

# 데이터베이스 프로그래밍

단일 행 함수

# 학습 목표

- 단일 행 함수의 특징을 이해한다.
- 문자함수 / 숫자함수 / 날짜함수 / 변환함수 / 기타함수들의 기능을 이해하고 사용할 수 있다.
- 필요한 함수들을 중첩 사용할 수 있도록 한다.

# 데이터베이스 종류별 SQL 주석

[illegible]

# SQL 함수

- 기본적인 QUERY 문을 더욱 강력하게 해 줌
- 질의가 쉬워지고 응용프로그램 코딩을 줄일 수 있음
- 종류
  - 내장 함수(Built-in Function) : 벤더에서 제공하는 함수
    - 단일행 함수(Single-Row Function) : 행 당 하나의 단일행 값이 입력
    - 다중행 함수(Multi-Row Function) : 여러 행의 값이 입력
      - 다중행 함수(Multi-Row Function) = 그룹함수 = 집계함수
  - 사용자가 정의할 수 있는 함수(User Defined Function)

# SQL 함수 사용

- 데이터의 계산 수행
- 개별 데이터의 항목 수정
- 표시할 날짜 및 숫자 형식 지정
- 열 데이터 유형 변환
- SQL 함수는 인수를 사용하여 항상 값을 반환

# 단일 행 함수

- 데이터 조작을 위해 사용
- 인자(Argument)를 받아들여 각 행을 각각 일정한 형태로 변환하여 행 당 하나의 결과 반환
- 반환되는 각 행에 대해 작업 수행
- SELECT, WHERE, ORDER BY 절에 사용
- 데이터 타입 변경 가능
- 중첩 사용 가능
- 단일행 함수의 인수
  - 사용자 지정 상수
  - 변수 값
  - 컬럼(열) 이름
  - 표현식

# 단일 행 함수의 종류

## - 데이터 타입에 따라

- 문자함수 : 문자를 입력하면 문자 또는 숫자 값 반환
- 숫자함수 : 숫자를 입력하면 숫자 값 반환
- 날짜함수 : 날짜 유형의 값 연산
- 변환함수 : 데이터 유형 변환
- 기타함수 : 타입에 관계없이 적용 가능

# 문자함수 I

함수	설 명
LOWER	· 대상 문자열을 모두 소문자로 변환
UPPER	· 대상 문자열을 모두 대문자로 변환
INITCAP	· 문자열 단어의 첫 문자는 대문자로 나머지는 소문자로 변환 · 첫 문자는 파라미터로 들어오는 문자열에서 문자 구분자(delimiter, 일반적으로 공백, 탭 등)로 구분되는 첫 문자를 모두 말함
CONCAT(s1, s2)	· 두 문자열을 연결
SUBSTR(str, m, n)	· 문자열 중 지정한 위치 m에서 지정한 길이 n만큼의 문자열 추출 · m이 0인 경우, 디폴트 1로 설정 · m이 음수인 경우, 시작 위치는 오른쪽부터 설정 · n이 1보다 작을 경우, null 반환 · n 생략시, m부터 문자열의 끝까지 반환
SUBSTRB(str, m, n)	· SUBSTR과 기능은 동일하지만, n으로 명시된 숫자만큼 문자열의 바이트 수를 잘라내어 그 결과를 반환
INSTR(s1, s2, m, n)	· 문자열 s1 내의 m위치부터 검색하여 특정 문자열 s2가 n번째 등장하는 위치를 숫자로 표시 · m과 n은 생략 가능하며, 디폴트 값은 1
LENGTH(s)	· 문자의 길이를 숫자 값으로 표시(바이트 단위)
CHR(n)	· ASCII 코드값이 n인 문자 반환
ASCII(s)	· s 문자의 ASCII 코드값 반환



## 다음 결과 확인

```
SELECT first_name, SUBSTR(first_name, 1, 3), SUBSTR(first_name, 3),  
SUBSTR(first_name, -3, 2)  
FROM employees  
WHERE department_id = 10;
```

```
Run SQL Command Line
```

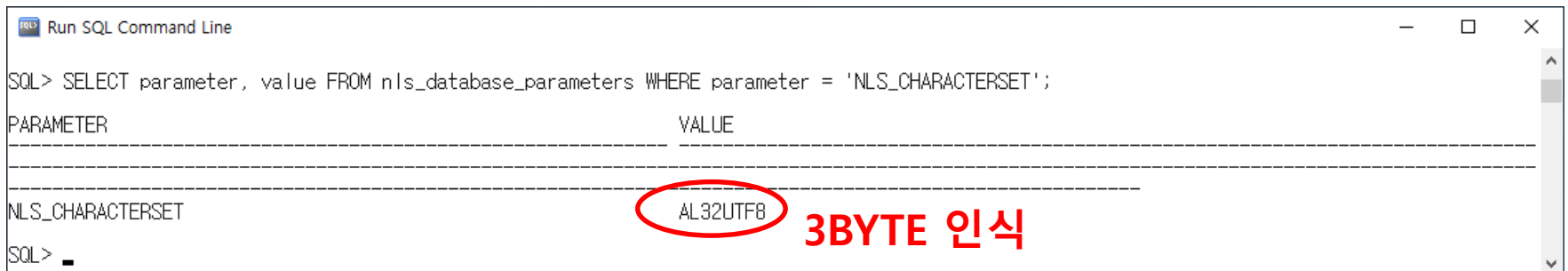
```
SQL> SELECT first_name, SUBSTR(first_name, 1, 3), SUBSTR(first_name, 3), SUBSTR(first_name, -3, 2)
2 FROM employees
3 WHERE department_id = 10;
```

FIRST_NAME	SUBSTR(FIRST_NAME,1,3)	SUBSTR(FIRST_NAME,3)	SUBSTR(FIRST_NAME,-3,2)
Mickey	Mic	key	ke
Jennifer	Jen	nnifer	fe

```
SQL>
```

# 현재 DB의 한글의 BYTE 수 조회

```
SELECT parameter, value  
FROM nls_database_parameters  
WHERE parameter = 'NLS_CHARACTERSET';
```



The screenshot shows a SQL Command Line window titled "Run SQL Command Line". The command entered is "SQL> SELECT parameter, value FROM nls\_database\_parameters WHERE parameter = 'NLS\_CHARACTERSET';". The output is a table with two columns: "PARAMETER" and "VALUE". The first row shows "NLS\_CHARACTERSET" as the parameter and "AL32UTF8" as the value. The value "AL32UTF8" is circled in red, and the text "3BYTE 인식" is written in red next to it.

PARAMETER	VALUE
NLS_CHARACTERSET	AL32UTF8

SQL> █

**3BYTE 인식**

# 현재 DB의 한글의 BYTE 수 조회 예시

```
SELECT SUBSTRB('I am here with you', 5, 3) 결과1, SUBSTR('나 여기 있어', 5, 3)
결과2, SUBSTRB('나 여기 있어', 5, 3) 결과3
FROM dual;
```

Run SQL Command Line

```
SQL> SELECT SUBSTRB('I am here with you', 5, 3) 결과1, SUBSTR('나 여기 있어', 5, 3) 결과2, SUBSTRB('나 여기 있어', 5, 3) 결과3 FROM dual;
```

?	결과2	결과3
he	있어	여

Run SQL Command Line

```
SQL> SELECT '+'||SUBSTRB('I am here with you', 5, 3)||'+ ' 결과1, '+'||SUBSTR('나 여기 있어', 5, 3)||'+ ' 결과2, '+'||SUBSTRB('나 여기 있어', 5, 3)||'+ ' 결과3 FROM dual;
```

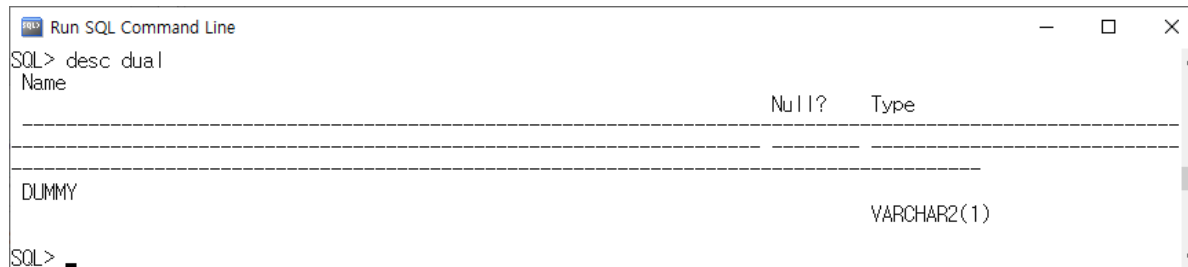
?	결과2	결과3
+ he+	+ 있어+	+여+

SQL> \_

한글 3byte처리 charset : (여)  
한글 2byte처리 charset : (기)

# dual Table

- 실제로는 존재하지는 않지만 존재하는 것처럼 동작
- MySQL에서는 Oracle과는 달리 `SELECT 3*4;`, `SELECT SYSDATE();`, `SELECT NOW();` 식으로 FROM 절이 없어도 가능
- 오라클 데이터베이스가 생성될 때 데이터 디렉터리에서 자동으로 생성
- 소유자는 SYS 사용자로서 모든 사용자들의 접근이 허용
- DUMMY라고 불리는 하나의 컬럼으로 구성되어 있으며 데이터형은 VARCHAR2(1)



```
Run SQL Command Line
SQL> desc dual
Name
-----
DUMMY
Type
-----
VARCHAR2(1)
SQL>
```

- ROW 또한 하나를 가지며 그 값은 'X'



```
Run SQL Command Line
SQL> select * from dual;
DU
--
X
SQL>
```

- 사용자 데이터가 있는 테이블에서 유래하지 않은 상수 값, 의사열(pseudo-column), 표현식 등의 값을 단 한번만 돌려거나 현재 날짜, 시각을 알고자 할 때 이용. 즉 일시적인 산술, 날짜 연산 등에 주로 이용

```
SQL> SELECT SYSDATE FROM dual;
SQL> SELECT SYSTIMESTAMP FROM dual;
SQL> SELECT 3*4 FROM dual;
```

# 문자함수 II

함수	설 명
LPAD(s1, n, [s2])	<ul style="list-style-type: none"> <li>· s1 문자열의 왼쪽 공간에 지정한 문자로 채움</li> <li>· s1을 n-length(s1)자리만큼의 s2로 채운 문자열로 만들어 반환</li> <li>· s1의 자릿수가 n보다 작은 경우 n-length(s1)만큼을 s2로 들어오는 문자열로 왼쪽을 채워 반환</li> <li>· s2 생략시 공백문자가 기본 적용</li> </ul>
RPAD(s1, n, [s2])	<ul style="list-style-type: none"> <li>· s1 문자열의 오른쪽 공간에 지정한 문자로 채움</li> <li>· 기능은 LPAD와 동일, 단, 채우는 위치가 오른쪽</li> </ul>
LTRIM(s, c)	<ul style="list-style-type: none"> <li>· 대상 문자열 s에서 c로 지정된 문자를 가장 왼쪽 끝에서 제거한 문자를 반환</li> <li>· c 생략시 기본값은 공백문자로 적용</li> </ul>
RTRIM(s, c)	<ul style="list-style-type: none"> <li>· 기능은 LTRIM과 동일, 단, 대상 문자열의 오른쪽부터 지정한 문자들을 제거</li> </ul>
TRIM([LEADING, TRAILING, BOTH][trim_char], trim_src)	<ul style="list-style-type: none"> <li>· 문자열의 왼쪽이나 오른쪽, 양쪽 모두에서 지정된 문자나 공백을 제거한 결과 반환</li> <li>· trim_char : 문자 하나</li> <li>· LEADING : 왼쪽에 있는 trim_char 제거</li> <li>· TRAILING : 오른쪽에 있는 trim_char 제거</li> <li>· BOTH : 양쪽 끝에 있는 trim_char 제거, 기본값</li> </ul>
TRANSLATE(s, from, to)	<ul style="list-style-type: none"> <li>· 문자열 하나를 일대일로 변환</li> <li>· from은 to보다 그 길이가 많아도 상관없지만 일대일로 대응되지 않는 문자는 NULL로 변환</li> <li>· to는 생략 불가능하고, NULL 가능 → 결과 NULL</li> </ul>
REPLACE(s, p, r)	<ul style="list-style-type: none"> <li>· p 문자열을 r 문자열로 대체하여 결과 반환</li> <li>· r은 생략 가능, 생략하거나 NULL이 올 경우 p 문자열을 제외한(즉, p 문자열을 NULL로 대체한) 결과 반환</li> </ul>

# LPAD / RPAD

```
SELECT first_name, LPAD(first_name, 15, '*'), salary, LPAD(salary, 10, '*')
FROM employees
WHERE department_id = 20;
```

Run SQL Command Line

```
SQL> SELECT first_name, LPAD(first_name, 15, '*'), salary, LPAD(salary, 10, '*')
2 FROM employees
3 WHERE department_id = 20;
```

FIRST_NAME	LPAD(FIRST_NAME,15,'*')	SALARY	LPAD(SALARY,10,'*')
-----	-----	-----	-----
Mary	*****Mary	3400	*****3400
Michael	*****Michael	13000	*****13000
Pat	*****Pat	6000	*****6000

SQL>

```
SELECT first_name, RPAD(first_name, 15, '*'), salary, RPAD(salary, 10, '*')
FROM employees
WHERE department_id = 20;
```

Run SQL Command Line

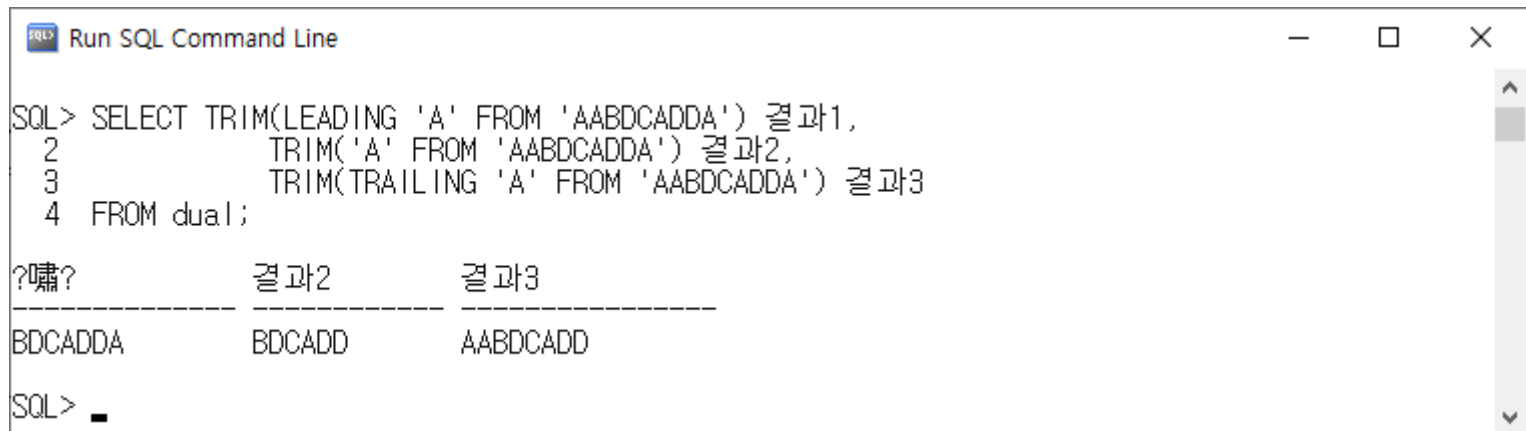
```
SQL> SELECT first_name, RPAD(first_name, 15, '*'), salary, RPAD(salary, 10, '*')
2 FROM employees
3 WHERE department_id = 20;
```

FIRST_NAME	RPAD(FIRST_NAME,15,'*')	SALARY	RPAD(SALARY,10,'*')
-----	-----	-----	-----
Mary	Mary*****	3400	3400*****
Michael	Michael*****	13000	13000*****
Pat	Pat*****	6000	6000*****

SQL>

## 다음 결과 확인

SELECT TRIM(LEADING 'A' FROM 'AABDCADDA') 결과1,  
TRIM('A' FROM 'AABDCADDA') 결과2,  
TRIM(TRAILING 'A' FROM 'AABDCADDA') 결과3  
FROM dual;



```
SQL> SELECT TRIM(LEADING 'A' FROM 'AABDCADDA') 결과1,  
2          TRIM('A' FROM 'AABDCADDA') 결과2,  
3          TRIM(TRAILING 'A' FROM 'AABDCADDA') 결과3  
4 FROM dual;
```

결과1	결과2	결과3
BDCADDA	BDCADD	AABDCADD

```
SQL> _
```

# 숫자함수

함수	설 명
ROUND(n, [m])	<ul style="list-style-type: none"> <li>· n 값을 반올림하는 함수</li> <li>· m이 양수이면 소수점 아래 반올림할 자릿수를 나타내고, 생략하면 디폴트 값 0</li> <li>· m이 음수이면 소수점 위 반올림할 자릿수를 나타냄</li> </ul>
TRUNC(c, [m])	<ul style="list-style-type: none"> <li>· m 자리에서 버림하는 함수</li> <li>· m이 양수이면, 소수점 아래 m 자리에서 버림을 나타내고, 생략하면 디폴트 값 0</li> <li>· m이 음수이면 소수점 위 m 자리에서 버림을 수행</li> </ul>
MOD(m, n)	<ul style="list-style-type: none"> <li>· m을 n으로 나눈 나머지 값을 반환</li> <li>· n이 0인 경우 m을 반환</li> <li>· 나머지 연산의 내부로직 나머지 값 = <math>m - n * \text{FLOOR}(m / n)</math></li> </ul>
ABS(n)	<ul style="list-style-type: none"> <li>· n을 절대값으로 변경</li> </ul>
FLOOR(n)	<ul style="list-style-type: none"> <li>· n 보다 작거나 같은 정수 중 가장 큰 정수 값을 반환(버림)</li> </ul>
CEIL(n)	<ul style="list-style-type: none"> <li>· n보다 크거나 같은 정수 중 가장 작은 정수 값을 반환(올림)</li> </ul>
SIGN(n)	<ul style="list-style-type: none"> <li>· n이 양수이면 1, 음수이면 -1, 0이면 0을 반환</li> </ul>
POWER(m, n)	<ul style="list-style-type: none"> <li>· m의 n 제곱값을 구하는 함수</li> </ul>



# ROUND

**SELECT ROUND(4567.678) 결과1, ROUND(4567.678, 0) 결과2,  
ROUND(4567.678, 2) 결과3, ROUND(4567.678, -2) 결과4  
FROM dual;**

SQL> Run SQL Command Line

SQL> SELECT ROUND(4567.678) 결과1, ROUND(4567.678, 0) 결과2, ROUND(4567.678, 2) 결과3, ROUND(4567.678, -2) 결과4  
2 FROM dual;

결과1	결과2	결과3	결과4
4568	4568	4567.68	4600

SQL> █

# TRUNC

```
SELECT TRUNC(4567.678) 결과1, TRUNC(4567.678, 0) 결과2, TRUNC(4567.678, 2) 결과3, TRUNC(4567.678, -2) 결과4  
FROM dual;
```

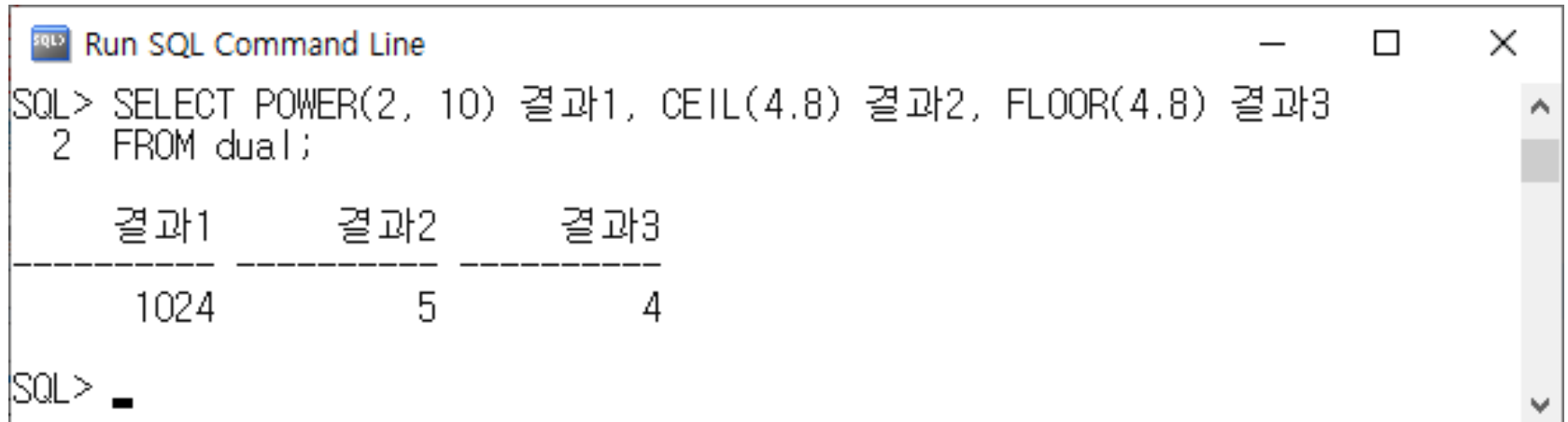
```
SQL> Run SQL Command Line
SQL> SELECT TRUNC(4567.678) 결과1, TRUNC(4567.678, 0) 결과2, TRUNC(4567.678, 2) 결과3, TRUNC(4567.678, -2) 결과4
2 FROM dual;
```

결과1	결과2	결과3	결과4
4567	4567	4567.67	4500

```
SQL>
```

# POWER / CEIL / FLOOR

**SELECT POWER(2, 10) 결과1, CEIL(4.8) 결과2, FLOOR(4.8) 결과3  
FROM dual;**



The screenshot shows a window titled "Run SQL Command Line" with a standard Windows interface (minimize, maximize, close buttons). The command prompt shows the following SQL query being executed:

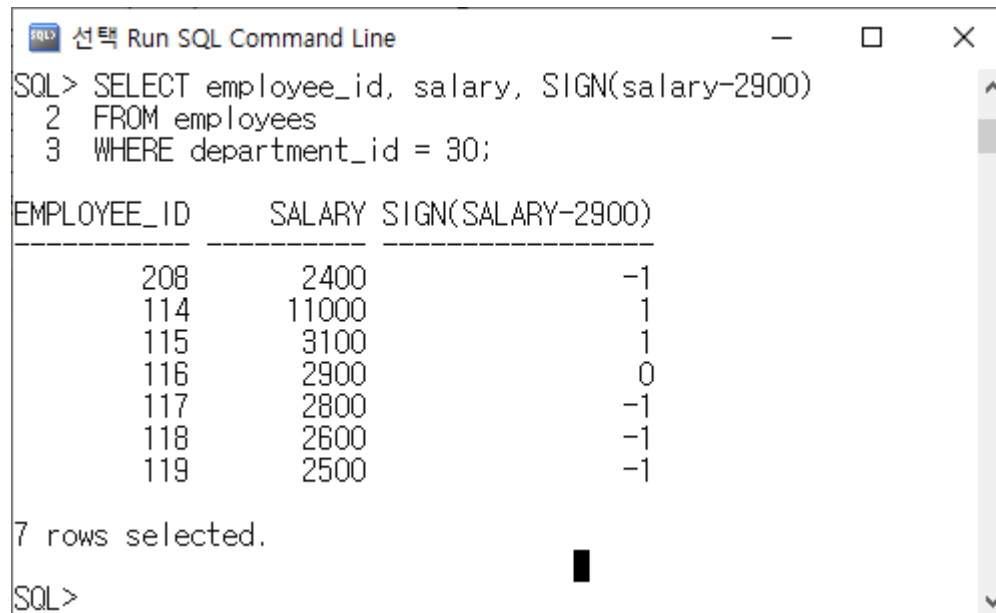
```
SQL> SELECT POWER(2, 10) 결과1, CEIL(4.8) 결과2, FLOOR(4.8) 결과3  
2 FROM dual;
```

The results are displayed in a table format with three columns: 결과1, 결과2, and 결과3. The first row of data shows the values 1024, 5, and 4 respectively. The prompt "SQL>" is visible at the bottom left of the window.

결과1	결과2	결과3
1024	5	4

# SIGN

```
SELECT employee_id, salary, SIGN(salary-2900)
FROM employees
WHERE department_id = 30;
```



The screenshot shows a SQL Command Line window titled "선택 Run SQL Command Line". The command entered is:

```
SQL> SELECT employee_id, salary, SIGN(salary-2900)
2 FROM employees
3 WHERE department_id = 30;
```

The results are displayed in a table with three columns: EMPLOYEE\_ID, SALARY, and SIGN(SALARY-2900). The data is as follows:

EMPLOYEE_ID	SALARY	SIGN(SALARY-2900)
208	2400	-1
114	11000	1
115	3100	1
116	2900	0
117	2800	-1
118	2600	-1
119	2500	-1

Below the table, it says "7 rows selected." and the prompt "SQL>" is visible at the bottom.

# 날짜 계산 결과

- 오라클 DB는 세기, 연도, 월, 일, 시, 분, 초 형태의 내부 숫자 형식으로 날짜 저장

연산	결 과
DATE + NUMBER	· 날짜(DATE)에 해당 일수(NUMBER)를 더한 결과를 날짜 형태로 결과 표현
DATE - NUMBER	· 날짜(DATE)에 해당 일수(NUMBER)를 뺀 결과를 날짜 형태로 결과 표현
DATE1 - DATE2	· 두 날짜(DATE1, DATE2) 사이의 결과 일수를 숫자로 표현
DATE + NUMBER / 24	· 날짜에 해당 시간을 더해서 날짜 형태로 결과 표현

# 오라클의 날짜/시간 포맷 조회

- 오라클의 날짜 포맷 조회

```
SELECT value FROM nls_session_parameters  
WHERE parameter = 'NLS_DATE_FORMAT';
```

Run SQL Command Line

```
SQL> SELECT value FROM nls_session_parameters  
2 WHERE parameter = 'NLS_DATE_FORMAT';
```

VALUE

RR/MM/DD

'NLS\_TIMESTAMP\_FORMAT'  
'NLS\_TIMESTAMP\_TZ\_FORMAT'  
'NLS\_LANGUAGE'

Run SQL Command Line

```
SQL> SELECT value FROM nls_session_parameters  
2 WHERE parameter = 'NLS_TIMESTAMP_FORMAT';
```

VALUE

RR/MM/DD HH24:MI:SSXFF

Run SQL Command Line

```
SQL> SELECT value FROM nls_session_parameters  
2 WHERE parameter = 'NLS_TIMESTAMP_TZ_FORMAT';
```

VALUE

RR/MM/DD HH24:MI:SSXFF TZR

Run SQL Command Line

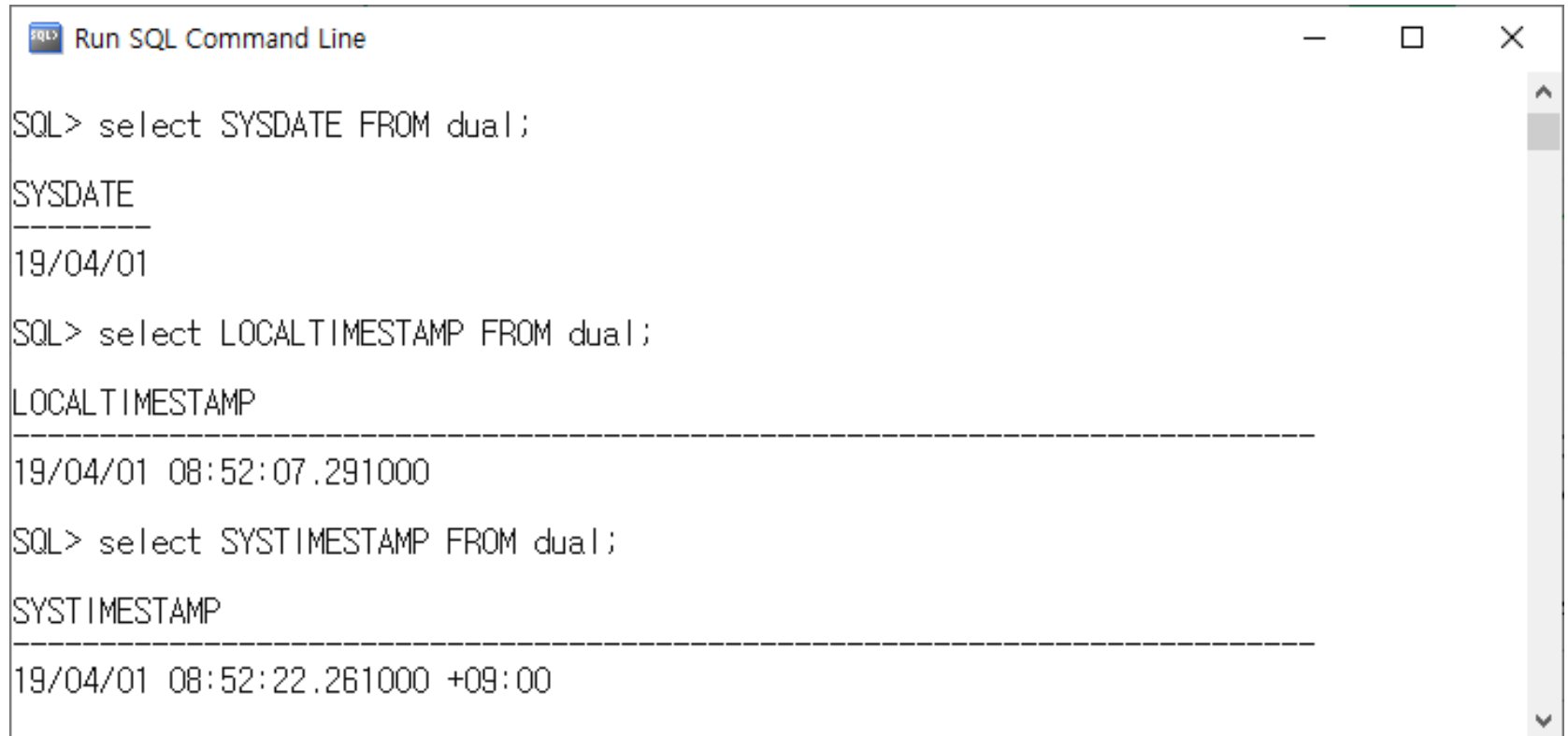
```
SQL> SELECT value FROM nls_session_parameters  
2 WHERE parameter = 'NLS_LANGUAGE';
```

VALUE

KOREAN

# 오라클의 현재 시간

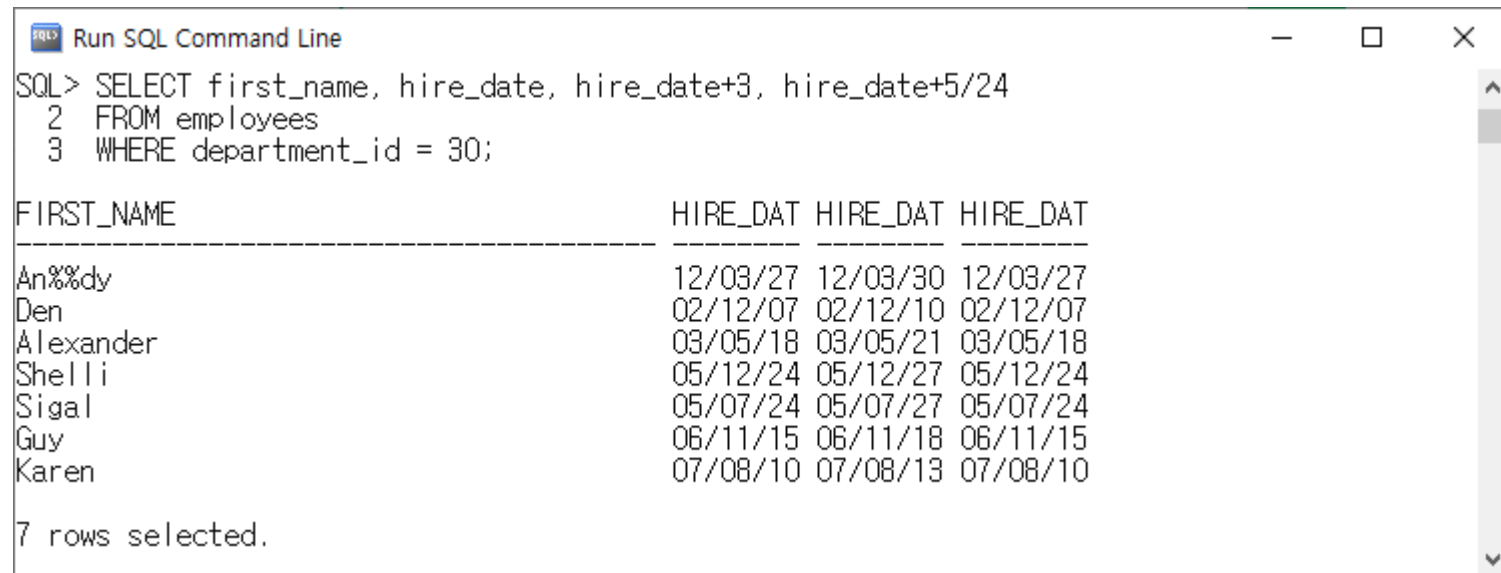
- 현재시간 조회 방법



```
SQL> select SYSDATE FROM dual;  
SYSDATE  
-----  
19/04/01  
  
SQL> select LOCALTIMESTAMP FROM dual;  
LOCALTIMESTAMP  
-----  
19/04/01 08:52:07.291000  
  
SQL> select SYSTIMESTAMP FROM dual;  
SYSTIMESTAMP  
-----  
19/04/01 08:52:22.261000 +09:00
```

## 날짜 계산 결과

```
SELECT first_name, hire_date, hire_date+3, hire_date+5/24
FROM employees
WHERE department_id = 30;
```



The screenshot shows a 'Run SQL Command Line' window with the following SQL query and its results:

```
SQL> SELECT first_name, hire_date, hire_date+3, hire_date+5/24
2  FROM employees
3  WHERE department_id = 30;
```

FIRST_NAME	HIRE_DAT	HIRE_DAT	HIRE_DAT
An%dy	12/03/27	12/03/30	12/03/27
Den	02/12/07	02/12/10	02/12/07
Alexander	03/05/18	03/05/21	03/05/18
Shelli	05/12/24	05/12/27	05/12/24
Sigal	05/07/24	05/07/27	05/07/24
Guy	06/11/15	06/11/18	06/11/15
Karen	07/08/10	07/08/13	07/08/10

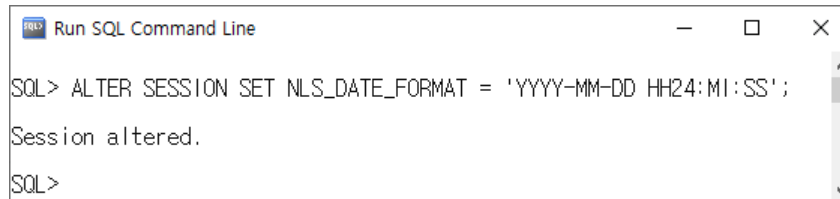
7 rows selected.



# 오라클의 날짜 포맷 변환

- 오라클의 날짜 포맷 변환

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
```



```
Run SQL Command Line
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
Session altered.
SQL>
```



```
Run SQL Command Line
SQL> SELECT first_name, hire_date, hire_date+3, hire_date+5/24
2 FROM employees
3 WHERE department_id = 30;
```

FIRST_NAME	HIRE_DATE	HIRE_DATE+3	HIRE_DATE+5/24
An%dy	2012-03-27 00:00:00	2012-03-30 00:00:00	2012-03-27 05:00:00
Den	2002-12-07 00:00:00	2002-12-10 00:00:00	2002-12-07 05:00:00
Alexander	2003-05-18 00:00:00	2003-05-21 00:00:00	2003-05-18 05:00:00
Shelli	2005-12-24 00:00:00	2005-12-27 00:00:00	2005-12-24 05:00:00
Sigal	2005-07-24 00:00:00	2005-07-27 00:00:00	2005-07-24 05:00:00
Guy	2006-11-15 00:00:00	2006-11-18 00:00:00	2006-11-15 05:00:00
Karen	2007-08-10 00:00:00	2007-08-13 00:00:00	2007-08-10 05:00:00

```
7 rows selected.
SQL>
```

# EXTRACT 함수

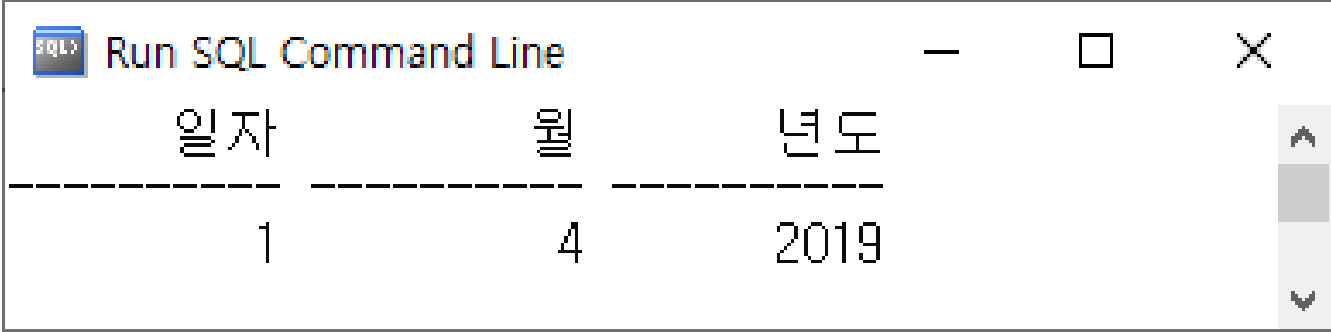
- datetime 또는 interval 값 표현식에 지정된 datetime 필드의 값을 추출하여 반환

```
EXTRACT([YEAR][MONTH][DAY][HOUR][MINUTE][SECOND]  
        [TIMEZONE_HOUR][TIMEZONE_MINUTE ]  
        [TIMEZONE_REGION ][TIMEZONE_ABBR]  
FROM [date_value_expression][interval_value_expression]);
```



## 오늘의 날짜를 일, 월, 년도 순으로 출력

```
SELECT EXTRACT(DAY FROM SYSDATE) 일자, EXTRACT(MONTH  
FROM SYSDATE) 월, EXTRACT(YEAR FROM SYSDATE) 년도  
FROM dual;
```



일자	월	년도
1	4	2019

## 다음 실행결과 확인

LOCALTIMESTAMP  
로 변경하여 실행

```
SELECT SYSTIMESTAMP A, EXTRACT(HOUR FROM SYSTIMESTAMP)
B, TO_CHAR(SYSTIMESTAMP, 'HH24') C
FROM dual;
```

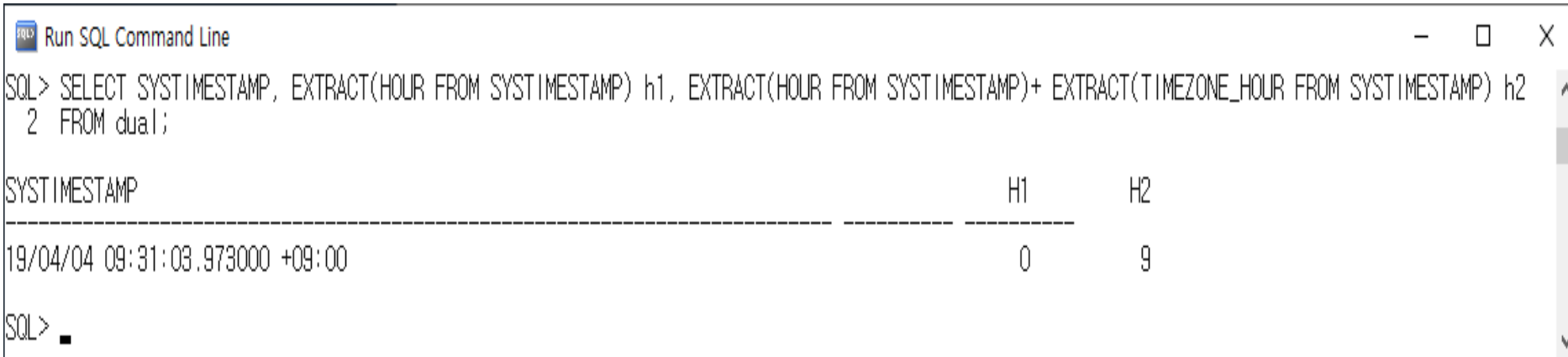
Run SQL Command Line			—	□	×
A	B	C			
19/04/01 10:17:08.613000 +09:00	1	10			
SQL>					

Run SQL Command Line			—	□	×
A	B	C			
19/04/01 14:07:03.775000 +09:00	5	14			
SQL>					

Run SQL Command Line			—	□	×
A	B	C			
19/04/01 14:11:01.980000 +09:00	14	14			
SQL>					

## 다음 실행결과 확인

```
SELECT SYSTIMESTAMP, EXTRACT(HOUR FROM SYSTIMESTAMP) h1,  
EXTRACT(HOUR FROM SYSTIMESTAMP)+  
EXTRACT(TIMEZONE_HOUR FROM SYSTIMESTAMP) h2  
FROM dual;
```



The screenshot shows a SQL Command Line window titled "Run SQL Command Line". The command entered is: `SQL> SELECT SYSTIMESTAMP, EXTRACT(HOUR FROM SYSTIMESTAMP) h1, EXTRACT(HOUR FROM SYSTIMESTAMP)+ EXTRACT(TIMEZONE_HOUR FROM SYSTIMESTAMP) h2 FROM dual;`. The output is displayed in a table format with three columns: SYSTIMESTAMP, H1, and H2. The first row shows the current timestamp and the calculated values for H1 and H2.

SYSTIMESTAMP	H1	H2
19/04/04 09:31:03.973000 +09:00	0	9

SQL> █

# 날짜함수

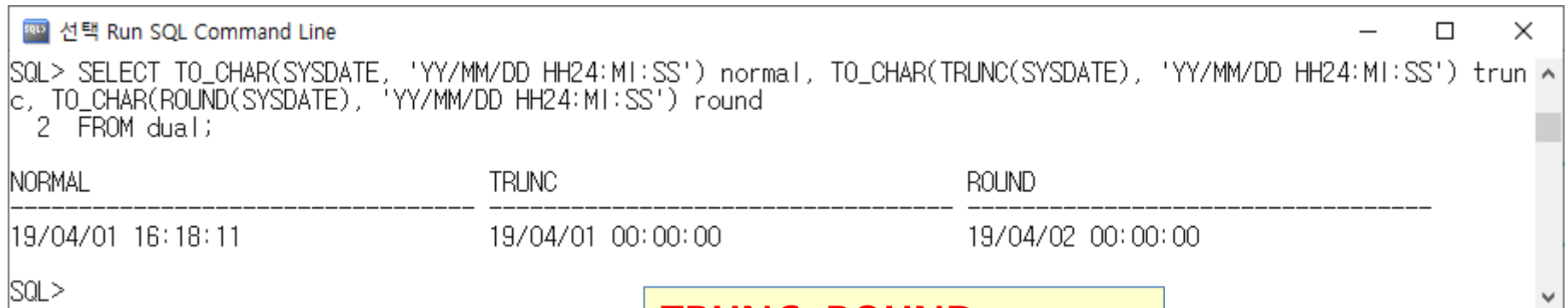
함수	설 명
MONTHS_BETWEEN(d1, d2)	<ul style="list-style-type: none"> <li>· d1, d2 두 날짜 간의 개월 수 반환</li> <li>· 반환하는 데이터 타입은 숫자형</li> <li>· d1 = d2일 경우 0 반환</li> </ul>
ADD_MONTH(d, n)	<ul style="list-style-type: none"> <li>· d 날짜에 n개월을 더한 결과 반환</li> <li>· 반환하는 데이터 타입은 날짜형</li> <li>· n은 일반적으로 정수형이나 실수형 가능(정수형으로 자동변환하여 연산)</li> </ul>
NEXT_DAY(d, 'CHAR')	<ul style="list-style-type: none"> <li>· d보다 이후로 지정한 요일에 해당되는 날짜</li> <li>· CHAR : 요일명 입력, 숫자 입력 가능(일요일 1, 월요일 2, ... 토요일 7)</li> </ul>
LAST_DAY	<ul style="list-style-type: none"> <li>· 해당 월의 마지막 날짜를 리턴</li> </ul>
ROUND(d, fm)	<ul style="list-style-type: none"> <li>· 날짜를 포맷모델인 fm(년, 월, 일) 단위로 반올림</li> <li>· 일 : 정오를 넘으면 다음날 자정, 아니면 그날 자정으로 출력</li> <li>· 월 : 15일 이상이면 다음달 1일, 아니면 현재 달 1일 출력</li> <li>· 년 : 6월이 넘으면 다음해 1월1일, 아니면 그해 1월 1일 출력</li> </ul>
TRUNC(d, fm)	<ul style="list-style-type: none"> <li>· 날짜를 포맷모델인 fm(년, 월, 일) 단위로 버림</li> <li>· 일 : 해당일 자정 출력</li> <li>· 월 : 해당월 1일 출력</li> <li>· 년 : 해당년도 1월 1일 출력</li> </ul>

# 포맷모델

포맷 모델	단 위
YEAR, YYYY, YYY, YY, Y	<ul style="list-style-type: none"><li>· 년</li><li>· 7월1일부터 반올림</li></ul>
Q	<ul style="list-style-type: none"><li>· 분기</li><li>· 한 분기 두번째 달의 16일부터 반올림</li></ul>
MONTH, MON, MM, RM	<ul style="list-style-type: none"><li>· 월</li><li>· 16일부터 반올림</li></ul>
DD	<ul style="list-style-type: none"><li>· 일</li><li>· 오후 12시부터 반올림</li></ul>
HH	<ul style="list-style-type: none"><li>· 시간</li><li>· 30분부터 반올림</li></ul>
MI	<ul style="list-style-type: none"><li>· 분</li><li>· 30초부터 반올림</li></ul>

## 다음 실행결과 확인

```
SELECT TO_CHAR(SYSDATE, 'YY/MM/DD HH24:MI:SS') normal,  
TO_CHAR(TRUNC(SYSDATE), 'YY/MM/DD HH24:MI:SS') trunc,  
TO_CHAR(ROUND(SYSDATE), 'YY/MM/DD HH24:MI:SS') round  
FROM dual;
```



```
SQL> SELECT TO_CHAR(SYSDATE, 'YY/MM/DD HH24:MI:SS') normal, TO_CHAR(TRUNC(SYSDATE), 'YY/MM/DD HH24:MI:SS') trunc,  
2  TO_CHAR(ROUND(SYSDATE), 'YY/MM/DD HH24:MI:SS') round  
FROM dual;
```

NORMAL	TRUNC	ROUND
19/04/01 16:18:11	19/04/01 00:00:00	19/04/02 00:00:00

SQL>

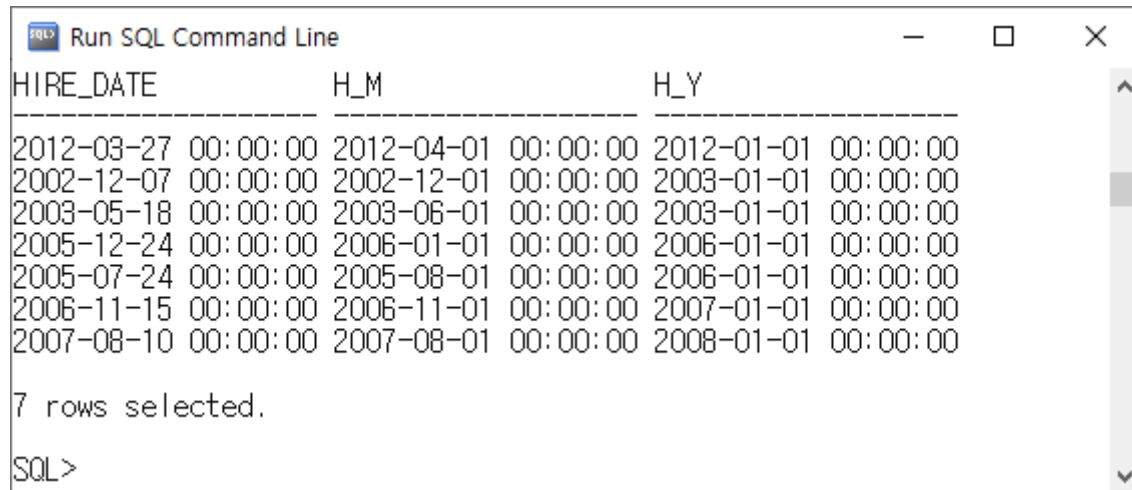
### TRUNC, ROUND

- 날짜 부분만 출력
- 시간은 자정으로 setting



## 다음 실행결과 확인

```
SELECT hire_date, ROUND(hire_date, 'MONTH') h_m, ROUND(hire_date, 'YEAR') h_y  
FROM employees  
WHERE department_id = 30;
```

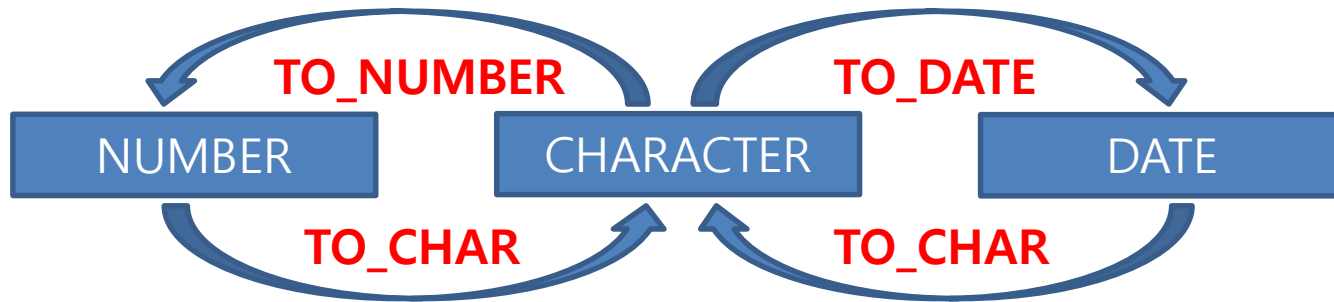


HIRE_DATE	H_M	H_Y
2012-03-27 00:00:00	2012-04-01 00:00:00	2012-01-01 00:00:00
2002-12-07 00:00:00	2002-12-01 00:00:00	2003-01-01 00:00:00
2003-05-18 00:00:00	2003-06-01 00:00:00	2003-01-01 00:00:00
2005-12-24 00:00:00	2006-01-01 00:00:00	2006-01-01 00:00:00
2005-07-24 00:00:00	2005-08-01 00:00:00	2006-01-01 00:00:00
2006-11-15 00:00:00	2006-11-01 00:00:00	2007-01-01 00:00:00
2007-08-10 00:00:00	2007-08-01 00:00:00	2008-01-01 00:00:00

7 rows selected.  
SQL>

# 변환함수

- 암시적 변환 : 변환함수 없이도 자동으로 변환
- 명시적 변환 : 자동으로 변환되지 않을 때 변환함수 이용하여 변환



- 특정 데이터형을 다양한 형식으로 출력하고 싶을 때 사용
- 함수이름은 일반적으로 TO\_데이터형식
- 문자데이터형과 숫자데이터형의 연산 시 문자데이터형이나 숫자데이터형으로 변환하여 연산 → 암시적(implicit) 형 변환
- 암시적 형변환의 경우 단점
  - 성능저하발생 가능
  - 자동적으로 데이터베이스가 알아서 계산하지 않는 경우가 있어 에러 발생 가능→ 명시적 형변환 사용이 바람직

# TO\_CHAR(숫자, format)

요소	설 명	예 제	결 과 (1234)
9	· 숫자 위치 · 9의 개수가 표시 폭 (width)을 결정	999999	1234
0	· 선행 제로 표시	099999	001234
\$	· 부동 달러 기호 표시	\$999999	\$1234
L	· 부동 지역 통화기호(Local 통화 단위)	L999999	FF1234
.	· 지정된 위치에 소수점 표시	999999.99	1234.00
,	· 지정된 위치에 쉼표 표시	999,999	1,234
MI	· 음수 값의 경우 빼기 기호를 오른쪽에 표시	999999MI	1234-
PR	· 음수 값의 경우 괄호로 묶어 표시	999999PR	(1234)
EEEE	· 과학 표기(형식에 4개의 E를 표시하여야 함)	99.999EEEE	1.23E+03
V	· 10을 n번 곱함(n은 V 뒤에 나오는 9의 개수)	9999V99	123400
B	· 0 값을 0이 아닌 공백으로 표시	B9999.99	1234.99

# TO\_CHAR(날짜, format)

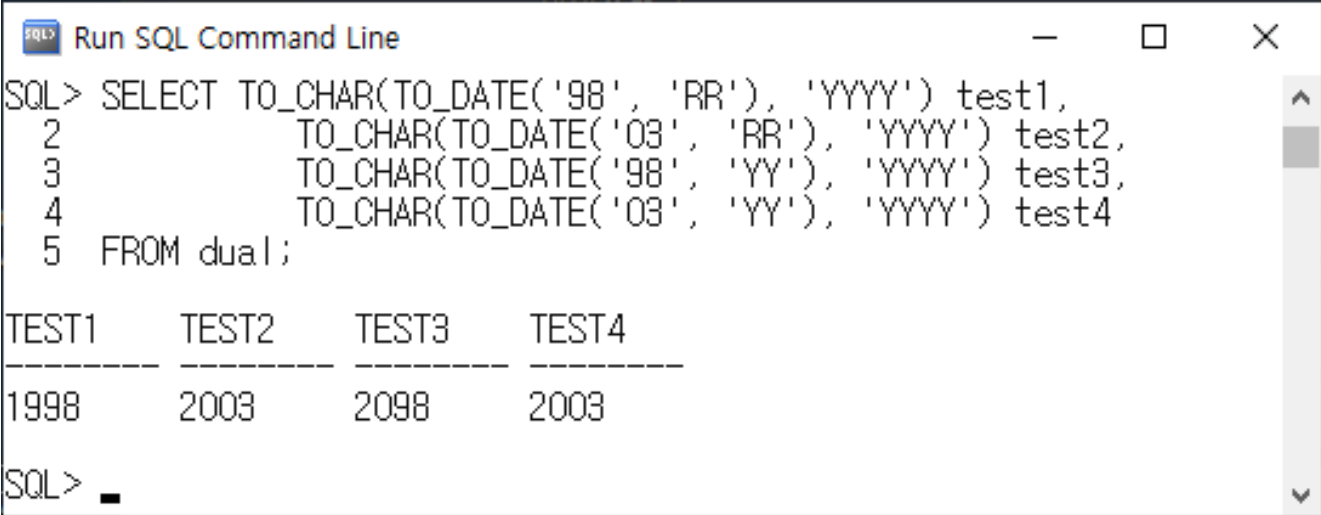
요소	설 명
SCC 또는 CC	· 세기. 기원전 날짜에는 -를 접두어로 붙임
날짜의 연도 YYYY 또는 SYYYY 또는 RRRR	· 연도. 기원전 날짜에는 -를 접두어로 붙임
YYY 또는 YY 또는 Y	· 연도의 마지막 세 자리, 두 자리 또는 한 자리
Y,YYY	· 해당 위치에 쉼표가 있는 연도
IYYY 또는 IYY 또는 IY 또는 I	· ISO 표준에 따르는 네 자리, 세 자리, 두 자리 또는 한 자리 연도
SYEAR 또는 YEAR	· 연도(문자). 기원전 날짜에는 -를 접두어로 붙임
BC 또는 AD	· BC/AD 표시자
B.C. 또는 A.D.	· 마침표가 있는 B.C./A.D. 표시자
Q	· 일년 중의 분기
MM	· 월 : 두 자리 값
MONTH 또는 Month 또는 month	· 9 문자길이가 되도록 공백을 채운 달의 이름
MON	· 달의 이름. 세자 약어
RM	· 로마 숫자 달
WW 또는 W	· 일년 중의 주(WW) 또는 한달 중의 주(W)
DDD 또는 DD 또는 D	· 일년 중의 일(DDD), 한달 중의 일(DD), 한주 중의 일(D)
DAY	· 요일의 이름. 세 장 약어
J	· 율리우스력의 일. 기원전 4713년 12월 31일부터의 일 수
AM 또는 PM	· 오전/오후 표시자
HH 또는 HH12 또는 HH24	· 하루 중의 시 또는 1-12 또는 0-24으로 표시되는 시
MI	· 분(0-59)
SS	· 초(0-59)
SSSSS	· 자정부터의 초(0-86399)

# TO\_DATE

- <char>를 날짜형으로 변환하여 리턴

## 다음 실행결과 확인

```
SELECT TO_CHAR(TO_DATE('98', 'RR'), 'YYYY') test1,  
       TO_CHAR(TO_DATE('03', 'RR'), 'YYYY') test2,  
       TO_CHAR(TO_DATE('98', 'YY'), 'YYYY') test3,  
       TO_CHAR(TO_DATE('03', 'YY'), 'YYYY') test4  
FROM dual;
```



The screenshot shows a window titled "Run SQL Command Line" with a SQL prompt. The query is executed, and the results are displayed in a table format. The table has four columns: TEST1, TEST2, TEST3, and TEST4. The values are 1998, 2003, 2098, and 2003 respectively.

```
SQL> SELECT TO_CHAR(TO_DATE('98', 'RR'), 'YYYY') test1,  
2         TO_CHAR(TO_DATE('03', 'RR'), 'YYYY') test2,  
3         TO_CHAR(TO_DATE('98', 'YY'), 'YYYY') test3,  
4         TO_CHAR(TO_DATE('03', 'YY'), 'YYYY') test4  
5 FROM dual;
```

TEST1	TEST2	TEST3	TEST4
1998	2003	2098	2003

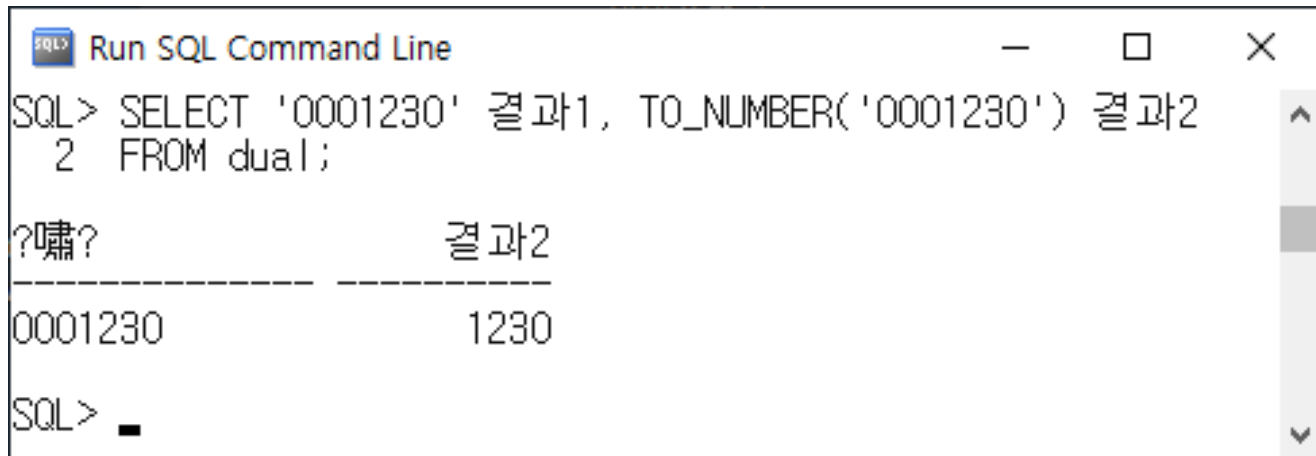
```
SQL> _
```

# TO\_NUMBER

- <char>를 숫자형으로 변환하여 리턴

## 다음 실행결과 확인

```
SELECT '0001230' 결과1, TO_NUMBER('0001230') 결과2  
FROM dual;
```



The screenshot shows a window titled "Run SQL Command Line". The command entered is: `SQL> SELECT '0001230' 결과1, TO_NUMBER('0001230') 결과2  
2 FROM dual;`. The output is displayed in a table format with two columns: "결과1" and "결과2". The first row of data shows "0001230" under "결과1" and "1230" under "결과2". The prompt "SQL> \_" is visible at the bottom.

결과1	결과2
0001230	1230

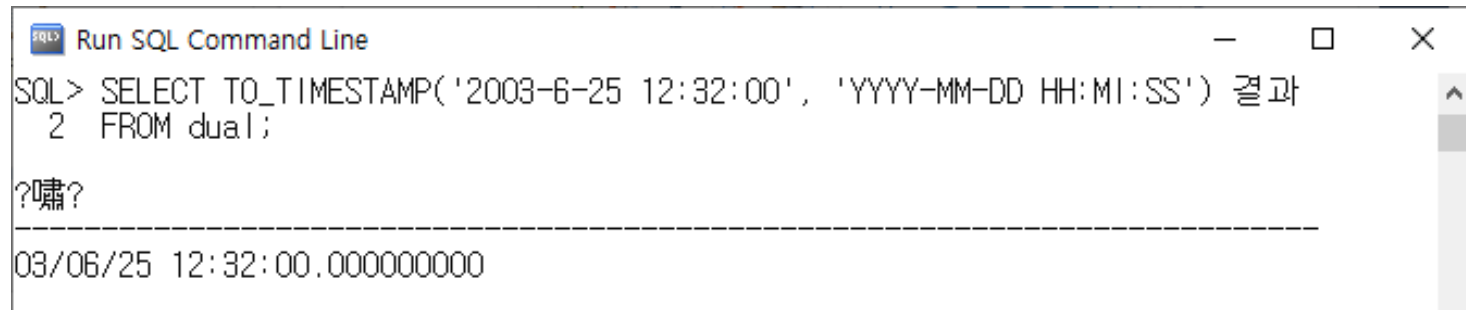


# 기타 변환함수

- **TO\_TIMESTAMP**
  - 지정한 문자열을 TIMESTAMP 날짜 데이터 유형으로 변환
- **TO\_YMINTERVAL**
  - 지정한 문자열을 INTERVAL YEAR TO MONTH 형태로 변환
- **TO\_DSINTERVAL**
  - 지정한 문자열을 INTERVAL DAY TO SECOND 값으로 변환

## 다음 실행결과 확인

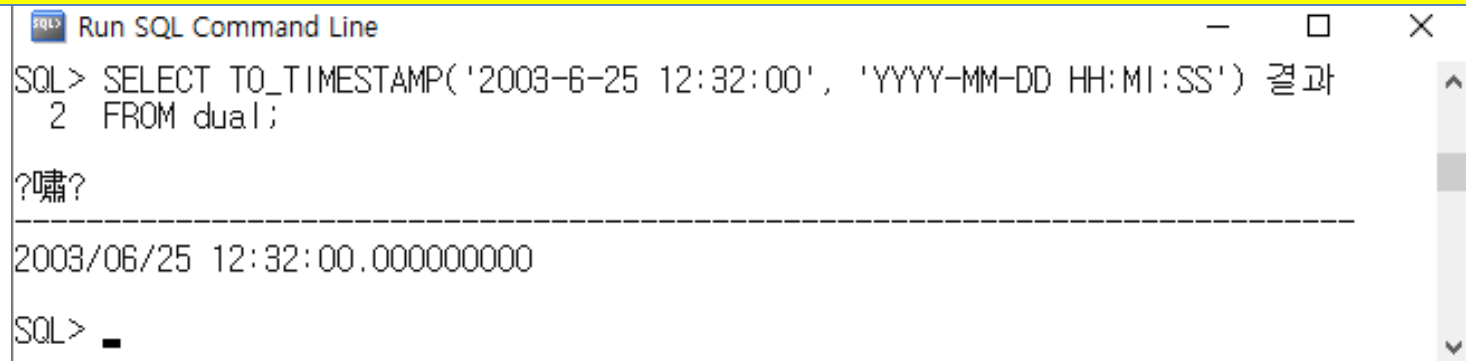
**SELECT TO\_TIMESTAMP('2003-6-25 12:32:00', 'YYYY-MM-DD HH:MI:SS') 결과  
FROM dual;**



```
SQL> Run SQL Command Line
SQL> SELECT TO_TIMESTAMP('2003-6-25 12:32:00', 'YYYY-MM-DD HH:MI:SS') 결과
2 FROM dual;

?肅?
-----
03/06/25 12:32:00.000000000
```

**ALTER SESSION SET NLS\_TIMESTAMP\_FORMAT = 'YYYY/MM/DD HH24:MI:SS.FF';**



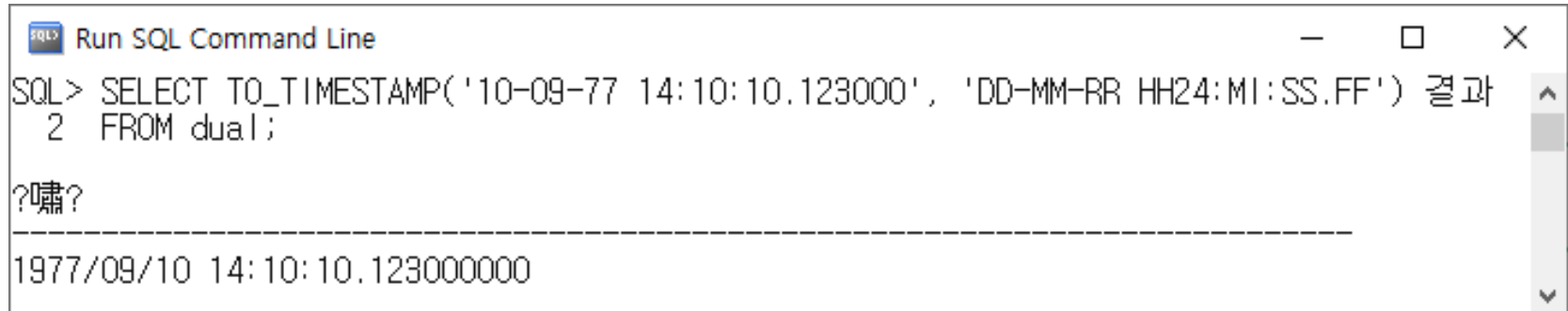
```
SQL> Run SQL Command Line
SQL> SELECT TO_TIMESTAMP('2003-6-25 12:32:00', 'YYYY-MM-DD HH:MI:SS') 결과
2 FROM dual;

?肅?
-----
2003/06/25 12:32:00.000000000

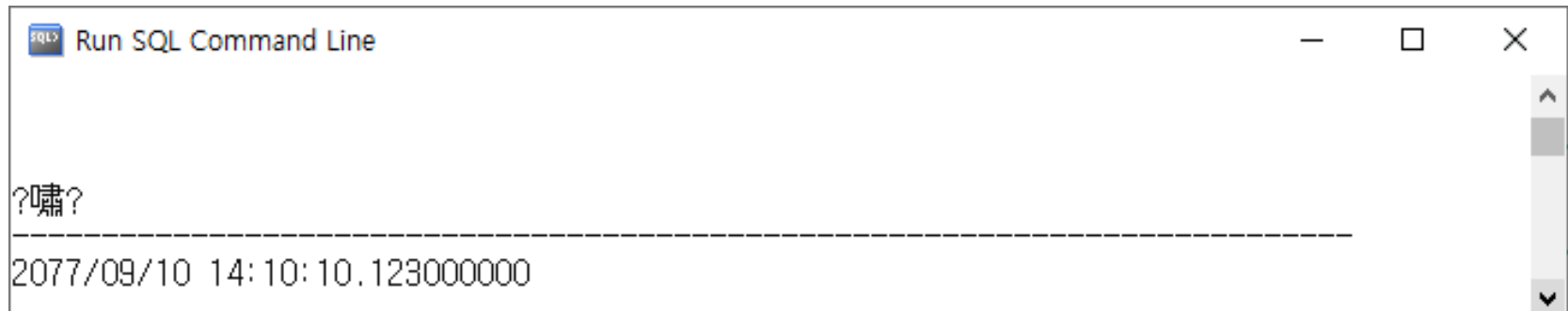
SQL> _
```

## 다음 실행결과 확인

```
SELECT TO_TIMESTAMP('10-09-77 14:10:10.123000',  
'DD-MM-RR HH24:MI:SS.FF') 결과  
FROM dual;
```



```
SQL> SELECT TO_TIMESTAMP('10-09-77 14:10:10.123000', 'DD-MM-RR HH24:MI:SS.FF') 결과  
2 FROM dual;  
  
?뽀?  
-----  
1977/09/10 14:10:10.123000000
```



```
SQL> SELECT TO_TIMESTAMP('20-09-77 14:10:10.123000', 'DD-MM-RR HH24:MI:SS.FF') 결과  
2 FROM dual;  
  
?뽀?  
-----  
2077/09/10 14:10:10.123000000
```

# 일반함수

함수	설 명
NVL(col, exp)	<ul style="list-style-type: none"> <li>· NULL을 다른 값으로 바꿀 때 사용</li> <li>· col이 NULL이면 exp 반환</li> <li>· col이 NULL이 아니면 col 반환</li> <li>· 모든 데이터 타입에 적용 가능하지만, col과 exp의 데이터 타입은 일치해야 함</li> </ul>
NVL2(exp1, exp2, exp3)	<ul style="list-style-type: none"> <li>· exp1이 NULL이 아닌 경우 exp2 반환</li> <li>· exp1이 NULL인 경우 exp3 반환</li> <li>· exp1는 임의 데이터형 가능</li> <li>· exp2와 exp3 데이터 타입 일치해야 함</li> </ul>
NULLIF(exp1, exp2)	<ul style="list-style-type: none"> <li>· exp1 = exp2이면 NULL 반환</li> <li>· exp1 != exp2이면 exp1 반환</li> <li>· exp1에는 NULL 지정 불가능</li> </ul>
COALESCE(exp1, ..., expN)	<ul style="list-style-type: none"> <li>· exp1, exp2, ..., expN 중 NULL이 아닌 첫 번째 인수 반환</li> <li>· 모든 인수가 NULL인 경우 NULL 반환</li> </ul>
GREATEST(exp1, ..., expN)	<ul style="list-style-type: none"> <li>· exp1, exp2, ..., expN 중 가장 큰 값을 반환</li> </ul>
LEAST(exp1, ..., expN)	<ul style="list-style-type: none"> <li>· exp1, exp2, ..., expN 중 가장 작은 값을 반환</li> </ul>
DECODE(value, if1, then1, if2, then2...)	<ul style="list-style-type: none"> <li>· 일반적인 프로그래밍의 if 문과 동일한 기능</li> <li>· FROM 절을 제외한 모든 절에서 사용 가능</li> <li>· 비교 연산은 '='만 가능</li> <li>· 일치하는 조건을 찾지 못하면 default 값 반환. default 값 생략시 NULL 반환</li> </ul>
CASE WHEN 비교조건1 THEN 처리1 ... WHEN 비교조건N THEN 처리N ELSE 디폴트처리 END	<ul style="list-style-type: none"> <li>· = 조건일 경우 CASE 항목 WHEN 비교값 형식으로 사용 가능</li> </ul>