/\* 이름: demo\_madang.sql \*/

/\* 설명 \*/

//visual studio code 터미널를 이용하여 mysql

cmd

chcp 65001

cd C:\

cd .\Bitnami\wampstack-8.1.11-0(확인 요망 -> 버전마다 다름)

mariadb\bin\mysql -u '유저' -p'비밀번호' '테이블' < '파일명.sql'

C:\Bitnami\wampstack-8.1.2-0 > chcp 65001

mysql -u root -p < demo\_madang.sql

mysql -u madang -p madang

/\* root 계정으로 접속, madang 데이터베이스 생성, madang 계정 생성 \*/

/\* MySQL Workbench에서 초기화면에서 +를 눌러 root connection을 만들어 접속한다. \*/

DROP DATABASE IF EXISTS madang;

DROP USER IF EXISTS madang@localhost;

use mysql;

create user madang@localhost identified by 'madang';

create database madang;

grant all privileges on madang.\* to madang@localhost with grant option;

commit;

/\* madang DB 자료 생성 \*/

/\* 이후 실습은 MySQL Workbench에서 초기화면에서 +를 눌러 madang connection을 만들어 접속하여 사용한다. \*/

USE madang;

CREATE TABLE Book (

bookid INTEGER PRIMARY KEY,

bookname VARCHAR(40),

publisher VARCHAR(40),

price INTEGER

);

CREATE TABLE Customer (

custid INTEGER PRIMARY KEY,

name VARCHAR(40),

address VARCHAR(50),

phone VARCHAR(20)

);

CREATE TABLE Orders (

orderid INTEGER PRIMARY KEY,

custid INTEGER ,

bookid INTEGER ,

saleprice INTEGER ,

orderdate DATE,

FOREIGN KEY (custid) REFERENCES Customer(custid),

FOREIGN KEY (bookid) REFERENCES Book(bookid)

);

INSERT INTO Book VALUES(1, '축구의 역사', '굿스포츠', 7000);

INSERT INTO Customer VALUES (1, '박지성', '영국 맨체스타', '000-5000-0001');

INSERT INTO Orders VALUES (1, 1, 1, 6000, STR\_TO\_DATE('2014-07-01','%Y-%m-%d'));

-- 여기는 3장에서 사용되는 Imported\_book 테이블

CREATE TABLE Imported\_Book (

bookid INTEGER,

bookname VARCHAR(40),

publisher VARCHAR(40),

price INTEGER

);

INSERT INTO Imported\_Book VALUES(21, 'Zen Golf', 'Pearson', 12000);

INSERT INTO Imported\_Book VALUES(22, 'Soccer Skills', 'Human Kinetics', 15000);

commit;

SELECT문의 기본 문법

|  |
| --- |
| select [all | distinct(중복삭제)] 속성이름(들) |
| from 테이블이름(들) |
| [where 조인조건(들), 검색조건(들)] |
| [group by 속성이름] |
| [having 검색조건(들)] |
| [order by 속성이름[asc:오름차순|desc:내림차순] |

where절에 조건으로 사용할 수 있는 술어

|  |  |  |
| --- | --- | --- |
| 술어 | 연산자 | 사용 예 |
| 비교 | ==, <>(== ‘!=’), <, <=, >, >= | price < 20000 |
| 범위 | between | price between 10000 and 20000 |
| 집합 | in, not in | price in 『(10000, 20000, 30000)』부속쿼리(subquery) |
| 패턴 | like | bookname like ‘축구의 역사(= bookname = ‘축구의 역사’) |
| NULL | is null, is not null | phone is null |
| 복합조건 | and, or, not | (price < 20000) and (bookname like ‘축구의 역사’) |

select 집계함수 『as ‘별명’』 -> 의미 있는 열 이름을 추력하고 싶으면 속성이름의 별칭을 자칭하는 as키워드를 사용하여 열 이름을 부여한다.  
count()의 괄호 안에는 \* 혹은 특정 속성의 이름이 사용되며, 해당 속성의 투플의 개수를 세어준다.

와일드 문자의 종류

|  |  |  |
| --- | --- | --- |
| 와일드 문자 | 의미 | 사용 예 |
| + | 문자열을 연결 | ‘골프’ + ‘바이블’ -> ‘골프 바이블’ |
| % | 0개 이상의 문자열과 일치 | ‘%축구%: 축구를 포함하는 문자열 |
| []. | 1개의 문자와 일치 | ‘[0-5]%’: 0-5사이 숫자로 시작하는 문자열->한글사용X |
| [^] | 1개의 문자와 불일치 | ‘[^0-5]%’: 0-5사이 숫자로 시작하지 않는 문자열->한글사용X |
| \_ | 특정 위치의 1개의 문자와 일치 | ‘\_구%’: 두 번째 위치에 ‘구’가 들어가는 문자열 |

group by와 having절의 문법의 주의 사항

|  |  |
| --- | --- |
| 문법 | 주의사항 |
| group by <속성> | group by로 투플을 그룹으로 묶음 후 select절에는 group by에서 사용한<속성>과 집계함수만 나올 수 있다. |
| Having <검색 조건> | ① 반드시 group by절과 같이 작성해야 한다.  ② where절보다 뒤에 나와야 한다.  ③ <검색조건>에는 SUM, AVG, MAX, MIN, COUNT와 같이 집계함수가 와야 한다. |

조인 문법

|  |  |  |
| --- | --- | --- |
| 명령 | 문법 | 설명 |
| 일반적인 조인 | select <속성들> from 테이블1, 테이블2  where <조인조건>and<검색조건> | SQL문에서 주로 동등조인을 사용한다. 두 가지 문법 중 하나를 사용할 수 있다. |
| select <속성들> from 테이블1 inner join 테이블2 on <조인조건>  where <검색조건> |
| 외부조인 | select <속성들> from 테이블1 {left|right|full[outer]} join 테이블2 on <조인조건>  where <검색조건> | 외부조인은 from 절에 조인 종류를 적고 on을 이용하여 조인조건을 명시한다. |

select sum(saleprice) as '총구매액' from orders, customer   
where orders.custid = customer.custid and customer.name = '김연아';

select customer.name, count(orders.bookid) as ' 도서의 수' from orders, customer   
where orders.custid = customer.custid and customer.name = '박지성';

부속질의  
select <속성들> from <테이블> where select <속성들> from <테이블> where <검색조건>