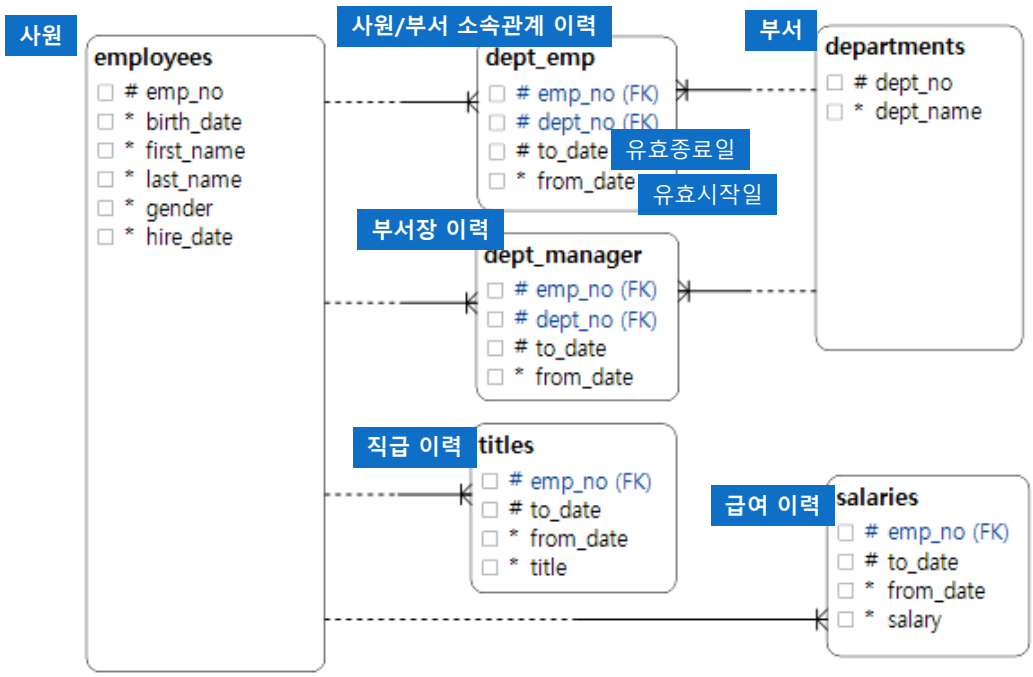
Data Analytics 과정 특강

고급 SQL 이해와 활용

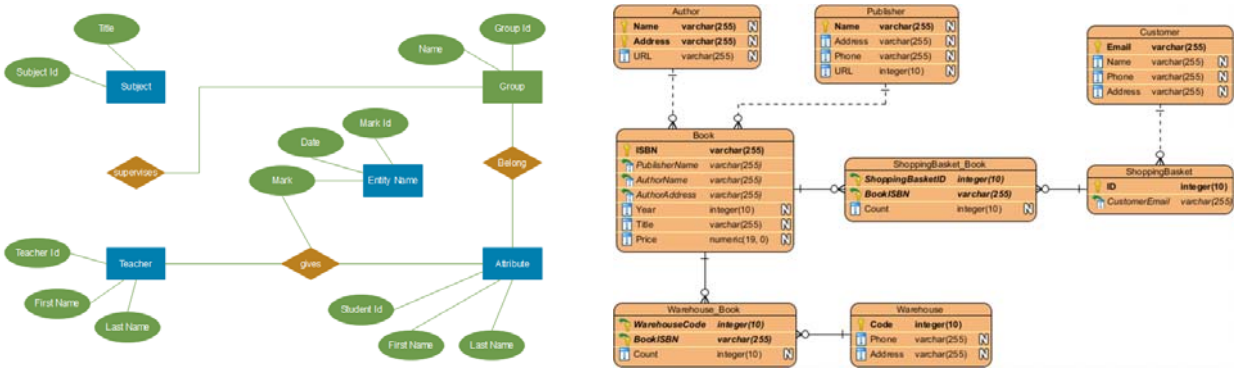
실습 1. 예제 데이터 모델 (ERD)

1-1. 예제 데이터 ERD



개념 이해 1. ERD(Entity Relationship Diagram)

- 엔터티(entity)와 관계(relationship)을 표현한 다이어그램
  - 구성요소 : 엔터티(Entity), 속성(Attribute), 제약조건(constraints), 관계(Relationship)
  - 관계 표현의 관점 : 대응수(mapping cardinality), 필수/선택(mandatory/optional), 절대종속/상대종속(strong/weak)
- 다양한 ERD 표기법 존재



....

실습 2. 예제 테이블 생성

2-1. Create table 예제

```
CREATE TABLE dept_manager (
   emp_no       NUMBER(6),
   dept_no      VARCHAR2(4)         NOT NULL,
   to_date      VARCHAR2(8)         NOT NULL,
   from_date    VARCHAR2(8)         NOT NULL,
   CONSTRAINTS DEPT_MANAGER_PK PRIMARY KEY (dept_no, to_date),  기본키
   CONSTRAINTS DEPT_MANAGER_FK1 FOREIGN KEY (emp_no)  REFERENCES employees (emp_no)    ON DELETE
CASCADE,  외래키
   CONSTRAINTS DEPT_MANAGER_FK2 FOREIGN KEY (dept_no) REFERENCES departments (dept_no) ON DELETE
CASCADE  외래키
);
```

개념 이해 2. 제약조건(Integrity constraints)

- 제약조건 : DB 무결성(integrity) 보장을 위한 장치
- 제약조건의 종류
  - Not Null
    - ✓ NULL을 허용하지 않음
    - ✓ Eg) name varchar2(10) NOT NULL
  - UNIQUE
    - ✓ 컬럼값의 유일성(uniqueness) 보장
    - ✓ Eg) bno number UNIQUE
  - Primary Key (기본키)
    - ✓ 해당 테이블의 대표 속성으로서 not null과 uniqueness 보장
    - ✓ Eg) bno number primary key
  - Foreign Key (외래키)
    - ✓ 참조 무결성(referential constraint) 보장을 위한 것으로 참조하는 테이블의 기본키 값만 가질 수 있음
    - ✓ Eg) constraints fk_cateId foreign key references cateTable(cateId)
  - Check
    - ✓ 입력될 수 있는 데이터의 범위 제한
    - ✓ Eg) bno number primary key CHECK (bno between 1 and 1000)

개념 이해 2. 제약조건(Integrity constraints)

- 참조무결성 제약조건을 보장하기 위한 4가지 옵션
  - 참조관계(부모-자식 관계)의 테이블에서 부모 테이블의 투플 삭제 시, 참조무결성 위배 가능성
  - 부모 테이블의 투플 삭제 시, 이를 참조하는 자식테이블 투플의 처리방법 4가지

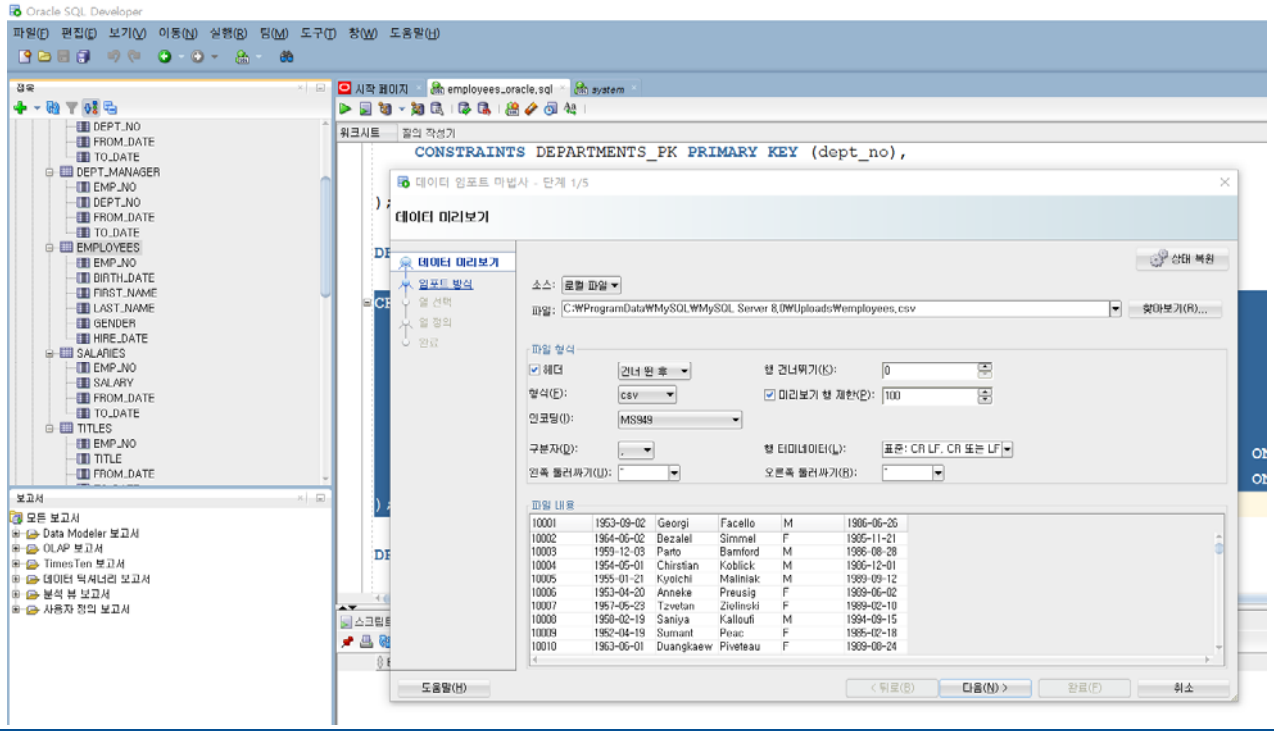
옵션	설명
RESTRICTED	자식 테이블에서 참조하고 있는 부모 테이블 투플 삭제(수정) 거부
CASCADE	부모 테이블의 투플 삭제(수정) 시, 이를 참조하는 자식 테이블 투플도 같이 삭제(수정)
DEFAULT	부모 테이블의 투플 삭제(수정) 시, 이를 참조하는 자식 테이블 투플의 속성값을 미리 지정한 기본값(default value)로 변경
NULL	부모 테이블의 투플 삭제 시(수정), 이를 참조하는 자식 테이블 투플의 속성값을 NULL로 변경(단, NULL값이 허용될 경우)

Oracle examples :

```
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE NO ACTION
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE CASCADE
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE SET DEFAULT '10000'
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE SET NULL
```

실습 3. 데이터 적재(Data Importing)

3-1. SQL Developer 상에서의 table import 예제 화면



실습 4. 데이터 정제(Data Cleansing)

4-1. 동시에 2개 부서에 소속된 사원을 찾아서 to_date(유효종료일)이 가장 큰 것만 취하고 나머지는 삭제

4-1-① 동시에 두개 부서에 소속된 사원 현황

```
select *
from dept_emp
where (emp_no, from_date)
        in ( select emp_no
                  , from_date
             from dept_emp
             group by emp_no, from_date
             having count(*) > 1 ) ;
```

4-1-② 중복 이력 중 하나만 선택하고 나머지 삭제 (cleansing rule : to_date가 가장 큰 것 선택)

```
delete
from dept_emp
where (emp_no, from_date, to_date)
      in ( select emp_no, from_date, to_date
           from dept_emp
           where (emp_no, from_date) in ( subquery ) ;
```

subquery : 삭제 할 투플의 (emp_no, from_date)를 구하는 SQL

실습 4. 데이터 정제(Data Cleansing)

4-1. 동시에 2개 부서에 소속된 사원을 찾아서 to_date(유효종료일)이 가장 큰 것만 취하고 나머지는 삭제

subquery : 삭제 할 투플의 (emp_no, from_date)를 구하는 SQL

```
select emp_no, from_date, to_date
from dept_emp
where (emp_no, from_date) in ( select emp_no
                                    , from_date
                               from dept_emp
                               group by emp_no, from_date
                               having count(*) > 1 )
                               minus
                               select emp_no, from_date, to_date
                               from dept_emp
                               where (emp_no, to_date)
                                     in ( select emp_no, max(to_date)
                                          from dept_emp
                                          where (emp_no, from_date) in ( select emp_no
                                                                              , from_date
                                                                         from dept_emp
                                                                         group by emp_no, from_date
                                                                         having count(*) > 1 )
                                          group by emp_no ) ;
```

4-1-③ data cleansing 후 PK 제약조건 add

```
ALTER TABLE dept_emp ADD CONSTRAINTS DEPT_EMP_PK PRIMARY KEY (emp_no, to_date) ;
```


실습 4. 데이터 정제(Data Cleansing)

4-2. 이력날짜 8자리로 맞추기 (예, '1992-05-10' → '19920510')

```
update salaries          -- 나머지 3개 이력테이블에도 동일하게 적용
set from_date = substr(from_date, 1, 4)||substr(from_date, 6, 2)||substr(from_date, 9, 2)
   , to_date = substr(to_date, 1, 4)||substr(to_date, 6, 2)||substr(to_date, 9, 2) ;
```

4-2-① Rollback segment 확장

rollback segments에 관련된 tablespace와 data file 및 사이즈 조회 (system 사용자로 로그인 후 조회)

```
select file_name, tablespace_name, bytes from dba_data_files ;
```

rollback segments에 관련된 data file size-up

```
ALTER DATABASE DATAFILE 'C:₩ORACLEXE₩APP₩ORACLE₩ORADATA₩XE₩UNDOTBS1.DBF' RESIZE 100M ;
```

4-2-② Column re-size

```
alter table dept_manager modify to_date varchar2(8) ;        -- 나머지 3개 이력테이블에도 동일하게 적용
alter table dept_manager modify from_date varchar2(8) ;      -- 나머지 3개 이력테이블에도 동일하게 적용
```

실습 4. 데이터 정제(Data Cleansing)

4-3. 현재 상태의 종료일자(to_date) : '99990101 -> 99991231'로 변경

```
update titles        -- 나머지 3개 이력테이블에도 동일하게 적용
set to_date = '99991231'
where to_date = '99990101' ;
```

4-4. 이력 관리 : 양편 넣기 -> 한편 넣기

① cleansing rule : to_date 컬럼값이 '99991231'이 아닌 투플을 대상으로 to_date를 하루 앞 당기게 수정

```
update dept_emp                    -- 나머지 3개 이력테이블에도 동일하게 적용
set to_date = to_char(to_date(to_date,'yyyymmdd') - 1, 'yyyymmdd')
where to_date <> '99991231' ;
```

② cleansing rule : 1)번 cleansing으로 인해 'to_date < from_date'인 튜플에 대해서 to_date를 from_date로 update

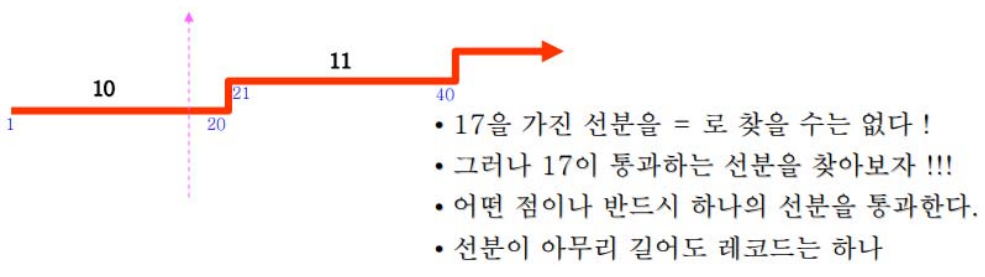
```
update salaries                    -- 나머지 3개 이력테이블에도 동일하게 적용
set to_date = from_date
where to_date < from_date ;
```

③ data cleansing 후 PK/FK 제약조건 추가

```
ALTER TABLE titles ADD CONSTRAINTS TITLES_PK PRIMARY KEY (emp_no, to_date) ;
ALTER TABLE titles ADD CONSTRAINTS TITLES_FK1 FOREIGN KEY (emp_no) REFERENCES employees (emp_no)
ON DELETE CASCADE ;
ALTER TABLE salaries ADD CONSTRAINTS SALARIES_PK PRIMARY KEY (emp_no, to_date) ;
ALTER TABLE salaries ADD CONSTRAINTS SALARIES_FK1 FOREIGN KEY (emp_no) REFERENCES employees
(emp_no) ON DELETE CASCADE ;
```


개념 이해 3. 이력 데이터 관리 기법 (1)

- 이력 관리 : 점(Event) vs. 선분



* 출처 : www.en-core.com

개념 이해 3. 이력 데이터 관리 기법 (2)

- 점(Event) 이력 모델

부서변경이력

사원번호	부서코드	변경일자	...
7788	10	1999/ 02/ 04	...
7788	30	2000/ 07/ 21	NULL
7788	20	2002/ 05/ 15	...
8123	40	2000/ 03/ 23	
8123	30	2001/ 11/ 05	
8123	10	2002/ 09/ 17	
...	...	...	

사원별, 부서개편일 전의 부서코드를 찾아라. (부서개편일 : 2002 년 3 월 1 일)

1 Subquery를 이용한 방법 (1)

```
SELECT 사원번호, 부서코드
FROM   부서변경이력
WHERE  변경일자 = (select MAX(변경일자) from 부서변경이력 where 변경일자 < '2002/03/01');
```

2 Subquery를 이용한 방법 (2)

```
SELECT 사원번호, 부서코드
FROM   부서변경이력
WHERE  ROWID = ( select /*+ index_desc(부서변경이력 idx_변경일자) */ ROWID  RID
                 from 부서변경이력 where 변경일자 < '2002/03/01 and rownum <= 1);
```

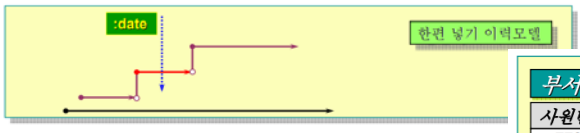
* 출처 : www.en-core.com

개념 이해 3. 이력 데이터 관리 기법 (3)

- 선분 이력 모델 (한편넣기)

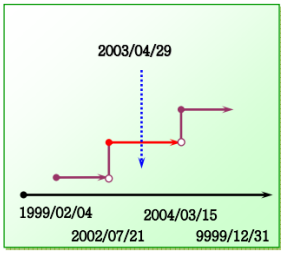
■ 정의
- '일자(Day)'로 이력관리가 되고 있는 엔터티에서, 동일 일자에 대해 이벤트가 최대 한번만 발생하는 경우에 사용되는 기법
- 시간(time)까지 관리하는 이력관리에서 적용

■ 조회 방법
- where :date >= 시작일 and :date <= 종료일
- where :date between 시작일 and 종료일



부서변경이력

사원번호	부서코드	유효시작일	유효종료일	~~~
7788	10	1999/ 02/ 04	2002/ 07/ 20	~~~
7788	30	2002/ 07/ 21	2004/ 05/ 14	NULL
7788	20	2004/ 05/ 15	9999/ 12/ 31	~~~
8123	40	2000/ 03/ 23	2001/ 11/ 04	
8123	30	2001/ 11/ 05	2002/ 09/ 16	
8123	10	2002/ 09/ 17	9999/ 12/ 31	
~~	~~~	~~~	~~~	



사번 '7788' 인 사람이, 2001 년 4 월 29 일에 근무했던 부서는 어디인가?

```
SELECT 사원번호, 부서코드
FROM   부서변경이력
WHERE  유효시작일 <= '2003/04/29'
AND    유효종료일 >= '2003/04/29'
AND    사원번호 = '7788'
```

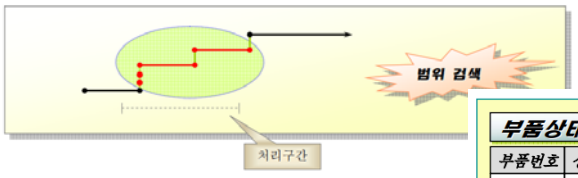
```
SELECT 사원번호, 부서코드
FROM 부서변경이력
WHERE '2003/04/29' BETWEEN 유효시작일
                       AND 유효종료일
AND  사원번호 = '7788'
```

* 출처 : www.en-core.com

개념 이해 3. 이력 데이터 관리 기법 (4)

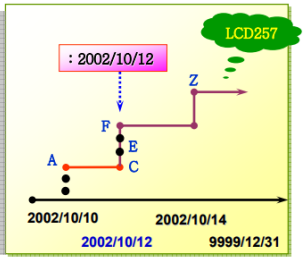
선분 이력 모델 (양편넣기)

- 용도
  - 이력에 이벤트까지 관리하고자 하는 경우 사용
  - 보험업무처럼 부문에 따라 이력을 보는 기준이 다른 경우
- 조회방법 : 원하는 정보의 형태에 따라 Query 문이 달라짐
  - ① where 시작일 between :구간1 and :구간2



부품상태변경이력

부품번호	상태코드	유효시작일	유효종료일	발생순번
CDR788	E	2002/10/12	9999/12/31	5
LCD257	A	2002/10/10	2002/10/12	1
LCD257	C	2002/10/12	2002/10/12	1
LCD257	E	2002/10/12	2002/10/12	2
LCD257	F	2002/10/12	2002/10/14	3
LCD257	Z	2002/10/14	9999/12/31	1
~~	~~~	~~~	~~~	~~~



1.코드번호 'LCD257' 인 부품의 당일 초기상태는 ? (2002/10/12 기준)

```
SELECT 부품번호, 상태코드, 발생순번 FROM 부품상태변경이력
WHERE 유효시작일  < '2002/10/12'
   AND 유효종료일 => '2002/10/12'
   AND 부품번호 = 'LCD257'
```

* 출처 : www.en-core.com

실습 5. 이력 조회

5-1. 부서별 현재 부서장의 사번(emp_no), 이름(first_name+last_name), 성별(gender), 입사일(hire_date), 직급(title), 급여(salary) 조회

```
select /*+ ordered use_nl(a b c d e) */ b.dept_name, c.emp_no, c.first_name||' '||c.last_name
     , c.gender, c.hire_date, d.title, e.salary
from dept_manager a, departments b, employees c, titles d, salaries e
where a.to_date = '99991231'
  and a.dept_no = b.dept_no
  and a.emp_no = c.emp_no
  and a.emp_no = d.emp_no and d.to_date = '99991231'
  and a.emp_no = e.emp_no and e.to_date = '99991231' ;
```

5-2. 'Lein Bale' 사원의 2000년 1월 1일 당시 소속부서, 직급, 급여 출력

5-2-① 조인 이용

과거 시점 조회는 BETWEEN 연산으로

```
select a.first_name, a.last_name
        , e.dept_name
        , c.title
        , d.salary
from employees a, dept_emp b, titles c, salaries d, departments e
where a.first_name = 'Lein' and a.last_name = 'Bale'
  and a.emp_no = b.emp_no and '20000101' between b.from_date and b.to_date
  and a.emp_no = c.emp_no and '20000101' between c.from_date and c.to_date
  and a.emp_no = d.emp_no and '20000101' between d.from_date and d.to_date
  and b.dept_no = e.dept_no ;
```

실습 5. 이력 조회

5-2. 'Lein Bale' 사원의 2000년 1월 1일 당시 소속부서, 직급, 급여 출력

5-2-② Scalar subquery 활용

scalar subquery

```
select a.first_name, a.last_name
     , (select dept_name from departments c where c.dept_no = b.dept_no) dept_name
     , (select title from titles d where d.emp_no = a.emp_no
                                 and '20000101' between d.from_date and d.to_date) title
     , (select salary from salaries e where e.emp_no = a.emp_no
                                 and '20000101' between e.from_date and e.to_date) salary
from employees a, dept_emp b
where a.first_name = 'Lein' and a.last_name = 'Bale'
  and a.emp_no = b.emp_no and '20000101' between b.from_date and b.to_date ;
```

5-2-③ 5-2-② SQL의 개선 : 어떤 면에서의 개선일까?

```
select a.first_name, a.last_name
     , (select min(dept_name) from departments c where c.dept_no = b.dept_no) dept_name
     , (select min(title) from titles d where d.emp_no = a.emp_no
                                 and '20000101' between d.from_date and d.to_date) title
     , (select min(salary) from salaries e where e.emp_no = a.emp_no
                                 and '20000101' between e.from_date and e.to_date) salary
from employees a, dept_emp b
where a.first_name = 'Lein' and a.last_name = 'Bale'
  and a.emp_no = b.emp_no and '20000101' between b.from_date and b.to_date ;
```


개념 이해 4. 서브쿼리(Sub-query)

- **서브쿼리(sub-query)**
  - SQL 문장의 하부 절에 사용하는 select 쿼리문
  - 서브쿼리 위치에 따른 명칭
    - ✓ SELECT문에 있는 서브쿼리 : **스칼라 서브쿼리(scalar subquery)**
    - ✓ FROM절에 있는 서브쿼리 : **인라인 뷰(inline view)**
    - ✓ WHERE절에 있는 서브쿼리 : **서브쿼리(subquery)**
  - 서브쿼리의 반환값에 따른 서브쿼리 종류
    - ✓ 단일 행 서브쿼리(Single-Row Subquery) : 서브쿼리의 결과가 1행
    - ✓ 다중 행 서브쿼리(Multiple-Row Subquery) : 서브쿼리의 결과가 여러 행
    - ✓ 다중 컬럼 서브쿼리(Multi-Column Subquery) : 서브쿼리의 결과가 여러 컬럼
  - Scalar subquery의 경우는 단일 행 서브쿼리만 허용 : 반드시 오직 하나의 행만 반환함을 보장해야 오류가 없음
  - 상호연관 서브쿼리(Correlated Subquery) : 메인쿼리의 값을 서브쿼리가 사용하고, 서브쿼리의 값을 받아서 메인쿼리가 계산하는 구조의 쿼리