3장(SQL 기초) 연습문제 해답

1. 마당서점의 고객이 요구하는 다음 질문에 대해 SQL 문을 작성하시오.

(1) 도서번호가 1인 도서의 이름

```
SELECT bookname FROM Book WHERE bookid=1;
```

(2) 가격이 20,000원 이상인 도서의 이름

```
SELECT bookname FROM Book WHERE price >= 20000;
```

(3) 박지성의 총 구매액

```
SELECT SUM(saleprice)
FROM Customer, Orders
WHERE Customer.custid=Orders.custid
        AND Customer.name LIKE '박지성';
```

(4) 박지성이 구매한 도서의 수

```
SELECT COUNT(*) FROM Customer, Orders
WHERE Customer.custid=Orders.custid
        AND Customer.name LIKE '박지성';
```

(5) 박지성이 구매한 도서의 출판사 수

```
SELECT COUNT(DISTINCT publisher)
FROM Customer, Orders, Book
WHERE Customer.custid=Orders.custid AND Orders.bookid=Book.bookid
        AND Customer.name LIKE '박지성';
```

(6) 박지성이 구매한 도서의 이름, 가격, 정가와 판매가격의 차이

```
SELECT bookname, price, price-saleprice
FROM Customer, Orders, Book
WHERE Customer.custid=Orders.custid AND Orders.bookid=Book.bookid
        AND Customer.name LIKE '박지성';
```

(7) 박지성이 구매하지 않은 도서의 이름

```
SELECT bookname FROM Book b1
WHERE NOT EXISTS
    (SELECT bookname FROM Customer, Orders
     WHERE Customer.custid=Orders.custid AND Orders.bookid=b1.bookid
            AND Customer.name LIKE '박지성');
```

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

(1) 마당서점 도서의 총 개수

```
SELECT count(*) FROM Book;
```

(2) 마당서점에 도서를 출고하는 출판사의 총 개수

```
SELECT COUNT(DISTINCT publisher)
FROM Book
```

(3) 모든 고객의 이름, 주소

```
SELECT name, address
FROM Customer
```

(4) 2014년 7월 4일~7월 7일 사이에 주문받은 도서의 주문번호

```
SELECT *
FROM Orders
WHERE orderdate BETWEEN '20140704' AND '20140707'
```

(5) 2014년 7월 4일~7월 7일 사이에 주문받은 도서를 제외한 도서의 주문번호

```
SELECT *
FROM Orders
WHERE orderdate NOT BETWEEN '20140704' AND '20140707'
```

(6) 성이 '김' 씨인 고객의 이름과 주소

```
SELECT name, address
FROM Customer
WHERE name LIKE '김%'
```

(7) 성이 '김' 씨이고 이름이 '아'로 끝나는 고객의 이름과 주소

```
SELECT name, address
FROM Customer
WHERE name LIKE '김%아'
```

(8) 주문하지 않은 고객의 이름(부속질의 사용)

```
SELECT name FROM Customer
WHERE name NOT IN
        (SELECT name
        FROM Customer, Orders
        WHERE Customer.custid=Orders.custid);
```

(9) 주문 금액의 총액과 주문의 평균 금액

```
SELECT SUM(saleprice), AVG(saleprice)
FROM Orders
```

(10) 고객의 이름과 고객별 구매액

```
SELECT name, SUM(saleprice)
FROM Customer, Orders
WHERE Customer.custid=Orders.custid
GROUP BY name;
```

(11) 고객의 이름과 고객이 구매한 도서 목록

```
SELECT name, Book.bookname
FROM Customer, Orders, Book
WHERE Customer.custid=Orders.custid
        AND Orders.bookid=Book.bookid
```

(12) 도서의 가격(Book 테이블)과 판매가격(Orders 테이블)의 차이가 가장 많은 주문

```
SELECT *
FROM Book, Orders
WHERE Book.bookid=Orders.bookid
```

```
        AND price-saleprice=
          (SELECT MAX(price-saleprice)
           FROM Book, Orders
           WHERE Book.bookid=Orders.bookid);
```

(13) 도서의 판매액 평균보다 자신의 구매액 평균이 더 높은 고객의 이름

```
SELECT name, AVG(saleprice)
FROM Customer, Orders
WHERE Customer.custid=Orders.custid
GROUP BY name
HAVING AVG(saleprice) >
          (SELECT AVG(saleprice) FROM Orders);
```

3. 마당서점에서 다음의 심화된 질문에 대해 SQL 문을 작성하시오.

(1) 박지성이 구매한 도서의 출판사와 같은 출판사에서 도서를 구매한 고객의 이름

```
SELECT name FROM Customer, Orders, Book
WHERE Customer.custid=Orders.custid
      AND Orders.bookid=Book.bookid AND name NOT LIKE '박지성'
        AND publisher IN
        (SELECT publisher FROM Customer, Orders, Book
         WHERE Customer.custid=Orders.custid
                AND Orders.bookid=Book.bookid
                AND name LIKE '박지성');
```

(2) 두 개 이상의 서로 다른 출판사에서 도서를 구매한 고객의 이름

```
SELECT name FROM Customer c1
WHERE 2 >=
          (SELECT COUNT(DISTINCT publisher) FROM Customer, Orders, Book
           WHERE Customer.custid=Orders.custid
               AND Orders.bookid=Book.bookid AND (name LIKE c1.name));
```

(3) 전체 고객의 30% 이상이 구매한 도서

```
SELECT bookname FROM Book b1
WHERE  ( (SELECT COUNT(Book.bookid) FROM Book, Orders
             WHERE Book.bookid=Orders.bookid AND Book.bookid=b1.bookid)
               >= 0.3 * (SELECT COUNT(*) FROM Customer));
```

4. 다음 질의에 대해 DML 문을 작성하시오.

(1) 새로운 도서 ('스포츠 세계', '대한미디어', 10000원)이 마당서점에 입고되었다. 삽입이 안 될 경우 필요한 데이터가 더 있는지 찾아보자.

```
INSERT INTO BOOK VALUES(11, '스포츠세계', '대한미디어', 10000);
```

(2) '삼성당'에서 출판한 도서를 삭제해야 한다.

```
DELETE FROM Book WHERE publisher LIKE '삼성당';
```

(3) '이상미디어'에서 출판한 도서를 삭제해야 한다. 삭제가 안 될 경우 원인을 생각해보자.

오류 : DELETE 문이 REFERENCE 제약조건과 충돌했습니다.

외래키 제약조건에 위배된다.

(4) 출판사 '대한미디어'가 '대한출판사'로 이름을 바꾸었다.

```
UPDATE Book SET publisher='대한출판사' WHERE publisher LIKE '대한미디어';
```

5. 다음 EXISTS 질의의 결과를 보이시오.

(1) 질의의 결과는 무엇인가?

주문이 없는 고객

(2) NOT을 지우면 결과는 무엇인가?

주문이 있는 고객

6. 극장 데이터베이스 다음은 4개의 지점을 가진 극장 데이터베이스다. 밑줄 친 속성은 기본키이다. 테이블의 구조를 만들고 데이터를 입력한 후 다음 질의에 대한 SQL 문을 작성하시오. 테이블의 구조를 만들 때 다음 제약조건을 반영하여 작성한다.

(1) 단순 질의

① 모든 극장의 이름과 위치를 보이시오.

```
SELECT 극장이름, 위치
FROM 극장;
```

② '잠실'에 있는 극장을 보이시오.

```
SELECT * FROM 극장
WHERE 위치 LIKE '잠실';
```

③ '잠실'에 사는 고객의 이름을 오름차순으로 보이시오.

```
SELECT 고객번호,이름,주소
FROM 고객
WHERE 주소 LIKE '잠실'
ORDER BY 이름;
```

④ 가격이 6,000원 이하인 영화의 극장번호, 상영관번호, 영화제목을 보이시오.

```
SELECT 극장번호, 상영관번호, 영화제목
FROM 상영관
WHERE 가격 <=6000;
```

⑤ 극장 위치와 고객의 주소가 같은 고객들을 보이시오.

```
SELECT 고객.이름, 극장.위치
FROM 고객, 극장
WHERE 고객.주소 LIKE 극장.위치;
```

(2) 집계질의

① 극장의 수는 몇 개인가?

```
SELECT COUNT(극장번호)
FROM 극장;
```

② 상영되는 영화의 평균 가격은 얼마인가?

```
SELECT AVG(가격)
```

```
        FROM 상영관;
```
③ 2013년 9월 1일에 영화를 관람한 고객의 수는 얼마인가?
```
        SELECT COUNT(이름)
        FROM 고객, 예약
        WHERE 예약.고객번호=고객.고객번호 AND 날짜 LIKE '2013-09-01';
```
(3) 부속질의와 조인

① '대한' 극장에서 상영된 영화제목을 보이시오.
```
        SELECT 영화제목
        FROM 극장, 상영관
        WHERE 극장.극장번호=상영관.극장번호
                AND 극장이름 LIKE '대한';
```
② '대한' 극장에서 영화를 본 고객의 이름을 보이시오.
```
        SELECT 고객.이름
        FROM 극장, 예약, 고객
        WHERE 극장.극장번호=예약.극장번호 AND 예약.고객번호=고객.고객번호
                AND 극장이름 LIKE '대한';
```
③ '대한' 극장의 전체 수입을 보이시오.
```
        SELECT SUM(가격)
        FROM 극장, 상영관, 예약
        WHERE 극장.극장번호=상영관.극장번호 AND
                상영관.극장번호=예약.극장번호 AND 상영간.상영관번호=예약.상영관번호;
```

(4) 그룹질의

① 극장별 상영관 수를 보이시오.
```
        SELECT 극장번호, COUNT(*)
        FROM  상영관
        GROUP BY 극장번호;
```
② '잠실'에 있는 극장의 상영관을 보이시오.
```
        SELECT  * FROM 극장, 상영관
        WHERE  극장.극장번호=상영관.극장번호 AND 위치 LIKE '잠실';
```
③ 2013년 9월 1일에 극장별 평균 관람 고객의 수를 보이시오.
```
        SELECT 극장번호, COUNT(*)
        FROM 예약
        WHERE 날짜 LIKE '2013-09-01'
        GROUP BY극장번호;
```
④ 2013년 9월 1일에 가장 많은 고객이 관람한 영화를 보이시오.
```
        SELECT 영화제목
        FROM 상영관, 예약
        WHERE 상영관.극장번호=예약.극장번호 AND 상영관.상영관번호=예약.상영관번호
                AND 날짜 LIKE '2013-09-01'
        GROUP BY 예약.극장번호, 예약.상영관번호
```

```
HAVING COUNT(*) = (   SELECT MAX(*)
        FROM 상영관, 예약
        WHERE 상영관.극장번호=예약.극장번호
                AND 상영관.상영관번호=예약.상영관번호
                AND 날짜 LIKE '2013-09-01'
        GROUP BY 예약.극장번호, 예약.상영관번호);
```

(5) DML

① 각 테이블에 데이터를 삽입하는 INSERT 문들을 하나씩 보이시오.

(생략)

② 영화의 가격을 10% 인상하시오.

```
UPDATE 상영관
SET 가격 = 가격 *1.1;
```

7. 판매원 데이터베이스 다음 릴레이션을 보고 물음에 답하시오. Salesperson은 판매원, Order는 주문, Customer는 고객을 나타낸다. 밑줄 친 속성은 기본키이고 custname과 salesperson은 각각 Customer.name과 Salesperson.name을 참조하는 외래키이다.

(1) 테이블을 생성하는 CREATE 문과 데이터를 삽입하는 INSERT 문을 작성하시오.

```
CREATE TABLE Order (
  number          PRIMARY KEY,
  custname        CHAR(10),
  salesperson     CHAR(10),
  amount          NUMBER,
  FOREIGN KEY(custname) REFERENCES Customer(name),
  FOREIGN KEY(salesperson) REFERENCES Salesperson(name));
  .. (이하 생략)
```

(2) 모든 판매원의 이름과 급여를 보이시오. 단, 중복 행은 제거한다.

```
SELECT    (DISTINCT) name, salary
FROM      Salesperson;
```

(3) 나이가 30세 미만인 판매원의 이름을 보이시오.

```
SELECT    name
FROM      Salesperson
WHERE     age < 30;
```

(4) 'S'로 끝나는 도시에 사는 고객의 이름을 보이시오.

```
SELECT    name
FROM      Customer
WHERE     city LIKE '%S';
```

(5) 주문을 한 고객의 수(서로 다른 고객만)를 보이시오.

```
SELECT    COUNT(DISTINCT custname)
FROM      Order;
```

(6) 판매원 각각에 대하여 주문의 수를 계산하시오.

```
SELECT      salesperson, COUNT(*)
FROM        Order
GROUP BY salesperson;
```

(7) 'LA'에 사는 고객으로부터 주문을 받은 판매원의 이름과 나이를 보이시오(부속질의를 사용).

```
SELECT name, age
FROM   Salesperson
WHERE name IN
        (SELECT salesperson FROM Order WHERE custname IN
                (SELECT name FROM Customer WHERE city LIKE 'LA') );
```

(8) 'LA'에 사는 고객으로부터 주문을 받은 판매원의 이름과 나이를 보이시오(조인을 사용).

```
SELECT     salesperson, age
FROM       Salesperson, Order, Customer
WHERE      Salesperson.name=Order.salesperson
           AND Order.custname=Customer.name AND city='LA';
```

(9) 두 번 이상 주문을 받은 판매원의 이름을 보이시오.

```
SELECT      Salesperson
FROM        Order
GROUP BY Salesperson
HAVING COUNT(*) > 1;
```

(10) 판매원 'TOM'의 봉급을 45,000원으로 변경하는 SQL 문을 작성하시오.

```
UPDATE Salesperson
SET        salary=45000
WHERE    name LIKE 'TOM';
```

8. 기업 프로젝트 데이터베이스 다음 릴레이션을 보고 물음에 답하시오. Employee는 사원, Department는 부서, Project는 프로젝트, Works는 사원이 프로젝트에 참여한 내용을 나타낸다. 한 사원이 여러 프로젝트에서 일할 수 있고, 한 프로젝트에 여러 사원이 일할 수 있다. hours-worked 속성은 각 사원이 각 프로젝트에서 일한 시간 수를 나타낸다. 밑줄 친 속성은 기본키이다.

※ 테스용 스크립트

```
CREATE TABLE Department
(
        deptno int not null,
        deptname varchar(20),
        manager varchar(20),
        primary key(deptno)
)

CREATE TABLE Employee
(
        empno int not null,
        name varchar(20),
        phoneno int,
        address varchar(20),
        sex varchar(20),
        position varchar(20),
```