

7. 서브 쿼리

컴퓨터공학부

김은경

[학습 목표]

1. 서브쿼리의 작성 규칙을 나열할 수 있다.
2. 서브쿼리의 유형별 특징을 설명할 수 있다.
3. 단일 행 서브쿼리를 작성할 수 있다.
4. 다중 행 서브쿼리를 작성할 수 있다.
5. 다중 열 서브쿼리를 작성할 수 있다.

1. 서브쿼리 개념

★ 서브쿼리(subquery)란?

- 내부 질의(내부 SELECT 문, 중첩된 SELECT 문)를 의미하며, 메인쿼리에서 사용할 값을 반환함

★ 메인쿼리(main query)란?

- 외부 질의를 의미하며, 서브쿼리가 반환한 값을 이용해서 메인쿼리가 완성됨



- ✓ 예: 김대정 직원보다 급여를 많이 받는 사원은 누구일까?

메인쿼리: 김대정 직원보다 급여를 많이 받는 사원은?

서브쿼리: 김대정 직원의 급여는 얼마인가?

1. 서브쿼리 개념

★ 서브쿼리의 형식

- WHERE 절의 연산자 뒤에, 괄호 안에 서브쿼리를 기술해야 함

```
SELECT   선택리스트1
FROM     테이블 이름1
WHERE    표현식 op
          (SELECT   선택리스트2
           FROM     테이블 이름2);
```

. 선택리스트 : 열 이름이나 그룹함수 나열

. op(operator, 연산자) : <, <=, = , IN 등과 같은 비교 연산자

1. 서브쿼리 개념

★ 서브쿼리의 특징

- ① 서브쿼리는 메인쿼리가 실행되기 전에 **한 번만** 실행됨
- ② 서브쿼리의 실행 결과는 메인쿼리를 완성하는데 사용됨
- ③ 서브쿼리는 **WHERE 절** 외에 **FROM 절**과 **HAVING 절**에서도 사용할 수 있음
- ④ WHERE나 HAVING 절에 하나 이상의 서브쿼리를 논리 연산자인 AND나 OR로 연결하여 사용할 수 있음

★ 서브쿼리 작성 규칙

- ① 서브쿼리는 **괄호**로 묶어서 기술해야 함
- ② 서브쿼리는 **비교 연산자의 우측**에 위치해야 함
- ③ 서브쿼리에 ORDER BY 절을 포함할 수 없음
- ④ 단일 행 서브쿼리에는 단일 행 비교 연산자(>, =, >=, <, <=, <>)만 사용 가능함
- ⑤ 다중 행 서브쿼리에는 다중 행 비교 연산자(IN, ANY, ALL)만 사용 가능함
- ⑥ SELECT 문의 **FROM 절, WHERE 절, HAVING 절**에서만 사용할 수 있음

1. 서브쿼리 개념

✓ 서브쿼리 활용 사례

- 사원 번호가 7872인 사원과 같은 업무를 담당하는 모든 사원의 사원번호와 이름을 출력하되, 사원이름의 오름차순으로 출력하기

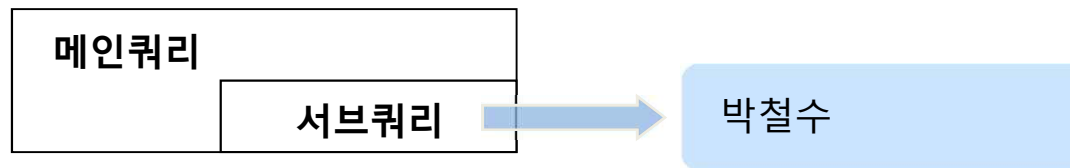
```
SELECT emp_id, emp_name, job
FROM emp
WHERE job = (SELECT job
              FROM emp
              WHERE emp_id = 7872)
ORDER BY emp_name ;
```

EMP_ID	EMP_NAME	JOB
7920	김마리아	사무직
7933	김철수	사무직
7872	이문정	사무직

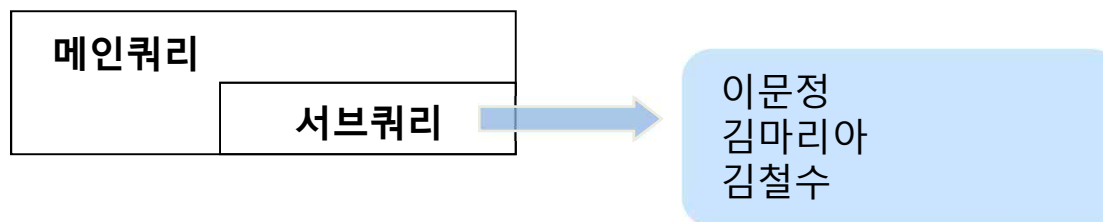
1. 서브쿼리 개념

★ 서브쿼리의 유형

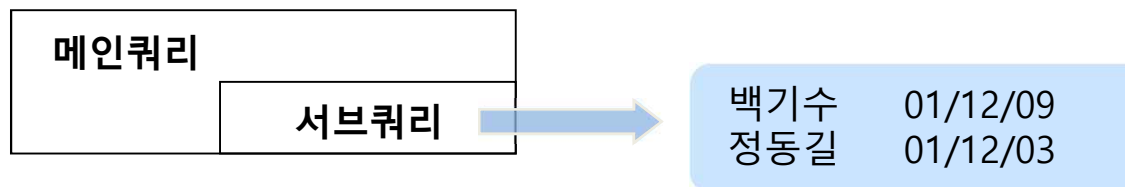
① 단일 행 서브쿼리 : 내부 질의에서 단 하나의 행만 반환



② 다중 행 서브쿼리 : 내부 질의에서 둘 이상의 행 반환



③ 다중 열 서브쿼리 : 내부 질의에서 둘 이상의 열 반환



2. 단일 행 서브쿼리

★ 단일 행 서브쿼리의 특징

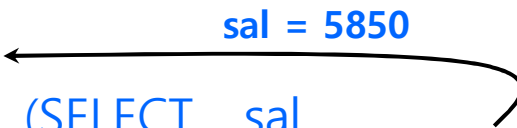
- ① 내부 질의에서 **단 하나의 행만 반환함**
- ② 단일 행 서브쿼리를 포함하는 질의문에서는 **단일 행 비교 연산자 (>, =, >=, <,<=, <>) 만 사용 가능함**
- ③ 서브쿼리의 **SELECT 절에서 그룹 함수**를 사용할 수 있음

2. 단일 행 서브쿼리

✓ 하나의 단일 행 서브쿼리를 포함하는 사례

- 사원 번호가 6351인 사원보다 적은 급여를 받는 모든 사원의 사원번호와 이름 출력하기

```
SELECT emp_id, emp_name
FROM emp
WHERE sal < (SELECT sal
              FROM emp
              WHERE emp_id = 6351);
```



EMP_ID	EMP_NAME
7489	민동규
7522	정종철
7910	이영희
7878	백기수
7854	진영진
7872	이문정
7920	김마리아
7901	정동길
7933	김철수

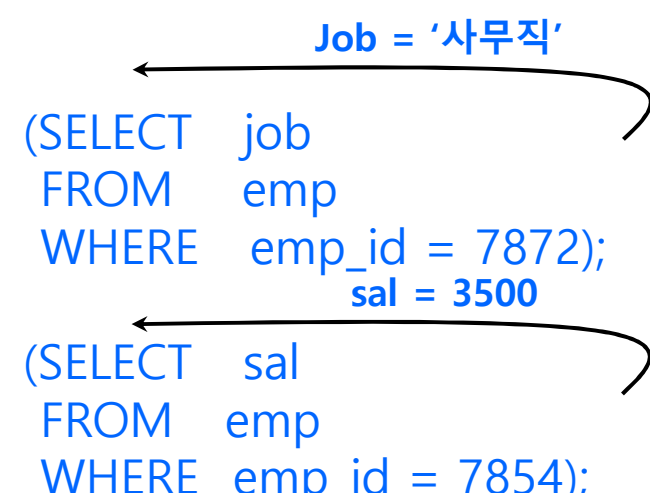
9 rows selected.

2. 단일 행 서브쿼리

✓ 두 개의 단일 행 서브쿼리를 포함하는 사례

- 업무는 사원 7872와 같고, 급여는 사원 7854보다 같거나 적은 모든 사원의 이름과 담당 업무, 그리고 급여 출력하기

```
SELECT emp_name, job, sal
FROM emp
WHERE job =
      (SELECT job
       FROM emp
       WHERE emp_id = 7872);
AND sal <=
      (SELECT sal
       FROM emp
       WHERE emp_id = 7854);
```



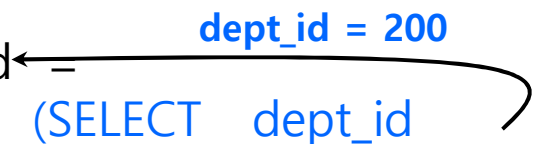
EMP_NAME	JOB	SAL
이문정	사무직	2500
김마리아	사무직	2650
김철수	사무직	2500

2. 단일 행 서브쿼리

✓ 서브쿼리와 메인쿼리가 서로 다른 테이블을 사용하는 사례

- 근무처가 대전인 모든 사원의 이름과 담당업무, 부서번호 출력하기

```
SELECT emp_name, job, dept_id
FROM emp
WHERE dept_id = (SELECT dept_id
                  FROM dept
                  WHERE dept_loc = '대전');
```



EMP_NAME	JOB	DEPT_ID
박철수	대표이사	200
김철수	부장	200
백기수	연구직	200
김철수	사무직	200

[비교] 서브쿼리 대신 테이블 조인을 활용해도 결과 동일함

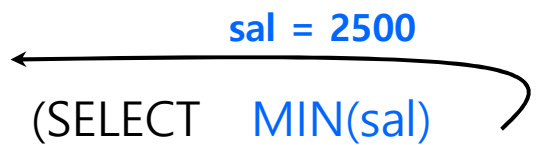
```
SELECT emp_name, job, e.dept_id
FROM emp e, dept d
WHERE dept_loc = '대전'
AND e.dept_id = d.dept_id;
```

2. 단일 행 서브쿼리

✓ 서브쿼리에서 그룹함수 사용하는 사례

- 모든 사원 가운데 최소 급여를 받는 사원 출력하기

```
SELECT emp_name, job, sal
FROM emp
WHERE sal = (SELECT MIN(sal)
              FROM emp);
```



EMP_NAME	JOB	SAL
이문정	사무직	2500
김철수	사무직	2500

[비교] 서브쿼리를 사용하지 않고 최소 급여 받는 사원 검색하기

```
SELECT emp_id, MIN(sal)
FROM emp;
```

```
SELECT emp_id, MIN(sal)
      *
ERROR at line 1:
ORA-00937: not a single-group group function
```

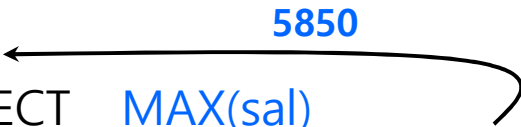
emp_id가 그룹 함수가 아니므로,
GROUP BY 절에 명시되지 않은 상태
에서 SELECT 절에 기술할 수 없음

2. 단일 행 서브쿼리

✓ HAVING절의 서브쿼리 사례(1)

- 300번 부서의 최대 급여보다 최대 급여가 큰 모든 부서번호와 그 부서의 최대 급여 출력하기

```
SELECT dept_id, MAX(sal)
FROM emp
GROUP BY dept_id
HAVING MAX(sal) > (SELECT MAX(sal)
                   FROM emp
                   WHERE dept_id = 300);
```



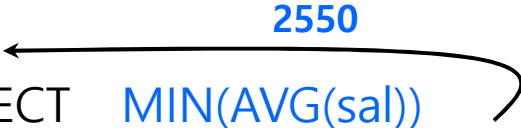
DEPT_ID	MAX(SAL)
100	6200
400	7000
200	9000

2. 단일 행 서브쿼리

✓ HAVING절의 서브쿼리 사례(1)

- 업무별 평균 급여가 가장 적은 업무와 그 업무의 평균 급여 출력하기

```
SELECT job, AVG(sal)
FROM emp
GROUP BY job
HAVING AVG(sal) = (SELECT MIN(AVG(sal))
                   FROM emp
                   GROUP BY job);
```



JOB	AVG(SAL)
사무직	2550

3. 서브쿼리의 오류

★ 서브쿼리 오류의 원인

- ① 단일 행 서브쿼리가 하나 이상의 행을 반환하면 오류가 발생함
- ② 단일 행 서브쿼리가 반환하는 행이 하나도 없어도 오류가 발생함

★ 서브쿼리 오류 수정 방법

- ① 단일 행 비교 연산자와 함께 기술된 서브쿼리가 하나 이상의 행을 반환하는 경우
→ 단일 행 비교 연산자 대신 다중 행 비교 연산자 사용
- ② 단일 행 서브쿼리가 반환하는 행이 하나도 없는 경우
→ 하나의 행을 반환하도록 수정

3. 서브쿼리의 오류

✓ 단일 행 서브쿼리가 하나 이상의 행을 반환하는 경우

- 각 부서별로 최대 급여를 받는 사원의 부서번호와 이름, 업무, 그리고 급여 검색하기

```
SELECT dept_id, emp_name, job, sal
FROM emp
WHERE sal = (SELECT MAX(sal)
              FROM emp
              WHERE dept_id IS NOT NULL
              GROUP BY dept_id);
```

← 4개 부서의 MAX(sal) 값(6200, 7000, 5850, 9000) 반환

```
(SELECT MAX(sal)
*)
```

ERROR at line 4:
ORA-01427: single-row subquery returns more than one row

3. 서브쿼리의 오류

➔ 오류 수정 : 단일 행 비교 연산자(=)를 다중 행 비교 연산자(IN)로 변경

- 각 부서별로 최대 급여를 받는 사원의 부서번호와 이름, 업무, 그리고 급여 검색하기

```
SELECT dept_id, emp_name, job, sal
FROM emp
WHERE sal IN
```

4개 부서의 MAX(sal) 값(6200, 7000, 5850, 9000) 반환

```
(SELECT MAX(sal)
FROM emp
WHERE dept_id IS NOT NULL
GROUP BY dept_id);
```

DEPT_ID	EMP_NAME	JOB	SAL
200	박철수	대표이사	9000
100	김대정	부장	6200
400	이종길	부장	7000
300	진대준	부장	5850

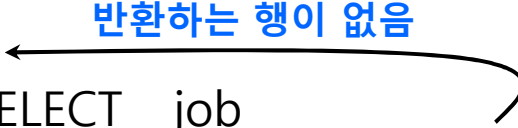
3. 서브쿼리의 오류

✓ 단일 행 서브쿼리가 반환하는 행이 없는 경우

- 이문정 사원과 같은 업무를 하는 모든 사원 검색하기

```
SELECT emp_name, job
FROM emp
WHERE job = (SELECT job
              FROM emp
              WHERE emp_name = '이문정');
```

반환하는 행이 없음



no rows selected

선택된 레코드가 없음



3. 서브쿼리의 오류

➔ 오류 수정 : 사원이름을 '이문정' 으로 수정

- 이문정 사원과 같은 업무를 하는 모든 사원 검색하기

```
SELECT emp_name, job
FROM emp
WHERE job = (SELECT job
              FROM emp
              WHERE emp_name = '이문정');
```

← '사무직' 반환

EMP_NAME	JOB
이문정	사무직
김마리아	사무직
김철수	사무직

4. 다중 행 서브쿼리

★ 다중 행 서브쿼리의 특징

- ① 내부 질의에서 하나 이상의 행을 반환함
- ② 다중 행 서브쿼리를 포함하는 질의문에서는 다중 행 비교 연산자 (IN, ANY, ALL)만 사용 가능함
- ③ 서브쿼리의 SELECT 절에서 그룹 함수를 사용할 수 있음

★ 다중 행 비교 연산자

연산자	역할
IN	서브쿼리가 반환한 목록의 어떤 값과 같은지 비교
ANY	서브쿼리가 반환한 목록의 각각의 값과 비교
ALL	서브쿼리가 반환한 목록의 모든 값과 비교

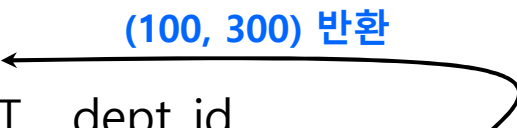
4. 다중 행 서브쿼리

✓ IN 연산자 사용 사례

- 근무처가 서울인 모든 사원의 이름과 업무, 부서번호 검색하기

```
SELECT emp_name, job, dept_id
FROM emp
WHERE dept_id IN (SELECT dept_id
                  FROM dept
                  WHERE dept_loc = '서울');
```

(100, 300) 반환



EMP_NAME	JOB	DEPT_ID
김대정	부장	100
진대준	부장	300
이영희	경리	300
이문정	사무직	100
김마리아	사무직	300

4. 다중 행 서브쿼리

✓ ANY 연산자 사용 사례(1)

- 어떤 세일즈 직원보다 급여가 작으면서, 업무가 세일즈가 아닌 모든 직원 검색하기

```
SELECT emp_name, job, sal
FROM emp
WHERE sal < ANY
      (SELECT sal
       FROM emp
       WHERE job = '세일즈')
AND job <> '세일즈';
```

(3700, 4520, 3500) 반환

sal의 최대값(4520)
보다 작은

EMP_NAME	JOB	SAL
김철수	사무직	2500
이문정	사무직	2500
김마리아	사무직	2650
이영희	경리	3200
정동길	연구직	4500

4. 다중 행 서브쿼리

✓ ANY 연산자 사용 사례(2)

- 어떤 한 부장 직원보다 급여가 많으면서, 부장이 아닌 모든 직원 검색하기

```
SELECT emp_name, job, sal
FROM emp
WHERE sal > ANY (6200, 7000, 5850, 6500) 반환
              (SELECT sal
               FROM emp
               WHERE job = '부장')
AND job <> '부장';
```

EMP_NAME	JOB	SAL
박철수	대표이사	9000

4. 다중 행 서브쿼리

✓ ALL 연산자 사용 사례(1)

- 급여가 모든 부서의 평균보다 많은 사원 검색하기

```
SELECT emp_name, job, sal, dept_id
FROM emp
WHERE sal > ALL
```

(4350, 4680, 3900, 5750) 반환

(SELECT AVG(sal)
FROM emp
WHERE dept_id IS NOT NULL
GROUP BY dept_id);

AVG(sal)의 최대값
보다 큰

EMP_NAME	JOB	SAL	DEPT_ID
박철수	대표이사	9000	200
김대정	부장	6200	100
이종길	부장	7000	400
진대준	부장	5850	300
김철수	부장	6500	200

4. 다중 행 서브쿼리

✓ ALL 연산자 사용 사례(2)

- 급여가 모든 부서의 평균보다 많은 사원 검색하기

```
SELECT emp_name, job, sal, dept_id
FROM emp
WHERE sal < ALL (SELECT AVG(sal)
                  FROM emp
                  WHERE dept_id IS NOT NULL
                  GROUP BY dept_id);
```

AVG(sal)의 최소값보다 작은

(4350, 4680, 3900, 5750) 반환

EMP_NAME	JOB	SAL	DEPT_ID
민동규	세일즈	3700	400
이영희	경리	3200	300
진영진	세일즈	3500	400
이문정	사무직	2500	100
김마리아	사무직	2650	300
김철수	사무직	2500	200

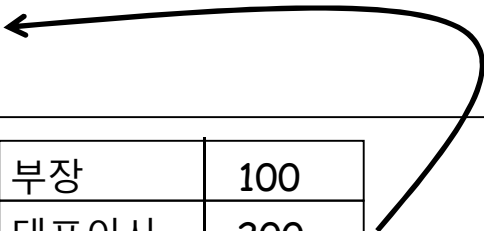
6 rows selected.

5. 다중 열 서브쿼리

★ 다중 열 서브쿼리의 특징

- ① 내부 질의에서 하나 이상의 열을 반환함
- ② 두 개 이상의 열을 비교하기 위해서 WHERE 절에 논리 연산자로 연결된 하나 이상의 조건을 기술하는 대신, **IN 연산자를 사용**하면 두 개 이상의 열을 한꺼번에 비교할 수 있게 함

<u>메인쿼리</u> : 대표이사 200		
<u>서브쿼리</u> :	부장	100
	대표이사	200
	경리	300



5. 다중 열 서브쿼리

★ 다중 열 서브쿼리의 형식

```
SELECT    선택리스트1
FROM      테이블 이름1
WHERE     (열이름1, 열이름2, ...) IN
                                   (SELECT (열이름1, 열이름2, ...)
                                   FROM      테이블 이름2
                                   WHERE     조건);
```

- ✓ 서브쿼리가 반환한 두개 이상의 열을 한꺼번에 비교함

5. 다중 열 서브쿼리

✓ 다중 열 서브쿼리 사례

- 급여와 보너스가 200번 부서에 있는 어떤 사원의 급여 및 보너스와 같으면서 200번 부서에 근무하지 않는 모든 사원 검색하기

```
SELECT emp_name, dept_id, sal, bonus
FROM emp
WHERE (sal, NVL(bonus, -1)) IN
      (SELECT sal, NVL(bonus, -1)
       FROM emp
       WHERE dept_id = 200)
AND dept_id <> 200;
```

2개 열의 값이 모두
같은 사원 검색

4행 2열 반환

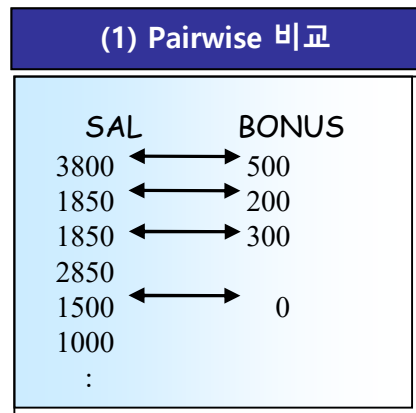
EMP_NAME	DEPT_ID	SAL	BONUS
이문정	100	2500	

SAL	NVL(BONUS,-1)
9000	-1
6500	-1
5000	-1
2500	-1

5. 다중 열 서브쿼리

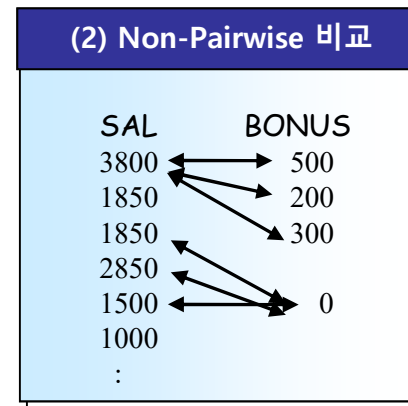
★ 다중 열 비교 방식

- 다중 열을 비교하는 방식은 Pairwise와 Non-Pairwise 방식으로 구분함



* Pairwise 비교란?

- 두 개 이상의 열이 쌍(Pair)을 이루어 비교되는 것
- 하나의 WHERE 절을 사용할 때 이런 비교 결과를 얻을 수 있음



* Non-Pairwise 비교란?

- 각각의 열이 별개로 비교되는 것
- 여러 개의 WHERE 절을 사용할 때 이런 비교 결과를 얻을 수 있음

5. 다중 열 서브쿼리

✓ Pairwise 비교 사례(1)

- 급여와 보너스가 모두 100번 부서에 있는 어떤 사원과 같으면서, 100번 부서에 근무하지 않는 모든 사원의 이름, 사원번호, 급여, 보너스 검색하기

```
SELECT emp_name, dept_id, sal, bonus
FROM emp
WHERE (sal, NVL(bonus, -1)) IN
      (SELECT sal, NVL(bonus, -1)
       FROM emp
       WHERE dept_id = 100)
AND dept_id <> 100;
```

2개 열의 값이 동시에
같은 사원 검색

2행 2열 반환

EMP_NAME	DEPT_ID	SAL	BONUS
김철수	200	2500	

SAL	NVL(BONUS,-1)
6200	-1
2500	-1

5. 다중 열 서브쿼리

✓ Pairwise 비교 사례(2)

- 각 부서별로 최소 급여를 받는 사원의 부서번호와 이름, 업무, 급여 검색하기

```
SELECT emp_name, dept_id, job, sal
FROM emp
WHERE (dept_id, sal) IN
      (SELECT dept_id, MIN(sal)
       FROM emp
       WHERE dept_id IS NOT NULL
       GROUP BY dept_id);
```

2개 열의 값이 동시에
같은 사원 검색

4행 2열 데이터 반환

EMP_NAME	DEPT_ID	JOB	SAL
진영진	400	세일즈	3500
이문정	100	사무직	2500
김마리아	300	사무직	2650
김철수	200	사무직	2500

DEPT_ID	MIN(SAL)
100	2500
400	3500
300	2650
200	2500

5. 다중 열 서브쿼리

✓ Non-Pairwise 비교 사례

- 급여는 200번 부서에 있는 어떤 사원과 같고, 업무는 300번 부서의 어떤 사원과 같으면서, 200과 300번 부서에 근무하지 않는 모든 사원의 이름, 부서번호, 급여, 그리고 업무 검색하기

```
SELECT emp_name, dept_id, sal, job
FROM emp
WHERE sal IN
      (SELECT sal
       FROM emp
       WHERE dept_id = 200)
AND job IN
      (SELECT job
       FROM emp
       WHERE dept_id = 300)
AND dept_id NOT IN (200, 300);
```

2개 조건을 각각 만족하는 모든 사원 검색

200번 부서의 급여(9000, 6500, 5000, 2500) 반환

300번 부서의 job(부장, 경리, 사무직) 반환

EMP_NAME	DEPT_ID	SAL	JOB
이문정	100	2500	사무직

6. 서브쿼리의 널 값 처리

★ 널 값 처리 방법

- Null 값은 여러 가지 오류의 원인이 될 수 있으며, 서브쿼리가 Null 값을 반환하는 경우에도 예기치 않은 결과를 얻을 수도 있으므로 주의해야 함
- ① 서브쿼리의 반환 값 중 하나라도 Null이면, **전체 질의의 결과는 Null**이 됨
 - Null 값을 비교하는 모든 조건은 Null이 되므로
- ② 서브쿼리 결과 집합의 일부라도 Null인 경우, **NOT IN 연산자를 사용하면 안 됨**
 - NOT IN 은 != ALL과 동일하므로
- ③ 서브쿼리 결과 집합의 일부가 Null이라도, **IN 연산자는 사용할 수 있음**
 - IN 은 = ANY와 동일하므로
- ④ 서브쿼리가 널 값을 반환할 가능성이 있는 경우, **NVL 함수를 이용해서 널 값을 다른 값으로 변환**시켜야 함

6. 서브쿼리의 널 값 처리

✓ 서브쿼리가 NULL 값을 반환하는 사례

- 부하 직원이 없는 모든 사원의 번호와 이름 검색하기

```
SELECT emp_id, emp_name
FROM emp
WHERE emp_id NOT IN
      (SELECT mgr_id
       FROM emp);
```

널 값은 NOT IN 연산자로 비교할 수 없음

두 명의 사원이 직속 상관이 없으므로 반환 값에 널 값 포함

no rows selected

선택된 레코드가 없음

MGR_ID	
	6200
	6321
	6321
	6200
	6200
	6351
	6200
	6361
	6321
	6311
	6351
	6361

14 rows selected.

6. 서브쿼리의 널 값 처리

➔ 오류 수정 : NVL 함수로 Null 값을 갖는 mgr_id를 9999로 변환

- 부하 직원이 없는 모든 사원의 번호와 이름 검색하기

```
SELECT emp_id, emp_name
FROM emp
WHERE emp_id NOT IN
      (SELECT NVL(mgr_id, 9999)
       FROM emp);
```

널 값을 9999로 변환해서
반환하게 됨

EMP_ID	EMP_NAME
7910	이영희
7901	정동길
7522	정종철
7854	진영진
7489	민동규
7878	백기수
7920	김마리아
7933	김철수
7872	이문정

9 rows selected.

6. 서브쿼리의 널 값 처리

✓ 서브쿼리가 반환한 NULL 값을 IN 연산자로 처리하는 사례

- 부하 직원이 있는 모든 사원의 번호와 이름 검색하기

```
SELECT emp_id, emp_name
FROM emp
WHERE emp_id IN (SELECT mgr_id
                  FROM emp);
```

일부가 널 값을 가져도
N 연산자는 사용 가능함

두 명의 사원이 직속 상관이
없으므로 널 값을 반환함

EMP_ID	EMP_NAME
6200	박철수
6321	이종길
6351	진대준
6361	김철수
6311	김대정

7. FROM절의 서브쿼리

★ FROM절에서 사용한 서브쿼리의 특징

- ① 검색할 테이블 이름을 반환하는 서브쿼리를 FROM절에 사용할 수 있음
- ② 서브쿼리는 실제 테이블의 일부인 가상 테이블을 반환함
- ③ 가상의 테이블인 뷰(View)를 사용하는 것과 유사함

7. FROM절의 서브쿼리

✓ FROM절의 서브쿼리 사용 사례

- 부서별로 각 부서의 평균 급여 이상을 받는 모든 사원의 이름, 급여, 부서번호, 그리고 부서 평균 검색하기

```
SELECT  a.emp_name, a.sal, a.dept_id, b.salavg
FROM    emp a, (SELECT dept_id, AVG(sal) salavg
                FROM    emp
                WHERE   dept_id IS NOT NULL
                GROUP BY dept_id) b
WHERE   a.dept_id = b.dept_id
AND     a.sal > b.salavg;
```

EMP_NAME	SAL	DEPT_ID	SALAVG
박철수	9000	200	5750
김대정	6200	100	4350
이종길	7000	400	4680
진대준	5850	300	3900
김철수	6500	200	5750



Q & A