033

DB

SUBQUERY (서브 쿼리)

- 하나으 SQL안에 포함된 또다른 SQL(SELECT ONLY)
- 메인쿼리(기존 쿼리)를 보고하기 위해 사용
- SELECT , FROM ,WHER,HAVING절에서 사용가능

서브쿼리 유형

- 단일행(+단일열) 서브쿼리 : 서브쿼리의 조회 결과 값의 개수가 1개일 때
- 다중행(+단일열) 서브쿼리 : 서브쿼리의 조회 결과 값의 개수가 여러개일 때
- 다중열 서브쿼리 : 서브쿼리의 SELECT 절에 자열된 항목수가 여러개 일 때
- 다중행 다중열 서브쿼리 : 조회 결과 행 수와 열 수가 여러개일 때 -
- 상관 서브쿼리 : 서브쿼리가 만든 결과 값을 메인 쿼리가 비교 연산할 때 메인 쿼리 테이블의 값이 변경되면 서브쿼리의 결과값도 바뀌는 서브쿼리
- 스칼라 서브쿼리 : 상관 쿼리이면서 결과 값이 하나인 서브쿼리

  [서브쿼리 유형에 따라 서브쿼리 앞에 붙은 연산자가 다름]

서브 쿼리 전용 함수

- EXISTS 결과 중 해당하는 행이 1개라도 존재하면 조회결과에 포함 시킨다.

단일행 쿼리

- 한행 한열

- 하나의 컬럼값만 비교한다.

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

Query Result | SQL | All Rows Fetched: 8 in 0.008 seconds

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

- 결과값이 '한개' 만 조회되는 SELECT절을 만들어 비교한다
-

다중행 쿼리

- 서브쿼리 조회 값이 여러개의 행
- 일반
- 일반 비교연산자 사용(> , < , >= , <= , != ,=) 불가

```
SELECT MAX(SALARY)
FROM EMPLOYEE
GROUP BY DEPT_CODE; --7행 1열
```

Query Result | SQL | All Rows Fetched: 7 in 0.004 seconds

	MAX(SALARY)
1	8000000
2	3900000
3	3760000
4	2550000
5	2890000
6	3660000
7	2490000

- 하나의 열 여러개의 행이 나옴 = **다중 행**

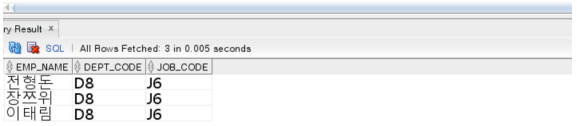
- IN / NOT IN : 여러 개의 결과값 중에서 한 개라도 일치하는 값이 있다
- > ANY, < ANY : 여러개의 결과값 중에서 한개라도 큰 / 작은 경우
- > ALL, < ALL : 여러개의 결과값의 모든 값보다 큰 / 작은 경우
- EXISTS / NOT EXISTS : 값이 존재하는가? / 존재하지 않는가

다중열 쿼리

- 열이 여러개 행은 한개

	EMP_NAME	JOB_CODE	DEPT_CODE	HIRE_DATE
1	이태림	J6	D8	1997-09-12 00:00:00

```
SELECT EMP_NAME,DEPT_CODE,JOB_CODE
FROM EMPLOYEE
WHERE (DEPT_CODE,JOB_CODE) = (
                        SELECT DEPT_CODE , JOB_CODE --다중으로 열을
                        FROM EMPLOYEE
                        WHERE SUBSTR(EMP_NO,8,1) IN (2,4)
                        AND ENT_YN='Y'
                        );
```



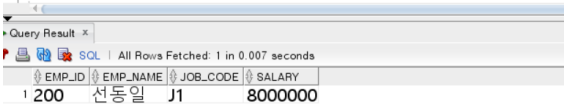
EMP_NAME	DEPT_CODE	JOB_CODE
전형도	D8	J6
장쯔위	D8	J6
이태림	D8	J6

하나의 행만 조회해서 그 행을 기준으로 조회할 수 있다

다중행 열 쿼리

- 다중행과 다중열 쿼리를 동시에 사용

```
SELECT EMP_ID,EMP_NAME,JOB_CODE,SALARY
FROM EMPLOYEE
WHERE (JOB_CODE,SALARY)  IN(
                        SELECT JOB_CODE,TRUNC(AVG(SALARY),-4)
                        FROM EMPLOYEE
                        GROUP BY JOB_CODE
                        );
```



EMP_ID	EMP_NAME	JOB_CODE	SALARY
1 200	선동일	J1	8000000

조회 결과와 행이 여러 개 일 때

- 직급과 직급별 평균 월급을 구한 뒤
- 직급 코드와 월급을 대조한 후
- IN : 지금 조회하는 사람과 직급 코드와 월급이 포함 될 경우 조회
  - THEN 서브 쿼리에서 조회한 직급과 해당 직급의 평균 월급이 동일할 경우 조회

상호연관 서브 쿼리

- 메인 쿼리의 테이블 값을 이용해 서브쿼리를 이용한다.

```
-- 사수가 있는 직원의 사번, 이름, 부서명, 사수사번 조회
-- EXISTS : 서브쿼리에 해당하는 행이 1개라도 존재하면 조회결과에 포함시킨다.
SELECT EMP_ID,EMP_NAME,DEPT_TITLE,MANAGER_ID
FROM EMPLOYEE MAIN
LEFT JOIN DEPARTMENT ON (DEPT_CODE = DEPT_ID)
WHERE EXISTS(SELECT EMP_ID FROM EMPLOYEE SUB
             WHERE MAIN.MANAGER_ID = SUB.EMP_ID);
```

- 메인쿼리에서 얻어온 메인쿼리에서 조회하는 사원의 매니저의 사번이 서브쿼리 안에서 행에 반복 접근하면서 조회한 사번과 같다면
- 사수가 있다는 의미임으로 메인쿼리에서 출력

- 이때 EXISTS는 값이 한개 이상있으면 조회한다는 의미 (약간 BOOLEAN 느낌..?)

```
SELECT EMP_ID,EMP_NAME,NVL(DEPT_TITLE,'소속없음'),JOB_NAME,TO_CHAR(HIRE_DATE,'YYYY-MM-DD'),DE
FROM EMPLOYEE E
LEFT JOIN DEPARTMENT ON(DEPT_CODE = DEPT_ID)
JOIN JOB USING(JOB_CODE)
WHERE ENT_YN = 'N' AND
    HIRE_DATE = (SELECT MIN(HIRE_DATE) FROM EMPLOYEE M
    WHERE  E.DEPT_CODE = M.DEPT_CODE
    )
ORDER BY HIRE_DATE;
```

SQL | All Rows Fetched: 5 in 0.024 seconds

	EMP_ID	EMP_NAME	NVL(DEPT_TITLE,'소속없음')	JOB_NAME	TO_CHAR(HIRE_DATE,'YYYY-MM-DD')	DEPT_CODE
1	200	선동일	총무부	대표	1990-02-06	D9
2	221	유하진	회계관리부	차장	1994-01-20	D2
3	207	하이유	해외영업1부	과장	1994-07-07	D5
4	203	송은희	해외영업2부	차장	1996-05-03	D6
5	217	전지연	인사관리부	대리	2007-03-20	D1

- 메인쿼리의 부서번호와 서브쿼리의 부서번호가 같을 경우
- 가장 적은 입사일을 조회한다 이때 메인쿼리의 입사일과 같을 경우
- 가장 빠른 입사일이라는 의미임으로 조회

스칼라 쿼리

- SELECT절에서 서브쿼리 결과로 1행만 반환 (SELECT에서 사용되는 단일 행 서브쿼리)

```
-- 각 직원들이 속한 직급의 급여 평균 조회
SELECT EMP_NAME,SALARY,JOB_CODE 직급,
    (SELECT FLOOR(AVG(SALARY))
    FROM EMPLOYEE SUB
    WHERE SUB.JOB_CODE=MAIN.JOB_CODE)평균
FROM EMPLOYEE MAIN
ORDER BY 3;
```

Query...

SQL | All Rows Fetched: 23 in 0.014 seconds

	EMP_NAME	SALARY	직급	평균
1	선동일	8000000	J1	8000000
2	송종기	6000000	J2	4850000
3	노옹철	3700000	J2	4850000
4	유재식	3400000	J3	3600000
5	정중하	3900000	J3	3600000
6	심봉선	3500000	J3	3600000
7	송은희	2800000	J4	2330000
8	임시환	1550000	J4	2330000
9	이중석	2490000	J4	2330000
10	유하진	2480000	J4	2330000
11	하이유	2200000	J5	2820000
12	김해술	2500000	J5	2820000

- 모든 해석이 마무리 된 후 SELECT해석 차례에서
- 서브쿼리의 직급 코드와 서브쿼리의 직급 코드가 같을 경우

- 직급의 평균을 조회한다

인라인 뷰(INLINE VIEW) : 행마다 1숫자를 1씩 추가한다.

- FROM절에서 서브쿼리를 사용하는 경우
- 서브쿼리가 만든 RESULTSET을 테이블 대신 사용한다

```
SELECT ROWNUM , EMP_NAME,SALARY
FROM
   (SELECT EMP_NAME,SALARY
   FROM EMPLOYEE
   ORDER BY SALARY DESC)--> FROM절 내부에 포함된 가상의 테이블(INLINE VIEW)

WHERE ROWNUM <=5;
```

  - 서브쿼리에서 이름과 월급을 조회 후 월급 내림차순으로 조회한다
  - 그 테이블을 조회 후 앞 ROWNUM을 이용해 순위를 구한다

가상의 테이블 == VIEW

위에서 사용된 것 처럼 FROM안에 사용하는 서브쿼리로 조회한 가상의 테이블

WITH : 자바에서 변수같은 역할 뷰를 미리 만들어 둔다

- 서브쿼리에 이름을 부여하고 변수처럼 사용 가능하다
- 인라인뷰로 사용시에 자주 사용

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

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

- 조회 할 SELECT문 위에서 우선 적으로
- TOP_SAL 이라는 가상의 테이블을 선언 후
- 메인 쿼리에서 별칭만 호출해 사용한다
  - 이 때 서브쿼리에서 ; (세미콜론)은 절대 금지

RANK( ) OVER / DENSE_RANK ( ) OVER :순위 매기는 함수 , 랭크 함수

- 표기법에 차이가 있다.
- ROWNUM으로 순위를 사용하지 않고
- 순위만 사용하고 싶을 때 사용한다

```sql
-- RANK() OVER : 동일한 순위 이후의 등수를 동일한 인원 수 만큼 건너뛰고 순위 계산
--               EX) 공동 1위가 2명이면 다음 순위는 2위가 아니라 3위

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

-- DENSE_RANK() OVER : 동일한 순위 이후의 등수를 이후의 순위로 계산
--                     EX) 공동 1위가 2명이어도 다음 순위는 2위
SELECT DENSE_RANK()OVER(ORDER BY SALARY DESC)순위 ,EMP_NAME 이름,SALARY 이름
FROM EMPLOYEE;
```

RANK : 동 순위 있을 경우 동일한 인원수 만큼 건너뛰고 계산

DENSE RANK : 동 순위 있을 경우 계산 안하고 그냥 정순대로 조회