

# Ch 4.

---

## Additional SELECT

4.1 | 함수<sup>Function</sup>

4.2 | Additional SELECT Option

# Objective

- ❖ SELECT 구문에서 함수를 사용할 수 있다
- ❖ SELECT 구문에서 ORDER BY option을 사용할 수 있다
- ❖ SELECT 구문에서 GROUP BY option을 사용할 수 있다
- ❖ SELECT 구문에서 HAVING option을 사용할 수 있다
- ❖ 하나 이상의 테이블로부터 데이터를 조회할 수 있다
- ❖ 두 개의 SELECT 구문 실행 결과를 하나로 결합할 수 있다
- ❖ SELECT 구문에서 서브쿼리<sup>Subquery</sup> option을 사용할 수 있다

# Ch 4.

## Additional SELECT

### 4.1 | 함수 Function

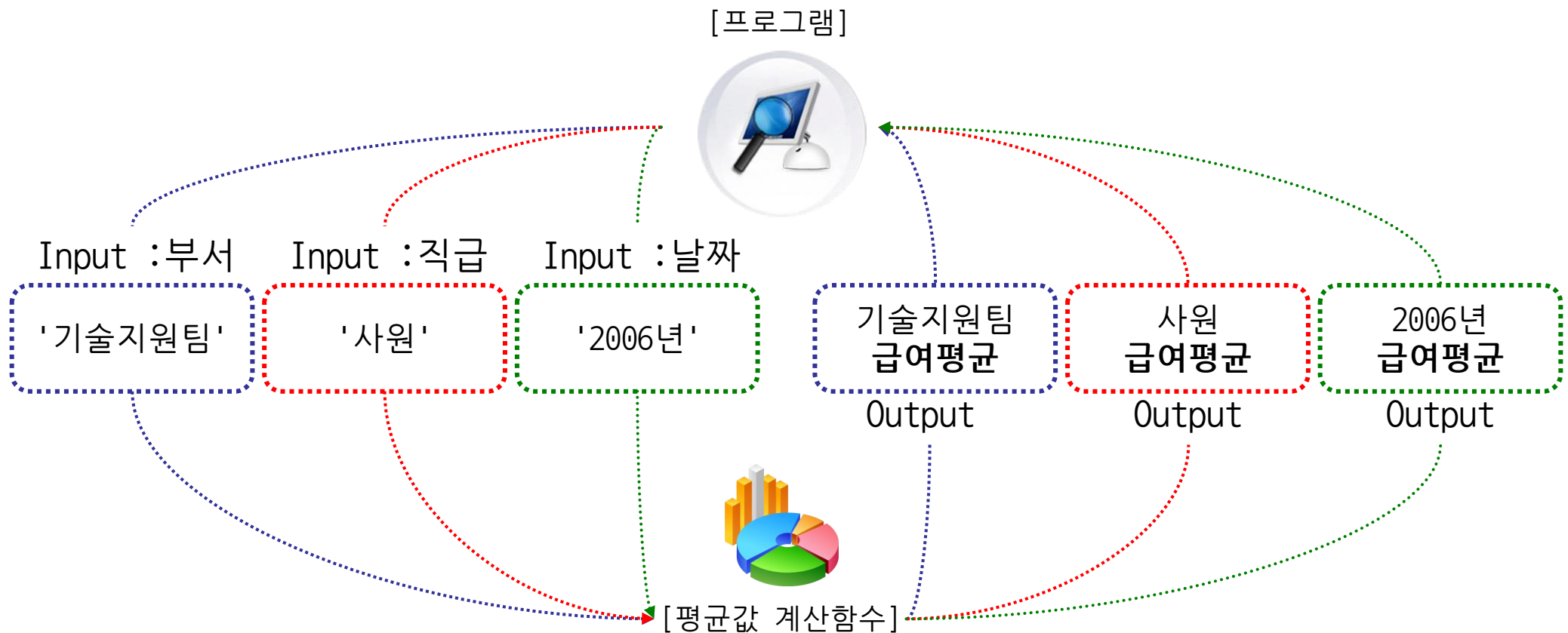
단일 행 함수 Single-Row Function

그룹 함수 Group Function

### 4.2 | Additional SELECT Option

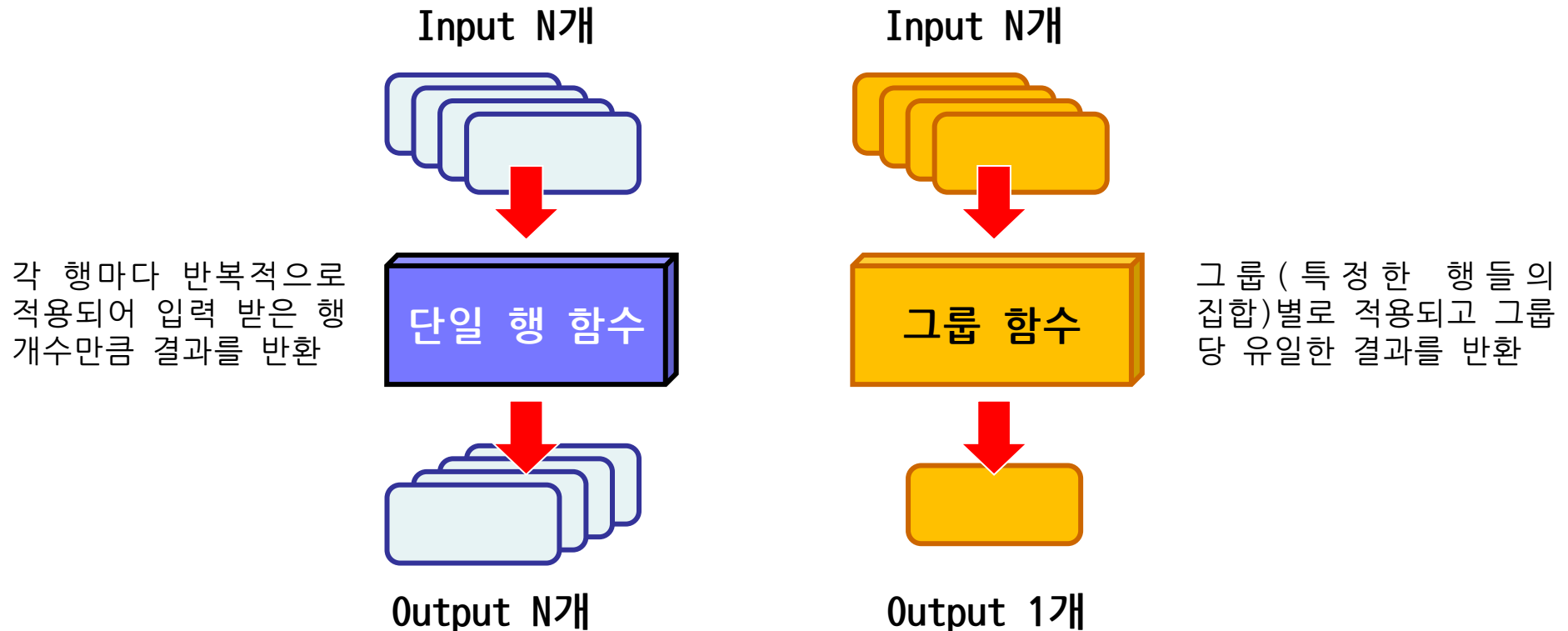
## 4.1.1 함수 개념

- 하나의 큰 프로그램에서 반복적으로 사용되는 부분들을 분리한 작은 서브 프로그램
- 호출<sup>call</sup>하고, 실행 결과를 리턴<sup>return</sup> 하는 방식으로 사용



## 4.1.2 함수 유형

반환 결과에 따라 단일 행 함수<sup>Single-Row Function</sup> 와 그룹 함수<sup>Group Function</sup> 로 구분



## 4.1.2 함수 유형 예

직원 테이블(일부)

사번	성별	근무지	나이
001	M	마포구 상암동 상암IT센터	29
002	M	중구 회현동 프라임 타워	30
003	F	영등포구 문래동 강서빌딩	25
004	M	영등포구 여의도동 LG 트윈타워	27
005	F	강남구 역삼동 GS타워	24

사번	근무지
001	마포구
002	중구
003	영등포구
004	영등포구
005	강남구

단일 행 함수  
사용 결과

성별	평균나이
M	28.6
F	24.5

그룹 함수  
사용 결과

4.1.3 주요 단일 행 함수

구분	입력 값 타입	리턴 값 타입	종류
문자(열) 함수	CHARACTER	CHARACTER	LPAD/RPAD, LTRIM/RTRIM/TRIM, SUBSTR
		NUMBER	INSTR, LENGTH/LENGTHB
숫자 함수	NUMBER	NUMBER	ROUND, TRUNC
날짜 함수	DATE	DATE	ADD_MONTHS, SYSDATE
		NUMBER	MONTHS_BETWEEN
타입변환 함수	ANY	ANY	TO_CHAR, TO_DATE, TO_NUMBER
기타 함수	ANY	ANY	NVL, DECODE

주오라클 전용 함수 중 일부만 표시

## 4.1.3 주요 단일 행 함수 - 문자열 함수 LENGTH

주어진 컬럼 값/문자열 길이(문자 개수)를 반환하는 함수

[구문]

[반환 타입]

[파라미터]

**LENGTH(*string* )**

NUMBER

CHARACTER 타입의 컬럼 또는 임의의 문자열

[구문 특징]

- CHAR 데이터 타입 컬럼 값을 입력 받은 경우  
실제 데이터 길이(문자 개수)에 상관없이 컬럼 전체 길이(문자 개수)를 반환
- VARCHAR2 데이터 타입 컬럼 값을 입력 받은 경우  
실제 데이터 길이(문자 개수) 반환



## 4.1.3 주요 단일 행 함수 - 문자열 함수 LENGTH 사용 예

[COLUMN\_LENGTH 테이블]

Columns of COLUMN_LENGTH	
Name	Type
CHARTYPE	CHAR(20)
VARCHARTYPE	VARCHAR2(20)

[샘플 데이터]

CHARTYPE	VARCHARTYPE
LG CNS	LG CNS
엘지씨엔에스	엘지씨엔에스

[적용 예]

```
SELECT LENGTH(CHARTYPE),
       LENGTH(VARCHARTYPE)
FROM   COLUMN_LENGTH;
```

LENGTH(CHARTYPE)	LENGTH(VARCHARTYPE)
20	6
14	6

CHAR 타입 컬럼에 한글이 포함되었을 때 LENGTH 함수 적용 결과

- 한글 1문자=2바이트, 6문자이므로  $6 \times 2 = 12$ 바이트
- 컬럼 길이(20바이트) - 데이터 길이(12바이트) = 8바이트
- 데이터 6문자 + 여유 공간 8문자(영문자 기준으로) = 14

참고 오라클 DBMS를 설치할 때 어떤 설정을 하느냐에 따라 한글 1문자의 바이트 수가 달라짐  
 (본 과정에서는 Oracle 10g K016MSWIN949 character set 사용 → 한글 1문자는 2바이트로 처리됨)

## 4.1.3 주요 단일 행 함수 - 문자열 함수 LENGTHB

주어진 컬럼 값/문자열 길이(Byte)를 반환하는 함수

[구문]

[반환 타입]

[파라미터]

**LENGTHB(*string* )**

NUMBER

CHARACTER 타입의 컬럼 또는 임의의 문자열

[구문 특징]

- Single-byte Character Set 경우  
LENGTH 함수 반환 값과 동일(문자 개수 = 바이트 수)
- Multi-byte Character Set 경우  
Character Set에 따라 동일한 데이터에 대한 바이트 처리 결과가 다름

## 4.1.3 주요 단일 행 함수 - 문자열 함수 LENGTHB 사용 예

[ COLUMN\_LENGTH 테이블 ]

Columns of COLUMN_LENGTH	
Name	Type
CHARTYPE	CHAR(20)
VARCHARTYPE	VARCHAR2(20)

[ 샘플 데이터 ]

CHARTYPE	VARCHARTYPE
LG CNS	LG CNS
엘지씨엔에스	엘지씨엔에스

[ 적용 예 ]

```
SELECT LENGTHB(CHARTYPE),
       LENGTHB(VARCHARTYPE)
FROM   COLUMN_LENGTH;
```

LENGTHB(CHARTYPE)	LENGTHB(VARCHARTYPE)
20	6
20	12

## 4.1.3 주요 단일 행 함수 - 문자열 함수 INSTR

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

[구문]

[반환 타입]

```
INSTR(string, substring, [ position, [ occurrence ] ] )
```

NUMBER

[파라미터]

파라미터	설명
string	문자 타입 컬럼 또는 문자열
substring	찾으려는 문자(열)
position	<ul style="list-style-type: none"> <li>어디부터 찾을지를 결정하는 시작 위치(기본값 1)</li> <li>POSITION &gt; 0 : STRING의 시작부터 끝 방향으로 찾는 의미</li> <li>POSITION &lt; 0 : STRING의 끝부터 시작 방향으로 찾는 의미</li> </ul>
occurrence	<ul style="list-style-type: none"> <li>SUBSTRING이 반복될 때의 지정하는 빈도(기본값 1)</li> <li>음수 값은 사용할 수 없음</li> </ul>

## 4.1.3 주요 단일 행 함수 - 문자열 함수 INSTR 사용 예

EMAIL 컬럼 값의 '@vcc.com' 문자열 중 "." 바로 앞의 문자 'c' 위치를 구하시오

[구문 1]

```
SELECT EMAIL,
       INSTR( EMAIL, 'c', -1, 2 ) 위치
FROM   EMPLOYEE ;
```

'POSITION'은 음수(-1)이므로 끝부터 시작 방향으로 찾음

'OCCURRENCE'는 2 이므로 두 번째 나타나는 문자 위치를 의미

[구문 2]

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

'POSITION'은 INSTR(EMAIL, '.')-1 결과(≡ 마침표 한 자리 앞 의미)

'OCCURRENCE'는 생략되었으므로 기본 값 1 적용(첫 번째 나타나는 문자 의미)

[결과(일부)]

EMAIL	위치
sg_ahn@vcc.com	10
jh_park@vcc.com	11
ms_choi@vcc.com	11
sy_kang@vcc.com	11
sg_han@vcc.com	10
jh_jo@vcc.com	9
jih_jeon@vcc.com	12
hs_kim@vcc.com	10
sw_cha@vcc.com	10
sm_kim@vcc.com	10
hg_song@vcc.com	11
ws_jeong@vcc.com	12
jh_um@vcc.com	9
hk_shin@vcc.com	11
sw_jo@vcc.com	9
hi_park@vcc.com	11
sw_kwon@vcc.com	11

## 4.1.3 주요 단일 행 함수 - 문자열 함수 LPAD/RPAD

주어진 컬럼/문자열에 임의의 문자(열)을 왼쪽/오른쪽에 덧붙여 길이 N의 문자열을 반환하는 함수

[구문]

[반환 타입]

LPAD( *string*, *N*, [*str*] )  
 RPAD( *string*, *N*, [*str*] )

CHARACTER

[파라미터]

파라미터	설명
string	문자 타입 컬럼 또는 문자열
N	<ul style="list-style-type: none"> <li>반환할 문자(열)의 길이(바이트)</li> <li>원래 STRING 길이보다 작다면 N만큼만 잘라서 표시</li> </ul>
str	<ul style="list-style-type: none"> <li>덧붙이려는 문자(열)</li> <li>생략하면 공백 한 문자</li> </ul>

## 4.1.3 주요 단일 행 함수 - 문자열 함수 LPAD 사용 예

EMAIL 컬럼 왼쪽에 '.' 을 덧붙여 길이를 20으로 맞추시오

```
SELECT EMAIL AS 원본데이터,
       LENGTH(EMAIL) AS 원본길이,
       LPAD(EMAIL, 20, '.') AS 적용결과,
       LENGTH(LPAD(EMAIL, 20, '.')) AS 결과길이
FROM   EMPLOYEE;
```

[결과(일부)]



원본데이터	원본길이	적용결과	결과길이
sg_ahn@vcc.com	14	.....sg_ahn@vcc.com	20
jh_park@vcc.com	15	.....jh_park@vcc.com	20
ms_choi@vcc.com	15	.....ms_choi@vcc.com	20
sy_kang@vcc.com	15	.....sy_kang@vcc.com	20
sg_han@vcc.com	14	.....sg_han@vcc.com	20
jh_jo@vcc.com	13	.....jh_jo@vcc.com	20
jih_jeon@vcc.com	16	....jih_jeon@vcc.com	20


오른쪽 정렬 효과를 나타냄

## 4.1.3 주요 단일 행 함수 - 문자열 함수 RPAD 사용 예

EMAIL 컬럼 오른쪽에 '.' 을 덧붙여 길이를 20으로 맞추시오

```
SELECT EMAIL AS 원본데이터,
       LENGTH(EMAIL) AS 원본길이,
       RPAD(EMAIL, 20, '.') AS 적용결과,
       LENGTH(RPAD(EMAIL, 20, '.')) AS 결과길이
FROM   EMPLOYEE;
```

[결과(일부)]



원본데이터	원본길이	적용결과	결과길이
sg_ahn@vcc.com	14	sg_ahn@vcc.com.....	20
jh_park@vcc.com	15	jh_park@vcc.com.....	20
ms_choi@vcc.com	15	ms_choi@vcc.com.....	20
sy_kang@vcc.com	15	sy_kang@vcc.com.....	20
sg_han@vcc.com	14	sg_han@vcc.com.....	20
jh_jo@vcc.com	13	jh_jo@vcc.com.....	20
jih_jeon@vcc.com	16	jih_jeon@vcc.com....	20

왼쪽 정렬 효과를 나타냄



## 4.1.3 주요 단일 행 함수 - 문자열 함수 LTRIM/RTRIM

- 주어진 컬럼/문자열의 왼쪽/오른쪽에서 지정한 STR에 포함된 모든 문자를 제거한 나머지를 반환하는 함수
- 패턴을 제거하는 의미가 아님

[구문]

[반환 타입]

LTRIM( *string*, *str* )  
 RTRIM( *string*, *str* )

CHARACTER

[파라미터]

파라미터	설명
string	문자 타입 컬럼 또는 문자열
str	<ul style="list-style-type: none"> <li>제거하려는 문자(열)</li> <li>생략하면 왼쪽/오른쪽 공백을 모두 제거</li> </ul>

## 4.1.3 주요 단일 행 함수 - 문자열 함수 LTRIM 사용 예

<sup>주</sup> SELECT LTRIM('...tech') FROM DUAL;	tech
SELECT LTRIM('...tech', '·') FROM DUAL;	tech
SELECT LTRIM('000123', '0') FROM DUAL;	123
SELECT LTRIM('123123Tech', '123') FROM DUAL;	Tech
SELECT LTRIM('123123Tech123', '123') FROM DUAL;	Tech123
SELECT LTRIM('xyxzyyyTech', 'xyz') FROM DUAL;	Tech
SELECT LTRIM('6372Tech', '0123456789') FROM DUAL;	Tech

<sup>주</sup> SELECT ... FROM DUAL 구문은 테스트 용도로 사용하는 구문임

## 4.1.3 주요 단일 행 함수 - 문자열 함수 RTRIM 사용 예

SELECT RTRIM('tech...') FROM DUAL;	tech
SELECT RTRIM('tech...', '.') FROM DUAL;	tech
SELECT RTRIM('123000', '0') FROM DUAL;	123
SELECT RTRIM('Tech123123', '123') FROM DUAL;	Tech
SELECT RTRIM('123Tech123', '123') FROM DUAL;	123Tech
SELECT RTRIM('Techxyxzyyy', 'xyz') FROM DUAL;	Tech
SELECT RTRIM('Tech6372', '0123456789') FROM DUAL;	Tech

## 4.1.3 주요 단일 행 함수 - 문자열 함수 TRIM

주어진 컬럼/문자열의 앞/뒤/양쪽에 있는 지정한 문자를 제거한 나머지를 반환하는 함수

[구문]

[반환 타입]

```
TRIM( trim_source )
TRIM( trim_char FROM trim_source )
TRIM( LEADING|TRAILING|BOTH [trim_char] FROM trim_source )
```

CHARACTER

[파라미터]

파라미터	설명
trim_source	문자 타입 컬럼 또는 문자열
trim_char	<ul style="list-style-type: none"> <li>제거하려는 <u>하나의 문자</u></li> <li>생략하면 기본값으로 공백 한 문자</li> </ul>
LEADING TRAILING BOTH	<ul style="list-style-type: none"> <li>trim_char 위치 지정</li> <li>앞/뒤/양쪽 지정 가능(기본값은 양쪽)</li> </ul>

## 4.1.3 주요 단일 행 함수 - 문자열 함수 TRIM 사용 예

SELECT TRIM('··tech··') FROM DUAL;	tech
SELECT TRIM('a' FROM 'aatechaaa') FROM DUAL;	tech
SELECT TRIM(LEADING '0' FROM '000123') FROM DUAL;	123
SELECT TRIM(TRAILING '1' FROM 'Tech1') FROM DUAL;	Tech
SELECT TRIM(BOTH '1' FROM '123Tech111') FROM DUAL;	23Tech
SELECT TRIM(LEADING FROM '··Tech··') FROM DUAL;	Tech··

## 4.1.3 주요 단일 행 함수 - 문자열 함수 SUBSTR

주어진 컬럼/문자열에서 지정한 위치부터 지정한 개수 만큼의 문자열을 잘라내어 반환하는 함수

[구문]

[반환 타입]

**SUBSTR( *string*, *position*, [ *length* ] )**

CHARACTER

[파라미터]

파라미터	설명
string	문자 타입 컬럼 또는 문자열
position	<ul style="list-style-type: none"> <li>• 잘라내는 시작 위치</li> <li>• position = 0 or 1 : 시작 위치(1번째)</li> <li>• position &gt; 0 : 끝 방향으로 지정한 수만큼 위치</li> <li>• position &lt; 0 : 시작 방향으로 지정한 수만큼 위치</li> </ul>
length	<ul style="list-style-type: none"> <li>• 반환할 문자 개수</li> <li>• 잘라내는 방향은 항상 끝 방향</li> <li>• 생략하면 position부터 문자열 끝까지를 의미</li> <li>• length &lt; 0 : NULL 반환</li> </ul>

## 4.1.3 주요 단일 행 함수 - 문자열 함수 SUBSTR 사용 예

SELECT SUBSTR('This·is·a·test', 6, 2) FROM DUAL;	is
SELECT SUBSTR('This·is·a·test', 6) FROM DUAL;	is·a·test
SELECT SUBSTR('이것은·연습입니다', 3, 4) FROM DUAL;	은·연습
SELECT SUBSTR('TechOnTheNet', 1, 4) FROM DUAL;	Tech
SELECT SUBSTR('TechOnTheNet', -3, 3) FROM DUAL;	Net
SELECT SUBSTR('TechOnTheNet', -6, 3) FROM DUAL;	The
SELECT SUBSTR('TechOnTheNet', -8, 2) FROM DUAL;	On

## 4.1.3 주요 단일 행 함수 - 숫자 함수 ROUND

지정한 자릿수에서 반올림 하는 함수

[구문]

**ROUND( *number*, [*decimal\_places*] )**

[반환 타입]

NUMBER

[파라미터]

파라미터	설명
number	숫자 타입 컬럼 또는 임의 숫자
decimal_places	<ul style="list-style-type: none"><li>반올림되어 표시되는 소수점 자리</li><li>반드시 정수 값 사용</li><li>생략되면 0 의미</li><li>decimal_places &gt; 0 : 소수점 이하 자리 의미</li><li>decimal_places &lt; 0 : 소수점 이상 자리 의미</li></ul>



## 4.1.3 주요 단일 행 함수 - 숫자 함수 ROUND 사용 예

SELECT ROUND(125.315) FROM DUAL;	125
SELECT ROUND(125.315, 0) FROM DUAL;	125
SELECT ROUND(125.315, 1) FROM DUAL;	125.3
SELECT ROUND(125.315, -1) FROM DUAL;	130
SELECT ROUND(125.315, 3) FROM DUAL;	125.315
SELECT ROUND(-125.315, 2) FROM DUAL;	-125.32

## 4.1.3 주요 단일 행 함수 - 숫자 함수 TRUNC

지정한 자릿수에서 버림 하는 함수

[구문]

[반환 타입]

```
TRUNC( number, [decimal_places] )
```

NUMBER

[파라미터]

파라미터	설명
number	숫자 타입 컬럼 또는 임의 숫자
decimal_places	<ul style="list-style-type: none"> <li>• 버림 결과가 표시되는 소수점 자리</li> <li>• 반드시 정수 값 사용</li> <li>• 생략되면 0 의미</li> <li>• decimal_places &gt; 0 : 소수점 이하 자리 의미</li> <li>• decimal_places &lt; 0 : 소수점 이상 자리 의미</li> </ul>

## 4.1.3 주요 단일 행 함수 - 숫자 함수 TRUNC 사용 예

SELECT TRUNC(125.315) FROM DUAL;	125
SELECT TRUNC(125.315, 0) FROM DUAL;	125
SELECT TRUNC(125.315, 1) FROM DUAL;	125.3
SELECT TRUNC(125.315, -1) FROM DUAL;	120
SELECT TRUNC(125.315, 3) FROM DUAL;	125.315
SELECT TRUNC(-125.315, -3) FROM DUAL;	0

## 4.1.3 주요 단일 행 함수 - 날짜 함수 SYSDATE

- 지정된 형식으로 현재 날짜와 시간을 표시하는 함수
- 한글 버전으로 설치된 경우 기본 설정 형식은 '년도(2자리)/월(2자리)/일(2자리)'

[구문]

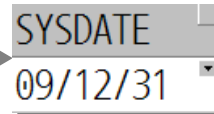
SYSDATE

[반환 타입]

DATE

[사용 예]

```
SELECT SYSDATE  
FROM DUAL;
```



SYSDATE
09/12/31

## 4.1.3 주요 단일 행 함수 - 날짜 함수 ADD\_MONTHS

지정한 만큼의 달 수를 더한 날짜를 반환하는 함수

[구문]

[반환 타입]

**ADD\_MONTHS( *date*, *N* )**

DATE

[파라미터]

파라미터	설명
date	기준이 되는 날짜
N	date에 더하려는 월 수

## 4.1.3 주요 단일 행 함수 - 날짜 함수 ADD\_MONTHS 사용 예

직원 별로 입사일 기준으로 근무한 지 20년이 되는 일자를 조회하시오.

```
SELECT EMP_NAME,
       HIRE_DATE,
       ADD_MONTHS( HIRE_DATE, 240 )
FROM   EMPLOYEE;
```

EMP_NAME	HIRE_DATE	ADD_MONTHS(HIRE_DATE,240)
한선기	90/04/01	10/04/01
강중훈	04/04/30	24/04/30
최만식	95/12/30	15/12/30
정도연	97/06/03	17/06/03
안석규	98/07/01	18/07/01
조재형	98/11/23	18/11/23
정지현	04/07/15	24/07/15
김예수	01/03/20	21/03/20
나승원	01/03/20	21/03/20
김순이	99/10/20	19/10/20
성해교	03/08/16	23/08/16
전우성	02/07/14	22/07/14
엄정하	04/07/21	24/07/21
심하균	04/09/30	24/09/30
고승우	03/04/11	23/04/11
박하일	04/11/10	24/11/10
권상후	01/05/20	21/05/20
임영애	00/01/31	20/01/31
염정하	03/09/17	23/09/17
김솔오	96/10/01	16/10/01
이중기	04/10/01	24/10/01
감우섭	05/07/31	25/07/31

## 4.1.3 주요 단일 행 함수 - 날짜 함수 MONTHS\_BETWEEN

지정한 두 날짜 사이의 월 수를 반환하는 함수

[구문]

[반환 타입]

MONTHS\_BETWEEN( *date1*, *date2* )

NUMBER

[파라미터]

파라미터	설명
<i>date1</i> <i>date2</i>	<ul style="list-style-type: none"><li>· <i>date1</i>, <i>date2</i> 는 날짜</li><li>· <i>date1</i> &gt; <i>date 2</i> : 양수 반환</li><li>· <i>date1</i> &lt; <i>date 2</i> : 음수 반환</li></ul>

날짜가 크다는 의미는 더 나중 날짜라는 의미임

## 4.1.3 주요 단일 행 함수 - 날짜 함수 MONTHS\_BETWEEN 사용 예

SELECT MONTHS_BETWEEN( '09/01/01', '09/03/14' ) FROM DUAL;	-2.41935483870968
SELECT MONTHS_BETWEEN( '09/07/01', '09/03/14' ) FROM DUAL;	3.58064516129032
SELECT MONTHS_BETWEEN( '09/03/01', '09/03/01' ) FROM DUAL;	0
SELECT MONTHS_BETWEEN( '09/08/02', '09/06/02' ) FROM DUAL;	2

결과 중 정수가 아닌 부분은 달의 일부를 의미함

'2010년 1월 1일' 기준으로 입사한지 10년이 넘는 직원들의 근무년수 조회

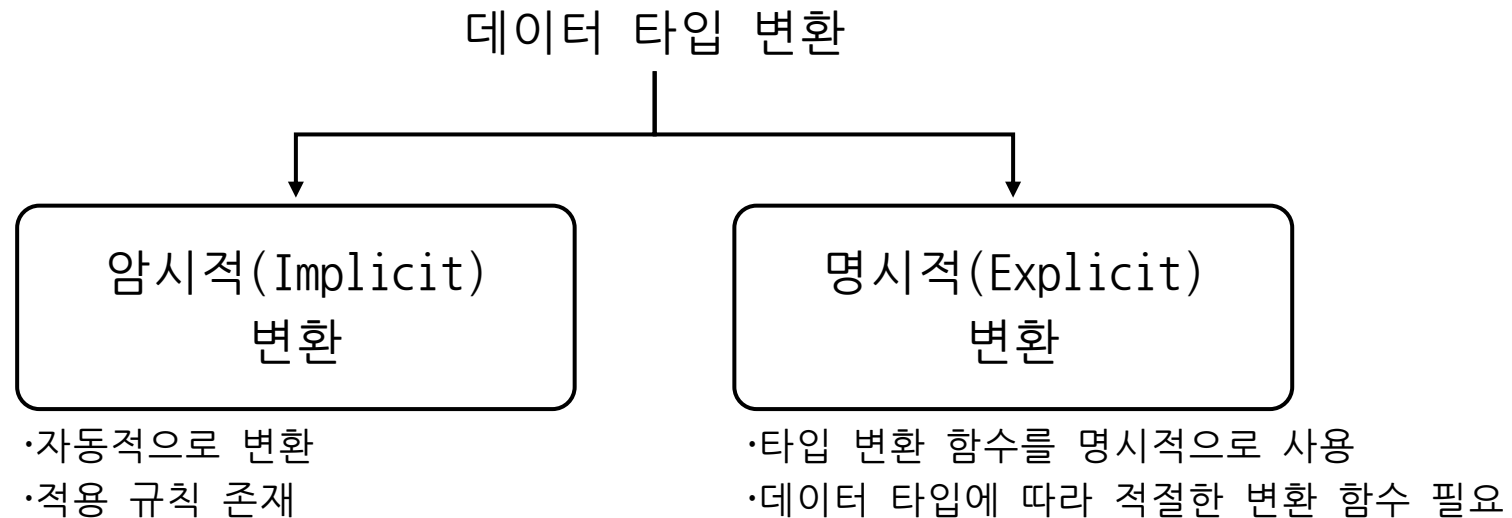
```
SELECT EMP_NAME,
       HIRE_DATE,
       MONTHS_BETWEEN('10/01/01', HIRE_DATE)/12 AS 근무년수
FROM   EMPLOYEE
WHERE  MONTHS_BETWEEN('10/01/01', HIRE_DATE) > 120;
```

EMP_NAME	HIRE_DATE	근무년수
최만식	95/12/30	14.005376344086
정도연	97/06/03	12.5779569892473
안석규	98/07/01	11.5
조재형	98/11/23	11.1075268817204
김순이	99/10/20	10.1989247311828
김술오	96/10/01	13.25



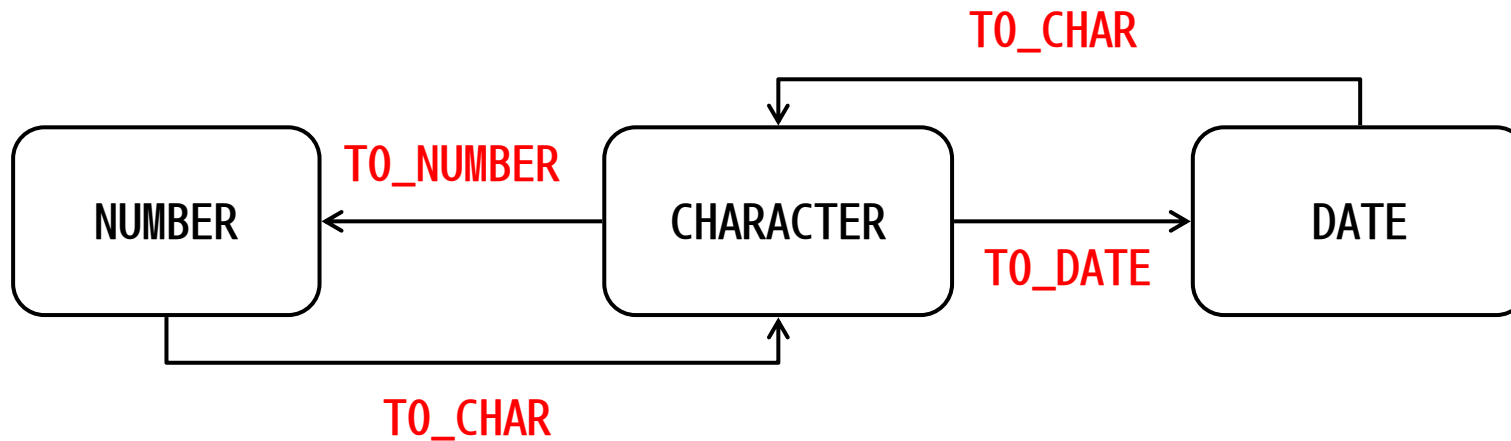
## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환

오라클 DBMS는 데이터 타입을 변환하는 두 가지 방법을 제공



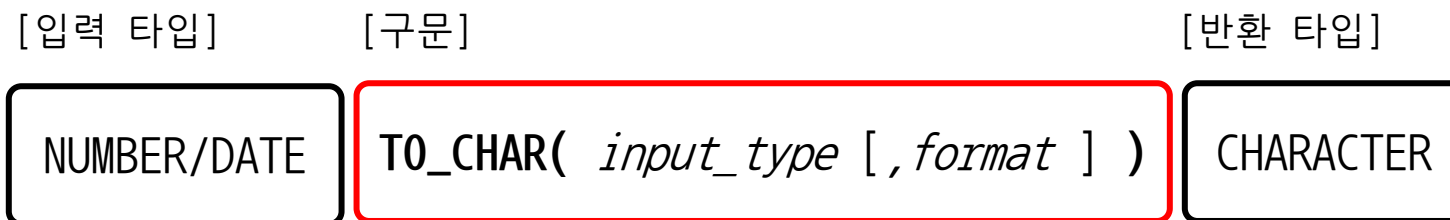
## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환

- 데이터 타입에 따라 사용할 수 있는 변환 함수가 다름
- 상호 타입 변환이 되지 않는 데이터 타입 존재



## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR

NUMBER/DATE 타입을 CHARACTER 타입으로 변환하는 함수



[함수 설명]

- NUMBER 타입을 CHARACTER 타입으로 변환이 필요한 경우
  - 표현 형식을 변경 : 1000 → 1,000
  - 숫자를 문자로 사용 : 100 → '100'
- DATE 타입을 CHARACTER 타입으로 변환이 필요한 경우
  - "년,월,일" 표현 형식을 변경 : 09/12/30 → 2009-12-30
  - 시간 정보 표시
  - CHARACTER 타입과 비교 : HIRE\_DATE = '03/06/14'
- 'format' 파라미터는 변경하려는 표현 형식을 위해 제공되는 규칙

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR 사용 예


사번이 100인 직원 이름과 급여 조회

- CHAR 타입과 NUMBER 타입 비교는 불가능
- 내부적으로 CHAR 타입인 EMP\_ID 값이 NUMBER 타입으로 변환되어 처리됨 → 암시적 변환

```
SELECT EMP_NAME,  
       SALARY  
FROM   EMPLOYEE  
WHERE  EMP_ID = 100;
```

TO\_CHAR 함수를 사용하여  
NUMBER 타입을 CHAR 타입으로  
변환 후 비교 → 명시적 변환

```
SELECT EMP_NAME,  
       SALARY  
FROM   EMPLOYEE  
WHERE  EMP_ID = TO_CHAR(100);
```



EMP_NAME	SALARY
한선기	9000000

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR 사용 예

[제공되는 숫자 표현형식]

형식	설명
9	자리 수 지정
0	남는 자리를 0으로 표시
\$ 또는 L	통화기호 표시
. 또는 ,	지정한 위치에 . 또는 , 표시
EEEE	과학 지수 표기법

SELECT TO_CHAR(1234, '99999') FROM DUAL;	1234
SELECT TO_CHAR(1234, '09999') FROM DUAL;	01234
SELECT TO_CHAR(1234, 'L99999') FROM DUAL;	₩1234
SELECT TO_CHAR(1234, '99,999') FROM DUAL;	1,234
SELECT TO_CHAR(1234, '09,999') FROM DUAL;	01,234
SELECT TO_CHAR(1000, '9.9EEEE') FROM DUAL;	1.0E+03
SELECT TO_CHAR(1234, '999') FROM DUAL;	###

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR 사용 예

[제공되는 날짜 표현형식]

형식	설명
YYYY/YY/YEAR	년도(4자리숫자/뒤 2자리숫자/문자)
MONTH/MON/MM/RM	달(이름/약어/숫자/로마 기호)
DDD/DD/D	일(1년 기준/1달 기준/1주 기준)
Q	분기(1, 2, 3, 4)
DAY/DY	요일(이름/약어 이름)
HH(12)/HH24	12시간/24시간 표시
AM PM	오전, 오후
MI	분(0~59)
SS	초(0~59)

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR 사용 예

SELECT TO_CHAR( SYSDATE, 'PM HH24:MI:SS' ) FROM DUAL;	오후 20:57:11
SELECT TO_CHAR( SYSDATE, 'AM HH:MI:SS' ) FROM DUAL;	오후 08:57:11
SELECT TO_CHAR( SYSDATE, 'MON DY, YYYY' ) FROM DUAL;	1월 월, 2010
SELECT TO_CHAR( SYSDATE, 'YYYY-fmMM-DD DAY' ) FROM DUAL;	2010-1-4 월요일
SELECT TO_CHAR( SYSDATE, 'YYYY-MM-fmDD DAY' ) FROM DUAL;	2010-01-4 월요일
SELECT TO_CHAR( SYSDATE, 'Year, Q' ) FROM DUAL;	Twenty Ten, 1

'fm' 모델을 사용하면 '01' 형식이 '1' 형식으로 표현됨

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR 사용 예

```
SELECT EMP_NAME AS 이름,
       TO_CHAR(HIRE_DATE, 'YYYY-MM-DD') AS 입사일
FROM   EMPLOYEE
WHERE  JOB_ID = 'J7';
```

이름	입사일
정지현	2004-07-15
성해교	2003-08-16
고승우	2003-04-11
염정하	2003-09-17

```
SELECT EMP_NAME AS 이름,
       TO_CHAR(HIRE_DATE, 'YYYY"년" MM"월" DD"일"') AS 입사일
FROM   EMPLOYEE
WHERE  JOB_ID = 'J7';
```

이름	입사일
정지현	2004년 07월 15일
성해교	2003년 08월 16일
고승우	2003년 04월 11일
염정하	2003년 09월 17일

```
SELECT EMP_NAME AS 이름,
       SUBSTR(HIRE_DATE,1,2)||'년 '||
       SUBSTR(HIRE_DATE,4,2)||'월 '||
       SUBSTR(HIRE_DATE,7,2)||'일' AS 입사일
FROM   EMPLOYEE
WHERE  JOB_ID = 'J7';
```

이름	입사일
정지현	04년 07월 15일
성해교	03년 08월 16일
고승우	03년 04월 11일
염정하	03년 09월 17일



## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_CHAR 사용 예

DATE 타입과 CHARACTER 타입 비교

```

SELECT EMP_NAME AS 이름,
       HIRE_DATE AS 기본입사일,
       TO_CHAR(HIRE_DATE,
                'YYYY/MM/DD HH24:MI:SS') AS 상세입사일
FROM   EMPLOYEE
WHERE  JOB_ID IN ('J1','J2');

```

이름	기본입사일	상세입사일
한선기	90/04/01	2090/04/01 13:30:30
강중훈	04/04/30	2004/04/30 00:00:00
최만식	95/12/30	1995/12/30 00:00:00

DATE 타입에 시간 정보 없이 날짜만 입력하면 자동으로 시간은 해당 날짜의 '00시 00분 00초'로 저장

```

SELECT EMP_NAME
FROM   EMPLOYEE
WHERE  HIRE_DATE = '04/04/30';

```

EMP\_NAME  
강중훈

시간 정보가 없는 경우에는 기본 날짜 형식으로 비교 가능

```

SELECT EMP_NAME
FROM   EMPLOYEE
WHERE  HIRE_DATE = '90/04/01';

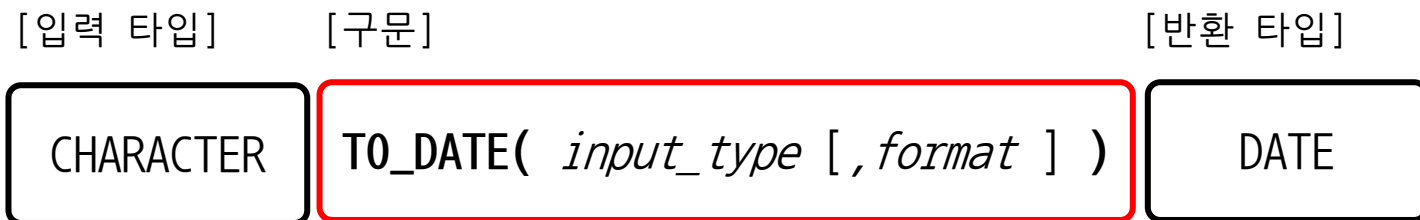
```

EMP\_NAME

시간 정보가 있는 경우에는 기본 날짜 형식으로는 비교 불가능

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_DATE

CHARACTER 타입을 DATE 타입으로 변환하는 함수



[함수 설명]

- 기본 날짜 형식이 아닌 문자열을 DATE 타입으로 인식시켜야 하는 경우
  - 2010-01-04 : 기본 날짜 형식이 아니므로 문자열로 인식
- DATE 타입과 비교하는 경우
  - HIRE\_DATE = '90/04/01 13:30:30'
- 제공되는 날짜 표현 형식을 이용


## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_DATE 사용 예

SELECT TO_DATE( '20100101', 'YYYYMMDD') FROM DUAL;	10/01/01
SELECT TO_CHAR( '20100101', 'YYYY, MON') FROM DUAL;	N/A(오류)
SELECT TO_CHAR( TO_DATE( '20100101', 'YYYYMMDD'), 'YYYY, MON') FROM DUAL;	2010, 1월
SELECT TO_DATE( '041030 143000', 'YYMMDD HH24MISS' ) FROM DUAL;	04/10/30
SELECT TO_CHAR( TO_DATE( '041030 143000', 'YYMMDD HH24MISS' ), 'DD-MON-YY HH:MI:SS PM' ) FROM DUAL;	30-10월-04 02:30:00 오후
SELECT TO_DATE( '980630', 'YYMMDD' ) FROM DUAL;	98/06/30
SELECT TO_CHAR( TO_DATE( '980630', 'YYMMDD' ), 'YYYY.MM.DD') FROM DUAL;	2098.06.30

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_DATE 사용 예

```
SELECT EMP_NAME,  
       HIRE_DATE  
FROM   EMPLOYEE  
WHERE  HIRE_DATE =  
       TO_DATE('900401 133030','YYMMDD HH24MISS');
```

```
SELECT EMP_NAME,  
       HIRE_DATE  
FROM   EMPLOYEE  
WHERE  TO_CHAR(HIRE_DATE, 'YYMMDD') = '900401';
```



EMP_NAME	HIRE_DATE
한선기	90/04/01

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_DATE

RR 날짜 형식

- YY 날짜 형식과 유사
- '지정한 년도'와 '현재 년도'에 따라 반환하는 '세기' 값이 달라짐 → 여러 세기 지정 가능

		지정한 년도(두 자리)	
		0 - 49	50 - 99
현재 년도 (두 자리)	0 - 49	현재 세기 반환	이전 세기 반환
	50 - 99	다음 세기 반환	현재 세기 반환

```
SELECT EMP_NAME,
       HIRE_DATE,
       TO_CHAR(HIRE_DATE, 'YYYY/MM/DD')
FROM   EMPLOYEE
WHERE  EMP_NAME = '한선기';
```

EMP_NAME	HIRE_DATE	TO_CHAR(HIRE_DATE, 'YYYY/MM/DD')
한선기	90/04/01	2090/04/01

- 해당 데이터는 두 자리 년도를 YY 형식으로 사용  
TO\_DATE('90/04/01 13:30:30', 'YY/MM/DD HH24:MI:SS')
- YY 형식은 항상 현재 세기를 의미  
→ 원래 의미 '1990' 대신 현재 세기인 '2090'으로 인식


## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_DATE 사용 예

현재 년도	지정 날짜	RR 형식	YY 형식
1995	95/10/27	1995	1995
1995	17/10/27	2017	1917
2009	17/10/27	2017	2017
2009	95/10/27	1995	2095

```

SELECT '2009/10/14' AS 현재,
       '95/10/27' AS 입력,
       TO_CHAR(TO_DATE('95/10/27', 'YY/MM/DD'), 'YYYY/MM/DD') AS YY형식1,
       TO_CHAR(TO_DATE('95/10/27', 'YY/MM/DD'), 'RRRR/MM/DD') AS YY형식2,
       TO_CHAR(TO_DATE('95/10/27', 'RR/MM/DD'), 'YYYY/MM/DD') AS RR형식1,
       TO_CHAR(TO_DATE('95/10/27', 'RR/MM/DD'), 'RRRR/MM/DD') AS RR형식2
FROM   DUAL;

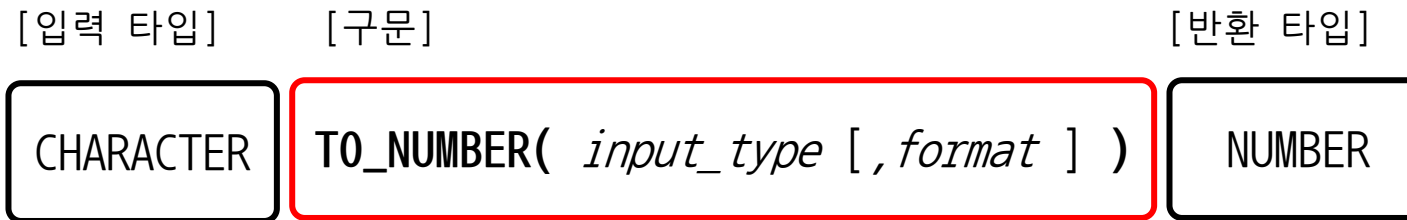
```



현재	입력	YY형식1	YY형식2	RR형식1	RR형식2
2009/10/14	95/10/27	2095/10/27	2095/10/27	1995/10/27	1995/10/27

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_NUMBER

CHARACTER 타입을 NUMBER 타입으로 변환하는 함수




[함수 설명]

- 숫자로 변환될 때 의미 있는 형태의 문자열인 경우  
'100' → 100 : 문자열 '100'을 숫자 100으로 변환

## 4.1.3 주요 단일 행 함수 - 데이터 타입 변환 함수 TO\_NUMBER 사용 예

```
SELECT EMP_NAME, EMP_NO,  
       SUBSTR(EMP_NO,1,6)AS 앞부분,  
       SUBSTR(EMP_NO,8) AS 뒷부분,  
       TO_NUMBER( SUBSTR(EMP_NO,1,6) ) + TO_NUMBER( SUBSTR(EMP_NO,8) ) AS 결과  
FROM   EMPLOYEE  
WHERE  EMP_ID = '101';
```



EMP_NAME	EMP_NO	앞부분	뒷부분	결과
강중훈	621136-1006405	621136	1006405	1627541

1006405 + 621136 = 1627541



## 4.1.3 주요 단일 행 함수 - 기타 함수 NVL

NULL을 지정한 값으로 변환하는 함수

[입력 타입]	[구문]	[반환 타입]
ANY	<b>NVL( <i>expr1</i>, <i>expr2</i> )</b>	ANY

[파라미터]

파라미터	설명
expr1	<ul style="list-style-type: none"><li>• NULL을 포함하는 컬럼</li><li>• NULL이 없는 경우는 expr1을 반환</li></ul>
expr2	<ul style="list-style-type: none"><li>• expr1이 NULL인 경우 변환할 값</li><li>• expr1의 데이터 타입과 동일한 타입으로 지정</li></ul>

## 4.1.3 주요 단일 행 함수 - 기타 함수 NVL 사용 예

```
SELECT EMP_NAME, SALARY, NVL(BONUS_PCT,0)
FROM   EMPLOYEE
WHERE  SALARY > 3500000;
```

EMP_NAME	SALARY	NVL(BONUS_PCT,0)
한선기	9000000	0.2
강중훈	5500000	0
최만식	3600000	0
조재형	3800000	0

```
SELECT EMP_NAME,
       (SALARY*12)+((SALARY*12)*BONUS_PCT)
FROM   EMPLOYEE
WHERE  SALARY > 3500000;
```

EMP_NAME	(SALARY*12)+((SALARY*12)*BONUS_PCT)
한선기	129600000
강중훈	
최만식	
조재형	

NULL에 대한 전체 계산 결과는 NULL임

```
SELECT EMP_NAME,
       (SALARY*12)+
       ((SALARY*12)*NVL(BONUS_PCT,0) )
FROM   EMPLOYEE
WHERE  SALARY > 3500000;
```

EMP_NAME	(SALARY*12)+((SALARY*12)*NVL(BONUS_PCT,0) )
한선기	129600000
강중훈	66000000
최만식	43200000
조재형	45600000

## 4.1.3 주요 단일 행 함수 - 기타 함수 DECODE

SELECT 구문으로 IF-ELSE 논리를 제한적으로 구현한 오라클 DBMS 전용 함수

[입력타입] [구문]

[반환타입]

ANY

`DECODE(expr, search1, result1 [ ,searchN, resultN, ...] [ , default ])`

ANY


[파라미터]

파라미터	설명
<i>expr</i>	대상 컬럼 또는 문자(열)
<i>search</i>	<i>expr</i> 과 비교하려는 값
<i>result</i>	IF <i>expr</i> = <i>search</i> 인 경우의 반환 값
<i>default</i>	<ul style="list-style-type: none"> <li>• <i>expr</i>과 <i>search</i>가 일치하지 않는 경우의 기본 반환 값</li> <li>• <i>default</i>를 지정하지 않고 <i>expr</i>과 <i>search</i>가 일치하지 않으면 NULL 반환</li> </ul>

## 4.1.3 주요 단일 행 함수 - 기타 함수 DECODE 사용 예

```
SELECT EMP_NAME,  
       DECODE(SUBSTR(EMP_NO,8,1),  
              '1', '남', '2', '여', '3', '남', '4', '여') AS 성별  
FROM   EMPLOYEE  
WHERE  DEPT_ID = '50';
```

```
SELECT EMP_NAME,  
       DECODE(SUBSTR(EMP_NO,8,1),  
              '1', '남', '3', '남', '여') AS 성별  
FROM   EMPLOYEE  
WHERE  DEPT_ID = '50';
```



EMP_NAME	성별
정지현	여
김예수	여
나승원	남
김순이	여
성해교	여
박하일	남

default 값을 지정하여 코드를 단순화 시킴

## 4.1.3 주요 단일 행 함수 - 기타 함수 DECODE 사용 예

```
SELECT EMP_ID, EMP_NAME,  
       DECODE(MGR_ID, NULL, '없음', MGR_ID) AS 관리자  
FROM   EMPLOYEE  
WHERE  JOB_ID = 'J4';
```

DECODE 함수는 2개의 NULL을 동일하게 취급함

```
SELECT EMP_ID, EMP_NAME,  
       NVL(MGR_ID, '없음') AS 관리자  
FROM   EMPLOYEE  
WHERE  JOB_ID = 'J4';
```

NULL을 처리하는 경우에는 NVL 함수와 동일한 결과를 나타냄

EMP_ID	EMP_NAME	관리자
103	정도연	104
207	김술오	없음
208	이중기	없음
210	감우섭	없음

## 4.1.3 주요 단일 행 함수 - 기타 함수 DECODE 사용 예

```

SELECT EMP_NAME,
       JOB_ID,
       SALARY,
       DECODE(JOB_ID,
               'J7', SALARY*1.1,
               'J6', SALARY*1.15,
               'J5', SALARY*1.2,
               SALARY*1.05) AS 인상급여
FROM   EMPLOYEE;

```

EMP_NAME	JOB_ID	SALARY	인상급여
한선기	J1	9000000	9450000
강중훈	J2	5500000	5775000
최만식	J2	3600000	3780000
정도연	J4	2600000	2730000
안석규	J3	3500000	3675000
조재형	J3	3800000	3990000
정지현	J7	1500000	1650000
김예수	J5	2100000	2520000
나승원	J5	2300000	2760000
김순이	J3	3400000	3570000
→ 성해교	J7	1900000	2090000
전우성	J6	2090000	2403500
엄정하	J6	2420000	2783000
심하균		2300000	2415000
고승우	J7	1500000	1650000
박하일	J5	2600000	3120000
권상후	J6	3410000	3921500
임영애	J6	2640000	3036000
엄정하	J7	1500000	1650000
김솔오	J4	2500000	2625000
이중기	J4	2500000	2625000
감우섭	J4	2500000	2625000

## 4.1.3 주요 단일 행 함수 - [참조] CASE

DECODE 함수와 유사한 ANSI 표준 구문

[입력타입] [구문]

[반환타입]

ANY

`CASE expr WHEN search1 THEN result1 [WHEN..THEN..][ELSE default] END`

`CASE WHEN condition1 THEN result1 [WHEN..THEN..][ELSE default] END`

ANY

[파라미터]

파라미터	설명
<code>expr</code>	대상 컬럼 또는 문자(열)
<code>search</code>	<code>expr</code> 과 비교하려는 값
<code>condition</code>	비교 조건
<code>result</code>	<ul style="list-style-type: none"> <li>IF <code>expr</code> = <code>search</code> 인 경우의 반환 값</li> <li>비교 조건을 만족시키는 경우의 반환 값</li> </ul>
<code>default</code>	<ul style="list-style-type: none"> <li><code>expr</code>과 <code>search</code>가 일치하지 않는 경우의 기본 반환 값</li> <li>비교 조건을 만족시키지 않는 경우의 기본 반환 값</li> <li><code>default</code>를 지정하지 않으면 일치하지 않거나 조건을 만족시키지 않는 경우 NULL 반환</li> </ul>

## 4.1.3 주요 단일 행 함수 - [참조] CASE 사용 예

```

SELECT EMP_NAME,
       JOB_ID,
       SALARY,
       CASE JOB_ID
         WHEN 'J7' THEN TO_CHAR(SALARY*1.1)
         WHEN 'J6' THEN TO_CHAR(SALARY*1.15)
         WHEN 'J5' THEN TO_CHAR(SALARY*1.2)
         ELSE TO_CHAR(SALARY*1.05) END AS 인상급여
FROM   EMPLOYEE;

```

EMP_NAME	JOB_ID	SALARY	인상급여
한선기	J1	9000000	9450000
강중훈	J2	5500000	5775000
최만식	J2	3600000	3780000
정도연	J4	2600000	2730000
안석규	J3	3500000	3675000
조재형	J3	3800000	3990000
정지현	J7	1500000	1650000
김예수	J5	2100000	2520000
나승원	J5	2300000	2760000
김순이	J3	3400000	3570000
성해교	J7	1900000	2090000
전우성	J6	2090000	2403500
엄정하	J6	2420000	2783000
심하균		2300000	2415000
고승우	J7	1500000	1650000
박하일	J5	2600000	3120000
권상후	J6	3410000	3921500
임영애	J6	2640000	3036000
엄정하	J7	1500000	1650000
김술오	J4	2500000	2625000
이중기	J4	2500000	2625000
감우섭	J4	2500000	2625000



## 4.1.3 주요 단일 행 함수 - [참조] CASE 사용 예

```

SELECT EMP_ID,
       EMP_NAME,
       SALARY,
       CASE WHEN SALARY <= 3000000 THEN '초급'
            WHEN SALARY <= 4000000 THEN '중급'
            ELSE '고급' END AS 구분
FROM   EMPLOYEE;

```

EMP_ID	EMP_NAME	SALARY	구분
100	한선기	9000000	고급
101	강중훈	5500000	고급
102	최만식	3600000	중급
103	정도연	2600000	초급
104	안석규	3500000	중급
107	조재형	3800000	중급
124	정지현	1500000	초급
141	김예수	2100000	초급
143	나승원	2300000	초급
144	김순이	3400000	중급
149	성해교	1900000	초급
174	전우성	2090000	초급
176	엄정하	2420000	초급
178	심하균	2300000	초급
200	고승우	1500000	초급
201	박하일	2600000	초급
202	권상후	3410000	중급
205	임영애	2640000	초급
206	엄정하	1500000	초급
207	김술오	2500000	초급
208	이중기	2500000	초급
210	감우섭	2500000	초급

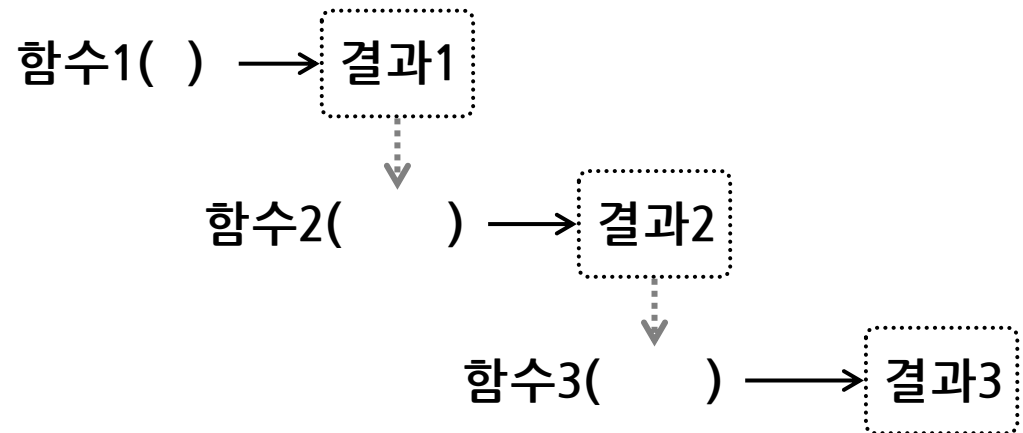
## 4.1.4 주요 단일 행 함수 - 함수 중첩

- 중첩 사용 가능
- 가장 안쪽의 함수부터 바깥 쪽 방향으로 차례대로 실행
- 먼저 실행된 함수의 반환 값이 바깥 쪽 함수의 입력 값이 됨 → 반환되는 함수 결과 데이터 타입에 주의

[구문]

```
함수3( 함수2( 함수1( ) ) ) )
```

[실행순서]



## 4.1.4 주요 단일 행 함수 - 함수 중첩

직원 별 Email ID 조회

```

SELECT EMP_NAME,
       EMAIL,
       SUBSTR(EMAIL, 1,
              INSTR(EMAIL, '@')-1) AS ID
FROM   EMPLOYEE;

```

EMP_NAME	EMAIL	ID
한선기	sg_ahn@vcc.com	sg_ahn
강중훈	jh_park@vcc.com	jh_park
최만식	ms_choi@vcc.com	ms_choi
정도연	sy_kang@vcc.com	sy_kang
안석규	sg_han@vcc.com	sg_han
조재형	jh_jo@vcc.com	jh_jo
정지현	jih_jeon@vcc.com	jih_jeon
김예수	hs_kim@vcc.com	hs_kim
나승원	sw_cha@vcc.com	sw_cha
김순이	sm_kim@vcc.com	sm_kim
성해교	hg_song@vcc.com	hg_song
전우성	ws_jeong@vcc.com	ws_jeong
엄정하	jh_um@vcc.com	jh_um
심하균	hk_shin@vcc.com	hk_shin
고승우	sw_jo@vcc.com	sw_jo
박하일	hi_park@vcc.com	hi_park
권상후	sw_kwon@vcc.com	sw_kwon
임영애	jangum_lee@vcc.com	jangum_lee
염정하	jh_yeum@vcc.com	jh_yeum
김술오	so_kim@vcc.com	so_kim
이중기	jk_lee@vcc.com	jk_lee
감우섭	manofking@vcc.com	manofking

## 4.1.5 주요 그룹 함수

그룹 함수는 NULL을 계산하지 않음


함수	설명
SUM	총합 계산
AVG	<ul style="list-style-type: none"><li>• 평균 계산</li><li>• NULL을 제외하므로 결과가 달라질 수 있음</li></ul>
MIN	<ul style="list-style-type: none"><li>• 최소 값 반환</li><li>• DATE 타입: 가장 오래 전 날짜를 의미</li><li>• CHARACTER 타입: 해당 character set의 내부 값이 가장 작은 문자를 의미</li></ul>
MAX	<ul style="list-style-type: none"><li>• 최대 값 반환</li><li>• DATE 타입: 가장 최근 날짜를 의미</li><li>• CHARACTER 타입: 해당 character set의 내부 값이 가장 큰 문자를 의미</li></ul>
COUNT	Result Set 전체 행 수 반환

## 4.1.5 주요 그룹 함수 - SUM

입력 값의 총합을 계산하여 반환하는 함수

[입력 타입]	[구문]	[반환 타입]
NUMBER	SUM( [DISTINCT] <i>expr</i> )	NUMBER

```
SELECT SUM(SALARY), SUM(DISTINCT SALARY)
FROM   EMPLOYEE
WHERE  DEPT_ID = '50'
OR     DEPT_ID IS NULL;
```



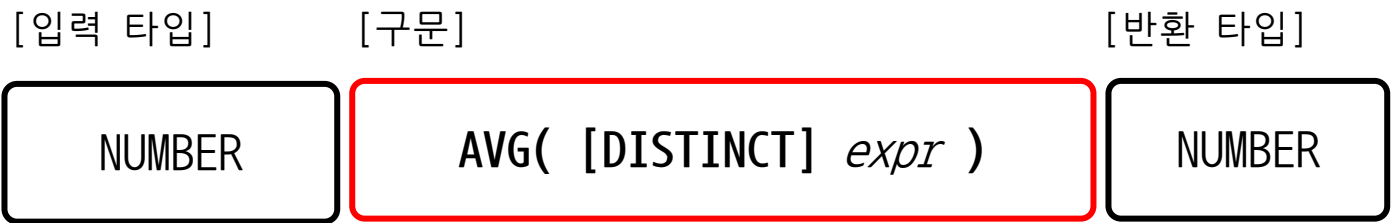
SUM(SALARY)	SUM(DISTINCT SALARY)
17600000	13800000

- SUM(SALARY): 전체 8건의 합
- SUM(DISTINCT SALARY): 중복 값 제외한 6건의 합

SALARY
1500000
2100000
2300000
3400000
1900000
2300000
2600000
1500000

4.1.5 주요 그룹 함수 - AVG

입력 값의 평균을 계산하여 반환하는 함수



```
SELECT AVG(BONUS_PCT) AS 기본평균,  
       AVG(DISTINCT BONUS_PCT) AS 중복제거평균,  
       AVG(NVL(BONUS_PCT,0)) AS NULL포함평균  
FROM   EMPLOYEE  
WHERE  DEPT_ID IN ('50', '90')  
OR     DEPT_ID IS NULL;
```

BONUS_PCT
0.2
0.1
0.1
0.3

기본평균	중복제거평균	NULL포함평균
0.175	0.2	0.0636363636363636

- 기본평균: BONUS\_PCT 값이 있는 4건의 평균
- 중복제거평균: 중복 값을 제외한 3건의 평균
- NULL포함평균: 전체 11건의 평균

## 4.1.5 주요 그룹 함수 - MIN/MAX

최소 값/최대 값을 반환하는 함수

[입력 타입]

[구문]

[반환 타입]

ANY

 $\text{MIN}( \text{expr} ), \text{MAX}( \text{expr} )$ 

ANY

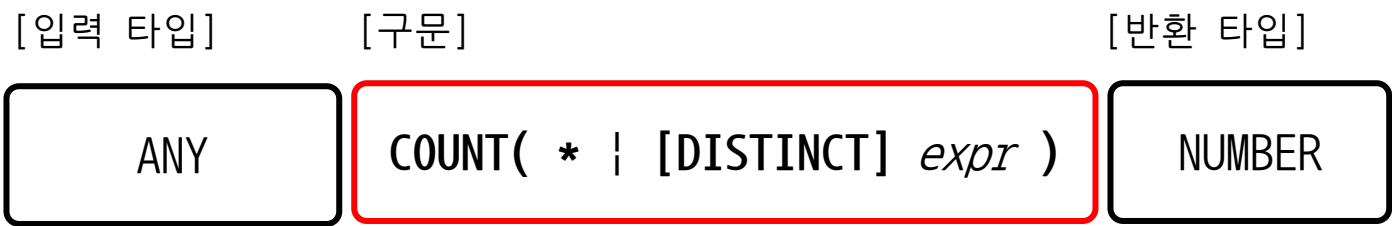
```
SELECT MAX(JOB_ID), MIN(JOB_ID),
       MAX(HIRE_DATE), MIN(HIRE_DATE),
       MAX(SALARY), MIN(SALARY)
FROM   EMPLOYEE
WHERE  DEPT_ID IN ( '50', '90' );
```

JOB_ID	HIRE_DATE	SALARY
J1	90/04/01	9000000
J2	04/04/30	5500000
J2	95/12/30	3600000
J7	04/07/15	1500000
J5	01/03/20	2100000
J5	01/03/20	2300000
J3	99/10/20	3400000
J7	03/08/16	1900000
J5	04/11/10	2600000

MAX(JOB_ID)	MIN(JOB_ID)	MAX(HIRE_DATE)	MIN(HIRE_DATE)	MAX(SALARY)	MIN(SALARY)
J7	J1	90/04/01	95/12/30	9000000	1500000

4.1.5 주요 그룹 함수 - COUNT

Result Set의 행 수를 반환하는 함수




[파라미터]

파라미터	설명
*	Result Set의 전체 행 수 반환
DISTINCT <i>expr</i>	<i>expr</i> 에 포함된 값 중 NULL과 중복 값을 제외한 행 수 반환
<i>expr</i>	<i>expr</i> 에 포함된 값 중 NULL을 제외한 행 수 반환



## 4.1.5 주요 그룹 함수 - COUNT 사용 예

```
SELECT COUNT(*),  
       COUNT(JOB_ID),  
       COUNT(DISTINCT JOB_ID)  
FROM   EMPLOYEE  
WHERE  DEPT_ID = '50'  
OR     DEPT_ID IS NULL;
```



COUNT(*)	COUNT(JOB_ID)	COUNT(DISTINCT JOB_ID)
8	7	3

JOB_ID
J7
J5
J5
J3
J7
J5
J7