

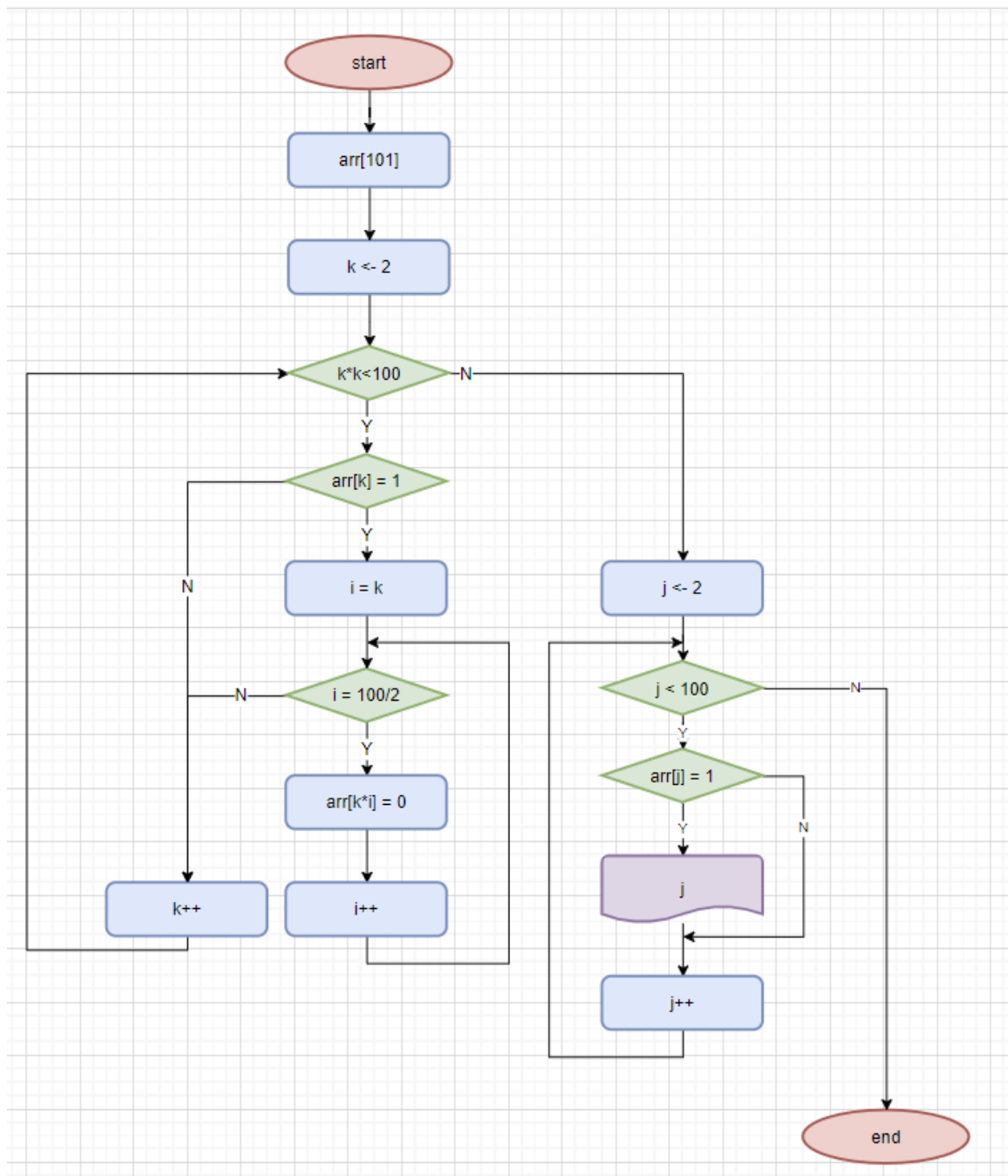
# Day27

🕒 작성일시	@2022년 8월 4일 오전 11:38
▼ 강의 번호	AUS-101
▼ 유형	
🔗 자료	
☑ 복습	<input type="checkbox"/>

## I . 알고리즘

### 1. 에라토스테네스의 체

어제 이론 공부 다했음



```

import java.util.Arrays;

public class SieveofEratosthenes {

    public static void main(String[] args) {

        int[] arr = new int[101];

        int k = 2;
  
```

```

while( k * k <100) {
    if (arr[k]==0) {
        int i = k;
        while(i<= 100/k) {
            arr[k*i] = 1;
            i++;
        }
    }
    k++;
}

for (int j = 2; j < 100; j++) {
    if (arr[j] == 0) {
        System.out.print(j + " ");
    }
}
}
}

```

## 2. 유클리드 알고리즘 (Euclidean Algorithms)

- 최대 공약수를 구하는 알고리즘
- 최대 공약수 : 약수들 중에서 가장 큰 수를 이야기 한다.

⇒ 반복 구조를 이용하는 중요한 알고리즘

### ▼ 최대 공약수

약수

3의 약수 - 1, 3

4의 약수 - 1, 2, 4

5의 약수 - 1, 5

6의 약수 - 1, 2, 3, 6

8의 약수 - 1, 2, 4, 8

12의 약수 - 1, 2, 3, 4, 6, 12

공약수 8과 12의 공통된 약수 - 1, 2, 4

최대공약수

두 수의 공약수 중 최대값, 8과 12의 최대 공약수는 4이다.

- 최대 공약수를 구하는 절차.

- 어떤 복수의 수를 **소수들의 곱셈 형태**로 분해하자. → 소인수 분해

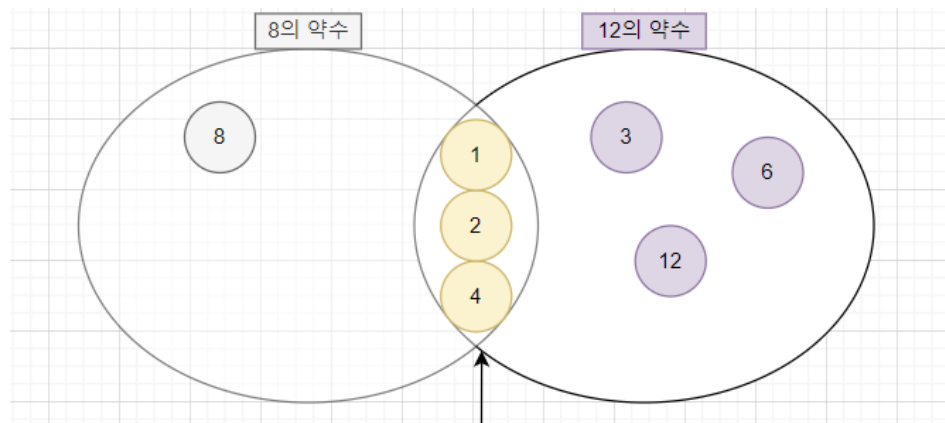
▼ 소인수 분해

$$8 = 1 * 2 * 2 * 2$$

$$12 = 1 * 2 * 2 * 3$$

→ 이들 중 공통 되는 소수를 서로 곱한 수가 바로 두 수의 최대 공약수이다.

$$1 * 2 * 2 = 4$$



- 그러나 이런 절차를 반복하는 것은 상당히 복잡하다. 어떤 수를 소인수 분해하려면 먼저 그 수 이하의 소수들을 모두 구해야 한다.

- 그리고 그 소수 중 작은 숫자부터 순서대로 원래의 수를 나누고, 나누어 지지 않으면 그 다음 소수의 순서로 계속 계산을 반복 해야 한다.

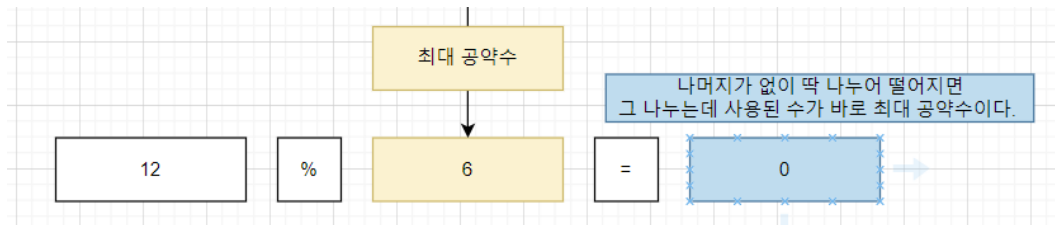
⇒ 이러한 복잡성에 비해 매우 간단한 방법으로 최대 공약수를 구하는 것이 바로 유클리드 알고리즘이다.

- 유클리드 알고리즘이란 ?

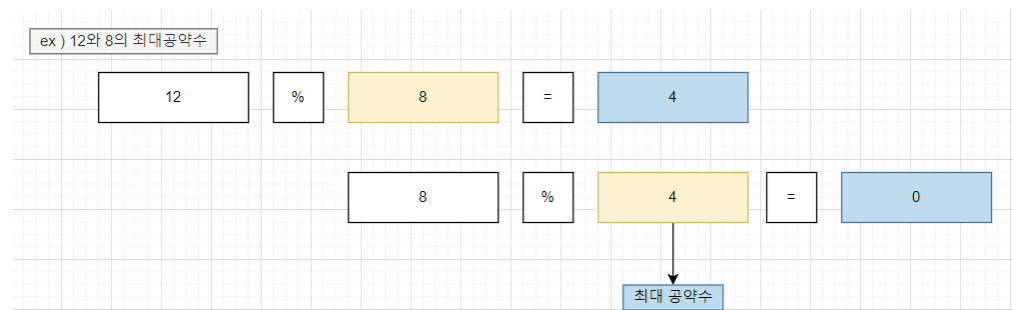
- 약 2300년 전의 고대 그리스의 수학자로 수많은 수학적 이론을 생각해냈다. 그 중 하나가 유클리드 알고리즘이다. 간단히 말하면 두 수의 나눗셈을 반복 하여 최대 공약수를 구하는 것이다.

- 나눗셈의 반복 : 먼저 큰 수를 작은 수로 나눈다. 나머지가 나오게 되면 (%) 그 때의 나누는데 사용된 작은 수가 바로 최대 공약수이다.

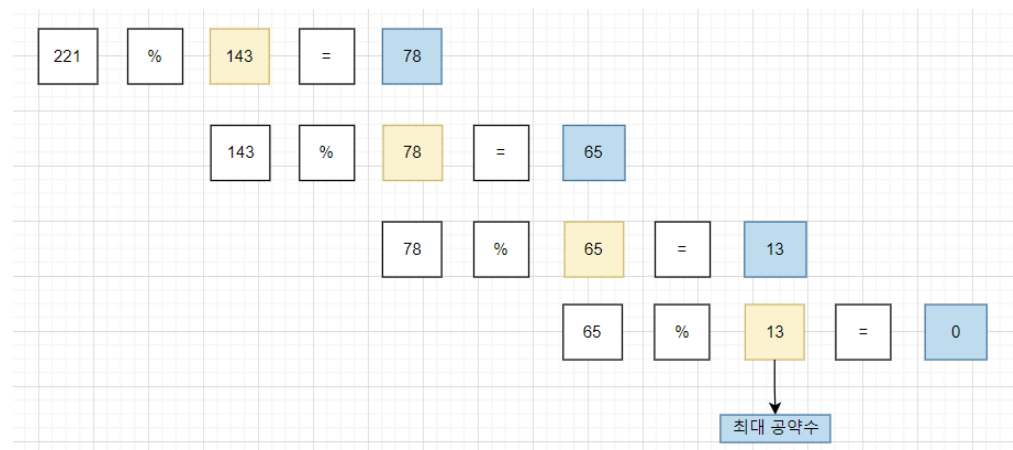
ex) 12와 6의 최대 공약수는 6이다.



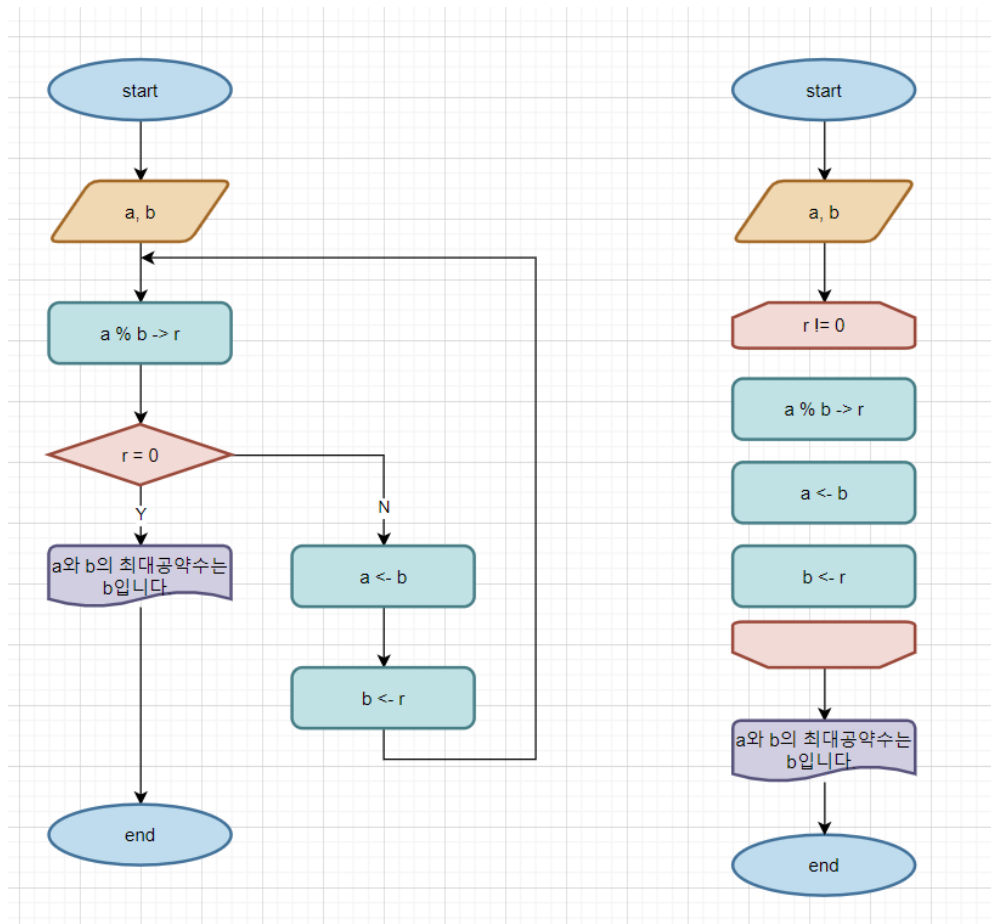
12와 8의 최대 공약수는 4이다



221과 143의 최대 공약수



- 알고리즘



```

import java.util.*;

public class Euclidean {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        System.out.println("두개의 숫자 입력. 첫번째 숫자가 더 커야 합니다.");

        System.out.println("첫번째 숫자");
        int a = scan.nextInt();
        System.out.println("두번째 숫자");
        int b = scan.nextInt();
        int r = 0;

        do {
            r = a % b;
            a = b;
            b = r;
        } while (r!=0);
        System.out.println("최대 공약수는 " + a + "입니다.");

    }

}

```

- Working with String Data 문자열 텍스트 변환
  - 테이블 생성

```
CREATE TABLE string_tbl
(char_fld CHAR(30),
vchar_fld VARCHAR(30),
text_fld TEXT
);
```

- 데이터 넣기

```
INSERT INTO string_tbl(char_fld, vchar_fld, text_fld)
VALUES ('This is char data',
       'This is varchar data',
       'This is text data');
```

```
SELECT vchar_fld
FROM String_tbl;
```

```
mysql> UPDATE string_tbl
-> SET text_fld = 'This string didn't work, but it does now';
/* 에러 난다. MYSQL 방식

UPDATE string_tbl SET text_fld =
'This string didn\'t work, but it does now'

오라클 방식 SQL에서도 먹힘 */
```

- Including special characters (특수문자)

```
SELECT CONCAT ('danke sch', CHAR(148), 'n');
```

/\* sql방식. 오라클은 || 로 쓴다.\*/

결과 #1 (1r x 1c)
CONCAT ('danke sch', CHAR(148), 'n')
danke sch백

- 문자열 조작

```
mysql> DELETE FROM string_tbl;  
Query OK, 1 row affected (0.02 sec)
```

```
mysql> INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)  
-> VALUES ('This string is 28 characters',  
-> 'This string is 28 characters',  
-> 'This string is 28 characters');  
Query OK, 1 row affected (0.00 sec)
```

⇒ 테이블의 모든 데이터를 삭제. (테이블은 그대로) /테이블 삭제까지 쓰려면 drop

- String functions that return numbers - 숫자를 반환하는 문자열 함수 ⇒ length() function

```
SELECT LENGTH(char_fld) CHAR_LENGTH,  
LENGTH(vchar_fld) varchar_length,  
LENGTH(text_fld) text_length  
FROM string_tbl;
```



1	SELECT LENGTH(char_fld) CHAR_LENGTH,
2	LENGTH(vchar_fld) varchar_length,
3	LENGTH(text_fld) text_length
4	FROM string_tbl;

결과 #1 (1r x 3c)		
CHAR_LENGTH	varchar_length	text_length
28	28	28

⇒ 앞에서 30글자로 char를 넣어 맞춰도 나올때는 글자로 나와서 28글자로 처리.

- 포지션

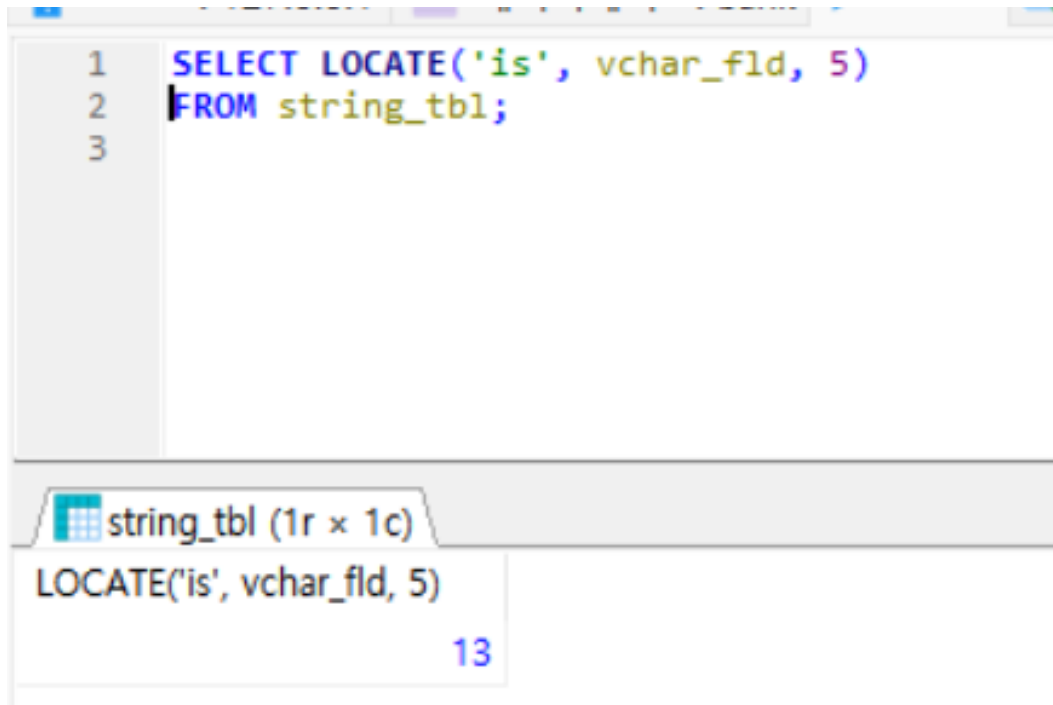
호스트: 127.0.0.1		데이터베이스: bank	쿼리*
1	SELECT POSITION('characters' IN vchar_fld)		
2	FROM STRING_tbl;		

STRING_tbl (1r x 1c)	
POSITION('characters' IN vchar_fld)	
19	

characters가 나타나는 위치를 보여준다. 반환이 없을때는 0을 반환함.

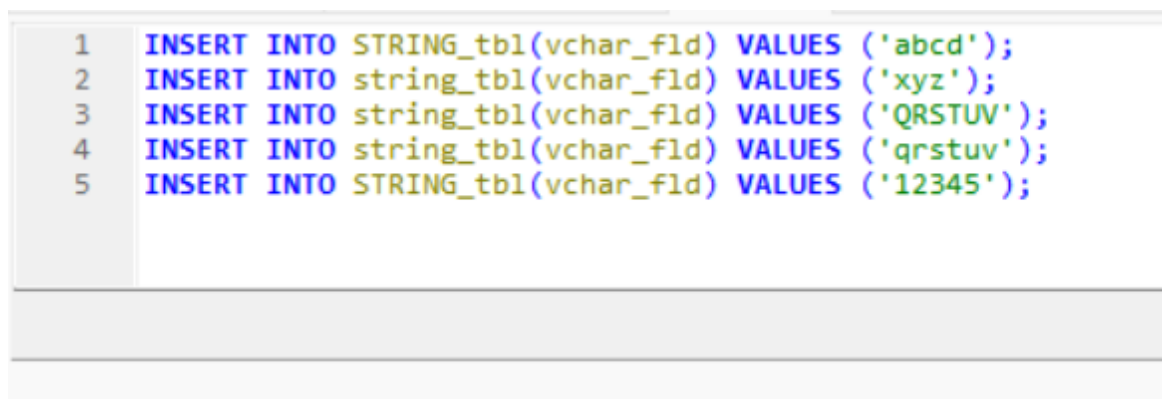
```
SELECT POSITION('characters' IN vchar_fld)
FROM STRING_tbl;
```



다섯번째 문자에서 시작하는 문자열 is의 위치를 찾는다. (오라클에서는 position()과 locate()를 사용할 수 없다.)

```
SELECT LOCATE('is', vchar_fld, 5)
FROM string_tbl;
```

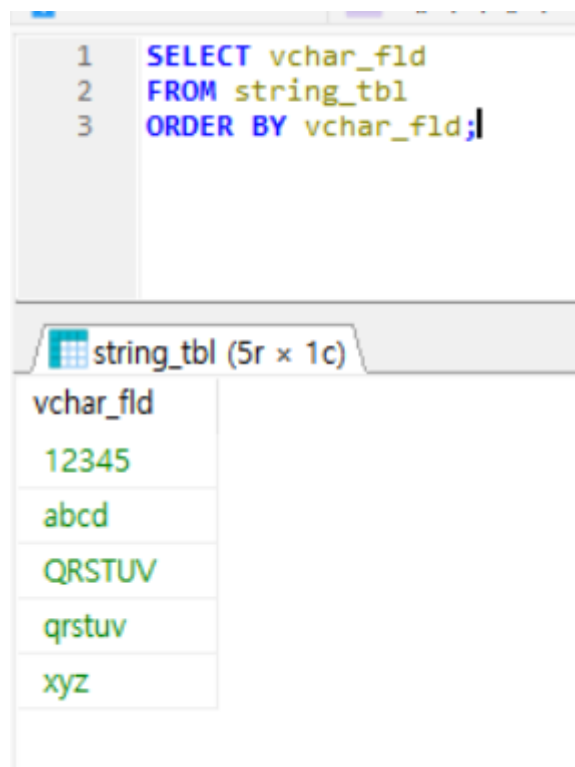
- 데이터 입력 : ~테이블에 어떤 값을 넣을건지.



```
INSERT INTO STRING_tbl(vchar_fld) VALUES ('abcd');
INSERT INTO string_tbl(vchar_fld) VALUES ('xyz');
INSERT INTO string_tbl(vchar_fld) VALUES ('QRSTUV');
```

```
INSERT INTO string_tbl(vchar_fld) VALUES ('qrstuv');  
INSERT INTO STRING_tbl(vchar_fld) VALUES ('12345');  
  
//외우기
```

- ORDER BY



```
1 SELECT vchar_fld  
2 FROM string_tbl  
3 ORDER BY vchar_fld;
```

vchar_fld
12345
abcd
QRSTUV
qrstuv
xyz

```
SELECT vchar_fld  
FROM string_tbl  
ORDER BY vchar_fld;
```

1	SELECT STRCMP('12345','12345') 12345_12345,
2	STRCMP('abcd','xyz') abcd_xyz,
3	STRCMP('abcd','QRSTUV') abcd_QRSTUV,
4	STRCMP('qrstuv','QRSTUV') qrstuv_QRSTUV,
5	STRCMP('12345','xyz') 12345_xyz,
6	STRCMP('xyz','qrstuv') xyz_qrstuv;

결과 #1 (1r x 6c)				
12345_12345	abcd_xyz	abcd_QRSTUV	qrstuv_QRSTUV	12345_xyz
0	-1	-1	0	-1

```
SELECT STRCMP('12345','12345') 12345_12345,
STRCMP('abcd','xyz') abcd_xyz,
STRCMP('abcd','QRSTUV') abcd_QRSTUV,
STRCMP('qrstuv','QRSTUV') qrstuv_QRSTUV,
STRCMP('12345','xyz') 12345_xyz,
STRCMP('xyz','qrstuv') xyz_qrstuv;
```

⇒ abcd가 xyz 작을게 앞으로 오므로 -1.

xyz이 prstuv 크게 앞에 올때는 1/

대소문자 구별이 없으므로 소문자 qrstuv이다.

왜 문자에 크기가 발생하고 비교가 되는걸까? ⇒ askii 에 따라서 구별이 된다.

- LIKE 사용

1	SELECT NAME, NAME LIKE '%ns' ends_in_ns
2	FROM department;

department (3r x 2c)	
NAME	ends_in_ns
Operations	1
Loans	1
Administration	0

⇒ 'ns'로 끝나면 1을 반환하고 그렇지 않으면 0을 반환한다.

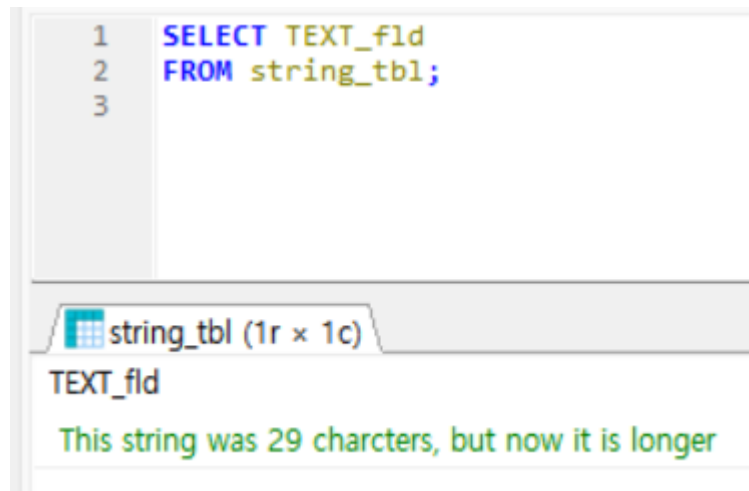
```
SELECT NAME, NAME LIKE '%ns' ends_in_ns
FROM department;
```

- 문자열을 반환하는 문자열 함수 (p139)

```
mysql> UPDATE string_tbl
-> SET text_fld = CONCAT(text_fld, ', but now it is longer');
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

The contents of the text\_fld column are now as follows:

```
mysql> SELECT text_fld
-> FROM string_tbl;
+-----+
| text_fld |
+-----+
| This string was 29 characters, but now it is longer |
+-----+
1 row in set (0.00 sec)
```



select → 임시로 바뀌서 보여주는것.

update → 진짜로 바뀌서 보여주는 것. (첫번째 concat을 사용해서 저 필드에 있는걸 바꿔버렸음)



```

SELECT CONCAT(fname, ' ', lname, 'has been a ',
title, ' since ' , start_date) emp_narrative
FROM employee
WHERE title = 'Teller' or title = 'Head Teller';

```

⇒ 문자열을 반환하는 함수

`concat()` 함수: 문자열에 저장된 데이터를 바꾸는 용도로 사용될 수 있다.

`update` - 영구변환

각 데이터 조각을 합쳐서 문자열을 만들어 임시로 보는 용도로도 사용된다

`selete` - 임시변환

※`concat`의 오라클 표현: ||

- Working with Numeric Data 숫자 데이터로 작업하기 (p142)

~~Performing Arithmetic Functions (산술함수, 안함.)~~

- mod 함수 (나머지 연산자)

```
mysql> SELECT MOD(10,4);
+-----+
| MOD(10,4) |
+-----+
|          2 |
+-----+
1 row in set (0.02 sec)
```

1	SELECT MOD(10,4);
결과 #1 (1r x 1c)	
MOD(10,4)	
2	

1	SELECT MOD(22.75,5);
결과 #1 (1r x 1c)	
MOD(22.75,5)	
2.75	

- 거듭제곱 pow 함수

1	SELECT POW(2,8);
결과 #1 (1r x 1c)	
POW(2,8)	256

- Controlling Number Precision (숫자 자리수 제어)

⇒ ceil() 올림, floor()버림, round()반올림, and truncate()버림.

- ceil() 올림, floor()버림

```
mysql> SELECT CEIL(72.445), FLOOR(72.445);
+-----+-----+
| CEIL(72.445) | FLOOR(72.445) |
+-----+-----+
|          73 |          72 |
+-----+-----+
1 row in set (0.06 sec)
```

1	SELECT CEIL(72.445), FLOOR(72.445);
2	
결과 #1 (1r x 2c)	
CEIL(72.445)	FLOOR(72.445)
73	72

- round() 반올림



```
mysql> SELECT ROUND(72.49999), ROUND(72.5), ROUND(72.50001);
+-----+-----+-----+
| ROUND(72.49999) | ROUND(72.5) | ROUND(72.50001) |
+-----+-----+-----+
|                72 |           73 |                73 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

1	SELECT ROUND (72.4999), ROUND(72.5), ROUND(72.999999999);		
2			
결과 #1 (1r x 3c)			
ROUND (72.4999)	ROUND(72.5)	ROUND(72.999999999)	
72	73	73	

⇒ 어제도 반올림은 했는데 어제는 자리값이 지정되어있었다. 만약에 round()함수에서 두번째 인수, 즉 반올림의 위치값을 생략하면 정수 값을 기준으로 자동 반올림 처리한다.

(위치값 있는 버전. 소수점 반올림하는 양수)

```
mysql> SELECT ROUND(72.0909, 1), ROUND(72.0909, 2), ROUND(72.0909, 3);
+-----+-----+-----+
| ROUND(72.0909, 1) | ROUND(72.0909, 2) | ROUND(72.0909, 3) |
+-----+-----+-----+
|                72.1 |           72.09 |           72.091 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

호스트: 127.0.0.1 | 데이터베이스: bank | 쿼리\*

1

2

SELECT ROUND (72.0909, 1), ROUND(72.0909, 2), ROUND(72.0909,3);

결과 #1 (1r x 3c)

ROUND (72.0909, 1)	ROUND(72.0909, 2)	ROUND(72.0909,3)
72.1	72.09	72.091

⇒ round() 함수의 두번째 인수인 위치값이 양수일때는 소수점 아래의 ~로 반올림하고, 음수일때는 소수점 위의 ~에서 반올림한다.

```
mysql> SELECT ROUND(17, -1), TRUNCATE(17, -1);
+-----+-----+
| ROUND(17, -1) | TRUNCATE(17, -1) |
+-----+-----+
|          20  |          10      |
+-----+-----+
1 row in set (0.00 sec)
```

1	SELECT ROUND (17, -1), TRUNCATE(17, -1);	
결과 #1 (1r x 2c)		
ROUND (17, -1)	TRUNCATE(17, -1)	
20	10	

```
SELECT ROUND (17, -1), TRUNCATE(17, -1);
```

## Working with Temporal Data 시간 데이터 작업

- Dealing with Time Zones 시간대 딜레이
  - ⇒ Greenwich Mean Time, or GMT, Coordinated Universal Time, or UTC. 우리나라 말로는 협정 세계 표준시. 서버가 위치한 시간대에 따른다.
- Generating Temporal Data 시간 데이터 생성

Table 7-2. Date format components

Component	Definition	Range
YYYY	Year, including century	1000 to 9999
MM	Month	01 (January) to 12 (December)
DD	Day	01 to 31
HH	Hour	00 to 23
HHH	Hours (elapsed)	-838 to 838
MI	Minute	00 to 59
SS	Second	00 to 59

분을 표현할때 소문자 mm을 쓰기도 한다.

Table 7-3. Required date components

Type	Default format
Date	YYYY-MM-DD
Datetime	YYYY-MM-DD HH:MI:SS
Timestamp	YYYY-MM-DD HH:MI:SS
Time	HHH:MI:SS

- String-to-date conversions 문자를 날짜 데이터로 변환.  
⇒ select cast () 함수 사용. 날짜처럼 보이는 문자를 진짜 날짜로 변경.

```
mysql> SELECT CAST('2008-09-17 15:30:00' AS DATETIME);
+-----+
| CAST('2008-09-17 15:30:00' AS DATETIME) |
+-----+
| 2008-09-17 15:30:00                      |
+-----+
1 row in set (0.00 sec)
```

```
1 SELECT CAST('2008-09-17 15:30:00' AS DATETIME);
```

결과 #1 (1r x 1c)

CAST('2008-09-17 15:30:00' AS DATETIME)
2008-09-17 15:30:00

```
SELECT CAST('2008-09-17 15:30:00' AS DATETIME);
```

```
mysql> SELECT CAST('2008-09-17' AS DATE) date_field,
->    CAST('108:17:57' AS TIME) time_field;
+-----+-----+
| date_field | time_field |
+-----+-----+
| 2008-09-17 | 108:17:57 |
+-----+-----+
1 row in set (0.00 sec)
```

```
1 SELECT CAST('2008-09-17 15:30:00' AS DATE) date_field,
2    CAST('108:17:57' AS TIME) time_field;
```

결과 #1 (1r x 2c)

date_field	time_field
2008-09-17	108:17:57

date 필드와 time 필드를 나누어서 작성

- Functions for generating dates 날짜 생성 관련 함수

⇒ cast() function 사용.

```
UPDATE individual
SET birth_date = STR_TO_DATE('September 17, 2008', '%M %d, %Y')
WHERE cust_id = 9999;
```

날짜 넣는 update

Table 7-4. Date format components

Format component	Description
%M	Month name (January to December)
%m	Month numeric (01 to 12)
%d	Day numeric (01 to 31)
%j	Day of year (001 to 366)
%W	Weekday name (Sunday to Saturday)
%Y	Year, four-digit numeric
%y	Year, two-digit numeric
%H	Hour (00 to 23)
%h	Hour (01 to 12)
%i	Minutes (00 to 59)
%s	Seconds (00 to 59)
%f	Microseconds (000000 to 999999)
%p	A.M. or P.M.

- CURRENT\_TIMESTAMP

```
mysql> SELECT CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP();
+-----+-----+-----+
| CURRENT_DATE() | CURRENT_TIME() | CURRENT_TIMESTAMP() |
+-----+-----+-----+
| 2008-09-18      | 19:53:12       | 2008-09-18 19:53:12 |
+-----+-----+-----+
1 row in set (0.12 sec)
```

```
1 SELECT CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP();
```

결과 #1 (1r x 3c)		
CURRENT_DATE()	CURRENT_TIME()	CURRENT_TIMESTAMP()
2022-08-04	16:41:10	2022-08-04 16:41:10

- 날짜 조작 함수
  - 날짜를 반환하는 함수

```
mysql> SELECT DATE_ADD(CURRENT_DATE(), INTERVAL 5 DAY);
+-----+
| DATE_ADD(CURRENT_DATE(), INTERVAL 5 DAY) |
+-----+
| 2008-09-22                                |
+-----+
1 row in set (0.06 sec)
```

1	SELECT DATE_ADD(CURRENT_DATE(), INTERVAL 5 day);	
결과 #1 (1r x 1c)		
DATE_ADD(CURRENT_DATE(), INTERVAL 5 day)		
2022-08-09		

오늘 날짜에서 5일을 더한 날짜 출력

Table 7-5. Common interval types

Interval name	Description
Second	Number of seconds
Minute	Number of minutes
Hour	Number of hours
Day	Number of days
Month	Number of months
Year	Number of years
Minute_second	Number of minutes and seconds, separated by ":"
Hour_second	Number of hours, minutes, and seconds, separated by ":"
Year_month	Number of years and months, separated by "-"

```
1 SELECT LAST_DAY('2008-09-17');
```

결과 #1 (1r x 1c)

LAST_DAY('2008-09-17')
2008-09-30

해당 달의 마지막 날 구하기 LAST\_DAY

- Temporal functions that return strings 문자열로 반환하는 시간 함수

```
mysql> SELECT DAYNAME('2008-09-18');
+-----+
| DAYNAME('2008-09-18') |
+-----+
| Thursday                |
+-----+
1 row in set (0.08 sec)
```

1	SELECT DAYNAME('1994-10-14');
결과 #1 (1r × 1c)	
	DAYNAME('1994-10-14')
	Friday

dayname(날짜)의 요일 추출.

1	SELECT EXTRACT(YEAR FROM '2008-09-18 22:19:05');
2008-09-18 22:19:05 (1r × 1c)	
	EXTRACT(YEAR FROM '2008-09-18 22:19:05')
	2,008

연도 추출

- Temporal functions that return numbers 숫자를 반환하는 시간 함수



1	SELECT DATEDIFF('2009-09-03', '2009-06-24');
결과 #1 (1r x 1c)	
DATEDIFF('2009-09-03', '2009-06-24')	71

두 날짜 사이의 기간을 계산. 이때 먼저 오는 날짜가 나중 날짜여야 한다. 그리고 시간은 무시하고 계산한다.

- Conversion Functions 변환 함수

- cast()를 사용하려면 값 또는 표현식 as 키워드 변환할 값의 자료형을 제공해야 한다.

```
mysql> SELECT CAST('1456328' AS SIGNED INTEGER);
+-----+
| CAST('1456328' AS SIGNED INTEGER) |
+-----+
|                                1456328 |
+-----+
1 row in set (0.01 sec)
```

자바 형변환의 개념. 위 예제는 문자를 숫자로 형변환 함 (int)

```
mysql> SELECT CAST('999ABC111' AS UNSIGNED INTEGER);
+-----+
| CAST('999ABC111' AS UNSIGNED INTEGER) |
+-----+
|                                999 |
+-----+
1 row in set, 1 warning (0.08 sec)
```

문자열 뒤에 숫자가 있으니 변환이 안되고 warning이 뜬다.

⇒ 문자열을 숫자로 변환할때 cast() 함수로는 전체 문자열을 왼쪽에서 오른쪽으로 변환을 시도 한다. 만약 숫자형 데이터가 아닌 문자가 있을 경우, 에러를 발생시키지는 않고 발견 되는 순간 변환을 중단시킨다.

## Grouping and Aggregates

- 그룹화의 개념



그룹 개념

```

1 SELECT open_emp_id
2 FROM account
3 GROUP BY open_emp_id;

```

account (4r × 1c)

open_emp_id	
1	
10	
13	
16	

전체 24개 행을 열린 emp id를 기준으로 그룹화 했다.

```

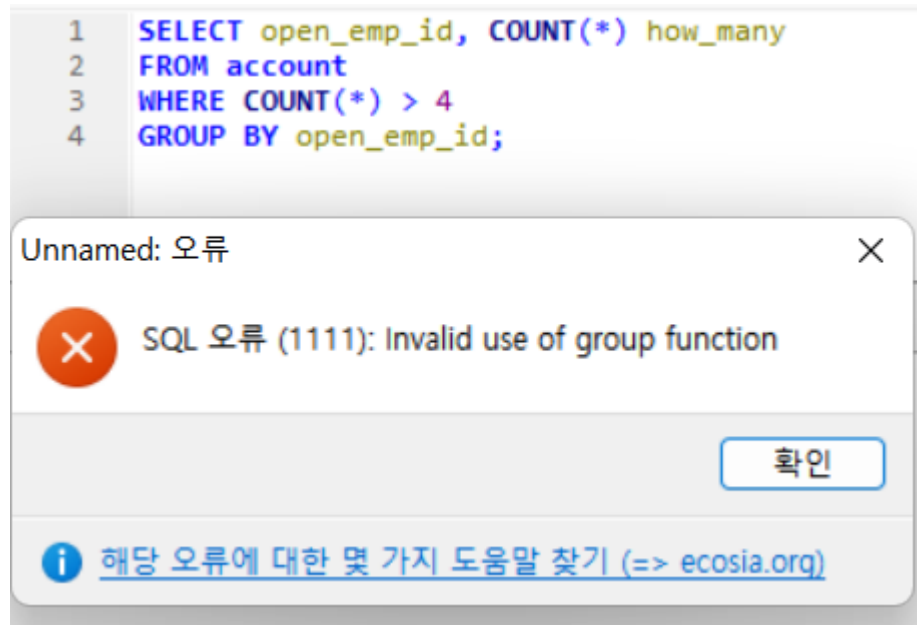
1 SELECT open_emp_id, COUNT(*) how_many
2 FROM account
3 GROUP BY open_emp_id;

```

account (4r × 2c)

open_emp_id		how_many
1		8
10		7
13		3
16		6

count(\*) → 집계함수. 같은 empID를 가지고 있는 것 들의 갯수를 센 것.



where 절이 적용 될 때 그룹이 아직 생성되지 않았기 때문에 where절에서 count(\*) 집계함수를 사용할 수 없다.

```
1 SELECT open_emp_id, COUNT(*) how_many
2 FROM account
3 GROUP BY open_emp_id
4 HAVING COUNT(*) > 4;
5
```

open_emp_id	how_many
1	8
10	7
16	6

대신 그룹 필터 조건을 having절에 넣어야 한다. 조건을 그룹 다음에 걸기 위해서.

- Aggregate Functions

1	SELECT MAX(avail_balance) MAX_balance,
2	MIN(avail_balance) min_balance,
3	AVG(avail_balance) avg_balance,
4	SUM(avail_balance) tot_balance,
5	COUNT(*) num_accounts
6	FROM account
7	WHERE product_cd = 'CHK';
8	

결과 #1 (1r x 5c)				
MAX_balance	min_balance	avg_balance	tot_balance	num_accounts
38,552.05	122.37	7,300.800985	73,008.01	10

이 쿼리의 결과는 account 테이블의 10가지 계정들을 체크해 교차한다. 맥시멈 밸런스는 \$38,552.05, 미니멈 밸런스는 \$122.37, 평균 밸런스는 \$7300.80,