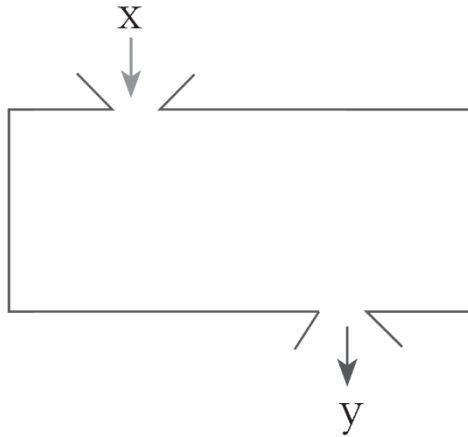


Chap06. 데이터 처리와 가공을 위한 오라클 함수

- ▶ 06-1 오라클 함수
- ▶ 06-2 문자 데이터를 가공하는 문자 함수
- ▶ 06-3 숫자 데이터를 연산하고 수치를 조정하는 숫자함수
- ▶ 06-4 날짜 데이터를 다루는 날짜 함수
- ▶ 06-5 자료형을 반환하는 형 변환 함수
- ▶ 06-6 NULL 처리 함수
- ▶ 06-7 상황에 따라 다른 데이터를 반환하는 DECODE 함수
와 CASE문

06-1 오라클 함수

▶ 함수



▶ 오라클 함수의 종류

- ▶ 내장 함수(built-in function)
- ▶ 사용자 정의 함수(user-defined function)

06-1 오라클 함수

▶ 내장 함수의 종류

▶ 단일행 함수(single-row function)

열 1	열 2	...	열 N		열 1	열 2	...	열 N
행 1				→	행 1			
행 2				→	행 2			
...				→	...			
행 N				→	행 N			

▶ 다중행 함수(multiple-row function)

열 1	열 2	...	열 N		열 1	열 2	...	열 N
행 1				→	행 1			
행 2								
...								
행 N								

06-2 문자 데이터를 가공하는 문자 함수

▶ UPPER, LOWER, INITCAP : 대소문자 변환

■ SELECT절 기본 사용

실습 6-1 UPPER, LOWER, INITCAP 함수 사용하기

```
01 SELECT ENAME, UPPER(ENAME), LOWER(ENAME), INITCAP(ENAME)
02 FROM EMP;
```

:: 결과 화면

ENAME	UPPER(ENAME)	LOWER(ENAME)	INITCAP(ENAME)
SMITH	SMITH	smith	Smith
ALLEN	ALLEN	allen	Allen
WARD	WARD	ward	Ward
JONES	JONES	jones	Jones
MARTIN	MARTIN	martin	Martin
BLAKE	BLAKE	blake	Blake
CLARK	CLARK	clark	Clark
SCOTT	SCOTT	scott	Scott
KING	KING	king	King
TURNER	TURNER	turner	Turner
ADAMS	ADAMS	adams	Adams
JAMES	JAMES	james	James
FORD	FORD	ford	Ford
MILLER	MILLER	miller	Miller

■ WHERE절에서의 활용

실습 6-2 UPPER 함수로 문자열 비교하기(사원 이름이 SCOTT인 데이터 찾기)

```
01 SELECT *
02 FROM EMP
03 WHERE UPPER(ENAME) = UPPER('scott');
```

■ LIKE문과 함께 활용

실습 6-3 UPPER 함수로 문자열 비교하기(사원 이름에 SCOTT 단어를 포함한 데이터 찾기)

```
01 SELECT *
02 FROM EMP
03 WHERE UPPER(ENAME) LIKE UPPER('%scott%');
```

06-2 문자 데이터를 가공하는 문자 함수

- ▶ **LENGTH** : 문자열 길이
 - ▶ 실습 6-4 : SELECT절 기본 사용
 - ▶ 실습 6-5 : WHERE절에서의 활용

실습 6-4 선택한 열의 문자열 길이 구하기

```
01 SELECT ENAME, LENGTH(ENAME)
02 FROM EMP;
```

:: 결과 화면

ENAME	LENGTH(ENAME)
▶ SMITH	5
ALLEN	5
WARD	4
JONES	5
MARTIN	6
BLAKE	5
CLARK	5
SCOTT	5
KING	4
TURNER	6
ADAMS	5
JAMES	5
FORD	4
MILLER	6

실습 6-5 사원 이름의 길이가 5 이상인 행 출력하기

```
01 SELECT ENAME, LENGTH(ENAME)
02 FROM EMP
03 WHERE LENGTH(ENAME) >= 5;
```

:: 결과 화면

ENAME	LENGTH(ENAME)
▶ SMITH	5
ALLEN	5
JONES	5
MARTIN	6
BLAKE	5
CLARK	5
SCOTT	5
TURNER	6
ADAMS	5
JAMES	5
MILLER	6

실습 6-6 LENGTH 함수와 LENGTHB 함수 비교하기

```
01 SELECT LENGTH('한글'), LENGTHB('한글')
02 FROM DUAL;
```

:: 결과 화면

LENGTH('한글')	LENGTHB('한글')
▶ 2	4

06-2 문자 데이터를 가공하는 문자 함수

▶ SUBSTR : 문자열 일부 추출

함수	설명
SUBSTR(문자열 데이터, 시작 위치, 추출 길이)	문자열 데이터의 시작 위치부터 추출 길이만큼 추출합니다. 시작 위치가 음수일 경우에는 마지막 위치부터 거슬러 올라간 위치에서 시작합니다.
SUBSTR(문자열 데이터, 시작 위치)	문자열 데이터의 시작 위치부터 문자열 데이터 끝까지 추출합니다. 시작 위치가 음수일 경우에는 마지막 위치부터 거슬러 올라간 위치에서 끝까지 추출합니다.

실습 6-7 SUBSTR 함수를 사용하는 예

```
01 SELECT JOB, SUBSTR(JOB, 1, 2), SUBSTR(JOB, 3, 2), SUBSTR(JOB, 5)
02 FROM EMP;
```

:: 결과 화면

JOB	SUBSTR(JOB,1,2)	SUBSTR(JOB,3,2)	SUBSTR(JOB,5)
CLERK	CL	ER	K
SALESMAN	SA	LE	SMAN
SALESMAN	SA	LE	SMAN
MANAGER	MA	NA	GER
SALESMAN	SA	LE	SMAN
MANAGER	MA	NA	GER
MANAGER	MA	NA	GER
ANALYST	AN	AL	YST
PRESIDENT	PR	ES	IDENT
SALESMAN	SA	LE	SMAN
CLERK	CL	ER	K
CLERK	CL	ER	K
ANALYST	AN	AL	YST
CLERK	CL	ER	K

JOB 값이 SALESMAN일 때를 예로 들어 SUBSTR 함수의 내용을 자세히 살펴봅시다.

• SUBSTR(JOB, 1, 2) 의미



• SUBSTR(JOB, 3, 2) 의미



• SUBSTR(JOB, 5) 의미



06-2 문자 데이터를 가공하는 문자 함수

▶ SUBSTR : 문자열 일부 추출

실습 6-8 SUBSTR 함수 안에 다른 함수(LENGTH) 함께 사용하기

```
01 SELECT JOB,  
02        SUBSTR(JOB, -LENGTH(JOB)),  
03        SUBSTR(JOB, -LENGTH(JOB), 2),  
04        SUBSTR(JOB, -3)  
05 FROM EMP;
```

:: 결과 화면

JOB	SUBSTR(JOB, -LENGTH(JOB))	SUBSTR(JOB, -LENGTH(JOB), 2)	SUBSTR(JOB, -3)
CLERK	CLERK	CL	ERK
SALESMAN	SALESMAN	SA	MAN
SALESMAN	SALESMAN	SA	MAN
MANAGER	MANAGER	MA	GER
SALESMAN	SALESMAN	SA	MAN
MANAGER	MANAGER	MA	GER
MANAGER	MANAGER	MA	GER
ANALYST	ANALYST	AN	YST
PRESIDENT	PRESIDENT	PR	ENT
SALESMAN	SALESMAN	SA	MAN
CLERK	CLERK	CL	ERK
CLERK	CLERK	CL	ERK
ANALYST	ANALYST	AN	YST
CLERK	CLERK	CL	ERK

- SUBSTR(JOB, -LENGTH(JOB))

-5 -4 -3 -2 -1
CLERK

-5자리(CLERK의 -LENGTH(JOB))부터 끝까지 출력

- SUBSTR(JOB, -LENGTH(JOB), 2)

-5 -4 -3 -2 -1
CLERK

-5자리(CLERK의 -LENGTH(JOB))부터
두 글자 출력

- SUBSTR(JOB, -3)

-5 -4 -3 -2 -1
CLERK

-3자리(CLERK의 -LENGTH(JOB))부터 끝까지 출력

06-2 문자 데이터를 가공하는 문자 함수

▶ INSTR : 문자열 데이터 내 특정 문자 위치 찾기

INSTR([대상 문자열 데이터(필수)],
[위치를 찾으려는 부분 문자(필수)],
[위치 찾기를 시작할 대상 문자열 데이터 위치(선택, 기본값은 1)],
[시작 위치에서 찾으려는 문자가 몇 번째인지 지정(선택, 기본값은 1)])

기본 형식

특정 문자 위치 찾기

실습 6-9 INSTR 함수로 문자열 데이터에서 특정 문자열 찾기

```
01 SELECT INSTR('HELLO, ORACLE!', 'L') AS INSTR_1,  
02        INSTR('HELLO, ORACLE!', 'L', 5) AS INSTR_2,  
03        INSTR('HELLO, ORACLE!', 'L', 2, 2) AS INSTR_3  
04 FROM DUAL;
```

:: 결과 화면

INSTR_1	INSTR_2	INSTR_3
3	12	4

- INSTR('HELLO, ORACLE!', 'L') : 시작 위치와 몇 번째 L인지 정해지지 않음

1	2	3	4	5	6	7	8	9	10	11	12	13	14
H	E	L	L	O	,		O	R	A	C	L	E	!

처음부터 검색

- INSTR('HELLO, ORACLE!', 'L', 5) : 다섯 번째 글자 O부터 L을 찾음

1	2	3	4	5	6	7	8	9	10	11	12	13	14
H	E	L	L	O	,		O	R	A	C	L	E	!

여기에서부터 검색

검색 시작 위치부터 첫 번째로 등장한 L

- INSTR('HELLO, ORACLE!', 'L', 2, 2) : 두 번째 글자 E부터 시작해서 두 번째 L을 찾음

1	2	3	4	5	6	7	8	9	10	11	12	13	14
H	E	L	L	O	,		O	R	A	C	L	E	!

여기에서부터 검색

검색 시작 위치부터 두 번째로 등장한 L

06-2 문자 데이터를 가공하는 문자 함수

▶ INSTR : 문자열 데이터 내 특정 문자 위치 찾기

특정 문자를 포함하고 있는 행 찾기

실습 6-10 INSTR 함수로 사원 이름에 문자 S가 있는 행 구하기

```
01  SELECT *  
02    FROM EMP  
03  WHERE INSTR(ENAME, 'S') > 0;
```

실습 6-11 LIKE 연산자로 사원 이름에 문자 S가 있는 행 구하기

```
01  SELECT *  
02    FROM EMP  
03  WHERE ENAME LIKE '%S%'
```

06-2 문자 데이터를 가공하는 문자 함수

▶ REPLACE : 특정 문자를 다른 문자로 대체

REPLACE([문자열 데이터 또는 열 이름(필수)], [찾는 문자(필수)], [대체할 문자(선택)])

기본 형식

실습 6-12 REPLACE 함수로 문자열 안에 있는 특정 문자 바꾸기

```
01 SELECT '010-1234-5678' AS REPLACE_BEFORE,  
02        REPLACE('010-1234-5678', '-', ' ') AS REPLACE_1,  
03        REPLACE('010-1234-5678', '-') AS REPLACE_2  
04 FROM DUAL;
```

:: 결과 화면

REPLACE_BEFORE	REPLACE_1	REPLACE_2
010-1234-5678	010 1234 5678	01012345678

06-2 문자 데이터를 가공하는 문자 함수

▶ LPAD, RPAD : 데이터의 빈 공간 채우기

기본 형식

LPAD([문자열 데이터 또는 열이름(필수)], [데이터의 자릿수(필수)], [빈 공간에 채울 문자(선택)])
RPAD([문자열 데이터 또는 열이름(필수)], [데이터의 자릿수(필수)], [빈 공간에 채울 문자(선택)])

실습 6-13 LPAD, RPAD 함수 사용하여 출력하기

```
01 SELECT 'Oracle',  
02        LPAD('Oracle', 10, '#') AS LPAD_1,  
03        RPAD('Oracle', 10, '*') AS RPAD_1,  
04        LPAD('Oracle', 10) AS LPAD_2,  
05        RPAD('Oracle', 10) AS RPAD_2  
06 FROM DUAL;
```

:: 결과 화면

'ORACLE'	LPAD_1	RPAD_1	LPAD_2	RPAD_2
Oracle	####Oracle	Oracle****	Oracle	Oracle

실습 6-14 RPAD 함수를 사용하여 개인정보 뒷자리 * 표시로 출력하기

```
01 SELECT  
02        RPAD('971225-', 14, '*') AS RPAD_JMNO,  
03        RPAD('010-1234-', 13, '*') AS RPAD_PHONE  
04 FROM DUAL;
```

:: 결과 화면

RPAD_JMNO	RPAD_PHONE
971225-*****	010-1234-****

06-2 문자 데이터를 가공하는 문자 함수


▶ CONCAT : 두 문자열 데이터를 합치기

실습 6-15 두 열 사이에 콜론(:) 넣고 연결하기

```
01 SELECT CONCAT(EMPNO, ENAME),  
02        CONCAT(EMPNO, CONCAT(' : ', ENAME))  
03 FROM EMP  
04 WHERE ENAME = 'SCOTT';
```

:: 결과 화면

CONCAT(EMPNO,ENAME)	CONCAT(EMPNO,CONCAT(':',ENAME))
7788SCOTT	7788 : SCOTT

 **한발 더 나가!!** 문자열 데이터를 연결하는 || 연산자

|| 연산자는 CONCAT 함수와 유사하게 열이나 문자열을 연결합니다. 실습 6-15에 사용된 CONCAT 함수 대신 || 연산자를 사용하면 다음과 같습니다.

```
SELECT EMPNO || ENAME,  
       EMPNO || ' : ' || ENAME  
FROM ...
```

06-2 문자 데이터를 가공하는 문자 함수

▶ TRIM, LTRIM, RTRIM : 특정 문자 지우기

TRIM([삭제 옵션(선택)] [삭제할 문자(선택)] FROM [원본 문자열 데이터(필수)])

기본 형식

실습 6-16 TRIM 함수로 공백 제거하여 출력하기

```
01 SELECT '[' || TRIM(' _Oracle_ ') || ']' AS TRIM,  
02      '[' || TRIM(LEADING FROM ' _Oracle_ ') || ']' AS TRIM_LEADING,  
03      '[' || TRIM(TRAILING FROM ' _Oracle_ ') || ']' AS TRIM_TRAILING,  
04      '[' || TRIM(BOTH FROM ' _Oracle_ ') || ']' AS TRIM_BOTH  
05 FROM DUAL;
```

:: 결과 화면

TRIM	TRIM_LEADING	TRIM_TRAILING	TRIM_BOTH
[_Oracle_]	[_Oracle_]	[_Oracle_]	[_Oracle_]

실습 6-17 TRIM 함수로 삭제할 문자 _ 삭제 후 출력하기

```
01 SELECT '[' || TRIM('_' FROM ' _Oracle_ ') || ']' AS TRIM,  
02      '[' || TRIM(LEADING '_' FROM ' _Oracle_ ') || ']' AS TRIM_LEADING,  
03      '[' || TRIM(TRAILING '_' FROM ' _Oracle_ ') || ']' AS TRIM_TRAILING,  
04      '[' || TRIM(BOTH '_' FROM ' _Oracle_ ') || ']' AS TRIM_BOTH  
05 FROM DUAL;
```

:: 결과 화면

TRIM	TRIM_LEADING	TRIM_TRAILING	TRIM_BOTH
[_Oracle_]	[_Oracle_]	[_Oracle_]	[_Oracle_]

06-2 문자 데이터를 가공하는 문자 함수

▶ TRIM, LTRIM, RTRIM : 특정 문자 지우기

LTRIM([원본 문자열 데이터(필수)], [삭제할 문자 집합(선택)]) - ❶

기본 형식

RTRIM([원본 문자열 데이터(필수)], [삭제할 문자 집합(선택)]) - ❷

번호	설명
❶	원본 문자열의 왼쪽에서 삭제할 문자열을 지정합니다(삭제할 문자열을 지정하지 않으면 공백이 삭제됨).
❷	원본 문자열의 오른쪽에서 삭제할 문자열을 지정합니다(삭제할 문자열을 지정하지 않으면 공백이 삭제됨).

실습 6-18 TRIM, LTRIM, RTRIM 사용하여 문자열 출력하기

```
01 SELECT '[' || TRIM(' _Oracle_ ') || ']' AS TRIM,  
02      '[' || LTRIM(' _Oracle_ ') || ']' AS LTRIM,  
03      '[' || LTRIM('<_Oracle_>', '<_') || ']' AS LTRIM_2,  
04      '[' || RTRIM(' _Oracle_ ') || ']' AS RTRIM,  
05      '[' || RTRIM('<_Oracle_>', '>_') || ']' AS RTRIM_2  
06 FROM DUAL;
```

:: 결과 화면

TRIM	LTRIM	LTRIM_2	RTRIM	RTRIM_2
[_Oracle_]	[_Oracle_]	[Oracle_>]	[_Oracle_]	[<_Oracle]

06-3 숫자 데이터를 연산하고 수치를 조정하는 숫자함수

▶ ROUND : 반올림

ROUND([숫자(필수)], [반올림 위치(선택)]) - 1

기본 형식

실습 6-19 ROUND 함수를 사용하여 반올림된 숫자 출력하기

```
01 SELECT ROUND(1234.5678) AS ROUND,  
02        ROUND(1234.5678, 0) AS ROUND_0,  
03        ROUND(1234.5678, 1) AS ROUND_1,  
04        ROUND(1234.5678, 2) AS ROUND_2,  
05        ROUND(1234.5678, -1) AS ROUND_MINUS1,  
06        ROUND(1234.5678, -2) AS ROUND_MINUS2  
07 FROM DUAL;
```

:: 결과 화면

	ROUND	ROUND_0	ROUND_1	ROUND_2	ROUND_MINUS1	ROUND_MINUS2
▶	1235	1235	1234.6	1234.57	1230	1200

06-3 숫자 데이터를 연산하고 수치를 조성하는 숫자함수

▶ TRUNC : 버림

TRUNC([숫자(필수)], [버림 위치(선택)]) — ❶

기본 형식

실습 6-20 TRUNC 함수를 사용하여 숫자 출력하기

```
01 SELECT TRUNC(1234.5678) AS TRUNC,  
02        TRUNC(1234.5678, 0) AS TRUNC_0,  
03        TRUNC(1234.5678, 1) AS TRUNC_1,  
04        TRUNC(1234.5678, 2) AS TRUNC_2,  
05        TRUNC(1234.5678, -1) AS TRUNC_MINUS1,  
06        TRUNC(1234.5678, -2) AS TRUNC_MINUS2  
07 FROM DUAL;
```

:: 결과 화면

TRUNC	TRUNC_0	TRUNC_1	TRUNC_2	TRUNC_MINUS1	TRUNC_MINUS2
1234	1234	1234.5	1234.56	1230	1200

06-3 숫자 데이터를 연산하고 수치를 조정하는 숫자함수

- ▶ **CEIL** : 지정된 숫자와 가장 가까운 큰 정수
- ▶ **FLOOR** : 지정된 숫자와 가장 가까운 작은 정수

CEIL([숫자(필수)])
FLOOR([숫자(필수)])

기본 형식

실습 6-21 CEIL, FLOOR 함수로 숫자 출력하기

```
01 SELECT CEIL(3.14),  
02         FLOOR(3.14),  
03         CEIL(-3.14),  
04         FLOOR(-3.14)  
05 FROM DUAL;
```

:: 결과 화면

	CEIL(3.14)	FLOOR(3.14)	CEIL(-3.14)	FLOOR(-3.14)
▶	4	3	-3	-4

06-3 숫자 데이터를 연산하고 수치를 조성하는 숫자함수

▶ MOD : 숫자를 나눈 나머지

MOD([나눠셈 될 숫자(필수)], [나눌 숫자(필수)]) - ①

기본 형식

실습 6-22 MOD 함수를 사용하여 나머지 값 출력하기

```
01 SELECT MOD(15, 6),  
02        MOD(10, 2),  
03        MOD(11, 2)  
04 FROM DUAL;
```

:: 결과 화면

	MOD(15,6)	MOD(10,2)	MOD(11,2)
▶	3	0	1

06-4 날짜 데이터를 다루는 날짜 함수

▶ 오라클의 DATE 데이터의 연산

연산	설명
날짜 데이터 + 숫자	날짜 데이터보다 숫자만큼 일수 이후의 날짜
날짜 데이터 - 숫자	날짜 데이터보다 숫자만큼 일수 이전의 날짜
날짜 데이터 - 날짜 데이터	두 날짜 데이터 간의 일수 차이
날짜 데이터 + 날짜 데이터	연산 불가, 지원하지 않음★

06-4 날짜 데이터를 다루는 날짜 함수

▶ **SYSDATE** : 현재 날짜와 시간

실습 6-23 SYSDATE 함수를 사용하여 날짜 출력하기

```
01 SELECT SYSDATE AS NOW,  
02     < SYSDATE-1 AS YESTERDAY,  
03     < SYSDATE+1 AS TOMORROW  
04 FROM DUAL;
```

:: 결과 화면

NOW	YESTERDAY	TOMORROW
▶ 2018-07-13 오후 11:47:43	2018-07-12 오후 11:47:43	2018-07-14 오후 11:47:43

06-4 날짜 데이터를 다루는 날짜 함수

▶ ADD_MONTHS : 몇 개월 이후 날짜

ADD_MONTHS([날짜 데이터(필수)], [더할 개월 수(정수)(필수)]) - ①

기본 형식

실습 6-24 SYSDATE와 ADD_MONTHS 함수로 3개월 후 날짜 구하기

```
01 SELECT SYSDATE,  
02        ADD_MONTHS(SYSDATE, 3)  
03 FROM DUAL;
```

:: 결과 화면

SYSDATE	ADD_MONTHS(SYSDATE,3)
2018-07-13 오후 11:49:32	2018-10-13 오후 11:49:32

실습 6-26 입사 32년 미만인 직원 데이터 출력하기

```
01 SELECT EMPNO,  
02        ENAME, HIREDATE, SYSDATE  
03 FROM EMP  
04 WHERE ADD_MONTHS(HIREDATE, 384) > SYSDATE;
```

:: 결과 화면

EMPNO	ENAME	HIREDATE	SYSDATE
7788	SCOTT	1987-04-19	2018-07-13 오후 11:51:34
7876	ADAMS	1987-05-23	2018-07-13 오후 11:51:34

실습 6-25 입사 10주년이 되는 직원들 데이터 출력하기

```
01 SELECT EMPNO, ENAME, HIREDATE,  
02        ADD_MONTHS(HIREDATE, 120) AS WORK10YEAR  
03 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	HIREDATE	WORK10YEAR
7369	SMITH	1980/12/17	1990/12/17
7499	ALLEN	1981/02/20	1991/02/20
7521	WARD	1981/02/22	1991/02/22
7566	JONES	1981/04/02	1991/04/02
7654	MARTIN	1981/09/28	1991/09/28

06-4 날짜 데이터를 다루는 날짜 함수

▶ MONTHS_BETWEEN : 두 날짜 간의 개월수 차이

실습 6-27 HIREDATE와 SYSDATE 사이의 개월 수를 MONTHS_BETWEEN 함수로 출력하기

```
01 SELECT EMPNO, ENAME, HIREDATE, SYSDATE,  
02        MONTHS_BETWEEN(HIREDATE, SYSDATE) AS MONTHS1,  
03        MONTHS_BETWEEN(SYSDATE, HIREDATE) AS MONTHS2,  
04        TRUNC(MONTHS_BETWEEN(SYSDATE, HIREDATE)) AS MONTHS3  
05 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	HIREDATE	SYSDATE	MONTHS1	MONTHS2	MONTHS3
7369	SMITH	1980-12-17	2018-07-13 오후 11:54:07	-450.90309401135	450.90309401135	450
7499	ALLEN	1981-02-20	2018-07-13 오후 11:54:07	-448.806319817802	448.806319817802	448
7521	WARD	1981-02-22	2018-07-13 오후 11:54:07	-448.741803688769	448.741803688769	448
7566	JONES	1981-04-02	2018-07-13 오후 11:54:07	-447.386964979092	447.386964979092	447
7654	MARTIN	1981-09-28	2018-07-13 오후 11:54:07	-441.548255301673	441.548255301673	441
7698	BLAKE	1981-05-01	2018-07-13 오후 11:54:07	-446.419223043608	446.419223043608	446
7782	CLARK	1981-06-09	2018-07-13 오후 11:54:07	-445.161158527479	445.161158527479	445
7788	SCOTT	1987-04-19	2018-07-13 오후 11:54:07	-374.838577882318	374.838577882318	374
7839	KING	1981-11-17	2018-07-13 오후 11:54:07	-439.90309401135	439.90309401135	439
7844	TURNER	1981-09-08	2018-07-13 오후 11:54:07	-442.193416591995	442.193416591995	442
7876	ADAMS	1987-05-23	2018-07-13 오후 11:54:07	-373.709545624253	373.709545624253	373
7900	JAMES	1981-12-03	2018-07-13 오후 11:54:07	-439.354706914576	439.354706914576	439
7902	FORD	1981-12-03	2018-07-13 오후 11:54:07	-439.354706914576	439.354706914576	439

06-4 날짜 데이터를 다루는 날짜 함수

▶ NEXT_DAY : 돌아오는요일

NEXT_DAY([날짜 데이터(필수)], [요일 문자(필수)]) - ❶

기본 형식

▶ LAST_DAY : 달의 마지막 날짜

LAST_DAY([날짜 데이터(필수)]) - ❶

기본 형식

실습 6-28 NEXT_DAY, LAST_DAY 함수를 사용하여 출력하기

```
01 SELECT SYSDATE,  
02        NEXT_DAY(SYSDATE, '월요일'),  
03        LAST_DAY(SYSDATE)  
04 FROM DUAL;
```

:: 결과 화면

SYSDATE	NEXT_DAY(SYSDATE, '월요일')	LAST_DAY(SYSDATE)
▶ 2018-07-13 오후 11:55:11	2018-07-16 오후 11:55:11	2018-07-31 오후 11:55:11

06-4 날짜 데이터를 다루는 날짜 함수

▶ ROUND, TRUNC : 날짜 반올림, 버림

입력 데이터 종류	사용 방식
숫자 데이터	ROUND([숫자(필수)], [반올림 위치])
	TRUNC([숫자(필수)], [버림 위치])
날짜 데이터	ROUND([날짜데이터(필수)], [반올림 기준 포맷])
	TRUNC([날짜데이터(필수)], [버림 기준 포맷])

포맷 모델	기준 단위
CC, SCC	네 자리 연도의 끝 두 자리를 기준으로 사용 (2016년이면 2050 이하이므로, 반올림할 경우 2001년으로 처리)
YYYY, YYYYY, YEAR, SYEAR, YYY, YY, Y	날짜 데이터의 해당 연·월·일의 7월 1일을 기준 (2016년 7월 1일 일 경우, 2017년으로 처리)
IYYY, IYY, IY, I	ISO 8601에서 제정한 날짜 기준년도 포맷을 기준
Q	각 분기의 두 번째 달의 16일 기준
MONTH, MON, MM, RM	각 달의 16일 기준
WW	해당 연도의 몇 주(1~53번째 주)를 기준
IW	ISO 8601에서 제정한 날짜 기준 해당 연도의 주(week)를 기준
W	해당 월의 주(1~5번째 주)를 기준
DDD, DD, J	해당 일의 청오(12:00:00)를 기준
DAY, DY, D	한 주가 시작되는 날짜를 기준
HH, HH12, HH24	해당일의 시간을 기준
MI	해당일 시간의 분을 기준

🔗 날짜 데이터 기준에 대해 조금 더 자세한 내용을 알고 싶다면 ISO 공식 웹사이트(iso.org/iso/home/standards/iso8601.htm)를 참조하세요.

실습 6-29 ROUND 함수 사용하여 날짜 데이터 출력하기

```
01 SELECT SYSDATE,  
02         ROUND(SYSDATE, 'CC') AS FORMAT_CC,  
03         ROUND(SYSDATE, 'YYYY') AS FORMAT_YYYY,  
04         ROUND(SYSDATE, 'Q') AS FORMAT_Q,  
05         ROUND(SYSDATE, 'DDD') AS FORMAT_DDD,  
06         ROUND(SYSDATE, 'HH') AS FORMAT_HH  
07 FROM DUAL;
```

:: 결과 화면

SYSDATE	FORMAT_CC	FORMAT_YYYY	FORMAT_Q	FORMAT_DDD	FORMAT_HH
2018-07-13 오후 11:56:28	2001-01-01	2018-07-13	2018-07-01	2018-07-14	2018-07-14

실습 6-30 TRUNC 함수 사용하여 날짜 데이터 출력하기

```
01 SELECT SYSDATE,  
02         TRUNC(SYSDATE, 'CC') AS FORMAT_CC,  
03         TRUNC(SYSDATE, 'YYYY') AS FORMAT_YYYY,  
04         TRUNC(SYSDATE, 'Q') AS FORMAT_Q,  
05         TRUNC(SYSDATE, 'DDD') AS FORMAT_DDD,  
06         TRUNC(SYSDATE, 'HH') AS FORMAT_HH  
07 FROM DUAL;
```

:: 결과 화면

SYSDATE	FORMAT_CC	FORMAT_YYYY	FORMAT_Q	FORMAT_DDD	FORMAT_HH
2018-07-13 오후 11:57:27	2001-01-01	2018-07-13	2018-07-01	2018-07-13	2018-07-13 오후 11:00:00

06-5 자료형을 반환하는 형 변환 함수

- ▶ TO_CHAR : 숫자 또는 날짜 데이터를 문자 데이터로
- ▶ TO_NUMBER : 문자 데이터를 숫자 데이터로
- ▶ TO_DATE : 문자 데이터를 날짜 데이터로

실습 6-31 숫자와 문자열(숫자)을 더하여 출력하기

```
01 SELECT EMPNO, ENAME, EMPNO + '500'  
02 FROM EMP  
03 WHERE ENAME = 'SCOTT';
```

:: 결과 화면

EMPNO	ENAME	EMPNO+'500'
7788	SCOTT	8288

실습 6-32 문자열(문자)과 숫자를 더하여 출력하기

```
01 SELECT 'ABCD' + EMPNO, EMPNO  
02 FROM EMP  
03 WHERE ENAME = 'SCOTT';
```

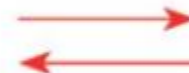
:: 결과 화면

Errors
[Error] Execution (1: 8): ORA-01722: 수치가 부적합합니다

숫자 데이터
(NUMBER)



문자 데이터
(CHARACTER)



날짜 데이터
(DATE)

06-5 자료형을 반환하는 형 변환 함수

▶ TO_CHAR : 숫자 또는 날짜 데이터를 문자 데이터로

TO_CHAR([날짜데이터(필수)], '[출력되길 원하는 문자 형태(필수)]') — ①

기본 형식

실습 6-33 SYSDATE 날짜 형식 지정하여 출력하기

```
01 SELECT TO_CHAR(SYSDATE, 'YYYY/MM/DD HH24:MI:SS') AS 현재날짜시간
02 FROM DUAL;
```

:: 결과 화면

현재날짜시간
▶ 2018/07/13 23:59:01

실습 6-34 월과 요일을 다양한 형식으로 출력하기

```
01 SELECT SYSDATE,
02         TO_CHAR(SYSDATE, 'MM') AS MM,
03         TO_CHAR(SYSDATE, 'MON') AS MON,
04         TO_CHAR(SYSDATE, 'MONTH') AS MONTH,
05         TO_CHAR(SYSDATE, 'DD') AS DD,
06         TO_CHAR(SYSDATE, 'DY') AS DY,
07         TO_CHAR(SYSDATE, 'DAY') AS DAY
08 FROM DUAL;
```

:: 결과 화면

SYSDATE	MM	MON	MONTH	DD	DY	DAY
▶ 2018-07-14 오전 12:00:13	07	7월	7월	14	토	토요일

형식	설명
CC	세기
YYYY, RRRR	연(4자리 숫자)
YY, RR	연(2자리 숫자)
MM	월(2자리 숫자)
MON	월(언어별 월 이름 약자)
MONTH	월(언어별 월 이름 전체)
DD	일(2자리 숫자)
DDD	1년 중 며칠 (1~366)
DY	요일(언어별 요일 이름 약자)
DAY	요일(언어별 요일 이름 전체)
W	1년 중 몇 번째 주 (1~53)

06-5 자료형을 반환하는 형 변환 함수

▶ TO_CHAR : 숫자 또는 날짜 데이터를 문자 데이터로

TO_CHAR([날짜데이터(필수)], '[출력되길 원하는 문자 형태(필수)]') — ①

기본 형식

실습 6-35 여러 언어로 날짜(월) 출력하기

```
01 SELECT SYSDATE,
02        TO_CHAR(SYSDATE, 'MM') AS MM,
03        TO_CHAR(SYSDATE, 'MON', 'NLS_DATE_LANGUAGE = KOREAN' ) AS MON_KOR,
04        TO_CHAR(SYSDATE, 'MON', 'NLS_DATE_LANGUAGE = JAPANESE') AS MON_JPN,
05        TO_CHAR(SYSDATE, 'MON', 'NLS_DATE_LANGUAGE = ENGLISH' ) AS MON_ENG,
06        TO_CHAR(SYSDATE, 'MONTH', 'NLS_DATE_LANGUAGE = KOREAN' ) AS MONTH_KOR,
07        TO_CHAR(SYSDATE, 'MONTH', 'NLS_DATE_LANGUAGE = JAPANESE') AS MONTH_JPN,
08        TO_CHAR(SYSDATE, 'MONTH', 'NLS_DATE_LANGUAGE = ENGLISH' ) AS MONTH_ENG
09 FROM DUAL;
```

:: 결과 화면

SYSDATE	MM	MON_KOR	MON_JPN	MON_ENG	MONTH_KOR	MONTH...	MONTH_ENG
▶ 2018-07-14 오전 12:01:24	07	7월	7月	JUL	7월	7月	JULY

실습 6-36 여러 언어로 날짜(요일) 출력하기

```
01 SELECT SYSDATE,
02        TO_CHAR(SYSDATE, 'MM') AS MM,
03        TO_CHAR(SYSDATE, 'DD') AS DD,
04        TO_CHAR(SYSDATE, 'DY', 'NLS_DATE_LANGUAGE = KOREAN' ) AS DY_KOR,
05        TO_CHAR(SYSDATE, 'DY', 'NLS_DATE_LANGUAGE = JAPANESE') AS DY_JPN,
06        TO_CHAR(SYSDATE, 'DY', 'NLS_DATE_LANGUAGE = ENGLISH' ) AS DY_ENG,
07        TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE = KOREAN' ) AS DAY_KOR,
08        TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE = JAPANESE') AS DAY_JPN,
09        TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE = ENGLISH' ) AS DAY_ENG
10 FROM DUAL;
```

:: 결과 화면

SYSDATE	MM	DD	DY_KOR	DY_JPN	DY_ENG	DAY_KOR	DAY_JPN	DAY_ENG
▶ 2018-07-14 오전 12:02:33	07	14	토	土	SAT	토요일	土曜日	SATURDAY

06-5 자료형을 반환하는 형 변환 함수

▶ TO_CHAR : 숫자 또는 날짜 데이터를 문자 데이터로

TO_CHAR([날짜데이터(필수)], '[출력되길 원하는 문자 형태(필수)]') — ①

기본 형식

형식	설명
HH24	24시간으로 표현한 시간
HH, HH12	12시간으로 표현한 시간
MI	분
SS	초
AM, PM, A.M., P.M.	오전, 오후 표시

실습 6-37 SYSDATE 시간 형식 지정하여 출력하기

```
01 SELECT SYSDATE,  
02        TO_CHAR(SYSDATE, 'HH24:MI:SS') AS HH24MISS,  
03        TO_CHAR(SYSDATE, 'HH12:MI:SS AM') AS HHMISS_AM,  
04        TO_CHAR(SYSDATE, 'HH:MI:SS P.M.') AS HHMISS_PM  
05 FROM DUAL;
```

:: 결과 화면

SYSDATE	HH24MISS	HHMISS_AM	HHMISS_PM
▶ 2018-07-14 오전 12:03:39	00:03:39	12:03:39 오전	12:03:39 오전

06-5 자료형을 반환하는 형 변환 함수

▶ TO_CHAR : 숫자 또는 날짜 데이터를 문자 데이터로

TO_CHAR([날짜데이터(필수)], '[출력되길 원하는 문자 형태(필수)]') — ①

기본 형식

형식	설명
9	숫자의 한 자리를 의미함(빈 자리를 채우지 않음)
0	빈 자리를 0으로 채움을 의미함
\$	달러(\$) 표시를 붙여서 출력함
L	L(Locale) 지역 화폐 단위 기호를 붙여서 출력함
.	소수점을 표시함
,	천 단위의 구분 기호를 표시함

실습 6-38 여러 가지 숫자 형식을 사용하여 급여 출력하기

```
01 SELECT SAL,  
02     TO_CHAR(SAL, '$999,999') AS SAL_$,  
03     TO_CHAR(SAL, 'L999,999') AS SAL_L,  
04     TO_CHAR(SAL, '999,999.00') AS SAL_1,  
05     TO_CHAR(SAL, '000,999,999.00') AS SAL_2,  
06     TO_CHAR(SAL, '000999999.99') AS SAL_3,  
07     TO_CHAR(SAL, '999,999,00') AS SAL_4  
08 FROM EMP;
```

:: 결과 화면

SAL	SAL_\$	SAL_L	SAL_1	SAL_2	SAL_3	SAL_4
800	\$800	₩800	800.00	000,000,800.00	000000800.00	8,00
1600	\$1,600	₩1,600	1,600.00	000,001,600.00	000001600.00	16,00
1250	\$1,250	₩1,250	1,250.00	000,001,250.00	000001250.00	12,50
2975	\$2,975	₩2,975	2,975.00	000,002,975.00	000002975.00	29,75
1250	\$1,250	₩1,250	1,250.00	000,001,250.00	000001250.00	12,50

06-5 자료형을 반환하는 형 변환 함수

▶ TO_NUMBER : 문자 데이터를 숫자 데이터로

TO_NUMBER('[문자열 데이터(필수)]', '[인식될 숫자형태(필수)]') - 1

기본 형식

▶ 암시적 형변환

실습 6-39 문자 데이터와 숫자 데이터를 연산하여 출력하기

```
01 SELECT 1300 - '1500',  
02        '1300' + 1500  
03 FROM DUAL;
```

:: 결과 화면

	1300-'1500'	'1300'+1500
▶	-200	2800

▶ 문자 연산(로 에러)

실습 6-40 문자 데이터끼리 연산하여 출력하기

```
01 SELECT '1,300' - '1,500'  
02 FROM DUAL;
```

:: 결과 화면

Errors
[Error] Execution (S: 8): ORA-01722: 수치가 부적합합니다

실습 6-41 TO_NUMBER 함수로 연산하여 출력하기

```
01 SELECT TO_NUMBER('1,300', '999,999') - TO_NUMBER('1,500', '999,999')  
02 FROM DUAL;
```

:: 결과 화면

	TO_NUMBER('1,300','999,999')-TO_NUMBER('1,500','999,999')
▶	-200

06-5 자료형을 반환하는 형 변환 함수

▶ TO_DATE : 문자 데이터를 날짜 데이터로

TO_DATE(' [문자열 데이터(필수)] ', ' [인식될 날짜형태(필수)] ') — ①

기본 형식

실습 6-42 TO_DATE 함수로 문자 데이터를 날짜 데이터 변환하기

```
01 SELECT TO_DATE('2018-07-14', 'YYYY-MM-DD') AS TODATE1,  
02        TO_DATE('20180714', 'YYYY-MM-DD') AS TODATE2  
03 FROM DUAL;
```

:: 결과 화면

TODATE1	TODATE2
2018/07/14	2018/07/14

실습 6-44 여러 가지 형식으로 날짜 데이터 출력하기

```
01 SELECT TO_DATE('49/12/10', 'YY/MM/DD') AS YY_YEAR_49,  
02        TO_DATE('49/12/10', 'RR/MM/DD') AS RR_YEAR_49,  
03        TO_DATE('50/12/10', 'YY/MM/DD') AS YY_YEAR_50,  
04        TO_DATE('50/12/10', 'RR/MM/DD') AS RR_YEAR_50,  
05        TO_DATE('51/12/10', 'YY/MM/DD') AS YY_YEAR_51,  
06        TO_DATE('51/12/10', 'RR/MM/DD') AS RR_YEAR_51  
07 FROM DUAL;
```

:: 결과 화면

YY_YEAR_49	RR_YEAR_49	YY_YEAR_50	RR_YEAR_50	YY_YEAR_51	RR_YEAR_51
2049/12/10	2049/12/10	2050/12/10	1950/12/10	2051/12/10	1951/12/10

실습 6-43 1981년 6월 1일 이후에 입사한 사원 정보 출력하기

```
01 SELECT *  
02 FROM EMP  
03 WHERE HIREDATE > TO_DATE('1981/06/01', 'YYYY/MM/DD');
```

:: 결과 화면

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
7782	CLARK	MANAGER	7839	1981/06/09	2450		10
7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
7839	KING	PRESIDENT		1981/11/17	5000		10
7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
7876	ADAMS	CLERK	7788	1987/05/23	1100		20
7900	JAMES	CLERK	7698	1981/12/03	950		30
7902	FORD	ANALYST	7566	1981/12/03	3000		20
7934	MILLER	CLERK	7782	1982/01/23	1300		10

06-6 NULL 처리 함수

- ▶ **NVL** : NULL이 아니면 그대로, NULL이면 지정한 값
- ▶ **NVL2** : NULL이 아닐때와 NULL일때 각각 지정한 값



06-6 NULL 처리 함수

▶ NVL : NULL이 아니면 그대로, NULL이면 지정한 값

NVL([NULL인지 여부를 검사할 데이터 또는 열(필수)],
[앞의 데이터가 NULL일 경우 반환할 데이터](필수)) - ①

기본 형식

번호	설명
①	열 또는 데이터를 입력하여 해당 데이터가 NULL이 아닐 경우 데이터를 그대로 반환하고, NULL인 경우 지정한 데이터를 반환합니다.

실습 6-45 NVL 함수를 사용하여 출력하기

```
01 SELECT EMPNO, ENAME, SAL, COMM, SAL+COMM,  
02        NVL(COMM, 0),  
03        SAL+NVL(COMM, 0)  
04 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	SAL	COMM	SAL+COMM	NVL(COMM,0)	SAL+NVL(COMM,0)
7369	SMITH	800			0	800
7499	ALLEN	1600	300	1900	300	1900
7521	WARD	1250	500	1750	500	1750
7566	JONES	2975			0	2975
7654	MARTIN	1250	1400	2650	1400	2650
7698	BLAKE	2850			0	2850
7782	CLARK	2450			0	2450
7788	SCOTT	3000			0	3000
7839	KING	5000			0	5000
7844	TURNER	1500	0	1500	0	1500
7876	ADAMS	1100			0	1100
7900	JAMES	950			0	950
7902	FORD	3000			0	3000
7934	MILLER	1300			0	1300

06-6 NULL 처리 함수

▶ NVL2 : NULL이 아닐때와 NULL일때 각각 지정한 값

NVL2([NULL인지 여부를 검사할 데이터 또는 열(필수)],
[앞 데이터가 NULL이 아닐 경우 반환할 데이터 또는 계산식(필수)],
[앞 데이터가 NULL일 경우 반환할 데이터 또는 계산식(필수)])

기본 형식

①

번호	설명
①	열 또는 데이터를 입력하여 해당 데이터가 NULL이 아닐 때와 NULL일 때 출력 데이터를 각각 지정합니다.

실습 6-46 NVL2 함수를 사용하여 출력하기

```
01 SELECT EMPNO, ENAME, COMM,  
02         NVL2(COMM, '0', 'X'),  
03         NVL2(COMM, SAL*12+COMM, SAL*12) AS ANNSAL  
04 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	COMM	NVL2(COMM,'0','X')	ANNSAL
7369	SMITH		X	9600
7499	ALLEN	300	O	19500
7521	WARD	500	O	15500
7566	JONES		X	35700
7654	MARTIN	1400	O	16400
7698	BLAKE		X	34200
7782	CLARK		X	29400
7788	SCOTT		X	36000
7839	KING		X	60000
7844	TURNER	0	O	18000
7876	ADAMS		X	13200
7900	JAMES		X	11400
7902	FORD		X	36000
7934	MILLER		X	15600

06-7 상황에 따라 다른 데이터를 반환하는 DECODE 함수와 CASE문

▶ DECODE

- ▶ 기준 데이터를 지정
- ▶ 기준 데이터에 따라 반환할 데이터 지정

▶ CASE

- ▶ 기준 데이터를 지정하는 방식
- ▶ 기준 데이터를 지정하지 않는 방식



06-7 상황에 따라 다른 데이터를 반환하는 DECODE 함수와 CASE문

▶ DECODE

- ▶ 기준 데이터를 지정
- ▶ 기준 데이터에 따라 반환할 데이터 지정

DECODE([검사 대상이 될 열 또는 데이터, 연산이나 함수의 결과],
[조건1], [데이터가 조건1과 일치할 때 반환할 결과],
[조건2], [데이터가 조건2와 일치할 때 반환할 결과],
...
[조건n], [데이터가 조건n과 일치할 때 반환할 결과],
[위 조건1~조건n과 일치한 경우가 없을 때 반환할 결과])

기본 형식

:: 결과 화면

EMPNO	ENAME	JOB	SAL	UPSAL
7369	SMITH	CLERK	800	824
7499	ALLEN	SALESMAN	1600	1680
7521	WARD	SALESMAN	1250	1312.5
7566	JONES	MANAGER	2975	3272.5
7654	MARTIN	SALESMAN	1250	1312.5
7698	BLAKE	MANAGER	2850	3135
7782	CLARK	MANAGER	2450	2695
7788	SCOTT	ANALYST	3000	3000
7839	KING	PRESIDENT	5000	5150
7844	TURNER	SALESMAN	1500	1575
7876	ADAMS	CLERK	1100	1133
7900	JAMES	CLERK	950	978.5
7902	FORD	ANALYST	3000	3000
7934	MILLER	CLERK	1300	1339

실습 6-47 DECODE 함수를 사용하여 출력하기

```
01 SELECT EMPNO, ENAME, JOB, SAL,  
02        DECODE(JOB,  
03                'MANAGER' , SAL*1.1,  
04                'SALESMAN' , SAL*1.05,  
05                'ANALYST' , SAL,  
06                SAL*1.03) AS UPSAL  
07 FROM EMP;
```

06-7 상황에 따라 다른 데이터를 반환하는 DECODE 함수와 CASE문

▶ CASE

- ▶ 기준 데이터를 지정하는 방식
- ▶ 기준 데이터를 지정하지 않는 방식

CASE [검사 대상이 될 열 또는 데이터, 연산이나 함수의 결과(선택)]
WHEN [조건1] THEN [조건1의 결과 값이 true일 때, 반환할 결과]
WHEN [조건2] THEN [조건2의 결과 값이 true일 때, 반환할 결과]
...
WHEN [조건n] THEN [조건n의 결과 값이 true일 때, 반환할 결과]
ELSE [위 조건1~조건n과 일치하는 경우가 없을 때 반환할 결과]
END

기본 형식

실습 6-48 CASE문을 사용하여 출력하기

```
01 SELECT EMPNO, ENAME, JOB, SAL,  
02     CASE JOB  
03         WHEN 'MANAGER' THEN SAL*1.1  
04         WHEN 'SALESMAN' THEN SAL*1.05  
05         WHEN 'ANALYST' THEN SAL  
06         ELSE SAL*1.03  
07     END AS UPSAL  
08 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	JOB	SAL	UPSAL
7369	SMITH	CLERK	800	824
7499	ALLEN	SALESMAN	1600	1680
7521	WARD	SALESMAN	1250	1312.5
7566	JONES	MANAGER	2975	3272.5
7654	MARTIN	SALESMAN	1250	1312.5
7698	BLAKE	MANAGER	2850	3135
7782	CLARK	MANAGER	2450	2695
7788	SCOTT	ANALYST	3000	3000
7839	KING	PRESIDENT	5000	5150
7844	TURNER	SALESMAN	1500	1575
7876	ADAMS	CLERK	1100	1133
7900	JAMES	CLERK	950	978.5
7902	FORD	ANALYST	3000	3000
7934	MILLER	CLERK	1300	1339

06-7 상황에 따라 다른 데이터를 반환하는 DECODE 함수와 CASE문

▶ CASE

실습 6-48 CASE문을 사용하여 출력하기

```
01 SELECT EMPNO, ENAME, JOB, SAL,  
02     CASE JOB  
03         WHEN 'MANAGER' THEN SAL*1.1  
04         WHEN 'SALESMAN' THEN SAL*1.05  
05         WHEN 'ANALYST' THEN SAL  
06         ELSE SAL*1.03  
07     END AS UPSAL  
08 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	JOB	SAL	UPSAL
7369	SMITH	CLERK	800	824
7499	ALLEN	SALESMAN	1600	1680
7521	WARD	SALESMAN	1250	1312.5
7566	JONES	MANAGER	2975	3272.5
7654	MARTIN	SALESMAN	1250	1312.5
7698	BLAKE	MANAGER	2850	3135
7782	CLARK	MANAGER	2450	2695
7788	SCOTT	ANALYST	3000	3000
7839	KING	PRESIDENT	5000	5150
7844	TURNER	SALESMAN	1500	1575
7876	ADAMS	CLERK	1100	1133
7900	JAMES	CLERK	950	978.5
7902	FORD	ANALYST	3000	3000
7934	MILLER	CLERK	1300	1339

실습 6-49 열 값에 따라서 출력 값이 달라지는 CASE문

```
01 SELECT EMPNO, ENAME, COMM,  
02     CASE  
03         WHEN COMM IS NULL THEN '해당사항 없음'  
04         WHEN COMM = 0 THEN '수당없음'  
05         WHEN COMM > 0 THEN '수당 : ' || COMM  
06     END AS COMM_TEXT  
07 FROM EMP;
```

:: 결과 화면

EMPNO	ENAME	COMM	COMM_TEXT
7369	SMITH		해당사항 없음
7499	ALLEN	300	수당 : 300
7521	WARD	500	수당 : 500
7566	JONES		해당사항 없음
7654	MARTIN	1400	수당 : 1400
7698	BLAKE		해당사항 없음
7782	CLARK		해당사항 없음
7788	SCOTT		해당사항 없음
7839	KING		해당사항 없음
7844	TURNER	0	수당없음
7876	ADAMS		해당사항 없음
7900	JAMES		해당사항 없음
7902	FORD		해당사항 없음
7934	MILLER		해당사항 없음