

초보자를 위한  
**Oracle SQL Database**



서브쿼리



한국기술교육대학교  
온라인평생교육원

## 학습내용

- 서브쿼리

## 학습목표

- 서브쿼리에 대하여 이해하고 활용할 수 있다.

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 서브쿼리란?

#### 서브쿼리란?

- 하나의 SQL 문 안에 포함되어 있는 또 다른 SQL 문
- SELECT 절, FROM 절, WHERE 절 또는 DML 문장(SELECT, INSERT, UPDATE, DELETE 등)에 들어 있는 SELECT 문장

메인쿼리(Main Query)

```
SELECT employee_id, first_name, salary
FROM employees
```

서브쿼리(Sub Query)

```
WHERE salary > (SELECT
ROUND(AVG(salary)) FROM employees);
```

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 서브쿼리의 필요성

사원 테이블에서 전체 사원에 대한 평균 급여보다  
높은 급여를 받는 사원의 명단 출력하려면 어떻게 해야 할까?

```
SELECT ROUND(AVG(salary)) FROM employees;
```

	ROUND(AVG(SALARY))
1	6462

먼저 전체 사원에 대한 평균 급여를 조회해야 함

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 서브쿼리의 필요성

```
워크시트  질의 작성기
SELECT employee_id, first_name, salary FROM employees
WHERE salary > 6462;
```

스크립트 출력 x 질의 결과 x
SQL 50개의 행이 인출됨(0.005초)
EMPLOYEE\_ID FIRST\_NAME SALARY
1 100 Steven 24000
2 101 Neena 17000
3 102 Lex 17000
4 103 Alexander 9000
5 108 Nancy 12000
6 109 Daniel 9000
7 110 John 8200
8 111 Ismael 7700
9 112 Jose Manuel 7800
10 113 Luis 6900
11 114 Den 11000
12 120 Matthew 8000
13 121 Adam 8200
14 122 Payam 7900
15 123 Shanta 6500
16 145 John 14000
17 146 Karen 13500

**SELECT employee\_id, first\_name, salary FROM employees  
WHERE salary > 6462;**

그 다음으로 평균 급여인 6462보다 큰 급여를 받는 전체 사원을 조회

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 서브쿼리의 필요성

```
워크시트  질의 작성기
SELECT employee_id, first_name, salary FROM employees
WHERE salary > ( SELECT ROUND(AVG(salary)) FROM employees );
```

스크립트 출력 x 질의 결과 x
SQL 50개의 행이 인출됨(0.003초)
EMPLOYEE\_ID FIRST\_NAME SALARY
1 100 Steven 24000
2 101 Neena 17000
3 102 Lex 17000
4 103 Alexander 9000
5 108 Nancy 12000
6 109 Daniel 9000
7 110 John 8200
8 111 Ismael 7700
9 112 Jose Manuel 7800
10 113 Luis 6900
11 114 Den 11000
12 120 Matthew 8000
13 121 Adam 8200
14 122 Payam 7900
15 123 Shanta 6500
16 145 John 14000
17 146 Karen 13500

**SELECT employee\_id, first\_name, salary FROM employees  
WHERE salary > ( SELECT ROUND(AVG(salary)) FROM employees );**

WHERE 절에서 중첩 서브쿼리를 사용하면 쉽게 구할 수 있음

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### ✚ 서브쿼리의 종류

#### 반환하는 값 개수에 따른 분류

- 단일행(Single Row) 서브쿼리  
: 실행 결과가 하나의 행이 나오는 서브쿼리
- 복수행(Multiple Row) 서브쿼리  
: 실행 결과가 여러 개의 행이 나오는 서브쿼리
- 복수열(Multiple Column) 서브쿼리  
: 실행 결과가 여러 컬럼으로 나오는 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### ✚ 서브쿼리의 종류

#### 실행 방식에 따른 분류

- 일반 서브쿼리 : 단일행, 복수행, 복수열 서브쿼리
- 상관(연관) 관계 서브쿼리  
: 메인쿼리의 컬럼이나 값을 사용하는 서브쿼리
- 스칼라 서브쿼리  
: 하나의 컬럼처럼 사용되는 서브쿼리
- 인라인 뷰 서브쿼리  
: 뷰 형태로 테이블을 리턴하는 서브쿼리로,  
FROM 절 다음에 테이블명 대신 SELECT 절이 오는  
서브쿼리
- 중첩 서브쿼리 : WHERE 절 다음에 오는 서브쿼리

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 단일행 서브쿼리

#### 단일행 서브쿼리

- 실행 결과가 하나의 행이 나오는 서브쿼리
- 메인쿼리와 비교할 때 단일행 연산자를 이용

단일행 연산자	설명
>	큽, 초과
>=	크거나 같음, 이상
=	같음
<=	작거나 같음, 이하
<	작음, 미만
!=, <>, ^=	같지 않음

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 단일행 서브쿼리

#### 단일행 서브쿼리

```
SELECT employee_id, first_name, salary
FROM employees
WHERE salary != (SELECT
ROUND(AVG(salary)) FROM employees);
```

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 복수행 서브쿼리

#### 복수행 서브쿼리

- 실행 결과가 여러 개의 행이 나오는 서브쿼리
- 메인쿼리와 비교할 때 복수행 연산자를 이용

복수행 연산자	설명
IN	결과 중 하나라도 일치하면 참
ANY, SOME	결과와 하나 이상 일치하면 참
ALL	결과와 모든 값이 일치하면 참
EXISTS	결과 중에서 값이 있으면 참 메인쿼리 수행, 없으면 거짓으로 메인쿼리 수행 안함

서브쿼리

서브쿼리

## ▣ 서브쿼리(Sub Query)

### + 복수행 서브쿼리

#### 복수행 서브쿼리

```
SELECT employee_id, first_name, salary
FROM employees
```

```
WHERE salary IN ( SELECT MAX(salary)
                  FROM employees GROUP BY
                  department_id );
```

# 서브쿼리

## 서브쿼리

### 서브쿼리

#### ▣ 서브쿼리(Sub Query)

##### ✚ 복수열 서브쿼리

EMPLOYEE_ID	FIRST_NAME	SALARY
1	Steven	24000
2	Alexander	9000
3	Nancy	12000
4	Den	11000
5	Adam	8200
6	John	14000
7	Jennifer	4400
8	Michael	13000
9	Susan	6500
10	Hermann	10000
11	Shelley	12000

SELECT employee\_id, first\_name, salary FROM employees  
 WHERE (department\_id, salary) IN (SELECT department\_id, MAX(salary)  
 FROM employees GROUP BY department\_id);

복수열 서브쿼리 : 실행 결과가 여러 컬럼으로 나오는 서브쿼리

## 서브쿼리

### 서브쿼리

#### ▣ 서브쿼리(Sub Query)

##### ✚ 서브쿼리 사용 주의사항

#### 서브쿼리 사용 주의사항

- 서브쿼리는 괄호로 둘러싸서 사용
- 단일행(Single Row) 비교 연산자는 서브쿼리의 결과가 반드시 1건 이하
- 복수행(Multiple Row) 비교 연산자는 서브쿼리의 결과 건수와 상관 없음
- 서브쿼리에서는 ORDER BY를 사용하지 못함
- ORDER BY 절은 SELECT 절에서 한 개만 올 수 있기 때문에 메인쿼리에서 사용

# 서브쿼리

서브쿼리

서브쿼리

## ◆ 서브쿼리(Sub Query)

### + 서브쿼리가 올 수 있는 SQL 문 위치

#### 서브쿼리가 올 수 있는 SQL 문 위치

- SELECT 절
- FROM 절
- WHERE 절
- HAVING 절
- ORDER BY 절
- INSERT 문의 VALUES 절
- UPDATE 문의 SET 절

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 단일행 서브쿼리

#### 단일행 서브쿼리

- 실행 결과가 하나의 행이 나오는 서브쿼리
- 메인쿼리와 비교할 때 단일행 연산자를 이용

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리 예

### + 단일행 서브쿼리

The screenshot shows the Oracle SQL Developer interface. In the top-left window, there is a code editor with the following SQL query:

```
SELECT employee_id, first_name, salary FROM employees
WHERE salary > ( SELECT ROUND(AVG(salary)) FROM employees );
```

The WHERE clause contains a subquery highlighted with a red box. Below the code editor is a results grid showing employee data:

EMPLOYEE_ID	FIRST_NAME	SALARY
1	Steven	24000
2	Neena	17000
3	Lex	17000
4	Alexander	9000
5	Nancy	12000
6	Daniel	9000
7	John	8200
8	Ismael	7700
9	Jose Manuel	7800
10	Luis	6900
11	Den	11000
12	Matthew	8000

To the right of the results grid, a callout box highlights the subquery part of the query with the following text:

employees 테이블에서 평균보다 많은 월급을 받은  
사원ID, 이름, 월급을 출력하는 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리 예

### + 복수행 서브쿼리

#### 복수행 서브쿼리

- 실행 결과가 여러 개의 행이 나오는 서브쿼리
- 메인쿼리와 비교할 때 복수행 연산자를 이용

# 서브쿼리

## 서브쿼리

### 서브쿼리

#### 서브쿼리 예

##### + 복수행 서브쿼리

###### IN 연산자

employees			
EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	100 Steven	24000	90
2	103 Alexander	9000	60
3	108 Nancy	12000	100
4	109 Daniel	9000	100
5	110 John	8200	100
6	114 Den	11000	30
7	121 Adam	8200	50
8	123 Shanta	6500	50
9	145 John	14000	80

조회된 결과 25행

부서가 달라도  
조회됨

```
SELECT employee_id,
       first_name, salary,
       department_id FROM employees
 WHERE salary IN ( SELECT MAX(salary) FROM
       employees GROUP BY
       department_id );
```

부서별 최고 월급과 같은 금액을 받는 모든 사원을 조회하는 예  
타 부서라도 부서별 최고 금액과 동일한 월급을 받는 사원은 전부 나옴

## 서브쿼리

### 서브쿼리

#### 서브쿼리 예

##### + 복수행 서브쿼리

###### ANY, SOME 연산자

employees			
EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	100 Steven	24000	90
2	103 Alexander	9000	60
3	108 Nancy	12000	100
4	109 Daniel	9000	100
5	110 John	8200	100
6	114 Den	11000	30
7	121 Adam	8200	50
8	123 Shanta	6500	50
9	145 John	14000	80
10	147 Alberto	12000	80

조회 결과 IN 연산자와  
동일한 25행

```
SELECT employee_id, first_name,
       salary, department_id FROM employees
 WHERE salary = ANY ( SELECT MAX(salary) FROM
       employees GROUP BY
       department_id );
```

일반적으로 = ANY 보다는 IN 연산자가  
명확하기 때문에 IN 연산자를 사용

IN 연산자로 구했던 것과 똑같은 것을 ANY(SOME) 연산자로 구현한 예

# 서브쿼리

## 서브쿼리

### 서브쿼리

#### 서브쿼리 예

##### + 복수행 서브쿼리

### ANY, SOME 연산자

워크시트   질의 작성기			
스크립트 출력 x   질의 결과 x			
SQL   인출된 모든 행: 25(0.007초)			
<code>SELECT employee_id, first_name, salary, department_id FROM employees WHERE salary = ANY ( SELECT MAX(salary) FROM employees GROUP BY department_id )</code>			
EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	100 Steven	24000	90
2	103 Alexander	9000	60
3	108 Nancy	12000	100
4	109 Daniel	9000	100
5	110 John	8200	100
6	114 Den	11000	30
7	121 Adam	8200	50
8	123 Shanta	6500	50
9	145 John	14000	80
10	147 Alberto	12000	80

```
SELECT employee_id, first_name, salary, department_id FROM employees  
WHERE salary = ANY ( SELECT MAX(salary) FROM employees GROUP BY department_id )
```

```
SELECT employee_id, first_name, salary, department_id FROM employees  
WHERE salary >= ANY ( SELECT MAX(salary) FROM employees GROUP BY department_id )
```

여기에서 = ANY를 > ANY나 >= ANY로 변경하면 어떤 결과가 나올까?

## 서브쿼리

### 서브쿼리

#### 서브쿼리 예

##### + 복수행 서브쿼리

### ANY, SOME 연산자

워크시트   질의 작성기			
스크립트 출력 x   질의 결과 x			
SQL   인출된 모든 행: 12(0.002초)			
<code>SELECT MAX(salary) FROM employees</code>			
MAX(SALARY)	조회된 결과 12행		
1	8200		
2	6500		
3	12000		
4	24000		
5	11000		
6	10000		
7	7000		
8	4400		
9	13000		

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

>= ANY 연산자를 만족하는 것은 >= 4400

ANY나 SOME 연산자는 “결과와 하나 이상 일치하면 참”이 되기 때문에 위의 서브쿼리를 먼저 봄

# 서브쿼리

## 서브쿼리

### 서브쿼리

#### ◆ 서브쿼리 예

##### + 복수행 서브쿼리

### ANY, SOME 연산자

워크시트 | 질의 작성기

```
SELECT employee_id, first_name, salary, department_id FROM employees
WHERE salary >= ANY ( SELECT MAX(salary) FROM employees GROUP BY department_id )
```

조회된 결과 50행

스크립트 출력 x | 질의 결과 x | 50개의 행이 인출됨(0.006초)

	EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	100 Steven	24000	90	
2	101 Neena	17000	90	
3	102 Lex	17000	90	
4	103 Alexander	9000	60	
5	104 Bruce	6000	60	
6	105 David	4800	60	
7	106 Valli	4800	60	
8	108 Nancy	12000	100	
9	109 Daniel	9000	100	

```
SELECT employee_id, first_name, salary, department_id FROM employees
WHERE salary >= ANY ( SELECT MAX(salary) FROM employees GROUP BY department_id );
```

따라서 이 서브쿼리의 결과는 다음과 같이 50개의 행이 조회됨

## 서브쿼리

### 서브쿼리

#### ◆ 서브쿼리 예

##### + 복수행 서브쿼리

### ANY, SOME 연산자

```
SELECT employee_id, first_name, salary, department_id FROM employees
WHERE salary >= ANY ( SELECT MAX(salary) FROM employees GROUP BY department_id );
```



```
SELECT employee_id, first_name, salary, department_id FROM employees
WHERE salary >= ( SELECT MIN(salary) FROM employees );
```

이 서브쿼리는 “결과와 하나 이상 일치하면 참”이 되기 때문에  
MIN 함수를 적용한 것과 같은 결과가 됨

# 서브쿼리

서브쿼리

서브쿼리

## 서브쿼리 예

### + 복수행 서브쿼리

#### ALL 연산자

```
SELECT employee_id, first_name, salary,
       department_id FROM employees
 WHERE salary >= ALL ( SELECT MAX(salary) FROM
 employees GROUP BY department_id );
```

이 SQL 문의 서브쿼리의 결과는 4400, 6500, 7000 … 14000, 24000이 됨



```
SELECT employee_id, first_name, salary, department_id FROM employees
 WHERE salary >= ALL ( 4400, 6500, 7000 … 14000, 24000 );
```

ALL은 “결과와 모든 값이 일치하면 참”이 되는 연산자

서브쿼리

서브쿼리

## 서브쿼리 예

### + 복수행 서브쿼리

#### ALL 연산자

워크시트 질의 작성기

```
SELECT employee_id, first_name, salary, department_id FROM employees
 WHERE salary >= ALL ( SELECT MAX(salary) FROM employees GROUP BY department_id );
```

스크립트 출력 x 질의 결과 x  
SQL | 인출된 모든 행: 1(0,008초)

EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	Steven	24000	90

```
SELECT employee_id, first_name, salary,
       department_id FROM employees
 WHERE salary >= ALL ( SELECT MAX(salary) FROM employees GROUP BY
 department_id );
```

```
SELECT employee_id, first_name, salary, department_id FROM employees
 WHERE salary >= ( SELECT MAX(salary) FROM employees );
```

ALL 연산자는 모든 조건을 만족해야 하기 때문에 24000 이상이 되어야 함  
이 서브쿼리는 아래와 같이 MAX 함수를 적용한 것과 같은 결과가 됨

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리 예

### + 복수행 서브쿼리

#### EXISTS 연산자

#### EXISTS 연산자

- 서브쿼리의 결과 값이 1개 이상 있으면 참이 되는 경우
- 참이 되는 경우는 메인쿼리를 실행, 거짓이면 메인쿼리를 실행하지 않음

서브쿼리

서브쿼리

## ▣ 서브쿼리 예

### + 복수행 서브쿼리

#### EXISTS 연산자

```
SELECT employee_id, first_name, hire_date FROM employees
WHERE EXISTS ( SELECT * FROM employees
    WHERE ( hire_date + TO_YMINTERVAL('30-6') < SYSDATE ) );
```

The screenshot shows the Oracle SQL Developer interface. The top window displays the SQL code:

```
SELECT employee_id, first_name, hire_date FROM employees
WHERE EXISTS ( SELECT * FROM employees
    WHERE ( hire_date + TO_YMINTERVAL('30-6') < SYSDATE ) );
```

The bottom window shows the results of the query, listing 15 employees along with their employee ID, first name, and hire date.

EMPLOYEEID	FIRSTNAME	HIRE_DATE
100	Steven	87/06/17
101	Neena	89/09/21
102	Lex	93/01/13
103	Alexander	90/01/03
104	Bruce	91/05/21
105	David	97/06/25
106	Valii	98/02/05
107	Diana	99/02/07
108	Nancy	94/08/17
109	Daniel	94/08/16
110	John	97/09/28
111	Ismael	97/09/30
112	Jose Manuel	98/03/07
113	Luis	99/12/07
114	Den	94/12/07
115	Alexander	95/05/18

근무를 30년 6개월 이상 한 사원이 있는 경우 메인쿼리를 실행하는 SQL 문

# 서브쿼리

서브쿼리

서브쿼리

## ▣ 서브쿼리 예

### + 복수행 서브쿼리

#### EXISTS 연산자

워크시트 질의 작성기

```
SELECT employee_id, first_name, hire_date FROM employees
WHERE EXISTS ( SELECT * FROM employees
                WHERE ( hire_date + TO_YMINTERVAL('35-6') < SYSDATE ) );
```

스크립트 출력 x 질의 결과 x

SQL | 인출된 모든 행: 0(0,003초)

{ EMPLO... { FIRST\_N... { HIRE\_DA...

SELECT employee\_id, first\_name, hire\_date FROM employees  
WHERE EXISTS ( SELECT \* FROM employees  
WHERE ( hire\_date + TO\_YMINTERVAL('35-6')  
< SYSDATE ) );

근무를 35년 6개월 이상 한 사원이 없어 메인쿼리가 실행되지 않음

서브쿼리

서브쿼리

## ▣ 서브쿼리 예

### + 인라인 뷰(inline view) 서브쿼리

#### 인라인 뷰(inline view) 서브쿼리

- 지금까지의 서브쿼리는 WHERE 절에 어떤 조건식으로 사용
- 이번에는 FROM 절에서 사용하는 서브쿼리로 인라인 뷰(inline view)라고도 함
- 인라인 뷰는 특정 테이블에 있는 전체 데이터가 아닌 SELECT 문을 통해 일부 데이터를 먼저 추출할 수 있고 별칭을 지정하여 사용할 수도 있음

# 서브쿼리

## 서브쿼리

### 서브쿼리

#### 서브쿼리 예

##### + 인라인 뷰(inline view) 서브쿼리

워크시트 질의 작성기

```
SELECT 학.학번, 학.성명, 수.교과목코드, 수.취득점수
FROM ( SELECT 학번, 성명 FROM 학생 ) 학
, ( SELECT * FROM 수강 WHERE 교과목코드 = 'CSE244' ) 수
WHERE 학.학번 = 수.학번;
```

스크립트 출력 x 질의 결과 x

SQL | 인출된 모든 행: 3(0.005초)

학번	성명	교과목코드	취득점수
1 2014172008	김성현	CSE244	84
2 2015136057	박재준	CSE244	72
3 2015136065	서지원	CSE244	72

**SELECT 학.학번, 학.성명, 수.교과목코드, 수.취득점수  
FROM ( SELECT 학번, 성명 FROM 학생 ) 학  
, ( SELECT \* FROM 수강 WHERE 교과목코드 = 'CSE244' ) 수  
WHERE 학.학번 = 수.학번;**

위 SQL 문은 교과목코드 CSE244를 수강한 학생의 정보  
(학번, 성명, 교과목코드, 취득점수)를 조회하는 인라인 뷰 서브쿼리

## 서브쿼리

### 서브쿼리

#### 서브쿼리 예

##### + 인라인 뷰(inline view) 서브쿼리

#### WITH 절 사용하기

```
SELECT 학.학번, 학.성명, 수.교과목코드, 수.취득점수
FROM ( SELECT 학번, 성명 FROM 학생 ) 학
, ( SELECT * FROM 수강 WHERE 교과목코드 = 'CSE244' ) 수
WHERE 학.학번 = 수.학번;
```

WITH  
학 AS (SELECT 학번, 성명 FROM 학생)  
, 수 AS (SELECT \* FROM 수강 WHERE 교과목코드 = 'CSE244')  
SELECT 학.학번, 학.성명, 수.교과목코드, 수.취득점수  
FROM 학, 수  
WHERE 학.학번 = 수.학번;

FROM 절에 너무 많은 서브쿼리를 지정하면  
가독성이 떨어지기 때문에 WITH 절을 사용하여 정리

# 서브쿼리

서브쿼리

서브쿼리

## 서브쿼리 예

### + 인라인 뷰(inline view) 서브쿼리

#### WITH 절 사용하기

```
워크시트 | 질의 작성기
SELECT 학.학번, 학.성명, 수.교과목코드, 수.취득점수
FROM ( SELECT 학번, 성명 FROM 학생 ) 학
      , ( SELECT * FROM 수강 WHERE 교과목코드 = 'CSE244' ) 수
WHERE 학.학번 = 수.학번;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 3(0,005초)

학번	성명	교과목코드	취득점수
1 2014172008	김성현	CSE244	84
2 2015136057	박재준	CSE244	72
3 2015136065	서지원	CSE244	72

```
워크시트 | 질의 작성기
WITH
    학 AS (SELECT 학번, 성명 FROM 학생 )
    ,수 AS ( SELECT * FROM 수강 WHERE 교과목코드 = 'CSE244' )
SELECT 학.학번, 학.성명, 수.교과목코드, 수.취득점수
FROM 학, 수
WHERE 학.학번 = 수.학번;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 3(0,001초)

학번	성명	교과목코드	취득점수
1 2014172008	김성현	CSE244	84
2 2015136057	박재준	CSE244	72
3 2015136065	서지원	CSE244	72

FROM 절에 너무 많은 서브쿼리를 지정하면  
가독성이 떨어지기 때문에 WITH 절을 사용하여 정리

서브쿼리

서브쿼리

## 서브쿼리 예

### + 스칼라(scalar) 서브쿼리

```
워크시트 | 질의 작성기
SELECT 수.학번, 학생.성명, 수.교과목코드, 수.취득점수
      , ( SELECT MAX(취득점수) FROM 수강 수1 where 수.교과목코드 = 수1.교과목코드 ) AS 최고점수
FROM 수강 수, 학생 WHERE 수.학번 = 학생.학번;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 39(0,004초)

학번	성명	교과목코드	취득점수	최고점수
1 2014172008	김성현	CSE244	74	94
2 2015136057	박재준	CSE242	94	94
3 2015136065	서지원	CSE242	64	94
4 2014172008	김성현	CSE244	84	84
5 2015136057	박재준	CSE244	72	84
6 2015136065	서지원	CSE244	72	84
7 2014172008	김성현	CSE440	92	98
8 2015136057	박재준	CSE440	98	98
9 2015136065	서지원	CSE440	58	98
10 2018120034	길지간	CSE122	91	95
11 2016174067	이정섭	CSE122	95	95
12 2018120034	길지간	ARB263	94	94
13 2014172008	김성현	CSE341	98	98
14 2015136057	박재준	CSE341	86	98
15 2015136065	서지원	CSE341	46	98

SELECT 수.학번, 학생.성명, 수.교과목코드, 수.취득점수  
, (SELECT MAX(취득점수) FROM 수강 수1 where 수.  
교과목코드 = 수1.교과목코드) AS 최고점수  
FROM 수강 수, 학생 WHERE 수.학번 = 학생.학번;

스칼라 서브쿼리는 SELECT 절에서 하나의 컬럼처럼 사용됨

위 SQL 문은 수강테이블에서 교과목 코드 중 최고 점수를 가져오는 스칼라 서브쿼리

# 서브쿼리

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 기타 서브쿼리

#### INSERT 문의 VALUES 절에서 서브쿼리 사용하기

- 서브쿼리의 결과에 연산을 하여 바로 테이블에 저장을 할 때 아주 편리하게 사용
- 사용 예시 : 현재 수강 테이블에는 1학기 데이터만 있는데 이를 가공하여 2학기 데이터를 생성하려고 함

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 기타 서브쿼리

#### INSERT 문의 VALUES 절에서 서브쿼리 사용하기

우선 입력 데이터와 테이블의 데이터 형식을 동일하게 맞춰야 함

```
SELECT 학번, 2021, 2, 이수구분,
ROUND(DBMS_RANDOM.VALUE(30, 100)),
재수강여부, 교과목코드, 교수사번 FROM 수강;
```

수강테이블에서 학번, 이수구분, 재수강여부, 교과목코드, 교수사번은  
1학기의 데이터를 그대로 사용

# 서브쿼리

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 기타 서브쿼리

#### INSERT 문의 VALUES 절에서 서브쿼리 사용하기

```
SELECT 학번, 2021, 2, 이수구분,
ROUND(DBMS_RANDOM.VALUE(30, 100)),
재수강여부, 교과목코드, 교수사번 FROM 수강;
```

수강년도, 수강학기는 고정 값으로 지정

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 기타 서브쿼리

#### INSERT 문의 VALUES 절에서 서브쿼리 사용하기

워크시트					
질의 작성기					
<pre>SELECT 학번, 2021, 2, 이수구분, 교과목코드, 교수사번 FROM 수강;</pre>					
스크립트 출력 x   질의 결과 x					
실행 SQL   인출된 모든 행: 39(0,003초)					
번호	학번	2021	2	이수구분	
1	2018120034	2021	2	A	57X ARB263 20091015
2	2018120034	2021	2	A	100X CSE114 20091039
3	2018120034	2021	2	A	63X CSE120 20092019
4	2018120034	2021	2	A	32X CSE121 20093032
5	2018120034	2021	2	A	60X CSE122 20120019
6	2018120034	2021	2	A	31X CSE130 20120154
7	2018120034	2021	2	A	35X CSE131 20130019
8	2018120034	2021	2	A	71X CSE132 20130056
9	2014172008	2021	2	A	98X CSE242 20091015
10	2014172008	2021	2	A	77X CSE244 20091039

SELECT 학번, 2021, 2, 이수구분,  
ROUND(DBMS\_RANDOM.VALUE(30, 100)),  
재수강여부, 교과목코드, 교수사번 FROM 수강;

취득점수는 난수를 발생하여 30점~100점을 지정하는 SQL 문 작성

# 서브쿼리

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 기타 서브쿼리

#### INSERT 문의 VALUES 절에서 서브쿼리 사용하기

워크시트 | 질의 작성기  
 INSERT 학번, 2021, 2, 이수구분, 교과목코드, 교수사번 FROM 수강  
 질의 출력 x | 질의 결과 x  
 SQL | 인출된 모든 행: 39(0,0032)  
 학번 | 2021 | 2 | 이수구분 |  
 1 2018120034 2021 2A  
 2 2018120034 2021 2A  
 3 2018120034 2021 2A  
 4 2018120034 2021 2A  
 5 2018120034 2021 2A  
 6 2018120034 2021 2A  
 7 2018120034 2021 2A  
 8 2018120034 2021 2A  
 9 2014172008 2021 2A  
 10 2014172008 2021 2A

학번	2021	2	이수구분
1 2018120034	2021	2A	
2 2018120034	2021	2A	
3 2018120034	2021	2A	
4 2018120034	2021	2A	
5 2018120034	2021	2A	
6 2018120034	2021	2A	
7 2018120034	2021	2A	
8 2018120034	2021	2A	
9 2014172008	2021	2A	
10 2014172008	2021	2A	

#### INSERT INTO 수강

```
SELECT 학번, 2021, 2, 이수구분,
ROUND(DBMS_RANDOM.VALUE(30, 100)),
재수강여부, 교과목코드, 교수사번
FROM 수강;
```

이제 여기에 INSERT 문을 추가하여 서브쿼리를 완성

서브쿼리

서브쿼리

## ◆ 서브쿼리 예

### + 기타 서브쿼리

#### UPDATE 문의 SET 절에서 서브쿼리 사용하기

워크시트 | 질의 작성기  
 UPDATE 수강 A  
 SET A.취득점수 =  
 (SELECT B.취득점수 - 5  
 FROM 수강 B  
 WHERE  
 B.수강년도 = 2021 AND  
 B.수강학기 = 1 AND  
 B.교과목코드 = A.교과목코드 AND  
 B.학번 = A.학번  
 )  
 WHERE  
 A.수강년도 = 2021 AND  
 A.수강학기 = 1;

```
UPDATE 수강 A
SET A.취득점수 =
(
  SELECT B.취득점수 - 5
  FROM 수강 B
  WHERE
    B.수강년도 = 2021 AND
    B.수강학기 = 1 AND
    B.교과목코드 = A.교과목코드 AND
    B.학번 = A.학번
)
WHERE
  A.수강년도 = 2021 AND
  A.수강학기 = 1;
```

이 쿼리는 Oracle 11.2 이하의 버전에서는 오류 발생

수강테이블의 2021년 1학기 데이터 중 모든 학생의 취득점수를 각각 5점씩 빼기

# 실습하기

서브쿼리

실습하기

- + Oracle Database 19c 버전 사용
- + 최신 버전인 Oracle Database 21c는 현재 cloud에서만 사용 가능하며, 지원 기간은 2년

- 수강 테이블의 2021년 1학기 데이터를 가공하여 2021년 2학기 데이터를 생성하는 SQL 문을 작성하시오.  
취득점수는 난수를 발생하여 50점~90점을 지정하는 SQL 문을 작성하시오.  
단, 학번, 이수구분, 재수강여부, 교과목코드, 교수사번은 2021년 1학기 데이터와 동일합니다.
- 같은 방법으로 2022년 1학기, 2학기 데이터를 생성하시오.

- 2022년 2학기에 CSE132과목을 수강한 학생의 점수를 +10점씩 올리기 바랍니다.

- 스칼라(scalar) 서브쿼리를 사용하여 2022년 1학기 학생들의  
학번, 성명, 수강년도, 수강학기, 교과목코드, 취득점수, 최고점수, 평균점수를 구하기 바랍니다.
- 스칼라(scalar) 서브쿼리를 사용하여 2022년 1학기 학생들중 각 과목 평균점수 이하인 학생들의  
학번, 성명, 수강년도, 수강학기, 교과목코드, 취득점수, 최고점수, 평균점수를 구하기 바랍니다.

## 서브쿼리로 데이터 삽입하기



### 실습단계

서브쿼리로 데이터 삽입하기

수강 테이블의 구조

DBMS\_RANDOM.VALUE 함수는 난수 발생시 범위를 지정하여 생성

서브쿼리로 데이터 수정하기

스칼라(scalar) 서브쿼리 예1

스칼라 서브쿼리 - 컬럼 부분에 서브쿼리가 있는 것

스칼라(scalar) 서브쿼리 예2

서브쿼리 예3

IN 연산자 - 결과중 하나라도 일치하면 참

오라클 JOIN을 ANSI JOIN으로 변경