Day(BE)119_서브쿼리2, JOIN, 함수2

Date	@2022/12/27
작성일시	@December 27, 2022 9:09 AM
강의 번호	US101
Skilll	Data_SQL
Instructor	조성희
Sum up	☑
Review	☐

GitHub - Kray273/SQLLab

Day23 https://www.notion.so/Day23-2dac3079bfe044e6ac6a0e905d76b237 Day24 https://www.notion.so/Day24-690b518731314b0ba14207dfd37c4a58 Day25 ...

https://github.com/Kray273/SQLLab

Kray273/**SQLLab**

1 Contributor 0 Issues 0 Stars 0 Forks

▼ Review

다른 SQL내부에 포함된 select문 - 서브쿼리

```
> CREATE TABLE emp_copy (select * from employees where emp_id >= 200);
> insert into emp_copy select * from employees where emp_id >= 200;

> select (select ...)
  from (select ...)
 where (select ...)
> update / delete
```

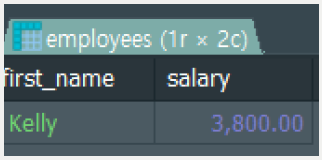
서브쿼리

▼ 단일행과 다중행 서브쿼리

리턴 되는 값의 수에 따라 구변된다.

단일행 SUBQUERY	= , > , <, <=, >=
다중행 SUBQUERY	IN(,,,,,) =ANY() > ANY() > ALL()
다중 열 다중행 subquery	where (a,b,c) in (select x, y, z .,..)

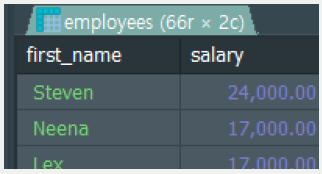
```
-- kelly 급여가 같은 사원의 이름, 급여 조회
  SELECT first_name, salary FROM employees
  WHERE salary > (SELECT salary FROM employees WHERE first_name ='kelly');
```



first_name	salary
Kelly	3,800.00

단일행 서브쿼리

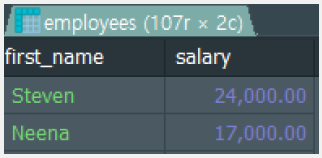
```
-- kelly 급여가 큰 사원의 이름, 급여 조회 _
SELECT first_name, salary FROM employees
WHERE salary > (SELECT salary FROM employees WHERE first_name ='kelly');
```



first_name	salary
Steven	24,000.00
Neena	17,000.00
Lex	17,000.00

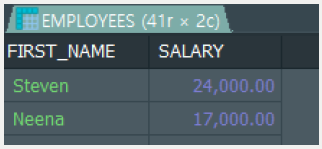
다중행 서브쿼리

```
-- 모든 william 보다 급여를 더 많이 받는 사원의 이름 급여조회,
SELECT first_name, salary FROM employees
WHERE salary > ALL (SELECT salary FROM employees WHERE first_name ='Welliam');
```



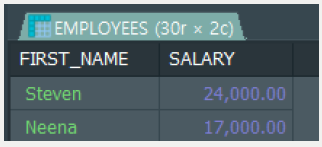
first_name	salary
Steven	24,000.00
Neena	17,000.00

```
-- 모든 william 보다 급여를 더 많이 받는 사원의 이름 급여조회,
SELECT FIRST_NAME, SALARY FROM EMPLOYEES
WHERE SALARY > ANY(SELECT SALARY FROM EMPLOYEES WHERE FIRST_NAME ='WILLIAM');
```



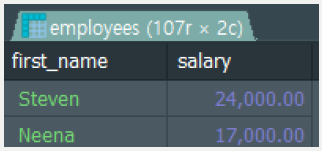
FIRST_NAME	SALARY
Steven	24,000.00
Neena	17,000.00

```
SELECT FIRST_NAME, SALARY FROM EMPLOYEES
WHERE SALARY > ALL(SELECT SALARY FROM EMPLOYEES WHERE FIRST_NAME ='WILLIAM');
```



FIRST_NAME	SALARY
Steven	24,000.00
Neena	17,000.00

```
-- 의미 없다...
SELECT first_name, salary FROM employees
WHERE salary = ALL (SELECT salary FROM employees WHERE first_name ='Welliam');

-- 'ANY' ==>
WHERE salary IN (7400, 8300);
WHERE salary = ANY (7400, 8300);
-- WHERE salary = ALL (7400, 8300);
```



first_name	salary
Steven	24,000.00
Neena	17,000.00

```
-- 변수 사용
SET @NAME='PETER';
SELECT FIRST_NAME, SALARY FROM EMPLOYEES
WHERE SALARY > ANY( SELECT SALARY FROM EMPLOYEES WHERE FIRST_NAME = @NAME);
```

EMPLOYEES (96r × 2c)

FIRST_NAME	SALARY
Steven	24,000.00
Neena	17,000.00
Lex	17,000.00

다중열 서브쿼리

```
-- 부서의 최대급여를 받는 사원의 이름 급여 조회 _ 다중열 서브쿼리
SELECT first_name, salary, department_id FROM employees
WHERE (department_id, salary)
  IN(SELECT department_id, MAX(salary) FROM employees GROUP BY department_id);
```

employees (11r × 3c)

first_name	salary	department_id
Steven	24,000.00	90
Alexander	9,000.00	60

```
-- 1. 런던 도시코드 조회
SELECT * FROM locations WHERE city = 'london'; -- 24000
-- 2. 런던 도시코드 같은 도시코드 부서코드 조회
SELECT * FROM departments; -- 데이터 확인
SELECT department_id FROM departments WHERE location_id = 2400; -- 40

-- 3.  해당부서의 사원 조회
SELECT first_name, department_id FROM employees
WHERE department_id = 40;

-- 중첩 서브쿼리
SELECT first_name, department_id FROM employees
WHERE department_id
  = (SELECT department_id FROM departments WHERE location_id
  = (SELECT location_id FROM locations WHERE city = 'london') );
```

employees (1r × 2c)

first_name	department_id
Susan	40

```
-- 부서의 평균급여보가 더 많이 받는 사원의 이름 급여 조회 _ 연관 서브쿼리
SELECT first_name,  department_id, salary,
  (SELECT AVG(salary) FROM employees WHERE e.department_id = department_id) AS '평균급
FROM employees e
WHERE salary > ANY(SELECT AVG(salary) FROM employees WHERE e.department_id = department_id);
```

employees (38r × 4c)

first_name	department_id	salary	평균급여
Steven	90	24,000.00	19,333.333333
Alexander	60	9,000.00	5,760.000000
Bruce	60	6,000.00	5,760.000000

```
-- inline view  : from절에 들어가는 서브쿼리
/*  SELECT *
  FROM (SELECT 결과 1개 이상의 가상 테이블 - inline VIEW)
  WHERE ...; */

-- 10000이상 급여 평균
SELECT AVG(salary)
FROM employees
WHERE salary >= 10000;

SELECT SAL_TBL.AVG_SAL -- from 절에서 조회한 내용만 출려 가능
FROM (SELECT AVG(salary) AVG_SAL FROM employees WHERE salary >= 10000) SAL_TBL;
```

결과 #1 (1r × 1c)

AVG_SAL
12,632.421053

```
-- 급여 수준에 따른 직급 조회
-- employees 테이블 급여칼럼 없다.
-- 직급을 생성한다. 직급은 급여가 20000이상 임원, 15000 이상 부장, 10000이상 5000이상 대리 이하 사원
-- 급여 salaty + salary * commission_pct
SELECT MAX(salary), MIN(salary) FROM employees;

SELECT first_name,
Case -- case when 조건문
-- null 값 칼럼 연산식 결과도 null => ifnull은 null인 경우 대체값.
When salary + salary * IfNull(commission_pct,0.1) >= 20000 Then '임원'
```

```
When salary + salary * IfNull(commission_pct,0.1) >= 15000 Then '부장'
When salary + salary * IfNull(commission_pct,0.1) >= 10000 Then '과장'
When salary + salary * IfNull(commission_pct,0.1) >= 5000 Then '대리'
ELSE '사원'
END 직급
FROM employees;

-- 위와 동일 결과
SELECT first_name,INSISAL,
Case When INSISAL >= 20000 Then '임원'
  When INSISAL >= 15000 Then '부장'
  When INSISAL >= 10000 Then '과장'
  When INSISAL >= 5000 Then '대리'
ELSE '사원'
END 직급
FROM (SELECT first_name, salary + salary * IfNull(commission_pct,0.1) AS INSISAL FROM employees) INSITBL;
```

INSITBL (107r × 2c)

first_name	직급
Steven	임원
Neena	부장
Lex	부장
Alexander	대리
Bruce	대리
David	대리

INSITBL (107r × 3c)

first_name	INSISAL	직급
Steven	26,400.0000	임원
Neena	18,700.0000	부장
Lex	18,700.0000	부장
Alexander	9,900.0000	대리
Bruce	6,600.0000	대리
David	5,280.0000	대리

```
-- update
-- emp_copy
-- kelly와 같은 부서의 사원의 부서를 100으로 이동
SELECT first_name, department_id FROM employees WHERE department_id = 100; -- 6명
SELECT first_name, department_id FROM employees WHERE department_id = 50; -- 45명

UPDATE emp_copy
SET department_id = 100
WHERE department_id = (SELECT department_id FROM emp_copy WHERE first_name = 'kelly');

SELECT first_name, department_id FROM emp_copy WHERE department_id = 100; -- 51명 추가됨
```

emp_copy (51r × 2c)

first_name	department_id
Nancy	100
Daniel	100
John	100

JOIN

▼ 집합연산자

2개의 물리적인 테이블을 합쳐서 조회

```
-- 조합 테이블 컬럼 갯수 타입 순서 일치
-- union / union all / intersect / minus(마리아db-except)
```

```
-- 50번 부서의 모든 부서원 복사 emp_dept_50 테이블 생성
CREATE TABLE dep_dept_50
(SELECT * FROM employees WHERE department_id = 50);
SELECT * from dep_dept_50; -- 45명

-- manager 계열 직종 사원들의 emp_job_man 테이블 생성
CREATE TABLE emp_job_man
(SELECT * FROM employees WHERE job_id LIKE '%man%');
SELECT * from emp_job_man; -- 12명
```

```
-- 재난 지원금을 지급하려고 한다.
-- 대상은 50번 부서원이거나 manager 직종으로 한정하여 조회해 본다.
SELECT employee_id, first_name, department_id, job_id
FROM emp_dept_50
UNION -- 합집합 중복 제거
SELECT employee_id, first_name, department_id, job_id
FROM emp_job_man
ORDER BY 1;

-- UNION은 중복이 제거되어 출력
```

emp_dept_50 (52r × 4c)

employee_id	first_name	department_id	job_id
114	Den	30	PU_MAN
120	Matthew	50	ST_MAN
121	Adam	50	ST_MAN

```
-- 중복 수령 가능
SELECT employee_id, first_name, department_id, job_id
FROM emp_dept_50
UNION ALL -- 합집합 중복 허용
SELECT employee_id, first_name, department_id, job_id
FROM emp_job_man
ORDER BY 1;

-- UNION ALL 중복을 포함하여 출력
```

emp_dept_50 (57r × 4c)

employee_id	first_name	department_id	job_id
114	Den	30	PU_MAN
120	Matthew	50	ST_MAN
120	Matthew	50	ST_MAN

```
-- 대상 50번 부서원이고 manager직종을 한정으로 지원(교집합)
SELECT employee_id, first_name, department_id, job_id
FROM emp_dept_50
INTERSECT -- 교집합
SELECT employee_id, first_name, department_id, job_id
FROM emp_job_man;
```

emp_dept_50 (5r × 4c)

employee_id	first_name	department_id	job_id
120	Matthew	50	ST_MAN
121	Adam	50	ST_MAN
122	Payam	50	ST_MAN
123	Shanta	50	ST_MAN
124	Kevin	50	ST_MAN

```
-- 대상 50번 부서원이고 manager직종을 제외하고 지원
SELECT employee_id, first_name, department_id, job_id
FROM emp_dept_50
EXCEPT -- 차집합
SELECT employee_id, first_name, department_id, job_id
  FROM emp_job_man;
```

emp_dept_50 (40r × 4c)

employee_id	first_name	department_id	job_id
125	Julia	50	ST_CLERK
126	Irene	50	ST_CLERK
127	James	50	ST_CLERK

```
-- 위치가 다르면 다른 조건이 출력된다.
SELECT employee_id, first_name, department_id, job_id
FROM emp_job_man
EXCEPT
SELECT employee_id, first_name, department_id, job_id
FROM emp_dept_50;
```

emp_job_man (7r × 4c)

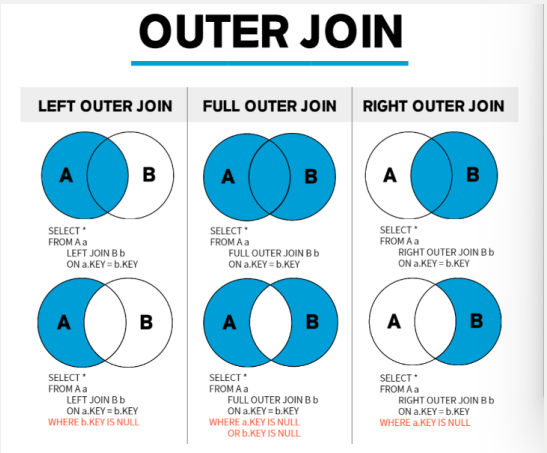
employee_id	first_name	department_id	job_id
114	Den	30	PU_MAN
145	John	80	SA_MAN
146	Karen	80	SA_MAN

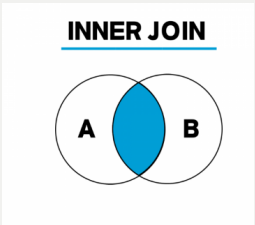
▼ JOIN

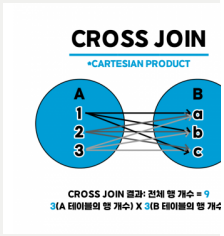
2개의 테이블 칼럼 갯수 합병

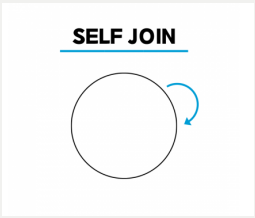
집합 연산자 - 2개 테이블의 행 갯수 합병-행갯수 변화
join - 2개 테이블 컬럼 갯수 합병 - 1개 레코드 열갯수 변화

```
-- db 표준 문법 join (ansi join)
SELECT a,b
FROM ansi JOIN bnsi ON a = b;

-- db 종속
SELECT a,b
FROM ansi, bnsi WHERE a = b;
```










```
/*
select
from a join b
  on a.컬럼 = b.컬럼
*/

-- 사원명 부서명 조회 , 2개의 테이블에는 동일한 값을 가진 컬럼 필요
SELECT first_name, department_id FROM employees;

SELECT department_name, department_id FROM departments;

SELECT first_name, department_name
FROM employees  INNER JOIN departments
  ON employees.department_id = departments.department_id;

-- 동일 결과
SELECT first_name, department_name
FROM employees  CROSS JOIN departments
  ON employees.department_id = departments.department_id;

-- 동일 결과
SELECT first_name, department_name
FROM employees JOIN departments
  ON employees.department_id = departments.department_id;
```

결과 #1 (106r × 2c)

first_name	department_name
Steven	Executive
Neena	Executive
Lex	Executive

```
SELECT first_name, department_name
FROM employees CROSS JOIN departments;
-- 조인의 조건절인 on이 없을 시 옆과 같이 모든 칼럼을 중복되게 합치는 결과를 보여준다.
-- 이런 cross join은 사용할 일이 거의 없다.
```

결과 #1 (2,889r × 2c)

first_name	department_name
Steven	Administration
Steven	Marketing
Steven	Purchasing

```sql
-- error 중복 조건인 department_id를 찾을 수 없어서.
SELECT first_name, department_id, department_name
FROM employees  CROSS JOIN departments
  ON employees.department_id = departments.department_id;

-- 다음과 같이 변경
SELECT employees.first_name, employees.department_id, departments.department_nam
FROM employees  CROSS JOIN departments
  ON employees.department_id = departments.department_id;

-- 동일 결과
SELECT first_name, employees.department_id, department_name
FROM employees  CROSS JOIN departments
  ON employees.department_id = departments.department_id;

-- 동일 결과
SELECT first_name, e.department_id, d.department_id, department_name
FROM employees e CROSS JOIN departments d
  ON e.department_id = d.department_id;
```

결과 #1 (106r × 3c)

first_name	department_id	department_name
Steven	90	Executive
Neena	90	Executive

결과 #1 (106r × 4c)

first_name	department_id	department_id	department_na
Steven	90	90	Executive
Neena	90	90	Executive
Lex	90	90	Executive

```sql
-- job 테이블 JOB_ID직종코드 -> IT_PROG JOB_TITLE직종이름
-- employess job_id references job(job_id)
-- 사원이름 , 직종 이름 , 급여 조회
SELECT  first_name, job_title, salary
FROM employees e JOIN jobs j
  ON e.job_id = j.job_id
ORDER BY salary DESC;
```

결과 #1 (107r × 3c)

first_name	job_title	salary
Steven	President	24,00
Neena	Administration Vice President	17,00

```sql
-- 사원이름, 직종이름, 부서이름 조회
SELECT  first_name, job_title, department_name
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
JOIN departments d ON e.department_id = d.department_id;
```

결과 #1 (106r × 3c)

first_name	job_title	department_name
Steven	President	Executive
Neena	Administration Vice President	Executive
Lex	Administration Vice President	Executive

```sql
-- 사원이름, 직종이름, 부서이름 조회
-- 단, 급여가 10000이상
SELECT  first_name, job_title, department_name, salary
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
JOIN departments d ON e.department_id = d.department_id
WHERE e.salary >= 10000
ORDER BY salary ASC;
```

결과 #1 (19r × 4c)

first_name	job_title	department_name	salary
Peter	Sales Representative	Sales	10,0
Hermann	Public Relations Representative	Public Relations	10,0
Janette	Sales Representative	Sales	10,0
Harrison	Sales Representative	Sales	10,0
Clara	Sales Representative	Sales	10,5
Eleni	Sales Manager	Sales	10,5

```sql
-- seattle 도시 근무 사원의 사원명, 부서명, 도시명 조회
SELECT e.first_name, d.department_name, city
FROM employees e
  JOIN departments d ON e.department_id = d.department_id
  JOIN locations l ON l.location_id = d.location_id
WHERE l.city LIKE 'seattle';
```

결과 #1 (18r × 3c)

first_name	department_name	city
Steven	Executive	Seattle
Neena	Executive	Seattle
Lex	Executive	Seattle
Nancy	Finance	Seattle
Daniel	Finance	Seattle

```sql
-- Inner Join : 양쪽 테이블 조건 만족하는 데이터들만 조인해서 불러온다.
SELECT e.first_name, d.department_name
FROM employees e
  INNER JOIN departments d ON e.department_id = d.department_id;
```

결과 #1 (106r × 2c)

first_name	department_name
Steven	Executive
Neena	Executive
Lex	Executive

```
-- LEFT OUTER Join : 왼쪽 테이블 조건 해당하는  데이터들만 조인해서 불러온다.
SELECT e.first_name, d.department_name
FROM employees e
  LEFT OUTER JOIN departments d ON e.department_id = d.department_id;
-- 부서가 배정되지 않은 사원도 포함됨.
```

결과 #1 (107r × 2c)

first_name	department_name
Steven	Executive
Neena	Executive
Lex	Executive

```
-- 부서명 사원명 조회 단, 1명의 사원도 소속되지 않은 부서도 포함 조인
SELECT e.first_name, d.department_name
FROM employees e
  RIGHT OUTER JOIN departments d ON e.department_id = d.department_id
ORDER BY first_name;
```

결과 #1 (122r × 2c)

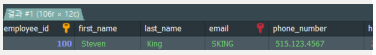
first_name	department_name
(NULL)	Shareholder Services
(NULL)	Operations
(NULL)	Retail Sales
(NULL)	Control And Credit

```
-- MariaDB에는 Full Outer Join이 없다. 하지만 union을 사용하면 동일한 기능을 구현 가능하다.
SELECT e.first_name, d.department_name
FROM employees e
  RIGHT OUTER JOIN departments d ON e.department_id = d.department_id
UNION
SELECT e.first_name, d.department_name
FROM employees e
  LEFT OUTER JOIN departments d ON e.department_id = d.department_id;
```

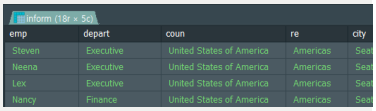
employees (117r × 2c)

first_name	department_name
Steven	Executive
Neena	Executive
Lex	Executive
Alexander	IT

```
SELECT e.* , d.department_id
FROM employees e JOIN departments d ON e.department_id = d.department_id;
```



```
-- seattle에 사는 사람의 정보 추출
SELECT inform.emp,  depart, coun, re, city
FROM (SELECT first_name emp, department_name depart,
      country_name coun, region_name re, city
FROM employees e
  JOIN departments d ON e.department_id = d.department_id
  JOIN locations l ON l.location_id = d.location_id
  JOIN countries c ON c.country_id = l.country_id
  JOIN regions r ON r.region_id = c.region_id
  WHERE city = 'seattle') inform;
```

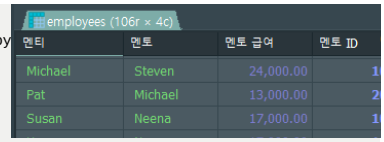


```
-- 자체 조인(self join) : 조인 대상 테이블이 자신 테이블 조회
--  내 상사의 이름, 급여 조회
SELECT manager_id FROM employees WHERE employee_id = 200; -- 101
SELECT first_name, salary, employee_id FROM employees WHERE employee_id = 101;

-- 동일 결과 _ 서브쿼리문
SELECT first_name, salary, employee_id FROM employees
WHERE employee_id = (SELECT manager_id FROM employees WHERE employee_id = 200);

-- 동일 결과 _ 서브쿼리문
SELECT first_name, salary, employee_id FROM employees
WHERE employee_id = (SELECT manager_id FROM employees WHERE employee_id = 200);

-- 동일 결과 _ 셀프조인
SELECT you.first_name, you.salary, you.employee_id
FROM employees me
  JOIN employees you ON me.manager_id = you.employee_id
WHERE me.employee_id = 200;
```

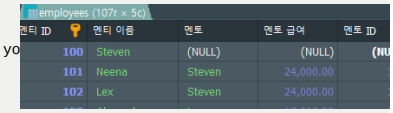
employees (1r × 3c)

first_name	salary	employee_id
Neena	17,000.00	1

```
SELECT me.first_name '멘티', you.first_name'멘토', you.salary'멘토 급여', you.employ
FROM employees me
  JOIN employees you ON me.manager_id = you.employee_id;
```




```
-- 상사가 없는 사원 포함
SELECT me.employee_id '멘티 ID', me.first_name '멘티 이름', you.first_name'멘토', yo
FROM employees me
  LEFT OUTER JOIN employees you ON me.manager_id = you.employee_id;
```



▼ 함수

💡 특정조건을 키워드를 통해 실행

- 집계 함수 : count max min sum abg ⇒ 여러개 레코드를 모아서 1개의 결과 리턴
- 조건 함수
  - CASE

```
case
when 조건식 then true일때의 값1
when 조건식 then true일때의 값1
....
else 위조건식 모두 불일치하는 경우 결과값
end ALIAS명;
```

  - - update
    -- emp_copy
    -- kelly와 같은 부서의 사원의 부서를 100으로 이동
    SELECT first_name, department_id FROM employees WHERE department_id = 100; -- 6명
    SELECT first_name, department_id FROM employees WHERE department_id = 50; -- 45명

UPDATE emp_copy
SET department_id = 100
WHERE department_id = (SELECT department_id FROM emp_copy WHERE first_name = 'kelly');

SELECT first_name, department_id FROM emp_copy WHERE department_id = 100; -- 51명 추가됨

  - IFNULL

```
ifnull(칼럼명,null시 대체값)
```

  - NULLIF : 값이 일치할 시 null을 리턴, 불일치시 1

```
nullif(값1,값2)
```

▼ join_subquery실습과제

join_subquery실습과제

```
1. 80번부서의 평균급여보다 많은 급여를 받는 직원의 이름, 부서id, 급여를 조회하시오.


2. 'South San Francisco'에 근무하는 직원의 최소급여보다 급여를 많이 받으면서 50 번부서의 평균급여보다 많은 급여를 받는 직원의 이름, 급여, 부서명,
부서id를 조회하시오.

3-1.각 직급별(job_title) 인원수를 조회하되 사용되지 않은 직급이 있다면 해당 직급도 출력결과에 포함시키시오.

3-2. 직급별 인원수가 10명 이상인 직급만 출력결과에 포함시키시오.


4. 각 부서별 최대급여를 받는 직원의 이름, 부서명, 급여를 조회하시오.


5. 직원의 이름, 부서id, 급여를 조회하시오. 그리고 직원이 속한 해당 부서의
최소급여를 마지막에 포함시켜 출력 하시오.


6. 월별 입사자 수를 조회하되, 입사자 수가 10명 이상인 월만 출력하시오.


7. 자신의 관리자(상사)보다 많은 급여를 받는 직원의 이름과 급여를 조회하시오.


8. 'Southlake'에서 근무하는 직원의 이름, 급여, 직책(job_title)을 조회하시오.


9. 국가별 근무 인원수를 조회하시오. 단, 인원수가 3명 이상인 국가정보만 출력되어야함.


10. 직원의 폰번호, 이메일과 상사의 폰번호, 이메일을 조회하시오. 단, 상사가 없는 직원은 '<관리자 없음>'이 출력되도록 해야 한다.


11. 각 부서 이름별로 최대급여와 최소급여를 조회하시오. 단, 최대/최소급여가 동일한 부서는 출력결과에서 제외시킨다.

12. 부서별, 직급별, 평균급여를 조회하시오.
   단, 평균급여가 50번부서의 평균보다 많은 부서만 출력되어야 합니다.
```

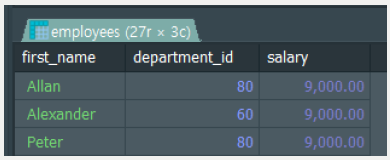
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/feaec0d6-d03d-4d4f-961d-7a5a61c3f526/join_subquery%EC%8B%A4%EC%8A%B5%EA%B3%BC%EC%A0%9C.txt

▼ Sol

join_subquery 해결

1. 80번부서의 평균급여보다 많은 급여를 받는 직원의 이름, 부서id, 급여를 조회하시오.

```
SELECT e.first_name, e.department_id, e.salary
FROM employees e
  JOIN departments d ON e.department_id = d.department_id
WHERE e.salary > (
  SELECT AVG(salary) FROM employees WHERE department_id = 80) -- 8955
ORDER BY e.salary;

SELECT FIRST_NAME, DEPARTMENT_ID, salary,(SELECT AVG(salary) FROM employees WHERE department_id=80)
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees WHERE department_id=80) ;
```



employees (27r × 3c)

first_name	department_id	salary
Allan	80	9,000.00
Alexander	60	9,000.00
Peter	80	9,000.00

2. 'South San Francisco'에 근무하는 직원의 최소급여보다 급여를 많이 받으면서
50 번부서의 평균급여보다 많은 급여를 받는 직원의 이름, 급여, 부서명, 부서id를 조회하시오.

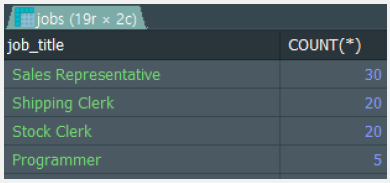
```
SELECT first_name, salary, d.department_name, e.department_id
FROM employees e
  JOIN departments d ON e.department_id = d.department_id
WHERE salary  >(
  SELECT MIN(salary)
    FROM employees e
      inner join departments d ON e.department_id = d.department_id
      INNER JOIN locations l ON d.location_id = l.location_id WHERE city= 'South San Francisco')
AND salary > (SELECT AVG(salary)FROM employees WHERE department_id=50)
ORDER BY salary;
```

결과 #1 (69r × 4c)

first_name	salary	department_name	department_id
Trenna	3,500.00	Shipping	50
Renske	3,600.00	Shipping	50
Jennifer	3,600.00	Shipping	50
Kelly	3,800.00	Shipping	50

3-1.각 직급별(job_title) 인원수를 조회하되 사용되지 않은 직급이 있다면 해당 직급도 출력결과에 포함시키시오.

```
SELECT j.job_title, COUNT(*)
FROM employees e
  LEFT OUTER JOIN jobs j ON e.job_id = j.job_id
GROUP BY j.job_title
ORDER BY COUNT(*) DESC;
```

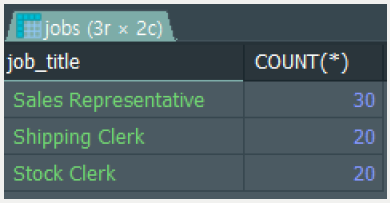


jobs (19r × 2c)

job_title	COUNT(*)
Sales Representative	30
Shipping Clerk	20
Stock Clerk	20
Programmer	5

3-2. 직급별 인원수가 10명 이상인 직급만 출력결과에 포함시키시오.

```
SELECT j.job_title, COUNT(*)
FROM employees e
  LEFT OUTER JOIN jobs j ON e.job_id = j.job_id
GROUP BY j.job_title
HAVING COUNT(*) >= 10;
```

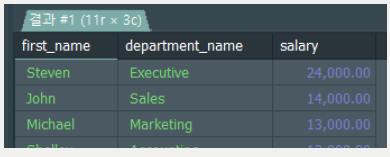


jobs (3r × 2c)

job_title	COUNT(*)
Sales Representative	30
Shipping Clerk	20
Stock Clerk	20

4. 각 부서별 최대급여를 받는 직원의 이름, 부서명, 급여를 조회하시오.

```
SELECT e.first_name, d.department_name, e.salary
FROM employees e
  left outer JOIN departments d ON e.department_id = d.department_id
WHERE (e.salary, d.department_id) = Any
  (SELECT MAX(salary), department_id FROM employees GROUP BY department_id)
ORDER BY e.salary DESC;
```



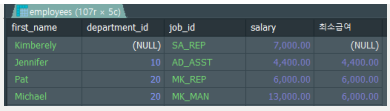
결과 #1 (11r × 3c)

first_name	department_name	salary
Steven	Executive	24,000.00
John	Sales	14,000.00
Michael	Marketing	13,000.00

5. 직원의 이름, 부서id, 급여를 조회하시오. 그리고 직원이 속한 해당 부서의 최소급여를 마지막에 포함시켜 출력 하시오.

```
SELECT e.first_name, e.department_id, e.job_id, e.salary,
```

```
  (SELECT MIN(salary) FROM employees WHERE e.department_id = department_id)
FROM employees e
ORDER BY department_id;
```



first_name	department_id	job_id	salary	최소급여
Kimberely	(NULL)	SA_REP	7,000.00	(NULL)
Jennifer	10	AD_ASST	4,400.00	4,400.00
Pat	20	MK_REP	6,000.00	6,000.00
Michael	20	MK_MAN	13,000.00	6,000.00

6. 월별 입사자 수를 조회하되, 입사자 수가 10명 이상인 월만 출력하시오.

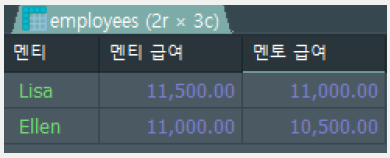
```
SELECT case
  when e.hire_date LIKE '_____01%' then '1월'
  when e.hire_date LIKE '_____02%' then '2월'
  when e.hire_date LIKE '_____03%' then '3월'
  when e.hire_date LIKE '_____04%' then '4월'
  when e.hire_date LIKE '_____05%' then '5월'
  when e.hire_date LIKE '_____06%' then '6월'
  when e.hire_date LIKE '_____07%' then '7월'
  when e.hire_date LIKE '_____08%' then '8월'
  when e.hire_date LIKE '_____09%' then '9월'
  when e.hire_date LIKE '_____10%' then '10월'
  when e.hire_date LIKE '_____11%' then '11월'
  when e.hire_date LIKE '_____12%' then '12월'
  ELSE '기타'
  END months, COUNT(*)
FROM employees e
GROUP BY months
HAVING COUNT(*) >= 10;
```

months	COUNT(*)
1월	14
2월	13
3월	17
6월	11

```
-- 동일결과
SELECT Month(e.hire_date) '월', COUNT(*)
FROM employees e
GROUP BY Month(e.hire_date)
HAVING COUNT(*) >= 10;
```

7. 자신의 관리자(상사)보다 많은 급여를 받는 직원의 이름과 급여를 조회하시오.

```
SELECT me.first_name '멘티', me.salary'멘티 급여', you.salary'멘토 급여'
FROM employees me
  JOIN employees you ON me.manager_id = you.employee_id
WHERE me.salary > you.salary;
```



멘티	멘티 급여	멘토 급여
Lisa	11,500.00	11,000.00
Ellen	11,000.00	10,500.00

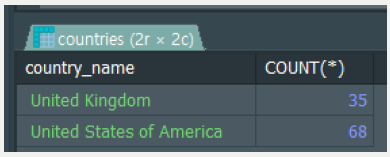
8. 'Southlake'에서 근무하는 직원의 이름, 급여, 직책(job_title)을 조회하시오.

```
SELECT e.first_name, e.salary, j.job_title, l.city
FROM employees e
  JOIN jobs j ON e.job_id = j.job_id
  JOIN departments d ON e.department_id = d.department_id
  JOIN locations l ON d.location_id = l.location_id
WHERE l.city = 'Southlake'
ORDER BY e.salary;
```

first_name	salary	job_title	city
Diana	4,200.00	Programmer	Southlake
David	4,800.00	Programmer	Southlake
Valli	4,800.00	Programmer	Southlake

9. 국가별 근무 인원수를 조회하시오. 단, 인원수가 3명 이상인 국가정보만 출력되어야함.

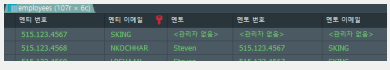
```
SELECT c.country_name, COUNT(*)
FROM employees e
  JOIN departments d ON e.department_id = d.department_id
  JOIN locations l ON d.location_id = l.location_id
  JOIN countries c ON c.country_id = l.country_id
GROUP BY c.country_name
HAVING COUNT(*) >= 3;
```



country_name	COUNT(*)
United Kingdom	35
United States of America	68

10. 직원의 폰번호, 이메일과 상사의 폰번호, 이메일을 조회하시오. 단, 상사가 없는 직원은 '<관리자 없음>'이 출력되도록 해야 한다.

```
SELECT me.first_name '멘티', me.phone_number '멘티 번호', me.email '멘티 이메일',
      IfNull(you.first_name ,'<관리자 없음>') '멘토' ,
      IfNull(you.phone_number ,'<관리자 없음>') '멘토 번호',
      IfNull(you.email ,'<관리자 없음>') '멘토 이메일'
```



```
FROM employees me
  LEFT OUTER JOIN employees you ON me.manager_id = you.employee_id;
```

11. 각 부서 이름별로 최대급여와 최소급여를 조회하시오. 단, 최대/최소급여가 동일한 부서는 출력결과에서 제외시킨다.

```
SELECT d.department_name, MAX(e.salary) AS max_sal, MIN(e.salary) AS min_sal
FROM employees e
 JOIN departments d USING (department_id)
group BY d.department_name
HAVING MAX(e.salary) != MIN(e.salary);
```

departments (8r × 3c)

department_name	max_sal	min_sal
Accounting	12,008.00	8,300.00
Executive	24,000.00	17,000.00
Finance	12,008.00	6,900.00
IT	9,000.00	4,200.00

12. 부서별, 직급별, 평균급여를 조회하시오. 단, 평균급여가 50번부서의 평균보다 많은 부서만 출력되어야 합니다.

```
SELECT d.department_name, j.job_title, AVG(e.salary)
FROM employees e
 LEFT OUTER JOIN departments d USING (department_id)
 LEFT OUTER JOIN jobs j USING (job_id)
group BY d.department_id, j.job_id
HAVING AVG(e.salary) > (SELECT AVG(salary) FROM employees WHERE department_id = 50);
```

결과 #1 (17r × 3c)

department_name	job_title	AVG(e.salary)
(NULL)	Sales Representative	7,000.000000
Administration	Administration Assistant	4,400.000000
Marketing	Marketing Manager	13,000.000000
Marketing	Marketing Representative	6,000.000000