# Database

(데이터베이스)

김민수

# ★ 서브쿼리 ★

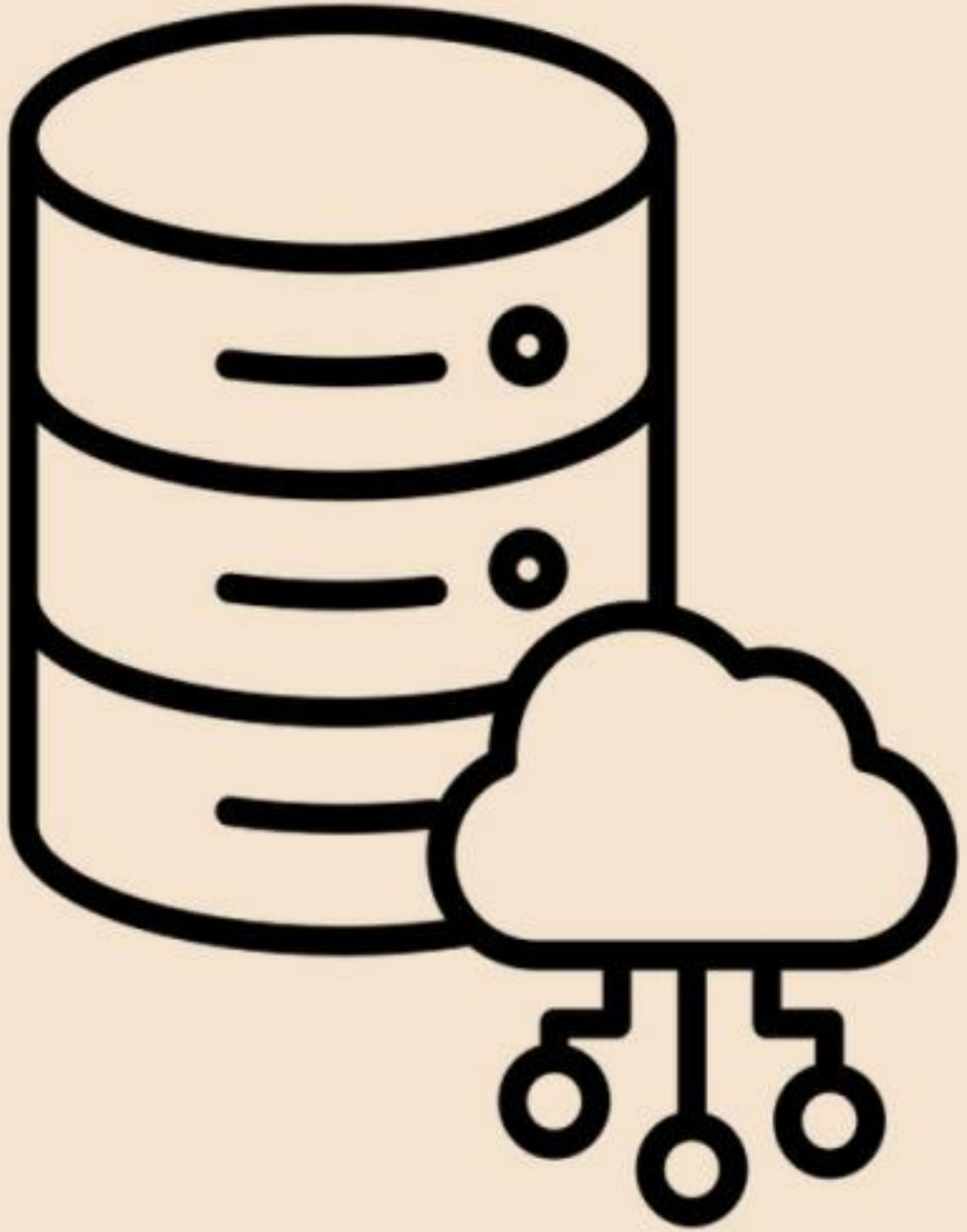
1 서브쿼리란?

2 스칼라 서브쿼리

3 인라인 뷰

4 중첩서브쿼리

5 다중행/단일행 연산자



# 서브쿼리란?

## 서브쿼리(SUB QUERY)

다른 쿼리 내부에 포함되어 있는 SELECT 쿼리를 의미, 보다 다양한 데이터를 출력하는 방법을 제공

→ " 실무에서 굉장히 많이 사용되는 기술 "

우리는 이미 서브쿼리를 사용해봤다!?

```
SELECT *
FROM (
        SELECT *
        FROM 게시판
        ORDER BY 작성일시 DESC)
WHERE ROWNUM <= 20;
```

FROM 절에서 사용되는 서브쿼리

→ " 인라인 뷰 "

# Part 2.
# 스칼라 서브쿼리

# 스칼라 서브쿼리

## 스칼라 서브쿼리 (Scala Subquery)

SELECT 에서 사용되는 서브쿼리로 하나의 컬럼에 대해 하나의 행만 반환하는 특징

출력되는 하나의 값이 없다면 NULL 값 반환

메인 쿼리

```
SELECT A.직원ID, A.연봉, A.부서ID, (
        SELECT 부서명                          서브 쿼리
        FROM 부서
        WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006'
```

| 직원ID | 연봉 | 부서ID | 부서명 |
|---|---|---|---|
| A0001 | 2800 | D001 | 인사부 |
| A0002 | 3000 | D002 | 급여부 |
| A0003 | 2600 | D003 | 전략기획부 |
| A0004 | 4500 | D004 | SI사업부 |
| A0005 | 5000 | (null) | (null) |
| A0006 | 7500 | D001 | 인사부 |

# 스칼라 서브쿼리

## 스칼라 서브쿼리의 실행 원리

**메인쿼리에서 출력되는 튜플의 수만큼 SELECT 에 있는 서브쿼리 반복 실행**

```
SELECT A.직원ID , A.연봉, A.부서ID,
        ( SELECT 부서명
          FROM 부서
          WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

**(1) 직원테이블에서 A0001 ~ A0006 인 조건에 부합하는 6개의 튜플(행)을 출력**

**(2) SELECT 부분에 6개의 튜플(행)이 하나씩 출력되면서 서브 쿼리도 6번 실행**

# 스칼라 서브쿼리

## 스칼라 서브쿼리의 실행 원리

**메인쿼리에서 출력되는 튜플의 수만큼 SELECT 에 있는 서브쿼리 반복 실행**

A0001 → D001

```
SELECT A.직원ID , A.연봉, A.부서ID,
    ( SELECT 부서명
      FROM 부서
      WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

D001 값 넣고 쿼리 실행 !

| 직원ID | 연봉 | 부서ID | 부서명 |
|---|---|---|---|
| A0001 | 2800 | D001 | 인사부 |
| A0002 | 3000 | D002 | 급여부 |
| A0003 | 2600 | D003 | 전략기획부 |
| A0004 | 4500 | D004 | SI사업부 |
| A0005 | 5000 | (null) | (null) |
| A0006 | 7500 | D001 | 인사부 |

# 스칼라 서브쿼리

## 스칼라 서브쿼리의 실행 원리

**메인쿼리에서 출력되는 튜플의 수만큼 SELECT 에 있는 서브쿼리 반복 실행**

A0002 → D002

```
SELECT A.직원ID , A.연봉, A.부서ID,
    ( SELECT 부서명
      FROM 부서
      WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

D002 값 넣고 쿼리 실행 !

| 직원ID | 연봉 | 부서ID | 부서명 |
|---|---|---|---|
| A0001 | 2800 | D001 | 인사부 |
| A0002 | 3000 | D002 | 급여부 |
| A0003 | 2600 | D003 | 전략기획부 |
| A0004 | 4500 | D004 | SI사업부 |
| A0005 | 5000 | (null) | (null) |
| A0006 | 7500 | D001 | 인사부 |

# 스칼라 서브쿼리

## 스칼라 서브쿼리의 실행 원리

**메인쿼리에서 출력되는 튜플의 수만큼 SELECT 에 있는 서브쿼리 반복 실행**

A0003 → D003

```
SELECT A.직원ID , A.연봉, A.부서ID,
    ( SELECT 부서명
      FROM 부서
      WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

D003 값 넣고 쿼리 실행 !

| 직원ID | 연봉 | 부서ID | 부서명 |
|---|---|---|---|
| A0001 | 2800 | D001 | 인사부 |
| A0002 | 3000 | D002 | 급여부 |
| A0003 | 2600 | D003 | 전략기획부 |
| A0004 | 4500 | D004 | SI사업부 |
| A0005 | 5000 | (null) | (null) |
| A0006 | 7500 | D001 | 인사부 |

# 스칼라 서브쿼리

## 실습 문제

**직원테이블에서 직원 A0001 부터 A0006 까지의 직원ID , 연봉 , 부서ID 를 출력하고 부서ID에 대한 부서명도 함께 출력하시오.**

**(스칼라 서브쿼리 방식 말고 직원 , 부서 테이블을 조인방식으로 해결해보자)**

**직원 테이블 내용**

| 직원ID | 연봉 | 부서ID |
|---|---|---|
| A0001 | 2800 | D001 |
| A0002 | 3000 | D002 |
| A0003 | 2600 | D003 |
| A0004 | 4500 | D004 |
| A0005 | 5000 | (null) |
| A0006 | 7500 | D001 |

**부서 테이블 내용**

| 부서ID | 부서명 |
|---|---|
| D001 | 인사부 |
| D002 | 급여부 |
| D003 | 전략기획부 |
| D004 | SI사업부 |
| D005 | 사업부 |
| D006 | 인프라서비스부 |

**출력할 결과**

| 직원ID | 연봉 | 부서ID | 부서명 |
|---|---|---|---|
| A0001 | 2800 | D001 | 인사부 |
| A0006 | 7500 | D001 | 인사부 |
| A0002 | 3000 | D002 | 급여부 |
| A0003 | 2600 | D003 | 전략기획부 |
| A0004 | 4500 | D004 | SI사업부 |
| A0005 | 5000 | (null) | (null) |

# 스칼라 서브쿼리

## 실습 문제

**직원테이블에서 직원 A0001 부터 A0006 까지의 직원ID , 연봉 , 부서ID 를 출력하고 부서ID에 대한 부서명도 함께 출력하시오.**

**(스칼라 서브쿼리 방식 말고 직원 , 부서 테이블을 조인방식으로 해결해보자)**

**답**

```
SELECT A.직원ID , A.연봉 , A.부서ID , B.부서명
 FROM 직원 A , 부서 B
WHERE A.부서ID = B.부서ID(+)
 AND A.직원ID BETWEEN 'A0001' AND 'A0006' ;
```

**출력할 결과**

| 직원ID | 연봉 | 부서ID | 부서명 |
|---|---|---|---|
| A0001 | 2800 | D001 | 인사부 |
| A0006 | 7500 | D001 | 인사부 |
| A0002 | 3000 | D002 | 급여부 |
| A0003 | 2600 | D003 | 전략기획부 |
| A0004 | 4500 | D004 | SI사업부 |
| A0005 | 5000 | (null) | (null) |

# 스칼라 서브쿼리

## 스칼라서브쿼리는 아우터 조인으로 변경 가능

```
SELECT A.직원ID , A.연봉 , A.부서ID , B.부서명
  FROM 직원 A , 부서 B
 WHERE A.부서ID = B.부서ID(+)
   AND A.직원ID BETWEEN 'A0001' AND 'A0006' ;
```

**아우터 조인 사용**

```
SELECT A.직원ID , A.연봉, A.부서ID,
       ( SELECT 부서명
         FROM 부서
         WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

**스칼라 서브쿼리 사용**

# 스칼라 서브쿼리

## [스칼라 서브쿼리 사용 시 주의사항]

- 스칼라 서브쿼리로 출력되는 행(튜플)은 1개 혹은 NULL

```
SELECT A.직원ID , A.연봉, A.부서ID,
       ( SELECT 부서명
         FROM 부서
         WHERE 부서ID = 부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

비교하려는 부서ID 컬럼 값에 대해

어디서 유입되는지 구분이 필요

```
ORA-01427: single-row subquery returns more than one row
01427. 00000 -  "single-row subquery returns more than one row"
*Cause:
*Action:
```

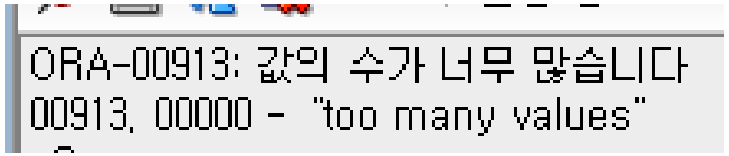
# 스칼라 서브쿼리

## [스칼라 서브쿼리 사용 시 주의사항]

- 스칼라 서브쿼리로 출력되는 **컬럼**은 반드시 **1개**

```
SELECT A.직원ID , A.연봉, A.부서ID,
    ( SELECT 부서ID, 부서명
      FROM 부서
      WHERE 부서ID = 부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

**오직 하나의 컬럼만 입력해야 실행**



# 스칼라 서브쿼리

## 실습 문제

**문제1) 직원테이블에서 직원 A0001 부터 A0006 까지의 직원ID , 연봉 , 부서ID 를 출력하고 부서ID에 대한 부서명도 함께 출력되도록 해주세요. (스칼라서브쿼리 사용)**

**문제2) 직원테이블에서 직원 A0006 부터 A0010 까지의 직원ID , 이름 , 주민등록번호 를 출력하고 휴대폰번호도 함께 출력되도록 해주세요. (없으면 NULL이 출력되도록 스칼라서브쿼리 활용)**

| 직원ID | 이름 | 주민등록번호 | 휴대폰번호 |
|---|---|---|---|
| A0006 | 송대주 | 790903-1566127 | 010-8373-5511 |
| A0007 | 메이슨 | 830629-1676551 | 010-2323-1133 |
| A0008 | 송진아 | 761212-2508143 | 010-8877-0087 |
| A0009 | 이서연 | 730317-259616 | (null) |
| A0010 | 김홍민 | 710513-1572876 | (null) |

# 스칼라 서브쿼리

**문제1) 직원테이블에서 직원 A0001 부터 A0006 까지의 직원ID , 연봉 , 부서ID 를 출력하고 부서ID에 대한 부서명도 함께 출력되도록 해주세요. (스칼라서브쿼리 사용)**

**정답)**

```
SELECT A.직원ID, A.연봉, A.부서ID,
        ( SELECT 부서명
          FROM 부서
          WHERE 부서ID = A.부서ID ) AS 부서명
FROM 직원 A
WHERE 직원ID BETWEEN 'A0001' AND 'A0006' ;
```

# 스칼라 서브쿼리

**문제2) 직원테이블에서 직원 A0006 부터 A0010 까지의 직원ID , 이름 , 주민등록번호 를 출력하고 휴대폰번호도 함께 출력되도록 해주세요. (없으면 NULL이 출력되도록 스칼라서브쿼리 활용)**

**정답)**

```
SELECT A.직원ID, A.이름, A.주민등록번호,
        ( SELECT 연락처
          FROM 직원연락처
          WHERE 직원ID = A.직원ID ) AS 휴대폰번호
FROM 직원 A
WHERE 직원ID BETWEEN 'A0006' AND 'A0010' ;
```

# 인라인 뷰

## 인라인 뷰(Inline View)

**FROM 에서 사용되는 서브쿼리로 마치 가상의 테이블처럼 이용 가능한 서브쿼리**

**→ 따로 떼어내서 실행해도 결과가 출력되는 독립적인 쿼리**

```
SELECT *
FROM (
        SELECT *
        FROM 게시판
        ORDER BY 작성일시 DESC)
WHERE ROWNUM <= 20;
```

**인라인 뷰를 이용해 가상의 테이블처럼 결과를 생성**

| | 게시판번호 | 작성자 | 게시물내용 |
|---|---|---|---|
| 1 | 1000000 | 아이디0 | 아이디0님이 작성하신 게시물입니다. 이 게시물은 |
| 2 | 999999 | 아이디9999 | 아이디9999님이 작성하신 게시물입니다. 이 게시 |
| 3 | 999998 | 아이디9998 | 아이디9998님이 작성하신 게시물입니다. 이 게시 |
| 4 | 999997 | 아이디9997 | 아이디9997님이 작성하신 게시물입니다. 이 게시 |
| 5 | 999996 | 아이디9996 | 아이디9996님이 작성하신 게시물입니다. 이 게시 |
| 6 | 999995 | 아이디9995 | 아이디9995님이 작성하신 게시물입니다. 이 게시 |
| 7 | 999994 | 아이디9994 | 아이디9994님이 작성하신 게시물입니다. 이 게시 |
| 8 | 999993 | 아이디9993 | 아이디9993님이 작성하신 게시물입니다. 이 게시 |
| 9 | 999992 | 아이디9992 | 아이디9992님이 작성하신 게시물입니다. 이 게시 |
| 10 | 999991 | 아이디9991 | 아이디9991님이 작성하신 게시물입니다. 이 게시 |
| 11 | 999990 | 아이디9990 | 아이디9990님이 작성하신 게시물입니다. 이 게시 |
| 12 | 999989 | 아이디9989 | 아이디9989님이 작성하신 게시물입니다. 이 게시 |
| 13 | 999988 | 아이디9988 | 아이디9988님이 작성하신 게시물입니다. 이 게시 |

# 인라인 뷰

## 실습 문제

**문제 ) 나이가 어린 직원 3명의 모든 정보를 출력해주세요.**

| 직원ID | 패스워드 | 이름 | 성별 | 나이 | 입사일시 | 주민등록번호 | 연봉 | 부서ID |
|---|---|---|---|---|---|---|---|---|
| A0001 | 12345 | 김철수 | 남 | 25 | 22/03/21 | 991212-1566123 | 2800 | D001 |
| A0002 | hello123! | 강홍수 | 남 | 28 | 21/09/12 | 950223-1562867 | 3000 | D002 |
| A0005 | test123 | 문현철 | 남 | 34 | (null) | 891231-1786155 | 5000 | (null) |

# 인라인 뷰

## 실습 문제

**문제 ) 나이가 어린 직원 3명의 모든 정보를 출력해주세요.**

```
SELECT *
FROM ( SELECT *
       FROM 직원
       ORDER BY 나이
       )
WHERE ROWNUM <= 3 ;
```

# 중첩서브쿼리

## 중첩서브쿼리(Nested Sub query)

**WHERE 에서 주로 사용되는 서브쿼리로 메인쿼리와 관계가 있는지에 따라 상관 서브쿼리와 비상관 서브쿼리로 나눔**

```
SELECT *
  FROM 직원
 WHERE 연봉 >= (SELECT AVG(연봉)
                  FROM 직원 ) ;
```

중첩서브쿼리

# 중첩서브쿼리

**비상관 서브쿼리 : 메인 쿼리의 컬럼을 사용하지 않는 서브쿼리**

**→ 서브쿼리가 먼저 실행되고 메인 쿼리 실행**

**상관 서브쿼리 : 메인 쿼리의 컬럼을 사용하는 서브쿼리**

**→ 메인쿼리가 먼저 실행되고 서브 쿼리 실행**

# 중첩서브쿼리

## 비상관 서브쿼리 원리

독단적인 쿼리이므로
한 번의 실행을 통해
정답 도출

```
SELECT *
FROM 직원
WHERE 연봉 >= (SELECT AVG(연봉)
               FROM 직원 ) ;
```

**전체 직원의 평균 연봉보다 높은 연봉을 받는 직원의 모든 정보 출력**

# 중첩서브쿼리

## 상관 서브쿼리 원리

메인 쿼리의 컬럼과 연관되어 있어 값이 정해지지 않음

```
SELECT *
FROM 직원 A
WHERE 연봉 = ( SELECT MIN(연봉)
               FROM 직원
               WHERE 부서ID = A.부서ID ) ;
```

**각 부서에서 연봉을 가장 적게 받는 사람들만 출력**

# 중첩서브쿼리

## 실습 문제

**문제1 ) 부서별로 가장 높은 연봉을 가진 직원들의 모든 정보를 출력해주세요.**

**(이번에는 인라인 뷰가 아니라 WHERE 절에 상관서브쿼리를 활용합니다)**

| 직원ID | 패스워드 | 이름 | 성별 | 나이 | 입사일시 | 주민등록번호 | 연봉 | 부서ID |
|---|---|---|---|---|---|---|---|---|
| A0006 | 774433 | 송대주 | 남 | 44 | 15/07/16 | 790903-1566127 | 7500 | D001 |
| A0007 | pwd123 | 메이슨 | 남 | 40 | 16/08/19 | 830629-1676551 | 6200 | D002 |
| A0008 | anjffhgkwl123 | 송진아 | 여 | 47 | 15/07/16 | 761212-2508143 | 7500 | D003 |
| A0009 | test123 | 이서연 | 여 | 50 | 13/11/23 | 730317-259616 | 9000 | D004 |
| A0010 | coffeegood! | 김홍민 | 남 | 52 | 13/11/23 | 710513-1572876 | 9300 | D005 |

# 중첩서브쿼리

## 실습 문제

**문제1 ) 부서별로 가장 높은 연봉을 가진 직원들의 모든 정보를 출력해주세요.**

**(이번에는 인라인 뷰가 아니라 WHERE 절에 상관서브쿼리를 활용합니다)**

**답 )**

```
SELECT *
FROM 직원 A
WHERE 연봉 = ( SELECT MAX(연봉)
               FROM 직원
               WHERE 부서ID = A.부서ID ) ;
```

# 중첩서브쿼리

## 실습 문제

**문제2 ) 입사를 가장 늦게 한 직원의 정보를 모두 출력해주세요.**

| 직원ID | 패스워드 | 이름 | 성별 | 나이 | 입사일시 | 주민등록번호 | 연봉 | 부서ID |
|---|---|---|---|---|---|---|---|---|
| A0003 | nono132 | 이현정 | 여 | (null) | 22/11/06 | 000112-4566123 | 2600 | D003 |

# 중첩서브쿼리

## 실습 문제

**문제2 ) 입사를 가장 늦게 한 직원의 정보를 모두 출력해주세요.**

| 직원ID | 패스워드 | 이름 | 성별 | 나이 | 입사일시 | 주민등록번호 | 연봉 | 부서ID |
|---|---|---|---|---|---|---|---|---|
| A0003 | nono132 | 이현정 | 여 | (null) | 22/11/06 | 000112-4566123 | 2600 | D003 |

답 ) SELECT *
FROM 직원
WHERE 입사일시 = (SELECT MAX(입사일시)
FROM 직원 ) ;

# 중첩서브쿼리

## 실습 문제

**문제3 ) 가장 고연봉인 직원의 정보를 모두 출력해주세요.**

| 직원ID | 패스워드 | 이름 | 성별 | 나이 | 입사일시 | 주민등록번호 | 연봉 | 부서ID |
|---|---|---|---|---|---|---|---|---|
| A0010 | coffeegood! | 김홍민 | 남 | 52 | 13/11/23 | 710513-1572876 | 9300 | D005 |

# 중첩서브쿼리

## 실습 문제

**문제3 ) 가장 고연봉인 직원의 정보를 모두 출력해주세요.**

| 직원ID | 패스워드 | 이름 | 성별 | 나이 | 입사일시 | 주민등록번호 | 연봉 | 부서ID |
|---|---|---|---|---|---|---|---|---|
| A0010 | coffeegood! | 김홍민 | 남 | 52 | 13/11/23 | 710513-1572876 | 9300 | D005 |

```
SELECT *
FROM 직원
WHERE 연봉 = (SELECT MAX(연봉)
              FROM 직원 ) ;
```

# 단일행/다중행 연산자

**WHERE 절에서 서브쿼리 결과를 받을 때**

**1. 단일행을 받을 수 있는 연산자 ( =, =< , !=, > 등등..)**

**2. 다중행을 받을 수 있는 연산자 ( IN, ANY, ALL, EXIST 등등..)**

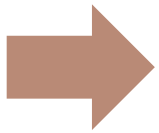
```
SELECT *
FROM 직원 A
WHERE 연봉 = ( SELECT 연봉
               FROM 직원
               WHERE 부서ID = 'D001'
             ) ;
```

```
SELECT *
FROM 직원 A
WHERE 연봉 IN ( SELECT 연봉
                FROM 직원
                WHERE 부서ID = 'D001'
              ) ;
```

**둘 중에서 오류가 나는 쿼리는 무엇일까?**

# 단일행/다중행 연산자

```
SELECT *
FROM 직원 A
WHERE 연봉 IN ( SELECT 연봉
                FROM 직원
                 WHERE 부서ID = 'D001'
) ;
```



| | 연봉 |
|---|---|
| 1 | 2800 |
| 2 | 7500 |

**서브쿼리 실행 결과가 2개 이상의 행을 출력했으므로 IN이 정답!**

**→ 다중행(2개 이상의 행) 연산!**

# 단일행/다중행 연산자

**단일행을 받을 수 있는 연산자 : = , >= , < . != 등**

**→ 비교 연산자는 이미 많이 경험해 봤으므로**

**다중행을 받을 수 있는 연산자 : IN , ANY , ALL , EXISTS , NOT EXISTS 등**

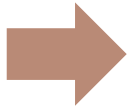
**→ 우리가 살펴봐야 할 것은 다중 연산자**

# 단일행/다중행 연산자

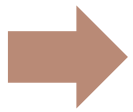
## 다중행 연산자 < IN >

**입력된 다중 행 중에서 일치하는 값들은 모두 출력**

```
SELECT *
 FROM 직원
WHERE 연봉 IN (
SELECT MAX(연봉)
FROM 직원
GROUP BY 부서ID
);
```



| MAX(연봉) |
|---|
| 7500 |
| 5000 |
| 9000 |
| 7500 |
| 6200 |



```
SELECT *
 FROM 직원
WHERE 연봉 IN ( 7500,
                5000,
                9000,
                7500,
                6200  );
```

# 단일행/다중행 연산자

## 다중행 연산자 < ANY >

**입력된 다중 행 중에서 하나라도 일치하면 출력**

```
SELECT 직원ID, 연봉
  FROM 직원
 WHERE 연봉 >= ANY ( 5000, 7500, 2800);
```

→ **연봉이 2800 이상인 모든 직원 출력**

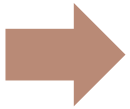
```
SELECT 직원ID, 연봉
  FROM 직원
 WHERE 연봉 <= ANY ( 5000, 7500, 2800);
```

→ **연봉이 7500 이하인 모든 직원 출력**

# 단일행/다중행 연산자

## ANY의 원리

```
SELECT 직원ID, 연봉
  FROM 직원
 WHERE 연봉 >= ANY ( 5000, 7500, 2800);
```



**연봉 >= 5000 OR 연봉 >= 7500 OR 연봉 >= 2800**

**동등 조건(=)에서는 IN과 같은 역할**

**비동등 조건 ( > , < , >= , <= ) 관련 문제가 많이 출제**

# 단일행/다중행 연산자

## 다중행 연산자 < ALL >

**입력된 다중 행 중에서 값이 모두 일치해야 출력**

```
SELECT 직원ID, 연봉
  FROM 직원
 WHERE 연봉 >= ALL ( 5000, 7500, 2800);
```

→ **연봉이 7500 이상인 모든 직원 출력**

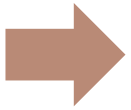
```
SELECT 직원ID, 연봉
  FROM 직원
 WHERE 연봉 <= ALL ( 5000, 7500, 2800);
```

→ **연봉이 2800 이하인 모든 직원 출력**

# 단일행/다중행 연산자

## ALL의 원리

```
SELECT 직원ID, 연봉
  FROM 직원
 WHERE 연봉 >= ANY ( 5000, 7500, 2800);
```



**연봉 >= 5000 AND 연봉 >= 7500 AND 연봉 >= 2800**

**ANY와 ALL의 차이점, 헷갈리기 때문에 시험에 자주 출제!**

# 단일행/다중행 연산자

## 실습 문제

직원 테이블에서 직원ID가 'A0006'인 사람과 같은 연봉을 받는 직원들의 이름과 연봉을 출력하세요. (단, 중첩서브쿼리와 IN을 사용)

| | 이름 | 연봉 |
|---|---|---|
| 1 | 송대주 | 7500 |
| 2 | 송진아 | 7500 |

# 단일행/다중행 연산자

## 실습 풀이

직원 테이블에서 직원ID가 'A0006'인 사람과 같은 연봉을 받는 직원들의 이름과 연봉을 출력하세요. (단, 중첩서브쿼리와 IN을 사용)

```
SELECT 이름, 연봉
FROM 직원
WHERE 연봉 IN ( SELECT 연봉
                FROM 직원
                WHERE 직원ID = 'A0006' )
```

| | 이름 | 연봉 |
|---|---|---|
| 1 | 송대주 | 7500 |
| 2 | 송진아 | 7500 |

# 단일행/다중행 연산자

## 실습 문제

다음 SQL문을 보고 실행 결과가 오른쪽과 같이 출력되려면 어떤 연산자를 사용해야 하는지 쓰세요. **(서브쿼리 부분만 실행시켜 놓고 문제풀이)**

```
SELECT 직원ID, 이름, 연봉
FROM 직원
WHERE 연봉 >  (     ) ( SELECT 연봉
                        FROM 직원
                        WHERE 직원ID BETWEEN
                              'A0003' AND 'A0006' ) ;
```

| | 직원ID | 이름 | 연봉 |
|---|---|---|---|
| 1 | A0009 | 이서연 | 9000 |
| 2 | A0010 | 김홍민 | 9300 |

# 단일행/다중행 연산자

## 실습 풀이

다음 SQL문을 보고 실행 결과가 오른쪽과 같이 출력되려면 어떤 연산자를 사용해야 하는지 쓰세요. **(서브쿼리 부분만 실행시켜 놓고 문제풀이)**

```
SELECT 직원ID, 이름, 연봉
FROM 직원
WHERE 연봉 > ALL ( SELECT 연봉
                   FROM 직원
                   WHERE 직원ID BETWEEN
                         'A0003' AND 'A0006' ) ;
```

| | 직원ID | 이름 | 연봉 |
|---|---|---|---|
| 1 | A0009 | 이서연 | 9000 |
| 2 | A0010 | 김홍민 | 9300 |

# 단일행/다중행 연산자

## 다중행 연산자 < EXISTS >

**입력된 다중 행 중에서 일치하는 행의 존재 여부를 확인 → 존재하면 TRUE 반환**

**서브쿼리를 꼭 사용해야 하는 연산자**

**예) 연락처가 존재하는 직원의 직원ID와 이름을 출력하세요**

```
SELECT 직원ID, 이름
FROM 직원 A
WHERE EXISTS ( SELECT 1
               FROM 직원연락처
               WHERE 직원ID = A.직원ID) ;
```

1은 참 OR 거짓에서
'참'일 경우를 의미
기능 X

# 단일행/다중행 연산자

## EXISTS 문법

```
SELECT 직원ID, 이름
FROM 직원 A
WHERE EXISTS ( SELECT 1
               FROM 직원연락처
               WHERE 직원ID = A.직원ID) ;
```

1. **WHERE 뒤에 특정 컬럼을 입력하지 않고 바로 EXISTS 를 입력하여 연산 ( =? )**
2. **조건에 일치하는 대상을 찾는 순간, 작업을 멈추고 다음 작업 수행 ( 효과적 )**
3. **상관서브쿼리와 비슷하게 메인쿼리의 컬럼을 참조**

# 단일행/다중행 연산자

## EXIST 와 IN의 차이점

IN은 일치하는 모든 것을 출력하기 위해 테이블의 모든 튜플에 접근

→ 출력한 내용을 눈으로 확인해야 할 때 사용

EXISTS는 일치하는 튜플이 있는 순간 해당 작업 중지

→ 해당 내용이 존재 하는지 확인되기만 하면 테이블의 다른 정보를 보여주고 싶을 때 ( 속도 및 성능 우위 )

# 단일행/다중행 연산자

## 다중행 연산자 < NOT EXISTS >

**입력된 다중 행 중에서 일치하는 행의 존재 여부를 확인→ 존재하면 FALSE 반환**

**조건과 일치하지 않는 내용에 대해 TRUE 출력**

**예) 연락처가 존재하지 않는 직원의 직원ID와 이름을 출력하세요**

```
SELECT 직원ID, 이름
FROM 직원 A
WHERE NOT EXISTS ( SELECT 1
                   FROM 직원연락처
                   WHERE 직원ID = A.직원ID) ;
```