

시험에
나오는 것만
공부한다!



매년 출제되는

SQL 17문제

정보처리기사 실기



[SQL 명령문의 기본 형식과 조건]

1. 다음과 같이 테이블을 정의하고 튜플을 삽입하였을 때 각 번호(①, ②)의 SQL문을 실행한 결과를 쓰시오.



```
CREATE TABLE 부서 (  
    부서코드 INT PRIMARY KEY,  
    부서명 VARCHAR(20)  
);  
  
CREATE TABLE 직원 (  
    직원코드 INT PRIMARY KEY,  
    부서코드 INT,  
    직원명 VARCHAR(20),  
    FOREIGN KEY(부서코드) REFERENCES 부서(부서코드)  
        ON DELETE CASCADE  
);  
  
INSERT INTO 부서 VALUES(10, '영업부');  
INSERT INTO 부서 VALUES(20, '기획부');  
INSERT INTO 부서 VALUES(30, '개발부');  
  
INSERT INTO 직원 VALUES(1001, 10, '이진수');  
INSERT INTO 직원 VALUES(1002, 10, '곽연경');  
INSERT INTO 직원 VALUES(1003, 20, '김선길');  
INSERT INTO 직원 VALUES(1004, 20, '최민수');  
INSERT INTO 직원 VALUES(1005, 20, '이용갑');  
INSERT INTO 직원 VALUES(1006, 30, '박종일');  
INSERT INTO 직원 VALUES(1007, 30, '박미경');
```

① SELECT DISTINCT COUNT(부서코드) FROM 직원 WHERE 부서코드 = 20;

답 : 3

② DELETE FROM 부서 WHERE 부서코드 = 20;
SELECT DISTINCT COUNT(부서코드) FROM 직원;

답 : 4

[해설]

CREATE TABLE 부서 (
 부서코드 INT PRIMARY KEY,
 부서명 VARCHAR(20)
);

CREATE TABLE 직원 (
 직원코드 INT PRIMARY KEY,
 부서코드 INT,
 직원명 VARCHAR(20),
 FOREIGN KEY(부서코드) REFERENCES 부서(부서코드)
 ON DELETE CASCADE
);

<부서> 테이블을 생성한다.
 '부서코드' 속성은 정수형이며, 기본키로 정의한다.
 '부서명' 속성은 가변길이 문자 20자이다.

<직원> 테이블을 생성한다.
 '직원코드' 속성은 정수형이며, 기본키로 정의한다.
 '부서코드' 속성은 정수형이다.
 '직원명' 속성은 가변길이 문자 20자이다.
 '부서코드' 속성은 <부서> 테이블의 '부서코드' 속성을 참조하는 외래키이다.
 <부서> 테이블에서 튜플이 삭제되면 관련된 모든 튜플이 함께 삭제된다.

① INSERT INTO 부서 VALUES(10, '영업부');
 ② INSERT INTO 부서 VALUES(20, '기획부');
 ③ INSERT INTO 부서 VALUES(30, '개발부');
 ④ INSERT INTO 직원 VALUES(1001, 10, '이진수');
 ⑤ INSERT INTO 직원 VALUES(1002, 10, '곽연경');
 ⑥ INSERT INTO 직원 VALUES(1003, 20, '김선길');
 ⑦ INSERT INTO 직원 VALUES(1004, 20, '최민수');
 ⑧ INSERT INTO 직원 VALUES(1005, 20, '이용갑');
 ⑨ INSERT INTO 직원 VALUES(1006, 30, '박종일');
 ⑩ INSERT INTO 직원 VALUES(1007, 30, '박미경');

- ①~③번 SQL문이 수행된 후 <부서> 테이블은 다음과 같습니다.

<부서>

부서코드	부서명
10	영업부
20	기획부
30	개발부

- ④~⑩번 SQL문이 수행된 후 <직원> 테이블은 다음과 같습니다.

<직원>

직원코드	부서코드	직원명
1001	10	이진수
1002	10	곽연경
1003	20	김선길
1004	20	최민수
1005	20	이용갑
1006	30	박종일
1007	30	박미경

① SQL문

SELECT DISTINCT COUNT(부서코드)	'부서코드'의 개수를 표시하되, 표시된 개수 중 중복된 값은 한 번만 표시한다.
FROM 직원	<직원> 테이블에서 검색한다.
WHERE 부서코드 = 20;	'부서코드'가 20인 튜플만을 대상으로 한다.

- 문제의 SQL문은 DISTINCT가 '부서코드'에 적용되는 것이 아니라 'COUNT(부서코드)'에 적용됨에 유의해야 합니다.
- WHERE 부서코드 = 20 : '부서코드'가 20인 자료만을 검색합니다.

부서코드
20
20
20

- SELECT DISTINCT COUNT(부서코드) : 'COUNT(부서코드)'의 결과인 3에는 중복된 값이 없으므로 3이 그대로 표시됩니다.

COUNT(부서코드)
3

② SQL문

DELETE FROM 부서	<부서> 테이블에서 튜플을 삭제한다.
WHERE 부서코드 = 20;	'부서코드'가 20인 튜플만을 대상으로 한다.
SELECT DISTINCT COUNT(부서코드)	'부서코드'의 개수를 표시하되, 표시된 개수 중 중복된 값은 한 번만 표시한다.
FROM 직원;	<직원> 테이블에서 검색한다.

- DELETE FROM 부서 WHERE 부서코드 = 20; : <직원> 테이블의 '부서코드'는 <부서> 테이블의 '부서코드'를 참조하고, <부서> 테이블의 '부서코드'가 삭제되면 이를 참조하는 <직원> 테이블의 모든 튜플도 같이 삭제되도록 정의되었으므로, DELETE문 수행 후의 <부서>와 <직원> 테이블은 다음과 같습니다.

<부서>

부서코드	부서명
10	영업부
30	개발부

<직원>

직원코드	부서코드	직원명
1001	10	이진수
1002	10	곽민경
1006	30	박종일
1007	30	박미경

- SELECT DISTINCT COUNT(부서코드) FROM 직원; : <직원> 테이블에 대한 'COUNT(부서코드)'의 결과인 4에는 중복된 값이 없으므로 4가 그대로 표시됩니다.

COUNT(부서코드)
4

2. 학생(STUDENT) 테이블에 전기과 학생이 50명, 전산과 학생이 100명, 전자과 학생이 50명 있다고 할 때, 다음 SQL문 ①, ②, ③의 실행 결과로 표시되는 튜플의 수를 쓰시오. (단, DEPT 필드는 학과를 의미한다.)



- ① SELECT DEPT FROM STUDENT;
 ② SELECT DISTINCT DEPT FROM STUDENT;
 ③ SELECT COUNT(DISTINCT DEPT) FROM STUDENT WHERE DEPT = '전산과';

답

- ① 200
- ② 3
- ③ 1

[해설]

- ①

SELECT DEPT FROM STUDENT;	'DEPT'를 표시한다. <STUDENT> 테이블을 대상으로 검색한다.
------------------------------	--
- <STUDENT> 테이블에서 'DEPT'를 검색합니다.
 - 총 200개의 튜플이 들어 있고 검색 조건이 없으므로 튜플의 수는 200입니다.
- ②

SELECT DISTINCT DEPT FROM STUDENT;	'DEPT'를 표시하되, 같은 'DEPT' 속성의 값은 한 번만 표시한다. <STUDENT> 테이블을 대상으로 검색한다.
---------------------------------------	--
- <STUDENT> 테이블에서 'DEPT'를 검색하는 데 중복된 결과는 처음의 한 개만 검색에 포함시킵니다.
 - 전기과 50개 튜플의 'DEPT' 속성의 값이 같으므로 1개, 전산과 100개 튜플의 'DEPT' 속성의 값이 같으므로 1개, 전자과 50개 튜플의 'DEPT' 속성의 값이 같으므로 1개를 검색에 포함시키므로 3개의 튜플이 검색됩니다.
- ③

SELECT COUNT(DISTINCT DEPT) FROM STUDENT; WHERE DEPT = '전산과';	'DEPT'의 개수를 표시하되, 같은 'DEPT' 속성의 값은 한 번만 계산한다. <STUDENT> 테이블을 대상으로 검색한다. 'DEPT'가 "전산과"인 자료만을 대상으로 검색한다.
---	--
- <STUDENT> 테이블에서 'DEPT' 속성의 값이 '전산과'인 튜플에 대해 중복을 제거하고 개수를 세므로 1이 검색 결과로 표시됩니다.

3. 다음은 <제품>(제품명, 단가, 제조사) 테이블을 대상으로 “H” 제조사에서 생산한 제품들의 ‘단가’보다 높은 ‘단가’를 가진 제품의 정보를 조회하는 <SQL문>이다. 괄호에 알맞은 답을 적어 <SQL문>을 완성하시오.



<SQL문>

```
SELECT 제품명, 단가, 제조사
FROM 제품
WHERE 단가 > (      ) (SELECT 단가 FROM 제품 WHERE 제조사 = 'H');
```

답 : ALL

[해설]

- | | |
|---|---|
| <p>② SELECT 제품명, 단가, 제조사
FROM 제품
WHERE 단가 > ALL (</p> <p>① SELECT 단가
FROM 제품
WHERE 제조사 = 'H');</p> | <p>‘제품명’, ‘단가’, ‘제조사’ 속성을 표시한다.
<제품> 테이블에서 검색한다.
‘단가’가 하위 질의로 검색된 모든(ALL) 단가보다 큰 자료만을 대상으로 한다.
‘단가’를 표시한다.
<제품> 테이블에서 검색한다.
제조사가 “H”인 자료만을 대상으로 한다.</p> |
|---|---|

- 문제의 질의문은 하위 질의가 있는 질의문입니다.
- 먼저 WHERE 조건에 지정된 하위 질의의 SELECT문을 해석한 다음 그 결과를 본 질의의 조건에 있는 ‘단가’ 속성과 비교합니다.
- <제품> 테이블에 다음과 같은 자료가 들어있다고 가정하여 설명합니다.

<제품>

제품명	단가	제조사
냉장고	200	H
TV	150	H
세탁기	300	H
건조기	250	A
핸드폰	400	B
컴퓨터	500	C

- ① <제품> 테이블에서 ‘제조사’ 속성의 값이 “H”인 튜플의 ‘단가’ 속성의 값을 검색합니다.

단가
200
150
300

- ② <제품> 테이블에서 ‘단가’ 속성의 값이 ①번에서 검색된 모든 단가보다 큰 자료를 대상으로 ‘제품명’, ‘단가’, ‘제조사’를 표시합니다.

제품명	단가	제조사
핸드폰	400	B
컴퓨터	500	C

4. 다음 <TABLE>을 참조하여 <SQL문>을 실행했을 때 출력되는 결과를 쓰시오. (<TABLE>에 표시된 'NULL'은 값이 없음을 의미한다.)



<TABLE>

INDEX	COL1	COL2
1	2	NULL
2	4	6
3	3	5
4	6	3
5	NULL	3

<SQL문>

```
SELECT COUNT(COL2)
FROM TABLE
WHERE COL1 IN (2, 3)
      OR COL2 IN (3, 5);
```

답 : 3

[해설]

```
SELECT COUNT(COL2)
FROM TABLE
WHERE COL1 IN (2, 3)
      OR COL2 IN (3, 5);
```

'COL2'의 개수를 표시한다.
<TABLE>에서 검색한다.
'COL1'이 2 또는 3이거나,
'COL2'가 3 또는 5인 튜플만을 대상으로 한다.

• 질의문의 조건을 만족하는 튜플은 다음과 같습니다.

INDEX	COL1	COL2
1	2	NULL
2	4	6
3	3	5
4	6	3
5	NULL	3

• 조건에 맞는 'COL2' 속성만 추출하면 다음과 같습니다.

COL2
NULL
5
3
3

• COUNT(COL2)는 'COL2' 필드의 개수를 계산하지만 'NULL' 값은 제외하므로 COUNT(COL2)의 결과는 3입니다.

5. 다음은 <EMPLOYEE> 릴레이션에 대해 <관계 대수식>을 수행했을 때 출력되는 <결과>이다. <결과>의 각 괄호(①~⑤)에 들어갈 알맞은 답을 쓰시오.



<관계 대수식>

$\pi_{TTL}(EMPLOYEE)$

<EMPLOYEE>

INDEX	AGE	TTL
1	48	부장
2	25	대리
3	41	과장
4	36	차장



<결과>

(①)
(②)
(③)
(④)
(⑤)

답

- ① TTL
- ② 부장
- ③ 대리
- ④ 과장
- ⑤ 차장

[해설]

문제의 <관계 대수식>에서 사용된 π 는 주어진 릴레이션에서 속성 리스트(Attribute List)에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 PROJECT 연산이므로, <EMPLOYEE> 릴레이션에서 'TTL' 속성이 추출되어 속성명인 'TTL'부터 모든 속성값이 <결과>로 나타납니다.



6. SQL과 관련한 다음 설명에서 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오.

UPDATE문은 테이블에 있는 튜플의 내용을 갱신할 때 사용하는 명령문으로, DML에 해당한다. 다른 DML로는 INSERT, DELETE가 있으며, 각각 새로운 튜플을 삽입하거나 삭제할 때 사용한다.

<학부생> 테이블

학부	학과번호	입학생수	담당관
정경대학	110	300	김해울
공과대학	310	250	이성관
인문대학	120	400	김해울
정경대학	120	300	김성수
인문대학	420	180	이윤해

다음은 <학부생> 테이블에서 '입학생수'가 300 이상인 튜플의 '학과번호'를 999로 갱신하는 SQL문이다.

(①) 학부생 (②) 학과번호 = 999 WHERE 입학생수 >= 300;

답

- ① UPDATE
- ② SET

[해설]

- SQL문

UPDATE 학부생	'학부생' 테이블을 갱신하라.
SET 학과번호 = 999	'학과번호'를 999로 갱신하라.
WHERE 입학생수 >= 300;	'입학생수'가 300이상인 튜플만을 대상으로 하라.

- SQL 실행 결과

학부	학과번호	입학생수	담당관
정경대학	999	300	김해울
공과대학	310	250	이성관
인문대학	999	400	김해울
정경대학	999	300	김성수
인문대학	420	180	이윤해



7. <EMP_TBL> 테이블을 참고하여 <SQL문>의 실행 결과를 쓰시오.

<EMP_TBL>

EMPNO	SAL
100	1500
200	3000
300	2000

<SQL문>

```
SELECT COUNT(*) FROM EMP_TBL WHERE EMPNO > 100 AND SAL >= 3000 OR EMPNO = 200;
```

답 : 1

[해설]

SQL도 프로그래밍 언어와 마찬가지로 OR 연산자에 비해 AND 연산자의 우선순위가 높다. 즉 식1 AND 식2 OR 식3과 같이 조건이 제시된 경우 식1 AND 식2의 조건을 먼저 확인한 후 그 결과와 식3의 OR 조건을 확인해야 합니다.

```
SELECT COUNT(*)
FROM EMP_TBL
WHERE EMPNO > 100
      AND SAL >= 3000
      OR EMPNO = 200;
```

튜플의 개수를 표시한다.

<EMP_TBL> 테이블에서 검색한다.

'EMPNO'가 100보다 크고

'SAL'이 3000 이상이거나,

'EMPNO'가 200인 튜플만을 대상으로 한다.

<과정>

① 'EMPNO'가 100보다 큰 튜플은 다음과 같습니다.

EMPNO	SAL
200	3000
300	2000

② 'SAL'이 3000 이상인 튜플은 다음과 같습니다.

EMPNO	SAL
200	3000

③ ①, ②의 조건을 동시에 만족(AND)하는 튜플은 다음과 같습니다.

EMPNO	SAL
200	3000

④ 'EMPNO'가 200인 튜플은 다음과 같습니다.

EMPNO	SAL
200	3000

⑤ ③번 또는 ④번의 튜플 중 한 번이라도 포함된(OR) 튜플은 다음과 같습니다.

EMPNO	SAL
200	3000

⑥ COUNT(*) 함수에 따라 ⑤번 튜플의 개수를 표시하면 다음과 같습니다.

COUNT(*)
1

8. <학생> 테이블에서 '이름'이 "민수"인 튜플을 삭제하고자 한다. 다음 <처리 조건>을 참고하여 SQL문을 작성하시오.



<처리 조건>

- 명령문 마지막의 세미콜론(:)은 생략이 가능하다.
- 인용 부호가 필요한 경우 작은따옴표(' ')를 사용한다.

답 : DELETE FROM 학생 WHERE 이름 = '민수';

[해설]

DELETE	삭제하라.
FROM 학생	<학생> 테이블을 대상으로 하라.
WHERE 이름 = '민수';	'이름'이 "민수"인 자료만을 대상으로 한다.

9. 다음 <속성 정의서>를 참고하여 <학생> 테이블에 대해 20자의 가변 길이를 가진 '주소' 속성을 추가하는 <SQL문>을 완성하시오. (단, SQL문은 ISO/IEC 9075 표준을 기반으로 작성하시오.)



<속성 정의서>

속성명	데이터타입	제약조건	테이블명
학번	CHAR(10)	UNIQUE	학생
이름	VARCAHR(8)	NOT NULL	학생
주민번호	CHAR(13)		학생
학과	VARCAHR(16)	FOREIGN KEY	학생
학년	INT		학생

<SQL문>

(①) TABLE 학생 (②) 주소 VARCHAR(20);

답

- ① ALTER
- ② ADD

[해설]

ALTER TABLE 학생
ADD 주소 VARCHAR(20);

수정할 테이블의 이름은 <학생>이다.
가변 길이의 문자 20자리인 '주소' 속성을 추가한다.



10. 다음 <학생> 테이블을 참고하여 <처리 조건>에서 요구하는 SQL문을 작성하시오.

<학생>

학번 (varchar)	이름 (varchar)	학년 (number)	수강과목 (varchar)	점수 (number)	연락처 (varchar)
20E0232	김인영	3	세무행정	4.5	010-5412-4544
19D0024	이성화	2	토목개론	3	010-1548-4796
20E0135	성유수	4	실용법학	3.5	010-9945-7411
20E0511	우인혁	1	데이터론	2	010-3451-4972

<처리 조건>

- 3, 4학년의 학번, 이름을 조회한다.
- IN 예약어를 사용해야 한다.
- 속성명 아래의 괄호는 속성의 자료형을 의미한다.

답 : **SELECT 학번, 이름 FROM 학생 WHERE 학년 IN (3, 4);**

[해설]

SELECT 학번, 이름
FROM 학생
WHERE 학년 IN (3, 4);

'학번', '이름'을 표시한다.

<학생> 테이블에서 검색한다.

'학년'의 값이 3 또는 4인 자료만을 대상으로 한다.

<결과>

학번	이름
20E0232	김인영
20E0135	성유수

11. 다음 <student> 테이블을 참고하여 'name' 속성으로 'idx_name'이라는 인덱스를 생성하는 SQL문을 작성하시오.



<student>

stid	name	score	deptid
2001	brown	85	PE01
2002	white	45	EF03
2003	black	67	UW11

답 : **CREATE INDEX idx_name ON student(name);**

[해설]

CREATE INDEX idx_name	'idx_name'이라는 이름의 인덱스를 생성한다.
ON student(name);	<student> 테이블의 'name' 속성을 사용한다.

2023
시나공

[정렬]

12. 다음은 <성적> 테이블에서 이름(name)과 점수(score)를 조회하되, 점수를 기준으로 내림차순 정렬하여 조회하는 <SQL문>이다. 괄호(①~③)에 알맞은 답을 적어 <SQL문>을 완성하시오.



<성적>

name	class	score
정기찬	A	85
이영호	C	74
환정형	C	95
김지수	A	90
최은영	B	82

<SQL문>

```
SELECT name, score  
FROM 성적  
( ① ) BY ( ② ) ( ③ )
```

답

- ① ORDER
- ② score
- ③ DESC

[해설]

SELECT name, score	'name'과 'score'를 표시한다.
FROM 성적	<성적> 테이블에서 검색한다.
ORDER BY score DESC	'score'를 기준으로 내림차순 정렬한다.

[그룹]

13. 다음은 <회원> 테이블에서 '이름'이 "이"로 시작하는 회원들을 '가입일' 순으로 내림차순 정렬하는 <SQL문>이다. 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오.



<회원> 테이블

회원번호	이름	성별	가입일
1001	이진성	남	2021-06-23
1002	조이령	여	2021-06-24
1003	최민수	남	2021-06-28
1004	김차희	여	2021-07-03
1005	이미경	여	2021-07-10

<SQL문>

```
SELECT * FROM 회원 WHERE 이름 LIKE '( )' ORDER BY 가입일 ( );
```

답

- ① 이%
- ② DESC

[해설]

• SQL문

SELECT *	모든 속성을 표시한다.
FROM 회원	<회원> 테이블에서 검색한다.
WHERE 이름 LIKE '이%'	'이름'이 '이'로 시작하는 튜플만을 대상으로 한다.
ORDER BY 가입일 DESC;	'가입일'을 기준으로 내림차순 정렬한다.

• SQL 실행 결과

회원번호	이름	성별	가입일
1005	이미경	여	2021-07-10
1001	이진성	남	2021-06-23

14. 다음 질의 내용에 대한 SQL문을 완성하시오.



질의	학생 테이블에서 학과별 튜플의 개수를 검색하시오. (단, 아래의 실행 결과가 되도록 한다.)
----	--

<학생>

학번	이름	학년	학과	주소
20160011	김영란	2	전기	서울
19210113	이재우	3	컴퓨터	대구
21168007	함소진	1	전자	부산
19168002	우혜정	3	전자	광주
18120073	김진수	4	컴퓨터	울산

<실행결과>

학과	학과별튜플수
전기	1
컴퓨터	2
전자	2

<처리 조건>

- WHERE 조건절은 사용할 수 없다.
- GROUP BY는 반드시 포함한다.
- 집계함수(Aggregation Function)를 적용한다.
- 학과별튜플수 컬럼이름 출력에 Alias(AS)를 활용한다.
- 문장 끝의 세미콜론(:)은 생략해도 무방하다.
- 인용부호 사용이 필요한 경우 단일 따옴표(' : Single Quotation)를 사용한다.

답 : SELECT 학과, COUNT(*) AS 학과별튜플수 FROM 학생 GROUP BY 학과;

[해설]

SELECT 학과, COUNT(*) AS 학과별튜플수	'학과'와 개수를 표시하되, 개수의 필드명을 '학과별튜플수'로 표시한다.
FROM 학생	<학생> 테이블을 대상으로 검색한다.
GROUP BY 학과;	'학과'를 기준으로 그룹을 지정한다.

15. 다음의 <성적> 테이블에서 과목별 점수의 평균이 90점 이상인 '과목이름', '최소점수', '최대점수'를 검색하고자 한다. <처리 조건>을 참고하여 적합한 SQL문을 작성하시오.



<성적>

학번	과목번호	과목이름	학점	점수
a2001	101	컴퓨터구조	6	95
a2002	101	컴퓨터구조	6	84
a2003	302	데이터베이스	5	89
a2004	201	인공지능	5	92
a2005	302	데이터베이스	5	100
a2006	302	데이터베이스	5	88
a2007	201	인공지능	5	93

<결과>

과목이름	최소점수	최대점수
데이터베이스	88	100
인공지능	92	93

<처리 조건>

- WHERE문은 사용하지 않는다.
- GROUP BY와 HAVING을 이용한다.
- 집계함수(Aggregation Function)를 사용하여 명령문을 구성한다.
- '최소점수', '최대점수'는 별칭(Alias)을 위한 AS문을 이용한다.
- 명령문 마지막의 세미콜론(;)은 생략이 가능하다.
- 인용 부호가 필요한 경우 작은따옴표(' ')를 사용한다.

답 : **SELECT 과목이름, MIN(점수) AS 최소점수, MAX(점수) AS 최대점수 FROM 성적 GROUP BY 과목이름 HAVING AVG(점수) >= 90;**

[해설]

- 1 SELECT 과목이름, MIN(점수) AS 최소점수, MAX(점수) AS 최대점수
- 2 FROM 성적
- 3 GROUP BY 과목이름
- 4 HAVING AVG(점수) >= 90;

- 1 '과목이름', '점수'의 최소값, '점수'의 최대값을 표시하되, '점수'의 최소값은 '최소점수'로, '점수'의 최대값은 '최대점수'로 표시한다.
- 2 <성적> 테이블을 대상으로 검색한다.
- 3 '과목이름'을 기준으로 그룹을 지정한다.
- 4 각 그룹별 '점수'의 평균이 90보다 크거나 같은 그룹만을 표시한다.

[조인]



16. <A> 테이블과 테이블을 참고하여 <SQL문>의 실행 결과를 쓰시오.

<A>

NAME
Smith
Allen
Scott

RULE
S%
%T%

<SQL문>

```
SELECT COUNT(*) CNT FROM A CROSS JOIN B WHERE A.NAME LIKE B.RULE;
```

답 : 4

[해설]

```
SELECT COUNT(*) CNT
FROM A CROSS JOIN B
WHERE A.NAME LIKE B.RULE;
```

질의문은 각 절을 분리하여 이해하면 쉽습니다.

- **SELECT COUNT(*) CNT** : 튜플의 개수를 표시하되, 필드명은 'CNT'로 표시합니다.
※ 'SELECT COUNT(*) AS CNT'에서 AS가 생략된 형태입니다.
- **FROM A CROSS JOIN B** : <A>와 를 교차 조인(CROSS JOIN)한 결과를 대상으로 검색합니다.

A.NAME	B.RULE
Smith	S%
Smith	%T%
Allen	S%
Allen	%T%
Scott	S%
Scott	%T%

- **WHERE A.NAME LIKE B.RULE** : <A> 테이블의 'NAME' 필드 값이 테이블의 'RULE' 필드에 저장된 문자열 패턴과 일치하는 튜플만을 대상으로 합니다.

※ 테이블의 'RULE' 필드에 저장된 값은 'S%'와 '% t %'와 같이 문자 패턴인 '%' 기호가 포함되어 있으므로, 조건문의 LIKE 연산자와 결합되면 다음과 같이 적용됩니다.

- **A.NAME LIKE S%** : 'A.NAME'이 "S"로 시작하는 레코드를 검색

NAME	RULE
Smith	S%
Smith	%T%
Allen	S%
Allen	%T%
Scott	S%
Scott	%T%

- A.NAME LIKE %T% : 'A.NAME'이 "T"를 포함하는 레코드를 검색

NAME	RULE
Smith	S%
Smith	%T%
Allen	S%
Allen	%T%
Scott	S%
Scott	%T%

※ CROSS JOIN된 결과에서 조건을 만족하는 튜플은 다음과 같습니다. 그러므로 검색된 튜플의 개수는 4입니다.

NAME	RULE
Smith	S%
Smith	%T%
Scott	S%
Scott	%T%

2023
시나공

17. 다음 <사원> 테이블과 <동아리> 테이블을 조인(Join)한 <결과>를 확인하여 <SQL문>의 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오.



<사원>

코드	이름	부서
1601	김명해	인사
1602	이진성	경영지원
1731	박영광	개발
2001	이수진	

<동아리>

코드	동아리명
1601	테니스
1731	탁구
2001	볼링



<결과>

코드	이름	동아리명
1601	김명해	테니스
1602	이진성	
1731	박영광	탁구
2001	이수진	볼링

<SQL문>

SELECT a.코드, 이름, 동아리명 FROM 사원 a LEFT JOIN 동아리 b (①) a.코드 = b.(②);

답

- ① ON
- ② 코드

[해설]

- ① SELECT a.코드, 이름, 동아리명
- ② FROM 사원 a LEFT JOIN 동아리 b
- ③ ON a.코드 = b.코드;

- ① a가 가리키는 <사원> 테이블의 '코드'와 '이름', '동아리명'을 표시한다.
- ② • LEFT JOIN이므로, 좌측의 <사원> 테이블이 기준이 되어 <사원> 테이블에 있는 튜플은 모두 표시하고, 우측의 <동아리> 테이블에서는 관련이 있는 튜플만 표시한다.
• <사원>, <동아리> 테이블의 별칭으로 <a>, 를 지정한다. <a>는 <사원> 테이블을, 는 <동아리> 테이블을 가리키게 된다.
- ③ <사원> 테이블의 '코드'와 <동아리> 테이블의 '코드'를 기준으로 서로 JOIN한다.