/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

서브쿼리(Sub Query)

- 쿼리안에서 select 쿼리를 사용하는 것.

- 메인 쿼리 ⊃ 서브쿼리

- 반드시 소괄호 ( ) 로 묶어야한다.

서브쿼리가 사용되는 구

- select절, from절, where절, having절

서브쿼리의 종류

- 어느 구절에 사용되었는지에 따른 구분

- 스칼라 서브쿼리 - select 절에 사용. 반드시 서브쿼리 결과가 1행 1열(값 하나-스칼라) 0행이 조회되면 null을 반환

- 인라인 뷰 - from 절에 사용되어 테이블의 역할을 한다.

- 서브쿼리 조회결과 행수에 따른 구분

- 단일행 서브쿼리 - 서브쿼리의 조회결과 행이 한행인 것.

- 다중행 서브쿼리 - 서브쿼리의 조회결과 행이 여러행인 것.

- 동작 방식에 따른 구분

- 비상관(비연관) 서브쿼리 - 서브쿼리에 메인쿼리의 컬럼이 사용되지 않는다. 메인쿼리에 사용할 값을 서브쿼리가 제공하는 역할을 한다.

- 상관(연관) 서브쿼리 - 서브쿼리에서 메인쿼리의 컬럼을 사용한다.

메인쿼리가 먼저 수행되어 읽혀진 데이터를 서브쿼리에서 조건이 맞는지 확인하고자 할때 주로 사용한다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

-- 직원\_ID(emp.emp\_id)가 120번인 직원과 같은 업무(emp.job\_id)가진

-- 직원의 id(emp\_id),이름(emp.emp\_name), 업무(emp.job\_id), 급여(emp.salary) 조회

select emp\_id, emp\_name, job\_id, salary

from emp

where job\_id = (select job\_id from emp where emp\_id = 120);

--subquery는 ( )로 묶어줘야 한다.

-- 직원\_id(emp.emp\_id)가 115번인 직원과 같은 업무(emp.job\_id)를 하고 같은 부서(emp.dept\_id)에 속한 직원들을 조회하시오.

select \* from emp

where job\_id = (select job\_id from emp where emp\_id = 115)

and dept\_id = (select dept\_id from emp where emp\_id = 115);

--위 방법은 매우 비효율적 아래 방법 사용 ( Pair 방식 서브쿼리)

select \* from emp

where (job\_id, dept\_id) = (select job\_id, dept\_id from emp where emp\_id = 115);

select \* from emp

where (job\_id, dept\_id) = ('PU\_MAN', 30); --pair방식은 서브쿼리에서만 작동

-- 직원들 중 급여(emp.salary)가 전체 직원의 평균 급여보다 적은

-- 직원들의 id(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary)를 조회. 급여(emp.salary) 오름차순 정렬.

select emp\_id, emp\_name, salary

from emp

where salary < (select avg(salary) from emp) --avg=6517.xxxx

order by salary;

-- 전체 직원의 평균 급여(emp.salary) 이상을 받는 부서의 이름(dept.dept\_name), 소속직원들의 평균 급여(emp.salary) 출력.

-- 평균급여는 소숫점 2자리까지 나오고 통화표시($)와 단위 구분자 출력

select d.dept\_name, to\_char(avg(e.salary),'$999,999.99') 평균급여

from emp e left join dept d on e.dept\_id = d.dept\_id

group by d.dept\_name

having avg(e.salary) >= (select avg(salary) from emp);

-- TODO: 직원의 ID(emp.emp\_id)가 145인 직원보다 많은 연봉을 받는 직원들의 이름(emp.emp\_name)과 급여(emp.salary) 조회.

-- 급여가 큰 순서대로 조회

select emp\_name, salary from emp

where salary>(select salary from emp where emp\_id=145)

order by salary desc;

-- TODO: 직원의 ID(emp.emp\_id)가 150인 직원과 같은 업무(emp.job\_id)를 하고 같은 상사(emp.mgr\_id)를 가진 직원들의

-- id(emp.emp\_id), 이름(emp.emp\_name), 업무(emp.job\_id), 상사(emp.mgr\_id) 를 조회

select emp\_id, emp\_name, job\_id, mgr\_id from emp

where (mgr\_id,job\_id) = (select mgr\_id, job\_id from emp where emp\_id = 150);

-- TODO : EMP 테이블에서 직원 이름이(emp.emp\_name)이 'John'인 직원들 중에서 급여(emp.salary)가 가장 높은 직원의 salary(emp.salary)보다 많이 받는

-- 직원들의 id(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary)를 직원 ID(emp.emp\_id) 오름차순으로 조회.

select emp\_id, emp\_name, salary from emp

where salary > (select max(salary) from emp where emp\_name ='John')

order by 1;

-- TODO: 급여(emp.salary)가 가장 높은 직원이 속한 부서의 이름(dept.dept\_name), 위치(dept.loc)를 조회.

select d.dept\_name, d.loc from emp e left join dept d on e.dept\_id = d.dept\_id

where salary = (select max(salary) from emp);

-- TODO: 30번 부서(emp.dept\_id) 의 평균 급여(emp.salary)보다 급여가 많은 직원들의 모든 정보를 조회.

select \* from emp

where salary>(select avg(salary) from emp where dept\_id = 30);

-- TODO: 담당 업무ID(emp.job\_id) 가 'ST\_CLERK'인 직원들의 평균 급여보다 적은 급여의 직원를 받는 직원들의 모든 정보를 조회. 단 업무 ID가 'ST\_CLERK'이 아닌 직원들만 조회.

select \* from emp

where salary < (select avg(salary) from emp where job\_id = 'ST\_CLERK')

and Job\_id <> 'ST\_CLERK';

-- TODO: 급여(emp.salary)를 제일 많이 받는 직원들의 이름(emp.emp\_name), 부서명(dept.dept\_name), 급여(emp.salary) 조회.

-- 급여는 앞에 $를 붙이고 단위구분자 , 를 출력

select e.emp\_name, d.dept\_name, to\_char(e.salary, '$999,999') 급여 from emp e left join dept d on e.dept\_id = d.dept\_id --부서가 없는 직원일 경우도 있다

where e.salary = (select max(salary) from emp);

-- TODO: EMP 테이블에서 업무(emp.job\_id)가 'IT\_PROG' 인 직원들의 평균 급여 이상을 받는

-- 직원들의 id(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary)를 급여 내림차순으로 조회.

select emp\_id, emp\_name, salary from emp

where salary >= (select avg(salary) from emp where job\_id = 'IT\_PROG')

order by salary desc;

-- TODO: 'IT' 부서(dept.dept\_name)의 최대 급여보다 많이 받는 직원의 ID(emp.emp\_id), 이름(emp.emp\_name), 입사일(emp.hire\_date), 부서 ID(emp.dept\_id), 급여(emp.salary) 조회

-- 입사일은 "yyyy년 mm월 dd일" 형식으로 출력

-- 급여는 앞에 $를 붙이고 단위구분자 , 를 출력

select emp\_id, emp\_name, to\_char(hire\_date,'yyyy"년" mm"월" dd"일"')hire\_date, dept\_id, to\_char(salary,'$999,999')salary from emp

where salary > (select max(e.salary) from emp e join dept d on e.dept\_id=d.dept\_id where dept\_name = 'IT');

/\* ----------------------------------------------

다중행 서브쿼리

- 서브쿼리의 조회 결과가 여러행인 경우

- where절 에서의 연산자

- in

- 비교연산자 any : 조회된 값들 중 하나만 참이면 참 (where 컬럼 > any(서브쿼리) ) 서브의 min보다 크다

- 비교연산자 all : 조회된 값들 모두와 참이면 참 (where 컬럼 > all(서브쿼리) ) 서브의 max보다 크다

------------------------------------------------\*/

--'Alexander' 란 이름(emp.emp\_name)을 가진 관리자(emp.mgr\_id)의

-- 부하 직원들의 ID(emp\_id), 이름(emp\_name), 업무(job\_id), 입사년도(hire\_date-년도만출력), 급여(salary)를 조회

-- 급여는 앞에 $를 붙이고 단위구분자 , 를 출력

select emp\_id, emp\_name, job\_id, to\_char(hire\_date,'yyyy')hire\_date, salary from emp

where mgr\_id in (select emp\_id from emp where emp\_name = 'Alexander');

-- 직원 ID(emp.emp\_id)가 101, 102, 103 인 직원들 보다 급여(emp.salary)를 많이 받는 직원의 모든 정보를 조회.

select \* from emp

where salary > all (select salary from emp where emp\_id in (101,102,103));

--같음(all = max와 일맥상통함)

select \* from emp

where salary > (select max(salary) from emp where emp\_id in (101,102,103));

-- 직원 ID(emp.emp\_id)가 101, 102, 103 인 직원들 중 급여가 가장 적은 직원보다 급여를 많이 받는 직원의 모든 정보를 조회.

select \* from emp

where salary > any (select salary from emp where emp\_id in (101,102,103));

--같음(any = min과 일맥상통)

select \* from emp

where salary > (select min(salary) from emp where emp\_id in (101,102,103));

-- TODO : 부서 위치(dept.loc) 가 'New York'인 부서에 소속된 직원의 ID(emp.emp\_id), 이름(emp.emp\_name), 부서\_id(emp.dept\_id) 를 sub query를 이용해 조회.

select emp\_id, emp\_name, dept\_id

from emp

where dept\_id in (select dept\_id from dept where loc = 'New York');

-- TODO : 최대 급여(job.max\_salary)가 6000이하인 업무를 담당하는 직원(emp)의 모든 정보를 sub query를 이용해 조회.

select \* from emp

where job\_id in (select job\_id from job where max\_salary <=6000);

-- TODO: 부서\_ID(emp.dept\_id)가 20인 부서의 직원들 보다 급여(emp.salary)를 많이 받는 직원들의 정보를 sub query를 이용해 조회.

select \* from emp

where salary > all(select salary from emp where dept\_id = 20);

--max사용

select \* from emp

where salary > (select max(salary) from emp where dept\_id = 20);

-- TODO: 부서별 급여의 평균중 가장 적은 부서의 평균 급여보다 보다 많이 받는 사원들이 이름, 급여, 업무를 서브쿼리를 이용해 조회

select emp\_name, salary, job\_id from emp

where salary > any (select avg(salary) from emp group by dept\_id);

--min사용

select emp\_name, salary, job\_id from emp

where salary > (select min(avg(salary)) from emp group by dept\_id);

-- TODO: 업무 id(job\_id)가 'SA\_REP' 인 직원들중 가장 많은 급여를 받는 직원보다 많은 급여를 받는 직원들의 이름(emp\_name), 급여(salary), 업무(job\_id) 를 subquery를 이용해 조회.

select emp\_name, salary, job\_id from emp

where salary > all (select salary from emp where job\_id = 'SA\_REP');

--max사용

select emp\_name, salary, job\_id from emp

where salary > (select max(salary) from emp where job\_id = 'SA\_REP');

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

상관(연관) 쿼리

메인쿼리문의 조회값을 서브쿼리의 조건에서 사용하는 쿼리.

메인쿼리를 실행하고 그 결과를 바탕으로 서브쿼리의 조건절을 비교한다.

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 부서별(DEPT) 급여(emp.salary)를 가장 많이 받는 직원들의 id(emp.emp\_id), 이름(emp.emp\_name), 연봉(emp.salary), 소속부서ID(dept.dept\_id) 조회

select emp\_id, emp\_name, salary, dept\_id

from emp e

where salary = (select max(salary) from emp where dept\_id = e.dept\_id);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXISTS, NOT EXISTS 연산자 (상관(연관)쿼리와 같이 사용된다)

-- 서브쿼리의 결과를 만족하는 값이 존재하는지 여부를 확인하는 조건. 조건을 만족하는 행이 여러개라도 한행만 있으면 더이상 검색하지 않는다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 직원이 한명이상 있는 부서의 부서ID(dept.dept\_id)와 이름(dept.dept\_name), 위치(dept.loc)를 조회

select dept\_id, dept\_name, loc

from dept d

where exists(select emp\_id from emp where emp.dept\_id = d.dept\_id);

-- 직원이 한명도 없는 부서의 부서ID(dept.dept\_id)와 이름(dept.dept\_name), 위치(dept.loc)를 조회

select dept\_id, dept\_name, loc

from dept d

where not exists(select 1 from emp where dept\_id = d.dept\_id);--select의 값의 내용이 중요하지 않고 존재여부만 보면된다.

-- 부서(dept)에서 연봉(emp.salary)이 13000이상인 한명이라도 있는 부서의 부서ID(dept.dept\_id)와 이름(dept.dept\_name), 위치(dept.loc)를 조회

select dept\_id, dept\_name, loc

from dept d

where exists(select 1 from emp where d.dept\_id = dept\_id and salary>=13000);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

주문 관련 테이블들 이용.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

--TODO: 고객(customers) 중 주문(orders)을 한번 이상 한 고객들을 조회.

select \* from customers c

where exists(select order\_id from orders where cust\_id = c.cust\_id);

--TODO: 고객(customers) 중 주문(orders)을 한번도 하지 않은 고객들을 조회.

select \* from customers c

where not exists(select order\_id from orders where cust\_id = c.cust\_id);

--TODO: 제품(products) 중 한번이상 주문된 제품 정보 조회

select \* from products p

where exists(select ordere\_item\_id from order\_items oi where oi.product\_id=p.product\_id);

--TODO: 제품(products)중 주문이 한번도 안된 제품 정보 조회

select \* from products p

where not exists(select ordere\_item\_id from order\_items oi where oi.product\_id=p.product\_id);

/\*

Inline-view: subquery를 from 절에서 사용

- subquery의 조회결과를 테이블로 해서 main 쿼리가 실행된다

- inline-view의 서브쿼리에서 컬럼 별칭(alias)를 지정하면 main쿼리에서는 별칭을 사용해야 한다.

\*/

select emp\_id, emp\_name, job\_id, dept\_id from emp where dept\_id = 60;

select e.e\_id, e.e\_name, d.dept\_name --e.emp\_id, e.emp\_name의 별칭 사용(inlineview에서 별칭을 준 결과 테이블을 참조하므로)

from (select emp\_id e\_id, emp\_name e\_name, job\_id, dept\_id from emp where dept\_id = 60) e, dept d

where e.dept\_id = d.dept\_id;

/\*inline-view를 이용한 n-top쿼리(조회)

- 상위 5개, 하위 3개를 조회

- ROWNUM 컬럼: 조회결과 행번호를 반환

\*/

select rownum, emp\_id, emp\_name

from emp

where rownum<=5;

select rownum, emp\_id, emp\_name, salary

from emp

--where rownum<=5;

order by salary desc; --정렬이 제일 마지막에 일어나기 때문에 rownum이 뒤죽박죽이 된다.

--해결: inline view를 이용하여 미리 정렬된 테이블을 사용한다.

select rownum, emp\_id, emp\_name, salary

from (select \* from emp order by salary desc)

where rownum<=5;

--salary 높은 순서 5 - 10인 직원정보 조회

select rank 순위, emp\_id, emp\_name, salary --4: 결과로 보여줄 칼럼

from (select rownum rank, emp\_id, emp\_name, salary --2: rank라는 이름으로 rownum을 붙인 테이블

from (select \* from emp order by salary desc)) --1: salary기준으로 정렬

where rank between 5 and 10; --3: rank에 조건을 줌

--실패: 이유: from절에서 정렬을 하면 where절에서 하나씩 행을 불러와 rownum을 1부터 붙이고 조건을 확인하는데

--5보다 크지 않으니 pass 다음 행도 불러와서 rownum을 또 1로 붙이게 됨 pass 따라서 결과가 아예 나오지 않게된다.

select rownum, emp\_id, emp\_name, salary

from (select \* from emp order by salary desc)

where rownum between 5 and 10;

select rownum, emp\_id, emp\_name, salary

from (select \* from emp order by salary desc)

where rownum between 5 and 10;

select rank 순위, emp\_id, emp\_name, salary, e.dept\_id, d.dept\_name

from (select rownum rank, emp\_id, emp\_name, salary, dept\_id

from (select \* from emp order by salary desc)) e, dept d

where e.dept\_id = d.dept\_id(+) and rank between 5 and 10; --조인연산, dept와 조인을 하면서 순서가 바뀐다.=>정렬전에 처리

select rank 순위, emp\_id, emp\_name, salary, dept\_id, dept\_name

from ( select rownum rank, emp\_id, emp\_name, salary, dept\_id, dept\_name

from ( select e.emp\_id, e.emp\_name, e.salary, d.dept\_id, d.dept\_name

from emp e left join dept d on e.dept\_id = d.dept\_id

order by e. salary desc))

where rank between 5 and 10;

-- 입사일이 빠른 직원 10명

select rownum, emp\_id, emp\_name, hire\_date

from (select \* from emp order by hire\_date)

where rownum <= 10;

-- 입사일이 늦은 직원 10명

select rownum, emp\_id, emp\_name, hire\_date

from (select \* from emp order by hire\_date desc)

where rownum <= 10;