

SQL - DML

문혜영

실습 테이블 생성

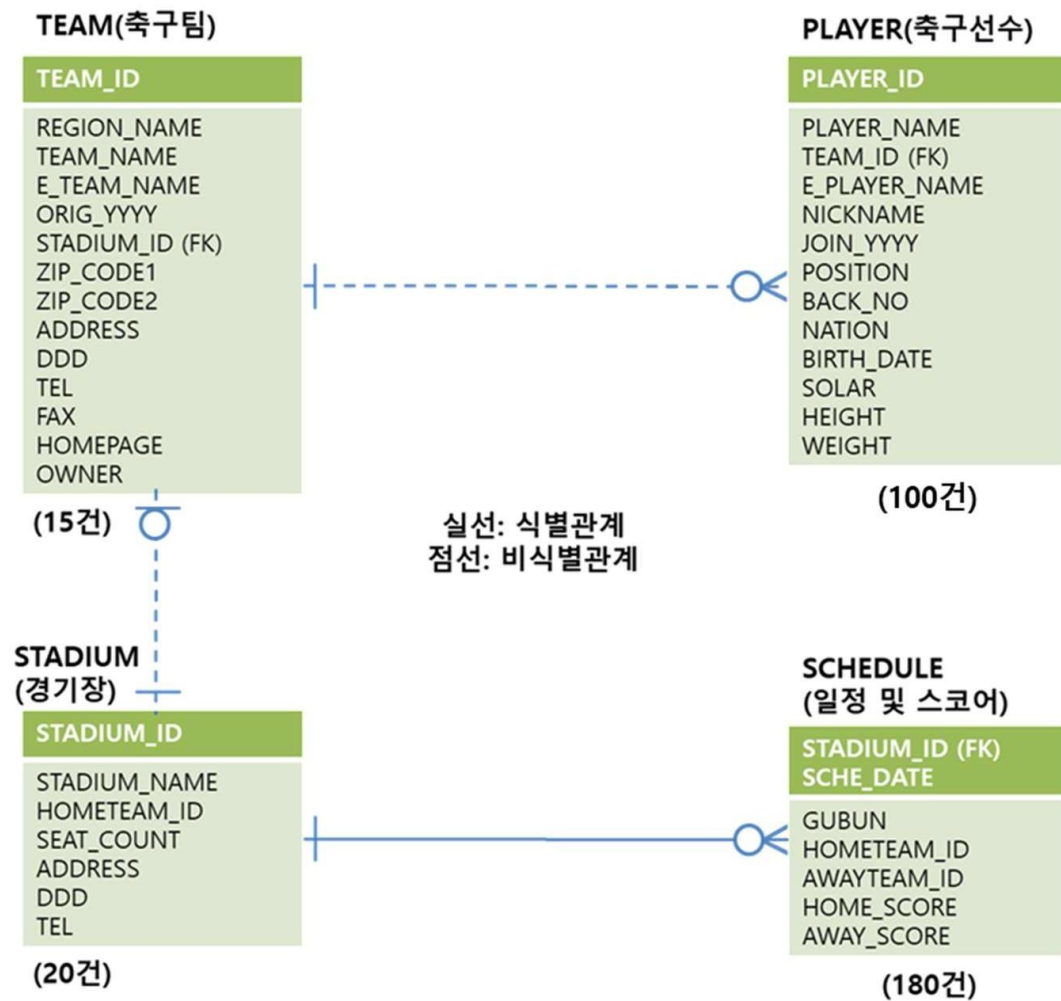
- K_League.sql
 - 4개의 테이블 모두 삭제 (DROP ~)
 - 4개의 테이블 다시 생성 (CREATE ~)
 - 4개의 테이블에 데이터 삽입 (INSERT ~)
- 실습 순서
 - SQL Developer 실행 my_conn 접속 (myid / mypw)
 - 워크시트에 K_League.sql 내용 복사 & 붙여넣기
 - 스크립트 실행() 또는 F5 로 실행
 -  은 한 문장만 실행(Ctrl+enter),  은 모든 문장 실행
 - 스크립트 출력 창에서 결과 확인

실습 테이블 생성

예제 데이터 (K-리그)

테이블	칼럼
PLAYER	<u>선수ID</u> , 선수명, <u>소속팀ID</u> , 영문선수명, 선수별명, 입단년도, 포지션, 등번호, 국적, 생년월일, 양/음, 키, 몸무게
TEAM	<u>팀ID</u> , 연고지명, 팀명, 영문팀명, 창단년도, <u>운동장ID</u> , 우편번호1, 우편번호2, 주소, 지역번호, 전화번호, 팩스, 홈페이지, 구단주
STADIUM	<u>운동장ID</u> , 운동장명, 홈팀ID, 좌석수, 주소, 지역번호, 전화번호
SCHEDULE	<u>운동장ID</u> , <u>경기일자</u> , 경기진행여부, 홈팀ID, 원정팀ID, 홈팀득점, 원정팀 득점

실습 테이블 생성



SELECT

- SELECT
 - 테이블에 존재하는 레코드의 값을 조회
- SELECT * FROM 테이블명 ;
 - 해당 테이블의 **모든** 칼럼값 조회
- SELECT [ALL / **DISTINCT**] 칼럼1, 칼럼2, ... FROM 테이블명
 - ALL: 중복 데이터 모두 출력 (default)
 - DISTINCT: **중복** 데이터를 1건으로 출력 (SELECT DISTINCT ~)
 - DISTINCT 키워드는 첫 칼럼의 앞에 위치해야 함
 - 칼럼의 조합에 대해 중복 체크
 - NULL 값도 하나의 값으로 간주함(NULL값도 1회만 표시됨)

SELECT

```
DESC PLAYER ; /* 구조확인 */
SELECT * FROM PLAYER ;
```

100개의 행이 인출됨(0.009초)

	PLAYER_ID	PLAYER_NAME	TEAM_ID
1	2009175	우르모브	K06
2	2007188	윤희순	K06
3	2012073	김규호	K06
4	2007178	김민성	K06

```
SELECT POSITION
FROM PLAYER;
```

100개의 행이 인출됨(0.009초)

	POSITION
1	DF
2	DF
3	DF

```
SELECT ALL POSITION
FROM PLAYER;
```

```
SELECT PLAYER_ID, PLAYER_NAME, TEAM_ID, POSITION
FROM PLAYER;
```

100개의 행이 인출됨(0.009초)

	PLAYER_ID	PLAYER_NAME	TEAM_ID	POSITION
1	2009175	우르모브	K06	DF
2	2007188	윤희순	K06	DF
3	2012073	김규호	K06	DF
4	2007178	김민성	K06	DF

```
SELECT DISTINCT POSITION
FROM PLAYER;
```

NULL값도하나의값

인출된 모든 행: 5

	POSITION
1	(null)
2	GK
3	DF
4	FW
5	MF

```
SELECT DISTINCT TEAM_ID, POSITION
FROM PLAYER;
```

인출된 모든 행: 11(0.004초)

	TEAM_ID	POSITION
1	K10	MF
2	K04	(null)
3	K10	DF
4	K10	FW
5	K06	DF
6	K04	DF
7	K10	GK
8	K06	MF
9	K06	FW
10	K04	GK
11	K06	GK

조합에 대한
중복체크

- SELECT-별칭 사용
 - 조회 결과에 일종의 별칭(ALIAS)을 부여하여 칼럼 레이블을 변경함
 - 칼럼명과 별칭 사이에 AS 키워드를 사용 (optional)
 - 별칭이 **공백**, 특수문자 등을 포함하는 경우 큰 **따옴표** 사용

```
SELECT  PLAYER_ID
FROM    PLAYER ;
```

	PLAYER_ID
1	2009175
2	2007188

```
SELECT  PLAYER_ID, 5+4
FROM    PLAYER ;
```

	PLAYER_ID	5+4
1	2009175	9
2	2007188	9

```
SELECT  *
FROM    DUAL ;
```

	DUMMY
1	X

```
SELECT  5+4
FROM    DUAL ;
```

	5+4
1	9

```
SELECT  5+4 AS RESULT
FROM    DUAL;
```

	RESULT
1	9

```
SELECT PLAYER_NAME , POSITION , HEIGHT
FROM PLAYER;
```

SQL | 100개의 행이 호출됨(0.006초)

	PLAYER_NAME	POSITION	HEIGHT
1	우르모브	DF	180
2	윤희순	DF	180
3	김규호	DF	177

```
SELECT PLAYER_NAME 선수명, POSITION 위치, HEIGHT 키
FROM PLAYER;
```

SQL | 100개의 행이 호출됨(0.007초)

	선수명	위치	키
1	우르모브	DF	180
2	윤희순	DF	180
3	김규호	DF	177

```
SELECT PLAYER_NAME AS 선수명, POSITION AS 위치, HEIGHT AS 키
FROM PLAYER;
```

```
SELECT PLAYER_NAME AS 선수 이름, POSITION AS 위치, HEIGHT AS 선수 키
FROM PLAYER;
```

```
SELECT PLAYER_NAME AS "선수 이름", POSITION AS 위치, HEIGHT AS "선수-키"
FROM PLAYER;
```

SQL | 100개의 행이 호출됨(0.006초)

	선수 이름	위치	선수-키
1	우르모브	DF	180
2	윤희순	DF	180
3	김규호	DF	177

- ORDER BY

- 출력시 정렬 기준 설정, SQL 문장의 맨 마지막에 위치
- 오름차순: ASC (생략 가능), 내림차순: DESC

- 참고) ORACLE에서 NULL은 가장 큰 값으로 취급됨

- 예) PLAYER_NAME과 HEIGHT를 키 오름차순으로 출력

```
SELECT PLAYER_NAME, HEIGHT
FROM PLAYER
ORDER BY HEIGHT ASC ;
```

```
SELECT PLAYER_NAME, HEIGHT
FROM PLAYER
ORDER BY HEIGHT ;
```

```
SELECT PLAYER_NAME, HEIGHT
FROM PLAYER
ORDER BY 2 ;
```

SQL | 100개의 행이 인출됨

	PLAYER_NAME	HEIGHT
1	하리	168
2	오비나	169
3	김장관	170
4	정농선	170
5	정민기	171
6	정호근	172
7	정국진	172
8	홍광철	172
9	장철우	172

- 예) HEIGHT 내림차순, PLAYER_NAME, 오름차순

```
SELECT PLAYER_NAME, HEIGHT
FROM PLAYER
ORDER BY HEIGHT DESC, PLAYER_NAME ASC ;
```

SQL | 100개의 행이 인출됨

	PLAYER_NAME	HEIGHT
1	김경태	(null)
2	김태호	(null)
3	전기현	(null)
4	정상수	(null)
5	최경훈	(null)
6	김석	194
7	이현	192
8	우성용	191
9	다오	190
10	정용대	189
11	박상남	188

- 예) 테이블에는 있지만 SELECT문에 없는 필드명을 사용할 수 있음

```
SELECT PLAYER_NAME, HEIGHT
FROM PLAYER
ORDER BY POSITION ;
```

SQL | 100개의 행이 인출됨

	PLAYER_NAME	HEIGHT
1	우르모브	180
2	윤희순	180
3	김규호	177
4	김민성	182
5	김장관	170

- WHERE 절

- 특정 **조건**을 만족하는 데이터를 한정하기 위해 사용
- 형태
 - SELECT ~ FROM ~ WHERE ~

- 예) PLAYER테이블에서 POSITION이 MF인 선수의 PLAYER_ID와 POSITION필드만 표시

```
SELECT PLAYER_ID, POSITION
FROM PLAYER
WHERE POSITION='MF';
```

```
SELECT PLAYER_ID, POSITION
FROM PLAYER;
```

```
SELECT PLAYER_ID, POSITION
FROM PLAYER
WHERE POSITION = 'GK';
```

```
SELECT PLAYER_ID, POSITION
FROM PLAYER
WHERE POSITION = 'gk';
```

SQL | 인출된 모든 행: 2600

	PLAYER_ID	POSITION
1	2007189	MF
2	2011049	MF

SQL | 인출된 모든 행: 1000

	PLAYER_ID	POSITION
1	2008499	GK
2	2011021	GK
3	2012052	GK

SQL | 인출된 모든 행: 0(0)

	PLAYER...	POSITION
--	-----------	----------

SQL | 100개의 행이

	PLAYER_ID	POSITION
1	2009175	DF
2	2007188	DF
3	2012073	DF
4	2007178	DF

WHERE - 연산자

- WHERE - 연산자

구분	연산자	의미
산술 연산자	+	덧셈
	-	뺄셈
	*	곱셈
	/	나눗셈
비교 연산자	=	같다
	<>	같지 않다
	>	크다
	>=	크거나 같다
	<	작다
	<=	작거나 같다
논리 연산자	AND	두 조건이 모두 참이면 참
	OR	한 조건만 참이어도 참
	NOT	제시된 조건이 거짓이어야 참

WHERE - 연산자

구분	연산자	의미
SQL 연산자		<ul style="list-style-type: none">• 두 문자열을 하나로 연결한 문자열 반환
	BETWEEN a AND b	<ul style="list-style-type: none">• a와 b의 사이의 값 반환 (a, b 포함)• cf) NOT BETWEEN a AND b (BETWEEN a AND b)의 반대
	IN (list)	<ul style="list-style-type: none">• list에 있는 값 중 하나만 일치해도 참• cf) NOT IN (list)
	LIKE '비교 문자열'	<ul style="list-style-type: none">• 비교 문자열과 일치하면 참• 와일드카드 (% , _) 사용 가능함• cf) NOT LIKE '비교 문자열'
	IS NULL	<ul style="list-style-type: none">• NULL 값인 경우 참• cf) IS NOT NULL

WHERE - 연산자

- 연산자 우선 순위

우선 순위	연산자
1	괄호 ()
2	NOT 연산자
3	비교 연산자, SQL 연산자
4	AND
5	OR

WHERE - 연산자

- 산술 연산자

- NUMBER와 DATE 자료형에 대해 적용, 연산자는 select문에도 사용가능

- *, /, +, -

```
SELECT PLAYER_NAME , (HEIGHT-100)* 0.9 - WEIGHT  
FROM PLAYER ;
```

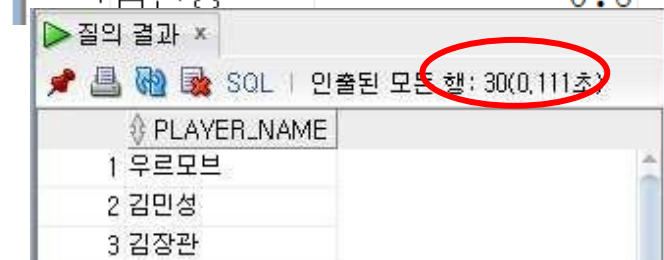
```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE ( (HEIGHT-100) * 0.9 - WEIGHT ) > 0 ;
```

```
SELECT PLAYER_NAME, (HEIGHT-100) * 0.9 - WEIGHT  
FROM PLAYER  
WHERE ( ( HEIGHT-100 ) * 0.9 - WEIGHT ) > 0 ;
```

```
SELECT PLAYER_NAME, (HEIGHT-100) * 0.9 - WEIGHT as 비만도  
FROM PLAYER  
WHERE ( ( HEIGHT-100 ) * 0.9 - WEIGHT ) > 0 ;
```



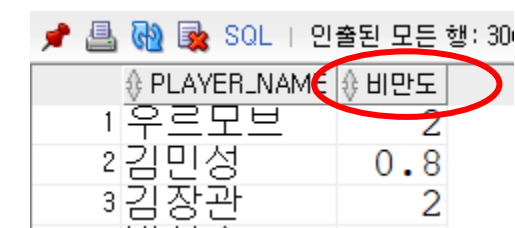
	PLAYER_NAME	(HEIGHT-100)*0.9-WEIGHT
1	우르모브	2
2	윤희순	-2
3	김규호	-2.7
4	김민성	0.8



	PLAYER_NAME	(HEIGHT-100)*0.9-WEIGHT
1	우르모브	
2	김민성	
3	김장관	



	PLAYER_NAME	(HEIGHT-100)*0.9-WEIGHT
1	우르모브	2
2	김민성	0.8
3	김장관	2



	PLAYER_NAME	비만도
1	우르모브	2
2	김민성	0.8
3	김장관	2


WHERE - 연산자

- NULL연산

- 데이터확인("김태호"의 Height가 null임)

```
SELECT PLAYER_NAME, HEIGHT  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호';
```

```
SELECT PLAYER_NAME, HEIGHT, HEIGHT + 0, HEIGHT - 0, HEIGHT * 0, HEIGHT / 0  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호 ' ;
```

 SQL | 인출된 모든 행: 1(0.003초)

	PLAYER_NAME	HEIGHT	HEIGHT+0	HEIGHT-0	HEIGHT*0	HEIGHT/0
1	김태호	(null)	(null)	(null)	(null)	(null)

WHERE - 연산자

- Number 와 date 연산

- 생년월일 확인

```
SELECT PLAYER_NAME, BIRTH_DATE  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호';
```

SQL | 인출된 모든 행: 1(0,001초)

	PLAYER_NAME	BIRTH_DATE
1	김태호	71/01/29

- 3일 더하기

```
SELECT PLAYER_NAME, BIRTH_DATE, BIRTH_DATE+3  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호';
```

SQL | 인출된 모든 행: 1(0,002초)

	PLAYER_NAME	BIRTH_DATE	BIRTH_DATE+3
1	김태호	71/01/29	71/02/01

- 3개월 더하기

```
SELECT PLAYER_NAME, BIRTH_DATE, ADD_MONTHS(BIRTH_DATE, 3)  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호';
```

SQL | 인출된 모든 행: 1(0,001초)

	PLAYER_NAME	BIRTH_DATE	ADD_MONTHS(BIRTH_DATE,3)
1	김태호	71/01/29	71/04/29

- 3년 더하기

```
SELECT PLAYER_NAME, BIRTH_DATE, ADD_MONTHS(BIRTH_DATE, 36)  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호';
```

SQL | 인출된 모든 행: 1(0,002초)

	PLAYER_NAME	BIRTH_DATE	ADD_MONTHS(BIRTH_DATE,36)
1	김태호	71/01/29	74/01/29

```
SELECT PLAYER_NAME, BIRTH_DATE, BIRTH_DATE + INTERVAL '3' YEAR  
FROM PLAYER  
WHERE PLAYER_NAME = '김태호';
```


WHERE - 연산자

- 비교 연산자

- =, <>, >=, >, <=, <
 - 모든 자료형에 대해 적용
 - 문자열의 크기 비교는 사전 순으로 수행됨
 - 예: '01' < '02' < '1' < '11' < '2'
 - NULL에는 비교 연산자는 사용 불가

SELECT PLAYER_ID, PLAYER_NAME, NATION
FROM PLAYER
WHERE NATION = NULL;

SQL | 인출된 모든 행: 0(0.002초)

PLAYER...	PLAYER...	NATION
-----------	-----------	--------

SELECT PLAYER_ID, PLAYER_NAME, NATION
FROM PLAYER
WHERE NATION IS NULL;

SQL | 50개의 행이 인출됨(0.002초)

PLAYER_ID	PLAYER_NAME	NATION
1 2007188	윤희준	(null)
2 2012073	김규호	(null)
3 2007178	김민성	(null)

SELECT PLAYER_ID, PLAYER_NAME, NATION
FROM PLAYER
WHERE NATION <> NULL;

SQL | 인출된 모든 행: 0(0.002초)

PLAYER...	PLAYER...	NATION
-----------	-----------	--------

SELECT PLAYER_ID, PLAYER_NAME, NATION
FROM PLAYER
WHERE NATION IS NOT NULL;

SQL | 인출된 모든 행: 6(0.002초)

PLAYER_ID	PLAYER_NAME	NATION
1 2009175	우르모브	유고
2 2012137	이고르	브라질
3 2012127	디디	브라질

WHERE - 연산자

- 논리 연산자
 - 모든 자료형에 대해 적용
 - NOT, AND, OR (우선순위: NOT > AND > OR)
 - Q) 다음 SQL문중 실행 결과가 나머지 두가지와 다른것은?

우선 순위	연산자
1	괄호 ()
2	NOT 연산자
3	비교 연산자, SQL 연산자
4	AND
5	OR

1

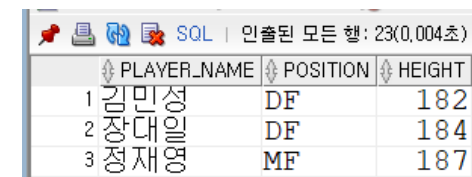
```
SELECT PLAYER_NAME, POSITION, HEIGHT  
FROM PLAYER  
WHERE POSITION <> 'GK' AND HEIGHT > 180;
```

2

```
SELECT PLAYER_NAME, POSITION, HEIGHT  
FROM PLAYER  
WHERE NOT(POSITION = 'GK') AND HEIGHT > 180;
```

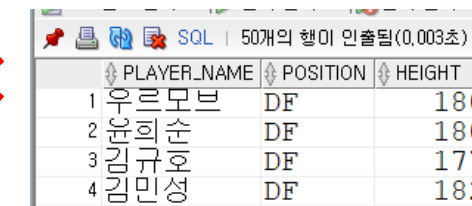
3

```
SELECT PLAYER_NAME, POSITION, HEIGHT  
FROM PLAYER  
WHERE NOT(POSITION = 'GK' AND HEIGHT > 180);
```



SQL | 인출된 모든 행: 23(0.004초)

	PLAYER_NAME	POSITION	HEIGHT
1	김민성	DF	182
2	장대일	DF	184
3	정재영	MF	187



SQL | 50개의 행이 인출됨(0.003초)

	PLAYER_NAME	POSITION	HEIGHT
1	우르모브	DF	180
2	윤희순	DF	180
3	김규호	DF	177
4	김민성	DF	182

WHERE - 연산자

- 논리 연산자
 - 진리표

P	Q	P AND Q
참	참	참
참	거짓	거짓
거짓	참	거짓
거짓	거짓	거짓

P	Q	P OR Q
참	참	참
참	거짓	참
거짓	참	참
거짓	거짓	거짓

P	NOT P
참	거짓
거짓	참

- Q) NULL을 포함한 진리표의 (1)~(7)을 완성하십시오.

P	Q	P AND Q
참	참	참
참	거짓	거짓
참	NULL	(1)
거짓	참	거짓
거짓	거짓	거짓
거짓	NULL	(2)
NULL	참	(1)
NULL	거짓	(2)
NULL	NULL	(3)

P	Q	P OR Q
참	참	참
참	거짓	참
참	NULL	(4)
거짓	참	참
거짓	거짓	거짓
거짓	NULL	(5)
NULL	참	(4)
NULL	거짓	(5)
NULL	NULL	(6)

P	NOT P
참	거짓
거짓	참
NULL	(7)

WHERE - 연산자

- SQL 연산자(||)
 - 합성(연결) 연산자 - 문자열과 문자열을 **연결** 함
 - 방법1: CONCAT(str1, str2)
 - 방법2: str1 || str2

```
SELECT PLAYER_NAME, Height || 'Cm' AS "선수 신장"  
FROM PLAYER;
```

```
SELECT PLAYER_NAME, CONCAT(Height, 'Cm') AS "선수 신장"  
FROM PLAYER;
```

```
SELECT PLAYER_NAME, CONCAT('A', 'B')  
FROM PLAYER ;
```

```
SELECT PLAYER_NAME, 'A' || 'B')  
FROM PLAYER ;
```

```
SELECT PLAYER_NAME, CONCAT('A', 'B', 'C')  
FROM PLAYER;
```

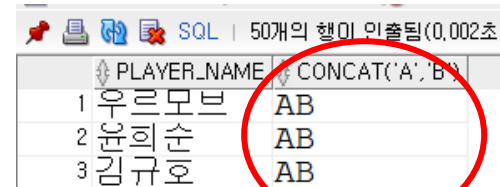
```
SELECT PLAYER_NAME, 'A' || 'B' || 'C'  
FROM PLAYER;
```



스크립트 출력 x | 질의 결과 x

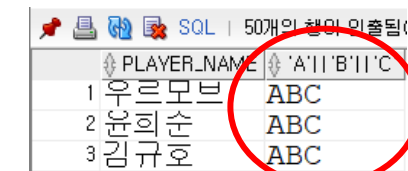
SQL | 인출된 모든 행: 100(0.015초)

	PLAYER_NAME	선수 신장
1	우르모브	180Cm
2	윤희준	180Cm
3	김규호	177Cm
4	김민성	182Cm
5	김장관	170Cm



SQL | 50개의 행이 인출됨(0.002초)

	PLAYER_NAME	CONCAT('A','B')
1	우르모브	AB
2	윤희준	AB
3	김규호	AB



SQL | 50개의 행이 인출됨(0.002초)

	PLAYER_NAME	'A' 'B' 'C'
1	우르모브	ABC
2	윤희준	ABC
3	김규호	ABC

- Q) 다음 결과를 출력하기위한 SQL문을 작성 하시오.



The screenshot shows a SQL query result window with the title '질의 결과' (Query Result). The window displays a table with the following data:

	선수 신장
1	우르모브 선수: 180Cm
2	윤희준 선수: 180Cm

WHERE - 연산자

- SQL 연산자 (BETWEEN)
- BETWEEN

```
SELECT PLAYER_NAME 선수이름, POSITION 포지션, HEIGHT 키
FROM PLAYER
WHERE HEIGHT BETWEEN 170 AND 180 ;
```

```
SELECT PLAYER_NAME 선수이름, POSITION 포지션, HEIGHT 키
FROM PLAYER
WHERE (HEIGHT >=170) AND (HEIGHT <=180) ;
```

SQL | 인출된 모든 행: 60(0,003초)

	선수이름	포지션	키
1	우르모브	DF	180
2	윤희순	DF	180
3	김규호	DF	177
4	김장관	DF	170
5	김정호	DF	174

- NOT BETWEEN

```
SELECT PLAYER_NAME 선수이름, POSITION 포지션, HEIGHT 키
FROM PLAYER
WHERE HEIGHT NOT BETWEEN 170 AND 180 ;
```

```
SELECT PLAYER_NAME 선수이름, POSITION 포지션, HEIGHT 키
FROM PLAYER
WHERE (HEIGHT <170) OR (HEIGHT >180) ;
```

SQL | 인출된 모든 행: 35(0,003초)

	선수이름	포지션	키
1	김민성	DF	182
2	장대일	DF	184
3	정재영	MF	187
4	이고르	MF	181

- Q)위의 두 연산결과에 포함되지않은 5개의 행을 출력하시오.

- Null 개수 확인

```
SELECT PLAYER_NAME 선수이름, POSITION 포지션, HEIGHT 키  
FROM PLAYER  
WHERE HEIGHT IS NULL ;
```

SQL | 인출된 모든 행: 5(0,001)

	선수이름	포지션	키
1	김경태	DF	(null)
2	김태호	DF	(null)
3	정상수	DF	(null)
4	전기현	DF	(null)
5	최경훈	DF	(null)

WHERE - 연산자

- SQL 연산자 (IN)

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION
FROM PLAYER
WHERE TEAM_ID='K04' ;
```

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION
FROM PLAYER
WHERE TEAM_ID='K04' OR TEAM_ID='K06' ;
```

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION
FROM PLAYER
WHERE TEAM_ID='K04' OR TEAM_ID='K06' OR TEAM_ID='K10' ;
```

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION
FROM PLAYER
WHERE TEAM_ID IN ( 'K04', 'K06', 'K10' ) ;
```

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION
FROM PLAYER
WHERE (TEAM_ID, POSITION) IN ( ('K04', 'GK'), ('K06', 'MF') ) ;
```

SQL | 인출된 모든 행: 18(0.003초)

	선수이름	TEAM_ID	POSITION
1	이운경	K04	DF
2	하재훈	K04	DF
3	김중호	K04	DF

SQL | 인출된 모든 행: 64(0.003초)

	선수이름	TEAM_ID	POSITION
29	박상남	K06	FW
30	빅토르	K06	FW
31	이운경	K04	DF
32	하재훈	K04	DF
33	김중호	K04	DF

SQL | 100개의 행이 인출됨(0.005초)

	선수이름	TEAM_ID	POSITION
43	조승호	K04	DF
44	윤송희	K04	DF
45	김범직	K04	DF
46	김상홍	K04	(null)
47	김태호	K10	DF

SQL | 인출된 모든 행: 14(0.003초)

	선수이름	TEAM_ID	POSITION
1	정재영	K06	MF
2	정태민	K06	MF
12	김중호	K04	GK
13	이현	K04	GK
14	한농진	K04	GK

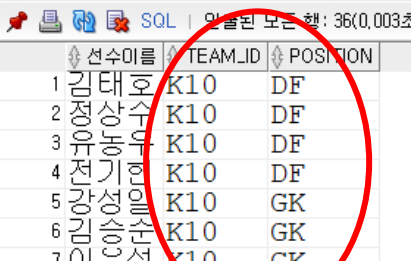
WHERE - 연산자

- SQL 연산자(NOT IN)

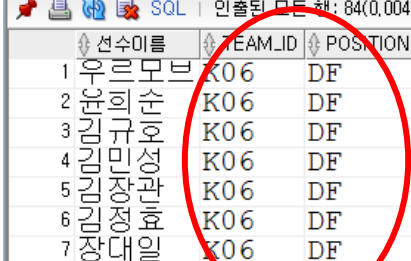
```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION  
FROM PLAYER  
WHERE TEAM_ID NOT IN ( 'K04', 'K06' ) ;
```

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION  
FROM PLAYER  
WHERE TEAM_ID <> 'K04' AND TEAM_ID <> 'K06' ;
```

```
SELECT PLAYER_NAME 선수이름, TEAM_ID, POSITION  
FROM PLAYER  
WHERE (TEAM_ID, POSITION) NOT IN ( ('K04', 'GK'), ('K06', 'MF') ) ;
```



	선수이름	TEAM_ID	POSITION
1	김태호	K10	DF
2	정상수	K10	DF
3	유동우	K10	DF
4	전기현	K10	DF
5	강성일	K10	GK
6	김성준	K10	GK
7	이근성	K10	GK

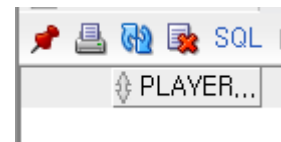


	선수이름	TEAM_ID	POSITION
1	우르모브	K06	DF
2	윤의순	K06	DF
3	김규호	K06	DF
4	김민성	K06	DF
5	김장관	K06	DF
6	김정효	K06	DF
7	장대일	K06	DF
8	박상수	K06	DF

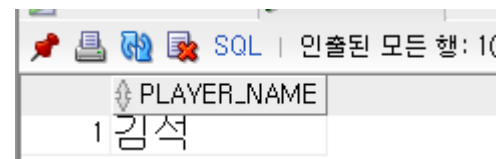
WHERE - 연산자

- SQL 연산자 (LIKE)
 - 문자열 비교 연산
 - 와일드카드 사용 가능
 - '%': 임의의 문자 N개 / '_' : 임의의 문자 1개)
- Q) 다음3개의 SQL문장 중, 우측의 실행결과를 얻기 위한 문장은?

```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE PLAYER_NAME LIKE '김' ;
```



```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE PLAYER_NAME LIKE '김_' ;
```



```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE PLAYER_NAME LIKE '김%' ;
```



WHERE - 출력 개수 지정

- ROWNUM (TOP N개의 레코드 반환)
 - ROWNUM - 사용자가 아닌 시스템이 관리하는 Pseudo Column

```
DESC PLAYER;
```



```
SELECT PLAYER_NAME, ROWNUM  
FROM PLAYER;
```

이름	널?	유형
PLAYER_ID	NOT NULL	CHAR (7)
PLAYER_NAME	NOT NULL	VARCHAR2 (20)
TEAM_ID	NOT NULL	CHAR (3)
E_PLAYER_NAME		VARCHAR2 (40)
NICKNAME		VARCHAR2 (30)
JOIN YYYY		CHAR (4)
POSITION		VARCHAR2 (10)
BACK_NO		NUMBER (2)
NATION		VARCHAR2 (20)
BIRTH_DATE		DATE
SOLAR		CHAR (1)
HEIGHT		NUMBER (3)
WEIGHT		NUMBER (3)

SQL | 인출된 모든 행: 100(0,127초)

PLAYER_NAME	ROWNUM
1 우르모브	1
2 윤희준	2
3 김규호	3
4 김민성	4
5 김장관	5

WHERE - 출력 개수 지정

ROWNUM (TOP N개의 레코드 반환)

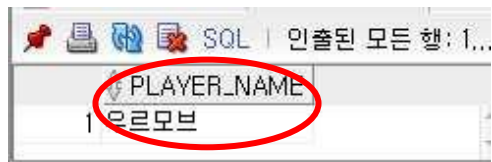
- Q) 다음 6개의 SQL문장중, 아무결과가 출력되지 않는문장은?

- 한 행만 반환할 때

```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE ROWNUM=1;
```

```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE ROWNUM<=1;
```

```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE ROWNUM<2;
```



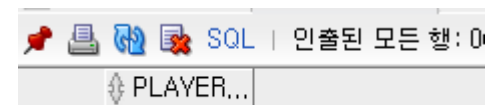
PLAYER_NAME
1 우르모브

- 여러 행 반환할 때

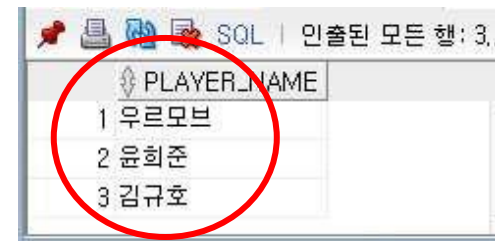
```
SELECT PLAYER_NAME  
FROM PLAYER WHERE  
ROWNUM=3;
```

```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE ROWNUM<=3;
```

```
SELECT PLAYER_NAME  
FROM PLAYER  
WHERE ROWNUM<4;
```



PLAYER...



PLAYER_NAME
1 우르모브
2 윤희준
3 김규호

WHERE - 출력 개수 지정

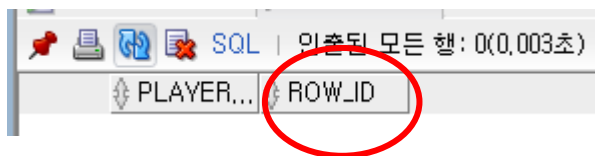
- ROWNUM (TOP N개의 레코드 반환)
 - 테이블 내의 UNIQUE한 값 설정에도 사용 가능
 - ROWNUM을 이용하여 ID 필드 생성
 - UPDATE 테이블명 SET 칼럼명 = ROWNUM;
 - UPDATE 문은 추후 설명

```
ALTER TABLE PLAYER ADD (ROW_ID NUMBER);  
DESC PLAYER;
```

```
UPDATE PLAYER SET ROW_ID = ROWNUM;
```

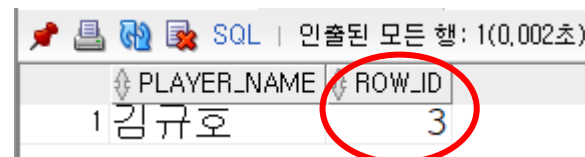
```
SELECT PLAYER_NAME, ROW_ID  
FROM PLAYER  
WHERE ROWNUM=3;
```

```
SELECT PLAYER_NAME, ROW_ID  
FROM PLAYER  
WHERE ROW_ID = 3;
```



인출된 모든 행: 0(0,003초)

PLAYER...	ROW_ID
-----------	--------



인출된 모든 행: 1(0,002초)

PLAYER_NAME	ROW_ID
1 김규호	3

이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)
ROW_ID		NUMBER

- INSERT
 - 테이블에 한 건의 레코드를 추가함
 - 여러 건 입력 시 INSERT ALL ~ 구문 활용
 - 문자 또는 날짜 값의 경우 작은 따옴표로 묶음
 - 숫자 데이터는 작은 따옴표 없이 사용
 - 두 가지 유형으로 입력 가능
 - INSERT INTO 테이블명 (COLUMN_LIST) VALUES (VALUE_LIST);
 - 일부 특정한 칼럼에 대응되는 값만 입력
 - INSERT INTO 테이블명 VALUES (전체 COLUMN의 VALUE_LIST);
 - 전체 칼럼에 대응되는 값을 순서대로 모두 입력

- INSERT

- INSERT INTO 테이블명 (COLUMN_LIST) VALUES (VALUE_LIST);

- 칼럼 순서는 실제 테이블의 칼럼 순서와 무관
 - 정의하지 않은 칼럼은 NULL 값이 입력됨

```
INSERT INTO PLAYER (PLAYER_ID, PLAYER_NAME, TEAM_ID, BIRTH_DATE) VALUES ('2999001', '손흥민', 'K07', '1999-01-01');
```

PLAYER_ID	PLAYER_NAME	TEAM_ID	E_PLAYER_NAME	NICKNAME	JOIN YYYY	POSITION	BACK_NO	NATION	BIRTH_DATE	SOLAR	HEIGHT	WEIGHT
1	2009175	무문오보	004	(null)	(null)	2009	DF	4	02	07/08/30	1	180
2	2007188	손흥민	004	(null)	(null)	2008	DF	15	(null)	02/11/91	2	180
3	2012078	김재환	004	(null)	(null)	2011	DF	20	(null)	09/07/19	1	177

- INSERT INTO 테이블명 VALUES (전체 COLUMN의 VALUE_LIST);

- 전체 칼럼의 모든 값을 순서대로 입력해야 함
 - 빈 값은 NULL 또는 작은따옴표 둘(")로 입력 (주의 ' ' 이 아님)

```
INSERT INTO PLAYER  
VALUES ('2999002', '이승우', 'K07', "", "", '2010', 'MF', '10', NULL, NULL, NULL, NULL, NULL);
```



레코드 삽입

- INSERT ALL
 - 동시에 여러 레코드를 추가하는 경우
 - 테이블 생성 후 초기 데이터 일괄 업로드
 - 기존 테이블의 레코드 조회 후 다른 테이블에 삽입
 - 예) Table1에서 레코드를 반환하여 Table2, Table3에 분할 저장

실행하세요

```
INSERT ALL  
  INTO TABLE2 VALUES (ID, NAME)  
  INTO TABLE3 VALUES (ID, SALARY)  
SELECT ID, NAME, SALARY FROM TABLE1 ;
```

Table 1

ID	NAME	SALARY
11	홍길동	70,000
22	강감찬	90,000
33	김유신	80,000

Table 2

ID	NAME
11	홍길동
22	강감찬
33	김유신

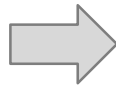
Table 3

ID	SALARY
11	70,000
22	90,000
33	80,000

INSERT ALL –cont'd (참고)

- INSERT ALL은 반드시 SELECT문을 사용해야 함
- 새로운 값을 직접 입력하는 경우 아래와 같이 DUAL 테이블 사용
DUAL –하나의 칼럼(DUMMY)에 하나의 값('X')을 저장하고 있음

```
SELECT * FROM DUAL;
```



DUMMY
X

- 예) STADIUM 테이블에 두 레코드를 한꺼번에 삽입하는 경우

```
INSERT ALL  
  INTO STADIUM( STADIUM_ID, STADIUM_NAME) VALUES ('TP1', '임시경기장1')  
  INTO STADIUM( STADIUM_ID, STADIUM_NAME) VALUES ('TP2', '임시경기장2')  
SELECT * FROM DUAL ;
```

```
SELECT * FROM STADIUM ;
```

- DELETE

- 테이블에 존재하는 전체 레코드 삭제

- DELETE 테이블명 / DELETE FROM 테이블명

```
DELETE PLAYER;
```

또는

```
DELETE FROM PLAYER;
```

- 일반적으로는 WHERE 절을 사용하여 특정 레코드를 삭제함

```
DELETE FROM STADIUM  
WHERE STADIUM_ID = 'TP1';
```

UPDATE

- 테이블에 존재하는 전체 레코드의 값 변경

UPDATE 테이블명 **SET** 칼럼명 = 새로운 값

예) PLAYER 테이블에서 모든 레코드의 POSITION을 'GK'로 변경

```
UPDATE PLAYER SET POSITION = 'GK';
```

- 일반적으로는 WHERE절을 사용하여 특정 레코드의 값을 변경함

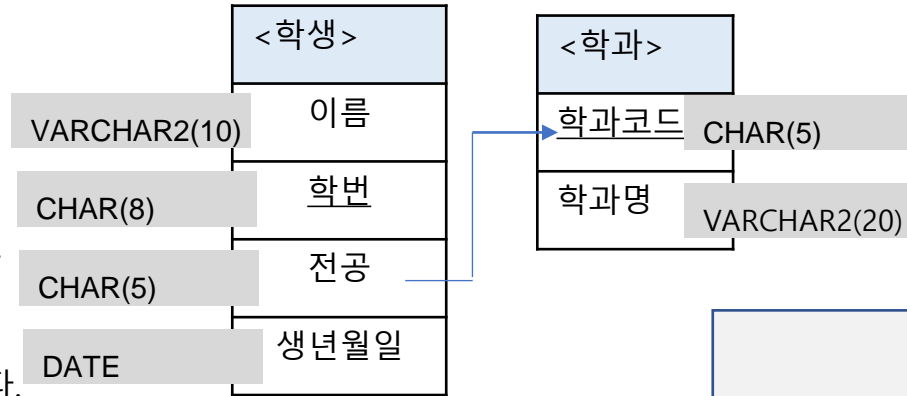
```
UPDATE STADIUM SET STADIUM_NAME = '우리경기장'  
WHERE STADIUM_ID = 'TP2';
```

('TP2', '임시경기장2') -> ('TP2', '우리경기장')

- 예제) '이름', '학번', '전공', '생년월일'로 구성된 <학생> 테이블을 정의하는 SQL문을 작성하시오.

- 단, 제약 조건은 다음과 같다.

- '이름'은 NULL이 올 수 없다.
- '학번'은 기본키이다.
- '전공'은 <학과> 테이블의 '학과코드'를 참조하는 외래키로 사용된다.
- 생년월일 속성은 DATE 자료형을 갖는다.
- <학과> 테이블에서 삭제가 일어나면 관련된 튜플들의 전공 값을 NULL로 만든다.
- '생년월일'은 1980-01-01 이후의 데이터만 저장할 수 있으며 제약 조건의 이름은 '생년월일제약'으로 한다.
- 각 속성의 데이터 타입은 적당하게 지정한다.



TEST

<학과>	
학과코드	학과명
A	경영정보
B	데이터사이언스
C	AI
D	성악

<학생>			
이름	학번	전공	생년월일
한나라	901	A	2000-10-09
홍장미	902	B	2000-06-01
이리아	903	C	2001-11-02
물보라	904	A	2001-01-03
다스리	905	B	2000-02-25
이루리	906	C	2002-03-04
은송이	907	D	2000-07-06

- <INSERT> 데이터를 삽입하시오.

SQL 인출된 모든 행: 4(0,002초)	
학과코드	학과명
1 A	경영정보
2 B	데이터사이언스
3 C	AI
4 D	성악

인출된 모든 행: 7(0,005초)

	이름	학번	전공	생년월일
1	한나라	K901	A	00/10/09
2	홍장미	K902	B	00/06/01
3	이리아	K903	C	01/11/02
4	물보라	K904	A	01/01/03
5	다스리	K905	B	00/02/25
6	이루리	K906	C	02/03/04
7	은송이	K907	D	00/07/06

TEST

- <DELETE>

<학과> 테이블에서 학과코드의 'D'를 삭제하시오.

- <UPDATE>

<학과> 테이블에서 학과명이 'AI'의 내용을 'AI융합'으로 변경하시오.

<학과>

SQL | 인출된 모든 행: 4(0,002초)

학과코드	학과명
1 A	경영정보
2 B	데이터사이언스
3 C	AI
4 D	성악

<학생>

SQL | 인출된 모든 행: 7(0,005초)

이름	학번	전공	생년월일
1 한나라	K901	A	00/10/09
2 홍장미	K902	B	00/06/01
3 이리아	K903	C	01/11/02
4 눌보라	K904	A	01/01/03
5 다스리	K905	B	00/02/25
6 이루리	K906	C	02/03/04
7 은송이	K907	D	00/07/06

SQL 인출된 모든 행: 3(0,002초)		SQL 인출된 모든 행: 7(0,002초)			
학과코드	학과명	이름	학번	전공	생년월일
1 A	경영정보	1 한나라	K901	A	00/10/09
2 B	데이터사이언스	2 홍장미	K902	B	00/06/01
3 C	AI	3 이리아	K903	C	01/11/02
		4 눌보라	K904	A	01/01/03
		5 다스리	K905	B	00/02/25
		6 이루리	K906	C	02/03/04
		7 은송이	K907	(null)	00/07/06

SQL 인출된 모든 행: 3(0,002초)		SQL 인출된 모든 행: 3(0,002초)	
학과코드	학과명	학과코드	학과명
1 A	경영정보	1 A	경영정보
2 B	데이터사이언스	2 B	데이터사이언스
3 C	AI	3 C	AI융합

- <ALTER TABLE>

<학생> 테이블에 '점수 NUMBER(3)' 속성을 추가하시오.

- <UPDATE>

<학생> 테이블의 점수란에 90, 80, 95, 85, 66, 55, 77을 추가하시오.

<학생> SQL | 인출된 모든 행: 7(0,002초)

	이름	학번	전공	생년월일
1	한나라	K901	A	00/10/09
2	홍장미	K902	B	00/06/01
3	이리아	K903	C	01/11/02
4	불보라	K904	A	01/01/03
5	다스리	K905	B	00/02/25
6	이루리	K906	C	02/03/04
7	은송이	K907	(null)	00/07/06



<학생> SQL | 인출된 모든 행: 7(0,003초)

	이름	학번	전공	생년월일	점수
1	한나라	K901	A	00/10/09	90
2	홍장미	K902	B	00/06/01	80
3	이리아	K903	C	01/11/02	95
4	불보라	K904	A	01/01/03	85
5	다스리	K905	B	00/02/25	66
6	이루리	K906	C	02/03/04	55
7	은송이	K907	(null)	00/07/06	77

TEST

- <학생> 테이블에서 다음의 조건에 만족하는 내용을 표시하시오.

- 이름이 '이'씨인 회원을 표시하시오.

- 전공이 null인 회원을 표시하시오.

- 점수가 90점 이상인 회원의 이름, 학번, 점수를 표시하시오.

- 전공이 A, B인 회원을 표시하시오.

- 점수에 '점'을 추가하여 점수표시 로 표시되게 하시오.

예. 90->90점

SQL | 인출된 모든 행: 2(0.002초)

	이름	학번	전공	생년월일	점수
1	이리아	K903	C	01/11/02	95
2	이루리	K906	C	02/03/04	55

SQL | 인출된 모든 행: 1(0.003초)

	이름	학번	전공	생년월일	점수
1	은송이	K907	(null)	00/07/06	77

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 2(0.004초)

	이름	학번	점수
1	한나라	K901	90
2	이리아	K903	95

SQL | 인출된 모든 행: 4(0.003초)

	이름	학번	전공	생년월일	점수
1	한나라	K901	A	00/10/09	90
2	홍장미	K902	B	00/06/01	80
3	늘보라	K904	A	01/01/03	85
4	다스리	K905	B	00/02/25	66

SQL | 인출된 모든 행: 7(0.003초)

	이름	학번	점수표시
1	한나라	K901	90점
2	홍장미	K902	80점
3	이리아	K903	95점
4	늘보라	K904	85점
5	다스리	K905	66점
6	이루리	K906	55점
7	은송이	K907	77점

<학생>

인출된 모든 행: 7(0.003초)

	이름	학번	전공	생년월일	점수
1	한나라	K901	A	00/10/09	90
2	홍장미	K902	B	00/06/01	80
3	이리아	K903	C	01/11/02	95
4	늘보라	K904	A	01/01/03	85
5	다스리	K905	B	00/02/25	66
6	이루리	K906	C	02/03/04	55
7	은송이	K907	(null)	00/07/06	77

수고하셨습니다 🙌