

# 서브쿼리 (SUBQUERY)

## ▶ SUBQUERY

SELECT 문장 안에 포함된 또 다른 SELECT 문장으로 메인 쿼리가 실행되기 전 한 번만 실행됨  
비교 연산자의 오른쪽에 기술해야 하며 반드시 괄호로 묶어야 함  
서브쿼리와 비교할 항목은 반드시 서브쿼리의 SELECT한 항목의 개수와 자료형을 일치시켜야 함

### ✓ 예시

전 직원의 평균 급여보다 많은 급여를 받고 있는 직원의  
사번, 이름, 직급코드, 급여 조회

```
SELECT EMP_ID, EMP_NAME, JOB_CODE, SALARY
FROM EMPLOYEE
WHERE SALARY >= (SELECT AVG(SALARY) FROM EMPLOYEE);
```

	EMP_ID	EMP_NAME	JOB_CODE	SALARY
1	200	선동일	J1	8000000
2	201	송종기	J2	6000000
3	202	노용철	J2	3700000
4	204	유재식	J3	3400000
5	205	정중하	J3	3900000
6	209	심봉선	J3	3500000
7	215	대북훈	J5	3760000
8	217	전지연	J6	3660000

# ▶ SUBQUERY

## ✓ 유형

### 1. 단일행 서브쿼리

서브쿼리의 조회 결과 값의 개수가 1개인 서브쿼리

### 2. 다중행 서브쿼리

서브쿼리의 조회 결과 값의 행이 여러 개인 서브쿼리

### 3. 다중열 서브쿼리

서브쿼리의 조회 결과 컬럼의 개수가 여러 개인 서브쿼리

### 4. 다중행 다중열 서브쿼리

서브쿼리의 조회 결과 컬럼의 개수와 행의 개수가 여러 개인 서브쿼리

### 5. 상(호연)관 서브쿼리

서브쿼리가 만든 결과 값을 메인 쿼리가 비교 연산할 때

메인 쿼리 테이블의 값이 변경되면 서브쿼리의 결과 값도 바뀌는 서브쿼리

### 6. 스칼라 서브쿼리

상관쿼리이면서 결과 값이 한 개인 서브쿼리

# ▶ 단일 행 서브쿼리(SINGLE ROW SUBQUERY)

서브쿼리의 조회 결과값의 개수가 1개일 때  
단일행 서브쿼리 앞에는 일반 연산자 사용  
( < , > , <= , >= , = , != / <> / ^ = (서브쿼리) )

- 전 직원의 급여 평균보다 많은 급여를 받는  
직원의 이름, 직급, 부서, 급여 조회

```
SELECT EMP_NAME, JOB_CODE, DEPT_CODE, SALARY
FROM EMPLOYEE E
WHERE SALARY >= (SELECT AVG(SALARY)
                  FROM EMPLOYEE)
ORDER BY 2;
```

	EMP_NAME	JOB_CODE	DEPT_CODE	SALARY
1	선동일	J1	D9	8000000
2	송종기	J2	D9	6000000
3	노용철	J2	D9	3700000
4	유재식	J3	D6	3400000
5	정중하	J3	D6	3900000
6	심봉선	J3	D5	3500000
7	대복훈	J5	D5	3760000
8	전지연	J6	D1	3660000

	AVG(SALARY)
1	3047662.60869565217391304347826086956522

# ▶ 다중 행 서브쿼리(MULTI ROW SUBQUERY)

서브쿼리의 조회 결과 값의 행이 여러 개인 서브쿼리

- 부서 별 최고 급여를 받는 직원의 이름, 직급, 부서, 급여 조회

```
SELECT EMP_NAME, JOB_CODE, DEPT_CODE, SALARY
```

```
FROM EMPLOYEE
```

```
WHERE SALARY IN (SELECT MAX(SALARY)
```

```
FROM EMPLOYEE
```

```
GROUP BY DEPT_CODE)
```

```
ORDER BY 3;
```

\* 다중 행 서브쿼리 앞에는 일반 비교 연산자 사용 불가  
(사용 가능 연산자 : IN/NOT IN, >ANY/<ANY,  
>ALL/<ALL, EXIST/NOT EXIST 등)

	MAX(SALARY)		EMP_NAME	JOB_CODE	DEPT_CODE	SALARY
1	2890000	→	1 전지연	J6	D1	3660000
2	3660000	→	2 이종석	J4	D2	2490000
3	8000000	→	3 대북혼	J5	D5	3760000
4	3760000	→	4 정중하	J3	D6	3900000
5	3900000	→	5 장프위	J6	D8	2550000
6	2490000	→	6 선동일	J1	D9	8000000
7	2550000	→	7 미오리	J7	(null)	2890000

## ▶ 다중 열 서브쿼리

서브쿼리의 조회 결과 컬럼의 개수가 여러 개인 서브쿼리

- 퇴사한 여직원과 같은 부서, 같은 직급에 해당하는 사원의 이름, 직급, 부서, 입사일 조회

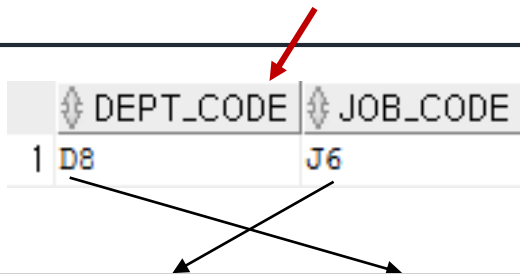
```
SELECT EMP_NAME, JOB_CODE, DEPT_CODE, HIRE_DATE
```

```
FROM EMPLOYEE
```

```
WHERE (DEPT_CODE, JOB_CODE) IN (SELECT DEPT_CODE, JOB_CODE
```

```
FROM EMPLOYEE
```

```
WHERE SUBSTR(EMP_NO, 8, 1)=2 AND ENT_YN='Y');
```



	DEPT_CODE	JOB_CODE
1	D8	J6

	EMP_NAME	JOB_CODE	DEPT_CODE	HIRE_DATE
1	장쯔위	J6	D8	15/06/17
2	전형돈	J6	D8	12/12/12

## ▶ 다중 행 다중 열 서브쿼리

서브쿼리의 조회 결과 컬럼의 개수와 행의 개수가 여러 개인 서브쿼리

- 직급별 최소 급여를 받는 직원의 사번, 이름, 직급, 급여 조회

```
SELECT EMP_ID, EMP_NAME, JOB_CODE, SALARY
FROM EMPLOYEE
WHERE (JOB_CODE, SALARY) IN (SELECT JOB_CODE, MIN(SALARY)
                             FROM EMPLOYEE
                             GROUP BY JOB_CODE)
ORDER BY 3;
```

EMP_ID	EMP_NAME	JOB_CODE	SALARY
1 200	선동일	J1	8000000
2 202	노웅철	J2	3700000
3 204	유재식	J3	3400000
4 219	임시환	J4	1550000
5 207	하미유	J5	2200000
6 211	전형돈	J6	2000000
7 214	방명수	J7	1380000

JOB_CODE	MIN(SALARY)
1 J2	3700000
2 J7	1380000
3 J3	3400000
4 J6	2000000
5 J5	2200000
6 J1	8000000
7 J4	1550000

## ▶ 상(호연)관 서브쿼리

서브쿼리가 만든 결과 값을 메인 쿼리가 비교 연산할 때  
메인 쿼리 테이블의 값이 변경되면 서브쿼리의 결과 값도 바뀌는 서브쿼리

- 관리자가 있는 사원들 중 관리자의 사번이 EMPLOYEE테이블에 존재하는  
직원의 사번, 이름, 소속 부서, 관리자 사번 조회

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, MANAGER_ID
FROM EMPLOYEE E
WHERE EXISTS (SELECT EMP_ID
              FROM EMPLOYEE M
              WHERE E.MANAGER_ID = M.EMP_ID);
```

EMP_ID	EMP_NAME	DEPT_CODE	MANAGER_ID
1 201	송종기	D9	200
2 202	노용철	D9	201
3 203	송은희	D6	204
4 204	유재식	D6	200
5 205	정중하	D6	204
6 206	박나라	D5	207
7 207	하미유	D5	200
8 208	김해술	D5	207
9 209	심봉선	D5	207
10 210	윤은해	D5	207
11 211	전형돈	D8	200
12 212	장프위	D8	211
13 214	방명수	D1	200
14 216	차태연	D1	214
15 217	전지연	D1	214



# ▶ 스칼라 서브쿼리

## ✓ SELECT절 예시

상관쿼리이면서 결과 값이 한 개인 서브쿼리

- 모든 사원의 사번, 이름, 관리자 사번, 관리자 명 조회

```
SELECT E.EMP_ID, E.EMP_NAME, E.MANAGER_ID,
       NVL((SELECT M.EMP_NAME
            FROM EMPLOYEE M
            WHERE E.MANAGER_ID = M.EMP_ID), '없음') AS 관리자명
FROM EMPLOYEE E
ORDER BY 1;
```

	EMP_ID	EMP_NAME	MANAGER_ID	관리자명
1	200	선동일	(null)	없음
2	201	송종기	200	선동일
3	202	노웅철	201	송종기
4	203	송은희	204	유재식
5	204	유재식	200	선동일
6	205	정중하	204	유재식
...				
21	220	이중석	(null)	없음
22	221	유하진	(null)	없음
23	222	이태림	100	없음

## ▶ 스칼라 서브쿼리

### ✓ WHERE절 예시

- 자신이 속한 직급의 평균 급여보다 많이 받는 직원의 이름, 직급, 급여 조회

```
SELECT EMP_NAME, JOB_CODE, SALARY
```

```
FROM EMPLOYEE E1
```

```
WHERE SALARY >= (SELECT AVG(SALARY)
```

```
FROM EMPLOYEE E2
```

```
WHERE E2.JOB_CODE = E1.JOB_CODE)
```

```
ORDER BY 2;
```

	EMP_NAME	JOB_CODE	SALARY
1	선동일	J1	8000000
2	송종기	J2	6000000
3	정중하	J3	3900000
4	송은희	J4	2800000
5	유하진	J4	2480000
6	이중석	J4	2490000
7	대북훈	J5	3760000
8	차태연	J6	2780000
9	전지연	J6	3660000
10	이오리	J7	2890000

# 스칼라 서브쿼리

## ✓ ORDER BY절 예시

- 모든 직원의 사번, 이름, 소속 부서를 조회 후 부서명 내림차순 정렬

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE
```

```
FROM EMPLOYEE
```

```
ORDER BY (SELECT DEPT_TITLE
```

```
FROM DEPARTMENT
```

```
WHERE DEPT_CODE = DEPT_ID)
```

```
DESC NULLS LAST;
```

\* **SELECT** DEPT\_ID, DEPT\_TITLE  
**FROM** DEPARTMENT  
**ORDER BY** DEPT\_TITLE DESC;

DEPT_ID	DEPT_TITLE
1 D2	회계관리부
2 D7	해외영업3부
3 D6	해외영업2부
4 D5	해외영업1부
5 D9	총무부
6 D1	인사관리부
7 D3	마케팅부
8 D8	기술지원부
9 D4	국내영업부

EMP_ID	EMP_NAME	DEPT_CODE
1 219	임시환	D2
2 220	이종석	D2
3 221	유하진	D2
4 204	유재식	D6
5 203	송은희	D6
6 205	정중하	D6
7 206	박나라	D5
8 207	하미유	D5
9 208	김해술	D5
10 209	심봉선	D5
11 210	윤은해	D5
12 215	대북혼	D5
13 202	노용철	D9
14 200	선동일	D9
15 201	송종기	D9
16 216	차태연	D1
17 217	전지연	D1
18 214	방명수	D1
19 212	장프위	D8
20 211	전형돈	D8
21 222	이태림	D8
22 213	하동운	(null)
23 218	이오리	(null)

# ▶ 인라인 뷰(INLINE-VIEW)

FROM절에 서브쿼리 사용한 것

## ✓ 예시

```
SELECT ROWNUM, EMP_NAME, SALARY
FROM EMPLOYEE
WHERE ROWNUM <= 5
ORDER BY SALARY DESC;
```

\* ROWNUM은 FROM절을 수행하면서 붙여지기 때문에 top-N분석 시 SELECT절에 사용한 ROWNUM이 의미 없게 됨

	ROWNUM	EMP_NAME	SALARY
1	1	선동일	8000000
2	2	송종기	6000000
3	3	노용철	3700000
4	5	유재식	3400000
5	4	송은희	2800000

```
SELECT ROWNUM, EMP_NAME, SALARY
FROM (SELECT *
      FROM EMPLOYEE
      ORDER BY SALARY DESC)
WHERE ROWNUM <= 5;
```

\* FROM절에 이미 정렬된 서브쿼리(인라인 뷰) 적용 시 ROWNUM이 top-N분석에 사용 가능

	ROWNUM	EMP_NAME	SALARY
1	1	선동일	8000000
2	2	송종기	6000000
3	3	정중하	3900000
4	4	대복혼	3760000
5	5	노용철	3700000

## ▶ WITH

서브쿼리에 이름을 붙여주고 인라인 뷰로 사용 시 서브쿼리의 이름으로 FROM절에 기술 가능  
같은 서브쿼리가 여러 번 사용될 경우 중복 작성을 피할 수 있고 실행속도도 빨라진다는 장점이 있음

### ✓ 예시

```
WITH TOPN_SAL AS (SELECT EMP_NAME, SALARY
                   FROM EMPLOYEE
                   ORDER BY SALARY DESC)

SELECT ROWNUM, EMP_NAME, SALARY
FROM TOPN_SAL;
```

ROWNUM	EMP_NAME	SALARY
1	선동일	8000000
2	송종기	6000000
3	정중하	3900000
4	대복훈	3760000
5	노용철	3700000
6	전지연	3660000
7	심봉선	3500000
8	유재식	3400000
9	미오리	2890000
10	송은희	2800000
11	차태연	2780000
12	장프위	2550000
13	김해술	2500000
14	이종석	2490000
15	유하진	2480000
16	이태림	2436240
17	하동운	2320000
18	하미유	2200000
19	전형돈	2000000
20	윤은해	2000000
21	박나라	1800000
22	임시환	1550000
23	방명수	1380000

## ▶ RANK() OVER

```
SELECT 순위, EMP_NAME, SALARY
FROM (SELECT EMP_NAME, SALARY,
      RANK() OVER(ORDER BY SALARY DESC) AS 순위
FROM EMPLOYEE
ORDER BY SALARY DESC);
```

	순위	EMP_NAME	SALARY
1	1	선동일	8000000
2	2	송종기	6000000
3	3	정중하	3900000
4	4	대복혼	3760000
5	5	노용철	3700000
6	6	전지연	3660000
7	7	심봉선	3500000
8	8	유재식	3400000
9	9	미오리	2890000
10	10	송은희	2800000
11	11	차태연	2780000
12	12	장프위	2550000
13	13	김해술	2500000
14	14	이종석	2490000
15	15	유하진	2480000
16	16	이태림	2436240
17	17	하동운	2320000
18	18	하미유	2200000
19	19	전형돈	2000000
20	19	윤은해	2000000
21	21	박나라	1800000
22	22	임시환	1550000
23	23	방명수	1380000

## ▶ DENSE\_RANK() OVER

```
SELECT 순위, EMP_NAME, SALARY
FROM (SELECT EMP_NAME, SALARY,
      DENSE_RANK() OVER(ORDER BY SALARY DESC) AS 순위
FROM EMPLOYEE
ORDER BY SALARY DESC);
```

	↕ 순위	↕ EMP_NAME	↕ SALARY
1	1	선동일	8000000
2	2	송종기	6000000
3	3	정중하	3900000
4	4	대북혼	3760000
5	5	노용철	3700000
6	6	전지연	3660000
7	7	심봉선	3500000
8	8	유재식	3400000
9	9	미오리	2890000
10	10	송은희	2800000
11	11	차태연	2780000
12	12	장프위	2550000
13	13	김해솔	2500000
14	14	이종석	2490000
15	15	유하진	2480000
16	16	이태림	2436240
17	17	하동운	2320000
18	18	하미유	2200000
19	19	전형돈	2000000
20	19	윤은해	2000000
21	20	박나라	1800000
22	21	임시환	1550000
23	22	방명수	1380000