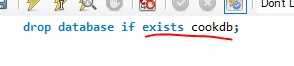
해당 DB가 있다면 삭제한다.(없는 DB 삭제 시 오류 나기 때문에 있다면 삭제하도록 조건 걸어줌)

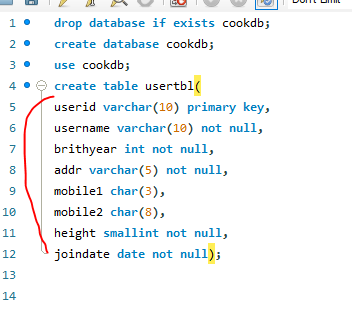
이런 경우는 존재하는 DB를 삭제하고 새로 데이터를 저장하기 위해서도 많이 사용함.



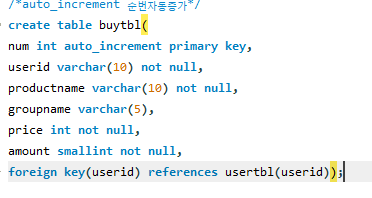
db생성



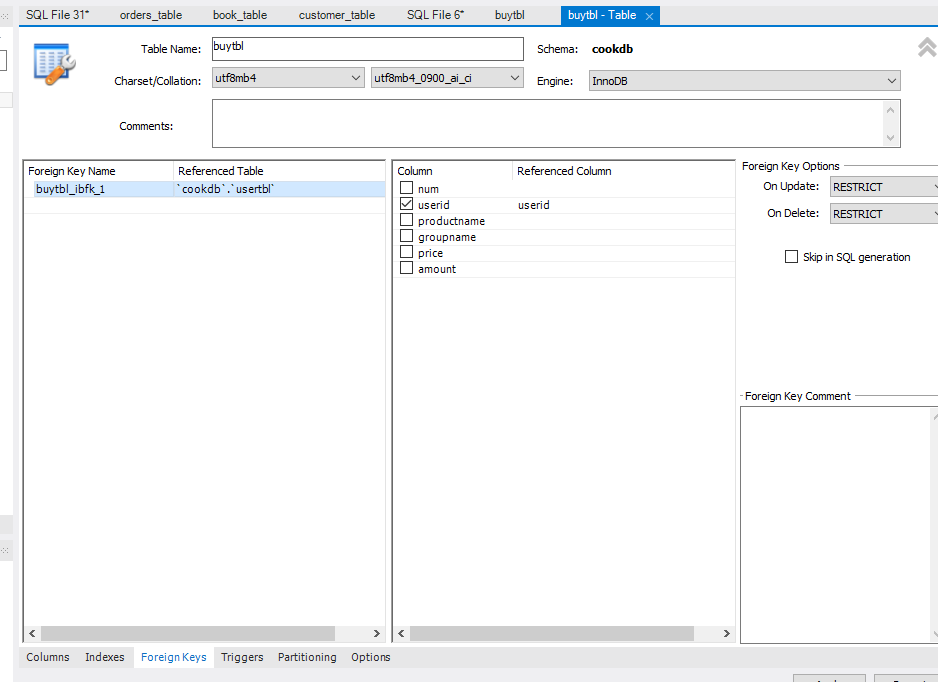
테이블 생성

\*smallint = short

외래키 설정



Foreign key(외래키로 설정할 현재 테이블의 필드명) references 참조할 테이블 명(참조할 필드명)



Column은 현재 테이블.

Referenced colunm은 Usertbl

외래키로 연결되어있으면 연결된 데이터는 지울수도 업데이트도 할수 없게 설정하는 것이

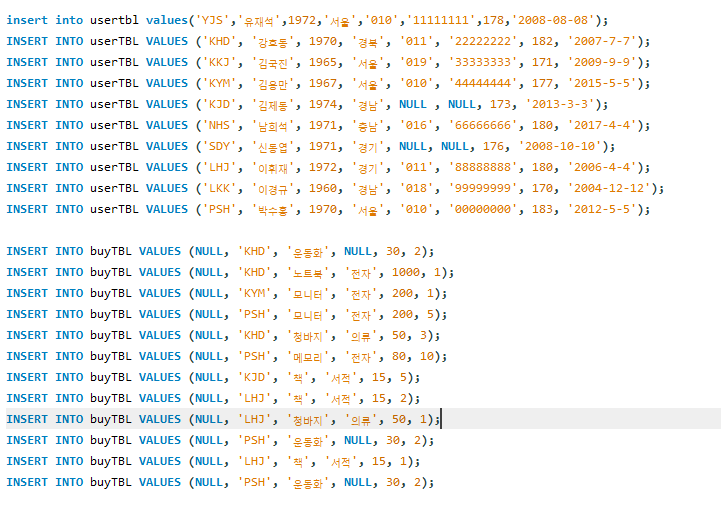
Foreign key options ->RESTRICT 이다.

Cascade는 좀 널널하게 해줌.

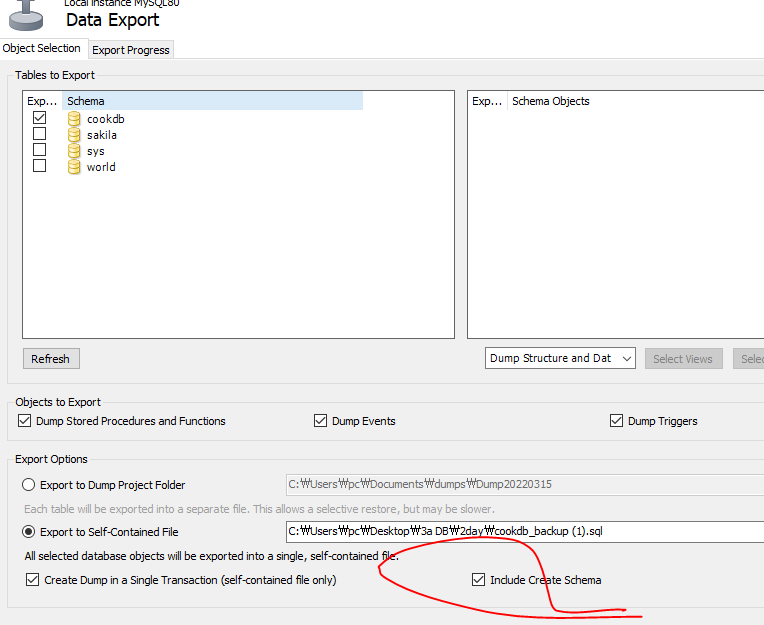
Not action 신경안쓰고

Set null은 null로 적용해줌.

전체 데이터에 넣고 싶을 때



콘솔로 저장할때와 밴치로 저장할 때 차이점



밴치로 저장할때는 데이터베이스가 없으면 인클루드하게하는 쿼리문이있다.

Sql 텍스트로 열어서 비교해보면 다름. CMD는 그게 없음.

Cooksqlbackup 1,2를 비교해보기.

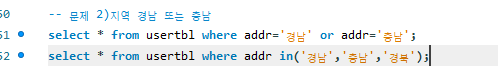
Sql 파일 실행하기



문제 1) 180~182의 키를 가진 회원 조회



문제 2) 경남 or 충남 회원 조회

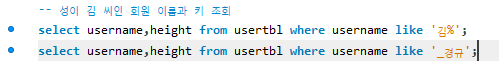


문제 2-1) 아닌 사람만





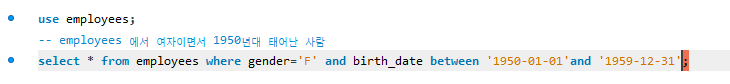
문제 3) 성이 김씨인 회원 이름 키 조회



와일드 문자 %(글자수 제한 없음) \_는 한 개당 글자 하나.

Like 저 형식과 비슷하면 조회.

문제 4)여자이면서 1950년대 태어난 사람



문제 5)입사일이 1990년 이후이며 firstname 에 k가 들어간 사람





Binary 는 대소문자 구분하고 싶을 때 필드명 앞에 붙여서 사용.

문제 6)titles에서 title이 engineer, staff, senior staff 인 정보 조회





필드 명 앞에 binary 붙여서 대소문자 구분을 하면 Staff만 출력된다.

보너스 문제)

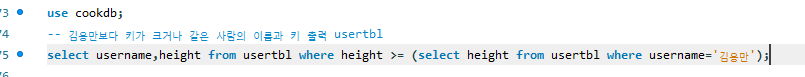
타이틀만 출력하고 싶을 때 이렇게 하는데 중복을 없애고 타이틀 종류만 확인하고 싶을 때

Distinct 사용.





문제 7) 김용만보다 키가 큰 사람의 이름과 키 출력



쿼리 안에 쿼리가 들어있는 것을 서브 쿼리라고 한다.

서브쿼리가 먼저 실행된다.

Where절 수행 시

레코드 하나씩 가져와서 비교해보고 참이면 레코드 가져오고 아니면 거르고 한다.

문제 8) 지역이 경기인 사람보다 키가 크거나 같은 사람 추출



\*any(or)

비교할 때 사용.

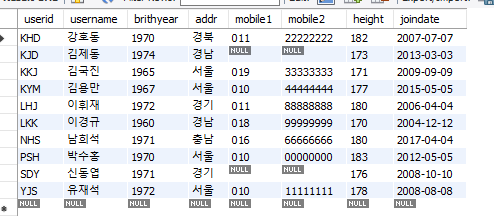
현재 select height from usertbl where addr =’경기’ 했을때 리턴되는 결과가 2개이기 때문에 두 결과 둘중 아무거나 충족할 때 출력하고 싶으면 any 사용.

\*all(and)

리턴된 결과 모두를 충족해야함.

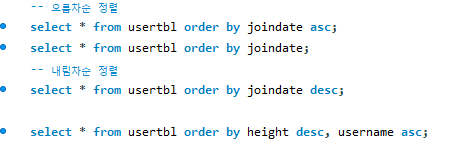
\*in

같다인 경우.



유재석부터 저장했는데 순서가 이렇게 되는 이유는 primary로 설정하며 그 기준으로 정렬되었기 때문.

보너스 문제 1) 정렬



보너스 문제 2) 거주지 정보만 빼오기



보너스 문제 3) 제일 오래된 입사일 검색



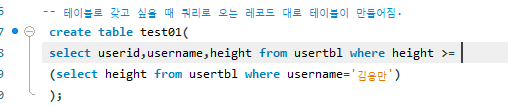
5명만 검색하기.



11번째부터 5개 검색.



문제 9) 테이블로 검색결과 저장 하고 싶을 때



이렇게 저장 했을 때는 키에 대한 제약조건까지 오지는 않기 때문에 따로 설정해줘야 함.

Primary key 같은 설정. 그래서 벤치에서 체크해서 설정해줘도 됨.

문제 10) 아이디끼리 묶어서 아이디당 돈을 얼마나 썼는지 합계를 구하기



많이 쓴 사람 기준으로 정렬.





구매 개수로 합산할 때



회원 수를 카운트 하고 싶을 때



제약조건 주고 싶을 때



지역별 인구 검색하고 싶을 때





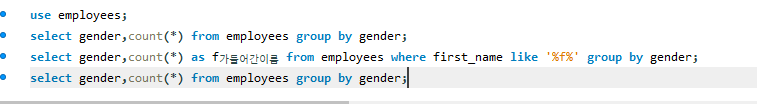
where절부터 실행되기 때문에 where절에 만족된 레코드의 그룹에서 카운트를 한다.

대상이 되는 데이터가 결정이 되면 집계함수가 실행되기 때문에 제일 마지막에 사용된다.

그래서 where절에서는 집계함수를 사용할 수 없다. Ex) count(\*)

혼자 해본 문제 )

회원 정보에서 f가 들어간 이름을 남녀 구분해서 카운트한 결과 출력



문제 11) max

키가 큰 사람 지역별로 나누기



평균으로 출력



최소값



문제 12) 가장 큰 키의 유저 정보를 알고싶다.



이렇게 하면 where절이 없기 때문에 가장 처음 만나는 레코드의 id,name이 출력되고 그 다음

가장 큰 키가 출력된다.

해결법)



서브 쿼리를 사용한다.

문제 13) 지역별로 가장 큰 사람의 정보 출력



문제 14) 총 구매액이 1000 넘는 유저 정보



Having은 where 과 같이 조건문 이지만 다르게 작동한다.

where절은 레코드를 빼오고 나서 직계함수 호출하는데 (제일 먼저 수행)

having절은 맨 마지막에 처리한다.

Having 사용시 무조건 집계함수 사용해야 한다.



이렇게도 됨.

오늘 한 쿼리문 전체

drop database if exists cookdb;

create database cookdb;

use cookdb;

create table usertbl(

userid varchar(10) primary key,

username varchar(10) not null,

brithyear int not null,

addr varchar(5) not null,

mobile1 char(3),

mobile2 char(8),

height smallint not null,

joindate date not null);

/\*auto\_increment 순번자동증가\*/

create table buytbl(

num int auto\_increment primary key,

userid varchar(10) not null,

productname varchar(10) not null,

groupname varchar(5),

price int not null,

amount smallint not null,

foreign key(userid) references usertbl(userid));

insert into usertbl values('YJS','유재석',1972,'서울','010','11111111',178,'2008-08-08');

INSERT INTO userTBL VALUES ('KHD', '강호동', 1970, '경북', '011', '22222222', 182, '2007-7-7');

INSERT INTO userTBL VALUES ('KKJ', '김국진', 1965, '서울', '019', '33333333', 171, '2009-9-9');

INSERT INTO userTBL VALUES ('KYM', '김용만', 1967, '서울', '010', '44444444', 177, '2015-5-5');

INSERT INTO userTBL VALUES ('KJD', '김제동', 1974, '경남', NULL , NULL, 173, '2013-3-3');

INSERT INTO userTBL VALUES ('NHS', '남희석', 1971, '충남', '016', '66666666', 180, '2017-4-4');

INSERT INTO userTBL VALUES ('SDY', '신동엽', 1971, '경기', NULL, NULL, 176, '2008-10-10');

INSERT INTO userTBL VALUES ('LHJ', '이휘재', 1972, '경기', '011', '88888888', 180, '2006-4-4');

INSERT INTO userTBL VALUES ('LKK', '이경규', 1960, '경남', '018', '99999999', 170, '2004-12-12');

INSERT INTO userTBL VALUES ('PSH', '박수홍', 1970, '서울', '010', '00000000', 183, '2012-5-5');

INSERT INTO buyTBL VALUES (NULL, 'KHD', '운동화', NULL, 30, 2);

INSERT INTO buyTBL VALUES (NULL, 'KHD', '노트북', '전자', 1000, 1);

INSERT INTO buyTBL VALUES (NULL, 'KYM', '모니터', '전자', 200, 1);

INSERT INTO buyTBL VALUES (NULL, 'PSH', '모니터', '전자', 200, 5);

INSERT INTO buyTBL VALUES (NULL, 'KHD', '청바지', '의류', 50, 3);

INSERT INTO buyTBL VALUES (NULL, 'PSH', '메모리', '전자', 80, 10);

INSERT INTO buyTBL VALUES (NULL, 'KJD', '책', '서적', 15, 5);

INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 2);

INSERT INTO buyTBL VALUES (NULL, 'LHJ', '청바지', '의류', 50, 1);

INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);

INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 1);

INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);

use cookdb;

-- 문제 1)키 180~182

select \* from usertbl where height>=180 and height <=182;

select \* from usertbl where height between 180 and 182;

-- 문제 2)지역 경남 또는 충남

select \* from usertbl where addr='경남' or addr='충남';

select \* from usertbl where addr in('경남','충남','경북');

select \* from usertbl where addr not in('경남','충남','경북');

select \* from usertbl where addr !='경남';

-- 성이 김 씨인 회원 이름과 키 조회

select username,height from usertbl where username like '김%';

select username,height from usertbl where username like '\_경규';

use employees;

-- employees 에서 여자이면서 1950년대 태어난 사람

select \* from employees where gender='F' and birth\_date between '1950-01-01'and '1959-12-31';

-- employees 에서 1990년 이후 입사한 사람 중에 first\_name에 k가 들어가는 사람

select \* from employees where hire\_date >='1990-01-01' and first\_name like '%k%';

select \* from employees where hire\_date >='1990-01-01' and binary first\_name like '%K%';

-- titles에서 title이 engineer, staff, senior staff만 보기

select \* from titles where title in('engineer','staff','senior staff');

select \* from titles where binary title in('engineer','Staff','senior staff');

select title from titles;

select distinct title from titles;

use cookdb;

-- 김용만보다 키가 크거나 같은 사람의 이름과 키 출력 usertbl

select username,height from usertbl where height >= (select height from usertbl where username='김용만');

-- 테이블로 갖고 싶을 때 쿼리로 오는 레코드 대로 테이블이 만들어짐.

create table test01(

select userid,username,height from usertbl where height >=

(select height from usertbl where username='김용만')

);

-- 지역이 경기인 사람보다 키가 크거나 같은 사람 추출

select \* from usertbl where height >= any (select height from usertbl where addr='경기');

-- 오름차순 정렬

select \* from usertbl order by joindate asc;

select \* from usertbl order by joindate;

-- 내림차순 정렬

select \* from usertbl order by joindate desc;

select \* from usertbl order by height desc, username asc;

-- 쇼핑몰 거주지 정보만 보고싶으면

select distinct addr from usertbl;

use employees;

select emp\_no,hire\_date from employees order by hire\_date asc limit 10,5;

-- 집계함수 (그룹을 묶어서 합계를 구하거나 평균을 내거나)

-- 돈을 제일 많이 쓴 사람을 구하기 위해 아이디끼리 묶어보기

select userid,sum(amount) as 총구매개수

from buytbl group by userid order by sum(amount) desc;

-- 회원 수가 몇명인지 카운트 하고 싶을때

select count(\*) from usertbl where height >=180;

select count(\*) from usertbl where addr='서울';

-- 각 각 지역별로 사는 사람 카운트 하고 싶을 때

select addr,count(\*) from usertbl where height >=180 group by addr;

-- 혼자 만들어 본 문제

-- 유저 검색해서 f가 들어간 유저들을 성별로 나누어 count

use employees;

select gender,count(\*) from employees group by gender;

select gender,count(\*) as f가들어간이름 from employees where first\_name like '%f%' group by gender;

select gender,count(\*) from employees group by gender;

use cookdb;

-- max,avg,min

select addr, min(height) from usertbl group by addr;

-- 가장 큰 키 사람의 모든 정보를 알고 싶다.

select userid,username, height from usertbl where height=(select max(height) from usertbl);

-- 지역별로 가장 큰 키 검색

select \* from usertbl where addr='서울' and height=183;

select \* from usertbl where (addr, height)=('서울',183);

select addr,username,height from usertbl where (addr,height) in ((select addr,max(height) from usertbl group by addr));

select addr,username,height from usertbl where (addr,height) in (('경북',182),('경남',173));

-- 총 구매액이 1000을 넘는 사람들만 보기

select userid,sum(price\*amount) as 총구매액 from buytbl group by userid having 총구매액>1000;

과제

drop database if exists Madang;

create database Madang;

use Madang;

create table Book\_table(bookid int auto\_increment primary key,bookname varchar(20) not null,publisher varchar(20) not null,

price int not null);

insert into Book\_table values(null,'축구의 역사','굿스포츠',7000);

insert into Book\_table values(null,'축구아는 여자','나무수',13000);

insert into Book\_table values(null,'축구의 이해','대한미디어',22000);

insert into Book\_table values(null,'골프 바이블','대한미디어',35000);

insert into Book\_table values(null,'피겨 교본','굿스포츠',8000);

insert into Book\_table values(null,'역도 단계별기술','굿스포츠',6000);

insert into Book\_table values(null,'야구의 추억','이상미디어',20000);

insert into Book\_table values(null,'야구를 부탁해','이상미디어',13000);

insert into Book\_table values(null,'올림픽 이야기','삼성당',7500);

insert into Book\_table values(null,'Olympic Champions','Pearson',13000);

create table Customer\_table(custid int auto\_increment primary key,username varchar(20) not null,addr varchar(30) not null,

phone char(13));

insert into Customer\_table values(null,'박지성','영국 맨체스타','000-5000-0001');

insert into Customer\_table values(null,'김연아','대한민국 서울','000-6000-0001');

insert into Customer\_table values(null,'장미란','대한민국 강원도','000-7000-0001');

insert into Customer\_table values(null,'추신수','미국 클리블랜드','000-8000-0001');

insert into Customer\_table values(null,'박세리','대한민국 대전',null);

create table Orders\_table(orderid int auto\_increment primary key,custid int not null,bookid int not null,saleprice int not null,

orderdate char(10) not null,foreign key(custid) references Customer\_table(custid),foreign key(bookid) references Book\_table(bookid));

insert into Orders\_table values(null,1,1,6000,'2014-07-01');

insert into Orders\_table values(null,1,3,21000,'2014-07-03');

insert into Orders\_table values(null,2,5,8000,'2014-07-03');

insert into Orders\_table values(null,3,6,6000,'2014-07-04');

insert into Orders\_table values(null,4,7,20000,'2014-07-05');

insert into Orders\_table values(null,1,2,12000,'2014-07-07');

insert into Orders\_table values(null,4,8,13000,'2014-07-07');

insert into Orders\_table values(null,3,10,12000,'2014-07-08');

insert into Orders\_table values(null,2,10,7000,'2014-07-09');

insert into Orders\_table values(null,3,8,13000,'2014-07-10');

-- 문제 1) 가격이 20000원 미만인 도서 검색

select \* from book\_table where price < 20000;

-- 문제 2) 가격이 10000원 이상 20000 이하인 도서 검색

select \* from book\_table where price >=10000 and price <=20000;

-- 문제 3) 출판사가 '굿스포츠' 혹은 '대한미디어'인 도서를 검색

select \* from book\_table where publisher in ('굿스포츠','대한미디어');

-- 문제 4) 출판사가 '굿스포츠' 혹은 '대한미디어'가 아닌 도서를 검색

select \* from book\_table where publisher not in ('굿스포츠','대한미디어');

-- 문제 5) '축구의 역사'를 출간한 출판사를 검색

select publisher as 출판사 from book\_table where bookname = '축구의 역사';

-- 문제 6) 도서이름에 '축구'가 포함된 출판사 검색

select publisher as 출판사 from book\_table where bookname like '%축구%';

-- 문제 7) 도서이름이 여섯 글자인 도서를 검색

select \* from book\_table where bookname like '\_\_\_\_\_\_';

-- 문제 8) 도서이름의 왼쪽 두 번째 위치에 '구'라는 문자열을 갖는 도서를 검색

select \* from book\_table where bookname like '\_구%';

-- 문제 9) 축구에 관한 도서 중 가격이 20000원 이상인 도서를 검색

select \* from book\_table where bookname like '%축구%' and price>=20000;

-- 문제 10) 야구에 관한 책을 모두 구입하려면 필요한 금액 계산.

select sum(price) as 야구도서총금액 from book\_table where bookname like '%야구%';

-- 문제 11) 도서를 가격 순으로 검색하고 가격이 같으면 이름순으로 검색

select \* from book\_table order by price asc ,bookname asc;

-- 문제 12) 도서를 가격의 내림차순으로 검색하고 만약 가격이 같다면 출판사의 오름차순으로 검색

select \* from book\_table order by price desc,publisher asc;

-- 문제 13) 주소가 우리나라나 영국인 선수정보 조회 customer table

select \* from customer\_table where addr like '대한민국%'or addr like'영국%';

-- 문제 14) 고객이 주문한 도서의 총 판매액 조회 orders\_table

select custid as 주문자명,sum(saleprice) as 총판매액 from orders\_table where saleprice group by custid ;

-- 문제 15) 2번 김연아 고객이 주문한 도서의 총 판매액 조회

select custid as 주문자명,sum(saleprice) as 총판매액 from orders\_table where custid=2 and saleprice group by custid;

-- 문제 16) 고객이 주문한 도서의 총 판매액, 평균값,최저가,최고가 조회(문제에대해질문하기)

select bookid as 도서아이디,sum(saleprice) as 총판매액, avg(saleprice) as 평균판매액, min(saleprice) as 최저가, max(saleprice) as 최고가 from orders\_table where saleprice group by bookid;

-- 문제 17) 마당서점의 도서 판매 건수 조회(문제에대해질문하기)

select count(\*) as 서점총판매 from orders\_table where orderid;

-- 문제 18) 고객별로 주문한 도서의 총 수량과 총 판매액 조회

select custid as 고객아이디,count(custid),sum(saleprice) from orders\_table where custid and saleprice group by custid;

-- 문제 19) 가격이 8000원 이상인 도서를 구매한 고객에 대하여 고객별 주문 도서의 총 수량을 구하시오. 단, 두 권 이상 구매한 고객만 조회

select custid as 고객아이디,count(custid) as 총구매한도서수량 from orders\_table where saleprice >=8000 group by custid having 총구매한도서수량 >=2;

-- 문제 20) 날짜별 총 구매건수와 총 판매액을 조회

select orderdate as 구매일,count(orderdate) as 구매건수,sum(saleprice)as 총판매액 from orders\_table group by orderdate;

-- 문제 21) 총 판매액이 20000이 넘는 날짜의 총 구매건수를 조회

select orderdate as 구매일,count(orderdate) as 구매건수,sum(saleprice)as 총판매액 from orders\_table group by orderdate having 총판매액 >20000;

-- 문제 22) 가장 구매건수가 많은 날짜를 조회 구매건수가 같은 경우 가장 최근 날짜를 조회

select orderdate,maxcount from orders\_table where(orderdate,maxcount) in (select orderdate as 구매일,count(orderdate) as 구매건수 from orders\_table group by orderdate)

group by orderdate;

select orderdate as 구매일,count(orderdate) as 구매건수 from orders\_table group by orderdate having 구매건수 >=(select) ;