE
FROM EMPLOYEE
WHERE JOB_CODE = 'J7' OR JOB_CODE = 'J2'
      AND SALARY > 2000000;
```

	EMP_NAME	SALARY	JOB_CODE
1	송종기	6000000	J2
2	노웅철	3700000	J2
3	박나라	1800000	J7
4	윤은혜	2000000	J7
5	방명수	1380000	J7
6	미오리	2890000	J7

- * 연산자 우선 순위에 의해서 AND가 먼저 실행됨
J2직급의 급여 2000000 이상 받는 직원이거나 J7직급인 직원이라는 의미

```
SELECT EMP_NAME, SALARY, JOB_CODE
FROM EMPLOYEE
WHERE (JOB_CODE = 'J7' OR JOB_CODE = 'J2')
      AND SALARY > 2000000;
```

	EMP_NAME	SALARY	JOB_CODE
1	송종기	6000000	J2
2	노웅철	3700000	J2
3	윤은혜	2000000	J7
4	미오리	2890000	J7

- * 우선순위를 고려하여 OR가 먼저 처리 되도록 ()를 이용해 우선 순위 변경
J7 직급이거나 J2직급인 직원들 중 급여 2000000 이상 받는 직원이라는 의미