

03. SELECT문 함수

문자 함수를 학습한다. 숫자 함수를 학습한다. 날짜 함수를 학습한다.
자료형을 변환시키고자 할 때 사용하는 함수를 학습한다.
NULL을 다른 값으로 변환하는 NVL 함수를 학습한다.
DECODE와 CASE에 대해서 학습한다.

1) DUAL 테이블과 SQL 함수 분류

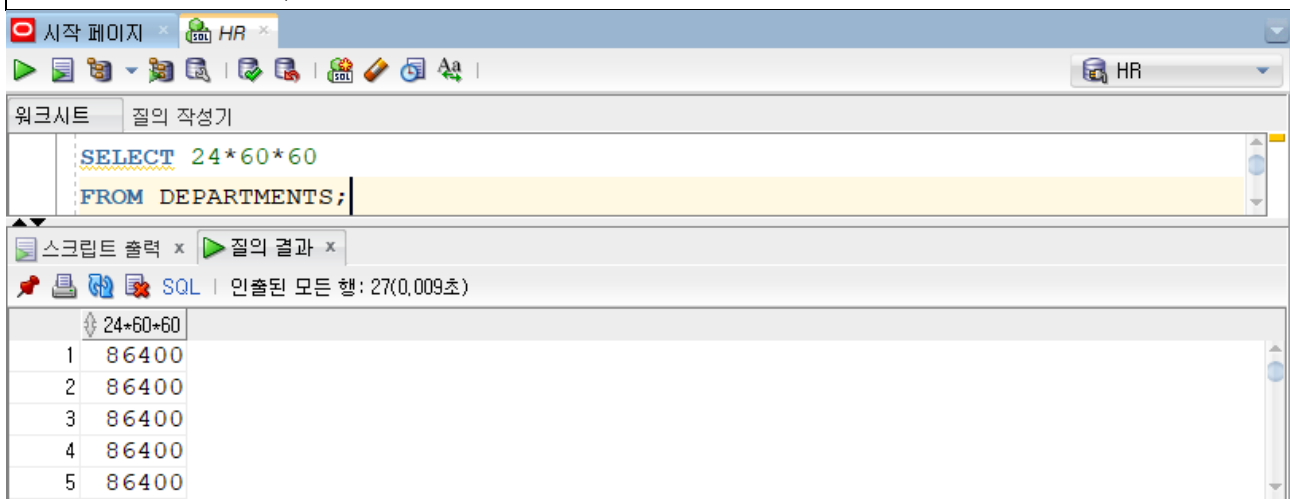
함수를 배우기 전에 한 행으로 결과를 출력하기 위한 테이블인 DUAL 테이블에 대해 살펴본다.

① DAUL 테이블

오라클에서 1일이 몇 초인지 환산하고자 한다. 1일 24시간이고 1시간은 60분이며 1분은 60초이므로 $24*60*60$ 하면 하루가 몇 초인지 계산된다. 결과를 얻으려고 이 산술식을 오라클 프롬프트에 바로 입력하면 오류가 발생한다. 왜냐하면 $24*60*60$ 은 SQL문의 종류에 속하지 않은 명령이기 때문이다.

<예> DEPARTMENTS 테이블로 1일을 초로 환산하는 산술 계산식을 작성하면 다음과 같다.

```
SELECT 24*60*60
FROM DEPARTMENTS;
```

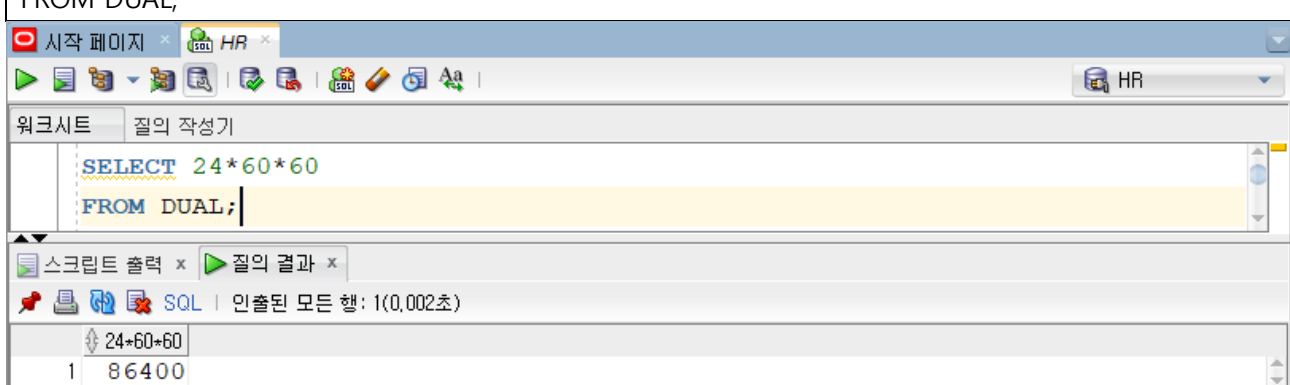


The screenshot shows the Oracle SQL Developer interface. The top pane contains the SQL query: `SELECT 24*60*60 FROM DEPARTMENTS;`. The bottom pane shows the execution results in a table with 5 rows, all containing the value 86400. The status bar indicates that 27 rows were returned from the DEPARTMENTS table.

	24*60*60
1	86400
2	86400
3	86400
4	86400
5	86400

DEPARTMENTS 테이블이 27개의 로우로 구성되어 있다.

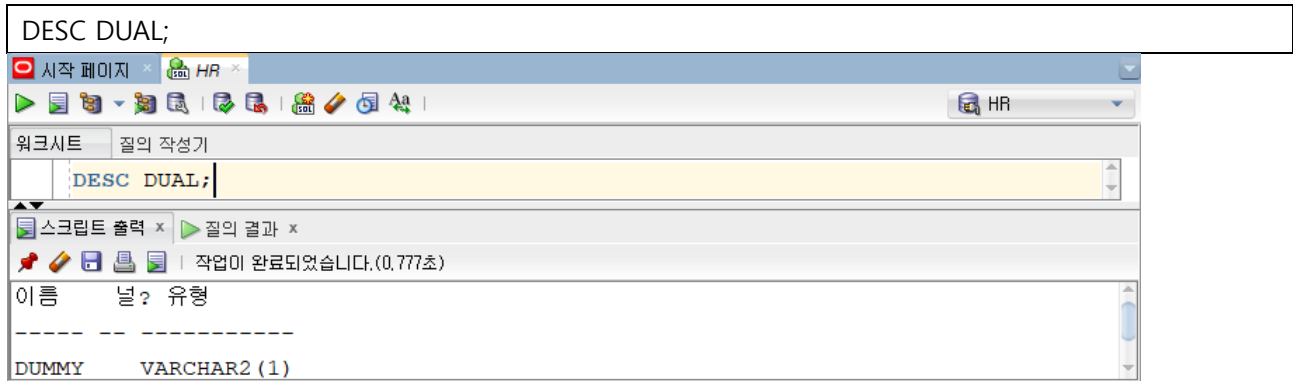
```
SELECT 24*60*60
FROM DUAL;
```



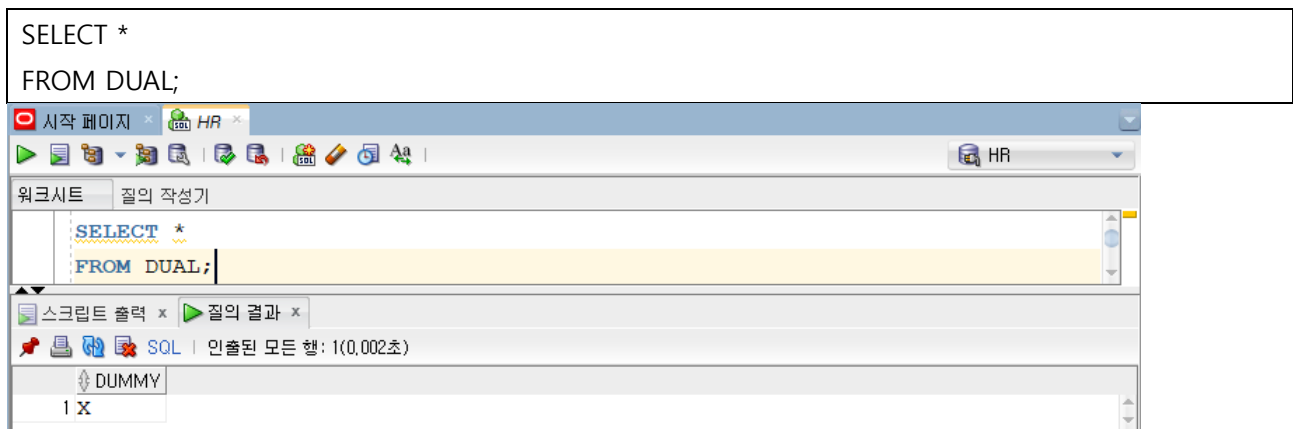
The screenshot shows the Oracle SQL Developer interface. The top pane contains the SQL query: `SELECT 24*60*60 FROM DUAL;`. The bottom pane shows the execution results in a table with 1 row, containing the value 86400. The status bar indicates that 1 row was returned from the DUAL table.

	24*60*60
1	86400

DUAL 테이블은 산술 연산이나 가상 칼럼 등의 값을 한번만 출력하고 싶을 때 많이 사용하는 아주 유용한 테이블로서 DUMMY라는 한 개의 칼럼으로 구성되어 있다.



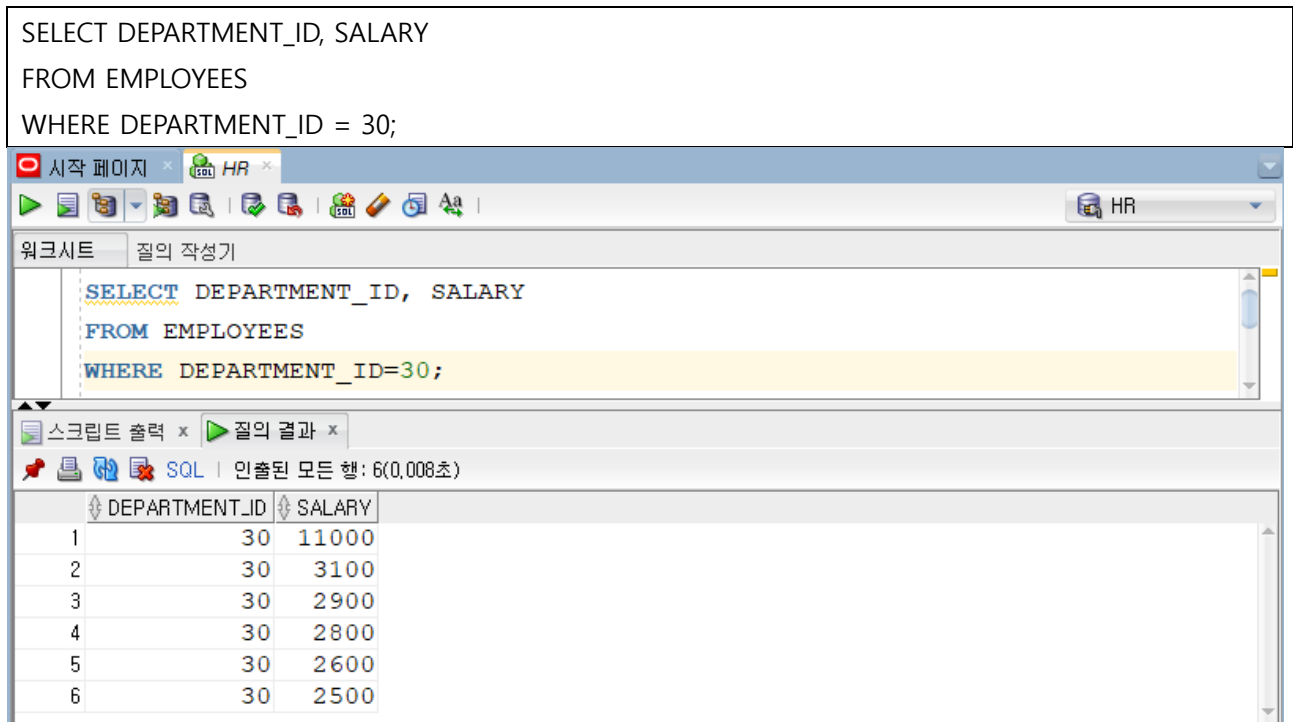
DUMMY칼럼에는 X라는 한 개의 문자만을 값으로 가진 단 하나의 로우만을 저장하고 있다.



② 단일 행 함수와 그룹함수로 SQL 함수 분류

함수를 이용하여 복잡한 질의를 간결하게 표현할 수 있다.

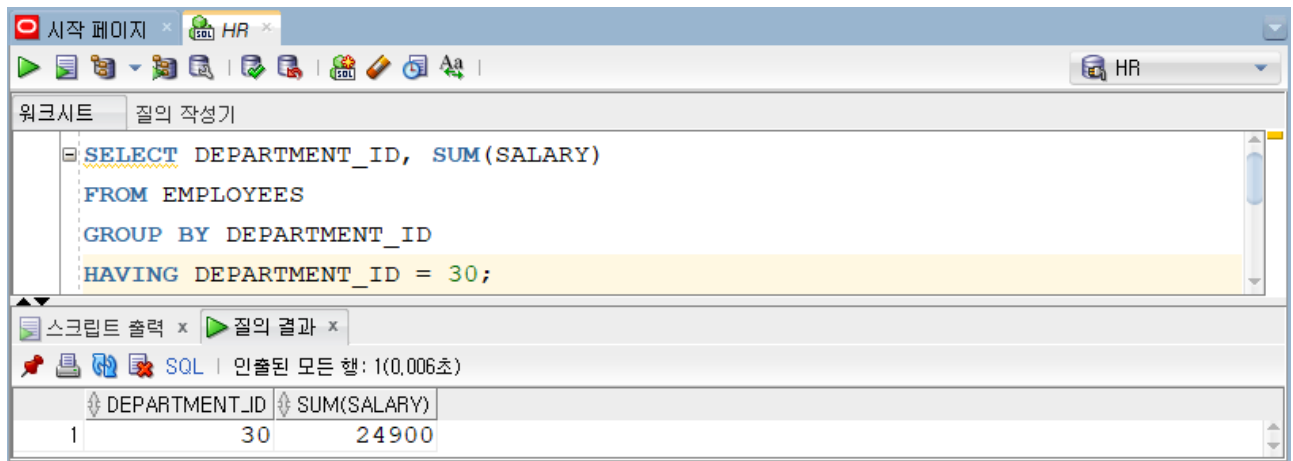
<예> 30번 부서 소속 사원의 급여를 출력하는 쿼리문



30번 부서 소속 사원이 6명이기에 결과가 6개의 행으로 나타난다.

<예> 그룹함수를 이용해서 30번 부서 소속 사원의 총 급여를 구하는 쿼리문

```
SELECT DEPARTMENT_ID, SUM(SALARY)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING DEPARTMENT_ID = 30;
```



30번 부서 소속 사원이 6명임에도 결과는 하나의 행으로 나온다.



6개의 행에 대해서 단일행 함수의 결과는 6개의 행으로 구해진다.	6개의 행에 대해서 그룹함수의 결과는 1개의 행으로 구해진다.
--------------------------------------	------------------------------------

단일행 함수(행 마다 함수가 적용되어 결과를 반환한다.)

구 분	설 명
문자 함수	문자열을 다른 형태로 변환하여 나타낸다.
숫자 함수	숫자 값을 다른 형태로 변환하여 나타낸다.
날짜 함수	날짜 값을 다른 형태로 변환하여 나타낸다.
변환 함수	문자, 날짜, 숫자 값을 서로 다른 타입으로 변환하여 나타낸다.
일반 함수	기타 함수

그룹함수(하나 이상의 행을 그룹으로 묶어 연산하여 총합, 평균 등 하나의 결과로 나타난다.)

구 분	설 명
SUM	그룹의 누적 합계를 반환한다.
AVG	그룹의 평균을 반환한다.
COUNT	그룹의 총 개수를 반환한다.
MAX	그룹의 최댓값을 반환한다.
MIN	그룹의 최솟값을 반환한다.
STDDEV	그룹의 표준편차(분산 값의 제곱근을 말한다)를 반환한다.
VARIANCE	그룹의 분산(주어진 범위의 개별값과 평균값과의 사이인 편차를 구해 이를 제곱해서 평균값을 말한다)을 반환한다.

2) 문자함수

문자형의 값을 조작하여 변환된 문자 값을 반환한다

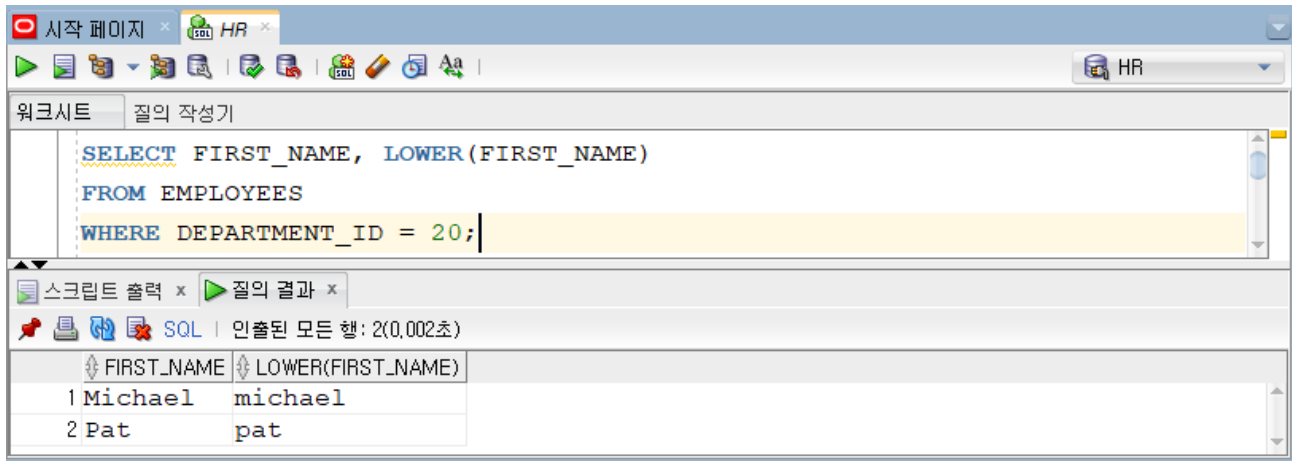
구 분	설 명
LOWER	소문자로 변환한다.
UPPER	대문자로 변환한다.
INITCAP	첫 글자만 대문자로 나머지 글자는 소문자로 변환한다.
CONCAT	문자의 값을 연결한다.
SUBSTR	문자를 잘라 추출한다. (한글 1Byte)
SUBSTRB	문자를 잘라 추출한다. (한글 2Byte)
LENGTH	문자의 길이를 반환한다.(한글 1Byte)
LENGTHB	문자의 길이를 반환한다.(한글 2Byte)
INSTR	특정 문자의 위치 값을 반환한다.(한글 1Byte)
INSTRB	특정 문자의 위치 값을 반환한다.(한글 2Byte)
LPAD, RPAD	입력 받은 문자열과 기호를 정렬하여 특정 길이의 문자열로 반환한다.
TRIM	잘라내고 남은 문자를 표시한다.
CONVERT	CHAR SET을 변환한다.
CHR	ASCII 코드 값으로 변환한다.
ASCII	ASCII 코드 값을 문자로 변환한다.
REPLACE	문자열에서 특정 문자를 변경한다.

① 소문자로 변환하는 LOWER 함수: 입력한 문자 값을 소문자로 변환하는 함수.

SELECT 'DataBase', LOWER ('DataBase')	
FROM DUAL;	
↕ 'DATABASE'	↕ LOWER('DATABASE')
1 DataBase	database

<예> 사원 테이블에서 부서번호가 20번인 사원명을 모두 소문자로 변환

```
SELECT FIRST_NAME, LOWER(FIRST_NAME)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 20;
```



② 대문자로 변환하는 UPPER 함수

```
SELECT 'DataBase', UPPER('DataBase')
FROM DUAL;
```

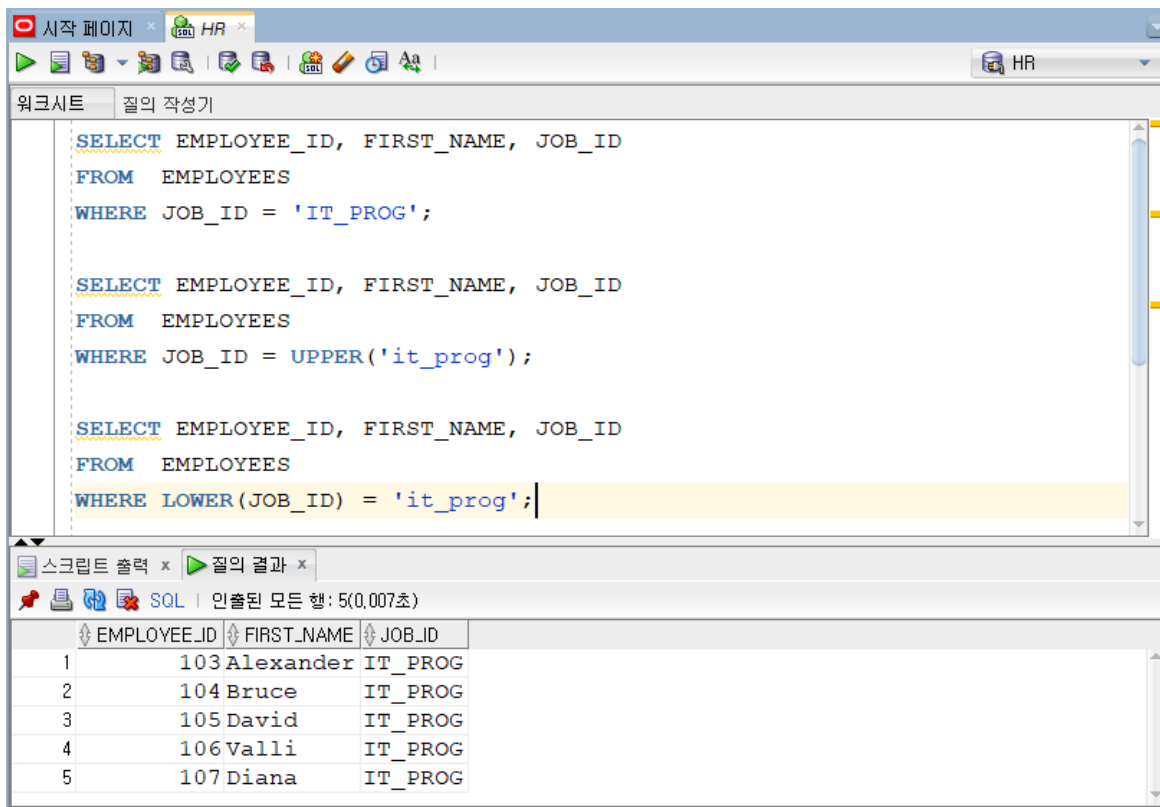
	'DATABASE'	UPPER('DATABASE')
1	DataBase	DATABASE

<예> job_id가 'it_prog'인 사원을 검색

```
SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID
FROM EMPLOYEES
WHERE JOB_ID = 'IT_PROG';
```

```
SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID
FROM EMPLOYEES
WHERE JOB_ID = UPPER('it_prog');
```

```
SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID
FROM EMPLOYEES
WHERE LOWER(JOB_ID) = 'it_prog';
```



③ 첫 글자만 대문자로 나머지는 소문자로 변환하는 INITCAP 함수

```

SELECT INITCAP('DATA BASE PROGRAM')
FROM DUAL;

```

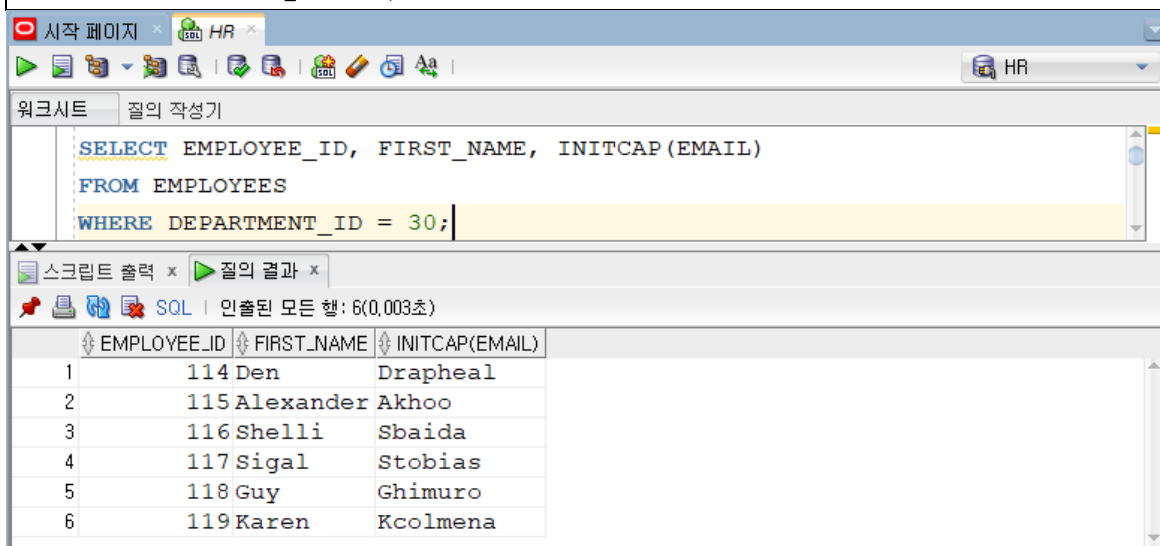
1	INITCAP('DATABASEPROGRAM')
1	Data Base Program

<예> 사원 테이블의 30번 부서에 소속된 사원이름의 첫 글자만 대문자로

```

SELECT EMPLOYEE_ID, FIRST_NAME, INITCAP(EMAIL)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;

```



<문제> 'jking'이란 이메일을 갖은 직원의 이름과 급여와 커미션을 출력하라. (INITCAP, UPPER 사용)

④ 두 문자를 연결하는 CONCAT 함수

```
SELECT CONCAT('Data', 'Base')
FROM DUAL;
```

	CONCAT('DATA','BASE')
1	DataBase

```
SELECT CONCAT(FIRST_NAME, '($' || SALARY || ')') AS "사원 정보"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The screenshot shows the Oracle SQL Developer interface. The top pane displays the following SQL query:

```
SELECT CONCAT(FIRST_NAME, '($' || SALARY || ')') AS "사원 정보"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The bottom pane shows the query results in a table format:

사원 정보
1 Den (\$11000)
2 Alexander (\$3100)
3 Shelli (\$2900)
4 Sigal (\$2800)
5 Guy (\$2600)
6 Karen (\$2500)

⑤ 문자 길이를 구하는 LENGTH/LENGTHB 함수

문자 상수나, 칼럼에 저장된 데이터 값이 몇 개의 문자로 구성되었는지 길이를 알려주는 함수
- 글자의 개수를 구한다.

```
SELECT LENGTH('DataBase'), LENGTH('데이터베이스')
FROM DUAL;
```

	LENGTH('DATABASE')	LENGTH('데이터베이스')
1	8	6

- 메모리에 차지하는 바이트 수를 구한다

```
SELECT LENGTHB('DataBase'), LENGTHB('데이터베이스')
FROM DUAL;
```

	LENGTHB('DATABASE')	LENGTHB('데이터베이스')
1	8	18

<예> 20번 부서 소속 직원들의 이름의 길이를 출력하기

```
SELECT DEPARTMENT_ID, EMPLOYEE_ID, FIRST_NAME, LENGTH(FIRST_NAME)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 20;
```

The screenshot shows the SQL Developer interface. The query editor contains the following SQL statement:

```
SELECT DEPARTMENT_ID, EMPLOYEE_ID, FIRST_NAME, LENGTH(FIRST_NAME)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 20;
```

The results pane shows the output of the query:

	DEPARTMENT_ID	EMPLOYEE_ID	FIRST_NAME	LENGTH(FIRST_NAME)
1	20	201	Michael	7
2	20	202	Pat	3

<예> 직원 중 이름이 4글자인 직원의 이름을 소문자로 출력

```
SELECT DEPARTMENT_ID, EMPLOYEE_ID, LOWER(FIRST_NAME)
FROM EMPLOYEES
WHERE LENGTH(FIRST_NAME)=4;
```

The screenshot shows the SQL Developer interface. The query editor contains the following SQL statement:

```
SELECT DEPARTMENT_ID, EMPLOYEE_ID, LOWER(FIRST_NAME)
FROM EMPLOYEES
WHERE LENGTH(FIRST_NAME)=4;
```

The results pane shows the output of the query:

	DEPARTMENT_ID	EMPLOYEE_ID	LOWER(FIRST_NAME)
1	100	110	john
2	100	113	luis
3	50	121	adam
4	50	139	john
5	80	145	john
6	80	167	amit
7	80	168	lisa
8	80	177	jack
9	50	181	jean

<문제> 이름이 6글자 이상인 직원의 직원번호와 이름과 급여를 출력하라.

⑥ 문자열 일부만 추출하는 SUBSTR/SUBSTRB 함수

SUBSTR(대상, 시작위치, 추출할 개수)

```
SELECT SUBSTR('DataBase', 1, 3)
FROM DUAL;
```

1	2	3	4	5	6	7	8
D	a	t	a	B	a	s	e

```
SUBSTR('DATABASE',1,3)
1 Dat
```

- 시작위치 인자 값: 음수
뒤 쪽에서부터 세어서 시작위치를 잡는다.

```
SELECT SUBSTR('DataBase',-4, 3)
FROM DUAL;
```

-8	-7	-6	-5	-4	-3	-2	-1
D	a	t	a	B	a	s	e

추출 글자수: 3개

시작위치: -4(뒤쪽에서 4번째)

```
SUBSTR('DATABASE',-4,3)
1 Bas
```

<예> 20번 부서 직원들 중의 입사 년도 알아내기

```
SELECT FIRST_NAME, HIRE_DATE, SUBSTR(HIRE_DATE, 1, 2)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 20;
```

The screenshot shows the SQL Developer interface. The top pane displays the query: `SELECT FIRST_NAME, HIRE_DATE, SUBSTR(HIRE_DATE, 1, 2) FROM EMPLOYEES WHERE DEPARTMENT_ID = 20;`. The bottom pane shows the results of the query execution, with 2 rows returned. The first row is for Michael, hired on 04/02/17, with the year 04 extracted. The second row is for Pat, hired on 05/08/17, with the year 05 extracted.

FIRST_NAME	HIRE_DATE	SUBSTR(HIRE_DATE,1,2)
1 Michael	04/02/17	04
2 Pat	05/08/17	05

<문제> 03년도에 입사한 사원 알아내기

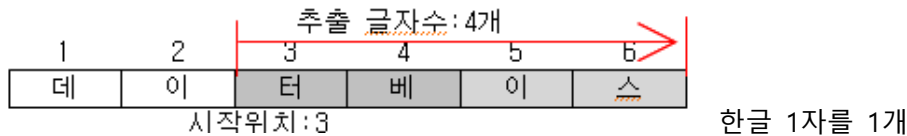
(비교 연산자와 AND 연산자, BETWEEN AND 연산자, SUBSTR 함수)

<문제> 이름이 k로 끝나는 직원을 검색

(LIKE 연산자와 와일드 카드 (%), SUBSTR 함수)

- 바이트 수를 기준으로 문자열 일부만 추출 SUBSTRB

```
SELECT SUBSTR('데이터베이스', 3, 4)
FROM DUAL;
```



```
SUBSTR('데이터베이스', 3, 4)
1 터베이스
```

```
SELECT SUBSTRB('데이터베이스', 4, 6)
FROM DUAL;
```

한글 1자를 3 바이트

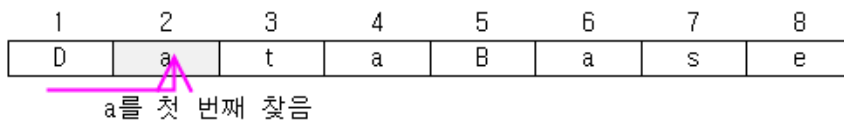
```
SUBSTRB('데이터베이스', 4, 6)
1 이터
```

- ⑦ 특정 문자의 위치를 구하는 INSTR/INSTRB 함수

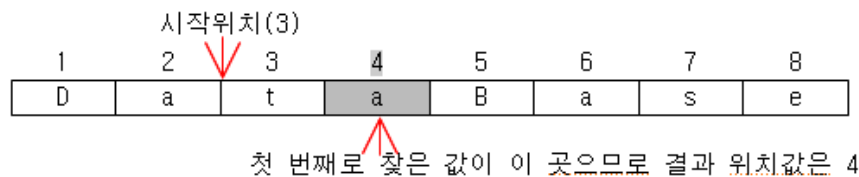
대상 문자열이나 칼럼에서 특정 문자가 나타나는 위치를 알려준다.

INSTR(대상, 찾을 글자, 시작위치, 몇 번째 발견)

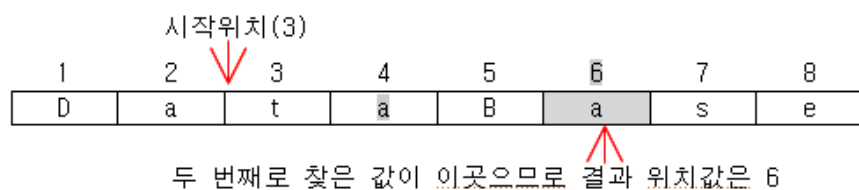
INSTR('DataBase', 'a')



INSTR('DataBase', 'a', 3, 1)



INSTR('DataBase', 'a', 3, 2)



```
SELECT INSTR('DataBase', 'B')
FROM DUAL;
```

1	2	3	4	5	6	7	8
D	a	t	a	B	a	s	e

B의 위치를 찾음

INSTR('DATABASE','B')
1 5

<예> 30번 부서 소속 사원이름에 e 자가 어디에 위치하는지 알려주는 쿼리문

```
SELECT DEPARTMENT_ID, FIRST_NAME, INSTR(FIRST_NAME, 'e')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The screenshot shows the Oracle SQL Developer interface. The query editor contains the following SQL statement:

```
SELECT DEPARTMENT_ID, FIRST_NAME, INSTR(FIRST_NAME, 'e')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The results grid displays the following data:

DEPARTMENT_ID	FIRST_NAME	INSTR(FIRST_NAME, 'E')
30	Den	2
30	Alexander	3
30	Shelli	3
30	Sigal	0
30	Guy	0
30	Karen	4

⑧ 특정 기호로 채우는 LPAD/RPAD 함수

- LPAD(LEFT PADDING) 함수는 칼럼이나 대상 문자열을 명시된 자릿수에서 오른쪽에 나타나고, 남은 왼쪽 자리를 특정 기호로 채운다.

```
SELECT LPAD('DataBase', 20, '$')
FROM DUAL;
```

LPAD('DATABASE',20,'\$')
1 \$\$\$\$\$\$\$\$\$\$\$\$\$\$DataBase

- RPAD(RIGHT PADDING) 함수는 칼럼이나 대상 문자열을 명시된 자릿수에서 왼쪽에 나타나고, 남은 오른쪽 자리를 특정 기호로 채운다.

```
SELECT RPAD('DataBase', 20, '$')
FROM DUAL;
```

RPAD('DATABASE',20,'\$')
1 DataBase\$\$\$\$\$\$\$\$\$\$\$\$\$

```
SELECT RPAD('123-4535', 12, '(02)')
FROM DUAL;
```

RPAD('123-4535',12,'(02)')
1 123-4535 (02)

⑨ 특정 문자를 잘라내는 TRIM 함수

칼럼이나 대상 문자열에서 특정 문자가 첫 번째 글자이거나 마지막 글자라면 잘라내고 남은 문자열만 반환.

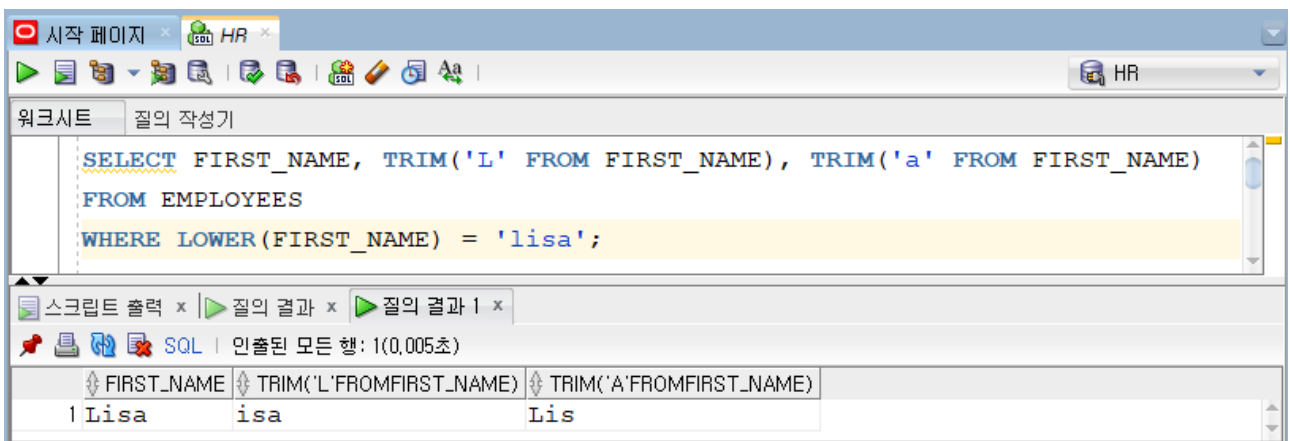
TRIM([LEADING, TRAILING, BOTH] [, trim_character] [FROM] trim_source)

```
SELECT TRIM('a' FROM 'aaaaDataBase programingaaaa')
FROM DUAL;
```

TRIM('A' FROM 'AAAADATABASEPROGRAMINGAAAA')
1 DataBase programing

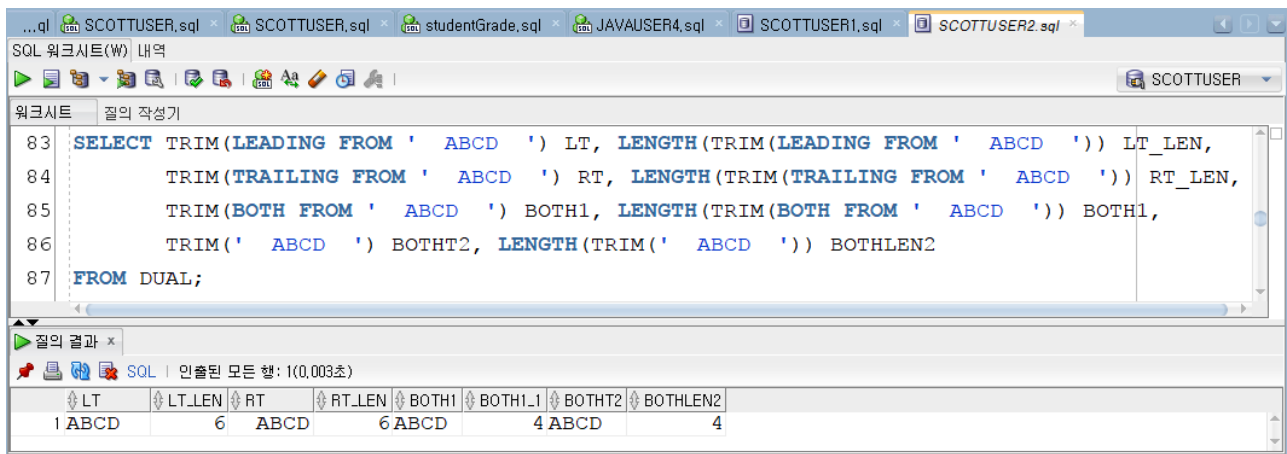
<예> Lisa란 사람의 이름에서 L 와 a 를 잘라내자.

```
SELECT FIRST_NAME, TRIM('L' FROM FIRST_NAME), TRIM('a' FROM FIRST_NAME)
FROM EMPLOYEES
WHERE LOWER(FIRST_NAME) = 'lisa';
```



```
SELECT TRIM(LEADING FROM ' ABCD ') LT, LENGTH(TRIM(LEADING FROM ' ABCD ')) LT_LEN,
       TRIM(TRAILING FROM ' ABCD ') RT, LENGTH(TRIM(TRAILING FROM ' ABCD ')) RT_LEN,
       TRIM(BOTH FROM ' ABCD ') BOTH1, LENGTH(TRIM(BOTH FROM ' ABCD ')) BOTH1,
       TRIM(' ABCD ') BOTHT2, LENGTH(TRIM(' ABCD ')) BOTHTLEN2
FROM DUAL;
```

LEADING는 왼쪽공백, TRAILING는 오른쪽 공백, BOTH는 양쪽 공백제거 디폴트는 BOTH 이다.



3) 숫자함수

숫자형 데이터를 조작하여 변환된 숫자 값을 반환하는 함수.

구 분	설 명
ABS	절대값을 반환한다.
COS	COSINE 값을 반환한다.
CEIL	소수점 아래를 올린다.(절상)
FLOOR	소수점 아래를 잘라낸다.(절사)
LOG	LOG값을 반환한다.
POWER	POWER(m, n) m의 n승을 반환한다.
SIGN	SIGN (n) n<0이면 -1, n=0이면 0, n>0이면 1을 반환한다.
SIN	SINE값을 반환한다.
TAN	TANGENT값을 반환한다.
ROUND	특정 자릿수에서 반올림한다.
TRUNC	특정 자릿수에서 잘라낸다. (버림)
MOD	입력 받은 수를 나눈 나머지 값을 반환한다.

① ABS 함수/ CEIL 함수/ FLOOR 함수

- ABS 함수는 절대값을 반환

SELECT ABS (-15)
FROM DUAL;
ABS(-15)
1 15

- CEIL(n) 함수는 소수점 아래를 올린다. FLOOR(n) 함수 소수점 아래를 잘라낸다.

SELECT CEIL (10.123), FLOOR (34.5678)
FROM DUAL;

	CEIL(10,123)	FLOOR(34,5678)
1	11	34

② 특정 자릿수에서 반올림하는 ROUND 함수

ROUND(대상, 표시할 자릿수)

ROUND 함수 예	결과	설 명
ROUND(35.12,1)	35.1	소수점 이하 2째 자리에서 반올림하여 소수점 이하 1째 자리까지 표시한다.
ROUND(21.125,2)	21.13	소수점 이하 3째 자리에서 반올림한다.
ROUND(47.51)	48	ROUND(47.51,0)과 동일한 문장으로 소수점 이하 1째 자리에서 반올림한다.
ROUND(834.12,-1)	830	두 번째 인자값이 음수(-1)이므로 일단위에서 반올림한다. 만약 ROUND(835.12,-1)이라면 결과는 840이 된다.
ROUND(653.54,-2)	700	두 번째 인자값이 -2이므로 십단위에서 반올림한다. 만약 ROUND(633.54,-2)이라면 결과는 600이 된다.

```
SELECT ROUND(12.345, 2), ROUND(34.567, 0), ROUND(56.789), ROUND(78.901, -1)
FROM DUAL;
```

	ROUND(12,345,2)	ROUND(34,567,0)	ROUND(56,789)	ROUND(78,901,-1)
1	12.35	35	57	80

③ 특정 자릿수에서 잘라내는 TRUNC 함수

지정한 자릿수 이하를 버린 결과를 구해주는 함수이다. 두 번째 전달 인자가 생략되면 0으로 간주한다.

```
SELECT TRUNC(12.345, 2), TRUNC(34.567, 0), TRUNC(56.789), TRUNC(78.901, -1)
FROM DUAL;
```

	TRUNC(12,345,2)	TRUNC(34,567,0)	TRUNC(56,789)	TRUNC(78,901,-1)
1	12.34	34	56	70

④ 나머지 값을 반환하는 MOD 함수

```
SELECT MOD(34, 2), MOD(34, 5), MOD(34, 7)
FROM DUAL;
```

	MOD(34,2)	MOD(34,5)	MOD(34,7)
1	0	4	6

<문제> 직원 번호가 짝수인 직원들의 직원번호와 이름과 직급을 출력하라.

4) 날짜함수

DATA(날짜)형에 사용하는 함수이며 결과 값으로 날짜 또는 기간을 얻는다.
기간은 주로 일 단위로 계산되지만, 월 단위로 계산되는 경우도 있다.

구 분	설 명
SYSDATE	시스템 저장된 현재 날짜를 반환한다.
MONTHS_BETWEEN	두 날짜 사이가 몇 개월인지를 반환한다.
ADD_MONTHS	특정 날짜에 개월 수를 더한다.
NEXT_DAY	특정 날짜에서 최초로 도래하는 인자로 받은 요일의 날짜를 반환한다.
LAST_DAY	해당 달의 마지막 날짜를 반환한다.
ROUND	인자로 받은 날짜를 특정 기준으로 반올림한다.
TRUNC	인자로 받은 날짜를 특정 기준으로 버림한다.

① 현재 날짜를 반환하는 SYSDATE

SELECT SYSDATE FROM DUAL;				
<table><tr><th></th><th>SYSDATE</th></tr><tr><td>1</td><td>21/02/17</td></tr></table>		SYSDATE	1	21/02/17
	SYSDATE			
1	21/02/17			

- 날짜형 데이터는 더하기나 빼기와 같은 연산을 할 수 있다.

SELECT SYSDATE-1 어제, SYSDATE 오늘, SYSDATE+1 내일
FROM DUAL;

	어제	오늘	내일
1	21/02/16	21/02/17	21/02/18

② 두 날짜 사이 간격을 계산하는 MONTHS_BETWEEN 함수

- 날짜와 날짜 사이의 개월 수를 구하는 함수

MONTHS_BETWEEN (date1, date2)

<예> 날짜 사이의 개월 수 구하기

SELECT FIRST_NAME, SYSDATE 오늘, TO_CHAR(HIRE_DATE,'YYYY/MM/DD') 입사일, TRUNC(MONTHS_BETWEEN(SYSDATE, HIRE_DATE)) 근무달수 FROM EMPLOYEES WHERE DEPARTMENT_ID = 30;
--

워크시트 | 질의 작성기

```
SELECT FIRST_NAME, SYSDATE 오늘, TO_CHAR(HIRE_DATE, 'YYYY/MM/DD') 입사일,
TRUNC(MONTHS_BETWEEN(SYSDATE, HIRE_DATE)) 근무달수
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

질의 결과 x

SQL | 인출된 모든 행: 6(0,011초)

	FIRST_NAME	오늘	입사일	근무달수
1	Den	21/02/09	2002/12/07	218
2	Alexander	21/02/09	2003/05/18	212
3	Shelli	21/02/09	2005/12/24	181
4	Sigal	21/02/09	2005/07/24	186
5	Guy	21/02/09	2006/11/15	170
6	Karen	21/02/09	2007/08/10	161

③ 개월 수를 더하는 ADD_MONTHS 함수

- 특정 개월 수를 더한 날짜를 구하는 함수

ADD_MONTHS(date, number)

<예> 입사일에서 3개월이 지난 날짜를 구하라

```
SELECT FIRST_NAME, HIRE_DATE, ADD_MONTHS(HIRE_DATE, 3)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

워크시트 | 질의 작성기

```
SELECT FIRST_NAME, HIRE_DATE, ADD_MONTHS(HIRE_DATE, 3)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

질의 결과 x

SQL | 인출된 모든 행: 6(0,002초)

	FIRST_NAME	HIRE_DATE	ADD_MONTHS(HIRE_DATE,3)
1	Den	02/12/07	03/03/07
2	Alexander	03/05/18	03/08/18
3	Shelli	05/12/24	06/03/24
4	Sigal	05/07/24	05/10/24
5	Guy	06/11/15	07/02/15
6	Karen	07/08/10	07/11/10

④ 해당 요일의 가장 가까운 날짜를 반환하는 NEXT_DAY 함수

지정 요일에 해당되는 날짜를 반환하는 데 해당 날짜를 기준으로 가장 빨리 다가오는 날짜를 반환한다.

NEXT_DAY (date, 요일)

오늘을 기준으로 최초로 다가오는 수요일은 언제인지 알아본다.

```
SELECT SYSDATE, NEXT_DAY(SYSDATE, '수요일')
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE, '수요일')
1	21/02/09	21/02/10

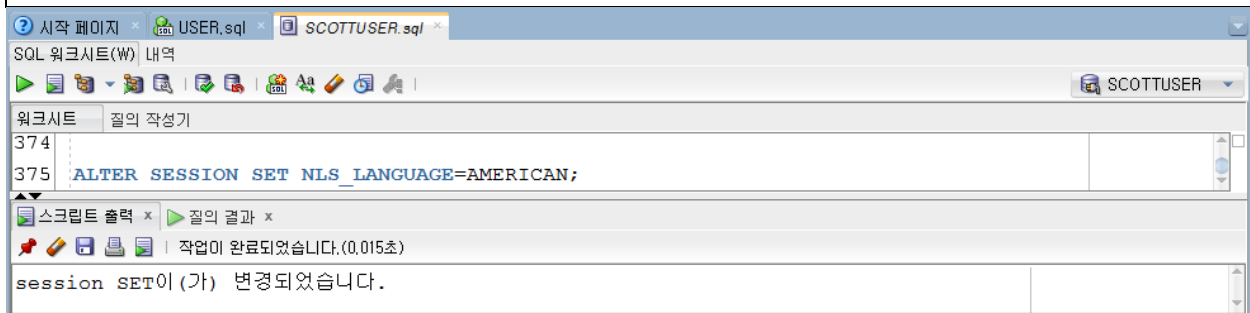
수요일 대신 '수' 로 가능하고 일요일은 1, 월요일은 2, ... 토요일은 7인 숫자로 입력할 수 있다.

```
SELECT SYSDATE, NEXT_DAY(SYSDATE, '수'), NEXT_DAY(SYSDATE, 4)
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE, '수')	NEXT_DAY(SYSDATE, 4)
1	21/02/09	21/02/10	21/02/10

요일을 영어로 사용할 경우에는 다음과 같이 언어를 AMERICAN으로 지정해야 한다.

```
ALTER SESSION SET NLS_LANGUAGE=AMERICAN;
```



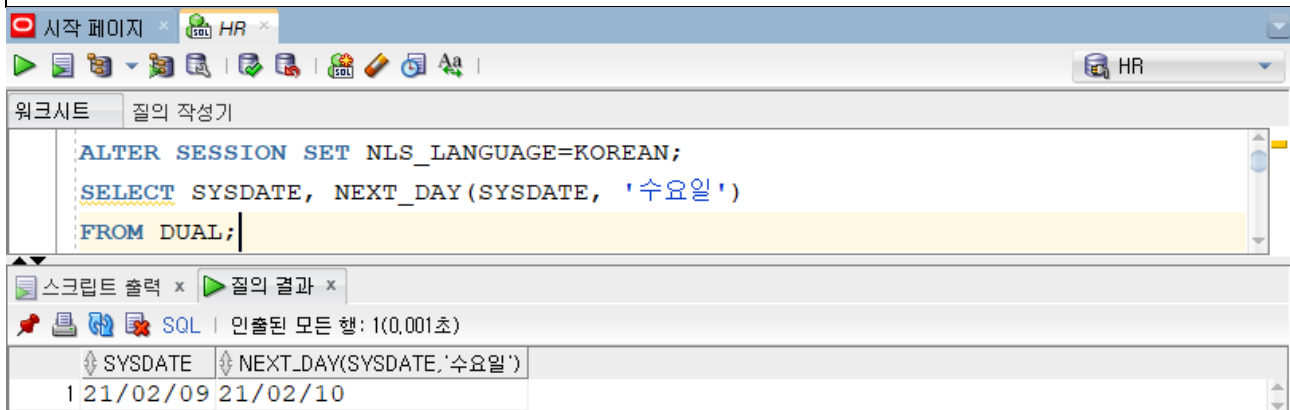
```
SELECT SYSDATE, NEXT_DAY(SYSDATE, 'WEDESDAY')
FROM DUAL;
```

	SYSDATE	NEXT_DAY(SYSDATE, 'WEDESDAY')
1	21/02/09	21/02/10

요일을 한글로 사용할 경우에는 다음과 같이 언어를 KOREAN으로 지정해야 한다.

```
ALTER SESSION SET NLS_LANGUAGE=KOREAN;
```

```
SELECT SYSDATE, NEXT_DAY(SYSDATE, '수요일')
FROM DUAL;
```



⑤ 해당 달의 마지막 날짜를 반환하는 LAST_DAY함수

SELECT SYSDATE, LAST_DAY (SYSDATE)			
FROM DUAL;			
	SYSDATE	LAST_DAY(SYSDATE)	
1	21/02/09	21/02/28	

⑥ ROUND 함수의 다양한 적용

함수에 포맷 모델(format)을 지정하면 숫자 이외의 날짜에 대해서도 반올림을 할 수 있다.

ROUND(date, format)

format	단 위	format	단 위
CC SCC	4자리 연도의 끝 두 글자를 기준으로 반올림	Q	한 분기의 두번째 달의 16일을 기준으로 반올림
YYYY/YYYY YEAR/SYEAR YYY/YY, Y	년(7월 1일부터 반올림)	MONTH MON MM RM	월(16일을 기준으로 반올림)
DDD, D, J	일을 기준	DAY, DY, D	한주가 시작되는 날짜
HH, HH12, HH24	시를 기준	MI	분을 기준

입사일을 달 기준으로 반올림한 예

SELECT HIRE_DATE, ROUND (HIRE_DATE, 'MONTH')			
FROM EMPLOYEES			
WHERE DEPARTMENT_ID = 30;			

The screenshot shows a SQL IDE window with the following content:

```

SELECT HIRE_DATE, ROUND(HIRE_DATE, 'MONTH')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
  
```

Below the query editor, the results are displayed in a table:

	HIRE_DATE	ROUND(HIRE_DATE, 'MONTH')
1	02/12/07	02/12/01
2	03/05/18	03/06/01
3	05/12/24	06/01/01
4	05/07/24	05/08/01
5	06/11/15	06/11/01
6	07/08/10	07/08/01

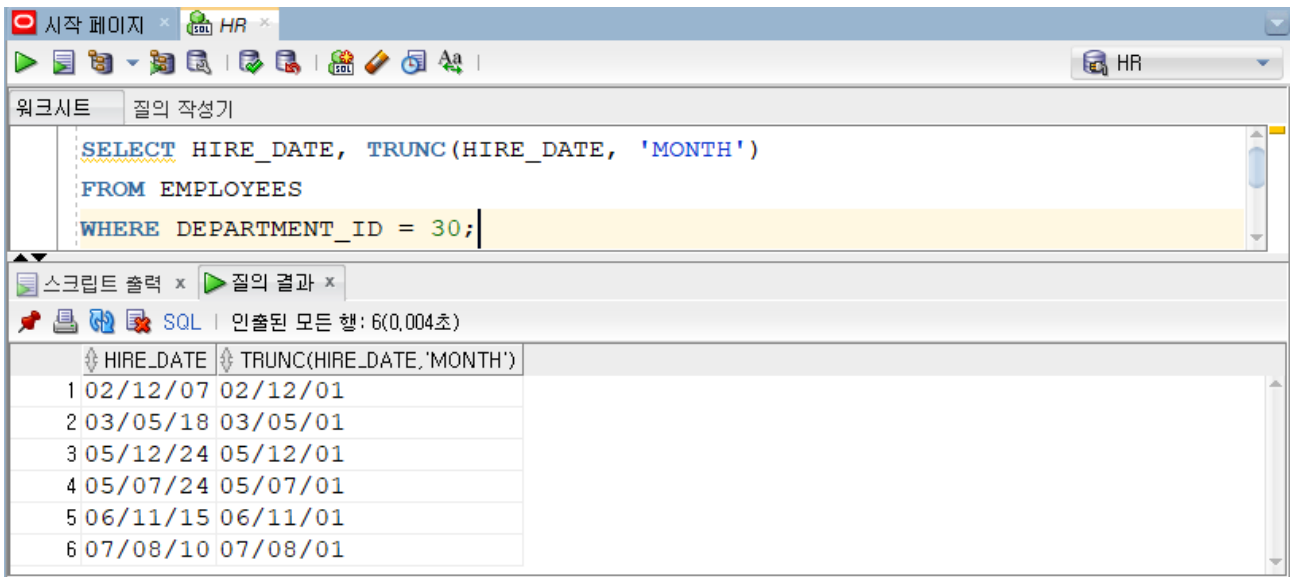
⑦ TRUNC 함수의 다양한 적용

TRUNC 함수는 숫자를 잘라내는 것뿐만 아니라 날짜도 잘라낼 수 있다.

TRUNC (date, format)

입사일을 월 기준으로 잘라내기

```
SELECT HIRE_DATE, TRUNC(HIRE_DATE, 'MONTH')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```



5) 변환 함수

자료형을 변환시키고자 할 때 사용하는 함수이다.

구 분	설 명
TO_CHAR	날짜형 혹은 숫자형을 문자형으로 변환
TO_DATE	문자형을 날짜형으로 변환
TO_NUMBER	문자형을 숫자형으로 변환



① 문자형으로 변환하는 TO_CHAR 함수

숫자나 날짜 형태의 데이터를 문자형으로 변환하는 함수

• 날짜형을 문자형으로 변환하기

TO_CHAR (날짜 데이터, '출력형식')

- 형 변환 함수의 날짜 출력 형식

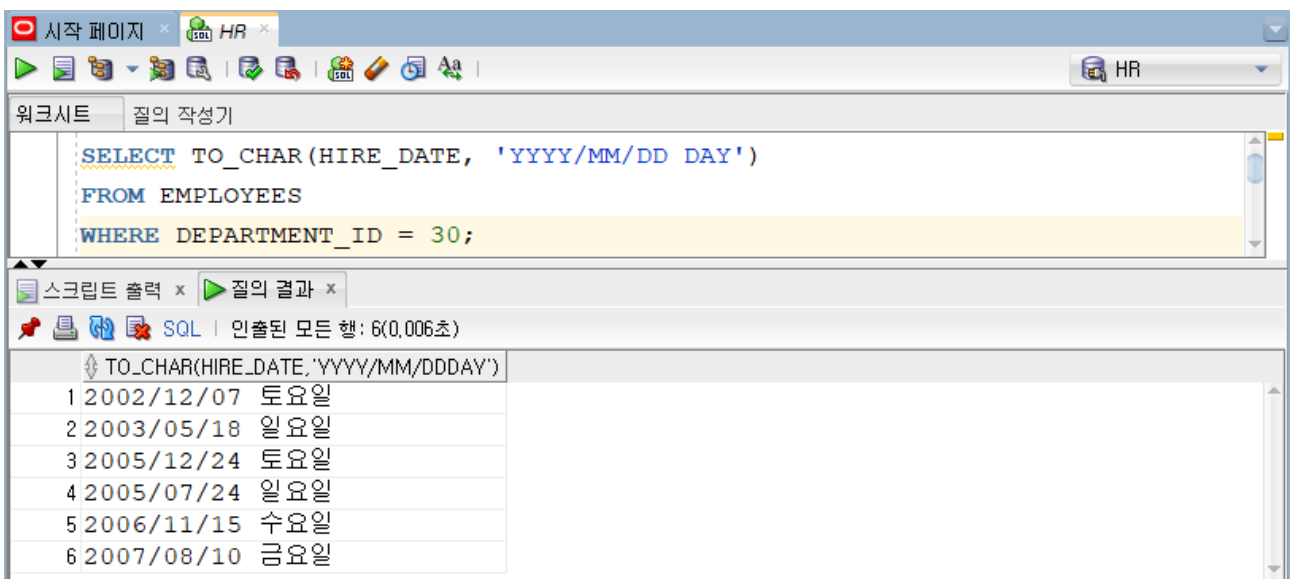
종류	의 미
YYYY	년도 표시(4자리)
YY	년도 표시(2자리)
MM	01 ~ 12 형태의 월 표시
DD	01 ~ 31 형태의 일 표시
DDD	01 ~ 365 형태의 일 표시
DAY	요일(월요일, 화요일 ...) 표시
DY	요일을 약어(월, 화 ...)로 표시
DL	현재 일을 요일까지 표시

```
SELECT SYSDATE, TO_CHAR(SYSDATE, 'YYYY-MM-DD'), TO_CHAR(SYSDATE, 'DL')
FROM DUAL;
```

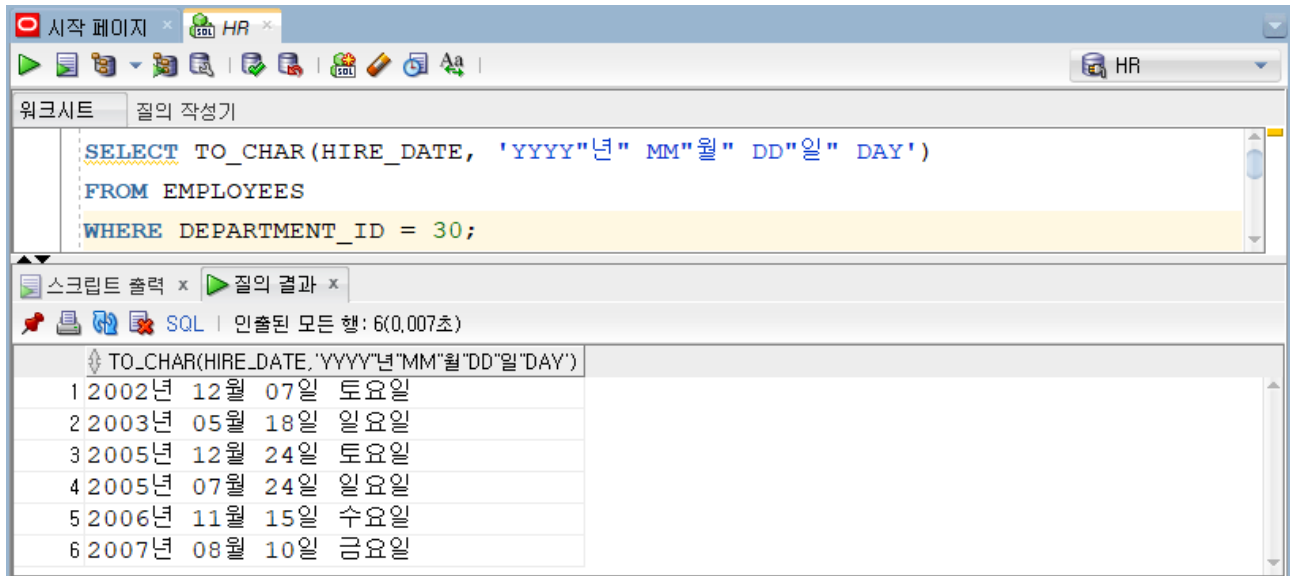
SYSDATE	TO_CHAR(SYSDATE, 'YYYY-MM-DD')	TO_CHAR(SYSDATE, 'DL')
1 21/02/09	2021-02-09	2021년 2월 9일 화요일

<예> 직원들의 입사일을 출력하되 요일까지 함께 출력하기

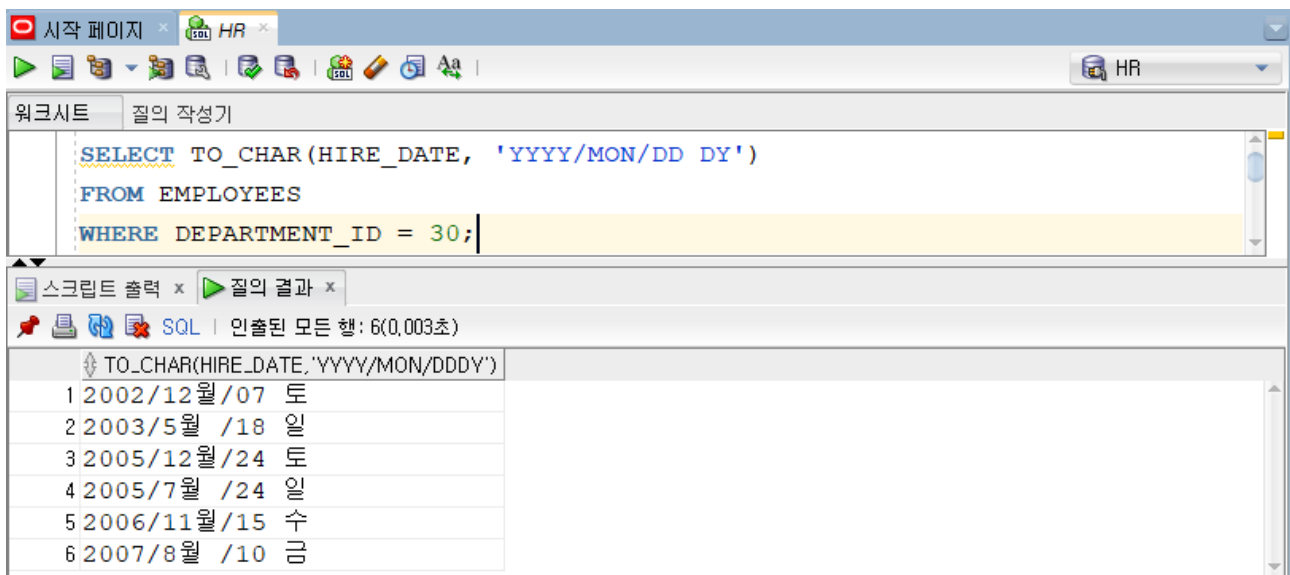
```
SELECT TO_CHAR(HIRE_DATE, 'YYYY/MM/DD DAY')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```



```
SELECT TO_CHAR(HIRE_DATE, 'YYYY"년" MM"월" DD"일" DAY')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```



```
SELECT TO_CHAR(HIRE_DATE, 'YYYY/MON/DD DY')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```



- 형변환 함수의 시간 출력 형식

종류	의 미
AM 또는 PM	오전(AM), 오후(PM) 시각 표시
A.M 또는 P.M	오전(A.M), 오후(P.M) 시각 표시

HH 또는 HH12	시간(1~12)
HH24	24시간으로 표시(0~23)
MI	분 표시
SS	초 표시

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

```
TO_CHAR(SYSDATE, 'YYYY/MM/DD, HH24:MI:SS')
1 2021/02/09, 14:52:21
```

• 숫자형을 문자형으로 변환하기

TO_CHAR(숫자, '출력형식')

- 형 변환 함수의 숫자 출력 형식

구 분	설 명
0	자릿수를 나타내며 자릿수가 맞지 않을 경우 0으로 채운다.
9	자릿수를 나타내며 자릿수가 맞지 않아도 채우지 않는다.
L	각 지역별 통화 기호를 앞에 표시한다.
.	소수점
,	천 단위 자리 구분

숫자	형 식	결 과
12345.67	999,999.999	12,345.67
12345.67	999999	12345
12345.67	\$999,999.99	\$12,345.67
12345.67	L999,999.99	₩12,345.67
12345.67	S999,999.99	+12,345.67

<예> 숫자출력

```
SELECT FIRST_NAME, SALARY, TO_CHAR(SALARY, '$999,999')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The screenshot shows the SQL Developer interface. The top pane contains the following SQL query:

```
SELECT FIRST_NAME, SALARY, TO_CHAR(SALARY, '$999,999')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The bottom pane shows the results of the query in a table format. The status bar indicates that 6 rows were returned in 0.004 seconds.

	FIRST_NAME	SALARY	TO_CHAR(SALARY, '\$999,999')
1	Den	11000	\$11,000
2	Alexander	3100	\$3,100
3	Shelli	2900	\$2,900
4	Sigal	2800	\$2,800
5	Guy	2600	\$2,600
6	Karen	2500	\$2,500

```
SELECT FIRST_NAME, SALARY, TO_CHAR(123456, '999,999,999')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The screenshot shows the SQL Developer interface. The top pane contains the following SQL query:

```
SELECT FIRST_NAME, SALARY, TO_CHAR(123456, '999,999,999')
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 30;
```

The bottom pane shows the results of the query in a table format. The status bar indicates that 6 rows were returned in 0.003 seconds.

	FIRST_NAME	SALARY	TO_CHAR(123456, '999,999,999')
1	Den	11000	123,456
2	Alexander	3100	123,456
3	Shelli	2900	123,456
4	Sigal	2800	123,456
5	Guy	2600	123,456
6	Karen	2500	123,456

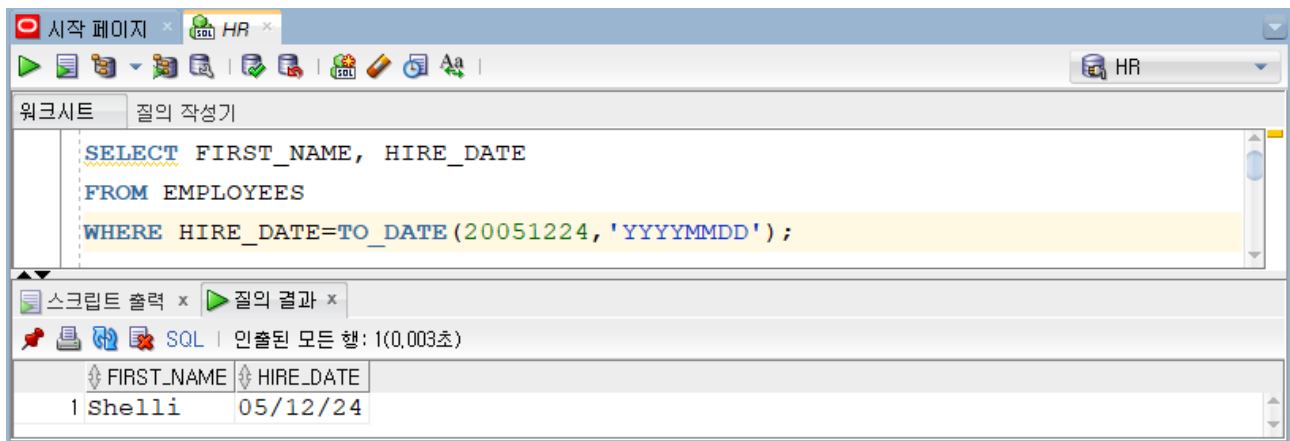
② 날짜형으로 변환하는 TO_DATE 함수

오라클에서 기본 날짜 형식은 'MM/DD/YY' 형식으로 '월/일/년' 예를 들면 '03/02/15' 식으로 나타낸다.

TO_DATE(문자, '출력형식')

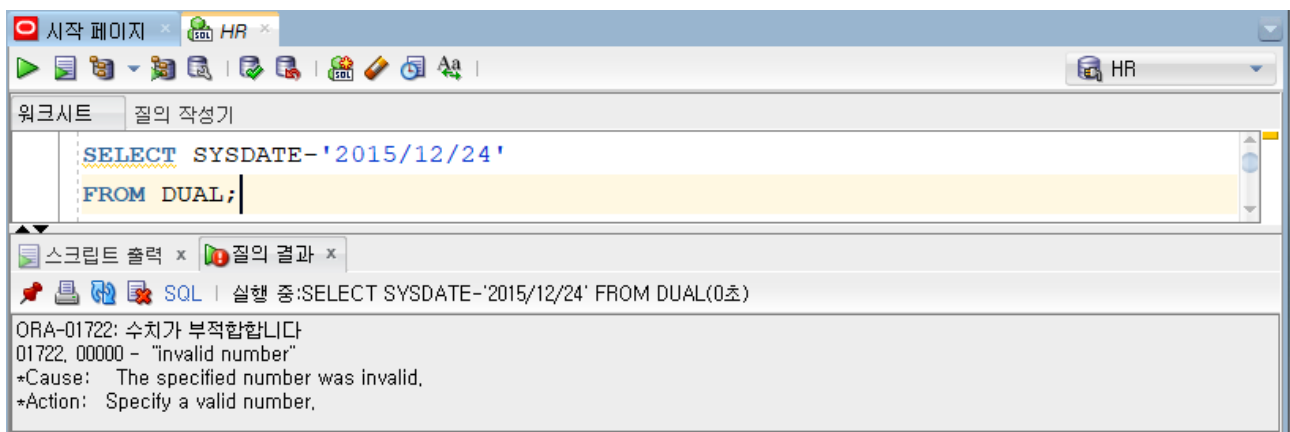
<예> 2005년 12월 24일에 입사한 직원을 검색

```
SELECT FIRST_NAME, HIRE_DATE
FROM EMPLOYEES
WHERE HIRE_DATE=TO_DATE(20051224, 'YYYYMMDD');
```



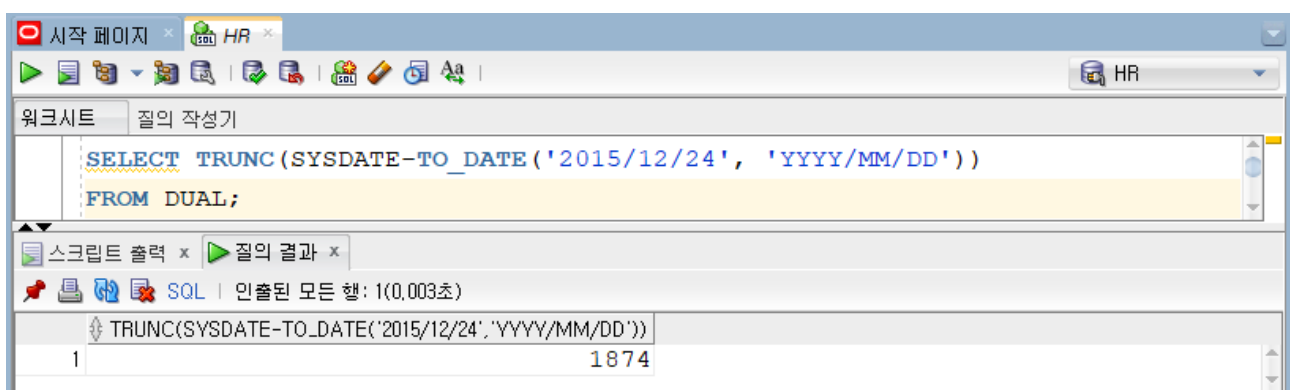
<예> 올해 며칠이 지났는지 날짜 계산 <=오류발생

```
SELECT SYSDATE-'2015/12/24'
FROM DUAL;
```



예> 올해 며칠이 지났는지 날짜 계산 <=오류해결

```
SELECT TRUNC(SYSDATE-TO_DATE('2015/12/24','YYYY/MM/DD'))
FROM DUAL;
```

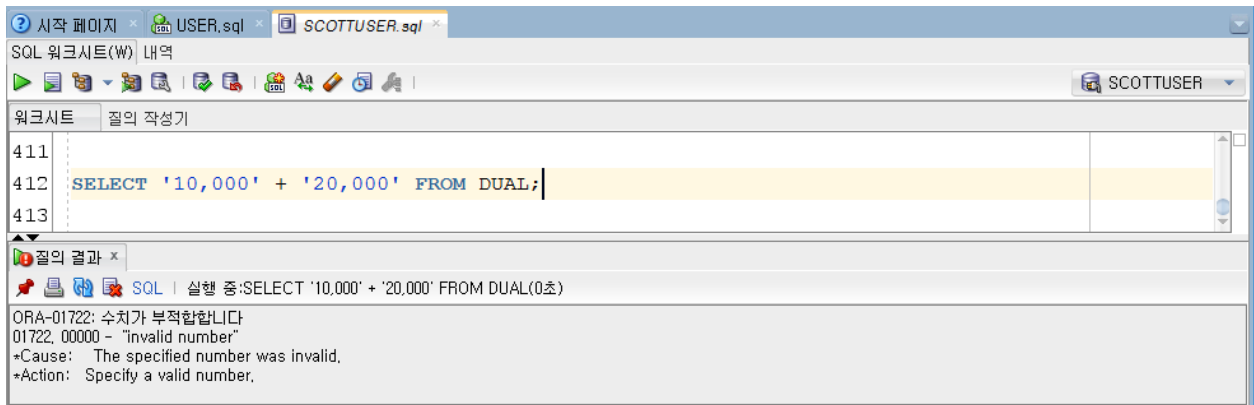


③ 숫자형으로 변환하는 TO_NUMBER 함수

TO_NUMBER ('문자', '출력형식')

<예> 수치 형태의 문자 값의 차 구하기 <=오류발생

```
SELECT '10,000' + '20,000'
FROM DUAL;
```



<예> 수치 형태의 문자 값의 차 구하기 <=오류해결

```
SELECT TO_NUMBER('10,000', '999,999') + TO_NUMBER('20,000', '999,999')
FROM DUAL;
```

	TO_NUMBER('10,000', '999,999')+TO_NUMBER('20,000', '999,999')
1	30000

6) 일반 함수

구 분	설 명
NVL	첫 번째 인자로 받은 값이 NULL과 같으면 두 번째 인자 값으로 변경
DECODE	첫 번째 인자로 받은 값을 조건에 맞춰 변경 (if와 유사)
CASE	조건에 맞는 문장을 수행한다. (switch와 유사)

NULL을 비교할 때는 IS NULL 또는 IS NOT NULL 구문을 사용하였는데, 오라클에서는 NULL을 연산 대상으로 처리하는 SQL 함수를 제공하고 있다.

① NULL을 다른 값으로 변환하는 NVL 함수

- NULL을 0 또는 다른 값으로 변환하기 위해서 사용하는 함수

NVL(컬럼 또는 표현식, 대체값)

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT, JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;
```

시작 페이지 x HR x

워크시트 | 질의 작성기

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT, JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.009초)

	FIRST_NAME	SALARY	COMMISSION_PCT	JOB_ID
1	William	8300	(null)	AC_ACCOUNT
2	Shelley	12008	(null)	AC_MGR
3	Jennifer	4400	(null)	AD_ASST
4	Steven	24000	(null)	AD_PRES
5	Lex	17000	(null)	AD_VP
6	Neena	17000	(null)	AD_VP
7	John	8200	(null)	FI_ACCOUNT
8	Daniel	9000	(null)	FI_ACCOUNT

```
SELECT FIRST_NAME, SALARY, NVL(COMMISSION_PCT, 0), JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;
```

시작 페이지 x HR x

워크시트 | 질의 작성기

```
SELECT FIRST_NAME, SALARY, NVL(COMMISSION_PCT, 0), JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.013초)

	FIRST_NAME	SALARY	NVL(COMMISSION_PCT,0)	JOB_ID
1	William	8300	0	AC_ACCOUNT
2	Shelley	12008	0	AC_MGR
3	Jennifer	4400	0	AD_ASST
4	Steven	24000	0	AD_PRES
5	Lex	17000	0	AD_VP
6	Neena	17000	0	AD_VP
7	John	8200	0	FI_ACCOUNT
8	Daniel	9000	0	FI_ACCOUNT

<예> 급여에 커미션을 더한 금액 구하기

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT, SALARY*COMMISSION_PCT AS COMMISSION,
SALARY+(SALARY*COMMISSION_PCT) AS TOTAL, JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;
```

시작 페이지 PDB_HR.sql

SQL 워크시트(W) 내역

워크시트 질의 작성기

```

SELECT FIRST_NAME, SALARY, COMMISSION_PCT, SALARY*COMMISSION_PCT AS COMMISSION,
SALARY+(SALARY*COMMISSION_PCT) AS TOTAL, JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;

```

질의 결과 x

SQL | 50개의 행이 인출됨(0.006초)

	FIRST_NAME	SALARY	COMMISSION_PCT	COMMISSION	TOTAL	JOB_ID
1	William	8300	(null)	(null)	(null)	AC_ACCOUNT
2	Shelley	12008	(null)	(null)	(null)	AC_MGR
3	Jennifer	4400	(null)	(null)	(null)	AD_ASST
4	Steven	24000	(null)	(null)	(null)	AD_PRES
5	Lex	17000	(null)	(null)	(null)	AD_VP
6	Neena	17000	(null)	(null)	(null)	AD_VP
7	John	8200	(null)	(null)	(null)	FI_ACCOUNT
8	Daniel	9000	(null)	(null)	(null)	FI_ACCOUNT
9	Luis	6900	(null)	(null)	(null)	FI_ACCOUNT
10	Ismael	7700	(null)	(null)	(null)	FI_ACCOUNT
11	Jose Manuel	7800	(null)	(null)	(null)	FI_ACCOUNT
12	Nancy	12008	(null)	(null)	(null)	FI_MGR
13	Susan	6500	(null)	(null)	(null)	HR_REP

```

SELECT FIRST_NAME, SALARY, SALARY*12, SALARY*12+NVL(COMMISSION_PCT, 0)
FROM EMPLOYEES
ORDER BY JOB_ID;

```

시작 페이지 PDB_HR.sql

SQL 워크시트(W) 내역

워크시트 질의 작성기

```

SELECT FIRST_NAME, SALARY, COMMISSION_PCT, SALARY*NVL(COMMISSION_PCT, 0) AS COMMISSION,
SALARY+(SALARY*NVL(COMMISSION_PCT, 0)) AS TOTAL, JOB_ID
FROM EMPLOYEES
ORDER BY JOB_ID;

```

질의 결과 x

SQL | 50개의 행이 인출됨(0.003초)

	FIRST_NAME	SALARY	COMMISSION_PCT	COMMISSION	TOTAL	JOB_ID
1	William	8300	(null)	0	8300	AC_ACCOUNT
2	Shelley	12008	(null)	0	12008	AC_MGR
3	Jennifer	4400	(null)	0	4400	AD_ASST
4	Steven	24000	(null)	0	24000	AD_PRES
5	Lex	17000	(null)	0	17000	AD_VP
6	Neena	17000	(null)	0	17000	AD_VP
7	John	8200	(null)	0	8200	FI_ACCOUNT
8	Daniel	9000	(null)	0	9000	FI_ACCOUNT
9	Luis	6900	(null)	0	6900	FI_ACCOUNT
10	Ismael	7700	(null)	0	7700	FI_ACCOUNT
11	Jose Manuel	7800	(null)	0	7800	FI_ACCOUNT
12	Nancy	12008	(null)	0	12008	FI_MGR
13	Susan	6500	(null)	0	6500	HR_REP

- NVL2 함수

NVL2(컬럼 또는 표현식, NULL값이 아니면 처리할 구문, NULL값이면 처리할 구문)

<예> 커미션이 NULL이 아니면 급여+커미션을, NULL이면 급여만 출력

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT,
NVL2(COMMISSION_PCT, SALARY+(SALARY*NVL(COMMISSION_PCT, 0)), SALARY) TOTAL_SAL
FROM EMPLOYEES;
```

The screenshot shows the SQL Developer interface with a query window titled 'PDB_HR.sql'. The query is:


```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT,
NVL2(COMMISSION_PCT, SALARY+(SALARY*NVL(COMMISSION_PCT, 0)), SALARY) TOTAL_SAL
FROM EMPLOYEES;
```

 The results window shows 15 rows of data. The columns are FIRST_NAME, SALARY, COMMISSION_PCT, and TOTAL_SAL. The COMMISSION_PCT column contains NULL for all rows, and the TOTAL_SAL column contains the SALARY value for all rows.

	FIRST_NAME	SALARY	COMMISSION_PCT	TOTAL_SAL
1	Steven	24000	(null)	24000
2	Neena	17000	(null)	17000
3	Lex	17000	(null)	17000
4	Alexander	9000	(null)	9000
5	Bruce	6000	(null)	6000
6	David	4800	(null)	4800
7	Valli	4800	(null)	4800
8	Diana	4200	(null)	4200
9	Nancy	12008	(null)	12008
10	Daniel	9000	(null)	9000
11	John	8200	(null)	8200
12	Ismael	7700	(null)	7700
13	Jose Manuel	7800	(null)	7800
14	Luis	6900	(null)	6900
15	Den	11000	(null)	11000

- NULLIF(표현식1, 표현식2) 두 표현식을 비교하여 동일한 경우에는 NULL을 반환하고, 동일하지 않으면 첫 번째 표현식을 반환한다.

```
SELECT NULLIF('A', 'A'), NULLIF('A', 'B')
FROM DUAL;
```

The screenshot shows the SQL Developer interface with a query window titled 'HR'. The query is:


```
SELECT NULLIF('A', 'A'), NULLIF('A', 'B')
FROM DUAL;
```

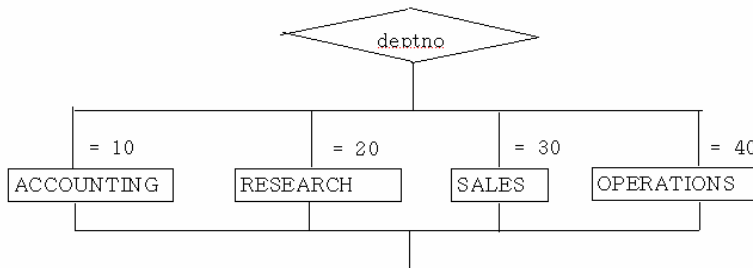
 The results window shows 1 row of data. The columns are NULLIF('A','A') and NULLIF('A','B'). The first column contains NULL and the second column contains 'A'.

	NULLIF('A','A')	NULLIF('A','B')
1	(null)	A

<문제> 모든 직원은 자신의 상관(MANAGER_ID)이 있다. 하지만 EMPLOYEES 테이블에 유일하게 상관이 없는 로우가 있는데 그 사원의 MANAGER_ID 칼럼 값이 NULL이다. 상관이 없는 대표이사만 출력하되 MANAGER_ID 칼럼 값 NULL 대신 CEO로 출력한다.

② 선택을 위한 DECODE 함수

SWITCH CASE 문과 같이 여러 가지 경우에 대해서 선택할 수 있는 함수



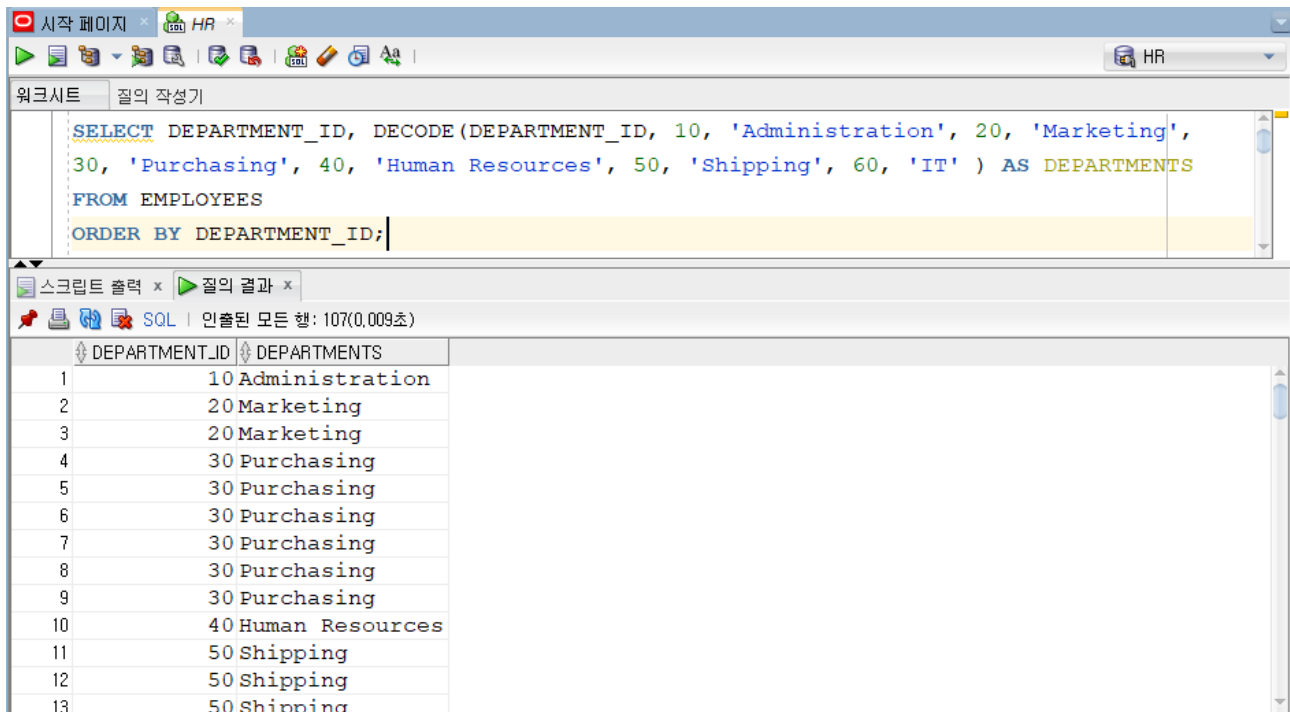
DECODE (표현식, 조건1, 결과1, 조건2, 결과2, 조건3, 결과3, ... 조건절외나머지결과n)

<예> 부서명 구하기

```
SELECT *
FROM DEPARTMENTS;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	30 Purchasing	114	1700
4	40 Human Resources	203	2400
5	50 Shipping	121	1500
6	60 IT	103	1400
7	70 Public Relations	204	2700
8	80 Sales	145	2500
9	90 Executive	100	1700
10	100 Finance	108	1700
11	110 Accounting	205	1700
12	120 Treasury	(null)	1700
13	130 Corporate Tax	(null)	1700
14	140 Control And Credit	(null)	1700
15	150 Shareholder Services	(null)	1700
16	160 Benefits	(null)	1700
17	170 Manufacturing	(null)	1700
18	180 Construction	(null)	1700
19	190 Contracting	(null)	1700
20	200 Operations	(null)	1700
21	210 IT Support	(null)	1700
22	220 NOC	(null)	1700
23	230 IT Helpdesk	(null)	1700
24	240 Government Sales	(null)	1700
25	250 Retail Sales	(null)	1700
26	260 Recruiting	(null)	1700
27	270 Payroll	(null)	1700

```
SELECT DEPARTMENT_ID, DECODE(DEPARTMENT_ID, 10, 'Administration', 20, 'Marketing',
30, 'Purchasing', 40, 'Human Resources', 50, 'Shipping', 60, 'IT' ) AS DEPARTMENTS
FROM EMPLOYEES
ORDER BY DEPARTMENT_ID;
```



③ 조건에 따라 서로 다른 처리가 가능한 CASE 함수

```
CASE WHEN 조건1 THEN 결과1
      WHEN 조건2 THEN 결과2
      WHEN 조건3 THEN 결과3
      ELSE 결과n
END
```

<예> 부서명 구하기

```
SELECT FIRST_NAME, DEPARTMENT_ID,
      CASE WHEN DEPARTMENT_ID=10 THEN 'Administration'
      WHEN DEPARTMENT_ID=20 THEN 'Marketing'
      WHEN DEPARTMENT_ID=30 THEN 'Purchasing'
      WHEN DEPARTMENT_ID=40 THEN 'Human Resources'
      WHEN DEPARTMENT_ID=50 THEN 'Shipping'
      WHEN DEPARTMENT_ID=60 THEN 'IT'
      END DEPARTMENT_NAME
FROM EMPLOYEES
ORDER BY DEPARTMENT_ID;
```

워크시트 | 질의 작성기

```

SELECT FIRST_NAME, DEPARTMENT_ID,
       CASE WHEN DEPARTMENT_ID=10 THEN 'Administration'
            WHEN DEPARTMENT_ID=20 THEN 'Marketing'
            WHEN DEPARTMENT_ID=30 THEN 'Purchasing'
            WHEN DEPARTMENT_ID=40 THEN 'Human Resources'
            WHEN DEPARTMENT_ID=50 THEN 'Shipping'
            WHEN DEPARTMENT_ID=60 THEN 'IT'
       END DEPARTMENT_NAME
FROM EMPLOYEES
ORDER BY DEPARTMENT_ID;

```

스크립트 출력 | 질의 결과

SQL | 50개의 행이 인출됨(0.011초)

	FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Jennifer	10	Administration
2	Michael	20	Marketing
3	Pat	20	Marketing
4	Den	30	Purchasing
5	Alexander	30	Purchasing
6	Shelli	30	Purchasing
7	Sigal	30	Purchasing
8	Guy	30	Purchasing
9	Karen	30	Purchasing
10	Susan	40	Human Resources
11	Matthew	50	Shipping
12	Adam	50	Shipping
13	Payam	50	Shipping
14	Shanta	50	Shipping
15	Kevin	50	Shipping
16	Julia	50	Shipping
17	Irene	50	Shipping
18	James	50	Shipping
19	Steven	50	Shipping
20	Laura	50	Shipping
21	Mozhe	50	Shipping

<문제> 부서별로 따라 급여를 인상하도록 하자. (직원번호, 직원명, 직급, 급여)

부서명이 'Marketing'인 직원은 5%, 'Purchasing'인 사원은 10%, 'Human Resources'인 사원은 15%, 'IT'인 직원은 20%인 인상한다.

④ GREATEST(exp1, exp2 ...), LEAST(exp1, exp2 ...)

GREATEST()는 매개변수로 표현식을 명시하고 가장 큰값을 반환하는 함수, LEAST()는 매개변수로 표현식을 명시하고 가장 작은 값을 반환하는 함수이다.

```

SELECT GREATEST(1,4,2,5,3,9) , LEAST(1,4,2,5,3,9)
FROM DUAL;

```

	GREATEST(1,4,2,5,3,9)	LEAST(1,4,2,5,3,9)
1	9	1

```
SELECT GREATEST('김희수','조현수','홍길동') , LEAST('김희수','조현수','홍길동')  
FROM DUAL;
```

	GREATEST('김희수','조현수','홍길동')	LEAST('김희수','조현수','홍길동')
1	홍길동	김희수