**[ VI ] Sub Query**

**Sub Query는 단일행 서브쿼리 ; 서브쿼리의 결과가 단일 행**

**단일 행 연산자 : =, >, >=, <, <=, <>**

**다중행 서브쿼리 ; 서브쿼리의 결과가 2행 이상의 행**

**복수 행 연산자 : IN, NOT IN, ANY, SOME, ALL, EXISTS**

1. 서브 쿼리의 개념

* 회사에서 급여를 제일 많이 받는 사람의 이름과 급여
* SELECT ENAME, SAL FROM EMP

WHERE SAL = (SELECT MAX(SAL) FROM EMP); -- ( )부분 : 서브쿼리

1. 서브 쿼리는 하나의 SQL 문장의 절 안에 포함된 또 하나의 SELECT 문장입니다.
2. 그렇기에 서브 쿼리를 포함하고 있는 쿼리문을 메인 쿼리, 포함된 또 하나의 쿼리를 서브 쿼리라 합니다.
3. 서브 쿼리는 비교 연산자의 오른쪽에 기술하고 반드시 괄호로 둘러 쌓아야 합니다.
4. 서브 쿼리는 메인 쿼리가 실행되기 이전에 한번만 실행이 됩니다
5. 단일행 서브쿼리
   1. 단일 행(Single Row) 서브 쿼리는 수행 결과가 오직 하나의 로우(행, row)만을 반환하는 서브 쿼리를 갖는 것을 말합니다.
   2. 단일 행 서브 쿼리문에서는 이렇게 오직 하나의 로우(행, row)로 반환되는 서브 쿼리의 결과는 메인 쿼리에 보내게 되는데 메인 쿼리의 WHERE 절에서는 단일 행 비교 연산자인 =, >, >=, <, <=, <>를 사용해야 합니다
   3. 예시
      * SCOTT과 같은 부서에서 근무하는 사원의 이름과 부서번호를 출력
        + SELECT E2.ENAME, E2.DEPTNO FROM EMP E1, EMP E2 WHERE E1.ENAME='SCOTT' AND E1.DEPTNO=E2.DEPTNO;. –- Self Join이용하여 이름과 부서번호 출력
        + SELECT E2.ENAME, DNAME FROM EMP E1, EMP E2, DEPT WHERE E1.ENAME='SCOTT' AND E1.DEPTNO=E2.DEPTNO AND DEPT.DEPTNO=E1.DEPTNO; -- Self Join이용하여 이름과 부서명 출력
        + SELECT ENAME, DEPTNO FROM EMP WHERE DEPTNO=(SELECT DEPTNO FROM EMP WHERE ENAME='SCOTT'); -- 서브 쿼리 이용하여 이름과 부서번호 출력
        + SELECT ENAME, DNAME FROM EMP E, DEPT D WHERE E.DEPTNO=D.DEPTNO AND E.DEPTNO=(SELECT DEPTNO FROM EMP WHERE ENAME='SCOTT'); -- 서브쿼리 이용하여 이름과 부서명 출력
      * ‘SCOTT’와 같은 부서에 근무하는 사원들의 급여 총합을 출력
        + SELECT SUM(SAL) FROM EMP WHERE DEPTNO=(SELECT DEPTNO FROM EMP WHERE ENAME=’SCOTT’);
      * ‘SCOTT’과 같은 근무지에 근무하는 사원들의 이름
        + SELECT ENAME FROM EMP E, DEPT D

WHERE E.DEPTNO=D.DEPTNO AND

LOC=(SELECT LOC FROM EMP E, DEPT D

WHERE E.DEPTNO=D.DEPTNO AND ENAME='SCOTT')

* + - SCOTT과 동일한 JOB을 가진 사원의 모든 정보를 출력
      * SELECT \* FROM EMP WHERE JOB=(SELECT JOB FROM EMP WHERE ENAME='SCOTT');
    - SCOTT의 급여와 동일하거나 더 많이 받는 사원명과 급여를 출력
      * SELECT ENAME, SAL FROM EMP WHERE SAL>=(SELECT SAL FROM EMP WHERE ENAME='SCOTT');
    - DALLAS에서 근무하는 사원의 이름, 부서번호를 출력
      * SELECT ENAME, DEPTNO FROM EMP WHERE DEPTNO=(SELECT DEPTNO FROM DEPT WHERE LOC='DALLAS');
      * SELECT ENAME, E.DEPTNO FROM EMP E, DEPT D WHERE E.DEPTNO=D.DEPTNO AND LOC='DALLAS'; -- join을 이용해도 된다
    - SALES 부서에서 근무하는 모든 사원의 이름과 급여를 출력
      * SELECT ENAME, SAL FROM EMP WHERE DEPTNO=(SELECT DEPTNO FROM DEPT WHERE DNAME='SALES');
      * SELECT ENAME, SAL FROM EMP E, DEPT D WHERE E.DEPTNO=D.DEPTNO AND DNAME='SALES'; -- join을 이용해도 된다
    - 직속상관이 KING인 사원의 이름과 급여를 출력
      * SELECT ENAME, SAL FROM EMP WHERE MGR=(SELECT EMPNO FROM EMP WHERE ENAME='KING');
      * SELECT W.ENAME, W.SAL

FROM EMP W, EMP M

WHERE W.MGR=M.EMPNO AND M.ENAME='KING';

* 1. 서브 쿼리에서 그룹함수의 사용
     + 사원들의 평균 급여 이하를 받는 사원의 사원번호와 이름, 급여를 출력
       - SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<=(SELECT AVG(SAL) FROM EMP);
  2. 다중열 서브쿼리도 가능하다.
     + 다중열서브쿼리도 가능합니다 EX. SCOTT과 JOB이 같고, SCOTT과 부서번호가 같은 사람의 모든 정보를 출력
       - SELECT JOB, COMM FROM EMP WHERE ENAME=’SCOTT’;--서브쿼리
       - SELECT \* FROM EMP WHERE (JOB, DEPTNO) = (SELECT JOB, DEPTNO FROM EMP WHERE ENAME=’SCOTT’);

1. 다중행 서브쿼리
   1. 다중 행 서브 쿼리는 서브쿼리에서 반환되는 결과가 하나 이상의 행일 때 사용하는 서브 쿼리입니다. 다중 행 서브 쿼리는 반드시 다중 행 연산자(Multiple Row Operator)와 함께 사용해야 합니다.
   2. 다중행 연산자의 종류
      * IN : 메인 쿼리의 비교 조건(‘=’ 연산자로 비교할 경우)이 서브 쿼리의 결과 중에서 하나라도 일치하면 참
      * ANY, SOME : 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
      * EXISTS : 메인 쿼리의 비교 조건이 서브 쿼리의 결과 중에서 만족하는 값이 하나라도 존재하면 참
   3. 예제
      * IN ; (EX) 급여가 3000 이상 받는 사원이 소속된 부서(10번, 20번)와 동일한 부서에서 근무하는 사원들의 모든 정보를 출력하는 SQL문
        + 급여가 3000이상 받는 사원들이 소속된 부서를 출력하기 위해서는 SELECT DEPTNO FROM EMP WHERE SAL>=3000;이라는 SQL문을 작성하므로 위의 예제에 맞는 SQL문은

SELECT \* FROM EMP WHERE DEPTNO=(SELECT DEPTNO FROM EMP WHERE SAL>=3000);라고 생각되지만 서브쿼리의 결과가 2개 이상의 행이 리턴 되어 **에러**를 만난다.

* + - * =, >, >=, <, <=, <>은 단일행 서브쿼리문에서 함께 사용되는 단일행 비교연산자이다

결과가 2행이상 구해지는 서브쿼리문에서는 다중행 연산자와 함께 사용해야 한다.

SELECT \* FROM EMP WHERE DEPTNO IN (SELECT DEPTNO FROM EMP WHERE SAL>=3000);

* + - 다중 열 서브쿼리도 가능하다
      * 부서별로 입사일이 가장 늦은 사람이 부서번호, 이름, 입사일을 출력하시오

SELECT DEPTNO, MAX(HIREDATE) FROM EMP GROUP BY DEPTNO;

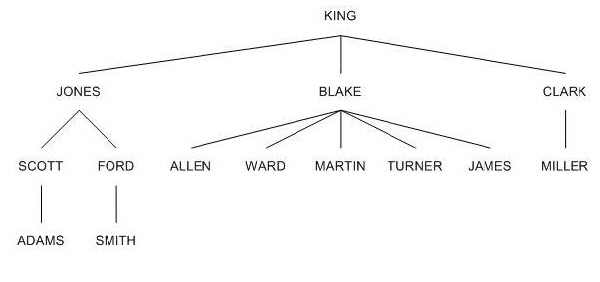
SELECT ENAME, HIREDATE FROM EMP WHERE (DEPTNO, HIREDATE) IN (SELECT DEPTNO, MAX(HIREDATE) FROM EMP GROUP BY DEPTNO);

* + - * JOB(직책별)로 가장 낮은 연봉을 받는 사람의 이름, 직책, 연봉을 출력
        + SELECT ENAME, JOB, SAL FROM EMP WHERE (JOB, SAL) IN (SELECT JOB, MIN(SAL) FROM EMP GROUP BY JOB);
    - ALL (EX) 30번 소속 사원들 중에서 급여를 가장 많이 받는 사원보다 더 많은 급여를 받는 사람의 이름, 급여를 출력하는 SQL문
      * SELECT ENAME, SAL FROM EMP WHERE SAL > ALL(SELECT SAL FROM EMP WHERE DEPTNO=30); -- SELECT SAL SAL FROM EMP WHERE DEPTNO=30; 의 모든값보다 큼을 의미
      * SELECT ENAME, SAL FROM EMP WHERE SAL > (SELECT MAX(SAL) FROM EMP WHERE DEPTNO=30);와 같은 의미이다
    - ANY : ANY 조건은 메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 하나 이상만 일치하면 참. ANY는 찾아진 값에 대해서 하나라도 크면 참이 되는 셈이 됩니다. 그러므로 찾아진 값 중에서 가장 작은 값 즉, 최소값 보다 크면 참
    - ANY (EX) 부서번호가 30번인 사원들의 급여 중 가장 작은 값(950)보다 많은 급여를 받는 사원의 이름, 급여를 출력하는 SQL문
      * SELECT ENAME, SAL FROM EMP WHERE SAL > ANY(SELECT SAL FROM EMP WHERE DEPTNO=30);
      * SELECT ENAME, SAL FROM EMP WHERE SAL > (SELECT MIN(SAL) FROM EMP WHERE DEPTNO=30);와 같은 의미이다
    - EXISTS : SQL문에서 Exists를 사용하는 것은 어떤 조건을 만족하는 집합의 존재 여부를 확인할 경우에 사용 (EX) 직속부하가 있는 사원의 사원번호, 이름, 급여를 출력
      * SELECT EMPNO, ENAME FROM EMP WHERE EMPNO IN (SELECT MGR FROM EMP);
      * SELECT empno, ename, sal FROM EMP manager

WHERE EXISTS (SELECT empno FROM EMP worker WHERE manager.empno=worker.mgr);

* + - * SELECT empno, ename, sal FROM EMP manager

WHERE EXISTS (SELECT empno FROM EMP WHERE manager.empno=mgr);



* + - 사번, 이름, 급여, 부서번호, 해당사원부서의 평균급여
      * SELECT EMPNO, ENAME, SAL, DEPTNO, (SELECT AVG(SAL) FROM EMP WHERE E.DEPTNO=DEPTNO) FROM EMP E;
  1. 탄탄다지기 (단일행, 다중행)

-- 탄탄1. 부서별로 가장 급여를 많이 받는 사원의 모든 정보를 출력(IN 연산자 이용)

* + - * SELECT \* FROM EMP WHERE (SAL) IN (SELECT MAX(SAL) FROM EMP GROUP BY DEPTNO);
      * 위를 보완하자
      * SELECT \* FROM EMP WHERE (DEPTNO, SAL) IN (SELECT DEPTNO, MAX(SAL) FROM EMP GROUP BY DEPTNO);

-- 탄탄2. 직급(JOB)이 MANAGER인 사람의 속한 부서의 부서 번호와 부서명과 지역을 출력(IN)

-- 탄탄3. 급여가 3000이상인 사람들 중 연봉 등급을 나누어서 해당 등급별 최고 급여를 받는 사람들의 사번, 이름, 직업, 입사일, 급여, 급여등급을 출력

-- 탄탄4. 응용심화 : 입사일 분기별로 가장 높은 급여를 받는 사람들의 분기, 사번, 이름, JOB, 상사사번, 입사일, 급여, 상여를 출력하세요

-- 탄탄5. 급여가 3000미만인 사람 중에 가장 최근에 입사한 사람의 사원번호와 이름, 급여, 입사일을 출력

-- 탄탄6. SALESMAN 모든 사원들 보다 급여를 많이 받는 사원들의 이름과 급여와 직급(담당 업무)를 출력하되 영업 사원은 출력하지 않는다.(ALL이용)

-- 탄탄7. SALESMAN 일부 어떤 한 사원보다 급여를 많이 받는 사원들의 이름과 급여와 직급(담당 업무)를 출력하되 영업 사원도 출력(ANY)

-- ★ 총 연습문제

--1. 사원테이블에서 가장 먼저 입사한 사람의 이름, 급여, 입사일

-- 2. 회사에서 가장 급여가 적은 사람의 이름, 급여

-- 3. 회사 평균보다 급여를 많이 받는 사람의 이름, 급여, 부서코드

--4. 회사 평균 이하의 급여를 받는 사람의 이름, 급여, 부서명

--5. SCOTT보다 먼저 입사한 사람의 이름, 급여, 입사일, 급여 등급

--6. 5번(SCOTT보다 먼저 입사한 사람의 이름, 급여, 입사일, 급여 등급)에 부서명 추가하고 급여가 큰 순 정렬

--7. BLAKE 보다 급여가 많은 사원들의 사번, 이름, 급여

--8. MILLER보다 늦게 입사한 사원의 사번, 이름, 입사일

--9. 사원전체 평균 급여보다 급여가 많은 사원들의 사번, 이름, 급여

--10. CLARK와 같은 부서번호이며, 사번이 7698인 직원의 급여보다 많은 급여를 받는 사원의 사번, 이름, 급여

--11. 응용심화. CLARK와 같은 부서명이며, 사번이 7698인 직원의 급여보다 많은 급여를 받는 사원의 사번, 이름, 급여

--12. BLAKE와 같은 부서에 있는 모든 사원의 이름과 입사일자

--13. 평균 급여 이상을 받는 모든 종업원에 대해서 사원번호와 이름 단 급여가 많은 순으로 출력)

**-- 여기서부터는 다중행서브쿼리와 단일행서브쿼리, 이전은 단일행서브쿼리**

-- 14. 이름에 “T”가 있는 사원이 근무하는 부서에서 근무하는 모든 직원의 사원 번호,이름,급여(단 사번 순 출력)

-- 15. 부서 위치가 Dallas인 모든 종업원에 대해 이름,업무,급여

-- 16. EMP 테이블에서 King에게 보고하는 모든 사원의 이름과 급여

-- 17. SALES부서 사원의 이름, 업무

-- 18. 월급이 부서 30의 최저 월급보다 높은 사원의 모든 필드

-- 19. FORD와 업무도 월급도 같은 사원의 모든 필드

-- 20. 이름이 JONES인 직원의 JOB과 같거나 FORD의 SAL 이상을 받는 사원의 정보를 이름, 업무, 부서번호, 급여

-- 단, 업무별 알파벳 순, 월급이 많은 순으로 출력

-- 21. SCOTT 또는 WARD와 월급이 같은 사원의 정보를 이름,업무,급여

-- 22. CHICAGO에서 근무하는 사원과 같은 업무를 하는 사원들의 이름,업무

-- 23. **부서 평균 월급**보다 월급이 높은 사원을 사번, 이름, 급여, 부서번호

-- 24. **업무별로 평균 월급**보다 적은 월급을 받는 사원을 부서번호, 이름, 급여

-- 25. 적어도 한 명 이상으로부터 보고를 받을 수 있는 사원을 업무, 이름, 사번, 부서번호를 출력(단, 부서번호 순으로 오름차순 정렬)

-- 26. 말단 사원의 사번, 이름, 업무, 부서번호