/* 07장 */

형변환

string을

날짜형식으로변경

Alias

SELECT CAST('2020-10-19 12:35:29.123' AS DATE) AS 'DATE' ;

SELECT CAST('2020-10-19 12:35:29.123' AS TIME) AS 'TIME' ;

SELECT CAST('2020-10-19 12:35:29.123' AS DATETIME) AS 'DATETIME' ;

-- <실습 1> --

Variable 저장 (변수저장)

USE sqlDB;

SET @myVar1 = 5 ;

SET @myVar2 = 3 ;

SET @myVar3 = 4.25 ;

SET @myVar4 = '가수 이름==> ' ;

SELECT @myVar1 ;

SELECT @myVar2 + @myVar3 ;

SELECT @myVar4 , Name FROM userTbl WHERE height > 180 ;

SET @myVar1 = 3 ;

지속사용하는 쿼리 저장

PREPARE myQuery

    FROM 'SELECT Name, height FROM userTbl ORDER BY height LIMIT ?';

↳ 변수값 지정 대입.

EXECUTE myQuery USING @myVar1 ;

쿼리 실행
저장해둔
↳ ? 대입 변수 지정

-- </실습 1> --

USE sqlDB ;

SELECT AVG(amount) AS '평균 구매 개수' FROM buyTbl ;

SELECT CAST(AVG(amount) AS SIGNED INTEGER) AS '평균 구매 개수'   FROM buyTbl ;

    -- 또는

SELECT CONVERT(AVG(amount) , SIGNED INTEGER) AS '평균 구매 개수'   FROM buyTbl ;

SELECT CAST('2020$12$12' AS DATE);

SELECT CAST('2020/12/12' AS DATE);

SELECT CAST('2020%12%12' AS DATE);

SELECT CAST('2020@12@12' AS DATE);

값을 붙여 출력.

SELECT num, CONCAT(CAST(price AS CHAR(10)), 'X', CAST(amount AS CHAR(4)) ,'=')   AS '단가X수량',

1   2   3   4

    price*amount AS '구매액'

  FROM buyTbl ;

SELECT '100' + '200' ; -- 문자와 문자를 더함 (정수로 변환되서 연산됨)

SELECT CONCAT('100', '200'); -- 문자와 문자를 연결 (문자로 처리)

SELECT CONCAT(100, '200'); -- 정수와 문자를 연결 (정수가 문자로 변환되서 처리)

SELECT IF (100>200, '참이다', '거짓이다');

출력

SELECT IFNULL(NULL, '널이군요'), IFNULL(100, '널이군요');

NULL이면 ↲   NULL아니면 리턴

SELECT NULLIF(100,100), IFNULL(200,100);

두 인자 같으면 NULL리턴 , 다르면 첫번째인자 리턴

SELECT   CASE 10

          WHEN 1   THEN   '일'

          WHEN 5   THEN   '오'

          WHEN 10 THEN   '십'

```
        ELSE '모름'

    END;
```

```
SELECT BIT_LENGTH('abc'),   CHAR_LENGTH('abc'), LENGTH('abc'); -- length : byte 길이

SELECT BIT_LENGTH('가나다'),   CHAR_LENGTH('가나다'), LENGTH('가나다');


SELECT CONCAT_WS('/', '2020', '01', '01'); -- 구분자로 문자열을 이어준다.


SELECT ELT(2, '하나', '둘', '셋'), FIELD('둘', '하나', '둘', '셋'), FIND_IN_SET('둘', '하나,둘,셋'), INSTR('하나
둘셋', '둘'), LOCATE('둘', '하나둘셋');


SELECT FORMAT(123456.123456, 4);
```

⤷ 3번째부터 4개를 없애고   @@@@ 으로 채움

```
SELECT INSERT('abcdefghi', 3, 4, '@@@@'), INSERT('abcdefghi', 3, 2, '@@@@');


SELECT LEFT('abcdefghi', 3), RIGHT('abcdefghi', 3);


SELECT LOWER('abcdEFGH'), UPPER('abcdEFGH');


SELECT LPAD('이것이', 5, '##'), RPAD('이것이', 5, '##');


SELECT LTRIM('    이것이'), RTRIM('이것이    ');
```
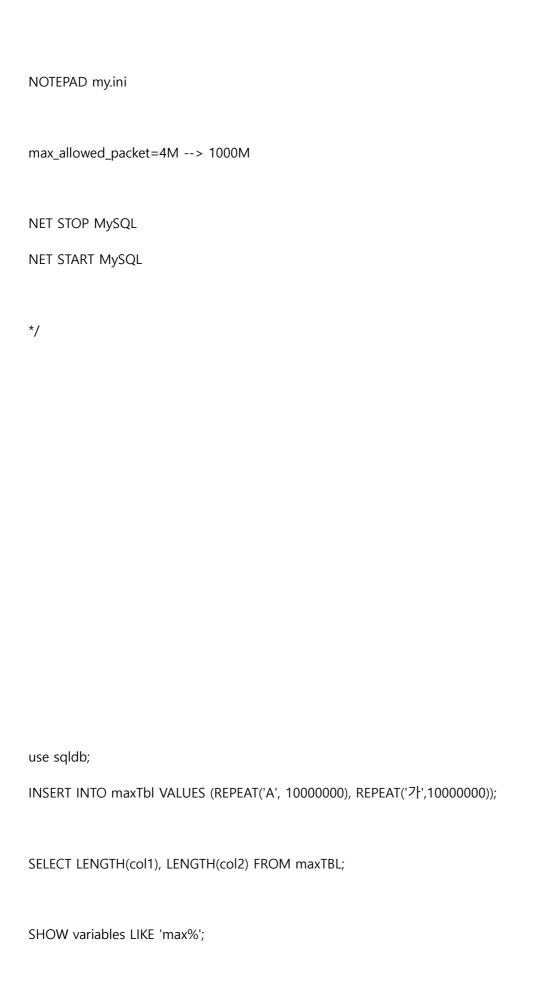
공백제거

```
SELECT REPEAT('이것이', 3);
```

3번 반복

SELECT REPLACE ('이것이 MySQL이다', '이것이' , 'This is');

SELECT REVERSE ('MySQL');

공백

SELECT CONCAT('이것이', SPACE(10), 'MySQL이다');

3번째자리휴터 2개 출력

SELECT SUBSTRING('대한민국만세', 3, 2);

SELECT SUBSTRING_INDEX('cafe.naver.com', '.', 2),   SUBSTRING_INDEX('cafe.naver.com', '.', -2);

SELECT ABS(-100);

올림 , 버림 , 반올림 .

SELECT CEILING(4.7), FLOOR(4.7), ROUND(4.7);

SELECT CONV('AA', 16, 2), CONV(100, 10, 8);

SELECT MOD(157, 10), 157 % 10, 157 MOD 10;   나머지

SELECT POW(2,3), SQRT(9);

SELECT SIGN(100), SIGN(0), SIGN(-100.123);

SELECT TRUNCATE(12345.12345, 2), TRUNCATE(12345.12345, -2);

USE   sqlDB;

CREATE TABLE maxTbl (col1 LONGTEXT,   col2 LONGTEXT);

INSERT INTO maxTbl VALUES (REPEAT('A', 1000000), REPEAT('가',1000000));

SELECT LENGTH(col1), LENGTH(col2) FROM maxTBL;

INSERT INTO maxTbl VALUES (REPEAT('A', 10000000), REPEAT('가',10000000));

```
/*
CD %PROGRAMDATA%

CD MySQL

CD "MySQL Server 5.7"

DIR
```

NOTEPAD my.ini

max_allowed_packet=4M --> 1000M

NET STOP MySQL

NET START MySQL

*/

use sqldb;

INSERT INTO maxTbl VALUES (REPEAT('A', 10000000), REPEAT('가',10000000));

SELECT LENGTH(col1), LENGTH(col2) FROM maxTBL;

SHOW variables LIKE 'max%';

```
/*

secure-file-priv=C:/TEMP

*/



USE sqlDB;

SELECT * INTO OUTFILE 'C:/TEMP/userTBL.txt' FROM userTBL;


CREATE TABLE memberTBL LIKE userTBL; -- 테이블 구조만 복사

LOAD DATA LOCAL INFILE 'C:/TEMP/userTBL.txt' INTO TABLE memberTBL;

select * from membertbl;
```
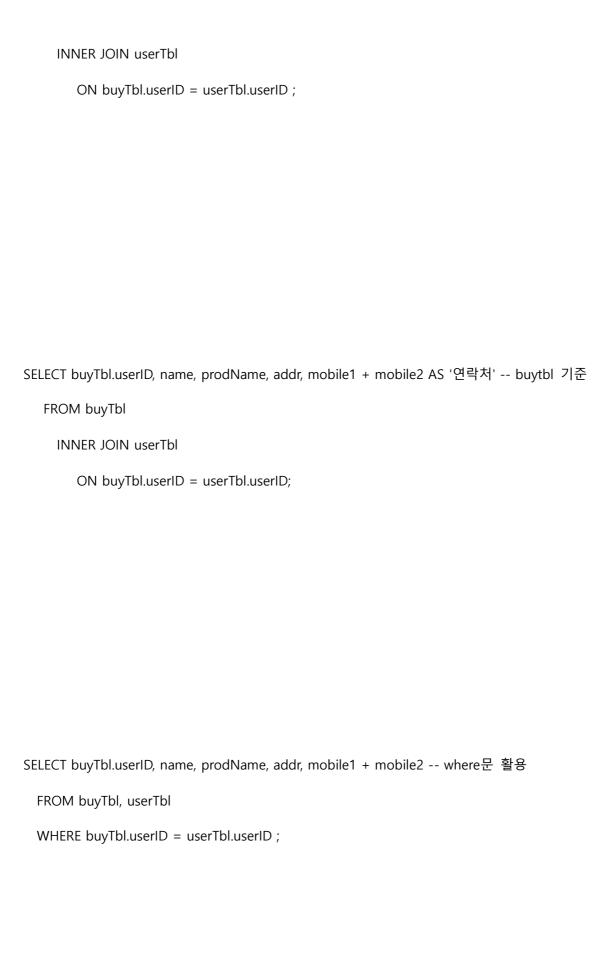
-- </실습 2> --

select * from buytbl;

select * from usertbl;

USE sqlDB; -- 구매자 주소 확인
SELECT *
    FROM buyTbl
        INNER JOIN userTbl
            ON buyTbl.userID = userTbl.userID
        WHERE buyTbl.userID = 'JYP';

SELECT userID, name, prodName, addr, mobile1 + mobile2 AS '연락처' -- 필요한 열만 추출 - error
    FROM buyTbl

```sql
        INNER JOIN userTbl
            ON buyTbl.userID = userTbl.userID ;
```

```sql
SELECT buyTbl.userID, name, prodName, addr, mobile1 + mobile2 AS '연락처' -- buytbl 기준
    FROM buyTbl
        INNER JOIN userTbl
            ON buyTbl.userID = userTbl.userID;
```

```sql
SELECT buyTbl.userID, name, prodName, addr, mobile1 + mobile2 -- where문 활용
    FROM buyTbl, userTbl
    WHERE buyTbl.userID = userTbl.userID ;
```

```sql
SELECT buyTbl.userID, userTbl.name, buyTbl.prodName, userTbl.addr,   -- 모두 테이블명
        userTbl.mobile1 + userTbl.mobile2   AS '연락처'
    FROM buyTbl
      INNER JOIN userTbl
        ON buyTbl.userID = userTbl.userID;
```

```sql
SELECT B.userID, U.name, B.prodName, U.addr, U.mobile1 + U.mobile2   AS '연락처'   -- 별칭
    FROM buyTbl B
      INNER JOIN userTbl U
        ON B.userID = U.userID;
```

```sql
SELECT B.userID, U.name, B.prodName, U.addr, U.mobile1 + U.mobile2   AS '연락처'

    FROM buyTbl B

       INNER JOIN userTbl U

           ON B.userID = U.userID

       WHERE B.userID = 'JYP';
```

```sql
SELECT U.userID, U.name, B.prodName, U.addr, U.mobile1 + U.mobile2   AS '연락처' -- 회원테이블
기준

    FROM userTbl U

       INNER JOIN buyTbl B

           ON U.userID = B.userID

       WHERE B.userID = 'JYP';
```

```sql
SELECT U.userID, U.name, B.prodName, U.addr, U.mobile1 + U.mobile2   AS '연락처'

    FROM userTbl U
```

```
        INNER JOIN buyTbl B

            ON U.userID = B.userID

        ORDER BY U.userID;
```

```
SELECT DISTINCT U.userID, U.name,   U.addr

    FROM userTbl U

        INNER JOIN buyTbl B

            ON U.userID = B.userID

        ORDER BY U.userID ;
```

-- <실습 4> --

```sql
USE sqlDB;

CREATE TABLE stdTbl
( stdName    VARCHAR(10) NOT NULL PRIMARY KEY,
   addr    CHAR(4) NOT NULL
);

CREATE TABLE clubTbl
( clubName     VARCHAR(10) NOT NULL PRIMARY KEY,
   roomNo     CHAR(4) NOT NULL
);

CREATE TABLE stdclubTbl
(   num int AUTO_INCREMENT NOT NULL PRIMARY KEY,
    stdName     VARCHAR(10) NOT NULL,
    clubName     VARCHAR(10) NOT NULL,
FOREIGN KEY(stdName) REFERENCES stdTbl(stdName),
FOREIGN KEY(clubName) REFERENCES clubTbl(clubName)
);

INSERT INTO stdTbl VALUES ('김범수','경남'), ('성시경','서울'), ('조용필','경기'), ('은지원','경북'),('바비킴','서울');

INSERT INTO clubTbl VALUES ('수영','101호'), ('바둑','102호'), ('축구','103호'), ('봉사','104호');

INSERT INTO stdclubTbl VALUES (NULL, '김범수','바둑'), (NULL,'김범수','축구'), (NULL,'조용필','축구'),
(NULL,'은지원','축구'), (NULL,'은지원','봉사'), (NULL,'바비킴','봉사');
```

```sql
select * from stdtbl;

select * from clubtbl;

select * from stdclubtbl;


SELECT S.stdName, S.addr, C.clubName, C.roomNo
    FROM stdTbl S
        INNER JOIN stdclubTbl SC
            ON S.stdName = SC.stdName
        INNER JOIN clubTbl C
        ON SC.clubName = C.clubName
    ORDER BY S.stdName;
```

```sql
SELECT C.clubName, C.roomNo, S.stdName, S.addr
    FROM   stdTbl S
        INNER JOIN stdclubTbl SC
            ON SC.stdName = S.stdName
        INNER JOIN clubTbl C
        ON SC.clubName = C.clubName
    ORDER BY C.clubName;
```

-- </실습 4> --

```
USE sqlDB;

SELECT U.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)   AS '연락처'

    FROM userTbl U

        LEFT OUTER JOIN buyTbl B

            ON U.userID = B.userID

    ORDER BY U.userID;
```

```sql
SELECT U.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)   AS '연락처'

    FROM buyTbl B

        RIGHT OUTER JOIN userTbl U

            ON U.userID = B.userID

    ORDER BY U.userID;
```

```sql
SELECT U.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)   AS '연락처'

    FROM userTbl U

        LEFT OUTER JOIN buyTbl B

            ON U.userID = B.userID

    WHERE B.prodName IS NULL

    ORDER BY U.userID;
```

-- <실습 5> --

USE sqlDB;

SELECT S.stdName, S.addr, C.clubName, C.roomNo

　　FROM stdTbl S

　　　　LEFT OUTER JOIN stdclubTbl SC

　　　　　　ON S.stdName = SC.stdName

　　　　LEFT OUTER JOIN clubTbl C

　　　　　　ON SC.clubName = C.clubName

　　ORDER BY S.stdName;

SELECT C.clubName, C.roomNo, S.stdName, S.addr

　　FROM　stdTbl S

　　　　LEFT OUTER JOIN stdclubTbl SC

　　　　　　ON SC.stdName = S.stdName

　　　　RIGHT OUTER JOIN clubTbl C

```
            ON SC.clubName = C.clubName

    ORDER BY C.clubName ;




SELECT S.stdName, S.addr, C.clubName, C.roomNo

    FROM stdTbl S

        LEFT OUTER JOIN stdclubTbl SC

            ON S.stdName = SC.stdName

        LEFT OUTER JOIN clubTbl C

            ON SC.clubName = C.clubName

UNION

SELECT S.stdName, S.addr, C.clubName, C.roomNo

    FROM   stdTbl S

        LEFT OUTER JOIN stdclubTbl SC

            ON SC.stdName = S.stdName

        RIGHT OUTER JOIN clubTbl C

            ON SC.clubName = C.clubName;
```

```
USE sqlDB;

SELECT stdName, addr FROM stdTbl   -- 중복된 열 포함/ union - 중복된 열 제거

    UNION ALL

SELECT clubName, roomNo FROM clubTbl;


SELECT name, CONCAT(mobile1, mobile2) AS '전화번호' FROM userTbl     -- 전화 없는 사람 제거

    WHERE name NOT IN ( SELECT name FROM userTbl WHERE mobile1 IS NULL) ;


SELECT name, CONCAT(mobile1, mobile2) AS '전화번호' FROM userTbl -- 전화 없는 사람 조회

    WHERE name IN ( SELECT name FROM userTbl WHERE mobile1 IS NULL) ;
```

```sql
DROP PROCEDURE IF EXISTS ifProc; -- 기존에 만든적이 있다면 삭제

DELIMITER $$

CREATE PROCEDURE ifProc()

BEGIN

    DECLARE var1 INT;   -- var1 변수선언

    SET var1 = 100;   -- 변수에 값 대입


    IF var1 = 100 THEN   -- 만약 @var1이 100이라면,

        SELECT '100입니다.';

    ELSE

        SELECT '100이 아닙니다.';

    END IF;

END $$

DELIMITER ;

CALL ifProc();
```

```
DROP PROCEDURE IF EXISTS ifProc3;

DELIMITER $$

CREATE PROCEDURE ifProc3()

BEGIN

    DECLARE point INT ;

    DECLARE credit CHAR(1);

    SET point = 77 ;


    IF point >= 90 THEN

        SET credit = 'A';

    ELSEIF point >= 80 THEN

        SET credit = 'B';

    ELSEIF point >= 70 THEN

        SET credit = 'C';

    ELSEIF point >= 60 THEN

        SET credit = 'D';

    ELSE

        SET credit = 'F';

    END IF;

    SELECT CONCAT('취득점수==>', point), CONCAT('학점==>', credit);

END $$
```

```
DELIMITER ;

CALL ifProc3();
```

```
DROP PROCEDURE IF EXISTS caseProc;

DELIMITER $$

CREATE PROCEDURE caseProc()

BEGIN

    DECLARE point INT ;

    DECLARE credit CHAR(1);

    SET point = 77 ;


    CASE

        WHEN point >= 90 THEN

            SET credit = 'A';

        WHEN point >= 80 THEN
```

```sql
            SET credit = 'B';

        WHEN point >= 70 THEN

            SET credit = 'C';

        WHEN point >= 60 THEN

            SET credit = 'D';

        ELSE

            SET credit = 'F';

    END CASE;

    SELECT CONCAT('취득점수==>', point), CONCAT('학점==>', credit);

END $$

DELIMITER ;

CALL caseProc();
```

```sql
use sqldb;

SELECT U.userID, U.name, SUM(price*amount) AS '총구매액',

    CASE

        WHEN (SUM(price*amount)   >= 1500) THEN '최우수고객'
```

```sql
        WHEN (SUM(price*amount)   >= 1000) THEN '우수고객'

        WHEN (SUM(price*amount) >= 1 ) THEN '일반고객'

        ELSE '유령고객'

    END AS '고객등급'

FROM buyTbl B

    RIGHT OUTER JOIN userTbl U

        ON B.userID = U.userID

GROUP BY U.userID, U.name

ORDER BY sum(price*amount) DESC ;
```

-- </실습 7> --

```sql
DROP PROCEDURE IF EXISTS whileProc;

DELIMITER $$

CREATE PROCEDURE whileProc()

BEGIN
```

```sql
DECLARE i INT; -- 1에서 100까지 증가할 변수

DECLARE hap INT; -- 더한 값을 누적할 변수

SET i = 1;

SET hap = 0;


WHILE (i <= 100) DO

    SET hap = hap + i;   -- hap의 원래의 값에 i를 더해서 다시 hap에 넣으라는 의미

    SET i = i + 1;          -- i의 원래의 값에 1을 더해서 다시 i에 넣으라는 의미

END WHILE;


SELECT hap;
END $$

DELIMITER ;

CALL whileProc();
```