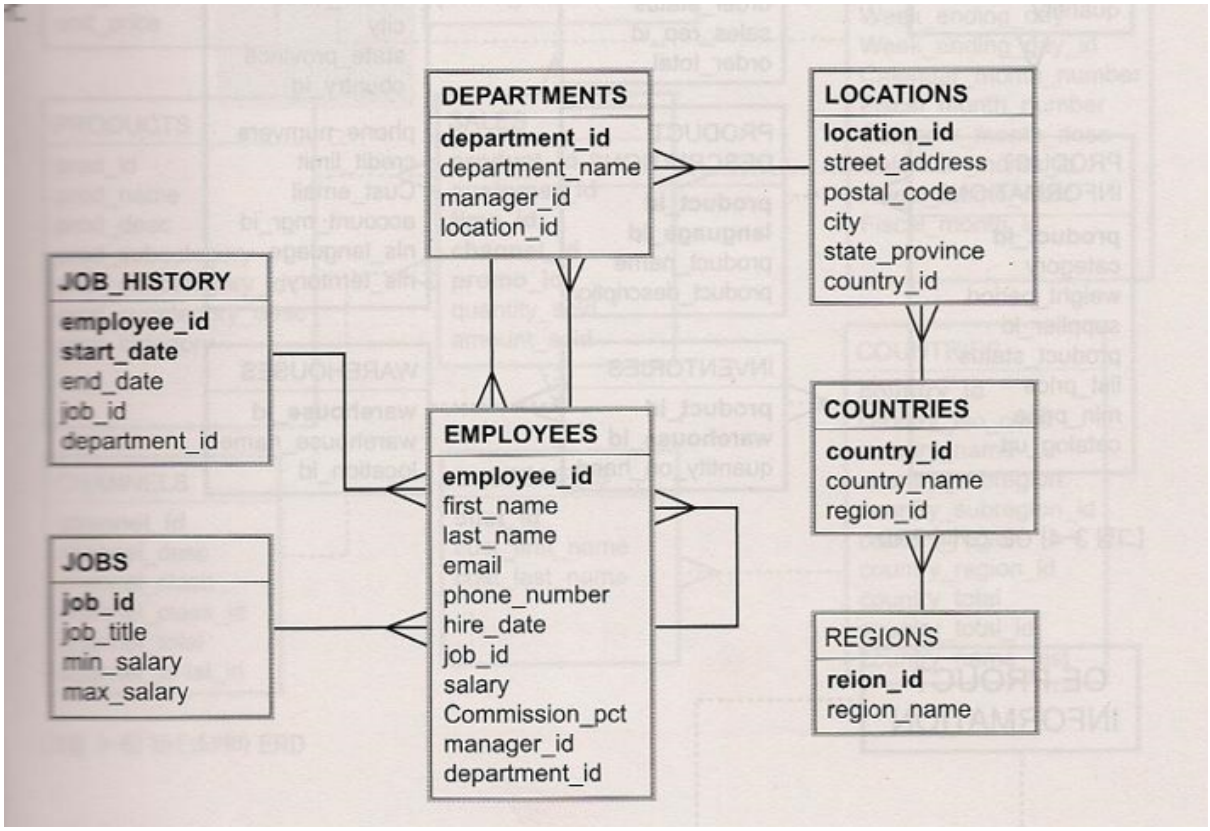
조인 및 서브쿼리 연습

조인

- 2개 이상의 테이블에서 데이터 검색하기 위해 사용

예시

- ‘King’의 부서 이름을 출력하라




```
SELECT employee_id, department_id
    FROM employees
    WHERE last_name = 'King';

SELECT department_id, department_name
    FROM departments IN(80, 90);
```

각 테이블 상세구조

emp(사원)		dept(부서)		salgrade(급여등급)	
컬럼명	*내용*	*컬럼명*	*내용*	*컬럼명*	*내용*
eno	사번	dno	부서번호	grade	급여등급
ename	이름	dname	부서명	losal	하한값
job	업무	loc	지역	hosal	상한값
mgr	관리자 사번	director	부서장사번		
hdate	입사일				
sal	급여(월)				
comm	보너스				
dno	부서번호				

student(학생)		professor(교수)		course(과목)	
컬럼명	*내용*	*컬럼명*	*내용*	*컬럼명*	*내용*
sno	학번	pno	교수번호	cno	과목번호
sname	이름	pname	이름	cname	과목명
sex	성별	section	소속학과	st_num	학점수
syear	학년	orders	직위	pno	교수번호
major	학과	hiredate	부임일		
avr	평점				

score(기말고사 점수)		scgrade(점수등급)	
컬럼명	*내용*	*컬럼명*	*내용*
sno	학번	grade	등급
cno	과목번호	hiscore	상한값
result	점수	loscore	하한값

조인 이용 방법

1. 내가 원하는 데이터는 무엇인가? (컬럼 목록)
2. 원하는 데이터가 어느 테이블에 있는가?
3. 여러 테이블에 있다면 각 테이블의 공통 컬럼을 찾는다

```
-- 1. 컬럼 목록 나열
-- 어느 테이블의 컬럼인지 앞에 테이블.으로 명시해 줘야 한다
SELECT e.employee_id, e.department_id, d.department_name
-- 2. 원하는 데이터가 어느 테이블에 있는가?
```

```
       FROM employees e, departments d
-- 3. 여러 테이블에 있다면 각 테이블의 공통 컬럼
       WHERE e.department_id = d.department_id
       AND last_name = 'King';
```

실습 1 [kosa 01]

- '송강' 교수가 강의하는 과목을 검색하라
  - 교수번호(pno), 교수이름(pname), 과목명(cname)

```
-- 1. 컬럼 목록 나열
SELECT p.pno, p.pname, c.cname
-- 2. 원하는 데이터가 어느 테이블?
    FROM professor p, course c
-- 3. 여러 테이블에 있다면 각 테이블의 공통 컬럼 추출
    WHERE p.pno = c.pno
    AND pname = '송강';
```

- 학점이 2학점인 과목과 이를 강의하는 교수를 검색해라

```
SELECT c.st_num, c.cname, p.pno, p.pname
    FROM professor p, course c
    WHERE c.st_num = 2 AND p.pno = c.pno;
```

- 화학과 1학년 학생의 기말고사 성적을 검색해라

```
SELECT s.sno, s.sname, s.major, ss.result
    FROM student s, score ss
    WHERE s.sno = ss.sno AND major = '화학' AND syear = 1 ;
```

- 화학과 1학년 학생이 수강하는 과목을 검색해라 (3개 테이블 조인)

```
SELECT s.sno, s.sname, s.syear, c.cname
    FROM student s, score ss, course c
    WHERE major = '화학' AND SYEAR = 1 AND s.sno = ss.sno AND ss.cno = c.cno;
```

ANSI 조인 (SQL-99)

이전

```
-- 1. 컬럼 목록 나열
-- 어느 테이블의 컬럼인지 앞에 테이블.으로 명시해 줘야 한다
SELECT e.employee_id, e.department_id, d.department_name
-- 2. 원하는 데이터가 어느 테이블에 있는가?
      FROM employees e, departments d
-- 3. 여러 테이블에 있다면 각 테이블의 공통 컬럼
      WHERE e.department_id = d.department_id
      AND last_name = 'King';
```

이후

```
SELECT e.employee_id, e.department_id, d.department_name
// 테이블 간 관계
    FROM employees e INNER JOIN departments d
// 공통 컬럼에 on 사용
    ON e.department_id = d.erpartment_id
    WHERE last_name = 'king';
```

3개 이상 조인

```
테이블1 JOIN 테이블 2
ON 공통컬럼1 = 공통컬럼1
JOIN 테이블 3
ON 공통컬럼2 = 공통컬럼2
```

실습 2 [hr]

- 3개 이상 테이블을 조인하여 사원 이름, 이메일, 부서 번호, 부서 이름, 직종 번호(job_id), 직종 이름(job_title)을 출력
- where

```
SELECT e.first_name, e.last_name, d.department_id, d.department_name, j.job_id, job_ti
tle
FROM employees e, departments d, jobs j
WHERE e.department_id = d.department_id AND e.job_id = j.job_id;
```

- ANSI

```
SELECT e.first_name, e.last_name, d.department_id, d.department_name, j.job_id, job_ti
tle
FROM employees e INNER JOIN departments d
```

```
ON e.department_id = d.department_id
INNER JOIN jobs j
ON e.job_id = j.job_id;
```

- ‘Seattle’ (city)에 근무하는 사원 이름, 부서 번호, 직종 번호, 직종 이름, 도시 이름을 출력
- where

```
SELECT e.first_name, e.last_name, e.department_id, j.job_id, j.job_title, city
FROM employees e, jobs j, departments d, locations l
WHERE e.job_id = j.job_id
AND e.department_id = d.department_id
AND d.location_id = l.location_id;
AND l.city = 'Seattle';
```

- ANSI

```
SELECT e.first_name, e.last_name, e.department_id, j.job_id, j.job_title, city
FROM employees e INNER JOIN jobs j
ON e.job_id = j.job_id
INNER JOIN departments d
ON e.department_id = d.department_id
INNER JOIN locations l
ON d.location_id = l.location_id;
WHERE l.city = 'Seattle';
```

Self JOIN

- ‘Kochhar’의 직속 상사 정보 출력 [hr]

```
SELECT A.last_name || '의 매니저는 ' || B.last_name||'이다.'
    FROM employees A, employees B
    WHERE A.manager_id = B.employee_id
    AND A.last_name = 'Kochhar';
```

실습 3 [kosa 02]

- 학생 중에 동명이인 검색

```
SELECT '학번 ' || A.sno || A.sname || ' 의 동명이인은' || '학번 ' || B.sno || B.sname ||
 '이다.'
    FROM student A, student B
```

```
    WHERE A.sname = B.sname
    AND A.sno != B.sno;
```

```
SELECT DISTINCT A.sno, A.sname
    FROM student A, student B
    WHERE A.sname = B.sname
    AND A.sno != B.sno;
```

OUTER JOIN

- 먼저 row 개수 체크

```
SELECT * FROM employees; => 107 row
```

- INNER JOIN

```
SELECT e.employee_id, e.department_id, d.department_name
    FROM employee e, departments d
    WHERE e.department_id = d.department_id; => 106 row
-- 한 명이 빠짐 (department_id의 값이 null인 경우)
-- INNER JOIN으로 짜면 누락되는 경우도 있다
```

- OUTER JOIN

```
SELECT e.employee_id, e.department_id, d.departmnet_name
    FROM employee e, departments d
    -- 데이터가 누락된 곳의 반대에 +를 한다
    WHERE e.department_id = d.department_id(+);
```

- ANSI JOIN

```
SELECT e.employee_id, e.department_id, d.departmnet_name
    -- 누락된 곳에 방향 표기 LEFT, RIGHT
    FROM employee e LEFT JOIN departments d
    ON e.department_id = d.department_id;
```

실습 4 [kosa 01]

- 등록된 과목에 대한 모든 교수를 검색하라

- 누락된 교수가 없도록 등록하지 않은 교수도 출력하라

```
    SELECT * FROM professor; -- 36개

    SELECT * -- 29개
    FROM professor p, course c
    WHERE p.pno = c.pno;

-- professor 기준
    SELECT *
    FROM professor p, course c
    -- professor가 누락되었으니 course에 + 하기
    WHERE p.pno = c.pno(+);

    SELECT *
    -- professor가 누락되었으니 누락된 방향인 왼쪽
    FROM professor p LEFT JOIN course c
 -- FROM course c RIGHT JOIN professor p
    ON p.pno = c.pno;

-- course 기준
    SELECT * FROM course; -- 32개
    SELECT *
    FROM course c, professor p
    -- course 가 누락되었으니 professor에 + 하기
    WHERE c.pno = p.pno(+);

-- FULL JOIN (양 테이블을 모두 만족시키기 위해 행이 더 늘어남)
    SELECT c.cno, c.cname, c.st_num, p.pname -- 39개
    FROM course c FULL JOIN professor p
    ON c.pno = p.pno;
```

실습 5 [hr]

- 이름이 ‘Himuro’의 사원의 부서명

```
SELECT first_name, last_name, department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id(+) AND e.last_name = 'Himuro';
```

```
SELECT first_name, last_name, department_name
FROM employees e LEFT JOIN departments d
ON e.department_id = d.department_id
WHERE e.last_name = 'Himuro';
```

- 직종명이 ‘Accountant’인 사원의 이름과 부서명

```
SELECT first_name, last_name, department_name, job_title
FROM employees e, departments d, jobs j
WHERE job_title = 'Accountant' AND e.department_id(+) = d.department_id AND j.job_id =
e.job_id;
```

```
SELECT first_name, last_name, department_name, job_title
FROM employees e LEFT JOIN departments d
ON e.department_id = d.department_id
LEFT JOIN jobs j
ON e.job_id = j.job_id
WHERE job_title = 'Accountant';
```

- 커미션을 받는 사람의 이름과 그가 속한 부서를 출력

```
SELECT first_name, last_name, department_name
FROM employees e, departments d
WHERE e.department_id(+) = d.department_id AND commission_pct > 0;
```

```
SELECT first_name, last_name, department_name
FROM employees e RIGHT JOIN departments d
ON e.department_id = d.department_id
WHERE e.commission_pct > 0;
```

서브쿼리

하위질의문

- WHERE, HAVING 절 → 하위 질의문
- FROM절 하위 질의 문 → n-tier: table 대체

실습 6.

- 사원의 평균 급여보다 많이 받는 사원의 수

```
SELECT AVG(salary) FROM employees
-- 6461.8317757...

SELECT last_name, salary
    FROM employees
    -- 실행 결과 값을 복사해야 하는 단점이 있다
    WHERE salary > 6461.8317
```

```
-- 서브 쿼리문

SELECT last_name, salary
    FROM employees
-- 그 값에 해당하는 쿼리문 집어넣기
    WHERE salary > (SELECT AVG(salary) FROM employees)
```

서브 쿼리 작성 순서

1. 서브쿼리문 먼저 작성
    - 결과 값을 확인해야 하기 때문
    - 메인 쿼리에 영향 끼침

실습 7. [hr]

- ‘Chen’ 보다 salary를 더 많이 받는 사원의 목록을 출력하라

```
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (SELECT salary
FROM employees
WHERE last_name = 'Chen');
```

[kosa 01]

- ‘정의찬’과 부서(dept)가 다르지만 동일한 업무(job)을 수행하는 사원 목록

```
SELECT * FROM emp
WHERE job = (SELECT job FROM emp
WHERE ename = '정의찬')
And dno != (SELECT dno FROM emp
WHERE ename = '정의찬');
```

- ‘관우’보다 일반화학 과목의 학점(grade)이 낮은 학생의 명단
    - 학점은 알파벳으로 출력

```
SELECT grade FROM student s, course c, score r, scgrade g
    WHERE s.sno = r.sno
    AND c.cno = r.cno
    AND cname = '일반화학'
    AND sname = '관우'
```

```
    AND result BETWEEN loscore AND hiscore

SELECT s.sno., sname, grade
    FROM student s, course c, score r, scgrade g
    WHERE s.sno = r.sno
    AND c.cno = r.cno
    AND cname = '일반화학'
    AND result BETWEEN loscore AND hiscore
    AND grade > (SELECT grade FROM student s, course c, score r, scgrade g
    WHERE s.sno = r.sno
    AND c.cno = r.cno
    AND cname = '일반화학'
    AND sname = '관우'
    AND result BETWEEN loscore AND hiscore);
```

Having절

실습 8.

- 부서 중 가장 평균 급여를 많이 받는 부서를 검색

```
-- 평균 급여의 최댓값
SELECT MAX(AVG(sal)) FROM emp
    GROUP BY dno

SELECT dno FROM emp
    GROUP BY dno
    HAVING AVG(sal) = (SELECT MAX(AVG(sal)) FROM emp
                        GROUP BY dno)
```

- [kosa 01] 학생 인원 수가 가장 많은 학과를 검색

```
SELECT major FROM student
-- GROUP BY. 절에서는 사용한 컬럼밖에 쓸 수 없다
GROUP BY major
-- 최고 값과 같으면 가장 많은 학과가 된다
Having COUNT(*) =(SELECT MAX(count(*)) FROM student
GROUP BY major);
```

- [kosa 01] 학생 중 기말고사 평균 성적이 가장 낮은 학생의 정보

```
SELECT MIN(AVG(result)) FROM score
    GROUP BY sno

SELECT s.sno, sname
      FROM student s, score r
      WHERE s.sno = r.sno
      GROUP BY s.sno, sname
```

```
      HAVING AVG(result) = (SELECT MIN(AVG(result)) FROM score
                            GROUP BY sno)
```

- 화학과 1학년 학생 중에 평점이 평균 이하인 학생

```
SELECT AVG(avr) FROM student
    WHERE major = '화학'
    AND syear = 1

SELECT * FROM student
  WHERE major = '화학'
  AND syear = 1
  AND avr = (SELECT AVG(avr) FROM student)
            WHERE major = '화학'
            AND syear = 1);
```

다중 컬럼, 다중 로우

- 문자열 비교 in 연산자
- 직무(job_id)별 사원의 최대 월급 내력 출력

```
SELECT MAX(salary) FROM employees
    GROUP BY job_id

SELECT employee_id, last_name, salary, job_id
      FROM employees;
    -- 다중 로우
      WHERE salary = SELECT(MAX(salary) FROM employees
          GROUP BY job_id;

-- 다중 로우/다중 컬럼 처리
SELECT employee_id, last_name, salary, job_id
      FROM employees;
    -- 다중 로우일 때는 IN 연산자 사용
      where (salary, job_id) IN(SELECT(MAX(salary)) FROM employees
          GROUP BY job_id);
```

- IN: 검색된 값 중 하나만 일치하면 참
- ANY: 검색된 값 중 조건에 맞는 것이 하나 이상이면 참
- ALL: 검색된 값 중에서 조건이 모두 일치하면 참

```
컬럼 > max() => 컬럼 > ALL (서브쿼리)
-- 가장 큰 값보다 크다

컬럼 < min() => 컬럼 < ALL (서브쿼리)
```

```
-- 가장 작은 값보다 작다

컬럼 > min() => 컬럼 > ANY (서브쿼리)
-- 가장 작은 값보다는 크다

컬럼 < max() -> 컬럼 < ANY (서브쿼리)
-- 가장 큰 값보다는 작다
```

실습 9.

- [kosa01] 10번 부서에서 가장 작은 급여자보다 작게 받는 급여자 출력

```
SELECT MIN(sal) FROM emp
      WHERE dno = 10

SELECT eno, ename, sal, dno
      FROM emp
      WHERE sal < (SELECT MIN(sal) FROM emp
      WHERE dno = 10);

SELECT eno, ename, sal, dno
      FROM emp
      -- 모든 값보다 더 작은 값 (그룹함수 사용하지 않아도 된다)
      WHERE sal < ALL(SELECT sal FROM emp
      WHERE dno = 10);
```

- [hr] 부서번호 30번의 최대 급여자보다 급여가 높은 사원

```
SELECT employee_id, first_name, last_name, job_id, salary, department_id FROM employee
s
WHERE salary > ALL
(SELECT salary FROM employees
WHERE department_id = 30);
```

- [hr] 부서번호 30번의 최대 급여자보다 급여가 낮은 사원

```
SELECT employee_id, first_name, last_name, job_id, salary, department_id FROM employee
s
WHERE salary < ANY
(SELECT salary FROM employees
WHERE department_id = 30);
```

- [kosa01] ‘손하늘’과 동일한 관리자(mgr)의 관리를 받으면서 업무도 같은 사원

```
SELECT eno, ename, job, mgr, dno FROM emp
WHERE ename != '손하늘' AND (mgr, job) IN
```

```
(SELECT mgr, job FROM emp
WHERE ename = '손하늘');
```

- [kosa01] 화학과 학생과 평점이 동일한 학생

```
SELECT * FROM student
WHERE major != '화학' AND avr IN (SELECT avr FROM student
WHERE major = '화학');
```

- [kosa01] 화학과 학생과 같은 학년에서 평점이 동일한 학생

```
SELECT sname, syear, major, avr FROM student
WHERE major != '화학' AND (avr, syear) IN(
SELECT avr, syear FROM student
WHERE major = '화학');
```

FROM절: 서브쿼리(n-tier)

실습 10. [hr]

- 입사 순서 오래된 5명 출력

```
SELECT employee_id, last_name, hire_date FROM employees
      ORDER BY hire_date

-- 3개의 컬럼이 다 들어가게 함
SELECT ROWNUM, alias.*
      FROM (SELECT employee_id, last_name, hire_date FROM employees
      -- 날짜 순서대로 정렬했기 때문에, 상위 5개만 뽑으면 된다
      ORDER BY hire_date) alias
      -- 데이터 값의 출력 인덱스
      WHERE ROWNUM <= 5;
```

- 급여 많이 받는 순서 3명의 사원 정보

```
SELECT ROWNUM, alias.*
FROM (SELECT employee_id, first_name, last_name, job_id, department_id, salary
FROM employees
ORDER BY salary DESC) alias
WHERE ROWNUM <= 3;
```

ROWNUM

- 쿼리를 통해 가져온 데이터를 이용하여 번호를 매기는 방식

📌 주의사항: **반드시 1번부터** 포함되어야 한다

- 페이징 처리 할 때 쓰이는 로직

```
CREATE TABLE board (
  seq NUMBER,
  title VARCHAR2(50),
  writer VARCHAR2(50),
  contents VARCHAR2(200),
  regdate date,
  hitcnt NUMBER
);

INSERT INTO board VALUES(9, 'a1', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(2, 'a7', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(3, 'a5', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(5, 'a4', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(7, 'a2', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(1, 'a8', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(4, 'a3', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(6, 'a9', 'a', 'a', sysdate, 0);
INSERT INTO board VALUES(8, 'a6', 'a', 'a', sysdate, 0);

SELECT * FROM board
    ORDER BY seq;

-- 1이 포함되어 있지 않아서 출력 안 됨
SELECT ROWNUM, temp.*
      FROM(SELECT * FROM board
              ORDER BY seq) tmp
      WHERE ROWNUM BETWEEN 6 AND 10;
```

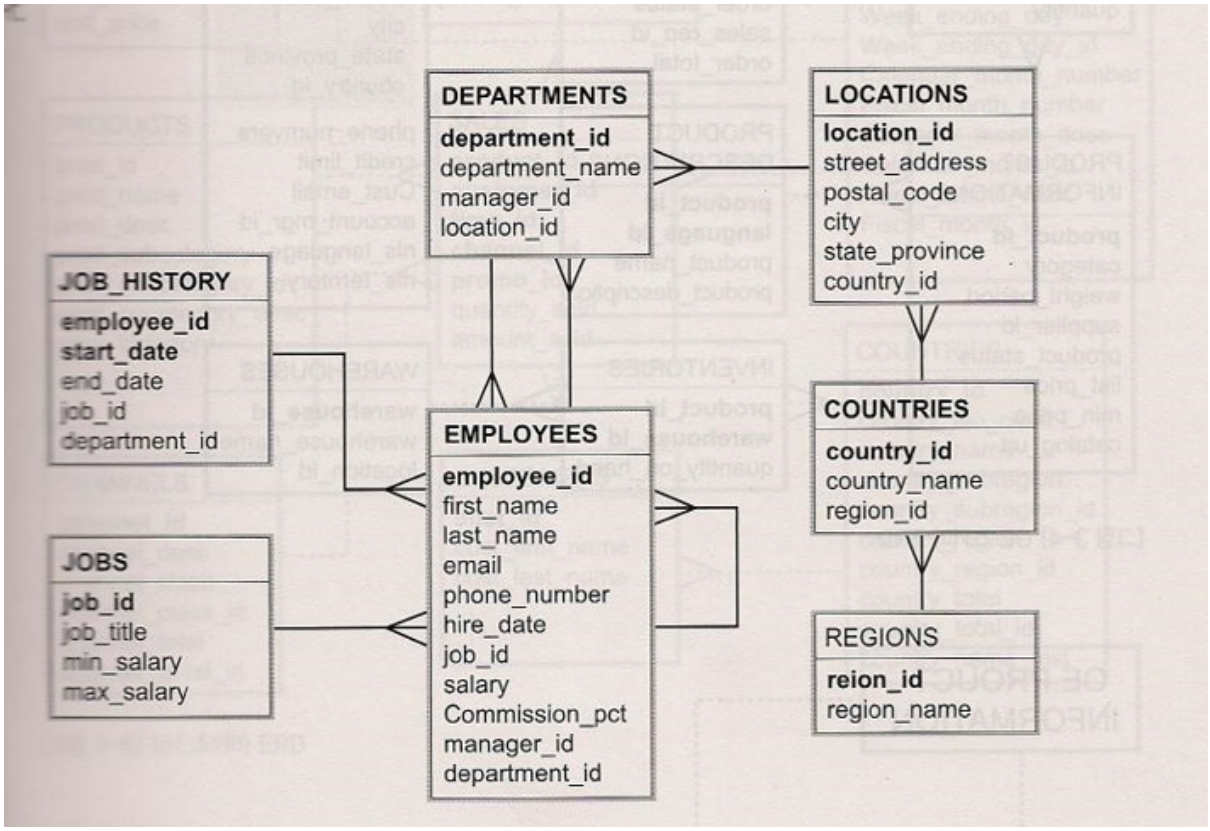
- 문제 해결

```
-- 하나 더 서브 쿼리문으로 만들기
SELECT ROWNUM AS ROW_NUM, temp.*
      FROM(SELECT * FROM board
              ORDER BY seq) temp

-- 실제 페이징 처리도 이러한 방식으로 진행
SELECT * FROM(
        SELECT ROWNUM AS ROW_NUM, temp.*
            FROM(SELECT * FROM board
              ORDER BY seq) temp
        )
        WHERE ROW_NUM BETWEEN 6 AND 10;
```

과제

- 정답 확인하기


DEPARTMENTS
department_id
department_name
manager_id
location_id
LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id
JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id
EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
Commission_pct
manager_id
department_id
COUNTRIES
country_id
country_name
region_id
JOBS
job_id
job_title
min_salary
max_salary
REGIONS
reion_id
region_name

각 테이블 상세구조

emp(사원)		dept(부서)		salgrade(급여등급)	
컬럼명	***내용***	***컬럼명***	***내용***	***컬럼명***	***내용***
eno	사번	dno	부서번호	grade	급여등급
ename	이름	dname	부서명	losal	하한값
job	업무	loc	지역	hosal	상한값
mgr	관리자 사번	director	부서장사번		
hdate	입사일				
sal	급여(월)				
comm	보너스				
dno	부서번호				

student(학생)		professor(교수)		course(과목)	
컬럼명	***내용***	***컬럼명***	***내용***	***컬럼명***	***내용***
sno	학번	pno	교수번호	cno	과목번호
sname	이름	pname	이름	cname	과목명
sex	성별	section	소속학과	st_num	학점수
syear	학년	orders	직위	pno	교수번호
major	학과	hiredate	부임일		
avr	평점				

score(기말고사 점수)		scgrade(점수등급)	
컬럼명	***내용***	***컬럼명***	***내용***
sno	학번	grade	등급
cno	과목번호	hiscore	상한값
result	점수	loscore	하한값

Q1 급여(SAL)가 2000 초과인 사원들의 부서 정보, 사원 정보를 오른쪽과 같이 출력해 보세요 (단 SQL-99 이전 방식과 SQL-99 방식을 각각 사용하여 작성하세요).

:: 결과 화면

DEPTNO	DNAME	EMPNO	ENAME	SAL
10	ACCOUNTING	7782	CLARK	2450
10	ACCOUNTING	7839	KING	5000
20	RESEARCH	7566	JONES	2975
20	RESEARCH	7788	SCOTT	3000
20	RESEARCH	7902	FORD	3000
30	SALES	7698	BLAKE	2850

```
-- 1.
SELECT e.deptno, dname, empno, ename, sal
FROM emp e, dept d
WHERE e.deptno = d.deptno AND sal > 2000;

-- 2.
SELECT e.deptno, dname, empno, ename, sal
FROM emp e JOIN dept d
ON e.deptno = d.deptno
WHERE sal > 2000;
```

Q2 오른쪽과 같이 각 부서별 평균 급여, 최대 급여, 최소 급여, 사원수를 출력해 보세요(단 SQL-99 이전 방식과 SQL-99 방식을 각각 사용하여 작성하세요).

:: 결과 화면

DEPTNO	DNAME	AVG_SAL	MAX_SAL	MIN_SAL	CNT
10	ACCOUNTING	2916	5000	1300	3
20	RESEARCH	2175	3000	800	5
30	SALES	1566	2850	950	6

```
SELECT e.deptno, d.dname, avg(e.sal) as AVG_SAL, max(e.sal) as MAX_SAL, min(e.sal) as
 MIN_SAL, count(d.deptno) as cnt
FROM emp e, dept d
WHERE e.deptno = d.deptno
GROUP BY deptno;

// 다시
```

Q3 모든 부서 정보와 사원 정보를 오른쪽과 같이 부서 번호, 사원 이름순으로 정렬하여 출력해 보세요(단 SQL-99 이전 방식과 SQL-99 방식을 각각 사용하여 작성하세요).

:: 결과 화면

DEPTNO	DNAME	EMPNO	ENAME	JOB	SAL
10	ACCOUNTING	7782	CLARK	MANAGER	2450
10	ACCOUNTING	7839	KING	PRESIDENT	5000
10	ACCOUNTING	7934	MILLER	CLERK	1300
20	RESEARCH	7876	ADAMS	CLERK	1100
20	RESEARCH	7902	FORD	ANALYST	3000
20	RESEARCH	7566	JONES	MANAGER	2975
20	RESEARCH	7788	SCOTT	ANALYST	3000
20	RESEARCH	7369	SMITH	CLERK	800
30	SALES	7499	ALLEN	SALESMAN	1600
30	SALES	7698	BLAKE	MANAGER	2850
30	SALES	7900	JAMES	CLERK	950
30	SALES	7654	MARTIN	SALESMAN	1250
30	SALES	7844	TURNER	SALESMAN	1500
30	SALES	7521	WARD	SALESMAN	1250
40	OPERATIONS				

```
SELECT d.deptno, dname, empno, ename, job, sal
FROM dept d, emp e
where e.deptno = d.deptno
```

Q4 다음과 같이 모든 부서 정보, 사원 정보, 급여 등급 정보, 각 사원의 직속 상관의 정보를 부서 번호, 사원 번호 순서로 정렬하여 출력해 보세요(단 SQL-99 이전 방식과 SQL-99 방식을 각각 사용하여 작성하세요).

:: 결과 화면

DEPTNO	DNAME	EMPNO	ENAME	MGR	SAL	DEPTNO_1	LOSAL	HISAL	GRADE	MGR_EMPNO	MGR_ENAME
10	ACCOUNTING	7782	CLARK	7839	2450	10	2001	3000	4	7839	KING
10	ACCOUNTING	7839	KING		5000	10	3001	9999	5		
10	ACCOUNTING	7934	MILLER	7782	1300	10	1201	1400	2	7782	CLARK
20	RESEARCH	7369	SMITH	7902	800	20	700	1200	1	7902	FORD
20	RESEARCH	7566	JONES	7839	2975	20	2001	3000	4	7839	KING
20	RESEARCH	7788	SCOTT	7566	3000	20	2001	3000	4	7566	JONES
20	RESEARCH	7876	ADAMS	7788	1100	20	700	1200	1	7788	SCOTT
20	RESEARCH	7902	FORD	7566	3000	20	2001	3000	4	7566	JONES
30	SALES	7499	ALLEN	7698	1600	30	1401	2000	3	7698	BLAKE
30	SALES	7521	WARD	7698	1250	30	1201	1400	2	7698	BLAKE
30	SALES	7654	MARTIN	7698	1250	30	1201	1400	2	7698	BLAKE
30	SALES	7698	BLAKE	7839	2850	30	2001	3000	4	7839	KING
30	SALES	7844	TURNER	7698	1500	30	1401	2000	3	7698	BLAKE
30	SALES	7900	JAMES	7698	950	30	700	1200	1	7698	BLAKE
40	OPERATIONS										

SE

1. 문제) ‘Patel’가 속해있는 부서의 모든 사람의 사원번호, 이름, 입사 일, 급여를 출력하라.

2. 문제) ‘Austin'의 직무(job)와 같은 사람의 이름, 부서명, 급여, 직무를 출력하라.

3. 문제) 'Seo'의 급여와 같은 사원의 사원번호, 이름, 급여를 출력하라.

4. 문제) 급여가 30번 부서의 최고 급여보다 높은 사원의 사원번호, 이름, 급여를 출력하라.

5. 문제) 급여가 30번 부서의 최저 급여보다 높은 사원의 사원번호,

이름, 급여를 출력하라.

6. 문제) 전체 사원의 평균 임금보다 많은 사원의 사원번호, 이름,
부서명, 입사일, 지역(city), 급여를 출력하라.

7. 문제) 100번 부서 중에서 30번 부서에는 없는 업무를 하는 사원의
사원번호, 이름, 부서명, 입사일, 지역을 출력하라.

Q1 전체 사원 중 ALLEN과 같은 직책(JOB)인 사원들의 사원 정보, 부서 정보를 다음과 같이 출력하는 SQL문을 작성하세요.

:: 결과 화면

JOB	EMPNO	ENAME	SAL	DEPTNO	DNAME
SALESMAN	7499	ALLEN	1600	30	SALES
SALESMAN	7844	TURNER	1500	30	SALES
SALESMAN	7654	MARTIN	1250	30	SALES
SALESMAN	7521	WARD	1250	30	SALES

Q2 전체 사원의 평균 급여(SAL)보다 높은 급여를 받는 사원들의 사원 정보, 부서 정보, 급여 등급 정보를 출력하는 SQL문을 작성하세요(단 출력할 때 급여가 많은 순으로 정렬하되 급여가 같을 경우에는 사원 번호를 기준으로 오름차순으로 정렬하세요).

:: 결과 화면

EMPNO	ENAME	DNAME	HIREDATE	LOC	SAL	GRADE
7839	KING	ACCOUNTING	1981-11-17	NEW YORK	5000	5
~~7788~~	~~SCOTT~~	~~RESEARCH~~	~~1987-04-19~~	~~DALLAS~~	~~3000~~	~~4~~
7902	FORD	RESEARCH	1981-12-03	DALLAS	3000	4
7566	JONES	RESEARCH	1981-04-02	DALLAS	2975	4
7698	BLAKE	SALES	1981-05-01	CHICAGO	2850	4
7782	CLARK	ACCOUNTING	1981-06-09	NEW YORK	2450	4

Q3 10번 부서에 근무하는 사원 중 30번 부서에는 존재하지 않는 직책을 가진 사원들의 사원 정보, 부서 정보를 다음과 같이 출력하는 SQL문을 작성하세요.

:: 결과 화면

EMPNO	ENAME	JOB	DEPTNO	DNAME	LOC
7839	KING	PRESIDENT	10	ACCOUNTING	NEW YORK

Q4 직책이 SALESMAN인 사람들의 최고 급여보다 높은 급여를 받는 사원들의 사원 정보, 급여 등급 정보를 다음과 같이 출력하는 SQL문을 작성하세요(단 서브쿼리를 활용할 때 다중행 함수를 사용하는 방법과 사용하지 않는 방법을 통해 사원 번호를 기준으로 오름차순으로 정렬하세요).

:: 결과 화면

EMPNO	ENAME	SAL	GRADE
7566	JONES	2975	4
7698	BLAKE	2850	4
7782	CLARK	2450	4
~~7788~~	~~SCOTT~~	~~3000~~	~~4~~
7839	KING	5000	5
7902	FORD	3000	4