

Day29

⌚ 작성일시	@2022년 8월 9일 오전 9:36
✖ 강의 번호	
✖ 유형	
✖ 자료	
☑ 복습	<input type="checkbox"/>

I. 오라클

1. 그룹함수

- 단일행 함수와는 달리 여러 행에 대해 함수가 적용되어 하나의 결과를 나타내는 함수이다.
ex) 학생들의 영어점수들의 합계. 수학점수들의 평균
- 열에 대해 같은 값끼리 묶어서 묶은 행들의 집합에 대해 그룹연산이 필요할때는 GROUP BY 절을 사용하여 처리하고, 또 묶은 그룹에 다시 조건이 필요할 때는 HAVING 절을 이용한다.

! SELECT 그룹함수(열이름)

FROM 테이블이름

[WHERE 조건식]

[ORDER BY 열이름];

- (1) COUNT : 개수 (애만 null의 값도 개수로 센다. 나머지는 X)
- (2) SUM : 합계
- (3) AVG : 평균
- (4) MAX : 최대값
- (5) MIN : 최소값
- (6) STDDEV : 표준 편차
- (7) VARIANCE : 분산

Q1. employee테이블에서 salary의 행의 개수가 몇개 인지 세어서 출력하세요.

```
SELECT COUNT (salary)
FROM hr.employees;
```

```

1 SELECT COUNT (salary)
2 FROM hr.employees;

```

COUNT(SALARY)
107

[Download CSV](#)

Q2. Employees 테이블에서 salary의 합계와 평균을 구하세요. 또한 avg함수를 사용하지 말고 salary의 평균을 구하세요

```
SELECT SUM (salary)AS 합계, AVG (salary)AS 평균, SUM (salary) / count (salary) AS 계산평균
FROM hr.employees;
```

```

1 SELECT SUM (salary)AS 합계, AVG (salary)AS 평균, SUM (salary) / count (salary) AS 계산평균
2 FROM hr.employees;

```

합계	평균	계산평균
691416	6461.831775700934579439252336448598130841	6461.831775700934579439252336448598130841

Q3. employees 테이블에서의 salary의 최대값과 최소값, first_name의 최대값과 최소값을 출력해보세요.

```
SELECT MAX (salary) "최대 월급", MIN (salary) "최소 월급", MAX(first_name) "이름 최대", MIN(first_name) "이름 최소"
FROM hr.employees;
```

```

1 SELECT MAX (salary) "최대 월급", MIN (salary) "최소 월급", MAX(first_name) "이름 최대", MIN(first_name) "이름 최소"
2 FROM hr.employees;

```

최대 월급	최소 월급	이름 최대	이름 최소
24000	2100	Winston	Adam

[Download CSV](#)

- GROUP BY

- 특정 열의 데이터 값을 그룹으로 묶어 그룹 함수를 적용한다.
- SELECT 절에 열이름과 그룹 함수를 함께 기술 할 때에는 GROUP BY 절을 사용하게 된다. 그룹화는 열 이름이 기술된 순서대로 수행 된다.

! SELECT 기준열, 그룹함수(열이름)
 FROM 테이블 이름
 GROUP BY 열이름;

⇒ 테이블에 접근 하여(FROM) 기술된 열 이름을 기준으로 (GROUP BY) 선택 열들을 나열하고 지정된 열이름에 그룹함수를 적용하게 된다(SELECT).

Q4. employees 테이블에서 employee_id가 10 이상인 직원들에 대해서 / job_id 별로 그룹화 하여 / job_id별 총 급여와 job_id별 평균 급여를 구하고, / job_id별 총 급여를 기준으로 / 내림차순 정렬하세요.

```

SELECT job_id 직무, SUM (salary) "직무 별 총급여", AVG (salary) "직무별 평균 급여"
FROM hr.employees
WHERE employee_id >= 10
GROUP BY job_id
ORDER BY job_id DESC;

```

```

1 SELECT job_id 직무, SUM (salary) "직무 별 총급여", AVG (salary) "직무별 평균 급여"
2 FROM hr.employees
3 WHERE employee_id >= 10
4 GROUP BY job_id
5 ORDER BY job_id DESC;

```

직무	직무 별 총급여	직무별 평균 급여
ST_MAN	36400	7280
ST_CLERK	55700	2785
SH_CLERK	64300	3215
SA REP	250500	8350

• HAVING

- 그룹화된 값에 조건식을 적용할 때 사용된다.
- 즉, WHERE 절에서는 그룹함수를 사용할 수 없기 때문에 HAVING 절을 사용하여 그룹함수의 결과 값에 대해 조건식을 적용할 수 있다.

! SELECT 기준열, 그룹함수(열이름)
 FROM 테이블 이름
 [WHERE 조건식]
 GROUP BY 열이름
 [HAVING 조건식]
 [ORDER BY 열이름[ASC/DESC]];

Q5. employees 테이블에서 employee_id가 10 이상인 직원에 대해서 / job_id별로 그룹화 하여 / job_id별로 총급여와 job_id별 평균 급여를 구하되, / job_id별로 총급여가 30000 보다 큰 값만 출력하고, 결과는 job_id별로 총급여를 기준으로 /내림차순 정렬하세요.

```

SELECT job_id 직무, SUM (salary) "직무 별 총급여", AVG (salary) "직무별 평균 급여"
FROM hr.employees
WHERE employee_id >= 10
GROUP BY job_id
HAVING SUM (salary) > 30000
ORDER BY "직무 별 총급여" DESC;

```

```

1 SELECT job_id 직무, SUM (salary) "직무 별 총급여", AVG (salary) "직무별 평균 급여"
2 FROM hr.employees
3 WHERE employee_id >= 10
4 GROUP BY job_id
5 HAVING SUM (salary) > 30000
6 ORDER BY "직무 별 총급여" DESC;

```

직무	직무 별 총급여	직무별 평균 급여
SA_REP	250500	8350
SH_CLERK	64300	3215
SA_MAN	61000	12200
ST_CLERK	55700	2785
FI_ACCOUNT	39600	7920

2. JOIN (조인)

- 관계형 데이터 베이스의 의미는 테이블들이 관계를 맺고 조작되는 원리에서 유래되었다.
- 테이블들에는 각각에 성질에 맞는 데이터들이 저장되어 있고, 그 테이블들은 특정한 규칙에 따라 상호 관계를 맺는다.
→ 데이터는 여러 테이블에 흩어져 저장되어 있어 사용하가 원하는 대로 데이터를 사용하려면 특별한 방법이 필요하다.
- 조인은 한개 이상의 테이블과 테이블을 서로 연결하여 사용하는 기법**을 의미한다.
- 실무에서는 동등 조인, 외부 조인, 자체 조인 등을 사용하게 된다. (이 외에도 곱집합, 비동등조인 등 다양한 조인이 있다.)

! SELECT 테이블 명1.열이름1, 테이블명2.열이름2
 FROM 테이블명 1, 테이블명 2
 WHERE 테이블명1.열이름1 = 테이블명2.열이름2;

- 조인을 사용할 때 규칙
 - SELECT 절에는 출력할 열 이름들을 모두 기술한다
 - FROM 절에는 접근하려는 테이블들을 기술한다.
 - WHERE 절에는 조인 조건을 기술한다.
 - 사용되는 테이블 이름들에는 별칭을 alias를 사용하면 편하다.

(1) 동등 조인 : 똑같은 데이터끼리 연결

- 동등 조인은 양 테이블에서 조인 조건이 일치하는 행들만 가져오는 가장 일반적으로 자주 사용되는 조인이다.
- = (등호연산자)를 사용하여 조건 값이 정확하게 일치할 때만 행을 가져오기 때문에 inner join이라고도 부른다.

- 동등조인은 데이터 값이 정확히 일치하는 경우에만 결과를 출력한다.

→ 데이터 값이 일치하지 않으면 결과가 조회되지 않는다.

```
SELECT *
FROM hr.employees, hr.departments
WHERE employees.department_id = departments.department_id;
```

```
1 SELECT *
2 FROM hr.employees, hr.departments
3 WHERE employees.department_id = departments.department_id;|
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-03	AD_ASST	4400	-
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000	-
202	Pat	Fay	PFAY	603.123.6666	17-AUG-05	MK_REP	6000	-
114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-02	PU_MAN	11000	-
115	Alexander	Khoo	AKHOO	515.127.4562	18-MAY-03	PU_CLERK	3100	-

```
SELECT *
FROM hr.departments;
```

```
1 SELECT *
2 FROM hr.departments;
3
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400

Q6. employees 테이블과 departments 테이블과 location 테이블을 조인하여 각 직원이 어느 부서에 속하는지와 부서의 소재지가 어디인지 확인하시오.

```

1 SELECT *
2 FROM hr.employees;

```

NAME	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
James	AD_PRES	24000	-	-	90
John	AD_VP	17000	-	100	90
David	AD_VP	17000	-	100	90
Mark	IT_PROG	9000	-	102	60

```

1 SELECT *
2 FROM hr.departments;

```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400

LOCATION_ID	STREET_ADDRESS
1000	297 Via Cola di Rie
1100	3091 Calle de la Plata, T
1200	017 Shinjuku-ku
1300	450 Kamiya-cho
1400	014 Jabberwocky Land

```

SELECT employees.employee_id 사원아이디, departments.department_id 부서아이디, departments.department_name 부서명,
departments.location_id 지역아이디, locations.city 도시
FROM hr.employees, hr.departments, hr.locations
WHERE employees.department_id = departments.department_id
AND departments.location_id = locations.location_id;

```

```

1 SELECT employees.employee_id 사원아이디, departments.department_id 부서아이디, departments.department_name 부서명,
2 departments.location_id 지역아이디, locations.city 도시
3 FROM hr.employees, hr.departments, hr.locations
4 WHERE employees.department_id = departments.department_id
5 AND departments.location_id = locations.location_id;
6

```

사원아이디	부서아이디	부서명	지역아이디	도시
100	90	Executive	1700	Seattle
101	90	Executive	1700	Seattle
102	90	Executive	1700	Seattle
103	60	IT	1400	Southlake
104	60	IT	1400	Southlake
105	60	IT	1400	Southlake

```

1 SELECT count(*) "조인된 건수"
2 FROM hr.employees E, hr.departments D
3 WHERE E.department_id = D.department_id;

```

조인된 건수

106

```

1 SELECT count(*) "건수"
2 FROM hr.employees E;

```

건수

107

- ⇒ 동등조인의 결과는 106건이 출력 되었는데 employees의 직원 정보는 모두 107건이다.
- ⇒ 1건의 차이가 나는 이유는 일치하지 않는 1건이 누락되었기 때문이다.

(2) 외부조인 : 모든 데이터를 연결

- 외부 조인(outer join)은 조건을 만족하지 않는 행도 모두 출력하는 조인 기법이다
- 외부 조인은 동등 조인 조건을 만족하지 못해 누락되는 행을 출력하기 위해 사용된다.
- 외부 조인은 (+) 기호를 사용한다. + 기호는 조인할 행이 없는 즉, 데이터가 부족한 테이블의 열 이름 뒤에 기술한다
- +기호는 외부 조인하려는 한쪽에만 기술할 수 있다. 테이블 양쪽에 모두 기술할 수는 없다.
- + 기호를 붙이면 데이터 값이 부족한 테이블에 null 값을 갖는 행이 생성되어 데이터 값이 충분한 테이블의 행들이 null행에 조인된다.

- 외부 조인은 동등조인과 함께 실무에서 가장 많이 사용되는 조인이다.
 - 양쪽 테이블 중 전부 출력하고 싶은 테이블 쪽을 앞에 둔다.
 - (+)는 다른쪽(뒤쪽) 테이블 조인조건에 붙인다.

```
SELECT count(*) "조인된 건수"
FROM hr.employees E, hr.departments D
WHERE E.department_id = d.department_id(+);
```

```
1 SELECT count(*) "조인된 건수"
2 FROM hr.employees E, hr.departments D
3 WHERE E.department_id = d.department_id(+);|
```

조인된 건수
107

[Download CSV](#)

(3) 자체 조인 (self join)

- employees 테이블의 직원 정보에는 manager_id 열이 있다. 이 열은 그 직원의 담당 매니저의 정보를 담고 있는 열이다.
- 이 경우, 어떤 직원의 담당 매니저를 알고 싶으면 그 테이블에 결국 다시 조인해야 한다.
⇒ 이렇게 자기 자신의 테이블을 조인하는 것을 자체 조인 self join이라고 한다.

! SELECT 열이름1, 열이름2, 열이름3
 FROM 테이블이름1 별명1, 테이블1 별명2
 WHERE 테이블.열이름1 = 테이블1.열이름2

⇒ 한개의 테이블의 별명을 2개로 각각 다른 이름으로 설정한다.

Q7. employees 테이블을 자체조인하여 직원 별 담당 매니저가 누구인지를 조회하세요.

```
SELECT e.employee_id "사원아이디", e.first_name || e.last_name "사원 이름",
m.manager_id "매니저 아이디", m.first_name|| m.last_name "매니저 이름"
FROM hr.employees e, hr.employees m
WHERE e.employee_id = m.employee_id;
```

```

1  SELECT e.employee_id "사원아이디", e.first_name || e.last_name "사원 이름",
2  m.manager_id "매니저아이디", m.first_name|| m.last_name "매니저 이름"
3  FROM hr.employees e, hr.employees m
4  WHERE e.employee_id = m.employee_id;
5

```

사원아이디	사원 이름	매니저아이디	매니저 이름
100	StevenKing	-	StevenKing
101	NeenaKochhar	100	NeenaKochhar
102	LexDe Haan	100	LexDe Haan

3. 서브쿼리

- SELECT 문 안에 SELECT
- 서브쿼리는 SELECT 구문 안에서 보조로 사용되는 또 다른 SELECT 구문이다.
- SELECT구문을 효율적으로 작성할 수 있도록 도와주며, 복잡한 SELECT 구문을 작성할 때 필수적으로 많이 사용되는 기법이다.

! SELECT 열이름1, 열이름2, 열이름3...
 FROM 테이블 이름
 WHERE 조건식
 (SELECT 열이름1, 열이름2, 열이름3...
 FROM 테이블 이름
 WHERE 조건식)

- 서브 쿼리는 논리가 복잡한 SQL 구문에서는 거의 필수로 많이 사용한다.
 - 서브쿼리는 ()괄호로 묶어서 사용한다. 메인은 묶지 않는다.
 - 메인쿼리와 서브쿼리를 연결하기 위해 단일행 연산자 또는 다중행 연산자를 사용한다.
 - 서브쿼리(괄호안)가 먼저 실행되고 그 다음 메인 쿼리가 실행된다.
 - 서브쿼리의 서브쿼리의 서브쿼리 형태로 계속 중첩이 가능하다.
 - 서브쿼리의 종류
 - (1) 단일행 서브쿼리 : 하나의 행을 검색하는 서브쿼리
 - 서브쿼리 SELECT문에서 얻은 한개의 행의 결과 값을 메인 쿼리로 전달하는 서브쿼리이다.
- Q8. employees 테이블의 last_name이 'De Haan'인 직원과 salary가 동일한 직원을 단일행 서브쿼리를 이용하여 검색 후 출력하시오.

```

1 | SELECT e.salary
2 | FROM hr.employees e
3 | WHERE e.last_name = 'De Haan';|

```

SALARY
17000

[Download CSV](#)

⇒ 이 전체를 서브쿼리로 놓는다.

```

SELECT *
FROM hr.employees e
WHERE salary = (
    SELECT e.salary
    FROM hr.employees e
    WHERE e.last_name = 'De Haan'
);

```

```

1 | SELECT *
2 | FROM hr.employees e
3 | WHERE salary = (
4 |     SELECT e.salary
5 |     FROM hr.employees e
6 |     WHERE e.last_name = 'De Haan'
7 | );

```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-

[Download CSV](#)

2 rows selected.

Q9. employees 테이블의 last_name이 'Taylor'인 직원과 salary가 동일한 직원에 대한 모든 정보를 단일행 서브쿼리를 이용하여 검색하여 출력하시오.

⇒ 예러 발생. 서브쿼리에서 보낼 데이터가 1개 이상이기 때문에. 다중행 서브쿼리에서 처리.

(2) 다중행 서브쿼리: 하나 이상의 행을 검색하는 서브쿼리

- 사용법은 단일행과 동일하다.
- 다중행 서브쿼리는 하나 이상의 결과를 메인 쿼리에 전달하는 경우에 사용된다

- 사용하는 연산자

- IN : 같은 값이 여러개 \Rightarrow IN(1, 2, 3)
- NOT IN : 같은 값이 아닌 여러개 \Rightarrow NOT IN (1, 2, 3)
- EXIST : 값이 있으면 True 없으면 False
- ANY : 최소한 하나라도 존재 (or)
- ALL : 모두 만족 (and)

Q9. employees 테이블에서 department_id 별로 가장 낮은 salary 가 얼마인지 찾아서 그에 해당하는 직원이 누구인지 다중행 서브 쿼리를 이용하여 찾기.

```
1 SELECT e.department_id, MIN(salary)
2 FROM hr.employees e
3 GROUP BY department_id;
4 |
```

DEPARTMENT_ID	MIN(SALARY)
50	2100
40	6500
110	8300
90	17000
30	2500
70	10000

```
SELECT *
FROM hr.employees e
WHERE e.salary IN (
    SELECT MIN(salary)
    FROM hr.employees e
    GROUP BY department_id
);
```

```

1 SELECT *
2 FROM hr.employees e
3 WHERE e.salary IN (
4     SELECT MIN(salary)
5         FROM hr.employees e
6             GROUP BY department_id
7 );
8

```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	
113	Luis	Popp	LPOPP	515.124.4567	07-DEC-07	FI_ACCOUNT	6900	
119	Karen	Colmenares	KCOLMENA	515.127.4566	10-AUG-07	PU_CLERK	2500	
123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-OCT-05	ST_MAN	6500	
131	James	Marlow	JAMRLOW	650.124.7234	16-FEB-05	ST_CLERK	2500	
132	Teja	Stiles	TSTILES	650.124.8234	10-APR-07	ST_CLERK	2100	

(3) 다중열 서브쿼리 : 하나 이상의 열을 검색하는 서브쿼리.

- 메인 쿼리와 서브 쿼리를 비교하는 WHERE 조건식에서 비교되는 열이 여러개일 경우 사용되는 서브쿼리이다.

! SELECT 열이름1, 열이름2, 열이름3...
 FROM 테이블 이름
 WHERE (열이름1, 열이름2, 열이름3...) IN
 (SELECT 열이름1, 열이름2, 열이름3...
 FROM 테이블 이름
 WHERE 조건식)

Q10. employees 테이블에서 job_id별로 가장 높은 salary가 얼마인지 찾아보고, 찾아낸 job_id별 salary에 해당하는 직원이 누구인지 다중열 서브쿼리를 사용해 검색.

```

SELECT *
FROM hr.employees e
WHERE (e.job_id, e.salary) IN
(
    SELECT e.job_id, MAX(salary)
    FROM hr.employees e
    GROUP BY e.job_id
)
ORDER BY e.salary DESC;

```

```

1  SELECT *
2  FROM hr.employees e
3  WHERE (e.job_id, e.salary) IN
4      (
5          SELECT e.job_id, MAX(salary)
6          FROM hr.employees e
7          GROUP BY e.job_id
8      )
9  ORDER BY e.salary DESC;
10

```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000
145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-04	SA_MAN	14000
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-02	AC_MGR	12008
168	Lisa	Ozer	LOZER	011.44.1343.929268	11-MAR-05	SA_REP	11500

II. Bootstrap5

1. Box

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0NsiGAN5c27a+lvFTNQ0FCw4+7lJGS/ExBqTfCv6IPtYVYv" crossorigin="anonymous">
    <style>
        #border{
            position: relative;

```

```

        margin-top: 1rem;
        margin-bottom: 1 rem;
    }
    #border span {
        display: inline-block;
        width: 8rem;
        height: 8rem;
        margin: 0.25rem;
        background-color: rgb(80,80,80);
    }

    #border.border-style [class^='border']{
        border-width : 2px !important
    }


```

</style>

</head>

<body>

<div class="container">

<h2> Box Border</h2>

<div id="border" class="border-style">

</div>

</div>

!-- JavaScript Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-A3rJD856KowSb7dwlZYEIz+0N9AN0DGw38VW32aQo7tIzQV4aUf+0JG4</script>

</body>

</html>

(1) 박스 테두리 설정

- span에 클래스 적용 - 네방향에 모두 적용된다.

```

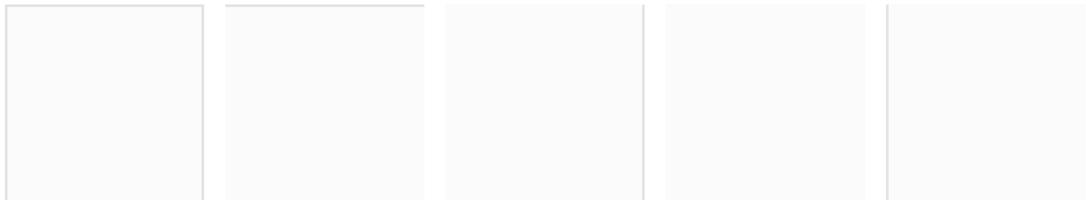
<!-- 하나씩 넣는 방식 -->
<div class="container">
    <h2> Box Border</h2>
    <div id="border" class="border-style">

        <span class="border"></span>
        <span class="border-top"></span>
        <span class="border-end"></span>
        <span class="border-bottom"></span>
        <span class="border-start"></span>

    </div>
</div>

```

Box Border Addictive



```

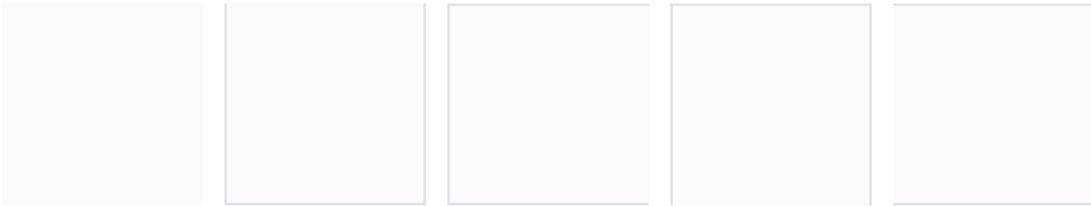
<!-- 모두 적용 후 빼는 방식 -->
<div class="container">
  <h2> Box Border Subreactive</h2>
  <div id = "border" class = "border-style">

    <span class="border border-0"></span>
    <span class="border border-top-0"></span>
    <span class="border border-end-0"></span>
    <span class="border border-bottom-0"></span>
    <span class="border border-start-0"></span>

  </div>
</div>

```

Box Border Subreactive



(2) 박스색 설정

- bootstrap의 색상 코드를 참고한다. (day28)

```

<!-- 색깔 설정 -->
<div class="container">
  <h2>Box Coloring</h2>
  <div id="border" class="border-style">

    <span class="border border-primary"></span>
    <span class="border border-secondary"></span>
    <span class="border border-success"></span>
    <span class="border border-danger"></span>
    <span class="border border-warning"></span>
    <span class="border border-info"></span>
    <span class="border border-light"></span>
    <span class="border border-dark"></span>
    <span class="border border-white"></span>

  </div>
</div>

```

Box Coloring



(3) 박스 선 넓이 설정

```

<!-- 박스 선 넓이 설정 -->
<div class="container">
  <h2> Box Border Width</h2>
  <div id="border" class="border-style">

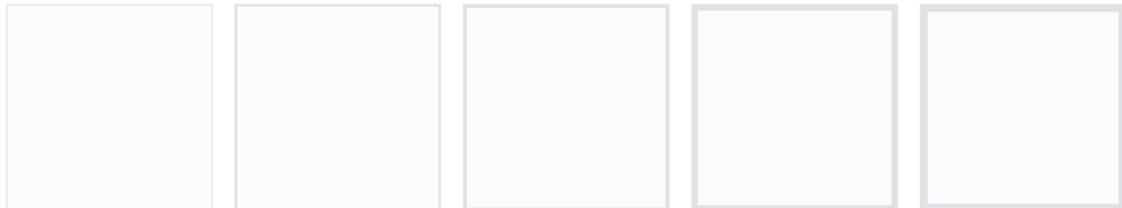
    <span class="border border-1"></span>
    <span class="border border-2"></span>
    <span class="border border-3"></span>
    <span class="border border-4"></span>
    <span class="border border-5"></span>

  </div>
</div>

```

```
</div>  
</div>
```

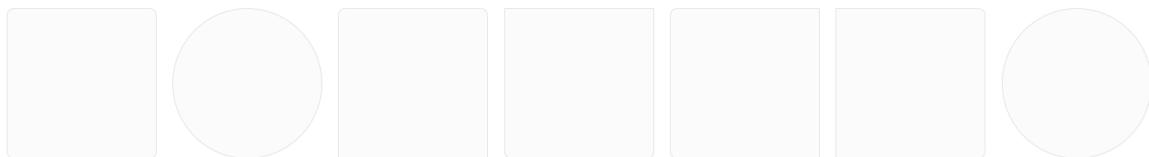
Box Border Width



(4) 박스 둘글게 처리하기.

```
<!-- 박스 둘글게 설정 -->  
<div class="container">  
  <h2> Box Border Radius</h2>  
  <div id="border" class="border-style">  
    <span class="border rounded"></span>  
    <span class="border rounded-circle"></span>  
    <span class="border rounded-top"></span>  
    <span class="border rounded-bottom"></span>  
    <span class="border rounded-start"></span>  
    <span class="border rounded-end"></span>  
    <span class="border rounded-pill"></span>  
  </div>  
</div>
```

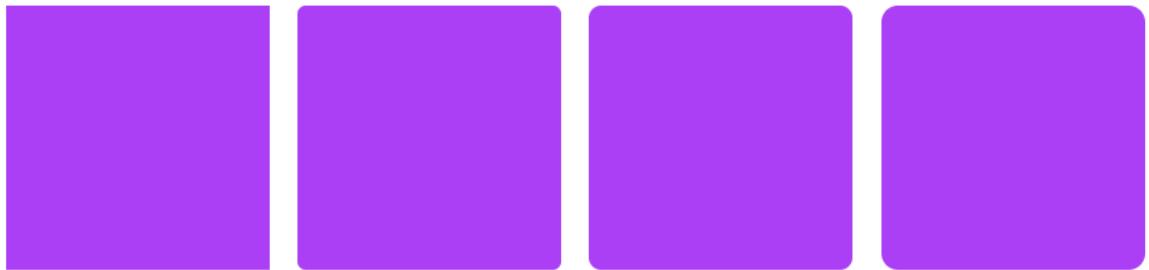
Box Border Radius



(5) 박스 둘글게 처리 - 곡률

```
<!-- 박스 곡률 설정 -->  
<div class="container">  
  <h2> Box Border Radius Size</h2>  
  <div id="border" class="border-style">  
    <span class="rounded-0"></span>  
    <span class="rounded-1"></span>  
    <span class="rounded-2"></span>  
    <span class="rounded-3"></span>  
  </div>  
</div>
```

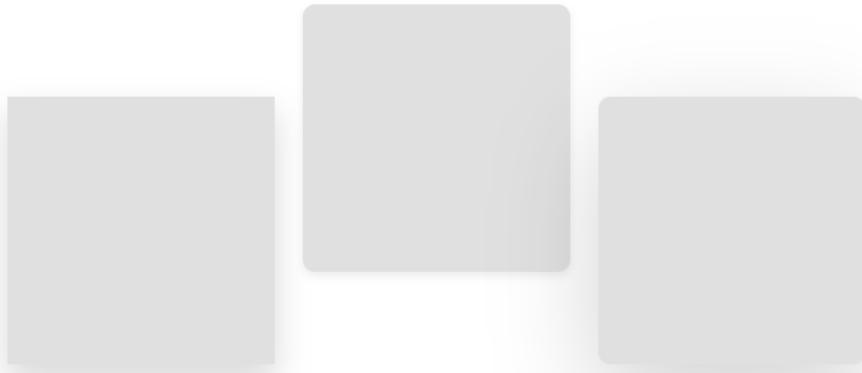
Box Border Radius Size



(6) 박스 그림자

```
<!-- 박스 그림자 -->
<div class="container">
  <h2> Box Shadow</h2>
  <div id="border" class="border-style">
    <span class="shadow"></span>
    <span class="shadow-sm p-3 mb-5 rounded"></span>
    <span class="shadow-lg rounded"></span>
  </div>
</div>
```

Box Shadow



(6) 박스 크기 - width

```
<!-- 박스 크기 w-비율로 글자크기에 맞춰 진행된다. -->
<!-- 그림도 이런식으로 조절이 가능하다. -->
<div class="container">
  <h2>Box Sizing - width </h2>
  <div id="border" class="border-style">
    <span class="w-25"> width 25%</span>
    <span class="w-50"> width 50%</span>
    <span class="w-75"> width 75%</span>
    <span class="w-100">width 100%</span>
    <span class="w-auto">width auto</span>
  </div>
</div>
```

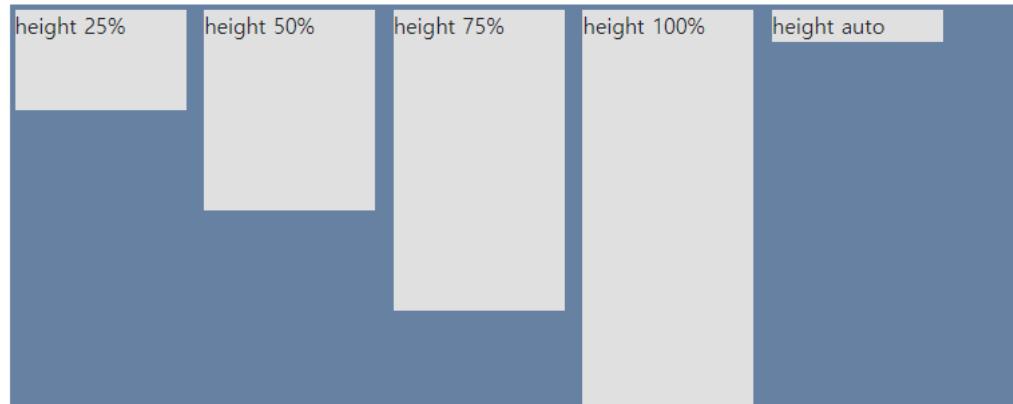
Box sizing



(7) 박스 크기 - height

```
<head>
<style>
#height{
    height:300px;
    background-color: rgb(88, 130, 167);
}
#height span {
    display: inline-block;
    width: 8rem;
    height: 8rem;
    margin: 0.25rem;
    background-color: rgb(224, 224, 224);
}
</style>
</head>
<body>
<div class="container">
    <h2> Box Sizing - height</h2>
    <div id="height" class="border-style">
        <span class="h-25"> height 25%</span>
        <span class="h-50"> height 50%</span>
        <span class="h-75"> height 75%</span>
        <span class="h-100"> height 100%</span>
        <span class="h-auto"> height auto</span>
    </div>
</div>
</body>
```

Box Sizing - height



(8) 이미지 크기

```
<div class="container">
  <h2> Img Sizing</h2>
  <div class="border-style">
    
    
    
    
    

  </div>
</div>
```

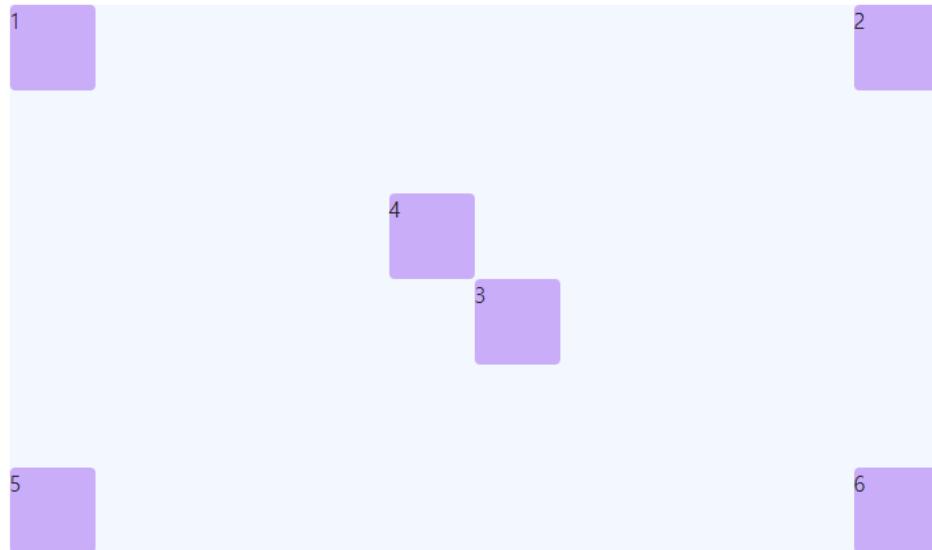


2. Position

(1) 절대위치

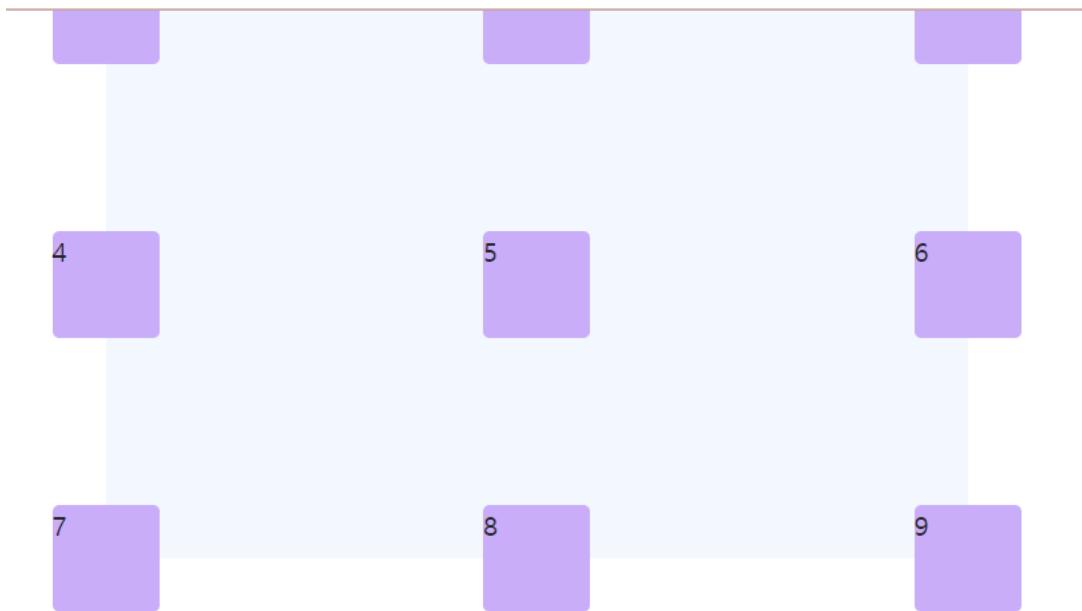
```
<div class="container">
  <h2> 절대 위치! </h2>
  <div id="parent" class="position-relative">
    <div id="child" class="position-absolute top-0 start-0">1</div>
    <div id="child" class="position-absolute top-0 end-0">2</div>
    <div id="child" class="position-absolute top-50 start-50">3</div>
    <div id="child" class="position-absolute bottom-50 end-50">4</div>
    <div id="child" class="position-absolute bottom-0 start-0">5</div>
    <div id="child" class="position-absolute bottom-0 end-0">6</div>
  </div>
</div>
```

절대 위치



(2) 절대위치 - 중간값 기준

```
<div class="container">
  <h2> 절대위치 중간값 기준</h2>
  <div id="parent" class="position-relative">
    <div id="child" class="position-absolute top-0 start-0 translate-middle">1</div>
    <div id="child" class="position-absolute top-0 start-50 translate-middle">2</div>
    <div id="child" class="position-absolute top-0 start-100 translate-middle">3</div>
    <div id="child" class="position-absolute top-50 start-0 translate-middle">4</div>
    <div id="child" class="position-absolute top-50 start-50 translate-middle">5</div>
    <div id="child" class="position-absolute top-50 start-100 translate-middle">6</div>
    <div id="child" class="position-absolute top-100 start-0 translate-middle">7</div>
    <div id="child" class="position-absolute top-100 start-50 translate-middle">8</div>
    <div id="child" class="position-absolute top-100 start-100 translate-middle">9</div>
  </div>
</div>
```



(3) 사진 위에서 글자 배치하기

```

<style>
    .backdrop{
        height: 500px;
        background-image: url('bg.jpg');
        margin-top: 10rem;
        margin-bottom: 10rem;
    }

    .overlay{
        background-color: rgb(0,0,0,0.2);
    }
</style>

<section>

    <div class="backdrop position-relative">
        <div class="overlay position-absolute top-0 start-0 w-100 h-100">
            <div class="overlay-text h-100 d-flex flex-column justify-content-center align-items-center text-center">
                <h1>Lorem ipsum dolor sit amet consectetur adipisicing.</h1>
                <p class="lead">Lorem ipsum, dolor sit amet consectetur adipisicing.</p>
            </div>
        </div>
    </div>

</section>

```



• position-relative

• position-absolute

• position-static

• position-fixed

• position-sticky

• position을 center로 잡는 방법 : .translate-middle (정 가운데) .translate-middle-x, .translate-middle-y