

### 03. SELECT 문으로 특정 데이터를 추출하기

데이터베이스로부터 정보를 검색할 수 있는 SELECT 명령어의 기본 구조를 학습한다.

특정 칼럼 내용만을 출력하는 방법을 학습한다.

중복된 데이터를 한 번씩만 출력하게 하는 DISTINCT에 대해서 학습한다.

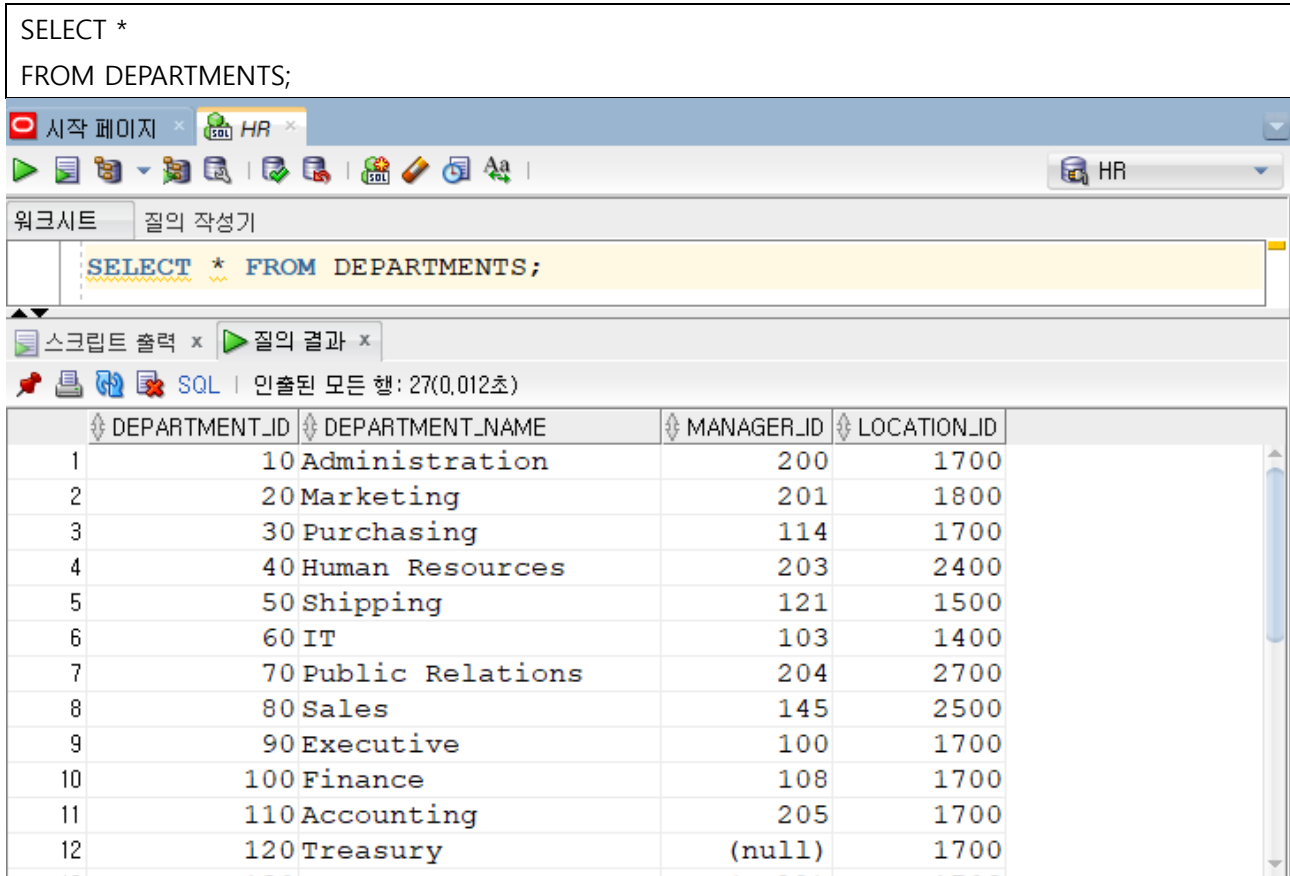
조건을 부여해서 특정 로우만 조회하는 방법을 학습한다.

특정 칼럼을 기준으로 내림차순 혹은 오름차순으로 출력하는 방법을 학습한다.

#### 1) 데이터를 조회하기 위한 SELECT

```
SELECT [DISTINCT] {*, column[Alias], . . .}  
FROM 테이블명;
```

<예> DEPARTMENTS 테이블의 모든 내용 출력



```
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

<문제> EMPLOYEES 테이블의 모든 내용 출력

#### 2) 칼럼 이름을 명시해서 특정 칼럼만 보기

<예> DEPARTMENTS 테이블에서 부서번호와 부서명 만 출력

```
SELECT DEPARTMENT_ID, DEPARTMENT_NAME  
FROM DEPARTMENTS;
```

The screenshot shows the SQL Developer interface. The top toolbar includes icons for running queries, saving, and other database functions. The main window is titled '시작 페이지' and 'HR'. Below the toolbar, there's a '워크시트' (Worksheet) tab and a '질의 작성기' (Query Editor) tab. The query editor contains the following SQL statement:

```
SELECT DEPARTMENT_ID, DEPARTMENT_NAME FROM DEPARTMENTS;
```

Below the query editor, there's a '스크립트 출력' (Script Output) tab and a '질의 결과' (Query Results) tab. The '질의 결과' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'DEPARTMENT\_ID' and 'DEPARTMENT\_NAME'. The table contains 10 rows of data:

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance

<문제> 사원의 이름과 급여와 입사일자 만을 출력하는 SQL 문을 작성해보자.

힌트 : 사원 정보가 저장된 테이블은 EMPLOYEES이고, 사원이름 칼럼은 FIRST\_NAME, LAST\_NAME과, 급여 칼럼은 SALARY, 입사일자 칼럼은 HIRE\_DATE이다.

### 3) 칼럼 이름에 별칭 지정하기

- **AS로 컬럼에 별칭 부여하기** : 컬럼을 기술한 바로 뒤에 AS라는 키워드를 쓴 후 별칭을 기술

```
SELECT DEPARTMENT_ID AS DepartmentNo, DEPARTMENT_NAME AS DepartmentName
FROM DEPARTMENTS;
```

The screenshot shows the SQL Developer interface. The top toolbar includes icons for running queries, saving, and other database functions. The main window is titled '시작 페이지' and 'HR'. Below the toolbar, there's a '워크시트' (Worksheet) tab and a '질의 작성기' (Query Editor) tab. The query editor contains the following SQL statement:

```
SELECT DEPARTMENT_ID AS DepartmentNo, DEPARTMENT_NAME AS DepartmentName
FROM DEPARTMENTS;
```

Below the query editor, there's a '스크립트 출력' (Script Output) tab and a '질의 결과' (Query Results) tab. The '질의 결과' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'DEPARTMENTNO' and 'DEPARTMENTNAME'. The table contains 9 rows of data:

DEPARTMENTNO	DEPARTMENTNAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive

- 별칭에 공백문자나 \$, \_ #등 특수 문자를 표현하고 싶거나 대소문자를 구별하고 싶으면 ""을 사용

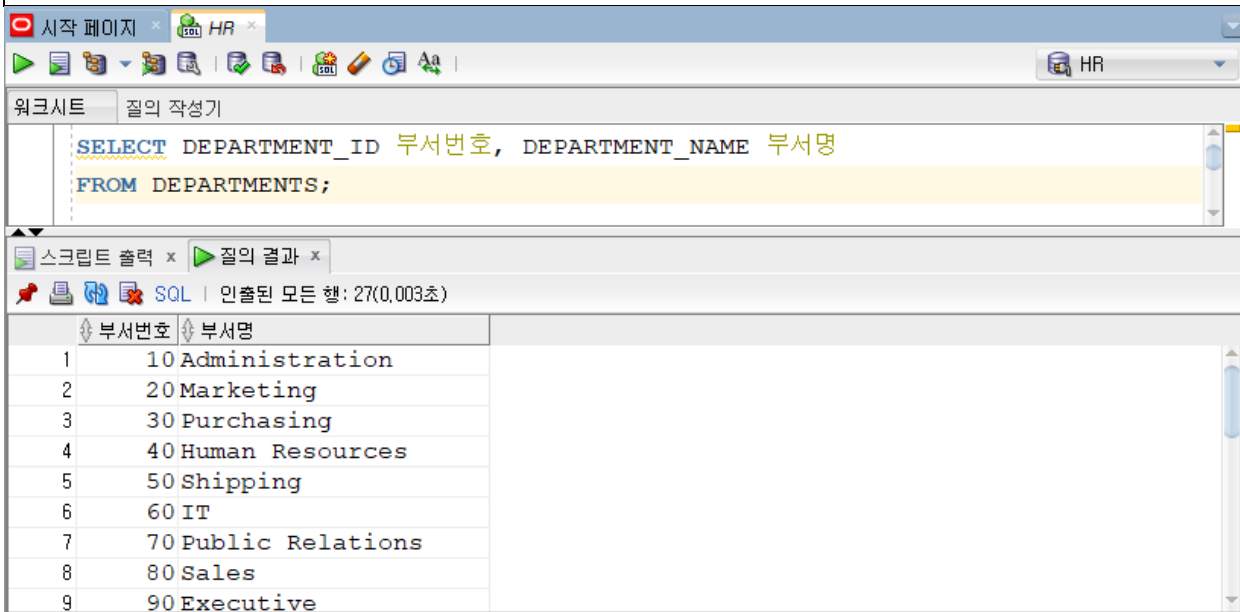
한다. AS를 생략하고 “ ”를 사용하여 별칭부여가 가능하다.

- AS 없이 컬럼에 별칭 부여하기

```
SELECT DEPARTMENT_ID "Department No", DEPARTMENT_NAME "Department Name"
FROM DEPARTMENTS;
```

- 별칭으로 한글 사용이 가능하다.

```
SELECT DEPARTMENT_ID 부서번호, DEPARTMENT_NAME 부서명
FROM DEPARTMENTS;
```



The screenshot shows the SQL Developer interface. The query editor contains the SQL statement: `SELECT DEPARTMENT_ID 부서번호, DEPARTMENT_NAME 부서명 FROM DEPARTMENTS;`. The results grid displays the following data:

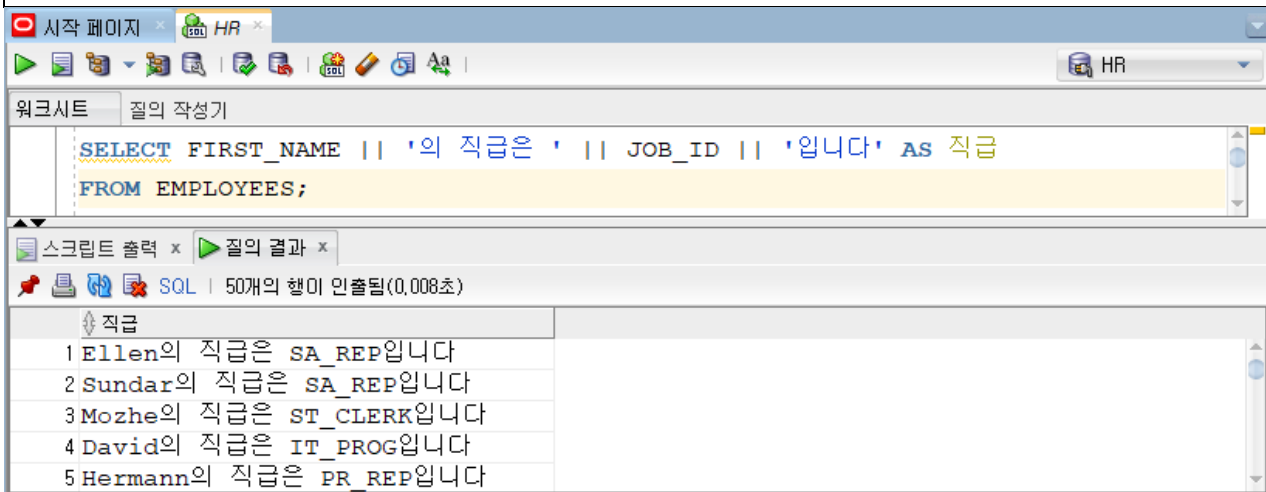
부서번호	부서명
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive

#### 4) Concatenation 연산자의 정의와 사용(연결 연산자)

오라클에서는 여러 개의 컬럼을 연결할 때 사용하기 위해서 Concatenation 연산자를 제공해 준다. 컬럼과 문자열 사이에 “||”(Concatenation 연산자)를 기술하여 하나로 연결하여 출력하게 된다.

<예> EMPLOYEES 테이블에서 여러 컬럼을 하나의 문자열로 출력하기

```
SELECT FIRST_NAME || '의 직급은 ' || JOB_ID || '입니다' AS 직급
FROM EMPLOYEES;
```



The screenshot shows the SQL Developer interface. The query editor contains the SQL statement: `SELECT FIRST_NAME || '의 직급은 ' || JOB_ID || '입니다' AS 직급 FROM EMPLOYEES;`. The results grid displays the following data:

직급
1 Ellen의 직급은 SA_REP입니다
2 Sundar의 직급은 SA_REP입니다
3 Mozhe의 직급은 ST_CLERK입니다
4 David의 직급은 IT_PROG입니다
5 Hermann의 직급은 PR_REP입니다

## 5) 중복된 데이터를 한번씩 만 출력하게 하는 DISTINCT

```
SELECT JOB_ID FROM EMPLOYEES;
```

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	AD_VP
7	FI_ACCOUNT
8	FI_ACCOUNT
9	FI_ACCOUNT
10	FI_ACCOUNT
11	FI_ACCOUNT
12	FI_MGR
13	HR_REP
14	IT_PROG
15	IT_PROG

<예> EMPLOYEES 테이블에서 칼럼 JOB\_ID를 표시하되 중복된 값은 한번만 표시하라.

```
SELECT DISTINCT JOB_ID
FROM EMPLOYEES;
```

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	FI_ACCOUNT
7	FI_MGR
8	HR_REP
9	IT_PROG
10	MK_MAN

<문제> 직원들이 어떤 부서에 소속되어 있는지 소속 부서번호(DEPARTMENT\_ID) 출력하되 중복되지 않고 한번씩 출력하는 쿼리문을 작성하자.

## 6) WHERE 조건과 비교 연산자

```
SELECT [DISTINCT] {*, column[Alias], . . .}
FROM 테이블명
WHERE 조건들;
```

- 두 쿼리문을 비교하자.

<예> 전체 직원을 대상

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES;
```

시작 페이지 x HR x

워크시트 | 질의 작성기

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.007초)

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	100	Steven	24000
2	101	Neena	17000
3	102	Lex	17000
4	103	Alexander	9000
5	104	Bruce	6000
6	105	David	4800
7	106	Valli	4800
8	107	Diana	4200
9	108	Nancy	12008

<예>급여를 3000 이상 받는 직원을 대상

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY >= 3000;
```

시작 페이지 x HR x

워크시트 | 질의 작성기

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY >= 3000;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.009초)

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	100	Steven	24000
2	101	Neena	17000
3	102	Lex	17000
4	103	Alexander	9000
5	104	Bruce	6000
6	105	David	4800
7	106	Valli	4800
8	107	Diana	4200
9	108	Nancy	12008
10	109	Daniel	9000
11	110	John	8200
12	111	Ismael	7700
13	112	Jose Manuel	7800
14	113	Luis	6900
15	114	Den	11000
16	115	Alexander	3100

• 비교 연산자

연산자	의미
=	같다.
>	보다 크다.
<	보다 작다.
>=	보다 크거나 같다.
<=	보다 작거나 같다
<>, !=, ^=	같지 않다.

<예>급여를 3000 미만 받는 직원을 대상

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY < 3000;
```

The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL statement:

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY < 3000;
```

The results pane shows the output of the query, displaying 16 rows of employee data. The columns are EMPLOYEE\_ID, FIRST\_NAME, and SALARY. The data is as follows:

EMPLOYEE_ID	FIRST_NAME	SALARY
116	Shelli	2900
117	Sigal	2800
118	Guy	2600
119	Karen	2500
126	Irene	2700
127	James	2400
128	Steven	2200
130	Mozhe	2800
131	James	2500
132	TJ	2100
134	Michael	2900
135	Ki	2400
136	Hazel	2200
139	John	2700
140	Joshua	2500
143	Randall	2600

<문제> EMPLOYEES 테이블에서 부서번호가 110번인 직원에 관한 모든 정보만 출력하라.

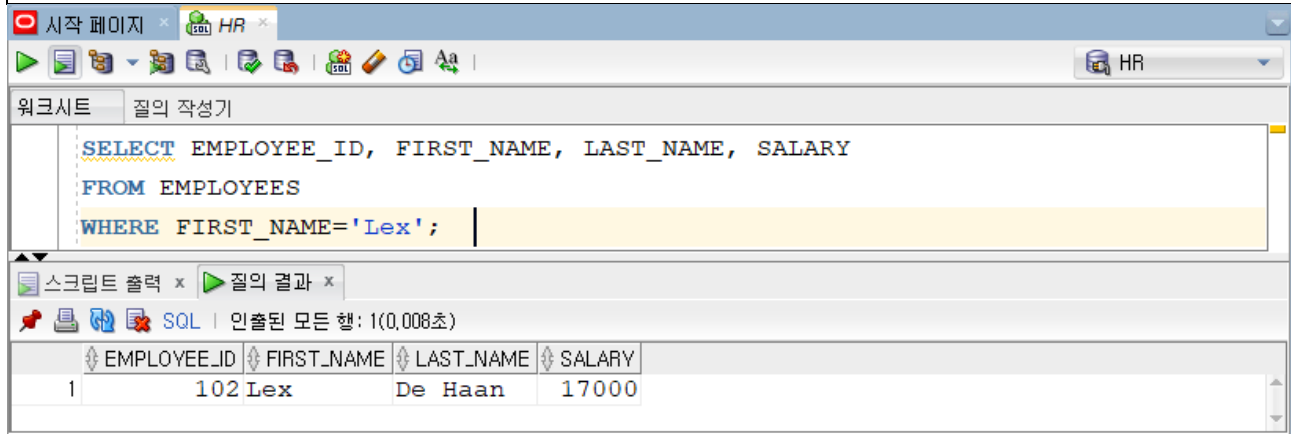
<문제> EMPLOYEES 테이블에서 급여가 5000미만이 되는 직원의 정보 중에서 사번과 이름, 급여를 출력하라.

## • 문자 데이터 조회

문자 데이터는 반드시 단일 따옴표 안에 표시한다. 대소문자를 구분한다.

<예> 이름(FIRST\_NAME)이 'Lex'인 직원

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
FROM EMPLOYEES
WHERE FIRST_NAME='Lex';
```



The screenshot shows the SQL Developer interface. The top pane contains the SQL query: `SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM EMPLOYEES WHERE FIRST_NAME='Lex';`. The bottom pane shows the execution results with a table containing one row: 

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
102	Lex	De Haan	17000

. The status bar indicates 1 row returned in 0.008 seconds.

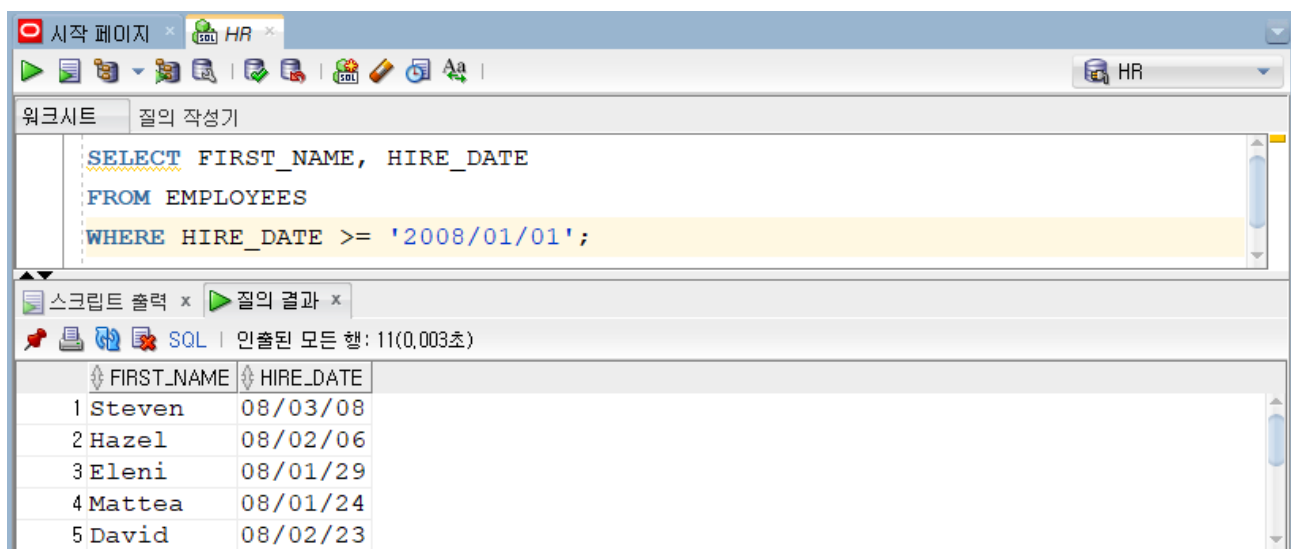
<문제> 이름이 John인 사람의 직원번호와 직원명과 직급을 출력하라.

## • 날짜 데이터 조회

반드시 단일 따옴표 안에 표시 한다. 년/월/일 형식으로 기술한다.

<예> 2008년 이후에 입사한 직원

```
SELECT FIRST_NAME, HIRE_DATE
FROM EMPLOYEES
WHERE HIRE_DATE >= '2008/01/01';
```



The screenshot shows the SQL Developer interface. The top pane contains the SQL query: `SELECT FIRST_NAME, HIRE_DATE FROM EMPLOYEES WHERE HIRE_DATE >= '2008/01/01';`. The bottom pane shows the execution results with a table containing five rows: 

FIRST_NAME	HIRE_DATE
Steven	08/03/08
Hazel	08/02/06
Eleni	08/01/29
Mattea	08/01/24
David	08/02/23

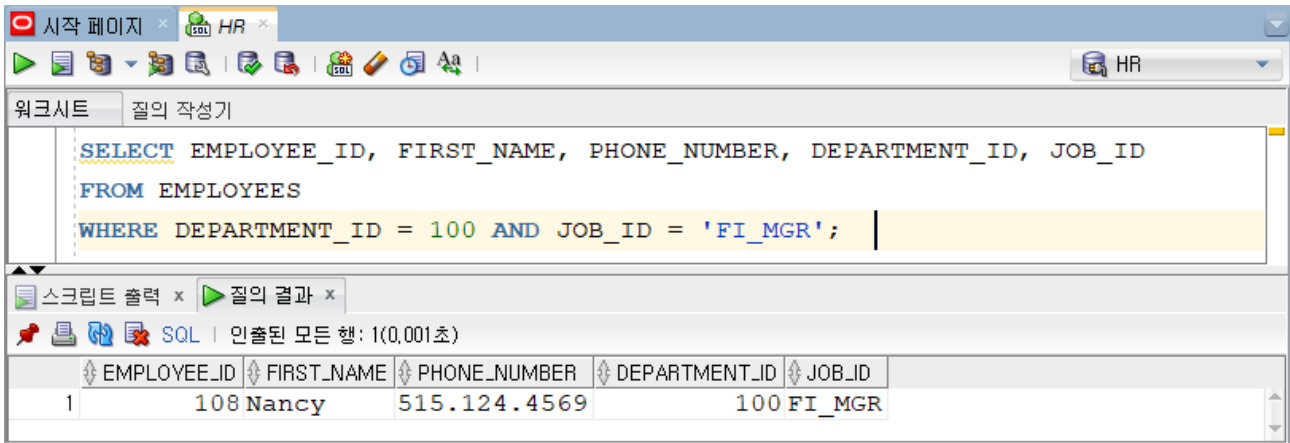
. The status bar indicates 11 rows returned in 0.003 seconds.

## 7) 논리 연산자

① AND 연산자 : 여러 조건을 모두 만족해야 할 경우 AND 연산자를 사용한다.

<예> 부서번호가 100번이고 직급이 FI\_MGR인 직원

```
SELECT EMPLOYEE_ID, FIRST_NAME, PHONE_NUMBER, DEPARTMENT_ID, JOB_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100 AND JOB_ID = 'FI_MGR';
```



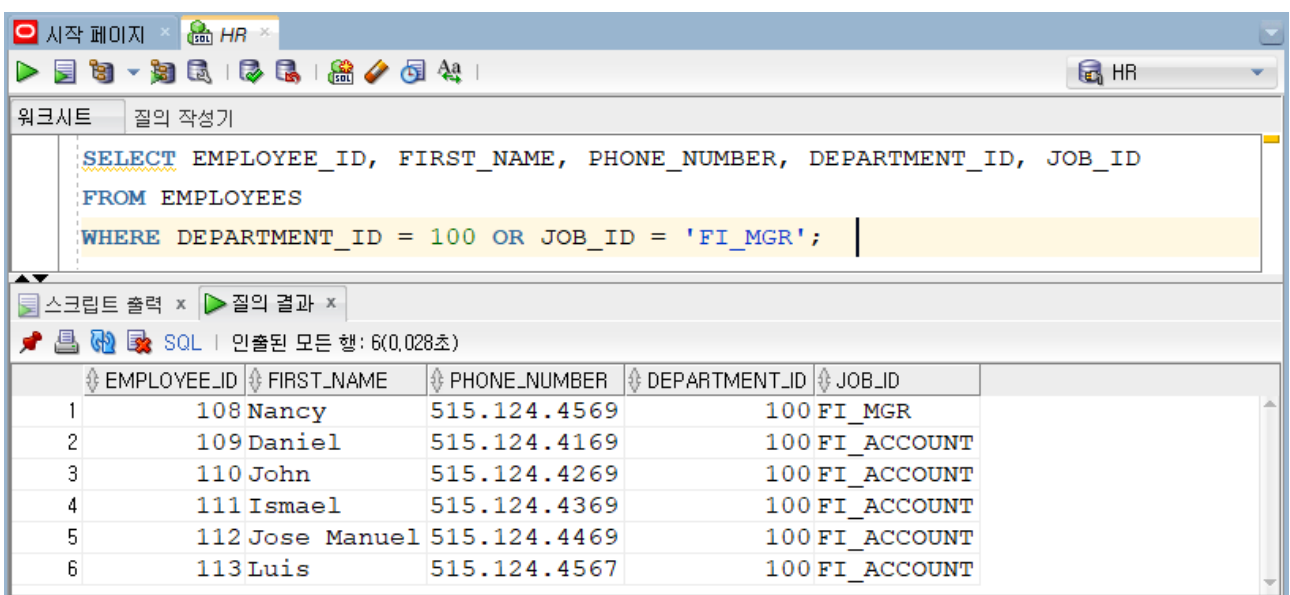
<문제>급여가 5000에서 10000이하 직원 정보 출력

② OR 연산자

: 두 가지 조건 중에서 한가지만 만족하더라도 검색할 수 있도록 하기 위해서는 OR연산자를 사용한다.

<예> 부서번호가 100번이거나 직급이 FI\_MGR인 직원

```
SELECT EMPLOYEE_ID, FIRST_NAME, PHONE_NUMBER, DEPARTMENT_ID, JOB_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100 OR JOB_ID = 'FI_MGR';
```





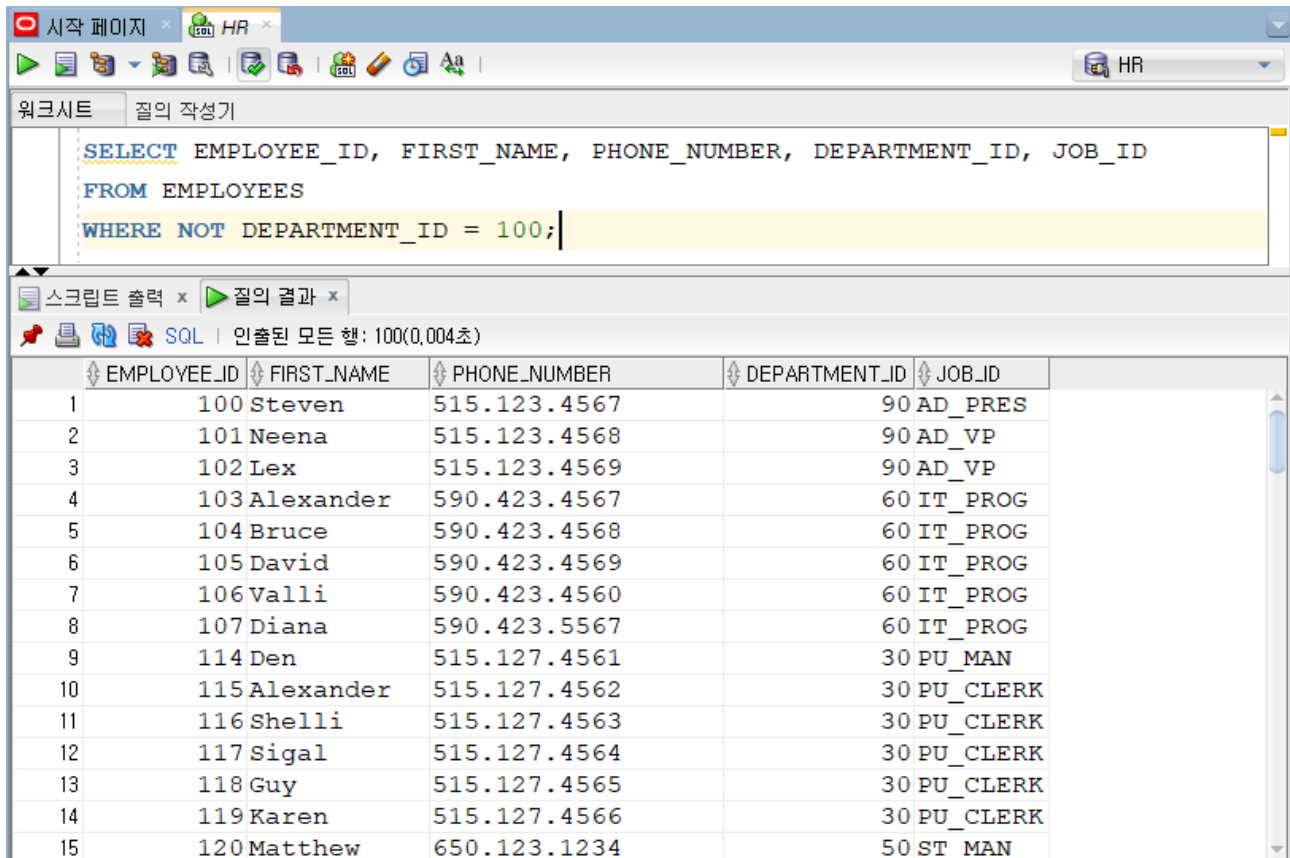
<문제> 직원번호가 134이거나 201이거나 107인 직원 정보 출력

③ NOT 연산자

반대되는 논리값을 구한다.

<예> 부서번호가 10번이 아닌 직원

```
SELECT EMPLOYEE_ID, FIRST_NAME, PHONE_NUMBER, DEPARTMENT_ID, JOB_ID
FROM EMPLOYEES
WHERE NOT DEPARTMENT_ID = 100;
```



스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 100(0,004초)

	EMPLOYEE_ID	FIRST_NAME	PHONE_NUMBER	DEPARTMENT_ID	JOB_ID
1	100	Steven	515.123.4567	90	AD_PRES
2	101	Neena	515.123.4568	90	AD_VP
3	102	Lex	515.123.4569	90	AD_VP
4	103	Alexander	590.423.4567	60	IT_PROG
5	104	Bruce	590.423.4568	60	IT_PROG
6	105	David	590.423.4569	60	IT_PROG
7	106	Valli	590.423.4560	60	IT_PROG
8	107	Diana	590.423.5567	60	IT_PROG
9	114	Den	515.127.4561	30	PU_MAN
10	115	Alexander	515.127.4562	30	PU_CLERK
11	116	Shelli	515.127.4563	30	PU_CLERK
12	117	Sigal	515.127.4564	30	PU_CLERK
13	118	Guy	515.127.4565	30	PU_CLERK
14	119	Karen	515.127.4566	30	PU_CLERK
15	120	Matthew	650.123.1234	50	ST_MAN

<문제> 직급이 FI\_MGR가 아닌 직원

④ BETWEEN AND 연산자

특정 범위 내에 속하는 데이터를 알아보려고 할 때 BETWEEN AND 연산자를 사용한다.

**column\_name BETWEEN A AND B**

<예> 급여가 1000에서부터 3000까지의 범위에 속한 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY BETWEEN 2000 AND 3000;
```

The screenshot shows the SQL Developer interface with a query window and a results window. The query is:

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY BETWEEN 2000 AND 3000;
```

The results window shows 8 rows of data:

EMPLOYEE_ID	FIRST_NAME	SALARY
116	Shelli	2900
117	Sigal	2800
118	Guy	2600
119	Karen	2500
126	Irene	2700
127	James	2400
128	Steven	2200
130	Mozhe	2800

<문제> 급여가 2500에서 4500까지의 범위에 속한 직원의 직원번호, 이름, 급여를 출력하라.  
(AND 연산자와 BETWEEN AND 연산자 사용)

#### ⑤ IN 연산자

동일한 칼럼이 여러 개의 값 중에 하나인지를 살펴보기 위해서 간단하게 표현할 수 있는 IN연산자를 사용한다.

**column\_name IN(A, B, C)**

<예> 직원번호가 67이거나 101이거나 184인 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE EMPLOYEE_ID=177 OR EMPLOYEE_ID=101 OR EMPLOYEE_ID=184;
```

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE EMPLOYEE_ID IN(177, 101, 184);
```

The screenshot shows the SQL Developer interface with a query window and a results window. The query is:

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE EMPLOYEE_ID IN(177, 101, 184);
```

The results window shows 3 rows of data:

EMPLOYEE_ID	FIRST_NAME	SALARY
101	Neena	17000
177	Jack	8400
184	Nandita	4200

<문제> 커미션비율이 0.3 이거나 0.05 이거나 0.1 중의 하나인 직원의 직원번호, 이름, 급여, 커미션 비율을 출력하라. (OR 연산자와 IN 연산자 사용)

## 8) LIKE 연산자

검색하고자 하는 값을 정확히 모를 경우 와일드카드와 함께 사용하여 원하는 내용을 검색하는 연산자.

column_name LIKE pattern
--------------------------

와일드카드	의미
%	문자가 없거나, 하나 이상의 문자가 어떤 값이 오든 상관없다.
_	하나의 문자가 어떤 값이 오든 상관없다.

### ① 와일드카드(%) 사용하기

%는 검색하고자 하는 값을 정확히 모를 경우 사용한다. %는 몇 개의 문자가 오든 상관없다는 의미

<예> K로 시작하는 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES
WHERE FIRST_NAME LIKE 'K%';
```

	EMPLOYEE_ID	FIRST_NAME
1	188	Kelly
2	119	Karen
3	197	Kevin
4	135	Ki
5	178	Kimberely
6	124	Kevin
7	146	Karen

<예> 이름 중에 k를 포함하는 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES
WHERE FIRST_NAME LIKE '%k%';
```

	EMPLOYEE...	FIRST_NAME
1	137	Renske
2	177	Jack
3	157	Patrick

<예> 이름이 k로 끝나는 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES
WHERE FIRST_NAME LIKE '%k';
```

	EMPLOYEE_ID	FIRST_NAME
1	177	Jack
2	157	Patrick

- ② 와일드카드( ) 사용하기  
 \_는 한 문자를 대신해서 사용한 것

<예> 이름의 두 번째 글자가 d인 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES
WHERE FIRST_NAME LIKE '_d%';
```

	EMPLOYEE_ID	FIRST_NAME
1	121	Adam

조건	예	설명
LIKE _A	AA, BA, CA	자료가 두 글자이되 이중 두 번째 글자가 'A'인 자료
LIKE _A%	AAA,BAA,CA213S	자료의 두 번째 글자가 'A'이고 그 뒤는 무엇이든 상관없는 자료
LIKE A__	AAA, ABC, ABF	A로 시작하고 꼭 세 글자여야만 한다. 뒤에 두 글자는 무엇이든 상관없는 자료
LIKE _a__	AaVC, Ba12	4글자로 구성되어야만 하되 두 번째 글자는 반드시 소문자 a여야 하는 자료

<문제> 이름에 a를 포함하지 않은 직원의 직원번호, 이름을 출력하라.

## 9) NULL을 위한 연산자

오라클에서는 칼럼에 NULL값이 저장되는 것을 허용한다.

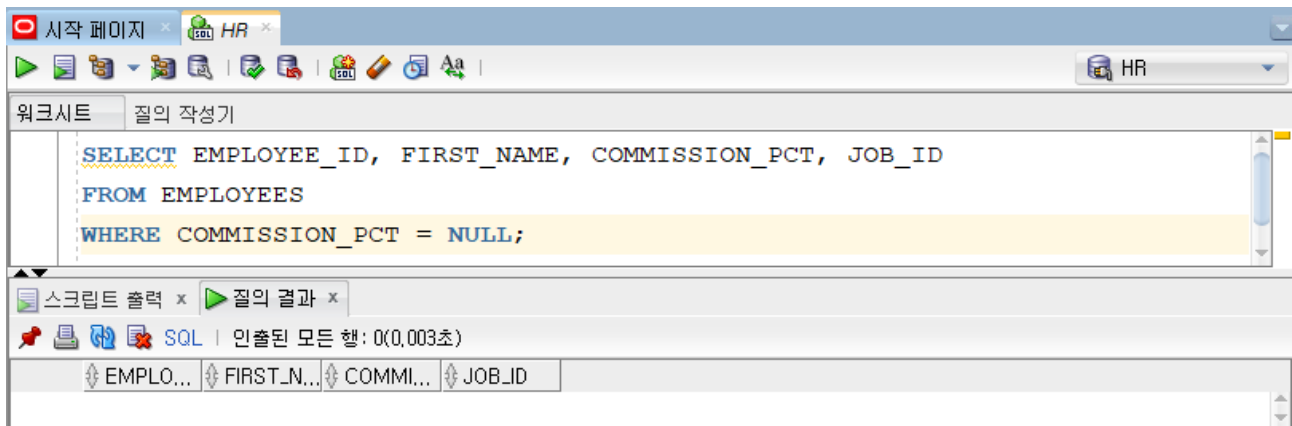
NULL은 미확정, 알 수 없는(unknown) 값을 의미한다. 0(Zero)도 빈 공간도 아닌 어떤 값이 존재하기는 하지만 어떤 값인지를 알아낼 수 없는 것을 의미한다. NULL은 연산, 할당, 비교가 불가능하다.

<예> 100 + NULL = NULL

<예> 커미션을 받지 않는 사원에 대한 검색

```
SELECT ENAME, COMM, JOB FROM EMP
WHERE COMM=NULL;
```

**NULL이 저장되어 있는 경우에는 = 연산자로 판단할 수 없다.**



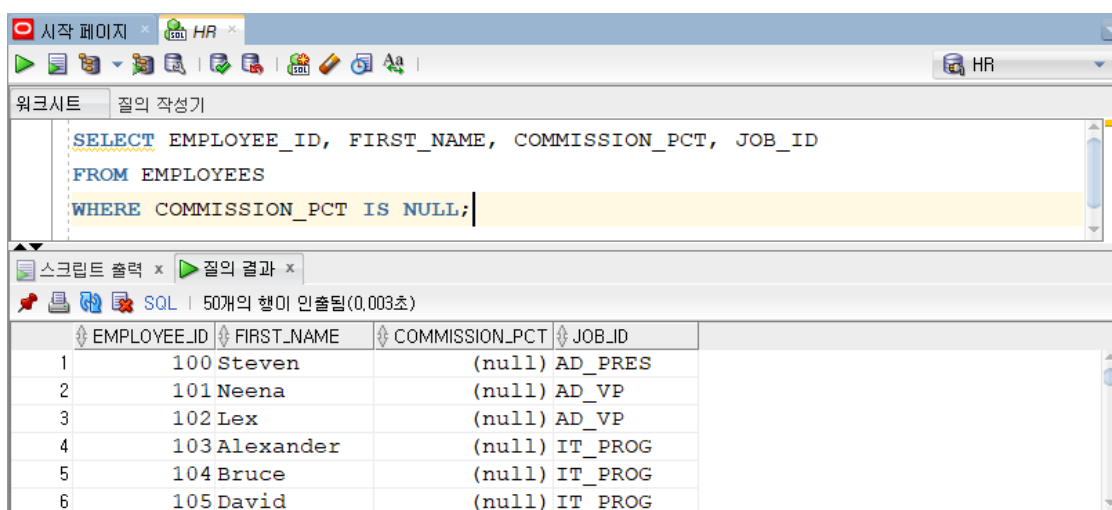
# ① IS NULL과 IS NOT NULL

특정 칼럼 값인지를 비교할 경우에는 비교연산자(=)를 사용하지 않고 IS NULL 연산자를, NULL 값이 아닌지를 알아보려면 비교연산자(<>)를 사용하지 않고 IS NOT NULL 연산자를 사용한다.

연산자	의미	비고
IS NULL	NULL이면 만족	NULL은 값이 아니므로 = 또는 !=으로 비교할 수 없다
IS NOT NULL	NULL이 아니면 만족	

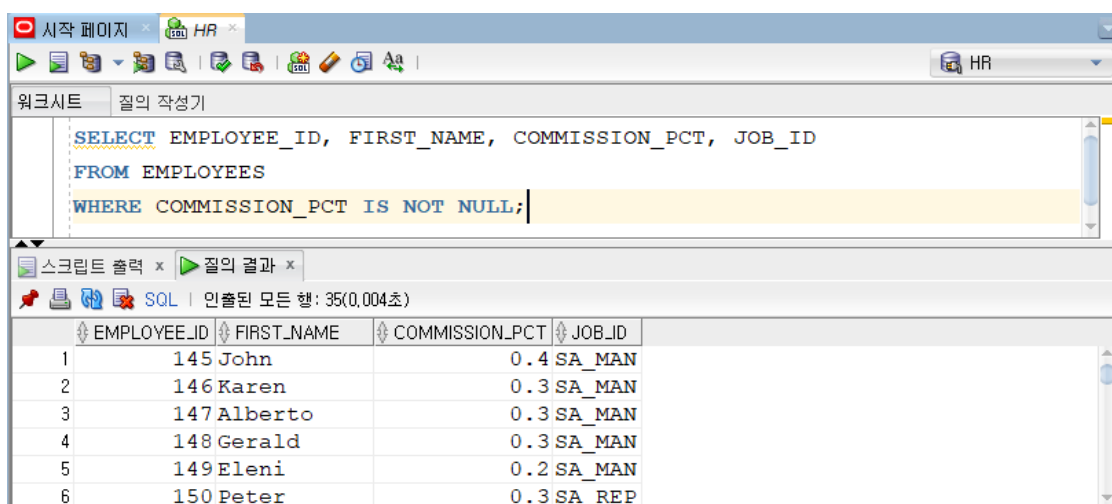
<예> 커미션을 받지 않는 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME, COMMISSION_PCT, JOB_ID
FROM EMPLOYEES
WHERE COMMISSION_PCT IS NULL;
```



<예> 커미션을 받는 사원

```
SELECT EMPLOYEE_ID, FIRST_NAME, COMMISSION_PCT, JOB_ID
FROM EMPLOYEES
WHERE COMMISSION_PCT IS NOT NULL;
```



<문제> 자신의 직속상관이 없는 직원의 전체 이름과 직급과 직원번호를 출력하라

## 10) 정렬을 위한 ORDER BY 절

ORDER BY 절은 로우(행)를 정렬하는데 사용하며 쿼리문 맨 뒤에 기술해야 하며, 정렬의 기준이 되는 칼럼 이름 또는 SELECT 절에서 명시된 별칭을 사용할 수 있다.

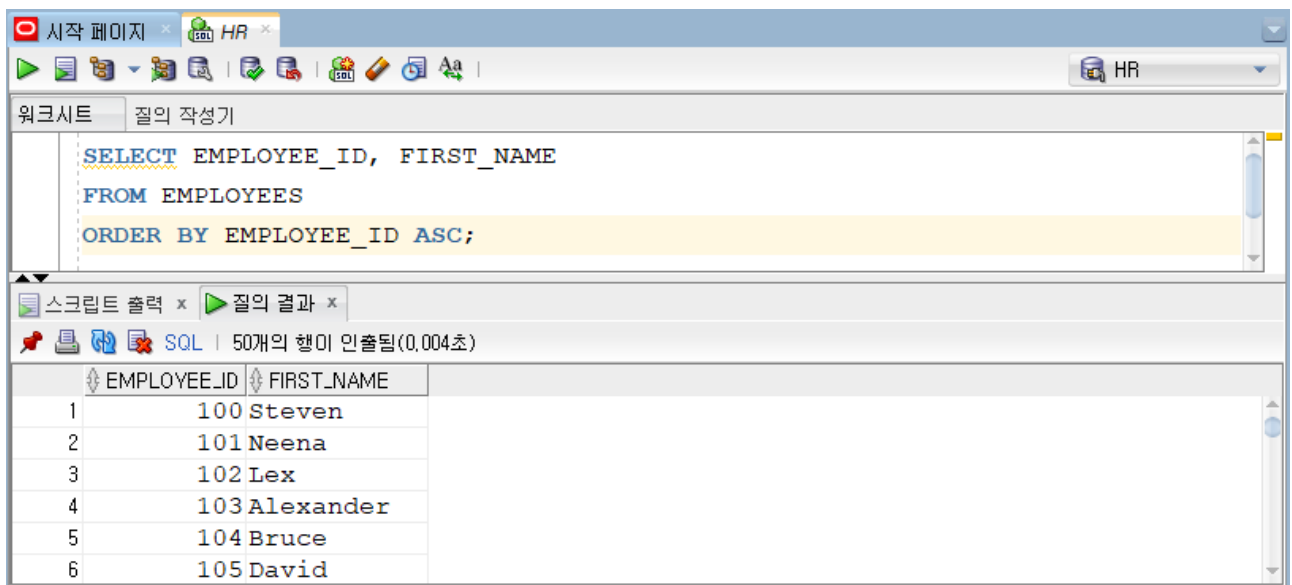
분류	ASC(오름차순)	DESC(내림차순)
숫자	작은 값부터 정렬	큰 값부터 정렬
문자	사전 순서로 정렬	사전 반대 순서로 정렬
날짜	빠른 날짜 순서로 정렬	늦은 날짜 순서로 정렬
NULL	가장 마지막에 나온다.	가장 먼저 나온다.

영문자의 경우 소문자를 가장 큰 값으로, NULL값은 모든 값을 중 가장 작은 값으로 인식한다.

### ① 오름차순 정렬을 위한 ASC

<예> 사번을 기준으로 오름차순으로 정렬

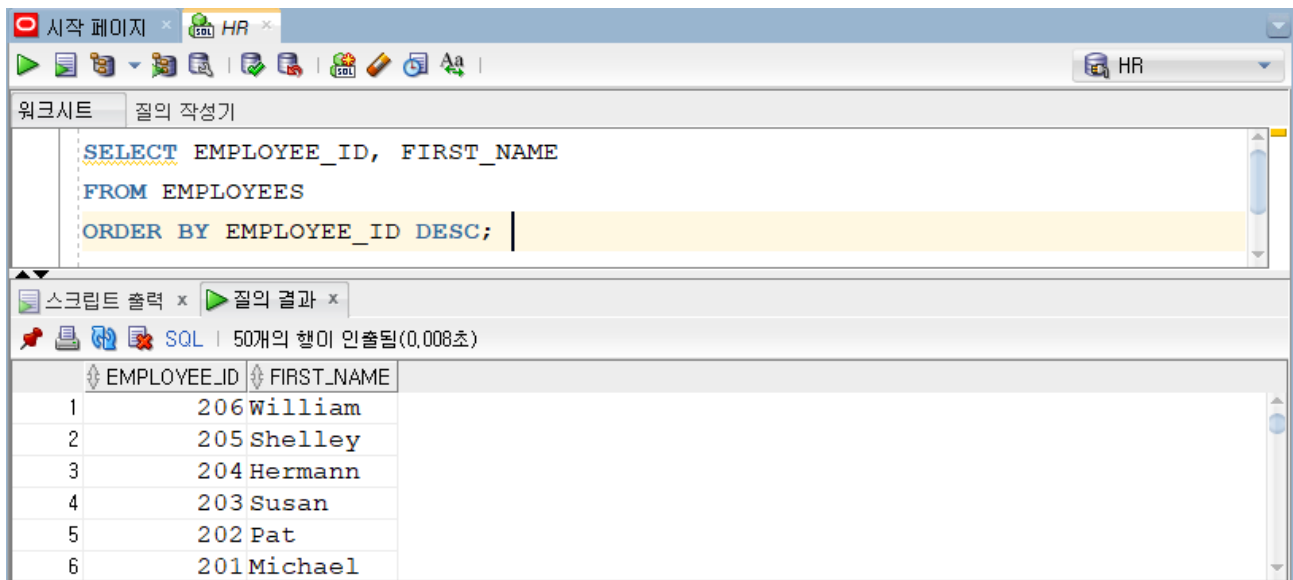
```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID ASC;
또는
ORDER BY EMPLOYEE_ID;
```



### ② 내림차순 정렬을 위한 DESC

<예> 사원번호를 기준으로 내림차순으로 정렬

```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID DESC;
```



<문제> 직원번호, 이름, 급여를 급여가 높은 순으로 출력하라.

<문제> 입사일이 가장 최근인 직원 순으로 직원번호, 이름, 입사일을 출력하라.