

## 6. 서브 쿼리

두 개 이상의 테이블의 정보가 필요할 경우 서브 쿼리를 사용한다.

단일 행 Sub Query, 다중 행 Sub Query에 대해서 학습한다.

서브 쿼리문을 이용해서 테이블을 생성한다.

서브 쿼리문을 이용해서 테이블에 데이터를 추가, 수정, 삭제한다.

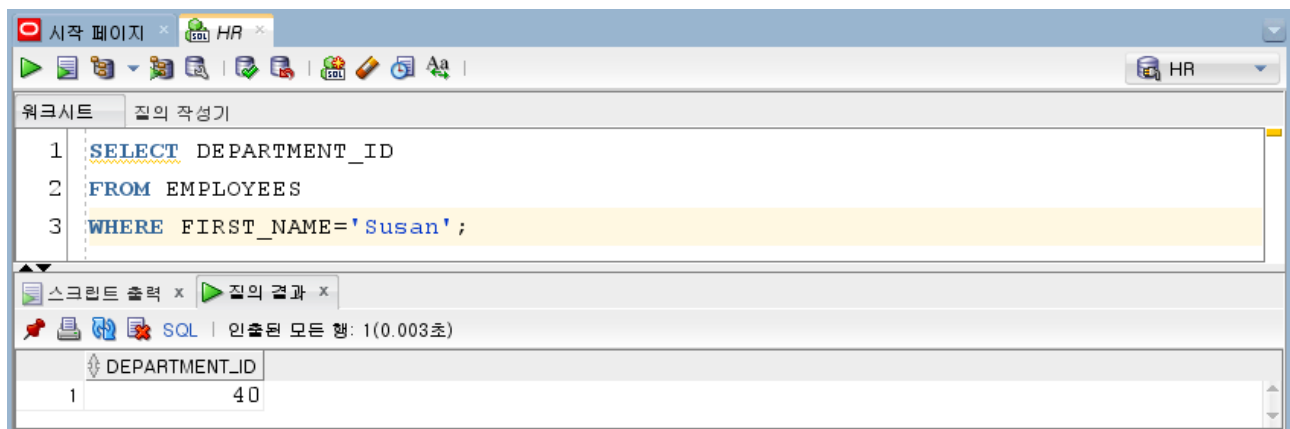
### 1) 서브 쿼리의 기본 개념

서브 쿼리는 하나의 SELECT 문장의 절 안에 포함된 또 하나의 SELECT 문장이다. 그렇기에 서브 쿼리를 포함하고 있는 쿼리문을 메인 쿼리, 포함된 또 하나의 쿼리를 서브 쿼리라 한다.

직원의 이름이 Susan인 직원이 어떤 부서 소속인지 소속 부서명을 알아내려면 조인을 사용해서 해결했지만 조인이 아닌 서브 쿼리문을 이용해서 해결한다.

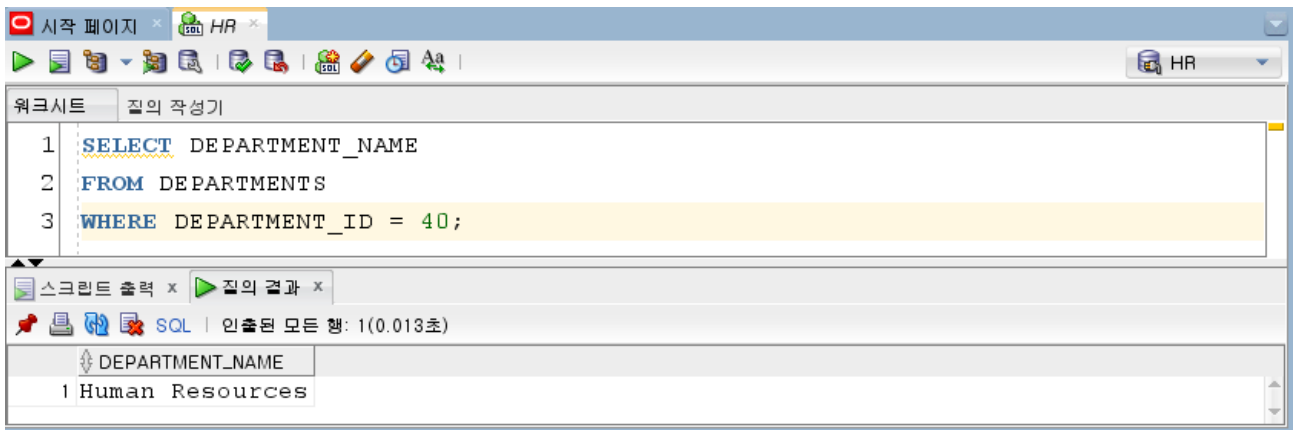
먼저 Susan의 부서명을 알기 위해서 부서 번호를 알아야 한다.

```
SELECT DEPARTMENT_ID
FROM EMPLOYEES
WHERE FIRST_NAME='Susan';
```



결과로 부서 번호가 40임을 알아내고 다음과 같이 쿼리문을 사용한다.

```
SELECT DEPARTMENT_NAME
FROM DEPARTMENTS
WHERE DEPARTMENT_ID = 40;
```



서브 쿼리로 변경하면 다음과 같다.

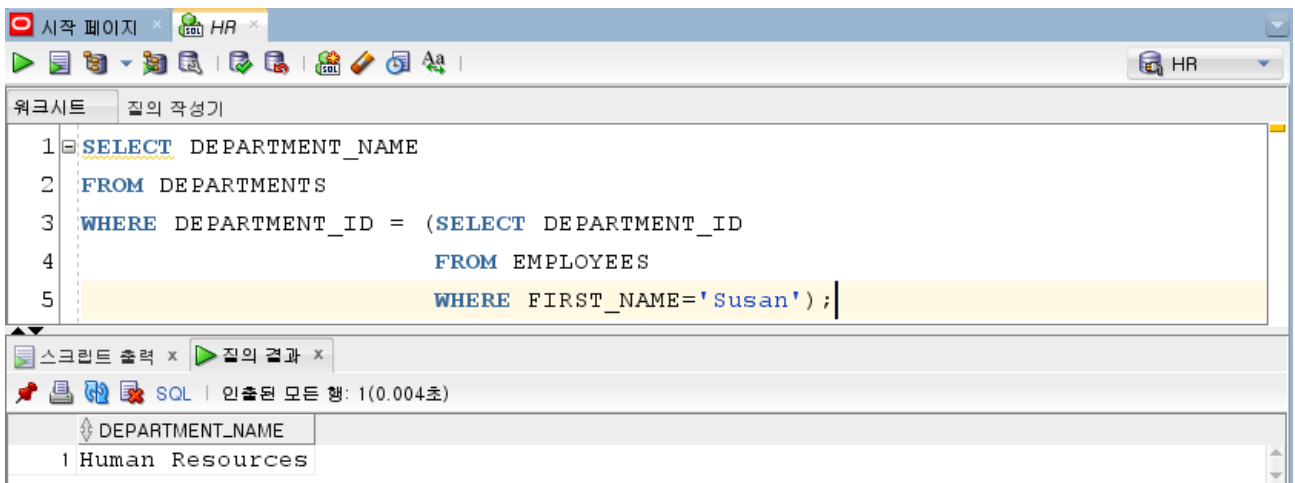
```
SELECT DEPARTMENT_NAME
FROM DEPARTMENTS
WHERE DEPARTMENT_ID = ( SELECT DEPARTMENT_ID
                        FROM EMPLOYEES
                        WHERE FIRST_NAME='Susan' );
```

메인 쿼리

```
SELECT DEPARTMENT_NAME
FROM DEPARTMENTS
WHERE DEPARTMENT_ID =
```

서브 쿼리

```
(SELECT DEPARTMENT_ID
FROM EMPLOYEES
WHERE FIRST_NAME='Susan');
```



서브 쿼리는 비교연산자의 오른쪽에 기술해야 하고 반드시 괄호로 둘러싸여야 한다.

서브 쿼리는 메인 쿼리가 실행되기 전에 한 번만 실행된다.

- 메인 쿼리에서 서브 쿼리의 결과값을 조건으로 사용할 때 SOME, ANY 또는 ALL 연산자를 사용하지 않는 일반적인 경우에는 서브 쿼리에서는 하나의 레코드값만 리턴해야 한다. 그러므로 대부분의 경우 서브 쿼리에는 GROUP BY, HAVING 문을 사용할 수 없다.
- 서브 쿼리에서 SELECT 하지 않은 컬럼은 주 쿼리에서 사용할 수 없다.
- 서브 쿼리 안에 서브 쿼리가 들어갈 수 있다. 메모리가 허용하는 한 무제한으로 중첩할 수 있다.

EMPLOYEES 테이블에서 Lex와 같은 부서에 있는 모든 사원의 이름과 입사일자(형식: 1981-11-17)를 출력하는 SELECT문을 작성하시오.

```
SELECT DEPARTMENT_ID
FROM EMPLOYEES
WHERE FIRST_NAME='Lex';
```

	DEPARTMENT_ID
1	90

```
SELECT FIRST_NAME, TO_CHAR(HIRE_DATE, 'YYYY-MM-DD') AS HIRE_DATE
FROM EMPLOYEES
WHERE DEPARTMENT_ID =90;
```

```
SELECT FIRST_NAME, TO_CHAR(HIRE_DATE, 'YYYY-MM-DD') AS HIRE_DATE, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = ( SELECT DEPARTMENT_ID
                        FROM EMPLOYEES
                        WHERE FIRST_NAME='Lex' );
```

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

```
1 SELECT FIRST_NAME, TO_CHAR(HIRE_DATE, 'YYYY-MM-DD') AS HIRE_DATE , DEPARTMENT_ID
2 FROM EMPLOYEES
3 WHERE DEPARTMENT_ID =(SELECT DEPARTMENT_ID
4                        FROM EMPLOYEES
5                        WHERE FIRST_NAME='Lex' );
```

The results pane shows the output of the query:

	FIRST_NAME	HIRE_DATE	DEPARTMENT_ID
1	Steven	2003-06-17	90
2	Neena	2005-09-21	90
3	Lex	2001-01-13	90

[문제 1] EMPLOYEES 테이블에서 CEO에게 보고하는 직원의 모든 정보를 출력하는 SELECT문을 작성하시오.

## [출력 결과]

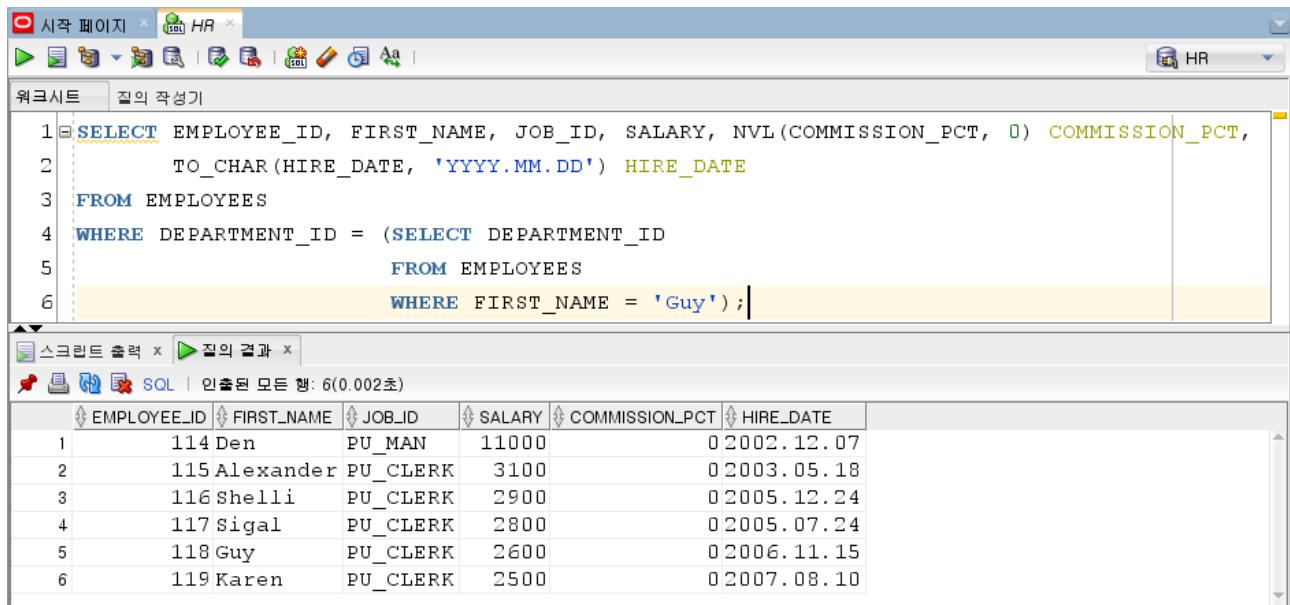
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	Michael	Hartstein	MHARTSTE	515.123.5555	04/02/17	MK_MAN	13000	(null)	100	20
2	Neena	Kochhar	NKochhar	515.123.4568	05/09/21	AD_VP	17000	(null)	100	90
3	Lex	De Haan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000	(null)	100	90
4	Den	Raphaely	DRAPHEAL	515.127.4561	02/12/07	PU_MAN	11000	(null)	100	30
5	Matthew	Weiss	MWEISS	650.123.1234	04/07/18	ST_MAN	8000	(null)	100	50
6	Adam	Fripp	AFRIPP	650.123.2234	05/04/10	ST_MAN	8200	(null)	100	50
7	Payam	Kaufling	PKAUFLIN	650.123.3234	03/05/01	ST_MAN	7900	(null)	100	50
8	Shanta	Vollman	SVOLLMAN	650.123.4234	05/10/10	ST_MAN	6500	(null)	100	50
9	Kevin	Mourgos	KMOURGOS	650.123.5234	07/11/16	ST_MAN	5800	(null)	100	50
10	John	Russell	JRUSSEL	011.44.1344.429268	04/10/01	SA_MAN	14000	0.4	100	80
11	Karen	Partners	KPARTNER	011.44.1344.467268	05/01/05	SA_MAN	13500	0.3	100	80
12	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	05/03/10	SA_MAN	12000	0.3	100	80
13	Gerald	Cambraut	GCAMBRAU	011.44.1344.619268	07/10/15	SA_MAN	11000	0.3	100	80
14	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	08/01/29	SA_MAN	10500	0.2	100	80

## 2) 단일 행 서브 쿼리

단일 행 Sub Query(Single Row)는 내부 SELECT 문장으로부터 오직 하나의 로우(행, row)만을 반환 받으며, 단일 행 비교 연산자(=, >, >=, <, <=, <>)를 사용한다.

Guy와 같은 부서에서 근무하는 사원의 정보를 출력하는 예

```
SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID, SALARY, NVL(COMMISSION_PCT, 0) COMMISSION_PCT,
       TO_CHAR(HIRE_DATE, 'YYYY.MM.DD') HIRE_DATE
FROM EMPLOYEES
WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID
                       FROM EMPLOYEES
                       WHERE FIRST_NAME = 'Guy');
```



## ① 서브 쿼리에서 그룹 함수의 사용

서브 쿼리를 사용하여 평균 급여보다 더 많은 급여를 받는 사원을 검색하는 쿼리

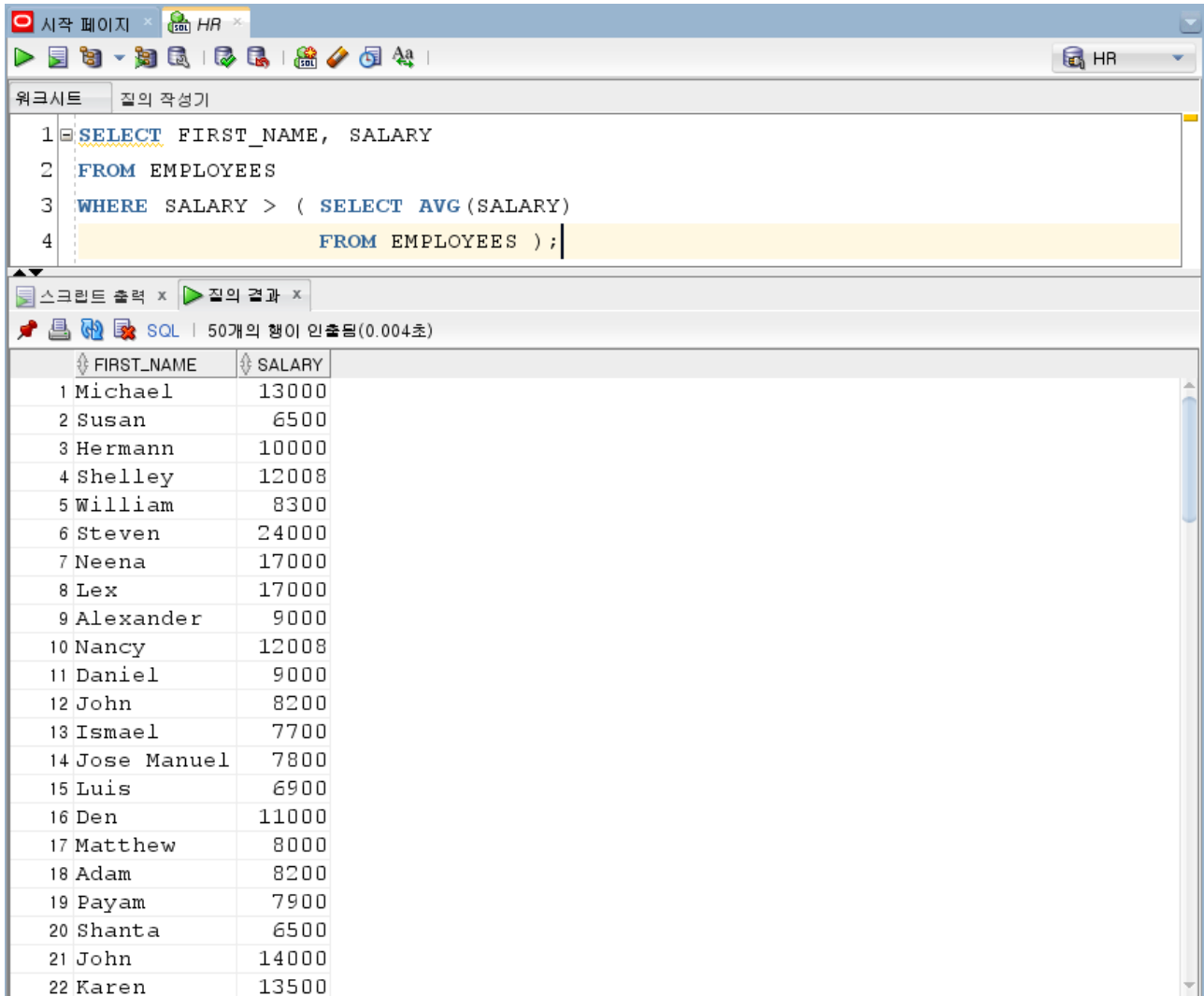
```
SELECT AVG(SALARY)
FROM EMPLOYEES;
```

평균을 구하는 쿼리문만 실행하면 아래와 같은 결과를 얻는다.

AVG(SALARY)

1 6461.831775700934579439252336448598130841

```
SELECT FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY > ( SELECT AVG(SALARY)
                  FROM EMPLOYEES );
```



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

```
1 SELECT FIRST_NAME, SALARY
2 FROM EMPLOYEES
3 WHERE SALARY > ( SELECT AVG (SALARY)
4                  FROM EMPLOYEES );
```

The results pane shows the output of the query, displaying a list of employees whose salary is greater than the average salary. The results are as follows:

	FIRST_NAME	SALARY
1	Michael	13000
2	Susan	6500
3	Hermann	10000
4	Shelley	12008
5	William	8300
6	Steven	24000
7	Neena	17000
8	Lex	17000
9	Alexander	9000
10	Nancy	12008
11	Daniel	9000
12	John	8200
13	Ismael	7700
14	Jose Manuel	7800
15	Luis	6900
16	Den	11000
17	Matthew	8000
18	Adam	8200
19	Payam	7900
20	Shanta	6500
21	John	14000
22	Karen	13500

### 3) 다중 행 서브 쿼리

다중 행 서브 쿼리는 서브 쿼리에서 반환되는 결과가 하나 이상의 행일 때 사용하는 서브 쿼리이다.

다중 행 서브 쿼리는 반드시 아래와 같은 다중 행 연산자(Multiple Row Operator)와 함께 사용해야 한다.

종류	의 미
IN	메인 쿼리의 비교 조건('=' 연산자로 비교할 경우)이 서브 쿼리의 결과 중에서 하나라도 일치하면 참
ANY	메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 하나 이상이 일치하면 참
ALL	메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 모든 값이 일치하면 참
EXIST	메인 쿼리의 비교 조건이 서브 쿼리의 결과 중에서 만족하는 값이 하나라도 존재하면 참

#### ① IN 연산자

결과가 2개 이상 구해지는 쿼리문을 서브 쿼리로 기술할 경우에는 다중 행 연산자와 함께 사용해야 한다. IN 연산자는 메인 쿼리의 비교 조건에서 서브 쿼리의 출력 결과와 하나라도 일치하면 메인 쿼리의 WHERE 절의 참이 되도록 하는 연산자이다.

**급여를 3000 이상 받는 사원이 소속된 부서와 동일한 부서에서 근무하는 사원을 출력하라**

서브 쿼리의 결과 중에서 하나라도 일치하면 참인 결과를 구하는 IN 연산자와 함께 사용해야 한다.

```

SELECT FIRST_NAME, SALARY, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( SELECT DISTINCT DEPARTMENT_ID
                        FROM EMPLOYEES
                        WHERE SALARY >= 13000 )
ORDER BY DEPARTMENT_ID;

```

FIRST_NAME	SALARY	DEPARTMENT_ID
Michael	13000	20
Pat	6000	20
Lindsey	8000	80
Jack	8400	80
Jonathon	8600	80
John	14000	80
Karen	13500	80

EMPLOYEES 테이블에서 이름에 "z"가 있는 직원이 근무하는 부서에서 근무하는 모든 직원에 대해 직원 번호, 이름, 급여를 출력하는 SELECT문을 작성하시오. 단 직원번호 순으로 출력하여라.

```
SELECT EMPLOYEE_ID, FIRST_NAME, DEPARTMENT_ID, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (SELECT DISTINCT DEPARTMENT_ID
                        FROM EMPLOYEES
                        WHERE FIRST_NAME LIKE '%z%')
ORDER BY DEPARTMENT_ID;
```

DEPARTMENT_ID	
1	50
2	80

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

```
1 SELECT EMPLOYEE_ID, FIRST_NAME, DEPARTMENT_ID, SALARY
2 FROM EMPLOYEES
3 WHERE DEPARTMENT_ID IN (SELECT DISTINCT DEPARTMENT_ID
4                         FROM EMPLOYEES
5                         WHERE FIRST_NAME LIKE '%z%')
6 ORDER BY DEPARTMENT_ID;
```

The results pane shows the output of the query, displaying columns EMPLOYEE\_ID, FIRST\_NAME, DEPARTMENT\_ID, and SALARY. The results are sorted by DEPARTMENT\_ID, with all employees from department 50 listed first, followed by employees from department 80.

EMPLOYEE_ID	FIRST_NAME	DEPARTMENT_ID	SALARY
41	193 Britney	50	3900
42	194 Samuel	50	3200
43	195 Vance	50	2800
44	196 Alana	50	3100
45	197 Kevin	50	3000
46	145 John	80	14000
47	179 Charles	80	6200
48	147 Alberto	80	12000
49	148 Gerald	80	11000
50	149 Eleni	80	10500
51	150 Peter	80	10000

EMPLOYEES 테이블에서 Susan 또는 Lex와 월급이 같은 직원의 이름, 업무, 급여를 출력하는 SELECT문을 작성하시오. (Susan, Lex는 제외)

```
SELECT FIRST_NAME, JOB_ID, SALARY
FROM EMPLOYEES
WHERE SALARY IN (SELECT SALARY
                FROM EMPLOYEES
                WHERE FIRST_NAME IN ('Susan', 'Lex')) AND
                (FIRST_NAME <> 'Susan' AND FIRST_NAME <> 'Lex');
```

시작 페이지 x HR x

워크시트 | 집의 작성기

```

1 SELECT FIRST_NAME, JOB_ID, SALARY
2 FROM EMPLOYEES
3 WHERE SALARY IN (SELECT SALARY
4                  FROM EMPLOYEES
5                  WHERE FIRST_NAME IN ('Susan', 'Lex')) AND
6                  (FIRST_NAME <> 'Susan' AND FIRST_NAME <> 'Lex');

```

스크립트 출력 x | 집의 결과 x

SQL | 인출된 모든 행: 2(0.004초)

	FIRST_NAME	JOB_ID	SALARY
1	Shanta	ST_MAN	6500
2	Neena	AD_VP	17000

EMPLOYEES 테이블에서 적어도 한 명 이상으로부터 보고를 받을 수 있는 직원의 직원번호, 이름, 업무, 부서번호를 출력하는 SELECT문을 작성하시오. (다른 직원을 관리하는 직원)

```

SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID, DEPARTMENT_ID
FROM EMPLOYEES
WHERE EMPLOYEE_ID IN ( SELECT DISTINCT MANAGER_ID
                       FROM EMPLOYEES);

```

	MANAGER_ID
1	124
2	108
3	121
4	145
5	101
6	(null)
7	103
8	120
9	122
10	146
11	148
12	149
13	201
14	114
15	100
16	205
17	102
18	123
19	147



시작 페이지 x HR x

워크시트 | 질의 작성기

```

1 SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID, DEPARTMENT_ID
2 FROM EMPLOYEES
3 WHERE EMPLOYEE_ID IN (SELECT DISTINCT MANAGER_ID
4                        FROM EMPLOYEES);

```

스크린샷 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 18(0.002초)

	EMPLOYEE_ID	FIRST_NAME	JOB_ID	DEPARTMENT_ID
1	201	Michael	MK_MAN	20
2	205	Shelley	AC_MGR	110
3	100	Steven	AD_PRES	90
4	101	Neena	AD_VP	90
5	102	Lex	AD_VP	90
6	103	Alexander	IT_PROG	60
7	108	Nancy	FI_MGR	100
8	114	Den	PU_MAN	30
9	120	Matthew	ST_MAN	50
10	121	Adam	ST_MAN	50
11	122	Payam	ST_MAN	50
12	123	Shanta	ST_MAN	50
13	124	Kevin	ST_MAN	50
14	145	John	SA_MAN	80
15	146	Karen	SA_MAN	80
16	147	Alberto	SA_MAN	80
17	148	Gerald	SA_MAN	80
18	149	Eleni	SA_MAN	80

[문제 2] EMPLOYEES 테이블에서 Accounting 부서에서 근무하는 직원과 같은 업무를 하는 직원의 이름, 업무명을 출력하는 SELECT문을 작성하시오.

[결과]

	FIRST_NAME	JOB_ID	DEPARTMENT_ID
1	Shelley	AC_MGR	110
2	William	AC_ACCOUNT	110

## ② ALL 연산

ALL 조건은 메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 모든 값이 일치하면 참이다. 찾아진 값에 대해서 AND 연산을 해서 모두 참이면 참이 되는 셈이다. > ALL 은 "모든 비교값 보다 크냐"고 묻는 것이 되므로 **최대값보다 더 크면 참**이 된다

30번 소속 직원들 중에서 급여를 가장 많이 받은 사람 보다 더 많은 급여를 받는 사람의 이름, 급여를 출력하는 쿼리문 작성(30번 부서 직원 급여들 모두에 대해서 커야 하므로 최대값보다 큰 급여만 해당)

SELECT FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY > ALL ( SELECT SALARY

FROM EMPLOYEES  
WHERE DEPARTMENT\_ID=30);

	SALARY
1	11000
2	3100
3	2900
4	2800
5	2600
6	2500

30번 부서의 최대 급여보다 많이 직원들만 출력되도록 서브 쿼리를 작성해야 하므로 ALL조건 연산자를 사용해야 한다. ALL 조건 연산자는 서브 쿼리의 결과 값 모두에 대해서 커야만 한다.

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

```

1 SELECT FIRST_NAME, SALARY
2 FROM EMPLOYEES
3 WHERE SALARY > ALL (SELECT SALARY
4                     FROM EMPLOYEES
5                     WHERE DEPARTMENT_ID=30);

```

The results pane shows the output of the query, displaying 10 rows of employee data:

	FIRST_NAME	SALARY
1	Lisa	11500
2	Alberto	12000
3	Nancy	12008
4	Shelley	12008
5	Michael	13000
6	Karen	13500
7	John	14000
8	Lex	17000
9	Neena	17000
10	Steven	24000

③ ANY 연산자(연산자 > ANY 는 찾아진 값에 대해서 하나라도 크면 참이 되는 셈이 된다.)  
ANY 조건은 메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 하나 이상만 일치하면 참이다.  
> ANY는 찾아진 값에 대해서 하나라도 크면 참이 되는 셈이다. 그러므로 찾아진 값 중에서 가장 작은 값 즉, 최소값 보다 크면 참이 된다.

부서 번호가 110번인 직원들의 급여 중 가장 작은 값(8300)보다 많은 급여를 받는 직원의 이름, 급여를 출력

```

SELECT FIRST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY > ANY ( SELECT SALARY
                     FROM EMPLOYEES
                     WHERE DEPARTMENT_ID=110);

```

	SALARY
1	12008
2	8300

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

```

1 SELECT FIRST_NAME, SALARY
2 FROM EMPLOYEES
3 WHERE SALARY > ANY(SELECT SALARY
4                     FROM EMPLOYEES
5                     WHERE DEPARTMENT_ID=110);

```

The results pane shows the output of the query, displaying a list of employees and their salaries. The results are as follows:

FIRST_NAME	SALARY
15 Eleni	10500
16 Harrison	10000
17 Janette	10000
18 Peter	10000
19 Hermann	10000
20 Tayler	9600
21 David	9500
22 Patrick	9500
23 Danielle	9500
24 Alexander	9000
25 Daniel	9000
26 Peter	9000
27 Allan	9000
28 Alyssa	8800
29 Jonathon	8600
30 Jack	8400

#### ④ EXISTS 연산자

EXISTS 연산자는 서브 쿼리문에서 주로 사용하며 **서브 쿼리의 결과 값이 존재하면** 바로 메인 쿼리의 결과 값을 리턴한다. 서브 쿼리의 결과 값이 존재하지 않는다면 메인 쿼리의 어떤 값도 리턴되지 않는 문장이다. 쿼리 속도 면에서는 서브 쿼리 사용 시 IN보다는 EXISTS가 훨씬 빠르다. EXISTS의 반대말로 NOT EXISTS도 사용 가능하다.

```

SELECT *
FROM DEPARTMENTS
WHERE EXISTS ( SELECT *
                FROM EMPLOYEES
                WHERE DEPARTMENT_ID = 10);

```

시작 페이지 x HR x

워크시트 | 질의 작성기

```

1 SELECT *
2 FROM DEPARTMENTS
3 WHERE EXISTS ( SELECT *
4                 FROM EMPLOYEES
5                 WHERE DEPARTMENT_ID = 10 );

```

스크린트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 27(0.005초)

	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

#### 4) 서브 쿼리로 테이블 작성하기

##### ① 서브 쿼리로 테이블 복사하기

```
DROP TABLE EMP01;
```

```
CREATE TABLE EMP01
```

```
AS
```

```
SELECT *
```

```
FROM EMPLOYEES;
```

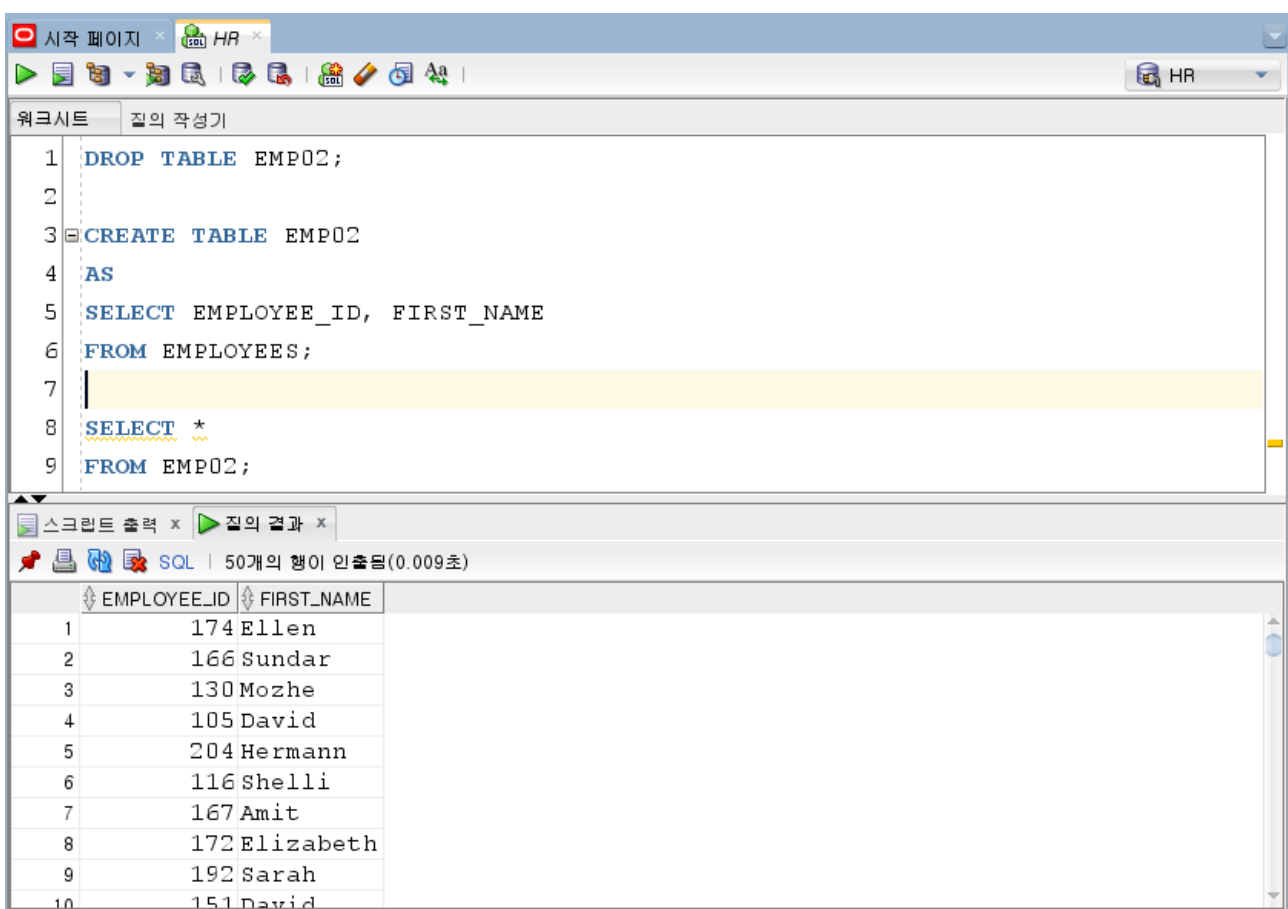
```
SELECT *
```

```
FROM EMP01;
```

모든 칼럼이 아닌 특정 칼럼만 선택적으로 복사하려면 복사할 칼럼의 이름을 기술해주면 된다.

```
DROP TABLE EMP02;  
  
CREATE TABLE EMP02  
AS  
SELECT EMPLOYEE_ID, FIRST_NAME  
FROM EMPLOYEES;
```

```
SELECT *  
FROM EMP02;
```



## ② 테이블의 구조만 복사하기

```
DROP TABLE EMP03;  
  
CREATE TABLE EMP03  
AS  
SELECT *  
FROM EMPLOYEES  
WHERE 1=0;
```

```
SELECT *
FROM EMP03;

DESC EMP03;
```

EMPLO...	FIRST_N...	LAST_N...	EMAIL	PHONE...	HIRE_D...	JOB_ID	SALARY	COMMIS...	MANAG...	DEPART...
----------	------------	-----------	-------	----------	-----------	--------	--------	-----------	----------	-----------

이름	널?	유형
EMPLOYEE_ID		NUMBER (6)
FIRST_NAME		VARCHAR2 (20)
LAST_NAME	NOT NULL	VARCHAR2 (25)
EMAIL	NOT NULL	VARCHAR2 (25)
PHONE_NUMBER		VARCHAR2 (20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2 (10)
SALARY		NUMBER (8, 2)
COMMISSION_PCT		NUMBER (2, 2)
MANAGER_ID		NUMBER (6)
DEPARTMENT_ID		NUMBER (4)

##### 5) 서브 쿼리를 이용한 데이터 추가

서브 쿼리를 사용하여 INSERT 문장을 작성하며, VALUES 절은 사용하지 않는다.

VALUES 절에 기술하는 자료를 서브 쿼리에서 얻어올 것이다. 단, 서브 쿼리의 값 개수와 INSERT할 테이블의 열 수가 일치해야 한다.

```
DROP TABLE DEPT01;
CREATE TABLE DEPT01
AS
SELECT *
FROM DEPARTMENTS
WHERE 1=0;
```

```
SELECT *
FROM DEPT01;
```

```
INSERT INTO DEPT01
SELECT *
FROM DEPARTMENTS;
```

```
SELECT *
FROM DEPT01;
```

## 6) 서브 쿼리를 이용한 데이터 수정

10번 부서의 지역번호를 40번 부서의 지역번호로 변경하기 위해서 서브 쿼리를 사용한 예

```
UPDATE DEPT01
SET LOCATION_ID = ( SELECT LOCATION_ID
                    FROM DEPARTMENTS
                    WHERE DEPARTMENT_ID=40 )
WHERE DEPARTMENT_ID=10;
```

```
SELECT *
FROM DEPT01;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	2400
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500

## 7) 서브 쿼리를 이용한 두 개 이상의 칼럼에 대한 값 변경

서브 쿼리를 사용한 UPDATE 형식 2가지

### ① UPDATE table\_name

**SET column\_name1 = (sub\_query1), column\_name2 = (sub\_query2), ...**  
**WHERE 조건;**

### ② UPDATE table\_name

**SET (column\_name1, column\_name2) = (sub\_query)**  
**WHERE 조건;**

20번 부서의 부서명과 지역명을 30번 부서의 부서명과 지역명으로 수정.

```
UPDATE DEPT01
SET DEPARTMENT_NAME = ( SELECT DEPARTMENT_NAME
                        FROM DEPT01
                        WHERE DEPARTMENT_ID = 30),
    LOCATION_ID = ( SELECT LOCATION_ID
                   FROM DEPT01
                   WHERE DEPARTMENT_ID = 30)
WHERE DEPARTMENT_ID = 20;
```

```
UPDATE DEPT01
SET (DEPARTMENT_NAME, LOCATION_ID) = ( SELECT DEPARTMENT_NAME, LOCATION_ID
                                       FROM DEPT01
```

```
WHERE DEPARTMENT_ID = 30)
```

```
WHERE DEPARTMENT_ID = 20;
```

```
SELECT *  
FROM DEPT01;
```

#### 8) 서브 쿼리를 이용한 데이터 삭제

예제로 사용할 테이블을 생성한다.

```
DROP TABLE EMP01;  
CREATE TABLE EMP01  
AS  
SELECT *  
FROM EMPLOYEES;
```

```
SELECT *  
FROM EMP01;
```

```
DELETE FROM EMP01  
WHERE DEPARTMENT_ID = ( SELECT DEPARTMENT_ID  
                        FROM DEPARTMENTS  
                        WHERE DEPARTMENT_NAME='Sales');
```

```
SELECT *  
FROM EMP01;
```

[문제 3] 직급이 'ST\_MAN'인 직원이 받는 급여들의 최소 급여보다 많이 받는 직원들의 이름과 급여를 출력하되 부서번호가 20번인 직원은 제외한다.

[문제 4] EMPLOYEES 테이블에서 Valli라는 이름을 가진 직원과 업무명 및 월급이 같은 사원의 모든 정보를 출력하는 SELECT문을 작성하시오. (결과에서 Valli는 제외)

[문제 5] EMPLOYEES 테이블에서 월급이 자신이 속한 부서의 평균 월급보다 높은 사원의 부서번호, 이름, 급여를 출력하는 SELECT문을 작성하시오.