Day29

Date	@09/08/2022
작성일시	@2022년 8월 9일 오전 9:37
강의 번호	BD101
유형	Bootstrap Data_SQL
강사명	AustinYoon
강의자료	☑
노션 복습	☑
코딩 복습	☐
주말숙제(교제)	☐
정리	☑

Day29

Aug

https://github.com/Kray273/SQLLab

BootstrapLab/Day29 at main · Kray273/BootstrapLab
Contribute to Kray273/BootstrapLab development by creating an account on GitHub.
https://github.com/Kray273/BootstrapLab/tree/main/Day29
Kray273/
BootstrapLab

1 Contributor 0 Issues 0 Stars 0 Forks

ORACLE

그룹함수

👉 단일행 함수와는 달리 여러 행에 대해 함수가 적용되어 하나의 결과를 나타내는 함수이다.
예를 들면 학생들의 영어점수들의 합계, 수학 점수들의 평균등 열에 대해 같은 값끼리 묶어서
묶은 행들의 집합에 대해 그룹 연산이 필요할때는 GROUP BY절을 사용하여 처리하고
또 묶은 그룹에 다시 조건이 필요할때는 HAVING절을 이용한다.

```
SELECT 그룹함수(열이름)
FROM 테이블이름
WHRER 조건식
ORDER BY 열이름
```

```
COUNT 개수 : null의 값도 셈
SUM 합계 : 나머지 함수들은 전부 null값은 제외하고 연산을 수행
AVG 평균, MAX 최대값, MIN 최소값, STDDEV 표준편차, VARIANCE 분산.
```

```
--행의 개수 파악
SELECT COUNT(salary)
FROM hr.employees;
--※주의 COUNT함수는 null값도 행으로 계산함.
```

COUNT(SALARY)
107

Download CSV

```
--합계, 평균
SELECT AVG(salary) AS 평균, SUM(salary) AS 합계,
    SUM(salary)/COUNT(salary) AS 합계평균
FROM hr.employees;
--※AVG 함수는 null값을 제외하고 평균을 구하는데 null값을 포함하여
-- 평균을 계산해야한다면, NVL함수를 사용
```

평균	합계	합계평균
6461.831775700934579439252336448598130841	691416	6461.831775700934579439252336448598130841

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

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

Download CSV

GROUP BY

👉 특정 열의 데이터 값들을 그룹으로 묶어 그룹함수를 적용한다.
SELECT절에 열이름과 그룹함수를 함께 기술할떄는 GROUP BY절을 사용하게 된다.

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

```
--그룹화하고 정렬해 출력
SELECT job_id 직무, SUM(salary) "직무별 총급여", AVG(salary) "직무별 평균급여"
```

```
FROM hr.employees
WHERE employee_id >= 10
GROUP BY job_id
ORDER BY job_id DESC;
```

직무	직무별 총급여	직무별 평균급여
ST_MAN	36400	7280
ST_CLERK	55700	2785
SH_CLERK	64300	3215
SA_REP	250500	8350

HAVING

👉 그룹화된 값에 조건식을 적용해야 할 때 사용, 즉 WHERE절에서는 그룹함수를 적용할 수 없기 때문에 HAVING절을 사용하여 그룹함수와 결과값에 대해 조건식을 적용할 수 있다.

```
SELECT 기준열, 그룹함수(열이름)
FROM 테이블 이름
[WHERE 조건식]
GROUP BY 열이름
[HAVING 조건식];
```

```
--그룹에 대한 조건식
SELECT job_id 직무, SUM(salary) "직무별 총급여", AVG(salary) "직무별
FROM hr.employees
WHERE employee_id >= 10
GROUP BY job_id
HAVING SUM(salary) > 30000
ORDER BY SUM(salary) DESC;
```

직무	직무별 총급여	직무별 평균급여
SA_REP	250500	8350
SH_CLERK	64300	3215
SA_MAN	61000	12200
ST_CLERK	55700	2785

JOIN

👉 관계형 데이터베이스의 의미는 테이블들이 관계를 맺고 조작되는 원리에서 유래되었다.
테이블들에는 각각의 성질에 맞는 데이터들이 저장되어 있고 그 데이터들은 특정 규칙에 따라 상화 관계를 맞는다. 데이터는 여러 테이블에 흩여져 저장되어 있어 사용자가 원하는 대로 데이터를 사용하려면 특별한 방법이 필요하다. 이 때 사용되는 기법이 바로 JOIN이다.

JOIN은 한개 이상의 테이블고 테이블을 서로 연결하여 사용하는 기법을 의미한다.

실무에서는 동등, 외부, 자체 JOIN등을 사용하게 된다.
(이 외에도 곱집합, 비동등 조인,....다양한 조인들도 있다.)

```
SELECT 테이블명1.열이름1, 테이블명2.열이름2
FROM 테이블명1, 테이블명2
WHERE 태이블명1.열이름1 = 테이블명2.열이름2;
/* JOIN사용시 규칙
1. SELECT절에는 출력할 열이름을 모두 기술한다.
2. FROM절에는 접급하려는 테이블을 기술한다.
3. WHERE절에는 조인 조건을 기술한다.
4. 사용되는 테이블 이름들에는 별칭을 사용하면 편하다.
*/
```

1. 동등 조인 - 똑같은 데이터끼리 연결

> 👉 동등 조인은 양 테이블에서 조인 조건이 일치하는 행들만 가져오는 가장 일반적으로 자주 사용되는 조인이다.
> = 등호 연산자를 사용하여 조건 값이 정확하게 일치할때만 행을 가져오기 때문에 inner join 이라고도 불린다.

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

MANAGER_ID	DEPARTMENT_ID	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	L
101	10	10	Administration	200	1
100	20	20	Marketing	201	1
201	20	20	Marketing	201	1
100	30	30	Purchasing	114	1
114	30	30	Purchasing	114	1

```
--각 직원이 어느 부서에 속하고 부서의 소재지는 어디인지 조회하시오.
SELECT e.employee_id 사원아이디, d.department_id 부서아이디,
    d.department_name 부서명, l.location_id 지역아이디,
    l.city 도시
FROM hr.employees e, hr.departments d, hr.locations l
WHERE e.department_id = d.department_id
    ANd d.location_id = l.location_id;
```

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

2. 외부 조인 - 모든 데이터를 연결

> 👉 동등조건은 데이터 값이 정확히 일치하는 경우에만 결과를 출력한다. 데이터 값이 일치하지 않으면 결과가 조회되지 않는다.

```
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 출력되었는데, employees의 직원정보는 모두 107건이다. 1건의 차이가 나는 이유는 일치하지 않는 1건이 누락되었기 때문이다.

외부조인(outer join)은 조건을 만족하지 않는 행도 모두 출력하는 조인 기법이다.

- 외부조인은 동등 조인 조건을 만족하지 못해 누락되는 행을 출력하기 위해 사용된다.
- 외부조인은 (+) 기호를 사용한다. 단, + 기호는 조인할 행이 없는 즉 데이터가 부족한 테이블의 열 이름 뒤에 기술한다.
- + 기호는 외부 조인 하려는 한쪽에만 기술할 수 있다.
- + 기호를 붙이면 데이터 값이 부족한 테이블에 null값을 갖는 행이 생성되어 데이터값이 충분한 테이블의 행들이 null행에 조인된다.

```
-- + 기호의 위치! 데이터는 d.department_id가 적다.
SELECT COUNT(*) "조인된 건수"
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id(+);
```

조인된 건수
107

Download CSV

👉 외부조인은 동등 조건과 함께 많이 사용된다.

- 양쪽 테이블 중 전부 출력하고 싶은 테이블 쪽을 앞에 둔다.
- (+)는 다른쪽(뒤쪽) 테이블 조인 조건에 붙인다.

3. 자체 조인

👉 employees테이블의 직원 정보에는 manager_id열이 있는데, 이 열은 그 직원의 담당 매니저의 정보를 담고 있는 열이다.
이 경우에 어떤 직원의 담당 매니저를 알고 싶을때 결국 그 테이블에 다시 조인해야한다.
이렇게 자기 자신의 테이블을 조인하는것을 자체 조인 self join이라고 한다.

```
SELECT 열이름1, 열이름2, 열이름3
FROM 테이블명1 별명 1, 테이블명2 별명2
WHERE 태이블명1.열이름1 = 테이블명1.열이름2;
--한개의 테이블에 별명을 2개로 각각 다른 이름으로 지정한다.
```

```
-- SELF JOIN
SELECT a.employee_id "사원 아이디", a.first_name || a.last_nam
    a.manager_id "매니저 아이디", b.first_name || b.last_name
FROM hr.employees a, hr.employees b
WHERE a.manager_id = b.employee_id;
```

사원 아이디	사원이름	매니저 아이디	매니저이름
101	NeenaKochhar	100	StevenKing
102	LexDe Haan	100	StevenKing
114	DenRaphaely	100	StevenKing

서브쿼리 - SELECT문 안에 SELECT

👉 서브쿼이는 SELECT구문안에서 보조로 사용되는 또 다른 SELECT구문이다.
SELECT구문을 효율적으로 작성할 수 있도록 도와준다.
복잡한 SELECT구문을 작성할 때, 필수적으로 많이 사용되는 기법이다.

```
SELECT 열이름1, 열이름2, 열이름3
FROM 테이블명
WHERE 조건식
      (SELECT 열이름1, 열이름2, 열이름3
      FROM 테이블명
      WHERE 조건식);
/* 서브쿼리는 논리가 복잡한 SQL구문에서는 거의 필수로 많이 사용된다.
  • 서브쿼리는 ()괄호로 묶여서 사용한다. 메인은 묶지 않는다.
  • 메인쿼리와 서브쿼리를 연결하기위해 단일행 연산자 또는 다중행 연상자를 사용한다.
  • 서브쿼리(괄호안)가 먼저 실행되고 그 다음 메인 쿼리가 실행된다.
  • 서브쿼리의 서브쿼리의 서브쿼리 형태로 계속 중첩될 수 있다.
```

👉 서브쿼이의 종류
- 단일행 서브쿼리 : 하나의 행을 검색하는 서브쿼리
- 다중행 서브쿼리 : 하나 이상의 행을 검색하는 서브쿼리
- 단일열 서브쿼리 : 하나 이상읜 열를 검색하는 서브쿼리

1. 단일행 서브쿼리 : 서브쿼리 SELECT문에서 얻은 한개의 행의 결과값을 메인 쿼리로 전달하는 서브쿼리이다.

```
-- 서브쿼리의 조건과 맞는 사람 찾기
SELECT *
FROM hr.employees
WHERE salary = (
                SELECT salary
                FROM hr.employees
                WHERE last_name = 'De Haan'
                );
```

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

```
--동일값으로 인한 중복 에러
SELECT *
FROM hr.employees
WHERE salary = (
                SELECT salary
                FROM hr.employees
                WHERE last_name = 'Taylor'
                );
```

```
ORA-01427: single-row subquery returns more than one row
```

2. 다중행 서브쿼리

👉 사용법은 단일행과 동일하다. 다중행 서브쿼리는 하나 이상의 결과를 메인 쿼리에 전달하는 경우에 사용된다.
IN ⇒ 같은 값이 여러개 ⇒ IN(1,2,3)
NOT IN ⇒ 같은 값이 아닌 여러개 ⇒ NOT IN(1,2,3)
EXIST ⇒ 값이 있으면 True, 없으면 False ⇒ EXIST(1,2,3)
ANY ⇒ 최소한 하나라도 존재(or) ⇒ ANY(10,20)
ALL ⇒ 모두 만족(and) ⇒ ALL(10,20)

```
/* employees 테이블에서 department_id 별로
가장 낮은 salary 가 얼마인지 찾아서
그에 해당하는 직원이 누구인지 다중 행 서브쿼리를 이용하여 찾아보자*/
SELECT *
FROM hr.employees
WHERE salary IN(
                SELECT MIN(salary)
                FROM hr.employees
                GROUP BY department_id
                )
ORDER BY salary DESC;
```

EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	CO
KOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-
LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-
BAER	515.123.8888	07-JUN-02	PR_REP	10000	-
BLOOM	011.44.1343.829268	23-MAR-06	SA_REP	10000	.2
KING	011.44.1345.429268	30-JAN-04	SA_REP	10000	.3

3. 다중열 서브쿼리

👉 메인 쿼리와 서브 쿼리를 비교하는 WHERE조건식에서 비교되는 열이 여러개일 경우 사용되는 서브쿼리

```
SELECT 열이름1, 열이름2, 열이름3...
FROM 테이블명
WHERE (열이름1, 열이름2, 열이름3..) IN(
      SELECT 열이름1, 열이름2, 열이름3
      FROM 테이블명
      WHERE 조건식);
```

```
/*employees 테이블에서 job_id 별로 가장 높은 salary가 얼마인지 찾아보고
찾아낸 job_id별 salary 에 해당하는 직원이 누구인지
다중 열 서브쿼리를 사용하여 찾아보자*/
SELECT job_id, salary, last_name
FROM hr.employees
WHERE (job_id, salary) IN(
                SELECT job_id, MAX(salary)
                FROM hr.employees
                GROUP BY job_id
                )
ORDER BY salary DESC;
```

JOB_ID	SALARY	LAST_NAME
AD_PRES	24000	King
AD_VP	17000	De Haan
AD_VP	17000	Kochhar
SA_MAN	14000	Russell

BOOTSTRAP

BOX

```
<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>
```

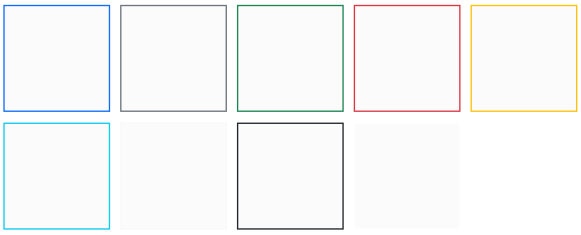
```
<div class="container">
    <h2> Box Border Subtractive</h2>
    <div id="border" class="border-style">
        <span class="border border-0"></span>
        <span class="border border-top-0"></span>
        <span class="border border-end-0"></span>
        <span class="border border-bottom-0"></span>
        <span class="border border-start-0"></span>
    </div>
</div>
```

Box Border Subtractive

Box Border Addictive


```
<div class="container">
<h2> Box Border 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 Border Coloring



```
<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>
```

Box Border Width


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

Box Border Radius


```
<div class="class">
    <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

```
<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 class="shadow-lg rounded"></span>

    </div>
</div>
```

Box Shadow

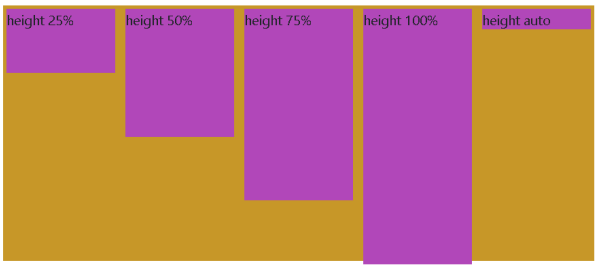
```
<div class="container">
    <h2> Box sizing</h2>
    <div id="border" class="border-style">
        <span class="w-25"> width 25%</span>
        <span class="w-50"> width 50%</span>
        <span class="w-75"> width 75%</span>
        <span class="w-100"> width 100%</span>
        <span class="w-auto"> width Auto</span>
    </div>
</div>
```

Box sizing


```
#height {
            height:300px;
            background-color: rgb(199, 151, 40);

        }
        #height span {
            display: inline-block;
            width:8rem;
            height:8rem;
```

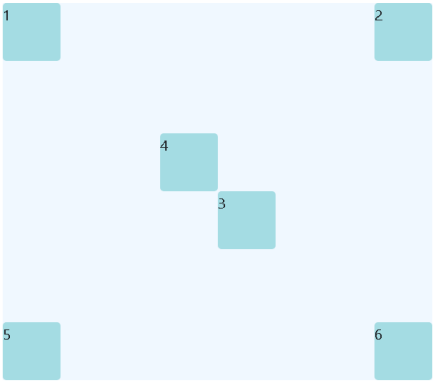
```
            margin:0.25rem;
            background-color: rgb(177, 71, 185);
        }



<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>
```

Box sizing - height

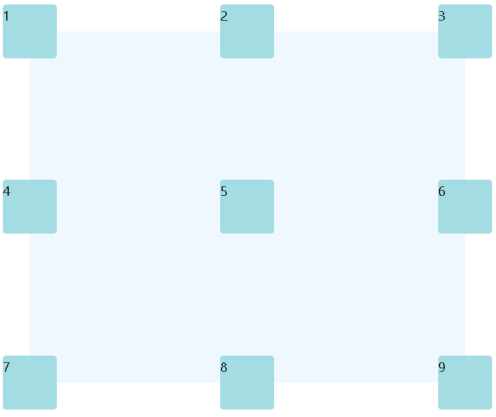



```
<div class="container">
        <h2>절대 위치</h2>
        <div id="parent" class="position-relative">
            <div id="child" class="position-absolute top-0 start-0">1<
            <div id="child" class="position-absolute top-0 end-0">2</d
            <div id="child" class="position-absolute top-50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

절대 위치

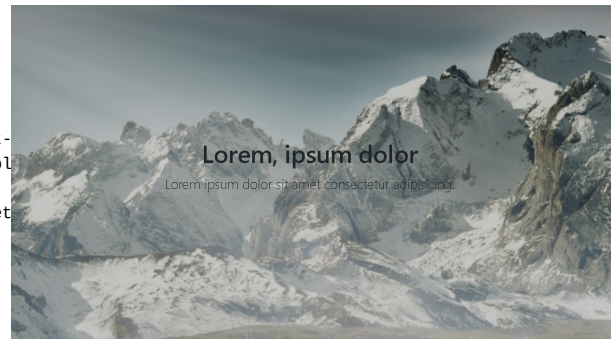



```
<div class="container">
        <div id="parent" class="position-relative">
            <div id="child" class="position-absolute top-0 start-0
            <div id="child" class="position-absolute top-0 start-5
            <div id="child" class="position-absolute top-0 start-1
            <div id="child" class="position-absolute top-50 start-
            <div id="child" class="position-absolute top-50 start-
            <div id="child" class="position-absolute top-50 start-
            <div id="child" class="position-absolute top-100 start
            <div id="child" class="position-absolute top-100 start
            <div id="child" class="position-absolute top-100 start
        </div>
    </div>
```




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

        .overlay{
            background-color: rgba(0,0,0,0.2);
```

```
        }
    </style>

    <section>
        <div class="backdrop position-relative">
            <div class="overlay position-absolute top-0 start-
                <div class="overlay-text h-100 d-flex flex-col
                    <h1>Lorem, ipsum dolor</h1>
                    <p class="lead">Lorem ipsum dolor sit amet
                </div>
            </div>
        </div>
    </section>
```



.position-relative

.position-absolute

.position-static

.position-fixed

.position-sticky

position center

.translate-middle

.translate-middle-x

.translate-middle-y
