연습문제

※ 1절에서 사용한 마당서점 데이터베이스를 비탕으로 다음 1~5번 문제에 답하시오. 마당서점의 초기 자료로 실습하고 싶으면 madang 사용자로 로그인한 후 demo_madang_init.sql을 실행시켜 모든 자료를 초기화하면 된다. 스크립트를 실행하려면 F3 키를 누르거나 스크립트 실행(▷) 이이콘을 클릭하면 된다.

01 마당서점의 고객이 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

(1) 도서번호가 1인 도서의 이름
```
SELECT 도서이름 FROM 도서 WHERE 도서번호='1'
```

(2) 가격이 20,000원 이상인 도서의 이름
```
SELECT 도서이름 FROM 도서 WHERE 가격 >=20000;
```

(3) 박지성이 총구매액
```
SELECT SUM(수량*판매가격) AS 총구매액
FROM 주문 JOIN 고객 ON 주문.고객번호 = 고객.고객번호 WHERE 고객.이름 ='박지성';
```

(4) 박지성이 구매한 도서의 수
```
SELECT COUNT(DISTINCT 도서번호) AS 구매도서수
FROM 주문 JOIN 고객 ON 주문.고객번호 = 고객.고객번호 WHERE 고객.이름 = '박지성';
```

(5) 박지성이 구매한 도서의 출판사 수
```
SELECT COUNT(DISTINCT 출판사)
FROM 주문 JOIN 고객 ON 주문.고객번호 = 고객.고객번호 JOIN 도서 ON 주문.도서번호 = 도서.도서번호 WHERE 고객.이름='박지성';
```

(6) 박지성이 구매한 도서의 이름, 가격, 정가와 판매가격의 차이
```
SELECT 도서.도서이름, 도서.정가, 주문.판매가격, (도서.정가-주문.판매가격) AS 가격차이
FROM 주문
JOIN 고객 ON 주문.고객번호 = 고객.고객번호
JOIN 도서 ON 주문.도서번호 = 도서.도서번호
WHERE 고객.이름
```

(7) 박지성이 구매하지 않은 도서의 이름
```
SELECT 도서이름
FROM 도서
WHERE 도서번호 NOT IN (
SELECT 도서번호 FROM 주문 JOIN 고객 ON 주문.고객번호 = 고객.고객번호 WHERE 고객.이름 = '박지성');
```

02 마당서점의 운영자와 경영자가 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

(1) 마당서점 도서의 총개수
```
SELECT COUNT(*) AS 도서수
FROM 도서;
```

(2) 마당서점에 도서를 출고하는 출판사의 총개수
```
SELECT COUNT(DISTINCT 출판사) AS 출판사수
FROM 도서;
```

(3) 모든 고객의 이름, 주소
```
SELECT 이름, 주소
FROM 고객;
```

(4) 2024년 7월 4일~7월 7일 사이에 주문받은 도서의 주문번호
```
SELECT 주문번호
FROM 주문 WHERE 주문일자 BETWEEN '2024-07-04' AND '2024-07-07';
```

(5) 2024년 7월 4일~7월 7일 사이에 주문받은 도서를 제외한 도서의 주문번호
```
SELECT 주문번호
FROM 주문
WHERE 주문일자 NOT BETWEEN '2024-07-04' AND '2024-07-07';
```

(6) 성이 '김' 씨인 고객의 이름과 주소
```
SELECT 이름, 주소
FROM 고객
WHERE 이름 LIKE '김%';
```

(7) 성이 '김' 씨이고 이름이 '아'로 끝나는 고객의 이름과 주소
```
SELECT 이름, 주소
FROM 고객
WHERE 이름 NOT LIKE '%아%';
```

(8) 주문하지 않은 고객의 이름(부속질의 사용)
```
SELECT 이름
FROM 고객
WHERE 고객번호 NOT IN (SELECT DISTINCT 고객번호 FROM 주문);
```

(9) 주문 금액의 총액과 주문의 평균 금액
```
SELECT SUM (수량 * 판매가격) AS 총액, AVG (수량 * 판매가격) AS 평균
FROM 주문;
```

(10) 고객의 이름과 고객별 구매액
```
SELECT 고객.이름, SUM (수량*판매가격)AS 구매액
FROM 주문 JOIN 고객 ON 주문.고객번호 = 고객.고객번호 GROUP BY 고객.이름;
```

(11) 고객의 이름과 고객이 구매한 도서 목록
```
SELECT 고객.이름, 도서.도서이름
FROM 주문
JOIN 고객 ON 주문.고객번호 = 고객.고객번호 JOIN 도서 ON 주문.도서번호 = 도서.도서번호;
```

(12) 도서의 가격(Book 테이블)과 판매가격(Orders 테이블)의 차이가 가장 많은 주문
```
SELECT 주문번호, MAX(도서.가격 - 주문.판매가격)AS 차이
FROM 주문
JOIN 도서 ON 주문.도서번호 = 도서.도서번호;
```

(13) 도서의 판매액 평균보다 자신의 구매액 평균이 더 높은 고객의 이름
```
SELECT 고객.이름
FROM 고객평균
JOIN 전체평균
JOIN 고객 ON 고객.고객번호 = 고객평균.고객번호
WHERE 고객평균.avg_customer > 전체평균.avg_all;
```


연습문제

03 다음 심화 질문에 대해 SQL 문을 작성하시오.

(1) 박지성이 구매한 도서의 출판사와 같은 출판사에서 도서를 구매한 고객의 이름
```
SELECT DISTINCT c2.name
FROM Customer c1
JOIN BUY b1 ON c1.custid=b1.custid
JOIN BOOK bk1 ON b1.bookid =bk1.bookid
JOIN BOOK bk2 ON bk1.publisher =bk2.publisher
JOIN Buy b2 ON bk2.bookid = b2.bookid
JOIN Customer c2 ON b2.custid=c2.custid
WHERE c1.name='박지성' AND c2.name !='박지성';
```

(2) 두 개 이상의 서로 다른 출판사에서 도서를 구매한 고객의 이름
```
SELECT c.name
FROM Customer c
JOIN Buy b ON c.custid = b.custid
JOIN book bk ON b.bookid = bk.bookid
GROUP BY c.name
HAVING COUNT(DISTINCT bk.publisher) >=2;
```

(3) 전체 고객의 30% 이상이 구매한 도서
```
SELECT bk.bookname
FROM Buy b
JOIN Book bk ON b.bookid=bk.bookid
GROUP BY bk.bookid, bk.bookname
HAVING COUNT(DISTINCT b.custid)
>= (SELECT COUNT(*) *0.3 FROM Customer);
```

04 다음 질의에 대해 DDL 문과 DML 문을 작성하시오.

(1) 새로운 도서 ('스포츠 세계', '대한미디어', 10000원)이 마당서점에 입고되었다. 삽입이 안 될 경우 필요한 데이터가 더 있는지 찾아보시오.
```
INSERT INTO Book( bookname, publisher, price)
VALUES ('스포츠 세계', '대한미디어', 10000);
```

(2) '삼성당'에서 출판한 도서를 삭제하시오.
```
DELETE FROM Book
WHERE publisher = '삼성당';
```

(3) '이상미디어'에서 출판한 도서를 삭제하시오. 삭제가 안 되면 원인을 생각해 보시오.
```
DELETE FROM Book
WHERE publisher = '이상미디어';
```
SELECT bookid FROM Book WHERE publisher = '이상미디어'
BOOK 테이블에서 남아있을 가능성

(4) 출판사 '대한미디어'를 '대한출판사'로 이름을 바꾸시오.
```
UPDATE Book
SET publisher = '대한출판사'
WHERE publisher = '이상미디어';
```

(5) (테이블 생성) 출판사에 대한 정보를 저장하는 테이블 Bookcompany(name, address, begin)을 생성하고자 한다. name은 기본키이며 VARCHAR(20), address는 VARCHAR(20), begin은 DATE 타입으로 선언하여 생성하시오.
```
CREATE TABLE Bookcompany (
name VARCHAR(20) PRIMARY KEY,
address VARCHAR(20),
begin DATE
);
```

(6) (테이블 수정) Bookcompany 테이블에 인터넷 주소를 저장하는 webaddress 속성을 VARCHAR(30)으로 추가하시오.
```
ALTER TABLE Bookcompany
ADD webaddress VARCHAR(30);
```

(7) Bookcompany 테이블에 임의의 투플 name=한빛아카데미, address=서울시 마포구, begin=1993-01-01, webaddress=http://hanbit.co.kr을 삽입하시오.
```
INSERT INTO Bookcompany (name, address, begin, webaddress)
VALUES ('한빛아카데미', '서울시마포구', '1993-01-01', 'http://hanbit.co.kr');
```

05 다음 EXISTS 질의의 결과를 나타내시오.

```
SELECT   *
FROM     Customer c1
WHERE    NOT EXISTS (SELECT   *
                     FROM     Orders c2
                     WHERE    c1.custid=c2.custid);
```

주문에 기록이 없는 고객의 정보를 조회

(1) 질의의 결과는 무엇인가?

(2) NOT을 지우면 질의의 결과는 어떻게 되는가? 주문을 한 적이 있는 고객의 정보가 출력

06 다음 SQL 명령문들에 대하여 DML, DDL, DCL을 구분하시오.

CREATE, ALTER, DROP, SELECT, INSERT, DELETE, UPDATE, GRANT, REVOKE

DML : SELECT,INSERT,DELETE,UPDATE
DDL : CREATE, ALTER, DROP
DCL : GRANT, REVOKE

07 릴레이션 R(A, B, C)과 S(C, D, E)에 대해, 관계대수를 의미가 같은 SQL 문으로 변환하시오.

(1) $\sigma_{A=a2}$ (R)

```
SELECT *
FROM R
WHERE A = B;
```

(2) $\pi_{A, B}$ (R)

```
SELECT A, B
FROM R;
```

(3) R $\bowtie_{R.c=S.c}$ S

```
SELECT *
FROM R
JOIN S ON R.C = S.C;
```

08 릴레이션 R(A, B, C)과 S(C, D, E)에 대해, 내부조인과 외부조인들에 대한 SQL 문의 결과를 작성하시오(FULL OUTER JOIN은 SQL 표준 문법에는 있지만 MySQL은 지원하지 않는다).

R

A	B	C
a1	b1	c1
a2	b1	c1
a3	b1	c2
a4	b2	c4

S

C	D	E
c1	d2	e1
c1	d1	e2
c2	d3	e3
c5	d3	e3

(1) SELECT * FROM R INNER JOIN S ON (R.C = S.C)

a1 b1 c1 c1 d2 e1
a1 b1 c1 c1 d1 e2
a2 b1 c1 c1 d2 e1
a2 b1 c1 c1 d1 e2

(2) SELECT * FROM R LEFT OUTER JOIN S ON (R.C = S.C)

a1 b1 c1 c1 d2 e1 a2 b1 c1 c1 d1 e2
a1 b1 c1 c1 d1 e2 a3 b2 c3 NULL NULL NULL
a2 b1 c1 c1 d2 e1 a4 b2 c4 NULL NULL NULL

(3) SELECT * FROM R RIGHT OUTER JOIN S ON (R.C = S.C)

a1 b1 c1 c1 d2 e1 a2 b1 c1 c1 d1 e2
a2 b1 c1 c1 d2 e1 NULL NULL NULL c2 d3 e3
a1 b1 c1 c1 d1 e2 NULL NULL NULL c5 d3 e3

(4) SELECT * FROM R FULL OUTER JOIN S ON (R.C = S.C)

a1 b1 c1 c1 d2 e1 a1 b1 c1 c1 d1 e2
a2 b1 c1 c1 d2 e1 a2 b1 c1 c1 d1 e2
a3 b2 c3 NULL NULL NULL a4 b2 c4 NULL NULL NULL
NULL NULL NULL c2 d3 e3 NULL NULL NULL c5 d3 e3

(5) SELECT * FROM R CROSS JOIN S 16개

09 릴레이션 R(col1: VARCHAR(20), col2: INT)에 대해, 다음 SQL 구문을 순서대로 수행하였을 때 마지막 SQL의 수행 결과를 적으시오. 20

```
INSERT INTO R(col1, col2) VALUES('ABCD', NULL);
INSERT INTO R(col1, col2) VALUES('BC', NULL);
ALTER TABLE R MODIFY col2 INT DEFAULT 10;
INSERT INTO R(col1, col2) VALUES('XY', NULL);
INSERT INTO R(col1) VALUES('EFG');
SELECT SUM(col2) FROM R;
```

10 릴레이션 R1, R2, R3에 대해 다음 SQL 문을 수행하였을 때 반환되는 행의 수는 얼마인가? 8개

```
(SQL)
SELECT a FROM R1; (결과) => 1, 2, 3, 4, 5, 6
SELECT a FROM R2; (결과) => 3, 7, 8
SELECT a FROM R3; (결과) => 4, 5, 6

SELECT a FROM R1
UNION ALL
SELECT a FROM R2
UNION
SELECT a FROM R3;
```

11 테이블 R1(a, b), R2(a, b)를 생성하고 데이터 삽입과 삭제를 수행하였다. 마지막 SELECT 문의 수행 결과를 적으시오. 자식 테이블에 대한 참조무결성 제약 중 ON DELETE CASCADE 조건의 의미를 확인하여 결과를 구해본다. 2,2

```
(SQL)
CREATE TABLE R1(
      a INTEGER PRIMARY KEY,
      b INTEGER
);
CREATE TABLE R2(
      a INTEGER,
      b INTEGER,
      FOREIGN KEY (b) REFERENCES R1 (a) ON DELETE CASCADE);
INSERT INTO R1 VALUES(1,1);
INSERT INTO R1 VALUES(2,2);
INSERT INTO R2 VALUES(1,1);
INSERT INTO R2 VALUES(2,2);
DELETE FROM R1 WHERE a=1;
SELECT * FROM R2;
```