

Structured Query Language (SQL)

Part 2. DML

Hyeokman Kim
School of Computer Science
Kookmin Univ.

ANSI/ISO SQL

SQL DML : DATA UPDATE (INSERT, DELETE, UPDATE)

1. INSERT 문

○ Syntax 1

```
INSERT INTO 테이블명 [(컬럼명_리스트)]  
VALUES      (컬럼값_리스트);
```

– 컬럼명과 컬럼값은 순서대로 1:1 매핑

– 컬럼값의 표현

- ◆ 컬럼의 데이터유형이 문자 : ' ' (single quotation)으로 값을 표현
- ◆ 컬럼의 데이터유형이 숫자 : ' ' (single quotation)을 사용 안함.

○ Syntax 2

```
INSERT INTO 테이블명 [(컬럼명_리스트)]  
SELECT 문;
```

○ 예제

```
INSERT INTO PLAYER (PLAYER_ID, PLAYER_NAME, TEAM_ID, POSITION,  
                     HEIGHT, WEIGHT, BACK_NO)  
VALUES ('2002007', '박지성', 'K07', 'MF', 178, 73, 7);
```

```
INSERT INTO PLAYER  
VALUES ('2002010','이청용','K07','BlueDragon','2002','MF','17',NULL,  
        NULL, '1', 180, 69);
```

○ 예제

```
INSERT INTO COMPUTER (SNO, SNAME, YEAR)  
SELECT      SNO, SNAME, YEAR  
FROM        STUDENT  
WHERE       DEPT = '컴퓨터';
```

2. DELETE 문

○ Syntax

```
DELETE FROM 테이블명  
[WHERE 조건식];
```

– WHERE절이 없으면, 테이블의 전체 데이터가 삭제됨.

○ 테이블의 전체 데이터를 삭제하는 경우

- 삭제된 데이터를 로그로 저장하는 DELETE FROM 보다는,
- 시스템 부하가 적은 TRUNCATE TABLE을 권함.

– 🖐 DDL과 DML의 차이

- ◆ DDL 명령어는 하드디스크의 테이블에 직접 적용되므로, 해당 작업이 즉시 완료됨(AUTO COMMIT).
- ◆ DML 명령어는 테이블을 메모리 버퍼에 올려놓고 작업을 하므로, 실시간으로 테이블에 영향을 미치는 않음. 따라서 버퍼에서 처리한 명령어가 테이블에 실제로 반영되기 위해서는 COMMIT 명령어를 입력하여 트랜잭션을 종료해야 함.

ANSI/ISO SQL ~~◆ (SQL Server의 경우, DML 명령도 AUTO COMMIT으로 처리되기 때문에, 실제 테이블에 반영하기 위해 COMMIT 명령어를 입력할 필요가 없음.)~~

○ 예제

```
DELETE FROM PLAYER
```

○ 예제

```
DELETE FROM STUDENT  
WHERE SNO=100;
```

```
DELETE FROM ENROL  
WHERE CNO='C413' AND FINAL<60 AND  
ENROL.SNO IN  
(SELECT SNO  
FROM STUDENT  
WHERE DEPT='컴퓨터');
```

3. UPDATE 문

- Syntax

```
UPDATE 테이블명  
SET      {컬럼명 = 산술식,}+  
[WHERE 조건식];
```

- 예제

```
UPDATE PLAYER  
SET    BACK_NO = 99
```

- 예제

```
UPDATE STUDENT  
SET    YEAR = 2  
WHERE  SNO = 300;
```

ANSI/ISO SQL

SQL DML : QUERY (SELECT)

1. SELECT 문

○ Syntax

```
SELECT      [ALL|DISTINCT] {{컬럼명 [[AS] 컬럼_별명],}+ | * }  
FROM        테이블_리스트  
[WHERE       조건식]  
[GROUP BY   컬럼명 [HAVING 그룹조건식]]  
[ORDER BY   {컬럼명|컬럼_별명|컬럼_순서 [ASC|DESC],}+];
```

☞ 조건식 := 컬럼명 비교연산자|SQL연산자 {숫자|문자|표현식}|컬럼명

☞ FROM 절은 일부 벤더(예. SQL Server)의 경우, 생략 가능함.

- 예제 : SELECT ALL이 디폴트 값임.

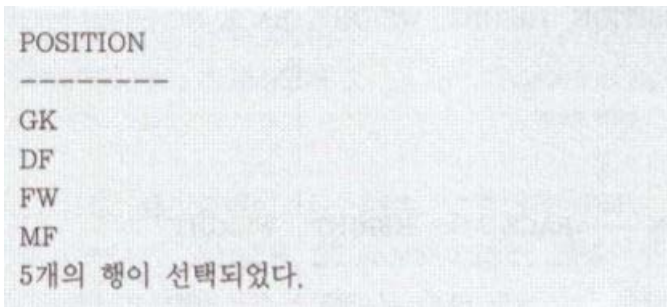
```
SELECT  PLAYER_ID, PLAYER_NAME, TEAM_ID, POSITION, BACK_NO,  
          HEIGHT,  WEIGHT  
FROM    PLAYER;
```

PLAYER_ID	PLAYER_NAME	TEAM_ID	POSITION	BACK_NO	HEIGHT	WEIGHT
2007155	정경량	K05	MF	19	173	65
2010025	정은익	K05	MF	35	176	63
2012001	레오마르	K05	MF	5	183	77
2008269	명재용	K05	MF	7	173	63
2007149	변재섭	K05	MF	11	170	63
2012002	보띠	K05	MF	10	174	68
2011123	비에라	K05	MF	21	176	73
2008460	서동원	K05	MF	22	184	78
2010019	안대현	K05	MF	25	179	72
2010018	양현정	K05	MF	14	176	72
2010022	유원섭	K05	MF	37	180	77
2012008	김수철	K05	MF	34	171	68
2012013	임다한	K05	DF	39	181	67
⋮	⋮	⋮	⋮	⋮	⋮	⋮


480개의 행이 선택되었다.

○ 예제

```
SELECT DISTINCT POSITION  
FROM   PLAYER;
```



A screenshot of a database query result. The title is "POSITION" followed by a dashed line. Below the line, the positions "GK", "DF", "FW", and "MF" are listed vertically. At the bottom, a note in Korean says "5개의 행이 선택되었다." (5 rows were selected).

—  NULL 값 포함 5건

○ 예제 : Wildcard

```
SELECT *
FROM PLAYER;
```

PLAYER_ID	PLAYER_NAME	TEAM_ID	E_PLAYER_NAME	NICKNAME	JOIN YYYY	POSITION	BACK_NO	NATION	BIRTH_DATE	SOLAR	HEIGHT	WEIGHT
2007155	정경량	K05	JEONG, KYUNGRYANG		2006	MF	19		1983-12-22	1	173	65
2010025	정은익	K05				MF	35		1991-03-09	1	176	63
2012001	레오마르	K05	Leomar Leiria	레오	2012	MF	5		1981-06-26	1	183	77
2008269	명재용	K05	MYUNG, JAEYOENG		2007	MF	7		1983-02-26	2	173	63
2007149	변재섭	K05	BYUN, JAESUB	작은탱크	2007	MF	11		1985-09-17	2	170	63
2012002	보띠	K05	Raphael JoseBotti Zacarias Sena	Botti	2012	MF	10		1991-02-23	1	174	68
2011123	비에라	K05	Vieira		2011	MF	21		1984-02-25	1	176	73
2008460	서동원	K05	SEO, DONGWON		2008	MF	22		1985-08-14	1	184	78
2010019	안대현	K05	AN, DAEHYUN		2010	MF	25		1987-08-20	1	179	72
2010018	양현정	K05	YANG, HYUNJUNG		2010	MF	14		1987-07-25	1	176	72
2010022	유원섭	K05	YOU, WONSUOB	앙마	2010	MF	37		1991-05-24	1	180	77
2012008	김수철	K05	KIM, SUCHEUL		2012	MF	34		1989-05-26	1	171	68
2012013	임다한	K05	LIM, DAHAN	달마	2012	DF	39		1989-07-21	1	181	67
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

480개의 행이 선택되었다.

○ 예제 : 컬럼_별명(alias) 부여

```
SELECT  PLAYER_NAME AS 선수명, POSITION AS 위치, HEIGHT AS 키,  
          WEIGHT AS 몸무게  
FROM    PLAYER;
```

```
SELECT  PLAYER_NAME 선수명, POSITION 위치, HEIGHT 키,  
          WEIGHT 몸무게  
FROM    PLAYER;
```

선수명	위치	키	몸무게
----	---	--	----
정경량	MF	173	65
정은익	MF	176	63
레오마르	MF	183	77
명재용	MF	173	63
변재섭	MF	170	63
보띠	MF	174	68
비에라	MF	176	73
서동원	MF	184	78
안대현	MF	179	72
양현정	MF	176	72
유원섭	MF	180	77
김수철	MF	171	68
임다한	DF	181	67
⋮	⋮	⋮	⋮

ANSI/IS

480개의 행이 선택되었다.



국민대학교
KOOKMIN UNIVERSITY

- 예제 : 컬럼_별명에 공백이 들어갈 때

```
SELECT  PLAYER_NAME "선수 이름", POSITION "그라운드 포지션",  
          HEIGHT 키, WEIGHT 몸무게  
FROM    PLAYER;
```

선수 이름	그라운드 포지션	키	몸무게
-----	-----	--	----
정경량	MF	173	65
정은익	MF	176	63
레오마르	MF	183	77
명재용	MF	173	63
변재섭	MF	170	63
보피	MF	174	68
비에라	MF	176	73
서동원	MF	184	78
안대현	MF	179	72
양현정	MF	176	72
유원섭	MF	180	77
김수철	MF	171	68
임다한	DF	181	67
⋮	⋮	⋮	⋮

480개의 행이 선택되었다.

○ 예제 : 산술 연산자

```
SELECT  PLAYER_NAME 이름,  
          ROUND(WEIGHT/((HEIGHT/100)*(HEIGHT/100)),2) "BMI 비만지수"  
FROM    PLAYER;
```

이름	BMI 비만지수
정경량	21.72
정은익	20.34
레오마르	22.99
명재용	21.05
변재섭	21.80
보띠	22.46
비에라	23.57
서동원	23.04
안대현	22.47
양현정	23.24
유원섭	23.77
김수철	23.26
임다한	20.45
...	...

480개의 행이 선택되었다.

○ 예제 : 합성(concatenation) 연산자

- 컬럼과 컬럼, 혹은 컬럼과 문자열을 연결하여, 새로운 컬럼을 생성.
- Oracle : ||, 혹은 CONCAT(string1,string2)

```
SELECT  PLAYER NAME || '선수,' || HEIGHT || 'cm,' || WEIGHT || 'kg'  
        체격정보  
FROM    PLAYER;
```

- SQL Server: +, 혹은 CONCAT(string1,string2)

```
SELECT  PLAYER NAME + '선수,' + HEIGHT + 'cm,' + WEIGHT + 'kg'  
        체격정보  
FROM    PLAYER;
```

체격정보

정경량선수, 173cm, 65kg
정은익선수, 176cm, 63kg
레오마르선수, 183cm, 77kg
명재용선수, 173cm, 63kg
변재섭선수, 170cm, 63kg
보띠선수, 174cm, 68kg
비에라선수, 176cm, 73kg
서동원선수, 184cm, 78kg
안대현선수, 179cm, 72kg
양현정선수, 176cm, 72kg
유원섭선수, 180cm, 77kg
김수철선수, 171cm, 68kg
임다한선수, 181cm, 67kg
...

480개의 행이 선택되었다.

2. SELECT 문 - WHERE 절

○ 역할

- FTS(Full Table Scan) : WHERE절이 없는 SELECT 문
 - ◆ 시스템에 매우 큰 부담, SQL 튜닝의 1차적 검토 대상
- 전체 튜플들에서 조건에 맞는 튜플들만 선택함.
- 집계 함수는 WHERE 절에는 올 수 없다.

○ WHERE 조건식의 일반형식

SELECT	[ALL DISTINCT] {{컬럼명 [AS] 컬럼_별명},}* * }
FROM	테이블_리스트
[WHERE	컬럼명 비교연산자 SQL연산자 {숫자 문자 표현식} 컬럼명]

○ 연산자의 종류

- 1. 비교 연산자
- 2. SQL 연산자
- 3. 논리 연산자
 - ◆ 우선순위: 비교 연산자 및 SQL 연산자가 논리 연산자 보다 우선 처리됨.

구분	연산자	연산자의 의미
비교 연산자	=	같다.
	>	보다 크다.
	>=	보다 크거나 같다.
	<	보다 작다.
	<=	보다 작거나 같다.
SQL 연산자	BETWEEN a AND b	a와 b의 값 사이에 있으면 된다.(a와 b 값이 포함됨)
	IN (list)	리스트에 있는 값 중에서 어느 하나라도 일치하면 된다.
	LIKE '비교문자열'	비교문자열과 형태가 일치하면 된다.(%, _ 사용)
	IS NULL	NULL 값인 경우
논리 연산자	AND	앞에 있는 조건과 뒤에 오는 조건이 참(TRUE)이 되면 결과도 참(TRUE)이 된다. 즉, 앞의 조건과 뒤의 조건을 동시에 만족해야 한다.
	OR	앞의 조건이 참(TRUE)이거나 뒤의 조건이 참(TRUE)이 되어야 결과도 참(TRUE)이 된다. 즉, 앞뒤의 조건 중 하나만 참(TRUE)이면 된다.
	NOT	뒤에 오는 조건에 반대되는 결과를 되돌려 준다.
부정 비교 연산자	!=	같지 않다.
	^=	같지 않다.
	◇	같지 않다.(ISO 표준, 모든 운영체제에서 사용 가능)
	NOT 칼럼명 =	~와 같지 않다.
	NOT 칼럼명 >	~보다 크지 않다.
부정 SQL 연산자	NOT BETWEEN a AND b	a와 b의 값 사이에 있지 않다. (a, b 값을 포함하지 않는다)
	NOT IN (list)	list 값과 일치하지 않는다.
	IS NOT NULL	NULL 값을 갖지 않는다.

2.1 비교 연산자

- 예제 : =, >, <, >=, <=

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE   (TEAM_ID = 'K02' OR TEAM_ID = 'K07') AND  
          POSITION = 'MF' AND  
          HEIGHT >= 170 AND HEIGHT <= 180;
```

- 부정 비교 연산자 : !=, ^=, <>

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE   (TEAM ID = 'K02' OR TEAM ID = 'K07') AND  
          POSITION <> 'MF' AND  
          NOT (HEIGHT >= 170 AND HEIGHT <= 180);
```

2.2 SQL 연산자 : [NOT] IN (list) 연산자

- SQL 문장을 짧게 만들어주며, 성능 측면에서도 장점이 있음.
- 예제 : IN (list) 연산자

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
        HEIGHT 키  
FROM    PLAYER  
WHERE   TEAM ID IN ('K02','K07');
```

- 예제

```
SELECT  ENAME, JOB, DEPTNO  
FROM    EMP  
WHERE   (JOB, DEPTNO) IN (('MANAGER',20),('CLERK',30));
```

ENAME	JOB	DEPTNO
-----	-----	-----
JONES	MANAGER	20
JAMES	CLERK	30

2개의 행이 선택되었다.

○ 예 제

```
SELECT ENAME, JOB, DEPTNO
FROM EMP
WHERE JOB IN ('MANAGER','CLERK') AND DEPTNO IN (20,30);
```



```
(JOB = 'MANAGER' OR JOB = 'CLERK') AND
(DEPTNO = 20 OR DEPTNO = 30);
```

ENAME	JOB	DEPTNO
-----	-----	-----
SMITH	CLERK	20
JONES	MANAGER	20
BLAKE	MANAGER	30
ADAMS	CLERK	20
JAMES	CLERK	30

5개의 행이 선택되었다.

SQL 연산자 : [NOT] LIKE 연산자

- = 의 의미
- Wildcard 사용
 - % : 0개 문자 이상의 임의의 문자열
 - _ : 1개의 단일 문자
- 예제

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE    POSITION LIKE 'MF';
```

○ 예제

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE   PLAYER NAME LIKE '장%';
```

선수이름	포지션	백넘버	키
장성철	MF	27	176
장윤정	DF	17	173
장서연	FW	7	180
장재우	FW	12	172
장대일	DF	7	184
장기봉	FW	12	180
장철우	DF	7	172
장형석	DF	36	181
장경진	DF	34	184
장성욱	MF	19	174
장철민	MF	24	179
장경호	MF	39	174
장동현	FW	39	178

13개의 행이 선택되었다.

SQL 연산자 : [NOT] BETWEEN a AND b 연산자

○ 예제

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE    HEIGHT BETWEEN 170 AND 180;
```

선수이름	포지션	백넘버	키
장철우	DF	7	172
홍광철	DF	4	172
강정훈	MF	38	175
공오균	MF	22	177
정국진	MF	16	172
정동선	MF	9	170
최경규	MF	10	177
최내철	MF	24	177
배성재	MF	28	178
삼	MF	25	174
김관우	MF	8	175
⋮	⋮	⋮	⋮

259개의 행이 선택되었다.

SQL 연산자 : IS [NOT] NULL 연산자

○ NULL (ASCII 00)

- 값이 존재하지 않는 것을 표현할 때 사용.
- 어떤 값과도 비교할 수도 없음.
 - ◆ 특정 값보다 크다 혹은 적다라고 말할 수 없음.
 - ◆ 즉, ' ' (공백, ASCII 32)이나 0 (Zero, ASCII 48)과 달리 비교 자체가 불가능한 값임.

○ NULL의 특징

- NULL 값과의 수치 연산은 **NULL** 값을 리턴함.
- NULL 값과의 비교 연산은 **FALSE**를 리턴한다.
- NULL 값과의 비교 연산은 **IS NULL, IS NOT NULL** 이라는 정해진 문구를 사용해야 제대로 된 결과를 얻을 수 있음.

○ 예제

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, TEAM_ID  
FROM    PLAYER  
WHERE    POSITION = NULL;
```

◆ 결과 없음 (항상 FALSE 리턴)

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, TEAM_ID  
FROM    PLAYER  
WHERE    POSITION IS NULL;
```

선수이름 -----	포지션 -----	TEAM_ID -----
정학범		K08
안익수		K08
차상광		K08
3개의 행이 선택되었다.		

2.3 논리 연산자

- 우선순위
 - (), NOT, AND, OR 순서대로 처리
- 예제

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE   (TEAM_ID = 'K02' OR TEAM_ID = 'K07') AND  
          POSITION = 'MF' AND  
          HEIGHT >= 170 AND HEIGHT <= 180;
```

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,  
          HEIGHT 키  
FROM    PLAYER  
WHERE   TEAM_ID IN ('K02','K07') AND  
          POSITION = 'MF' AND  
          HEIGHT BETWEEN 170 AND 180;
```

☞ Pseudo Column : Oracle의 ROWNUM

○ 정의 및 용도

- SQL 처리 결과 집합의 각 행에 대해 임시로 부여되는 일련번호.
- 결과 집합으로 출력되는 행의 개수를 제한할 때 주로 사용.

○ 예제

```
SELECT  PLAYER_NAME  
FROM    PLAYER  
WHERE   ROWNUM <= 5;
```

☞ SQL Server의 TOP 절

- 용도
 - 결과 집합으로 출력되는 행의 개수를 제한.

- Syntax

TOP(Expression) [PERCENT] [WITH TIES]

- 예제

```
SELECT  TOP(1) PLAYER_NAME  
FROM    PLAYER
```

```
SELECT  TOP(N) PLAYER_NAME  
FROM    PLAYER
```

3. SELECT 문 – SQL 단일행 내장 함수

- SQL 내장 함수(Built-in function)

- 데이터 값을 간편하게 조작하고, SQL을 더욱 강력하게 해줌.
- 벤더별로 가장 큰 차이를 보이지만, 핵심적인 기능들은 이름이나 표현법이 다르더라도 대부분의 데이터베이스가 공통적으로 제공함.

- 종류

- 단일행 함수(single-row function) : 함수의 입력이 단일 행
 - ◆ 문자형, 숫자형, 날짜형, 변환형, NULL 관련 함수
- 다중행 함수(multi-row function) : 입력이 여러 행, 하나의 결과만 리턴
 - ◆ 집계 함수(aggregate function) : COUNT, SUM, AVG, MIN, MAX, STDDEV
 - ◆ 그룹 함수(group function) : ROLLUP, CUBE, GROUPING SETS
 - ◆ 윈도우 함수(window function) : 다양한 분석 기능

- Syntax

함수명 (컬럼 | 표현식 [, Arg1, Arg2, ...])

○ 단일행 함수의 특징

- SELECT, WHERE, ORDER BY 절에 사용 가능하다.
- 각 행들에 대해 개별적으로 작용하여 데이터 값들을 조작하고, 각각의 행에 대한 조작 결과를 리턴한다.
- 여러 인자(argument)를 입력해도 단 하나의 결과만 리턴한다.
- 함수의 인자로 상수, 변수, 표현식이 사용 가능하고, 하나의 인수를 가지는 경우도 있지만 여러 개의 인수를 가질 수도 있다.
- 특별한 경우가 아니면, 함수의 인자로 함수를 사용하는 함수의 중첩이 가능하다.

○ 단일행 함수의 종류

– Oracle 함수/SQL Server 함수 표시, '/' 없는 것은 공통 함수

종류	내용	함수의 예
문자형 함수	문자를 입력하면 문자나 숫자 값을 반환한다.	LOWER, UPPER, SUBSTR/SUBSTRING, LENGTH/ LEN, LTRIM, RTRIM, TRIM, ASCII,
숫자형 함수	숫자를 입력하면 숫자 값을 반환한다.	ABS, MOD, ROUND, TRUNC, SIGN, CHR/CHAR, CEIL/CEILING, FLOOR, EXP, LOG, LN, POWER, SIN, COS, TAN
날짜형 함수	DATE 타입의 값을 연산한다.	SYSDATE/GETDATE, EXTRACT/DATEPART, TO_NUMBER(TO_CHAR(d,'YYYY' 'MM' 'DD')) / YEAR MONTH DAY
변환형 함수	문자, 숫자, 날짜형 값의 데이터 타입을 변환한다.	TO_NUMBER, TO_CHAR, TO_DATE / CAST, CONVERT
NULL 관련 함수	NULL을 처리하기 위한 함수	NVL/ISNULL, NULLIF, COALESCE

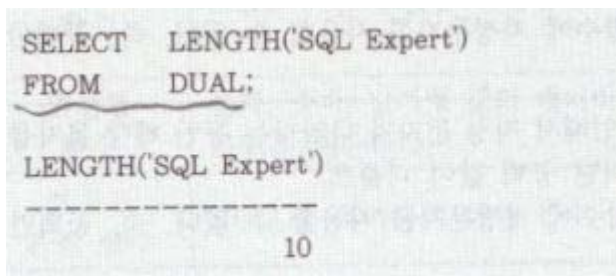
※ 주: Oracle함수/SQL Server함수 표시, '/' 없는 것은 공통 함수

3.1 문자형 함수

문자형 함수	함수 설명
LOWER(문자열)	문자열의 알파벳 문자를 소문자로 바꾸어 준다.
UPPER(문자열)	문자열의 알파벳 문자를 대문자로 바꾸어 준다.
ASCII(문자)	문자나 숫자를 ASCII 코드 번호로 바꾸어 준다.
CHR/CHAR(ASCII번호)	ASCII 코드 번호를 문자나 숫자로 바꾸어 준다.
CONCAT (문자열1, 문자열2)	Oracle, My SQL에서 유효한 함수이며 문자열1과 문자열2를 연결한다. 합성 연산자' '(Oracle)나 '+'(SQL Server)와 동일하다.
SUBSTR/SUBSTRING (문자열, m[, n])	문자열 중 m위치에서 n개의 문자 길이에 해당하는 문자를 돌려준다. n이 생략 되면 마지막 문자까지이다.
LENGTH/LEN(문자열)	문자열의 개수를 숫자값으로 돌려준다.
LTRIM (문자열 [, 지정문자])	문자열의 첫 문자부터 확인해서 지정 문자가 나타나면 해당 문자를 제거한다. (지정 문자가 생략되면 공백 값이 디폴트) SQL Server에서는 LTRIM 함수에 지정문자를 사용할 수 없다. 즉, 공백만 제거할 수 있다.
RTRIM (문자열 [, 지정문자])	문자열의 마지막 문자부터 확인해서 지정 문자가 나타나는 동안 해당 문자를 제거한다. (지정 문자가 생략되면 공백 값이 디폴트) SQL Server에서는 LTRIM 함수에 지정문자를 사용할 수 없다. 즉, 공백만 제거할 수 있다.
TRIM ([leading trailing both] 지정문자 FROM 문자열)	문자열에서 머리말, 꼬리말, 또는 양쪽에 있는 지정 문자를 제거한다. (leading trailing both 가 생략되면 both가 디폴트) SQL Server에서는 TRIM 함수에 지정문자를 사용할 수 없다. 즉, 공백만 제거할 수 있다.

- 예제
 - Oracle

```
SELECT LENGTH('SQL Expert')  
FROM DUAL
```

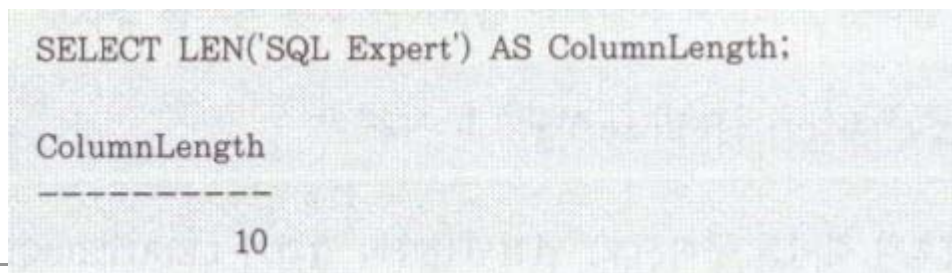


A screenshot of an Oracle SQL query and its result. The query is: `SELECT LENGTH('SQL Expert') FROM DUAL;`. The result shows a single row with the value 10. The text is slightly blurred and has a light blue background.

```
SELECT LENGTH('SQL Expert')  
FROM DUAL;  
  
LENGTH('SQL Expert')  
-----  
10
```

- SQL Server

```
SELECT LEN('SQL Expert') AS ColumnLength;
```



A screenshot of a SQL Server query and its result. The query is: `SELECT LEN('SQL Expert') AS ColumnLength;`. The result shows a single row with the value 10. The text is slightly blurred and has a light blue background.

```
SELECT LEN('SQL Expert') AS ColumnLength;  
  
ColumnLength  
-----  
10
```

○ 📌 Oracle의 DUAL 테이블

- 사용자 SYS가 소유하며 모든 사용자가 액세스 가능한 테이블.
- SELECT ~ FROM ~ 의 형식을 갖추기 위한 일종의 dummy table.
- DUMMY라는 VARCHAR2(1) 유형의 칼럼에 'X'라는 값이 들어 있는 행을 1건 포함함.

```
DESC DUAL;
```

칼럼	NULL 가능	데이터 유형
DUMMY		VARCHAR2(1)

```
SELECT * FROM DUAL;
```

DUMMY
X

1개의 행이 선택되었다.

- 📌 SQL Server는 SELECT 절만으로도 문장이 가능하므로, DUAL 테이블과 같은 dummy table이 필요가 없음.

-
- 예제
 - 공통

```
SELECT  PLAYER_ID, CONCAT(PLAYER NAME, ' 축구선수') AS 선수명
FROM    PLAYER;
```

- Oracle

```
SELECT  PLAYER_ID, PLAYER NAME || ' 축구선수' AS 선수명
FROM    PLAYER;
```

- SQL Server

```
SELECT  PLAYER_ID, PLAYER NAME + ' 축구선수' AS 선수명
FROM    PLAYER;
```

3.2 숫자형 함수

숫자형 함수	함수 설명
ABS(숫자)	숫자의 절대값을 돌려준다.
SIGN(숫자)	숫자가 양수인지, 음수인지 0인지를 구별한다.
MOD(숫자1, 숫자2)	숫자1을 숫자2로 나누어 나머지 값을 리턴한다. MOD 함수는 % 연산자로도 대체 가능함 (ex:7%3)
CEIL/CEILING(숫자)	숫자보다 크거나 같은 최소 정수를 리턴한다.
FLOOR(숫자)	숫자보다 작거나 같은 최대 정수를 리턴한다.
ROUND(숫자 [, m])	숫자를 소수점 m자리에서 반올림하여 리턴한다. m이 생략되면 디폴트 값은 0이다.
TRUNC(숫자 [, m])	숫자를 소수 m자리에서 잘라서 버린다. m이 생략되면 디폴트 값은 0이다. SQL SERVER에서 TRUNC 함수는 제공되지 않는다.
SIN, COS, TAN,...	숫자의 삼각함수 값을 리턴한다.
EXP(), POWER(), SQRT(), LOG(), LN()	숫자의 지수, 거듭 제곱, 제곱근, 자연 로그 값을 리턴한다.

※ 주: Oracle 함수/SQL Server 함수 표시 '/' 없는 것은 공통 함수

○ 예제

```
SELECT ENAME, ROUND(SAL/12,1), TRUNC(SAL/12,1)
FROM EMP
```

ENAME	ROUND(SAL/12,1)	TRUNC(SAL/12,1)
SMITH	66.7	66.6
ALLEN	133.3	133.3
WARD	104.2	104.1
JONES	247.9	247.9
MARTIN	104.2	104.1
BLAKE	237.5	237.5
CLARK	204.2	204.1
SCOTT	250	250
KING	416.7	416.6
TURNER	125	125
ADAMS	91.7	91.6
JAMES	79.2	79.1
FORD	250	250
MILLER	108.3	108.3

14개의 행이 선택되었다.

3.3 날짜형 함수

- 날짜는 내부적으로는 하나의 숫자 형식으로 변환하여 저장 .
 - 공간 절약, 산술 연산이 가능
 - 출력시에는 년/월/일/시/분/초 단위로 추출하여 변환해야 함.

[표 II -1-29] 단일행 날짜형 함수 종류

날짜형 함수	함수 설명
SYSDATE / GETDATE()	현재 날짜와 시각을 출력한다.
EXTRACT('YEAR' 'MONTH' 'DAY' from d) / DATEPART('YEAR' 'MONTH' 'DAY', d)	날짜 데이터에서 년/월/일 데이터를 출력할 수 있다. 시간/분/초도 가능함
TO_NUMBER(TO_CHAR(d,'YYYY')) / YEAR(d), TO_NUMBER(TO_CHAR(d,'MM')) / MONTH(d), TO_NUMBER(TO_CHAR(d,'DD')) / DAY(d)	날짜 데이터에서 년/월/일 데이터를 출력할 수 있다. Oracle EXTRACT YEAR/MONTH/DAY 옵션이나 SQL Server DEPART YEAR/MONTH/DAY 옵션과 같은 기능이다. TO_NUMBER 함수 제외시 문자형으로 출력됨

※ 주: Oracle함수/SQL Server함수 표시, '/' 없는 것은 공통 함수

-
- 예제: 시스템 시간 출력
 - Oracle : 날짜 변수

```
SELECT  SYSDATE  
FROM    DUAL;
```

```
SYSDATE  
-----  
12/07/18
```

- SQL Server : 날짜 관련 함수

```
SELECT  GETDATE() AS CURRENTTIME;
```

```
CURRENTTIME  
-----  
2012-07-18 13:10:02.047
```

-
- 예제: 아래 4개 문장 모두 동일한 기능
– Oracle

```
SELECT  ENAME, HIREDATE,  
         EXTRACT(YEAR FROM HIREDATE) 입사년도,  
         EXTRACT(MONTH FROM HIREDATE) 입사월,  
         EXTRACT(DAY FROM HIREDATE) 입사일  
FROM    EMP;
```

```
SELECT  ENAME, HIREDATE,  
         TO NUMBER(TO CHAR(HIREDATE, 'YYYY')) 입사년도,  
         TO NUMBER(TO CHAR(HIREDATE, 'MM')) 입사월,  
         TO NUMBER(TO CHAR(HIREDATE, 'DD')) 입사일  
FROM    EMP;
```

– SQL Server

```
SELECT  ENAME, HIREDATE,  
         DATEPART(YEAR, HIREDATE) 입사년도,  
         DATEPART(MONTH, HIREDATE) 입사월,  
         DATEPART(DAY, HIREDATE) 입사일  
FROM    EMP;
```

```
SELECT  ENAME, HIREDATE,  
         YEAR(HIREDATE) 입사년도,  
         MONTH(HIREDATE) 입사월,  
         DAY(HIREDATE) 입사일  
FROM    EMP;
```

ENAME	HIREDATE	입사년도	입사월	입사일
SMITH	1980-12-17	1980	12	17
ALLEN	1981-02-20	1981	2	20
WARD	1981-02-22	1981	2	22
JONES	1981-04-02	1981	4	2
MARTIN	1981-09-28	1981	9	28
BLAKE	1981-05-01	1981	5	1
CLARK	1981-06-09	1981	6	9
SCOTT	1987-07-13	1987	7	13
KING	1981-11-17	1981	11	17
TURNER	1981-09-08	1981	9	8
ADAMS	1987-07-13	1987	7	13
JAMES	1981-12-03	1981	12	3
FORD	1981-12-03	1981	12	3
MILLER	1982-01-23	1982	1	23

14개의 행이 선택되었다.

3.4 변환형 함수

○ 데이터 유형의 변환

- 명시적 변화 : 변환형 함수를 사용
- 묵시적 변환 : DBMS가 자동으로 변환,
성능 저하 및 에러 발생 가능성

변환형 함수 - Oracle	함수 설명
TO_NUMBER(문자열)	alphanumeric 문자열을 숫자로 변환한다.
TO_CHAR(숫자 날짜 [, FORMAT])	숫자나 날짜를 주어진 FORMAT 형태로 문자열 타입으로 변환한다.
TO_DATE(문자열 [, FORMAT])	문자열을 주어진 FORMAT 형태로 날짜 타입으로 변환한다.
변환형 함수 - SQL Server	함수 설명
CAST (expression AS data_type [(length)])	expression을 목표 데이터 유형으로 변환한다.
CONVERT (data_type [(length)], expression [, style])	expression을 목표 데이터 유형으로 변환한다.

-
- 예제
 - Oracle

```
SELECT  TO_CHAR(SYSDATE, 'YYYY/MM/DD') 날짜,  
         TO_CHAR(SYSDATE, 'YYYY, MON, DAY') 문자형  
FROM    DUAL;
```

날짜	문자형
2012/07/19	2012. 7월 , 월요일

- SQL Server

```
SELECT  CONVERT(VARCHAR(10), GETDATE(), 111) AS CURRENTDATE;
```

CURRENTDATE
2012/07/19

○ 예제
– Oracle

```
SELECT TO_CHAR(123456789/1200, '$999,999,999.99') 환율반영달러,  
        TO_CHAR(123456789, 'L999,999,999') 원화  
FROM    DUAL;
```

☞ L : 로컬 화폐 단위

환율반영달러	원화
-----	-----
\$102,880.66	₩123,456,789

두 번째 칼럼의 L999에서 L은 로컬 화폐 단위를 의미한다.

- 예제
 - Oracle

```
SELECT TEAM_ID, TO NUMBER(ZIP_CODE1,'999') +  
          TO NUMBER(ZIP_CODE2,'999') 우편번호합  
FROM    TEAM;
```

- SQL Server

```
SELECT TEAM_ID, CAST(ZIP_CODE1 AS INT) + CAST(ZIP_CODE2 AS INT)  
          우편번호합  
FROM    TEAM;
```

TEAM_ID	우편번호합
K05	750
K08	592
K03	840
K07	554
K09	359
K04	838
K11	333
K01	742
K10	331
K02	660
K12	869
K06	620
K13	777
K14	1221
K15	1665

ANSI/S 15개의 행이 선택되었다.

3.5 NULL 관련 함수

○ NULL 값의 특징

- 테이블을 생성할 때 NOT NULL 또는 PRIMARY KEY로 정의되지 않은 모든 데이터 유형은 널 값을 포함할 수 있다.
- 널 값을 포함하는 연산의 경우, 결과 값도 널 값이다.

○ NULL 값의 처리 방법

- 결과값을 NULL이 아닌 다른 값을 얻고자 할 때 NVL() / ISNULL() 함수를 사용한다.
- 널 값의 대상이 숫자 유형 데이터인 경우는 주로 0(Zero)으로, 문자 유형 데이터인 경우는 blank 보다는 'x' 같이 해당 시스템에서 의미 없는 문자로 바꾸는 경우가 많다.

일반형 함수	함수 설명
NVL(표현식1, 표현식2) / ISNULL(표현식1, 표현식2)	표현식1의 결과값이 NULL이면 표현식2의 값을 출력한다. 단, 표현식1과 표현식2의 결과 데이터 타입이 같아야 한다. NULL 관련 가장 많이 사용되는 함수이므로 상당히 중요하다.
NULLIF(표현식1, 표현식2)	표현식1이 표현식2와 같으면 NULL을, 같지 않으면 표현식1을 리턴한다.
COALESCE(표현식1, 표현식2,)	임의의 개수 표현식에서 NULL이 아닌 최초의 표현식을 나타낸다. 모든 표현식이 NULL이라면 NULL을 리턴한다.

Oracle의 NVL() 함수 (NULL Value Logic)

- Syntax

NVL (표현식, 'NULL일 때 대체값')

- 예제

```
SELECT  NVL(NULL, 'NVL-OK') AS NVL_TEST
FROM    DUAL;
```

NVL_TEST

NVL-OK

1개의 행이 선택되었다.

```
SELECT  NVL('Not-Null', 'NVL-OK') AS NVL_TEST
FROM    DUAL;
```

NVL_TEST

Not-Null

1개의 행이 선택되었다.

SQL Server의 ISNULL() 함수

○ Syntax

ISNULL (EXPR, string)

– EXPR이 널 값이면 string을, 널 값이 아니면 EXPR의 결과를 리턴함

○ 예제

```
SELECT  ISNULL(NULL, 'NVL-OK') AS ISNULL_TEST;
```

ISNULL_TEST

NVL-OK

1개의 행이 선택되었다.

```
SELECT  ISNULL('Not-Null', 'NVL-OK') AS ISNULL_TEST;
```

ISNULL_TEST

Not-Null

1개의 행이 선택되었다.

예제 : NVL() 함수 / ISNULL() 함수

○ 예제 1

– Oracle

```
SELECT  PLAYER_NAME 선수명, POSITION, NVL(POSITION,'없음') 포지션
FROM    PLAYER
WHERE    TEAM_ID = 'K08';
```

– SQL Server

```
SELECT  PLAYER_NAME 선수명, POSITION, ISNULL(POSITION,'없음') 포지션
FROM    PLAYER
WHERE    TEAM_ID = 'K08';
```

– 공통 : NVL/ISNULL 함수는 CASE 문장으로 표현 가능

```
SELECT  PLAYER_NAME 선수명, POSITION,
          CASE WHEN POSITION IS NULL           THEN  '없음'
          ELSE    POSITION
          END AS 포지션
FROM    PLAYER
WHERE    TEAM_ID = 'K08';
```

선수명	POSITION	포지션
차경복	DF	DF
정학범		없음
안익수		없음
차상광		없음
권찬수	GK	GK
정경두	GK	GK
정해운	GK	GK
양영민	GK	GK
가이모토	DF	DF
정두영	DF	DF
정명휘	DF	DF
정영철	DF	DF
박치국	MF	MF
정상식	MF	MF
서관수	FW	FW
김성운	FW	FW
김정운	FW	FW
장동현	FW	FW
⋮	⋮	⋮

45개의 행이 선택되었다.

○ 예제2 (Oracle) : 널 값이 포함된 연산

```
SELECT  ENAME 사원명, SAL 월급, COMM 커미션,  
          (SAL * 12) + COMM 연봉A, (SAL * 12) + NVL(COMM,0) 연봉B  
FROM    EMP;
```

사원명	월급	커미션	연봉A	연봉B
SMITH	800			9600
ALLEN	1600	300	19500	19500
WARD	1250	500	15500	15500
JONES	2975			35700
MARTIN	1250	1400	16400	16400
BLAKE	2850			34200
CLARK	2450			29400
SCOTT	3000			36000
KING	5000			60000
TURNER	1500	0	18000	18000
ADAMS	1100			13200
JAMES	950			11400
FORD	3000			36000
MILLER	1300			15600

14개의 행이 선택되었다.

☞ NVL() / ISNULL() 함수 사용시의 주의 사항

- 예제 (Oracle) : 존재하는 튜플의 “속성 값인 널”만을 리턴하는 경우
– ‘KING’의 매니저는 NULL (‘KING’이 사장임).

```
SELECT  MGR
FROM    EMP
WHERE   ENAME='KING';
```

MGR

1개의 행이 선택되었다.

```
SELECT  NVL(MGR,9999) AS MGR
FROM    EMP
WHERE   ENAME='KING';
```

MGR

9999

1개의 행이 선택되었다.

- 예제 (Oracle) : 조건에 맞는 튜플이 없는 (결과가 “공집합”인) 경우
 - ‘JSC’ 튜플이 존재하지 않는 경우:
 - ◆ MGR에 널 값이 할당되지 않으므로, NVL() 함수를 적용하여 9999를 출력할 수 없음.

```
SELECT MGR
FROM EMP
WHERE ENAME='JSC';
```

데이터를 찾을 수 없다.

```
SELECT NVL(MGR,9999) AS MGR
FROM EMP
WHERE ENAME='JSC';
```

데이터를 찾을 수 없다.

/* NVL의 잘못된 시도임 */

-
- 이 경우는 NVL 함수 대신, 적절한 집계 함수를 대신 적용함.
 - ◆ 다른 함수와 달리, 집계 함수와 Scalar subquery의 경우는 인수의 결과 값이 공집합인 경우에도 NULL을 출력함. 따라서 NVL() 함수 적용 가능.

```
SELECT MAX(MGR) AS MGR
FROM EMP
WHERE ENAME='JSC';
```

MGR

1개의 행이 선택되었다.

```
SELECT NVL(MAX(MGR),9999) AS MGR
FROM EMP
WHERE ENAME='JSC';
```

MGR

9999

1개의 행이 선택되었다.

NULLIF() 함수

○ Syntax

```
NULLIF (EXPR1, EXPR2)
```

– “EXPR1 = EXPR2” 이면 널 값을 리턴함.

○ 예제

```
SELECT  ENAME, EMPNO, MGR, NULLIF(MGR,7698) NUIF  
FROM    EMP;
```

– NULLIF() 함수는 CASE 문장으로 표현 가능

```
SELECT  ENAME, EMPNO, MGR,  
        CASE    WHEN    MGR = 7698      THEN    NULL  
              ELSE    MGR  
        END NUIF  
FROM    EMP;
```

ENAME	EMPNO	MGR	NUIF
SMITH	7369	7902	7902
ALLEN	7499	7698	
WARD	7521	7698	
JONES	7566	7839	7839
MARTIN	7654	7698	
BLAKE	7698	7839	7839
CLARK	7782	7839	7839
SCOTT	7788	7566	7566
KING	7839		
TURNER	7844	7698	
ADAMS	7876	7788	7788
JAMES	7900	7698	
FORD	7902	7566	7566
MILLER	7934	7782	7782

14개의 행이 선택되었다.

COALESCE 함수

○ Syntax

COALESCE (*EXPR1*, *EXPR2*, ...)

- 임의 개수의 **EXPR**에서, 널이 아닌 최초의 **EXPR**을 나타냄.
모든 **EXPR**이 널이라면, 널을 리턴함.

○ 예제

```
SELECT  ENAME, COMM, SAL, COALESCE(COMM, SAL) COAL
FROM    EMP;
```

- **COALESCE** 함수는 n 개의 중첩된 **CASE** 문장으로 표현 가능

```
SELECT  ENAME, COMM, SAL,
         CASE    WHEN  COMM IS NOT NULL      THEN  COMM
              ELSE    (CASE    WHEN  SAL IS NOT NULL THEN  SAL
                        ELSE    NULL
                        END)
         END COAL
FROM    EMP;
```

ENAME	COMM	SAL	COAL
SMITH		800	800
ALLEN	300	1600	300
WARD	500	1250	500
JONES		2975	2975
MARTIN	1400	1250	1400
BLAKE		2850	2850
CLARK		2450	2450
SCOTT		3000	3000
KING		5000	5000
TURNER	0	1500	0
ADAMS		1100	1100
JAMES		950	950
FORD		3000	3000
MILLER		1300	1300

14개의 행이 선택되었다.

3.6 CASE Expression

- 특징

- IF-THEN-ELSE 논리와 유사, SQL의 비교 연산 기능을 보완하는 역할
- 중첩이 가능함.


- Syntax

- Simple case expression : Oracle의 DECODE() 함수와 같은 기능.

```
CASE      EXPR  
      {WHEN COMPARISON_EXPR      THEN  RETURN_EXPR}+  
      ELSE      표현식  
END
```

- Searched case expression

```
CASE  
      {WHEN CONDITION_EXPR      THEN  RETURN_EXPR}+  
      ELSE      표현식  
END
```

 **CONDITION_EXPR** : "EXPR 비교연산자 **COMPARISON_EXPR**"

Simple Case Expression

○ 예제

```
SELECT LOC,
CASE LOC
WHEN 'NEW YORK' THEN 'EAST'
WHEN 'BOSTON' THEN 'EAST'
WHEN 'CHICAGO' THEN 'CENTER'
WHEN 'DALLAS' THEN 'CENTER'
ELSE 'ETC'
END
AS AREA
FROM DEPT;
```

LOC	AREA
NEW YORK	EAST
DALLAS	CENTER
CHICAGO	CENTER
BOSTON	EAST

4개의 행이 선택되었다.

Searched Case Expression

○ 예제

```
SELECT  ENAME,  
        CASE WHEN SAL >= 3000 THEN 'HIGH'  
              WHEN SAL >= 1000 THEN 'MID'  
              ELSE 'LOW'  
        END AS SALARY_GRADE  
FROM EMP;
```

ENAME	SALARY_GRADE
SMITH	LOW
ALLEN	MID
WARD	MID
JONES	MID
MARTIN	MID
BLAKE	MID
CLARK	MID
SCOTT	HIGH
KING	HIGH
TURNER	MID
ADAMS	MID
JAMES	LOW
FORD	HIGH
MILLER	MID

14개의 행이 선택되었다.

중첩된 Case Expressions

○ 예제

```
SELECT  ENAME, SAL,  
        CASE WHEN SAL >= 2000 THEN 1000  
              ELSE (CASE WHEN SAL >= 1000 THEN 500  
                        ELSE 0  
                      END)  
        END AS BONUS  
FROM    EMP;
```

ENAME	SAL	BONUS
SMITH	800	0
ALLEN	1600	500
WARD	1250	500
JONES	2975	1000
MARTIN	1250	500
BLAKE	2850	1000
CLARK	2450	1000
SCOTT	3000	1000
KING	5000	1000
TURNER	1500	500
ADAMS	1100	500
JAMES	950	0
FORD	3000	1000
MILLER	1300	500

14개의 행이 선택되었다.

Oracle의 DECODE() 함수

- Simple case expression의 간단한 표현
- Syntax

DECODE (COMPARISON_EXPR, RET_EXPR1, ..., RET_EXPRn, 디폴트수식)

4. SELECT 문 – GROUP BY 절, HAVING 절

- 집계 함수(Aggregate function)
 - Syntax

집계함수명 ([DISTINCT|ALL] 컬럼명|표현식)

[표 II-1-36] 집계 함수의 종류

집계 함수	사용 목적
COUNT(*)	NULL 값을 포함한 행의 수를 출력한다.
COUNT(표현식)	표현식의 값이 NULL 값인 것을 제외한 행의 수를 출력한다.
SUM([DISTINCT ALL] 표현식)	표현식의 NULL 값을 제외한 합계를 출력한다.
AVG([DISTINCT ALL] 표현식)	표현식의 NULL 값을 제외한 평균을 출력한다.
MAX([DISTINCT ALL] 표현식)	표현식의 최대값을 출력한다. (문자, 날짜 데이터 타입도 사용가능)
MIN([DISTINCT ALL] 표현식)	표현식의 최소값을 출력한다. (문자, 날짜 데이터 타입도 사용가능)
STDDEV([DISTINCT ALL] 표현식)	표현식의 표준 편차를 출력한다.
VARIAN([DISTINCT ALL] 표현식)	표현식의 분산을 출력한다.
기타 통계 함수	벤더별로 다양한 통계식을 제공한다.

4.1 GROUP BY 절

- 튜플들을 그룹들로 분류하여, 그룹 별 통계 정보를 얻을 때 사용함.
- 특징
 - **GROUP BY 절을 통해 그룹별 기준**을 정한 후, **SELECT 절에 집계 함수**를 사용한다.
 - 집계 함수는 **NULL 값을 가진 행을 제외**하고 수행한다.
 - GROUP BY 절에서는 SELECT 절과는 달리 컬럼 별명(alias)을 사용할 수 없다.
 - WHERE 절은 전체 데이터를 그룹별로 나누기 전에 튜플들을 미리 제거한다. 따라서 **WHERE 절은 GROUP BY 절보다 먼저** 수행된다.
 - 집계 함수는 **WHERE 절에는 올 수 없다.**

```
SELECT      [ALL|DISTINCT] {{컬럼명 [[AS] 컬럼_별명],}* | * }  
FROM        테이블_리스트  
[WHERE      조건식]  
[GROUP BY   컬럼명 [HAVING 그룹조건식]]  
[ORDER BY   {컬럼명|컬럼_별명|컬럼_순서 [ASC|DESC],}*];
```

○ 예제

```
SELECT POSITION 포지션, COUNT(*) 인원수, COUNT(HEIGHT) 키대상,
        MAX(HEIGHT) 최대키, MIN(HEIGHT) 최소키,
        ROUND(AVG(HEIGHT),2) 평균키
FROM    PLAYER
GROUP BY POSITION;
```

포지션	인원수	키대상	최대키	최소키	평균키
	3	0			
GK	43	43	196	174	186.26
DF	172	142	190	170	180.21
FW	100	100	194	168	179.91
MF	162	162	189	165	176.31

5개의 행이 선택되었다.

○ GROUP BY 절의 실행

- GROUP BY 절을 사용하면, “그룹핑 기준 컬럼”과 “집계 함수에서 사용되는 숫자형 데이터 컬럼”들의 튜플 집합을 새로 만들고, **그 이외의 컬럼들은 메모리에서 제거함.**
- 앞의 예의 경우, 메모리에는 POSITION, HEIGHT 만으로 구성되는 임시 PLAYER 테이블이 저장됨.

```
SELECT  POSITION 포지션, COUNT(*) 인원수, COUNT(HEIGHT) 키대상,  
          MAX(HEIGHT) 최대키, MIN(HEIGHT) 최소키,  
          ROUND(AVG(HEIGHT),2) 평균키  
FROM    PLAYER  
GROUP BY POSITION;
```

- 따라서 이들 이외의 컬럼을 SELECT 절에서 사용하면 에러임.

```
SELECT  POSITION 포지션, COUNT(*) 인원수, COUNT(HEIGHT) 키대상,  
          MAX(HEIGHT) 최대키, MIN(HEIGHT) 최소키,  
          ROUND(AVG(HEIGHT),2) 평균키, TEAM ID  
FROM    PLAYER  
GROUP BY POSITION;                                // Error !! (GROUP BY)
```

```
SELECT  JOB, SAL
FROM    EMP
GROUP BY JOB   HAVING COUNT(*) > 0
ORDER BY SAL;                                     // Error !! (GROUP BY)
```

4.2 HAVING 절

- HAVING 절은 GROUP BY 절의 기준 항목이나 그룹의 집계 함수를 이용할 조건을 표시함.
- 특징
 - GROUP BY 절에 의해 그룹별로 만들어진 튜플 중, HAVING 절에서 추가의 제한 조건을 두어 이 조건을 만족하는 튜플만 출력한다.
 - HAVING 절은 일반적으로 GROUP BY 절 뒤에 위치한다.

```
SELECT      [ALL|DISTINCT] {{컬럼명 [[AS] 컬럼_별명],}* | * }  
FROM        테이블_리스트  
[WHERE      조건식]  
[GROUP BY   컬럼명 [HAVING 그룹조건식]]  
[ORDER BY   {컬럼명|컬럼_별명|컬럼_순서 [ASC|DESC],}*];
```


○ 예제

```
SELECT POSITION 포지션, ROUND(AVG(HEIGHT),2) 평균키
FROM PLAYER
GROUP BY POSITION HAVING AVG(HEIGHT) >= 180;
```

포지션	평균키
-----	-----
GK	186.26
DF	180.21

2개의 행이 선택되었다.

- ☞ **WHERE절에 집계함수를 사용할 수 없음**
(WHERE 절의 조건은 각각의 튜플에게 적용되는 것이어야 함.)

```
SELECT POSITION 포지션, ROUND(AVG(HEIGHT),2) 평균키
FROM PLAYER
WHERE AVG(HEIGHT) >= 180 /* 에러 */
GROUP BY POSITION;
```

- 예제: 아래 두 질의는 동일한 결과
 - 그러나 **WHERE** 절에 먼저 조건을 적용하여, 대상 튜플 수를 줄이는 것이 더 효율적임.

```
SELECT TEAM_ID 팀ID, COUNT(*) 인원수
FROM    PLAYER
WHERE    TEAM_ID IN ('K09', 'K02')
GROUP BY TEAM_ID;
```

팀ID	인원수
-----	-----
K02	49
K09	49

2개의 행이 선택되었다.

```
SELECT TEAM_ID 팀ID, COUNT(*) 인원수
FROM    PLAYER
GROUP BY TEAM_ID HAVING TEAM_ID IN ('K09', 'K02');
```

예제: 집계 함수(CASE 문) ~ GROUP BY ~ 형태의 활용 1

- 질의 : 부서별로 월별 입사자의 평균 급여를 구하라. (월별 데이터)
- Step 1 (Oracle)

```
SELECT ENAME, DEPTNO, EXTRACT(MONTH FROM HIREDATE) 입사월, SAL  
FROM EMP;
```

ENAME	DEPTNO	입사월	SAL
-----	-----	-----	-----
SMITH	20	12	800
ALLEN	30	2	1600
WARD	30	2	1250
JONES	20	4	2975
MARTIN	30	9	1250
BLAKE	30	5	2850
CLARK	10	6	2450
SCOTT	20	7	3000
KING	10	11	5000
TURNER	30	9	1500
ADAMS	20	7	1100
JAMES	30	12	950
FORD	20	12	3000
MILLER	10	1	1300

14개의 행이 선택되었다.
ANSI/ISO SQL

○ Step 2 (Oracle) : simple case expression 사용

```
SELECT  ENAME, DEPTNO,  
        CASE MONTH WHEN 1 THEN SAL END M01,  
        CASE MONTH WHEN 2 THEN SAL END M02,  
        CASE MONTH WHEN 3 THEN SAL END M03,  
        CASE MONTH WHEN 4 THEN SAL END M04,  
        CASE MONTH WHEN 5 THEN SAL END M05,  
        CASE MONTH WHEN 6 THEN SAL END M06,  
        CASE MONTH WHEN 7 THEN SAL END M07,  
        CASE MONTH WHEN 8 THEN SAL END M08,  
        CASE MONTH WHEN 9 THEN SAL END M09,  
        CASE MONTH WHEN 10 THEN SAL END M10,  
        CASE MONTH WHEN 11 THEN SAL END M11,  
        CASE MONTH WHEN 12 THEN SAL END M12  
FROM    (SELECT  ENAME, DEPTNO,  
            EXTRACT(MONTH FROM HIREDATE) MONTH, SAL  
        FROM EMP);
```


ENAME	DEPTNO	M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12
SMITH	20												800
ALLEN	30		1600										
WARD	30		1250										
JONES	20				2975								
MARTIN	30									1250			
BLAKE	30					2850							
CLARK	10						2450						
SCOTT	20							3000					
KING	10											5000	
TURNER	30									1500			
ADAMS	20							1100					
JAMES	30												950
FORD	20												3000
MILLER	10	1300											

14개의 행이 선택되었다.

- Step 3 (Oracle)

```
SELECT  DEPTNO,  
          AVG(CASE MONTH WHEN 1 THEN SAL END) M01,  
          AVG(CASE MONTH WHEN 2 THEN SAL END) M02,  
          AVG(CASE MONTH WHEN 3 THEN SAL END) M03,  
          AVG(CASE MONTH WHEN 4 THEN SAL END) M04,  
          AVG(CASE MONTH WHEN 5 THEN SAL END) M05,  
          AVG(CASE MONTH WHEN 6 THEN SAL END) M06,  
          AVG(CASE MONTH WHEN 7 THEN SAL END) M07,  
          AVG(CASE MONTH WHEN 8 THEN SAL END) M08,  
          AVG(CASE MONTH WHEN 9 THEN SAL END) M09,  
          AVG(CASE MONTH WHEN 10 THEN SAL END) M10,  
          AVG(CASE MONTH WHEN 11 THEN SAL END) M11,  
          AVG(CASE MONTH WHEN 12 THEN SAL END) M12  
FROM    (SELECT  ENAME, DEPTNO,  
              EXTRACT(MONTH FROM HIREDATE) MONTH, SAL  
            FROM EMP)  
GROUP  BY DEPTNO;
```

DEPTNO	M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12
30		1425			2850				1375			950
20				2975			2050					1900
10	1300					2450					5000	

3개의 행이 선택되었다.

- Step 3 (Oracle) : Simple case expression 대신 DECODE 함수 사용

```
SELECT  DEPTNO,  
          AVG(DECODE(MONTH,1,SAL)) M01,  
          AVG(DECODE(MONTH,2,SAL)) M02,  
          AVG(DECODE(MONTH,3,SAL)) M03,  
          AVG(DECODE(MONTH,4,SAL)) M04,  
          AVG(DECODE(MONTH,5,SAL)) M05,  
          AVG(DECODE(MONTH,6,SAL)) M06,  
          AVG(DECODE(MONTH,7,SAL)) M07,  
          AVG(DECODE(MONTH,8,SAL)) M08,  
          AVG(DECODE(MONTH,9,SAL)) M09,  
          AVG(DECODE(MONTH,10,SAL)) M10,  
          AVG(DECODE(MONTH,11,SAL)) M11,  
          AVG(DECODE(MONTH,12,SAL)) M12  
FROM    (SELECT  ENAME, DEPTNO,  
            EXTRACT(MONTH FROM HIREDATE) MONTH, SAL  
          FROM EMP)  
GROUP BY DEPTNO;
```


☞ 집계 함수와 NULL 처리

- 집계 함수에서는 NULL 값을 갖는 튜플(행)은 제외하고 실행 함.
 - 따라서 집계 함수 내에서 NVL / ISNULL 함수는 피해야 함. (계산량만 증가)

```
SUM(NVL(SAL,0)) 혹은 SUM(ISNULL(SAL,0))
```

- 집계 함수의 결과는 다음과 같이 NULL 값 대신 0을 디스플레이 하는 것은 바람직함.

```
NVL(SUM(SAL),0) 혹은 ISNULL(SUM(SAL),0)
```

- CASE에서 ELSE 절을 생략하게 되면 Default 값이 NULL이다.
 - 만일 아래처럼 ELSE 절에서 0(Zero)을 지정하면, 불필요하게 0이 SUM 연산에 사용되므로 자원의 사용이 많아진다. 가능하면 ELSE 절의 상수값을 지정하지 않거나, ELSE 절을 작성하지 않도록 한다

```
SUM(CASE MONTH WHEN 1 THEN SAL ELSE 0 END)
```

예제: 집계 함수(CASE 문) ~ GROUP BY ~ 형태의 활용 2

- 질의 : 팀별로 포지션별(FW,MF,DF,GK) 인원수, 그리고 팀별 전체 인원수를 구하라. 단 데이터가 없는 경우는 0으로 표시한다.
- Step 1 (Oracle)

```
SELECT TEAM_ID,  
       NVL(SUM(CASE POSITION WHEN 'FW' THEN 1 ELSE 0 END) FW,  
       NVL(SUM(CASE POSITION WHEN 'MF' THEN 1 ELSE 0 END) MF,  
       NVL(SUM(CASE POSITION WHEN 'DF' THEN 1 ELSE 0 END) DF,  
       NVL(SUM(CASE POSITION WHEN 'GK' THEN 1 ELSE 0 END) GK,  
       COUNT(*) SUM  
FROM   PLAYER  
GROUP BY TEAM_ID;
```

○ Step 2 (Oracle)

```

SELECT TEAM_ID,
       NVL(SUM(CASE POSITION WHEN 'FW' THEN 1 END) FW,
       NVL(SUM(CASE POSITION WHEN 'MF' THEN 1 END) MF,
       NVL(SUM(CASE POSITION WHEN 'DF' THEN 1 END) DF,
       NVL(SUM(CASE POSITION WHEN 'GK' THEN 1 END) GK,
       COUNT(*) SUM
FROM   PLAYER
GROUP BY TEAM_ID;

```

TEAM_ID	FW	MF	DF	GK	SUM
K14	0	1	1	0	2
K06	11	11	20	4	46
K13	1	0	1	1	3
K15	1	1	1	0	3
K02	10	18	17	4	49
K12	1	0	1	0	2
K04	13	11	18	4	46
K03	6	15	23	5	49
K07	9	22	16	4	51
K05	10	19	17	5	51
K08	8	15	15	4	45
K11	1	1	1	0	3
K01	12	15	13	5	45
K10	5	15	13	3	36
K09	12	18	15	4	49

15개의 행이 선택되었다.

5. SELECT 문 – ORDER BY 절

- 테이블을 특정 컬럼을 기준으로 정렬함.
 - 오름차순(ascending), 내림차순(descending)
- 컬럼명 대신 컬럼 별명이나 컬럼 순서를 나타내는 정수도 가능.

```
SELECT      [ALL|DISTINCT] {{컬럼명 [[AS] 컬럼_별명],}* | * }  
FROM        테이블_리스트  
[WHERE      조건식]  
[GROUP BY   컬럼명 [HAVING 그룹조건식]]  
[ORDER BY   {컬럼명|컬럼_별명|컬럼_순서 [ASC|DESC],}*];
```

- 특징
 - 디폴트는 오름차순(ASC)이다.
 - Oracle에서는 NULL 값을 가장 큰 값으로 간주함.
 - SQL Server에서는 NULL 값을 가장 작은 값으로 간주함.

○ 예제 (Oracle)

```
SELECT  PLAYER_NAME 선수명, POSITION 포지션, BACK_NO 백넘버  
FROM    PLAYER  
ORDER BY 포지션 DESC;
```

선수명	포지션	백넘버
정학범	---	---
차상광	---	---
안익수	---	---
백영철	MF	22
조태용	MF	7
올리베	MF	29
김리네	MF	26
자스민	MF	33
⋮	⋮	⋮

480개의 행이 선택되었다.

○ 예제

```
SELECT  PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버,
          HEIGHT 키
FROM    PLAYER
WHERE    HEIGHT IS NOT NULL
ORDER    BY HEIGHT DESC, BACK_NO;
```

선수명	포지션	백넘버	키
서동명	GK	21	196
권정혁	GK	1	195
김석	FW	<u>20</u>	194
정경두	GK	<u>41</u>	194
이현	GK	1	192
황연석	FW	16	192
미트로	FW	19	192
김대희	GK	31	192
조의손	GK	44	192
김창민	GK	1	191
우성용	FW	22	191
최동석	GK	1	190
샤샤	FW	10	190
⋮	⋮	⋮	⋮

ANSI/ISO 447개의 행이 선택되었다.



국민대학교
KOOKMIN UNIVERSITY

○ 예제

```
SELECT  PLAYER_NAME 선수명, POSITION 포지션, BACK_NO 백넘버,  
FROM    PLAYER  
WHERE    BACK_NO IS NOT NULL  
ORDER    BY 3 DESC, 2, 1;
```

선수명	포지션	백넘버
-----	-----	-----
뚜파	FW	99
<u>쿠키</u>	FW	99
황승주	DF	98
무스타파	MF	77
다보	FW	63
다오	DF	61
김충호	GK	60
최동우	GK	60
최주호	GK	51
안동원	DF	49
오재진	DF	49
...

439개의 행이 선택되었다.

- 예제: 아래 세 경우는 모두 동일

```
SELECT DNAME, LOC AREA, DEPTNO  
FROM DEPT  
ORDER BY DNAME, LOC, DEPTNO DESC;
```

```
SELECT DNAME, LOC AREA, DEPTNO  
FROM DEPT  
ORDER BY DNAME, AREA, DEPTNO DESC;
```

```
SELECT DNAME, LOC AREA, DEPTNO  
FROM DEPT  
ORDER BY 1, AREA, 3 DESC;
```

DNAME	AREA	DEPTNO
ACCOUNTING	NEW YORK	10
OPERATIONS	BOSTON	40
RESEARCH	DALLAS	20
SALES	CHICAGO	30

4개의 행이 선택되었다.

GROUP BY와 ORDER BY 동시 사용시 주의사항

- ORDER BY 절에는 SELECT 목록에 나타나지 않은 문자형 컬럼이 포함될 수 있음.
 - DBMS가 데이터를 메모리에 올릴 때 튜플 단위로 모든 칼럼을 가져오게 되므로, SELECT 절에서 일부 칼럼만 선택하더라도 ORDER BY 절에서 메모리에 올라와 있는 다른 칼럼의 데이터를 사용할 수 있기 때문임.

```
SELECT EMPNO, ENAME  
FROM EMP  
ORDER BY MGR; // No Error !!
```

- 그러나 **GROUP BY 절을 같이 사용**하면, SELECT 목록에 나타나지 않은 문자형 컬럼이 ORDER BY에 포함될 수 없음.
 - GROUP BY 절을 사용하면, “**그룹핑 기준 컬럼**”과 “**집계 함수에서 사용되는 숫자형 데이터 컬럼**”들의 튜플 집합을 새로 만들고, 그 이외의 컬럼들은 메모리에서 제거하기 때문임.

```

SELECT JOB, SAL
FROM EMP
GROUP BY JOB HAVING COUNT(*) > 0
ORDER BY SAL;                                     // Error !! (GROUP BY)

```

```

SELECT JOB
FROM EMP
GROUP BY JOB HAVING COUNT(*) > 0
ORDER BY SAL;                                     // Error !! (ORDER BY)

```

```

SELECT JOB
FROM EMP
GROUP BY JOB HAVING COUNT(*) > 0
ORDER BY MAX(EMPNO), MAX(MGR), SUM(SAL), COUNT(DEPTNO),
MAX(HIREDATE);                                     // No Error !!

```

```

JOB
-----
MANAGER
PRESIDENT
SALESMAN
ANALYST
CLERK
5개의 행이 선택되었다.

```

Top-N Query

- Top-n query
 - 순위가 높은 n개의 튜플을 추출하는 질의.
- 문제점
 - Oracle의 경우, ROWNUM 변수와 ORDER BY 절을 직접 사용하여 top-N query를 작성하면 잘못된 결과를 얻게됨.
 - SELECT문에서는 정렬이 완료된 후 데이터의 일부가 출력되는 것이 아니라, 데이터의 일부가 먼저 추출된 후 정렬 작업이 실행됨.
 - 즉, ORDER BY 절은 결과 집합을 결정하는데 관여하지 않음.
- 해결책
 - 1. Oracle: ORDER BY 절을 inline view를 이용하여 작성한 후, ROWNUM 변수를 사용함.
 - 2. SQL Server의 TOP(n) 함수를 사용

TOP (Expression) [PERCENT] [WITH TIES]

- 예제(Oracle) : 사원에서 급여가 높은 3명만 내림차순으로 출력.
 - 급여 순서에 관계없이, 무작위로 3개의 튜플을 출력 (semantic error!)

```
SELECT  ENAME, SAL
FROM    EMP
WHERE   ROWNUM < 4
ORDER   BY SAL DESC;                                // Semantic error !!!
```

- 급여가 높은 3명만 출력 (inline view 이용)

```
SELECT  ENAME, SAL
FROM    (SELECT  ENAME, SAL
        FROM    EMP
        ORDER   BY SAL DESC)
WHERE   ROWNUM < 4;
```

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000

3개의 행이 선택되었다.

○ 예제(SQL Server) : TOP(n) 함수

– 사원 테이블에서 급여가 높은 2명을 내림차순으로 출력함.

```
SELECT TOP(2) ENAME, SAL
FROM EMP
ORDER BY SAL DESC;
```

ENAME	SAL
KING	5000
SCOTT	3000

2개의 행이 선택되었다.

– 사원 테이블에서 급여가 높은 2명을 내림차순으로 출력하며,
같은 급여를 받는 사원이 있으면 같이 출력함. (동점자 처리)

```
SELECT TOP(2) WITH TIES ENAME, SAL
FROM EMP
ORDER BY SAL DESC;
```

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000

3개의 행이 선택되었다.

👉 SELECT 문의 실행순서

○ 실행순서

```
5. SELECT [ALL|DISTINCT] {{컬럼명 [[AS] 컬럼_별명],}* | * }  
1. FROM 테이블_리스트  
2. [WHERE 조건식]  
3. [GROUP BY 컬럼명]  
4. [HAVING 그룹조건식]  
6. [ORDER BY {컬럼명|컬럼_별명|컬럼_순서 [ASC|DESC],}*];
```

- 1. 실행 대상 테이블을 참조한다. (FROM)
- 2. 조건에 맞는 튜플만 선택한다. (WHERE)
- 3. 정해진 컬럼의 값에 따라, 튜플들을 그룹화 한다. (GROUP BY)
- 4. 각 그룹별로, 그룹 조건에 맞는 튜플만 다시 선택한다. (HAVING)
- 5. 선택된 튜플에서, 기술된 컬럼/표현식만 출력/계산한다. (SELECT)
- 6. 튜플을 정렬한다. (ORDER BY)

- Query optimizer가 SQL 문장의 syntactic/semantic error를 점검하는 순서와 동일함.