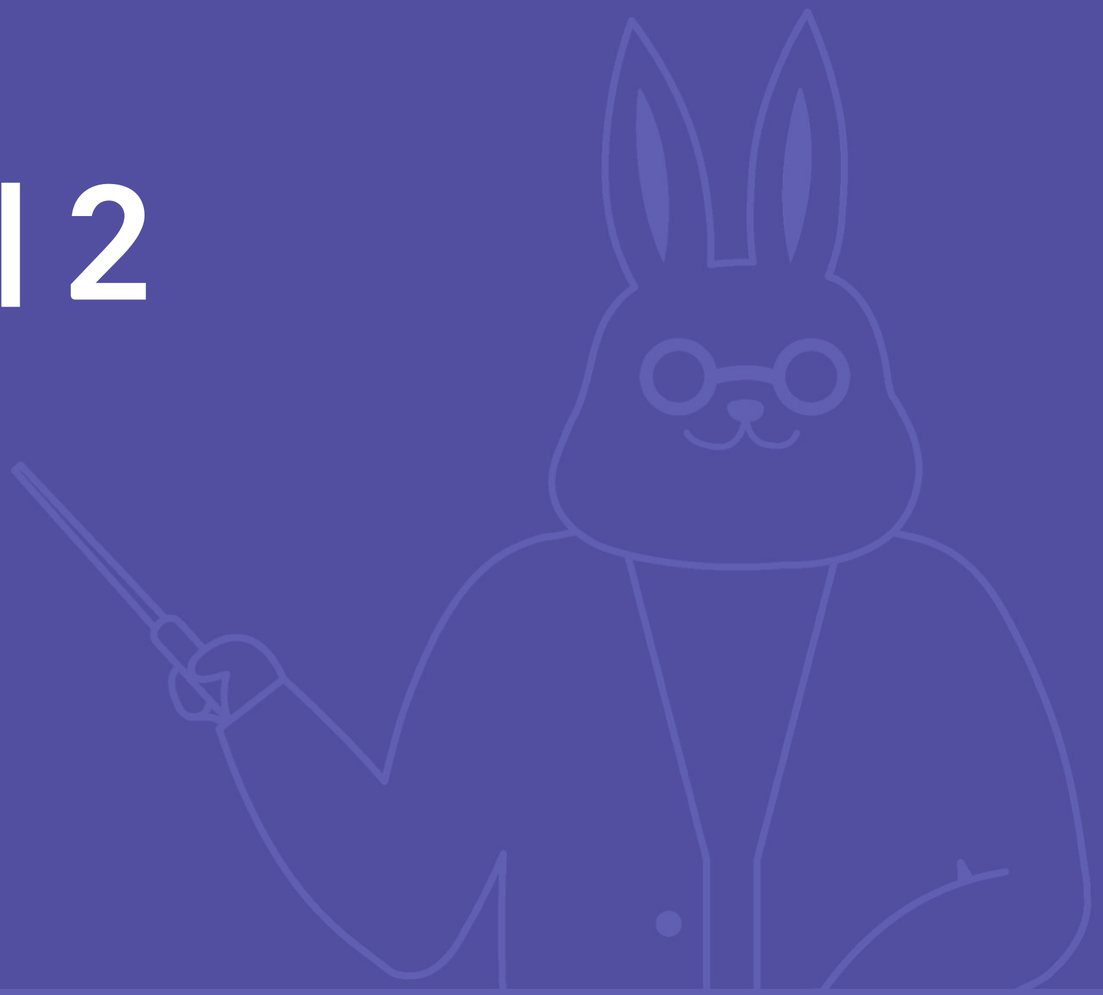
SQL로 데이터 다루기 2

03 서브쿼리 심화



/* elice */

목차

01. 동작하는 방식에 따른 서브쿼리 분류

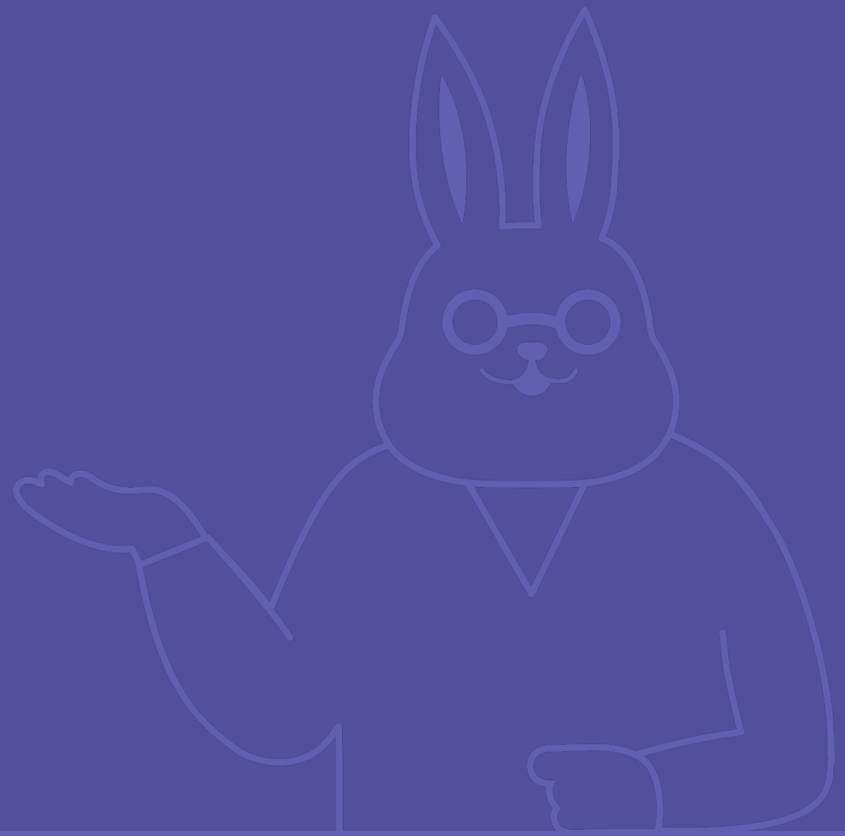
02. 반환되는 데이터 형태에 따른 서브쿼리 분류

03. 스칼라 서브쿼리

04. 뷰

01

동작하는 방식에 따른 서브쿼리 분류



서브쿼리 분류

서브쿼리에

메인쿼리의 컬럼이 포함되는지에 따라 구분

연관 서브쿼리
(Correlated Subquery)

비연관 서브쿼리
(Un-Correlated Subquery)

연관 서브쿼리

메인쿼리의 컬럼이 서브쿼리에 포함되며,

메인쿼리의 컬럼은 서브쿼리에 특정 조건으로 사용된다

연관 서브쿼리

본인이 속한 부서의 평균 급여보다 높은 급여를 받는 직원들을 출력한다

코드

```
SELECT
    ID, DEPARTMENT_ID, NAME, SALARY
FROM
    EMPLOYEE A
WHERE
 SALARY > (SELECT
    AVG(SALARY)
  FROM
    EMPLOYEE B
  WHERE
    B.DEPARTMENT_ID = A.DEPARTMENT_ID);
```

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT_ID	AVG(SALARY)
1	6900
2	6700
3	7500

↓

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000

비연관 서브쿼리

메인쿼리 컬럼이 서브쿼리에 포함되지 않으며,

주로 메인 쿼리에 특정한 값을 제공할 때 사용된다

비연관 서브쿼리

ELICE가 속한 부서의 평균 급여를 출력한다

코드

```
SELECT
    AVG(SALARY)
FROM
    EMPLOYEE
WHERE
    DEPARTMENT_ID = (SELECT
                        DEPARTMENT_ID
                     FROM
                        EMPLOYEE
                     WHERE
                        NAME = 'ELICE');
```

EMPLOYEE

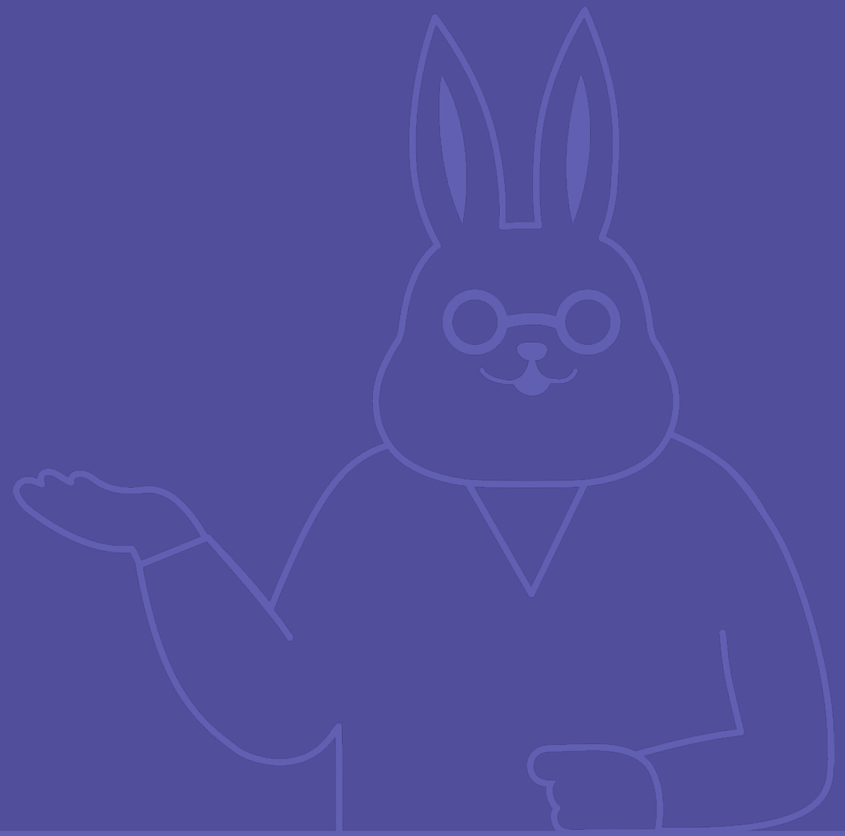
ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT_ID
1

AVG_SALARY
6900

02

반환되는 데이터 형태에 따른 서브쿼리 분류



서브쿼리 분류

단일 행 서브쿼리
(Single Row Subquery)

다중 행 서브쿼리
(Multi Row Subquery)

다중 컬럼 서브쿼리
(Multi Column Subquery)

단일 행 서브쿼리

서브쿼리의 결과가 **한 개의 행**을 반환하며,

단일 행 비교 연산자(=, <, >, <=, >=)와 같이 사용된다

단일 행 서브쿼리

ELICE가 속한 부서의 직원들을 출력한다

코드

```
SELECT
    ID, NAME, SALARY
FROM
    EMPLOYEE
WHERE
    DEPARTMENT_ID = (SELECT
                        DEPARTMENT_ID
                     FROM
                        EMPLOYEE
                     WHERE
                        NAME = 'ELICE');
```

DEPARTMENT_ID
1

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1003	1	LINDA	5500
1005	1	ROBERT	3200

다중 행 서브쿼리

서브쿼리의 결과가 **두 개 이상 행**을 반환할 수 있으며,

다중 행 비교 연산자(IN, ALL, ANY, EXISTS)와 같이 사용된다

다중 행 서브쿼리

특징	설명
IN	서브쿼리 결과에 존재하는 값들 중 하나와 일치해야 한다
EXISTS	서브쿼리 결과 값이 존재하는지 여부를 확인한다
ALL	서브쿼리 결과에 존재하는 모든 값들에 대해 조건을 만족해야 한다
ANY	서브쿼리 결과에 존재하는 값들 중 조건을 만족하는 것이 하나 이상 존재해야 한다

다중 행 서브쿼리 - IN

영업 또는 개발 팀에 속하는 직원들을 출력한다

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT

ID	NAME
1	개발
2	품질
3	영업

코드

```
SELECT
    NAME
FROM
    EMPLOYEE
WHERE
    DEPARTMENT_ID IN (SELECT
                        ID
                      FROM
                        DEPARTMENT
                      WHERE
                        NAME = '품질' OR NAME = '영업');
```

DEPARTMENT_ID
2
3

NAME
JESSICA
STEVE
DAVID

다중 행 서브쿼리 - EXISTS

급여가 10000을 넘는 직원이 존재하는 부서에
소속된 모든 직원들을 출력한다

코드

```
SELECT
    NAME
FROM
    EMPLOYEE A
WHERE
    EXISTS (SELECT
              ID
            FROM
              EMPLOYEE B
            WHERE
              B.SALARY >= 10000 AND
              A.DEPARTMENT_ID = B.DEPARTMENT_ID);
```

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

NAME
ELICE
LINDA
ROBERT

다중 행 서브쿼리 - ALL

개발 팀 소속 모든 직원들 급여보다
급여가 큰 직원들을 출력한다

코드

```
SELECT
    NAME
FROM
    EMPLOYEE
WHERE
    SALARY >= ALL (SELECT
                     SALARY
                   FROM
                     EMPLOYEE
                   WHERE
                     DEPARTMENT_ID = 1);
```

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT

ID	NAME
1	개발
2	품질
3	영업

SALARY
12000
5500
3200

NAME
ELICE

다중 행 서브쿼리 - ANY

개발 팀 소속 임의의 직원들 급여보다 급여가 큰 직원들을 출력한다

코드

```
SELECT
    NAME
FROM
    EMPLOYEE
WHERE
    SALARY >= ANY (SELECT
                     SALARY
                   FROM
                     EMPLOYEE
                   WHERE
                     DEPARTMENT_ID = 1);
```

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT

ID	NAME
1	개발
2	품질
3	영업

SALARY
12000
5500
3200

NAME
ELICE
JESSICA
STEVE
LINDA
DAVID
ROBERT

다중 컬럼 서브쿼리

서브쿼리의 결과가 **여러 개의 컬럼을 반환**하며,

메인쿼리의 조건과 동시에 비교된다.

다중 컬럼 서브쿼리

각 부서에서 가장 높은 급여를 받는 직원의 이름과 급여를 출력

코드

```
SELECT
    NAME,
    SALARY
FROM
    EMPLOYEE
WHERE
    (DEPARTMENT_ID, SALARY) IN
    (SELECT
        DEPARTMENT_ID,
        MAX(SALARY)
     FROM
        EMPLOYEE
    GROUP BY
        DEPARTMENT_ID);
```

EMPLOYEE

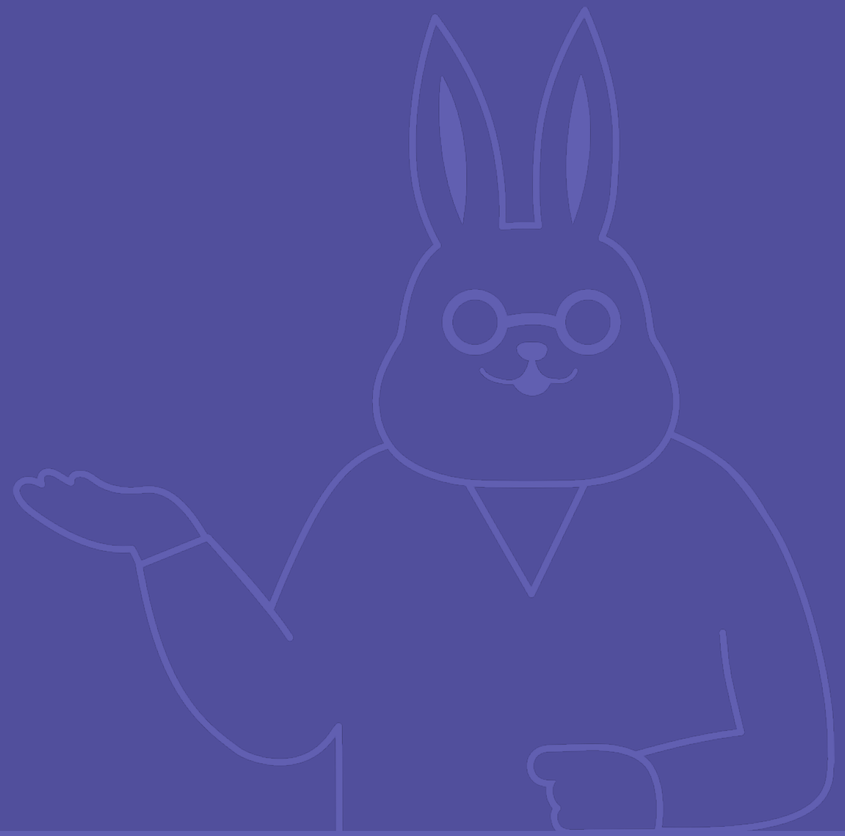
ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT_ID	SALARY
1	12000
2	9000
3	7500

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500

03

스칼라 서브쿼리



스칼라 서브쿼리

스칼라 서브쿼리는

하나의 속성을 가지면서, 하나의 행만을 반환하는 쿼리이다.

그리고 이는 SELECT, WHERE, HAVING 절 등에서 사용할 수 있다.

스칼라 서브쿼리 예시

부서명과 부서의 구성원 수를 출력한다

코드

```
SELECT
    NAME,
    (SELECT
        COUNT(*)
    FROM
        EMPLOYEE E
    WHERE
        E.DEPARTMENT_ID = D.ID)
FROM
    DEPARTMENT D;
```

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT_ID	COUNT(*)
1	3
2	2
3	1

ID	NAME
1	개발
2	품질
3	영업

NAME	COUNT(*)
개발	3
품질	2
영업	1

스칼라 서브쿼리 예시 - DUAL

FROM을 사용한 특정 테이블 참조 없이 쿼리를 작성하는 경우도 있다
전체 임직원 중에 팀장 직급이 차지하는 비율을 출력한다

코드

```
SELECT
    (SELECT COUNT(*) FROM EMPLOYEE WHERE DEPARTMENT_ID = 1) /
    (SELECT COUNT(*) FROM EMPLOYEE)
    AS DEVELOPER_RATIO
FROM
    DUAL;
```

EMPLOYEE

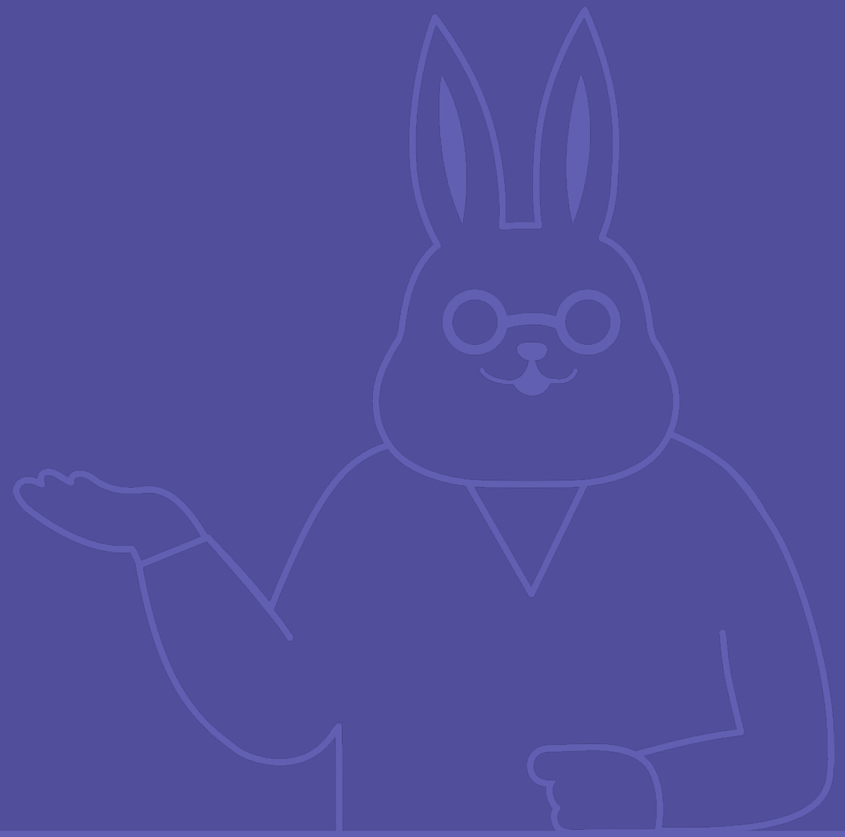
ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

↓

DEVELOPER_RATIO
0.5

04

보안



VIEW

뷰는 다른 테이블에서 파생된 테이블이다.

물리적으로 데이터가 저장되는 것이 아니라,

논리적으로만 존재하며 뷰를 사용한 질의 시에는

DBMS에서 뷰 정의에 따라 질의를 재작성하여 수행한다

VIEW의 장점

특징	설명
독립성	테이블 구조가 변경되어도 뷰를 사용하고 있는 응용 프로그램은 변경하지 않아도 된다
편리성	자주 사용되는 복잡한 쿼리를 미리 뷰로 정의해 놓으면, 추후 쿼리는 간단한 형태로 표현할 수 있다
보안성	사용자의 권한에 따라 열람 가능한 데이터를 다르게할 수 있다 권한에 따라 확인 가능한 컬럼을 정의하여 뷰를 생성하면, 기본 테이블 노출 없이 접근 제어를 할 수 있다

VIEW의 특징

- 생성된 뷰는 또 다른 뷰를 생성하는데 사용될 수 있다
- 뷰의 정의는 변경할 수 없으며, 삭제 후 재생성이 필요하다
- 뷰를 통한 갱신에는 제약이 따른다. 갱신을 위해서는 기본적으로 원천 테이블의 기본키가 포함되어야 한다
- 원천이 되는 테이블이나 뷰가 삭제되면 이를 기반으로 하는 뷰도 함께 삭제된다

VIEW 정의 쿼리

Expression

```
CREATE [OR REPLACE] VIEW view_name AS
SELECT column1, column2, …
FROM table_name
WHERE condition;
```

같은 이름의 뷰가 존재하면 기존 뷰를 무시하고 대체

뷰를 생성할 쿼리 구문

VIEW의 정의 쿼리

EMPLOYEE

ID	DEPARTMENT_ID	NAME	SALARY
1000	1	ELICE	12000
1001	2	JESSICA	9000
1002	3	STEVE	7500
1003	1	LINDA	5500
1004	2	DAVID	4400
1005	1	ROBERT	3200

DEPARTMENT

ID	NAME
1	개발
2	품질
3	영업

코드

```
CREATE VIEW EMPLOYEE_FULL AS
(
  SELECT
    E.ID AS EMPLOYEE_ID,
    E.NAME AS EMPLOYEE_NAME,
    E.SALARY,
    E.DEPARTMENT_ID,
    D.NAME AS DEPARTMENT_NAME
  FROM
    EMPLOYEE E
  LEFT OUTER JOIN
    DEPARTMENT D
  ON
    E.DEPARTMENT_ID = D.ID
);
```

```
SELECT * FROM EMPLOYEE_FULL
```

ID	NAME	SALARY	DEPARTMENT_ID	DEPARTMENT_NAME
1000	ELICE	12000	1	개발
1001	JESSICA	9000	2	품질
1002	STEVE	7500	3	영업
1003	LINDA	5500	1	개발
1004	DAVID	4400	2	품질
1005	ROBERT	3200	1	개발

크레딧

/* elice */

코스 매니저
하주희

콘텐츠 제작자
하주희, 문범우

강사
문범우

감수자
장석준

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io