ch03 연습문제

20190937 융합보안공학과 이혜린

연습문제2

(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 이름, 주소 from customer;

(4) 2014년 7월 4일~7월7일 사이에 주문받은 도서의 주문번호

select orderid from orders where order날짜 between 날짜_format("2014-07-04","%Y-%m-%d") and 날짜_format("2014-07-07","%Y-%m-%d");

select orderid from orders where order날짜 between '2014-07-04' and '2014-07-07';

(5) 2014년 7월 4일~7월7일 사이에 주문받은 도서를 제외한 도서의 주문번호

select orderid from orders where order날짜 not between 날짜_format("2014-07-04","%Y-%m-%d") and 날짜_format("2014-07-07","%Y-%m-%d");

select orderid from orders where orderid not in (select orderid from orders where order날짜 between '2014-07-04' and '2014-07-07');

select orderid from orders where order날짜 not between '2014-07-04' and '2014-07-07';

(6) 성이 '김' 씨인 고객의 이름과 주소

select 이름, 주소 from customer where 이름 like '김%';

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

select 이름, 주소 from customer where 이름 like '김%아';

(8) 주문하지 않은 고객의 이름(부속질의 사용)

select 이름 from customer where not exists (select custid from orders where customer.custid = orders.custid);

select 이름 from customer where custid not in (select distinct custid from orders);

select 이름 from customer where 이름 not in (select 이름 from customer inner join orders on customer.custid = orders.custid);

(9) 주문 금액의 총액과 주문의 평균 금액

select sum(orders.sale가격) as "총액", round(avg(orders.sale가격)) as "평균 금액" from orders;

select sum(sale가격), avg(sale가격) from orders;

(10) 고객의 이름과 고객별 구매액

select customer.이름, sum(orders.sale가격) from orders join customer on orders.custid = customer.custid group by customer.custid;

select C.이름, sum(O.sale가격) from customer C left outer join orders O on (C.custid=O.custid) group by C.이름;

(11) 고객의 이름과 고객이 구매한 도서 목록

select customer.이름, book.book이름 from orders left join customer on orders.custid = customer.custid left join book on orders.bookid = book.bookid;

select 이름, book이름 from customer C inner join on orders O on C.custid = O.custid inner join book B on O.bookid = B.bookid order by 이름;

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

select * from orders join book on orders.bookid = book.bookid where 가격 - sale가격 like (select MAX(가격 - sale가격) from orders join book on orders.bookid = book.bookid);

select * from orders inner join book on orders.bookid = book.bookid and 가격 - sale가격 = (select MAX(가격 - sale가격) from orders inner join book on orders.bookid = book.bookid);

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

select customer.이름 from orders join customer on orders.custid = customer.custid group by customer.custid having avg(orders.sale가격) > (select avg(sale가격) from orders);

심화문제 6

데이터 베이스 생성

```
create database movie;
```

데이터 베이스 선택

```
use movie;
```

테이블 생성 : 극장 (극장)

```
CREATE TABLE 극장 (
    극장번호 INTEGER PRIMARY KEY,
    극장이름 VARCHAR(20),
    극장위치 VARCHAR(20)
);
```

테이블 생성 : 고객 (고객)

```
CREATE TABLE 고객 (
    고객번호 INTEGER PRIMARY KEY,
    고객이름 VARCHAR(20),
    고객주소 VARCHAR(20)
);
```

테이블 생성 : 상영관 (상영관)

```
CREATE TABLE 상영관 (
    극장번호 INTEGER NOT NULL,
    상영관번호 INTEGER PRIMARY KEY,
    movie이름 VARCHAR(20),
    가격 INTEGER,
    seat INTEGER,
    FOREIGN KEY (극장번호)
        REFERENCES 극장 (극장번호)
);
```

테이블 생성 : 예약 (예약)

```
CREATE TABLE 예약 (
    극장번호 INTEGER NOT NULL,
    상영관번호 INTEGER NOT NULL,
    고객번호 INTEGER NOT NULL,
    seat번호 INTEGER,
    날짜 날짜,
    FOREIGN KEY (극장번호)
```

```
        REFERENCES 극장 (극장번호),
    FOREIGN KEY (상영관번호)
        REFERENCES 상영관 (상영관번호),
    FOREIGN KEY (고객번호)
        REFERENCES 고객 (고객번호)
);
```

테이블 데이터 생성 : 극장 (극장)

```
insert into 극장(극장번호, 극장이름, 극장위치)
        values(1,"롯데", "잠실");
insert into 극장(극장번호, 극장이름, 극장위치)
        values(2,"메가", "강남");
insert into 극장(극장번호, 극장이름, 극장위치)
        values(3,"대한", "잠실");
```

테이블 데이터 생성 : 상영관 (상영관)

```
insert into 상영관(극장번호, 상영관번호, movie이름, 가격, seat)
        values(1, 1, "어려운 영화", 15000, 48);
insert into 상영관(극장번호, 상영관번호, movie이름, 가격, seat)
        values(3, 1, "멋진 영화", 7500, 120);
insert into 상영관(극장번호, 상영관번호, movie이름, 가격, seat)
        values(3, 2, "재밌는 영화", 8000, 110);
```

테이블 데이터 생성 : 고객 (고객)

```
insert into 고객(고객번호, 고객이름, 고객주소)
        values(3,"홍길동","강남");
```

```
insert into 고객(고객번호, 고객이름, 고객주소)
        values(4,"김철수","잠실");
insert into 고객(고객번호, 고객이름, 고객주소)
        values(9,"박영희","강남");
```

테이블 데이터 생성 : 예약 (예약)

```
insert into 예약(극장번호, 상영관번호, 고객번호, seat번호, 날짜)
        values(3,2,3,15,"2014-09-01");
insert into 예약(극장번호, 상영관번호, 고객번호, seat번호, 날짜)
        values(3,1,4,16,"2014-09-01");
insert into 예약(극장번호, 상영관번호, 고객번호, seat번호, 날짜)
        values(1,1,9,48,"2014-09-01");
```

```
alter table 상영관 modify 가격 integer check (가격 <=20000);
alter table 상영관 modify 상영관번호 integer check (상영관번호 between 1 and 10);
alter table 예약 modify 좌석번호 integer unique;
```

데이터 베이스 질의 SQL

(1) 단순질의

1. 모든 극장의 이름과 위치를 보이시오.

```
SELECT 극장이름, 극장위치 FROM 극장;
```

2. '잠실'에 있는 극장을 보이시오.

```
SELECT * FROM 극장 WHERE 극장위치 = '잠실';
```

3. '잠실'에 사는 고객의 이름을 오름차순으로 보이시오.

SELECT 고객이름 FROM 고객 WHERE 고객주소 = '잠실' ORDER BY 고객이름 ASC;

4. 가격이 8,000원 이하인 영화의 극장번호, 상영관번호, 영화제목을 보이시오.

SELECT 극장번호, 상영관번호, movie이름 FROM 상영관 WHERE 가격 <= 8000;

5. 극장 위치와 고객의 주소가 같은 고객을 보이시오.

SELECT DISTINCT 극장.극장위치, 고객.고객이름 FROM 극장, 고객 WHERE 극장.극장위치 = 고객.고객주소;

select 이름, 주소 from 고객 where 주소 in (select 위치 from 극장);

(2) 집계질의

1. 극장의 수는 몇 개인가?

SELECT COUNT(*) FROM 극장;

2. 상영되는 영화의 평균 가격은 얼마인가?

SELECT AVG(가격) FROM 상영관;

3. 2014년 9월 1일에 영화를 관람한 고객의 수는 얼마인가?

SELECT COUNT(*) FROM 예약, 고객 WHERE 예약.고객번호 = 고객.고객번호 AND 날짜 = '2014-09-01';

select count(*) from 예약 where 날짜 ='2014-09-01';

(3) 부속질의와 조인

1. '대한' 극장에서 상영된 영화제목을 보이시오.

SELECT movie 이름 FROM 극장, 상영관 WHERE 극장.극장이름 = '대한' AND 극장.극장번호 = 상영관.극장번호;

select 영화제목 from 극장 inner join 상영관 on 극장.극장번호=상영관.극장번호 and 극장이름='대한';

2. '대한' 극장에서 영화를 본 고객의 이름을 보이시오.

SELECT 고객이름 FROM 고객 WHERE 고객번호 IN (SELECT 고객번호 FROM 예약 WHERE 극장번호 = (SELECT 극장번호 FROM 극장 WHERE 극장이름 = '대한'));

select 이름 from 고객 where 고객번호 in (select 예약.고객번호 from 예약 inner join 극장 on 극장.극장번호 = 예약.극장번호 and 극장이름='대한');

3. 대한 극장의 전체 수입을 보이시오.

SELECT SUM(가격) FROM 상영관 WHERE 상영관번호 IN (SELECT 상영관번호 FROM 예약 WHERE 극장번호 = (SELECT 극장번호 FROM 극장 WHERE 극장이름 = '대한'));

select 예약.극장번호, sum(가격) from 예약 inner join 상영관 on 예약.극장번호=상영관.극장번호 and 예약.상영관번호 =상영관.상영관번호 group by 예약.극장번호 having 예약.극장번호 = (select 극장.극장번호 from 극장 where 극장이름 = '대한');

(4) 그룹질의

1. 극장별 상영관 수를 보이시오.

SELECT 극장.극장이름, COUNT(*) FROM 상영관 JOIN 극장 ON 상영관.극장번호 = 극장.극장번호 GROUP BY 극장.극장번호;

select 극장이름, count(상영관번호) from 상영관 group by 극장번호;

select 극장이름, count(상영관번호) from 극장 left outer join 상영관 on 극장.극장번호=상영관.극장번호 group by 상영관.극장번호;

2. '잠실'에 있는 극장의 상영관을 보이시오.

SELECT * FROM 상영관 WHERE 극장번호 IN (SELECT 극장번호 FROM 극장 WHERE 극장위치 = '잠실');

select * from 상영관 inner join 극장 on 상영관.극장번호=극장.극장번호 and 극장.위치='잠실';

3. 2014년09월01일의 극장별 평균 관람 고객 수를 보이시오.

SELECT 극장.극장이름, COUNT(*) FROM 예약 JOIN 극장 ON 예약.극장번호 = 극장.극장번호 WHERE 예약.날짜 = '2014-09-01' GROUP BY 예약.극장번호;

select 극장번호, count(예약.고객번호) from 예약 where 날짜='2014-09-01' group by 극장번호;

4. 2014년09월01일에 가장 많은 고객이 관람한 영화를 보이시오.

SELECT movie이름 FROM 상영관, 예약 WHERE 상영관.극장번호 = 예약.극장번호 AND 상영관.상영관번호 = 예약.상영관번호 AND 날짜 LIKE '2014-09-01' GROUP BY 예약.극장번호 , 예약.상영관번호;

select 상영관.영화제목, count(*) from 상영관 inner join 예약 on 상영관.극장번호 = 예약.극장번호 and 상영관.상영관번호 = 예약.상영관번호 and 예약날짜='2014-09-01' group by 상영관.영화제목 having count(*) = (select max(c) from (select count(*) c from 상영관 inner join 예약 on 상영관.극장번호 =예약.극장번호 and 상영관 .상영관번호 =예약.상영관번호 and 예약.날짜='2014-09-01' group by 상영관.영화제목) t);

(마지막 t는 별명, 안붙이면 안됨...)

DML

1. 각 테이블에 데이터를 삽입하는 INSERT 문을 하나씩 실행시켜 보시오.

insert into 극장(극장번호, 극장이름, 극장위치)

values(4,"CGV", "마포");

insert into 상영관(극장번호, 상영관번호, movie이름, 가격, seat)

values(4, 3, "귀여운 영화", 9000, 60);

insert into 고객(고객번호, 고객이름, 고객주소)

values(7,"이혜린","마포");

insert into 예약(극장번호, 상영관번호, 고객번호, seat번호, 날짜)

values(4,3,7,20,"2014-09-03");

2. 영화의 가격을 10%씩 인상하시오.

```
UPDATE 상영관 SET 가격 = 가격 * 1.1;
```