sql-study-day8

06 - 4 날짜 데이터를 다루는 날짜 함수

- 오라클에서 날짜 데이터, 즉 DATE형 데이터는 다음과 같이 간단한 연산이 가능하다.

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

- 오라클에서 제공하는 날짜 함수 중 가장 대표 함수는 SYSDATE 함수이다.

```
SELECT SYSDATE AS NOW,
SYSDATE-1 AS YESTERDAY,
SYSDATE+1 AS TOMORROW
FROM DUAL;
```

1을 빼거나 더했을 때 결과 날짜가 하루 이전이나 이후 날짜로 출력된다.

	NOW	YESTERDAY	TOMORROW
1	22/07/20	22/07/19	22/07/21

몇 개월 이후 날짜를 구하는 AD_MONTHS 함수

- 특정 날짜에 지정한 개월 수 이후 날짜 데이터를 반환하는 함수이다.

```
ADD_MONTHS([날짜 데이터(필수)], [더할 개월 수(정수)(필수)]) // 1.
```

번호	설명
1.	특정 날짜 데이터에 입력한 개월 수만큼의 이후 날짜를 출력합니다.

```
SELECT SYSDATE,
ADD_MONTHS(SYSDATE, 3)
FROM DUAL;
```

	SYSDATE	ADD_MONTHS(SYSDATE,3)
1	22/07/20	22/10/20

- ADD_MONTHS 함수를 WHERE절에 사용하는 것도 가능하다.

```
SELECT EMPNO,
ENAME, HIREDATE, SYSDATE
FROM EMP
WHERE ADD_MONTHS(HIREDATE, 486) > SYSDATE;
```

	EMPNO	ENAME	HIREDATE	SYSDATE
1	7788	SCOTT	87/04/19	22/07/20
2	7876	ADAMS	87/05/23	22/07/20
3	7934	MILLER	82/01/23	22/07/20

두 날짜 간의 개월 수 차이를 구하는 MONTHS_BETWEEN 함수

- 두 개의 날짜 데이터를 입력하고 두 날짜 간의 개월 수 차이를 구하는 데 사용한다.

```
MONTHS_BETWEEN([날짜 데이터1(필수)], [날짜 데이터2(필수)]) // 1.
```

번호	설명
1.	두 날짜 데이터 간의 날짜 차이를 개월 수로 계산하여 출력합니다.

```
SELECT EMPNO, ENAME, HIREDATE, SYSDATE,
MONTHS_BETWEEN(HIREDATE, SYSDATE) AS MONTHS1,
MONTHS_BETWEEN(SYSDATE, HIREDATE) AS MONTHS2,
TRUNC(MONTHS_BETWEEN(SYSDATE, HIREDATE)) AS MONTHS3
FROM EMP;
```

	EMPNO	ENAME	HIREDATE	SYSDATE	MONTHS1	MONTHS2	MONTHS3
1	7369	SMITH	80/12/17	22/07/20	-499.120609692353643966547192353643966547	499.120609692353643966547192353643966547	499
2	7499	ALLEN	81/02/20	22/07/20	-497	497	497
3	7521	WARD	81/02/22	22/07/20	-496.959319369772998805256869772998805257	496.959319369772998805256869772998805257	496
4	7566	JONES	81/04/02	22/07/20	-495.604480660095579450418160095579450418	495.604480660095579450418160095579450418	495
5	7654	MARTIN	81/09/28	22/07/20	-489.765770982676224611708482676224611708	489.765770982676224611708482676224611708	489
6	7698	BLAKE	81/05/01	22/07/20	-494.636738724611708482676224611708482676	494.636738724611708482676224611708482676	494
7	7782	CLARK	81/06/09	22/07/20	-493.378674208482676224611708482676224612	493.378674208482676224611708482676224612	493
8	7788	SCOTT	87/04/19	22/07/20	-423.056093563321385902031063321385902031	423.056093563321385902031063321385902031	423
9	7839	KING	81/11/17	22/07/20	-488.120609692353643966547192353643966547	488.120609692353643966547192353643966547	488
10	7844	TURNER	81/09/08	22/07/20	-490.410932272998805256869772998805256 87	490.410932272998805256869772998805256 87	490
11	7876	ADAMS	87/05/23	22/07/20	-421.927061305256869772998805256869772999	421.927061305256869772998805256869772999	421
12	7900	JAMES	81/12/03	22/07/20	-487.57222259557945041816009557945041816	487.57222259557945041816009557945041816	487
13	7902	FORD	81/12/03	22/07/20	-487.57222259557945041816009557945041816	487.57222259557945041816009557945041816	487
14	7934	MILLER	82/01/23	22/07/20	-485.927061305256869772998805256869772999	485.927061305256869772998805256869772999	485

MONTHS1, MONTHS2에서 알 수 있듯이 비교 날짜의 입력 위치에 따라 음수 또는 양수가 나올 수 있다. 개월 수 차이는 소수점 단위까지 결과가 나오므로 MONTHS3과 같이 TRUNC 함수를 조합하면 개월 수 차이를 정수로 출력할 수도 있다.

돌아오는 요일, 달의 마지막 날짜를 구하는 NEXT_DAY, LAST_DAY 함수

- NEXT_DAY 함수는 날짜 데이터와 요일 문자열을 입력한다. 입력한 날짜 데이터에서 돌아오는 요일의 날짜를 반환한다.

```
NEXT_DAY([날짜 데이터(필수)], [요일 문자(필수)]) // 1.
```

번호	설명
1.	특정 날짜를 기준으로 돌아오는 요일의 날짜를 출력해 주는 함수입니다.

- LAST_DAY 함수는 하나의 날짜 데이터만을 입력 데이터로 사용하며 해당 날짜가 속한 달의 마지막 날짜를 반환해 주는 함수이다.

```
LAST_DAY([날짜 데이터(필수)]) // 1.
```

번호	설명
1.	특정 날짜가 속한 달의 마지막 날짜를 출력해 주는 함수이다.

날짜의 반올림, 버림을 하는 ROUND, TRUNC 함수

- 숫자 데이터를 반올림, 버림 처리에 사용한 ROUND, TRUNC 함수는 날짜 데이터를 입력 데이터로 사용할 수도 있다. 이때는 소수점 위치 정보를 입력하지 않고 반올림, 버림의 기준이 될 포맷(format) 값을 지정해 준다.

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])

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

---

06 - 5 자료형을 변환하는 형 변환 함수

- '자동 형 변환' 이라고도 불리는 암시적 형 변환

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

	EMPNO	ENAME	EMPNO+'500'
1	7788	SCOTT	8288

- 자료형을 직접 지정해 주는 방식인 명시적 형 변환

종류	설명
TO_CHAR	숫자 또는 날짜 데이터를 문자 데이터로 변환
TO_NUMBER	문자 데이터를 숫자 데이터로 변환
TO_DATE	문자 데이터를 날짜 데이터로 변환

문자를 중심으로 숫자 또는 날짜 데이터의 변환이 가능하다.

날짜, 숫자 데이터를 문자 데이터로 변환하는 TO_CHAR 함수

- TO_CHAR 함수는 날짜, 숫자 데이터를 문자 데이터로 변환해 주는 함수이다.

```
TO_CHAR([날짜 데이터(필수)], '[출력되길 원하는 문자 형태(필수)]') // 1.
```

번호	설명
1.	날짜 데이터를 원하는 형태의 문자열로 출력합니다.

문자 데이터를 숫자 데이터로 변환하는 TO_NUMBER 함수

```
TO_NUMBER('[문자열 데이터(필수)]', '[인식될 숫자형태(필수)]') // 1.
```

번호	설명
1.	문자열을 지정한 형태의 숫자로 인식하여 숫자 데이터로 변환합니다.

문자 데이터를 날짜 데이터로 변환하는 TO_DATE 함수

```
TO_DATE('[문자열 데이터(필수)]', '[인식될 날짜형태(필수)]') // 1.
```

번호	설명
1.	문자열 데이터를 날짜형의 데이터로 변환합니다.

06 - 6 NULL 처리 함수

- 04장과 05장에서 데이터가 NULL이면 산술 연산자나 비교 연산자가 예상한 대로 동작하지 않는 것을 확인할 수 있었다. 하지만 특정 열의 데이터가 NULL일 경우에 연산 수행을 위해 데이터를 NULL이 아닌 다른 값으로 대체해 주어야 할 때가 종종 발생한다. 이때 여기에서 배울 NVL 함수와 NVL2 함수를 유용하게 사용할 수 있다.

NVL 함수의 기본 사용법

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

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

```
SELECT EMPNO, ENAME, SAL, COMM, SAL+COMM,
NVL(COMM, 0),
SAL+NVL(COMM, 0)
FROM EMP;
```

	EMPNO	ENAME	SAL	COMM	SAL+COMM	NVL(COMM,0)	SAL+NVL(COMM,0)
1	7369	SMITH	800	(null)	(null)	0	800
2	7499	ALLEN	1600	300	1900	300	1900
3	7521	WARD	1250	500	1750	500	1750
4	7566	JONES	2975	(null)	(null)	0	2975
5	7654	MARTIN	1250	1400	2650	1400	2650
6	7698	BLAKE	2850	(null)	(null)	0	2850
7	7782	CLARK	2450	(null)	(null)	0	2450
8	7788	SCOTT	3000	(null)	(null)	0	3000
9	7839	KING	5000	(null)	(null)	0	5000
10	7844	TURNER	1500	0	1500	0	1500
11	7876	ADAMS	1100	(null)	(null)	0	1100
12	7900	JAMES	950	(null)	(null)	0	950
13	7902	FORD	3000	(null)	(null)	0	3000
14	7934	MILLER	1300	(null)	(null)	0	1300

EMP 테이블의 급여 외 추가 수당을 의미하는 COMM 열 값이 NULL인 데이터를 0으로 대체하여 연산이 가능하다는 것을 확인할 수 있다. 이렇게 NVL 함수는 NULL 처리를 위해 자주 사용한다.

NVL2 함수의 기본 사용법

- NVL2 함수는 NVL 함수와 비슷하지만 데이터가 NULL이 아닐 때 반환할 데이터를 추가로 지정해 줄 수 있다.

```
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)]) // 1.
```

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

```
SELECT EMPNO, ENAME, COMM,
NVL2(COMM, 'O', 'X'),
NVL2(COMM, SAL*12+COMM, SAL*12) AS ANNSAL
FROM EMP;
```

	EMPNO	ENAME	COMM	NVL2(COMM,'O','X')	ANNSAL
1	7369	SMITH	(null)	X	9600
2	7499	ALLEN	300	O	19500
3	7521	WARD	500	O	15500
4	7566	JONES	(null)	X	35700
5	7654	MARTIN	1400	O	16400
6	7698	BLAKE	(null)	X	34200
7	7782	CLARK	(null)	X	29400
8	7788	SCOTT	(null)	X	36000
9	7839	KING	(null)	X	60000
10	7844	TURNER	0	O	18000
11	7876	ADAMS	(null)	X	13200
12	7900	JAMES	(null)	X	11400
13	7902	FORD	(null)	X	36000
14	7934	MILLER	(null)	X	15600

NVL2 함수는 NVL함수와는 달리 NULL이 아닌 경우에 반환 데이터까지 지정할 수 있으므로 좀 더 다양한 용도로 활용 가능하다.

06 - 7 상황에 따라 다른 데이터를 반환하는 DECODE 함수와 CASE문

- 특정 열 값이나 데이터 값에 따라 어떤 데이터를 반환할지 정할 때는 DECODE 함수 또는 CASE문을 사용한다.

DECODE 함수

- DECODE 함수는 기준이 되는 데이터를 먼저 지정한 후 해당 데이터 값에 따라 다른 결과 값을 내보내는 함수이다.

```
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])
```

CASE문

- 기준 데이터를 반드시 명시하고 그 값에 따라 반환 데이터를 정하는 DECODE 함수와 달리 CASE문은 각 조건에 사용하는 데이터가 서로 상관없어도 된다. 또 기준 데이터 값이 같은(=) 데이터 외에 다양한 조건을 사용할 수 있다.

```
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
```

잊기 전에 한 번 더!

- Q1. EMPNO 열에는 EMP 테이블에서 사원 이름(ENAME)이 다섯 글자 이상이며 여섯 글자 미만인 사원 정보를 출력합니다. MASKING_EMPNO 열에는 사원 번호(EMPNO) 앞 두 자리 외 뒷자리를 * 기호로 출력합니다. 그리고 MASKING_ENAME 열에는 사원 이름의 첫 글자만 보여 주고 나머지 글자 수만큼 * 기호로 출력하세요.

```
SELECT EMPNO, RPAD(SUBSTR(EMPNO, 1, 2), 4, '*') AS MASKING_EMPNO,
ENAME, RPAD(SUBSTR(ENAME, 1, 1), 5, '*') AS MASKING_ENAME
FROM EMP
WHERE LENGTH(ENAME) >= 5
AND LENGTH(ENAME) < 6;
```

- Q2. EMP 테이블에서 사원들의 월 평균 근무일 수는 21.5일입니다. 하루 근무 시간을 8시간으로 보았을 때 사원들의 하루 급여(DAY_PAY)와 시급(TIME_PAY)을 계산하여 결과를 출렵합니다. 단 하루 급여는 소수점 세 번째 자리에서 버리고, 시급은 두 번째 소수점에서 반올림하세요.

```
SELECT EMPNO, ENAME, SAL,
TRUNC(SAL/21.5, 2) AS DAY_PAY,
ROUND(SAL/21.5/8, 1) AS TIME_PAY //ROUND(DAY_PAY/8, 1) AS TIME_PAY
FROM EMP;
```

- Q3. EMP 테이블에서 사원들은 입사일(HIREDATE)을 기준으로 3개월이 지난 후 첫 월요일에 정직원이 됩니다. 사원들이 정직원이 되는 날짜(R_JOB)를 YYYY-MM-DD 형식으로 오른쪽과 같이 출력해 주세요. 단 추가 수당(COMM)이 없는 사원의 추가 수당은 N/A로 출력하세요.

```
SELECT EMPNO, ENAME, HIREDATE,
       TO_CHAR(NEXT_DAY(ADD_MONTHS(HIREDATE, 3), '월요일'), 'YYYY-MM-DD') AS R_JOB,
       NVL(TO_CHAR(COMM), 'N/A') AS COMM
  FROM EMP;
```

- Q4. EMP 테이블의 모든 사원을 대상으로 직속 상관의 사원 번호(MGR)를 다음과 같은 조건을 기준으로 변환해서 CHG_MGR 열에 출력하세요.

1. 직속 상관의 사원 번호가 존재하지 않을 경우 : 0000
2. 직속 상관의 사원 번호 앞 두자리가 75일 경우 : 5555
3. 직속 상관의 사원 번호 앞 두자리가 76일 경우 : 6666

4. 직속 상관의 사원 번호 앞 두자리가 77일 경우 : 7777
5. 직속 상관의 사원 번호 앞 두자리가 78일 경우 : 8888
6. 그 외 직속 상관 사원 번호의 경우 : 본래 직속 상관의 사원 번호 그대로 출력

```
SELECT EMPNO, ENAME, MGR,
       CASE
          WHEN MGR IS NULL THEN '0000'
          WHEN SUBSTR(MGR, 1, 2) = '78' THEN '8888'
          WHEN SUBSTR(MGR, 1, 2) = '77' THEN '7777'
          WHEN SUBSTR(MGR, 1, 2) = '76' THEN '6666'
          WHEN SUBSTR(MGR, 1, 2) = '75' THEN '5555'
          ELSE TO_CHAR(MGR)
       END AS CHG_MGR
  FROM EMP;
```