Day24

Date	
작성일시	@2022년 8월 2일 오전 9:35
강의 번호	BD101
유형	Algorithm Data
강사명	AustinYoon
강의자료	☐
노션 복습	☐
코딩 복습	☐
주말숙제(교제)	☐
정리	☑

Day24

Aug

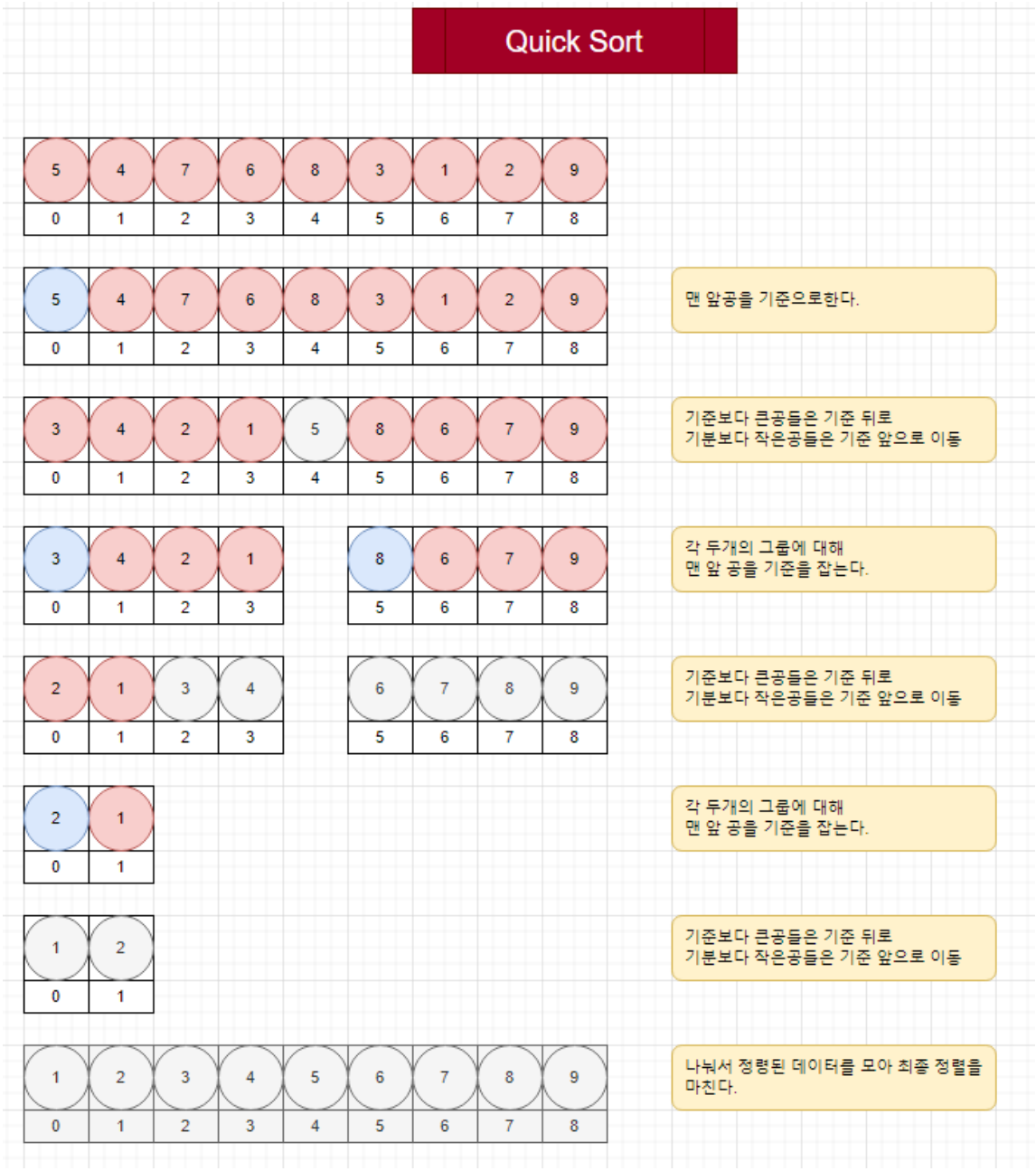
정렬

▼ 퀵 정렬_Quick Sort

- 데이터를 대소그룹 둘로 나누어 분해한 후에 전체를 최종적으로 정렬하는 알고리즘이다.

- Divide and conquer(분할 정복법)
- 퀵정렬은 대량의 데이터를 정렬할 때 매우 자주 사용된다.
- 유명한 알고리즘 중에서도 실제로 많이 사용되는 빈도가 가장 높고 중요한 알고리즘이기도 하다.
- 퀵정렬은 '기준값을 선택한 후 그 보다 작은 데이터 그룹과 큰데이터 그룹으로 나눈다.' 라는 처리를 반복수행하여 데이터를 정렬하게 된다.

▼ Process



▼ Algorithm

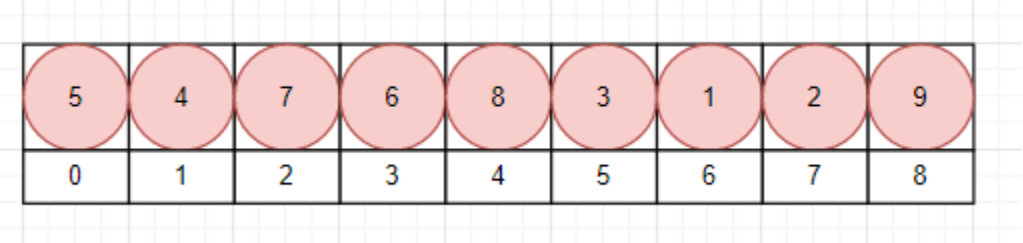
퀵 정렬은 크게 2개의 처리로 구성된다.

▼ (1) 기준값을 경계로 데이터를 대소로 나누는 처리

- 퀵정렬의 핵심은 데이터를 대소로 나누는 처리이다.
- 베열의 왼쪽과 오른쪽부터 각각 변수를 움직여 대소로 정렬하자.

👉 기준값보다 작은 공을 기준값의 앞으로 이동시키고 기준값보다 큰 공은 뒤로 이동시키는 것이 바로 퀵정렬의 초석이 되는 처리이다.

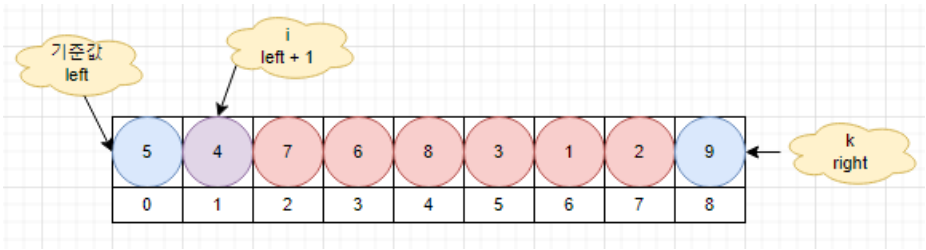
1. 배열설정 : 먼저 배열을 준비하자, 정수형 배열로 이름은 arr 요소의 수는 9개로 정한다.



⇒ 변수설정_변수는 5개

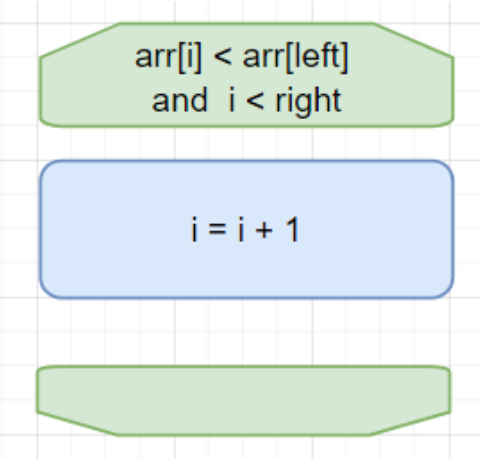
1. left - 정렬 범위에서 맨 앞 요소에 첨자를 넣는 변수
2. right - 정렬 범위에서 맨 끝 요소에 첨자를 넣는 변수
3. i - 기준값보다 큰 요소를 찾기 위한 변수
4. k - 기준값보다 작은 요소를 찾기 위한 변수
5. w - 데이터 교환용 임시 변수 temp

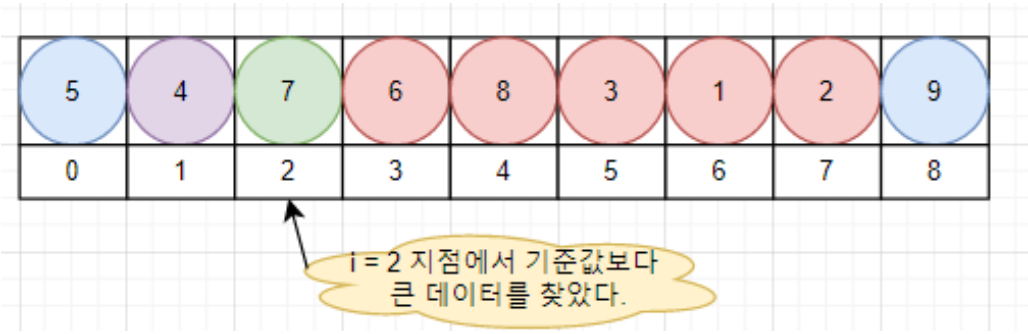
👉 이 5개의 변수를 사용하여 우선 left와 right에 각각 정렬 범위 맨 앞 요소의 첨자와 마지막 요소의 첨자를 대입한다. 따라서 이번에는 (처음에는) left 0, right 8이 된다. 기준은 맨 앞 요소로 하기 때문에 arr[left]가 된다. 그리고 i에 left의 하나 오른쪽에 left +1 로 정하고 k에는 right을 대입한다.



2. 변수 i를 사용하여 기준값보다 큰 요소 찾기

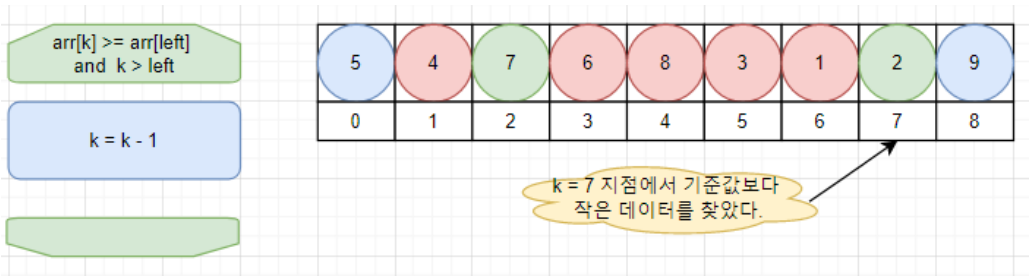
👉 i는 '기준갑보다 큰 요소를 찾는 변수'이다. 현재 위치에서 하나씩 오른쪽으로 이동하면서 기준값보다 큰 요소가 있는지 확인하고 발견되면 그곳에서 멈춘다.
⇒ arr[i] > arr[left]
*2가지 조건
1. arr[left]보다 큰 값을 찾고
2. 오른쪽 끝까지 반복.





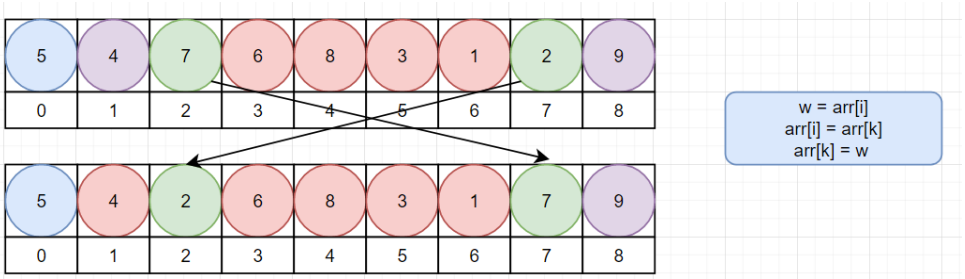
👉 기준값보다 큰 요소를 발견했기 때문에 i는 일단 여기서 멈춘다. 그대로 반대쪽 변수 k 즉 작은 값 찾기로 넘어간다.

3. 변수 k를 사용하여 기준값보다 작은 요소 찾기

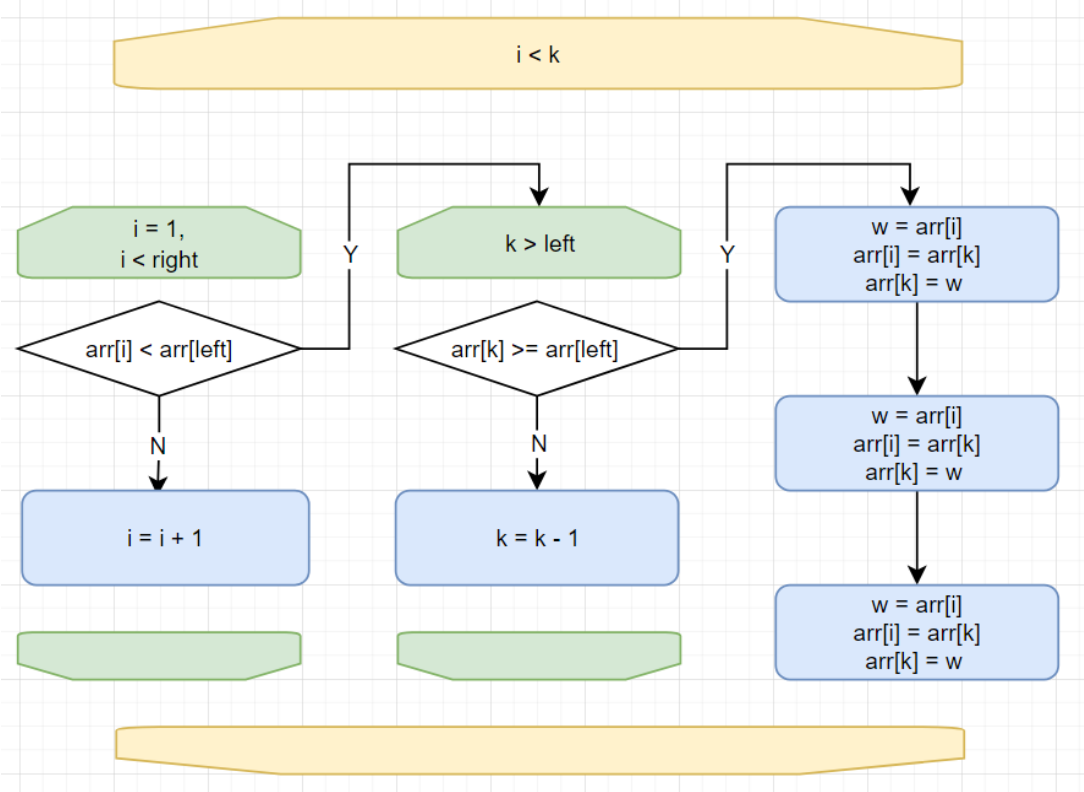


👉 기준값보다 작은 요소를 발견했기 때문에 k도 일단 여기서 멈춘다.

▼ 두 데이터를 교환



▼ (2) 나눈 데이터에 대해 반복적으로 똑같은 작업을 실행



▼ Java code

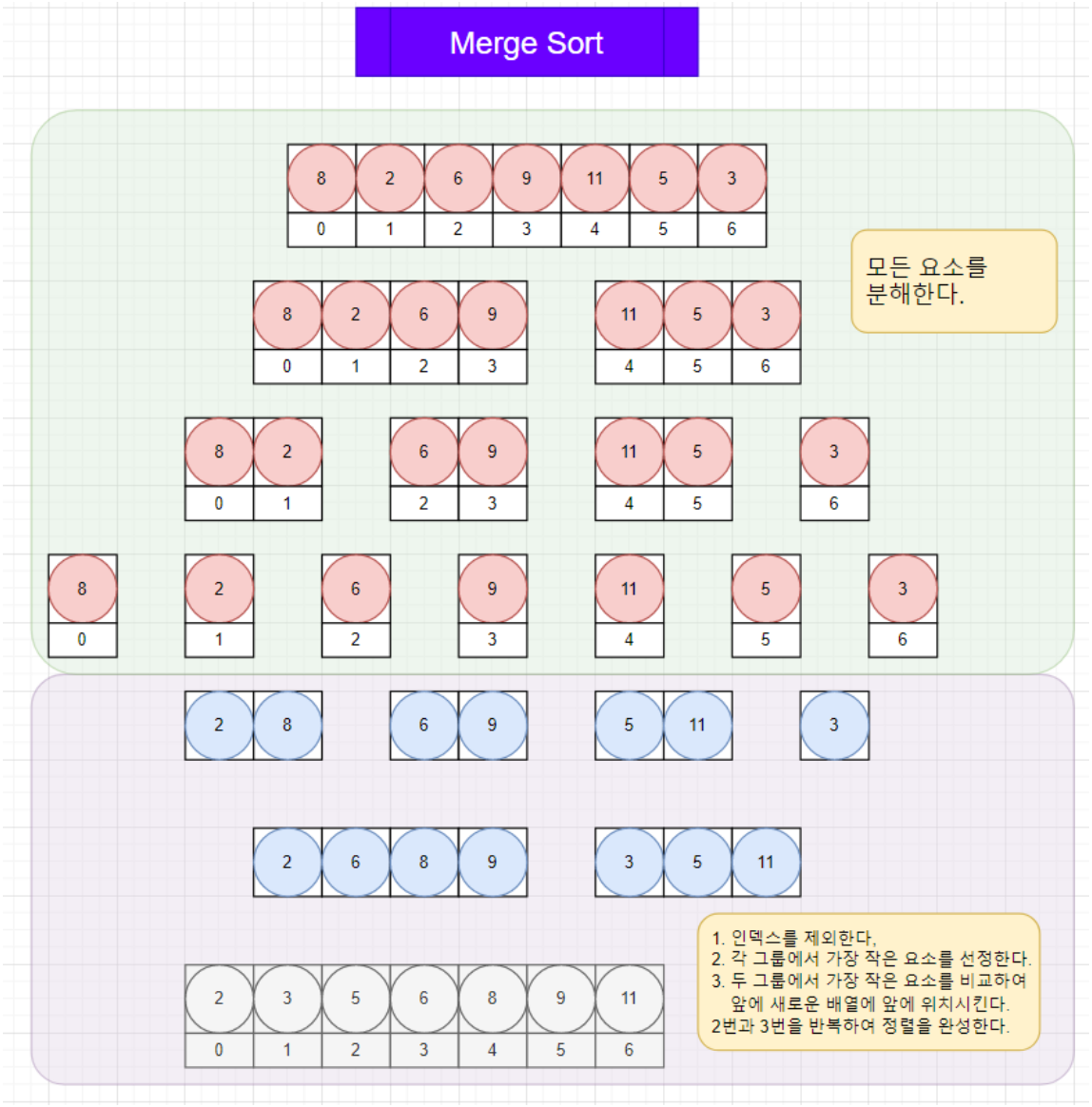
```
package QuickSort;

import java.util.*;

public class QuickSort {

  public static void main(String[] args) {
    int[] arr = {5,4,7,6,8,3,1,2,9};
    System.out.println(Arrays.toString(arr)); //배열 출력

    arr = quickSort(arr,0,arr.length-1);//어레이, 시작, 끝
    System.out.println(Arrays.toString(arr));
  }

  static int[] quickSort(int[] arr, int start, int end) {
    int p = partition(arr, start, end);

    if(start < p-1) {
      quickSort(arr, start, p-1);
    }else if(p < end) {
      quickSort(arr, p, end);
    }
    return arr;
  }
```

```
  static int partition(int[] arr, int start, int end) {
    int pivot = arr[(start+end)/2]; //기준값을 가운데로
    while(start <= end) {
      while(arr[start] < pivot) {
        start++;
      }
      while(arr[end]>pivot) {
        end--;
      }
      if (start <= end) {
        int tmp = arr[start];
        arr[start] = arr[end];
        arr[end] = tmp;
        start++;
        end--;
      }
    }
    return start;
  }
 }
Outputs :
[5, 4, 7, 6, 8, 3, 1, 2, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

▼ 병합 정렬_Merge Sort

👉 분할 정복(Divide and Conquer)을 사용하는 방법이다. 분할 정복은 주어진 문제를 해결하기 쉬운 단계까지 분할 한 후에 분할된 문제를 해결하고 그 결과를 다시 결합하는 알고리즘이다.

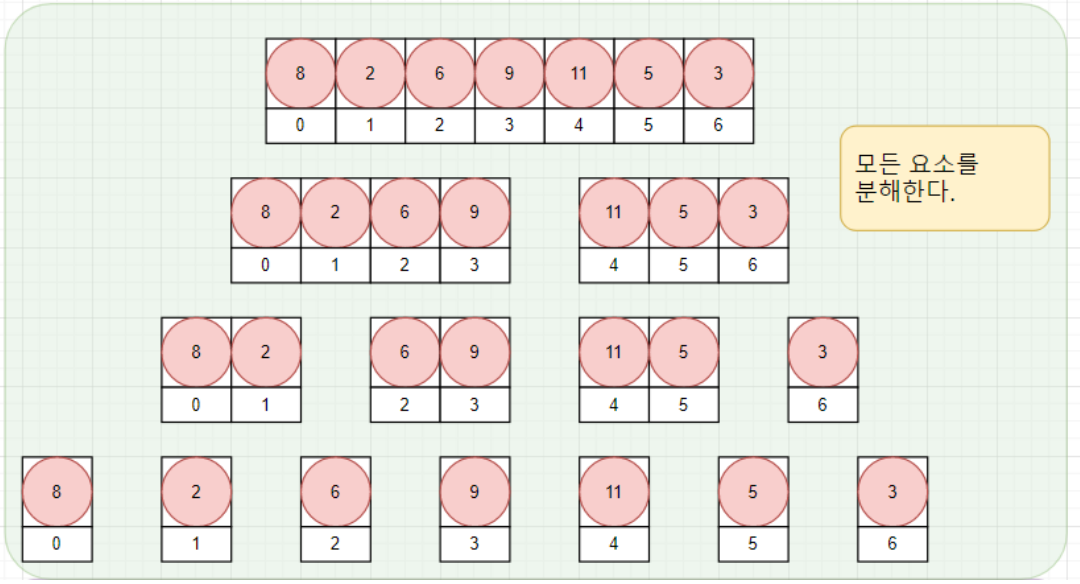
▼ Process



▼ Algorithm

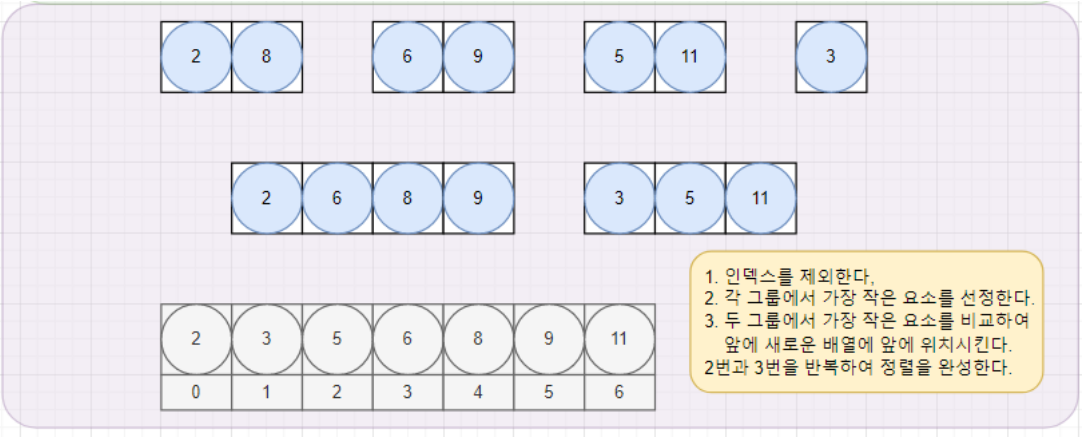
▼ (1) 데이터를 분해한다.

👉 정렬을 생각하지 않고 1개씩 될 때까지 나눈다. 나누는 방법은 배열을 반으로 분해하는 과정을 반복한다. 위에서는 7개의 데이터를 분해하기 위해 3번의 진행하였다.



▼ (2) 비교후 결합한다.

👉 분해가 완료된 후 데이터를 비교하면서 결합한다.
1. 제일 앞에 있는 2와 8을 다시 합치는데, 작은 수가 2가 앞으로 큰수 8이 뒤로 보낸 결합 상태가 된다.
2. 이때 각각의 그룹에서 먼저 제일 첫번째 값을 비교하여 작은 값은 추출한다.
3. 그 다음 다시 한번 각각의 그룹의 제일 왼쪽 즉 작은 값을 또 다시 비교하여 둘 줄 작은 값을 다시 추출한다.
4. 위의 과정을 반복한다.



▼ Java code

```

```

DB

Chapter4 : Filtering

- 새로운 데이터 웨어하우스 피드를 준비할 때 사용한 테이블에서 모든 데이터 제거
- 새열이 추가된 후 테이블의 모든 행 수정
- 메시지 큐 테이블에서 모든 행 검색.

조건 평가.

```
WHERE title = 'Teller' AND start_date < '2007-01-01'
--두가지 조건을 모두 and충족해야한다. (타이틀 텔러, 날짜 07년 1월1일 이전)
--조건이 여러개여도 AND연산자로 구분될 경우에는 결과셋에 모든 조건이 True인 경우만 포함
```

- 괄호 사용

Table 4-2. Three-condition evaluation using and, or

Intermediate result	Final result
WHERE true AND (true OR true)	True
WHERE true AND (true OR false)	True
WHERE true AND (false OR true)	True
WHERE true AND (false OR false)	False

👉 세가지 조건이 있을 경우 최종 결과는 괄호 안에
2가지 조건의 결과와 최종 마지막 결과의 평가에 따라 최종결과가 정해지게 된다.

- NOT 연산자

👉 NOT 연산자를 사용하여 평가 결과를

Table 4-3. Three-condition evaluation using and, or, and not

Intermediate result	Final result
WHERE true AND NOT (true OR true)	False
WHERE true AND NOT (true OR false)	False
WHERE true AND NOT (false OR true)	False
WHERE true AND NOT (false OR false)	True
WHERE false AND NOT (true OR true)	False
WHERE false AND NOT (true OR false)	False
WHERE false AND NOT (false OR true)	False
WHERE false AND NOT (false OR false)	False

반대로 바꾼다.
TRUE <=> FALSE

- 조건 작성
  - 숫자
  - 테이블 또는 뷰의 컬럼
  - 텔러와 같은 문자열
  - concat과 같은 내장 함수들
  - 서브쿼리, 헤드텔러 등등과 같은 표현식 목록

- 조건의 유형
  - 동등조건 : 열 = '값'
    - 동등조건을 활용한 데이터 수정

```
DELETE FROM account
WHERE status = 'CLOSED' AND YEAR(close_date) = 2002;
```

  - 부등조건 : 두 표현식이 부등하지 않을때 사용 열 <> '값'
  - 범위 조건

```
SELECT emp_id, fname, lname,
FROM employee
WHERE start_date < '2007-01-0
```

👉 특정 범위내에 원하는 조건이 있는지를 확인하는 범위조건을 작성할 수 있다.
이 유형은 보통 숫자 또는 시간

데이터로 작업할 때 주로 발생한다.

- between 연산자

```
SELECT emp_id, fname, lname,
FROM employee
WHERE start_date BETWEEN '200                                    L';
```

employee (2r × 4c)

emp_id	fname	lname	start_date
6	Helen	Fleming	2004-03-17
7	Chris	Tucker	2004-09-15

👉 범위의 상한과 하한이 모두 있을때 사용한 between 연산자를 사용하여 하나의 조건으로 사용할 수 있다.

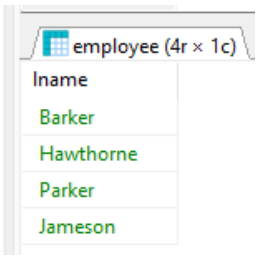
- 와일드 카드 사용하기
  - 특정문자로 시작/종료하는 문자열
  - 부분 문자열로 시작/종료하는 문자열
  - 문자열 내에 특정 문자를 포함하는 모든 문자열
  - 문자열 내에 특정 문자열을 포함하는 모든 문자열
  - 특정한 형식의 개별문자의 관계없이 특정한 형식의 문자열

Table 4-4. Wildcard characters

Wildcard character	Matches
_	Exactly one character
%	Any number of characters (including 0)

👉 _ 정확히 한글자
% 0을 포함한 개수에 상관 없이 모든 문자.
⇒ F% : F로 시작하는 모든 문자 _ _ f
_ : 두글자 사이에 f 그리고 한글자
%t : t로 끝나는 모든문자
%abs% : 어떤 글자 안에 abs 포힘

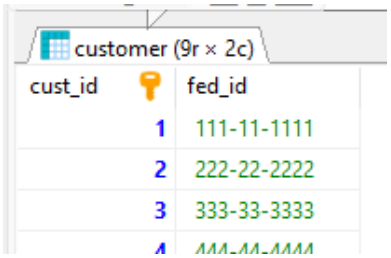
```
SELECT lname
FROM employee
WHERE lname LIKE '_a%e%';
```



employee (4r × 1c)

lname
Barker
Hawthorne
Parker
Jameson

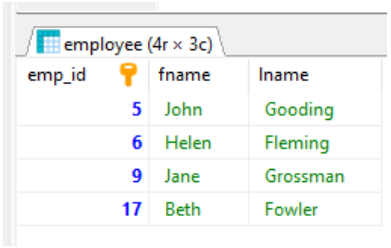
```
SELECT cust_id, fed_id
FROM customer
WHERE fed_id LIKE '___-__-____';
```



customer (9r × 2c)

cust_id	fed_id
1	111-11-1111
2	222-22-2222
3	333-33-3333
4	444-44-4444

```
SELECT emp_id, fname, lname
FROM employee
WHERE lname LIKE 'F%' OR lname LIKE 'G%';
```

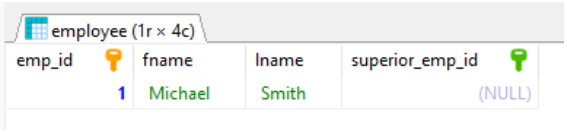


employee (4r × 3c)

emp_id	fname	lname
5	John	Gooding
6	Helen	Fleming
9	Jane	Grossman
17	Beth	Fowler

- Null
  - null일 수는 있지만 Null과 같을 수는 없다.

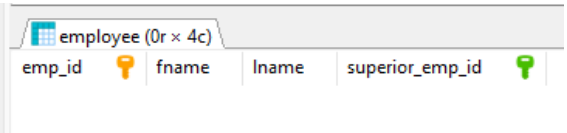
```
SELECT emp_id, fname, lname, superior_emp_id
FROM employee
WHERE superior_emp_id IS NULL;
```



employee (1r × 4c)

emp_id	fname	lname	superior_emp_id
1	Michael	Smith	(NULL)

```
SELECT emp_id, fname, lname, superior_emp_id
FROM employee
WHERE superior_emp_id = NULL;
```

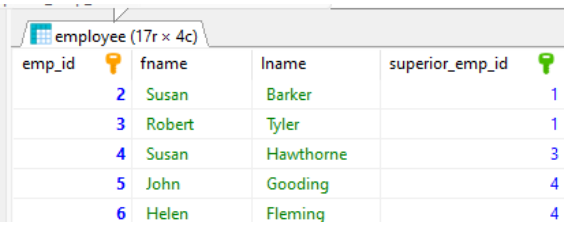


employee (0r × 4c)

emp_id	fname	lname	superior_emp_id

- 두개의 Null은 서로 같지 않다.

```
SELECT emp_id, fname, lname, superior_emp_id
FROM employee
WHERE superior_emp_id IS NOT NULL;
```



employee (17r × 4c)

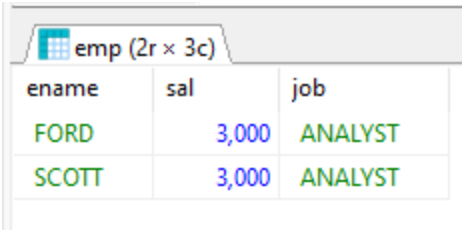
emp_id	fname	lname	superior_emp_id
2	Susan	Barker	1
3	Robert	Tyler	1
4	Susan	Hawthorne	3
5	John	Gooding	4
6	Helen	Fleming	4

문제풀이

Q7.

월급이 3000인 사원들의 이름, 월급, 직업을 출력해 보겠습니다.

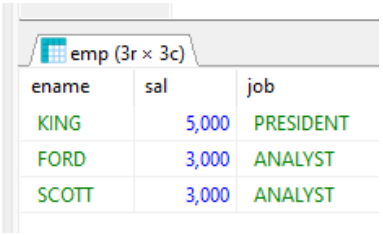
```
SELECT ename, sal, job
FROM emp
WHERE sal = 3000 ;
```



emp (2r × 3c)

ename	sal	job
FORD	3,000	ANALYST
SCOTT	3,000	ANALYST

```
SELECT ename, sal, job
FROM emp
WHERE sal >= 3000 ;
```



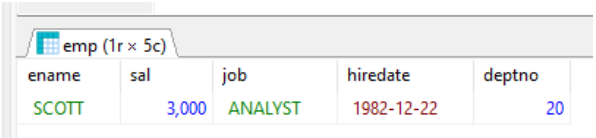
emp (3r × 3c)

ename	sal	job
KING	5,000	PRESIDENT
FORD	3,000	ANALYST
SCOTT	3,000	ANALYST

Q8.

이름이 SCOTT인 사원의 이름, 월급, 직업, 입사일, 부서 번호를 출력해 보겠습니다.

```
SELECT ename, sal, job, hiredate, deptno
FROM emp
WHERE ename = 'SCOTT';
```



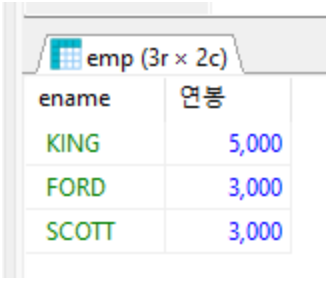
emp (1r × 5c)

ename	sal	job	hiredate	deptno
SCOTT	3,000	ANALYST	1982-12-22	20

Q9.

연봉이 36000 이상인 사원들의 이름과 연봉을 출력해 보겠습니다.

```
SELECT ename, (sal*12) AS 연봉
FROM emp
WHERE sal*12 >= 36000 ;
```



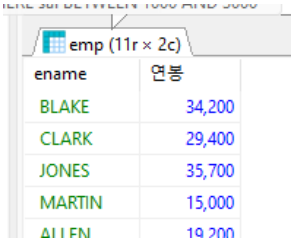
emp (3r × 2c)

ename	연봉
KING	5,000
FORD	3,000
SCOTT	3,000

Q10.

월급이 1000에서 3000 사이인 사원들의 이름과 월급을 출력해 보겠습니다.

```
SELECT ename, (sal*12) AS 연봉
FROM emp
WHERE sal BETWEEN 1000 AND 3000;
```



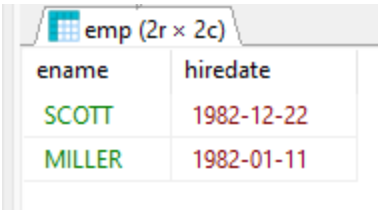
emp (11r × 2c)

ename	연봉
BLAKE	34,200
CLARK	29,400
JONES	35,700
MARTIN	15,000
ALLEN	19,200

Q11.

1982년도에 입사한 사원들의 이름과 입사일을 조회

```
SELECT ename, hiredate
FROM emp
WHERE hiredate BETWEEN '1982-01-01' AND '1982-12-31';
--WHERE hiredate LIKE '1982-__-__';
```



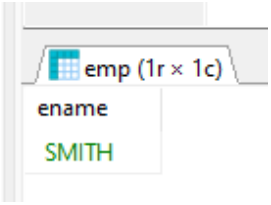
emp (2r × 2c)

ename	hiredate
SCOTT	1982-12-22
MILLER	1982-01-11

Q12.

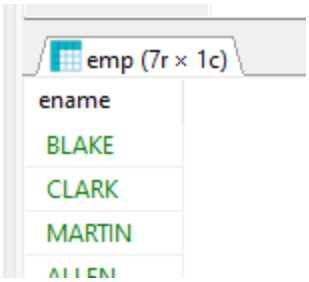
이름의 두 번째 철자가 M인 사원의 이름을 출력

```
SELECT ename
FROM emp
WHERE ename LIKE '_M%';
```



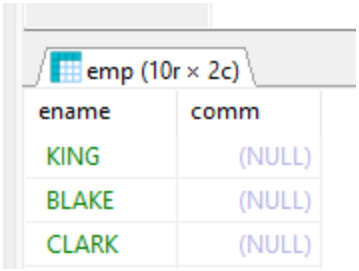
Q12-2 이름이 A가 포함된 사원들을 전부 검색

```
SELECT ename
FROM emp
WHERE ename LIKE '%A%';
```



Q13. 커미션이 NULL인 사원들의 이름과 커미션을 출력해 보겠습니다.

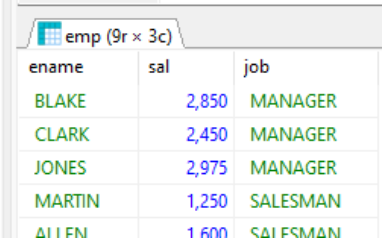
```
SELECT ename, comm
FROM emp
WHERE comm IS null;
```



Q14. 직업이 SALESMAN, ANALYST, MANAGER인 사원들의 이름, 월급, 직업을 출력해 보겠습니다.

```
SELECT ename, sal, job
```

```
FROM emp
WHERE job = 'SALESMAN' OR job ='ANALYST' OR job ='MANAGI
--WHERE job IN ('SALESMAN','ANALYST','MANAGER');
```



emp (9r × 3c)

ename	sal	job
BLAKE	2,850	MANAGER
CLARK	2,450	MANAGER
JONES	2,975	MANAGER
MARTIN	1,250	SALESMAN
ALLEN	1,600	SALESMAN

AND 연산자 진리 연산표

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR 연산자 진리 연산표

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT 연산자 진리 연산표

NOT	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL
