06 INSERT/UPDATE/DELETE

Part 1.

INSERT

01 기본

문법

INSERT [INTO] 테이블 [(열1,열2...)] VALUES(값1,값2...)

기본 예제

```
USE sqlDB;
CREATE TABLE testTbl1 (id int,userName char(10),age int);

INSERT INTO testTbl1 VALUES (1,'홍길동',25); -- 모든열 데이터 입력
INSERT INTO testTbl1(id,userName) VALUES(2,'강동원'); -- 특정열 데이터 입력
INSERT INTO testTbl1(username,age,id) VALUES('이수근',26,3) --순서변경;
SELECT * FROM testTbl1;
```

02 AUTO_INCREMENT

AUTO_INCREMENT - 자동 숫자 증가

```
USE sqldb;
CREATE TABLE testTbl2(
id int AUTO_INCREMENT PRIMARY Key,
userName nchar(3),
age int
);

INSERT INTO testTbl2 VALUES(NULL,'길동',25);
INSERT INTO testTbl2 VALUES(NULL,'만수',27);
SELECT * FROM testTbl2;
```

02 AUTO_INCREMENT

Auto_increment 다음값 변경

```
alter table testTbl2 auto_INCREMENT=100;
INSERT INTO testTbl2 values(NULL,'지성',28);
SELECT * FROM testTbl2;
```

증가값변경

```
use sqldb;
create table testTbl3
(id int auto_increment primary key,
userName char(3),
age int);
alter table testtbl3 auto_increment=1000;
set @@auto_increment_increment=3; --서버변수
insert into testtbl3 values (NULL,'도연','21');
insert into testtbl3 values (NULL,'상민','22');
insert into testtbl3 values (NULL,'산화','33');
select * from testtbl3;
```

03 중간문제

문제

1 다음 조건을 만족하며 테이블을 만드세요
테이블명 : CustomerTbl
NUM(int)　　Name(char(10))　　Age(int)　addr(char(20))

- NUM 행은 자동으로 숫자가 부여되어야 합니다
- NUM 행의 시작값은 1000으로 합니다
- NUM 행의 증가값은 5입니다

2 다음 값을 삽입하세요
정우균　35　　대구
박희석　27　　서울
홍길동　30　　부산

04 대량 샘플데이터 생성

사용법

INSERT INTO 테이블명(열이름1, 열이름2 ,...) SELECT 문;

예제 01

```
use sqldb;
create table testTbl4 (id int,Fname varchar(50),Lname varchar(50));
insert into testTbl4 select emp_no, first_name,last_name from employees.employees;

select * from testTbl4;
```

04 대량 샘플데이터 생성

예제 02 - 테이블 정의 생략

```
use sqldb;
create table testTbl6 (select emp_no , first_name, last_name from employees.employees);
select * from testTbl6;
```

Part 2.

UPDATE

01 UPDATE

사용법

UPDATE 테이블이름 SET 열1=값1,열2=값2... WHERE 조건;

UPDATE - 조건 변경

```
update testtbl4
set Lname='없음'
where Fname='Kyoichi';
select * from testtbl4;
```

UPDATE - 전체 변경

```
USE sqlDB;
SELECT * FROM buyTbl;
UPDATE buyTbl SET price = price*1.5;
select * from buyTbl;
```

Part 3.

DELETE

01 DELETE

사용법

DELETE 테이블이름 WHERE 조건

DELETE - 조건 내 모두 삭제

```
use sqldb;
delete from testTbl4 where Fname = 'Aamer';
```

DELETE - 조건 상위 삭제

```
use sqldb;
delete from testTbl4 where Fname='Aamer' LIMIT 5;
```

01 삭제 속도 비교

사용법

```
use sqldb;
create table bigTbl1 (select * from employees.employees);
create table bigTbl2 (select * from employees.employees);
create table bigTbl3 (select * from employees.employees);

delete from bigTbl1;
drop table bigTbl2;
Truncate table bigTbl3;
```

01 삭제 속도 비교

✔	81	10:52:18	delete from bigTbl1	300024 row(s) affected	1.766 sec
✔	82	10:52:20	drop table bigTbl2	0 row(s) affected	0.031 sec
✔	83	10:52:20	truncate table bigTbl3	0 row(s) affected	0.031 sec

설명

delete 는 트랜잭션 로그를 기록하는 작업 때문에 삭제시 시간이 상대적으로 오래걸린다

drop은 테이블 자체를 삭제한다

truncate는 delete와 동일하지만 트랜젝션 로그를 기록하지 않아서 삭제 속도가 훨씬 빠르다
대용량 데이터 삭제시 테이블이 필요없으면 drop
테이블이 필요있으면 truncate로 삭제하는것이 효율적이다

Part 4.

INSERT 추가

01 INSERT IGNORE

테이블생성+PK설정

```
use sqldb;
create table memberTbl (SELECT userID, name, addr from usertbl LIMIT 3);
ALTER TABLE memberTBL ADD CONSTRAINT pk_memberTBL PRIMARY KEY (userID);
select * from memberTbl;
```

값 삽입

```
insert into memberTbl values('bbk','비비코','미국'); --PK 중복 허용x
insert into memberTbl values('sjh','서장훈','서울'); -- 상위 명령어 실패로 인한 종료
insert into memberTbl values('jhy','현주엽','경기');-- 상위 명령어 실패로 인한 종료
select * from memberTbl;

insert IGNORE into memberTbl values('bbk','비비코','미국'); --PK 제약조건 위반시 무시넘어감
insert IGNORE into memberTbl values('sjh','서장훈','서울'); -- SQL문 실행
insert IGNORE into memberTbl values('jhy','현주엽','경기'); -- SQL문 실행
select * from memberTbl;
```

01 INSERT IGNORE

키중복시 데이터 수정

```
insert into memberTbl values('bbk','비비코','미국')
ON DUPLICATE KEY UPDATE name='비비코',addr='미국';
insert into memberTbl values('DJM','동짜몽','일본')
ON DUPLICATE KEY UPDATE name ='동짜몽', addr='일본';
select * from memberTbl;
```

Part 4.

WITH CTE

01 WITH CTE

WITH CTE란

Common Table Expression 를 표현하기 위한 구문
기존의 뷰 , 파생 테이블 , 임시테이블 등으로 사용되던 것을 대신할 수 있으며 더간결하게 사용할 수 있는 장점
Mysql 8.0 이후 버전부터 사용가능

비재귀적 CTE

```
WITHCTE_테이블이름(열이름)
AS
(
  〈쿼리문〉
)
SELECT 열 이름 FROM CTE_테이블이름;
```

01 WITH CTE

WITH CTE 사용 안함

```
use sqldb;
select userid as '사용자' ,sum(price*amount) as '총구매액'
from buyTbl
group by userid
order by 총구매액 desc;
```

WITH CTE 사용

```
WITH abc(userid, total)
AS
(
    select userid, sum(price*amount) from buytbl group by userid
)select * from abc order by total desc;
```

01 WITH CTE

WITH CTE 사용

1 각 지역별로 가장 큰키
select addr, MAX(height) from usertbl group by addr;

2 위 쿼리를 with구문으로 묶는다
WITH cte_userTbl(addr,maxHeight)
AS
(
select addr, MAX(height) from usertbl group by addr;
)

3 키의 평균을 구하는 쿼리를 작성한다
WITH cte_userTbl(addr,maxHeight)
AS
(select addr, MAX(height) from usertbl group by addr)
select AVG(maxHeight*1.0) as '각지역별 최고키의 평균' from cte_userTbl;

E N D.

고생하셨습니다.