[스파르타코딩클럽] 엑셀보다 쉬운 SQL - 4주차

 **매 주차 강의자료 시작에 PDF파일을 올려두었어요!**

▼ PDF 파일

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/721f99c7-4a9e-4f91-89fd-48d8a27b8668/_SQL_4_updated_210621.pdf.pdf

[수업 목표]

1. Subquery(서브쿼리)의 사용 방법을 배워본다
2. 실전에서 유용한 SQL 문법을 더 배워본다
3. SQL을 사용하여 실전과 같은 데이터분석을 진행해본다

[목차]

 모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

01. 오늘 배울 것

▼ 1) Subquery: 원하는 데이터를 더 쉽게 얻어보기

> 👉 Subquery란? 쿼리 안의 쿼리라는 의미입니다.
>
> 하위 쿼리의 결과를 상위 쿼리에서 사용하면, SQL 쿼리가 훨씬 간단해져요!

- 즉, Subquery를 사용하지 않아도 원하는 데이터를 얻어낼 수 있겠지만, 더 편하고 간단하게 원하는 데이터를 얻기 위해 사용되는 파워풀한 기능입니다.
- Subquery에 대한 이해도가 생기면, With구문을 이용해서 더 간단하게 만들어볼게요!

▼ 2) 실전에서 유용한 SQL 문법들

> 😂 생각보다 실무에서의 데이터는 지저분하고 복잡합니다.
>
> 주어진 데이터를 원하는 유의미한 정보로 만들기 위해서는 이것저것 해야할 일이 많습니다. 문자열을 한 번에 정리한다든지, 조건에 따라 데이터를 구분한다든지요. 실무 속 날것의 데이터에서도 원하는 데이터를 뽑아낼 수 있는 유용한 기능을 배워봅니다!

▼ 3) 이제 여러분은...

> 🎉 축하합니다!
>
> 이제 여러분은 DB에 저장된 데이터를 꺼내서 필요한 분석을 해낼 수 있는 사람이 되었습니다. 하지만, 실전에서의 데이터분석을 위해서 아직 배워야 할 것들이 아직 조금 남았습니다.

02. 원하는 데이터를 더 쉽게: Subquery

▼ 4) Subquery 사용방법 익혀보기

> 👉 하나의 SQL 쿼리 안에 또다른 SQL 쿼리가 있는 것을 의미합니다.
> 여러 번 듣는 것보다, 한 번 보는게 이해가 빠르겠죠? 바로 가보시죠!

- kakaopay로 결제한 유저들의 정보 보기

  → 우선, 이렇게 볼 수 있겠죠? `users` 와 `orders` 의 inner join으로!

```
select u.user_id, u.name, u.email from users u
inner join orders o on u.user_id = o.user_id
where o.payment_method = 'kakaopay'
```

- 그런데, 이것을 이렇게 할 수도 있습니다. 조금 더 직관적이지 않나요?

  1. 우선 kakaopay로 결제한 user_id를 모두 구해보기 → `K` 라고 합시다.

```
select user_id from orders
where payment_method = 'kakaopay'
```

  2. 그 후에, user_id가 `K` 에 있는 유저들만 골라보기

  → 이게 바로 서브쿼리!

```
select u.user_id, u.name, u.email from users u
where u.user_id in (
  select user_id from orders
  where payment_method = 'kakaopay'
)
```

03. Subquery 본격 사용해보기

▼ 5) 자주 쓰이는 Subquery 유형 알아보기

> 👉 자주 쓰는 Subquery를 먼저 알아볼까요?
> Subquery는 where, select, from 절에서 유용하게 사용될 수 있어요!

▼ 6) Where 에 들어가는 Subquery

> 👉 Where은 조건문이죠? Subquery의 결과를 조건에 활용하는 방식으로 유용하게 사용합니다.
>
> where 필드명 in (subquery) 이런 방식으로요!

- 예를 들면, 카카오페이로 결제한 주문건 유저들만, 유저 테이블에서 출력해주고 싶을 때는 아래와 같이 표현할 수 있겠죠.

```
select * from users u
where u.user_id in (select o.user_id from orders o
          where o.payment_method = 'kakaopay');
```

- 쿼리가 실행되는 순서를 이렇게 상상하면 편해요!

👉 (1) from 실행: users 데이터를 가져와줌
(2) Subquery 실행: 해당되는 user_id의 명단을 뽑아줌
(3) where .. in 절에서 subquery의 결과에 해당되는 'user_id의 명단' 조건으로 필터링 해줌
(4) 조건에 맞는 결과 출력

▼ 7) Select 에 들어가는 Subquery

👉 Select는 결과를 출력해주는 부분이죠? 기존 테이블에 함께 보고싶은 통계 데이터를 손쉽게 붙이는 것에 사용합니다.

select 필드명, 필드명, (subquery) from .. 이렇게요!

- 앞서 보았던것처럼, '오늘의 다짐' 데이터를 보고 싶은데 '오늘의 다짐' 좋아요의 수가, 본인이 평소에 받았던 좋아요 수에 비해 얼마나 높고 낮은지가 궁금할 수 있겠죠?
- 그럼, 평균을 먼저 구해봅시다! `user_id='4b8a10e6'` 를 예시로!

```
select avg(likes) from checkins c2
where c2.user_id = '4b8a10e6'
```

- 그러면, 이렇게 표현할 수 있어요!

```
select c.checkin_id, c.user_id, c.likes,
  (select avg(likes) from checkins c2
  where c2.user_id = c.user_id) as avg_like_user
from checkins c;
```

- 쿼리가 실행되는 순서를 이렇게 상상하면 편해요!

👉 (1) 밖의 select * from 문에서 데이터를 한줄한줄 출력하는 과정에서
(2) select 안의 subquery가 매 데이터 한줄마다 실행되는데
(3) 그 데이터 한 줄의 user_id를 갖는 데이터의 평균 좋아요 값을 subquery에서 계산해서
(4) 함께 출력해준다!

▼ 8) From 에 들어가는 Subquery (가장 많이 사용되는 유형!)

👉 From은 언제 사용하면 좋을까요? 내가 만든 Select와 이미 있는 테이블을 Join하고 싶을 때 사용하면 딱이겠죠!

- 자, 우선 유저 별 좋아요 평균을 먼저 구해볼까요?

  → checkins 테이블을 user_id로 group by 하면 되겠죠?

```
select user_id, round(avg(likes),1) as avg_like from checkins
group by user_id
```

- 자, 이제 여기서 해당 유저 별 포인트를 보고 싶다면?

  → 그러면, 포인트와 like의 상관정도를 알 수 있겠죠?

```
select pu.user_id, a.avg_like, pu.point from point_users pu
inner join (
  select user_id, round(avg(likes),1) as avg_like from checkins
  group by user_id
) a on pu.user_id = a.user_id
```

- 쿼리가 실행되는 순서를 이렇게 상상하면 편해요!

👉 (1) 먼저 서브쿼리의 select가 실행되고,
(2) 이것을 테이블처럼 여기고 밖의 select가 실행!

04. Subquery 연습해보기 (where, select)

▼ 9) Where 절에 들어가는 Subquery 연습해보기

▼ [연습] 전체 유저의 포인트의 평균보다 큰 유저들의 데이터 추출하기

👉 포인트가 평균보다 많은 사람들의 데이터를 추출해보자!
*참고: 평균 포인트는 5380점

힌트! → `point_users` 테이블을 이용해서 avg를 구하고, 다시 `point_users`와 조인하세요!

	point_user_id	created_at	updated_at	user_id	point
1	5f27dae4ee305500095e51de	2020-08-03 18:37:41	2020-08-12 12:26:26	39e5b35b	5,521
2	5f2940e4e268350009a6a449	2020-08-04 20:05:08	2020-08-12 20:59:51	4f9ca5b4	5,564
3	5f2bb9c90df1de0009154270	2020-08-06 17:05:29	2020-08-12 21:43:50	8882e93c	5,567
4	5f29bdbae268350009a6a761	2020-08-05 04:57:46	2020-08-12 03:39:38	a2c78b69	5,583
5	5f28b967d80bcb000982b5b6	2020-08-04 10:27:04	2020-08-08 16:13:51	b2b9db3e	5,654
6	5f28d3b5ee305500095e576b	2020-08-04 12:19:18	2020-08-11 00:19:14	bb6f03f3	5,718
7	5f27b629371f2b0009bbb55f	2020-08-03 16:00:58	2020-08-12 18:51:01	2638550d	5,746
8	5f2bd0840df1de000915431c	2020-08-06 18:42:28	2020-08-12 17:17:18	5aafaccc	5,756
9	5f28039fd80bcb000982b2e5	2020-08-03 21:31:28	2020-08-11 00:15:03	41041f94	5,770
10	5f29e1c8e268350009a6a782	2020-08-05 07:31:37	2020-08-11 10:22:45	c37f27e2	5,822
11	5f281bb3d80bcb000982b431	2020-08-03 23:14:12	2020-08-12 01:13:25	3bc9f624	5,837
12	5f27a7783b48020009f5ad5d	2020-08-03 14:58:16	2020-08-10 14:29:24	3ff0ac1f	5,870
13	5f2773e88ff89200093fc428	2020-08-03 11:18:16	2020-08-12 17:27:22	03667091	5,895
14	5f313a37171df600097d9422	2020-08-10 21:14:48	2020-08-12 00:32:30	7f6a1bfb	5,904

Save Cancel Script 200 119 Rows: 1

▼ **[코드스니펫] 포인트 평균 조건으로 Subquery 연습해보기**

```
select * from point_users pu
where pu.point > (select avg(pu2.point) from point_users pu2);
```

👉 **[오늘의 팁!]** 위와 같이, 같은 테이블을 Subquery로 사용할 수도 있어요.

▼ [연습] 이씨 성을 가진 유저의 포인트의 평균보다 큰 유저들의 데이터 추출하기

👉 이씨 성을 가진 유저들의 평균 포인트보다 더 많은 포인트를 가지고 있는 데이터를 추출해보자!
*참고: 이씨 성을 가진 유저들의 평균 포인트는 7454점

힌트! → 위 구문의 서브쿼리 내에서 `users` 와 `inner join` 을 해보세요!

	point_user_id	created_at	updated_at	user_id	point
1	5f2945b77d52a200093197d1	2020-08-04 20:25:43	2020-08-11 15:57:24	95621907	7,459
2	5f27d06bfc7e0e000993409e	2020-08-03 17:53:00	2020-08-10 20:28:50	4b0b333c	7,562
3	5f2a9d8fc00d800009633877	2020-08-05 20:52:47	2020-08-11 22:50:16	12316c05	7,642
4	5f277244d3943c0009e066db	2020-08-03 11:11:17	2020-08-10 09:32:02	1d610865	7,761
5	5f27bf0a371f2b0009bbb5f8	2020-08-03 16:38:51	2020-08-11 22:32:26	09d75364	7,763
6	5f27888bc6a85d000932082c	2020-08-03 12:46:19	2020-08-11 22:13:53	ebef6f9b	7,817
7	5f27dfe8d80bcb000982b0a8	2020-08-03 18:59:05	2020-08-11 23:54:09	640672b6	7,982
8	5f292d78ab63ee0009b2d0b1	2020-08-04 18:42:16	2020-08-11 21:40:44	0bfe0771	7,992
9	5f27dbcdd80bcb000982b07d	2020-08-03 18:41:33	2020-08-12 18:28:53	d848f82d	8,153
10	5f27f387ee305500095e52da	2020-08-03 20:22:48	2020-08-06 02:34:43	bd424673	8,187
11	5f333ad4390108000936c1de	2020-08-12 09:41:57	2020-08-12 21:27:55	8367f602	8,439
12	5f1fce3ce28fa10009b46244	2020-07-28 16:05:33	2020-08-02 18:56:33	17ebdeb2	8,477
13	5f26d67ad3943c0009e065b2	2020-08-03 00:06:35	2020-08-10 23:40:24	ccc2bdb7	8,537
14	5f27dbd1ee305500095e51e5	2020-08-03 18:41:37	2020-08-12 05:02:09	d9efc5c8	8,732

Save Cancel Script 200 60 Rows: 1

▼ **[코드스니펫] 성씨 조건을 걸어서 Subquery 연습해보기**

```
select * from point_users pu
where pu.point >
  (select avg(pu2.point) from point_users pu2
  inner join users u
  on pu2.user_id = u.user_id
  where u.name = "0|**");
```

👉 **[오늘의 팁!]** 필요한 경우, Subquery 안에서 여러 테이블을 Join 할수도 있어요

▼ 10) Select 절에 들어가는 Subquery 연습해보기

▼ [연습] checkins 테이블에 course_id별 평균 likes수 필드 우측에 붙여보기

> 👉 감이 아직 안 오나요? 저랑 같이 해봐요!

힌트! → 이건 힌트 필요 없음! 🤓

	checkin_id	course_id	user_id	likes	course_avg
1	5f2b821f1c03ab000904e862	5f0ae408765dae0006002817	4b8a10e6	4	2.5
2	5f2bca728cf86b0009520d7f	5f0ae408765dae0006002817	fac91699	3	2.5
3	5f2bcafd8cf86b0009520dc7	5f0ae408765dae0006002817	c9ea3998	3	2.5
4	5f2bcc0d8cf86b0009520dd1	5f0ae408765dae0006002817	c6752f58	5	2.5
5	5f2bce748cf86b0009520e1d	5f0ae408765dae0006002817	4b8a10e6	3	2.5
6	5f2bd2368cf86b0009520e24	5f0ae408765dae0006002816	c9ea3998	4	1.6
7	5f2beb3cb63cef00093e40d3	5f0ae408765dae0006002816	95621907	3	1.6
8	5f2beb48b63cef00093e40d4	5f0ae408765dae0006002817	deb54ce4	4	2.5
9	5f2bf08153c92c00099be3ff	5f0ae408765dae0006002817	12dd94fe	4	2.5
10	5f2bf21cb63cef00093e410f	5f0ae408765dae0006002816	76f656d4	3	1.6
11	5f2bf22953c92c00099be412	5f0ae408765dae0006002817	deb54ce4	4	2.5
12	5f2bf8cab63cef00093e415f	5f0ae408765dae0006002816	f0fc6b71	3	1.6
13	5f2bfb8b53c92c00099be470	5f0ae408765dae0006002816	326fa9eb	3	1.6
14	5f2bfd2bb63cef00093e4184	5f0ae408765dae0006002817	dda1b48e	2	2.5

Save Cancel Script 200 134 Rows: 1

▼ **[코드스니펫] course_id별 평균 like수 붙여보기**

```
select checkin_id, course_id, user_id, likes,
(select avg(c2.likes) from checkins c2
where c.course_id = c2.course_id)
from checkins c;
```

▼ [연습] checkins 테이블에 과목명별 평균 likes수 필드 우측에 붙여보기

> 👉 직전에 실습했던 것에, courses 테이블을 join 하면 되겠죠?

	checkin_id	title	user_id	likes	course_avg
1	5f2bd2368cf86b0009520e24	웹개발 종합반	c9ea3998	4	1.6
2	5f2beb3cb63cef00093e40d3	웹개발 종합반	95621907	3	1.6
3	5f2bf21cb63cef00093e410f	웹개발 종합반	76f656d4	3	1.6
4	5f2bf8cab63cef00093e415f	웹개발 종합반	f0fc6b71	3	1.6
5	5f2bfb8b53c92c00099be470	웹개발 종합반	326fa9eb	3	1.6
6	5f2c05d5b63cef00093e41df	웹개발 종합반	ef860677	3	1.6
7	5f2c174453c92c00099be502	웹개발 종합반	50158300	3	1.6
8	5f2c21bcb63cef00093e42ad	웹개발 종합반	c1493778	3	1.6
9	5f2c46a5b63cef00093e4303	웹개발 종합반	9716841a	3	1.6
10	5f2ccdc9b63cef00093e4460	웹개발 종합반	5df4291d	2	1.6
11	5f2cd0c653c92c00099be652	웹개발 종합반	09d75364	1	1.6
12	5f2d02a22a5c850009b78615	웹개발 종합반	96556738	1	1.6
13	5f2d2393a9ff0a0009b4abac	웹개발 종합반	326fa9eb	0	1.6
14	5f2d37492417790009 0aeef0	웹개발 종합반	1e13df35	0	1.6

Save Cancel Script 200 134

▼ **[코드스니펫] 과목명별 평균 like수 붙여보기**

```
select checkin_id, c3.title, user_id, likes,
(select round(avg(c2.likes),1) from checkins c2
where c.course_id = c2.course_id) as course_avg
from checkins c
inner join courses c3
on c.course_id = c3.course_id;
```

05. Subquery 연습해보기 (from, inner join)

▼ 11) From 절에 들어가는 Subquery 연습해보기

▼ [준비1] course_id별 유저의 체크인 개수를 구해보기!

👉 checkins 테이블을 course_id로 group by 하면 되겠죠! 너무 쉽다!
그리고 distinct로 유저를 세면 되겠네요!

	course_id	cnt_checkins
1	5f0ae408765dae0006002816	40
2	5f0ae408765dae0006002817	47

▼ **[코드스니펫] course_id별 체크인 개수**

```
select course_id, count(distinct(user_id)) as cnt_checkins from checkins
group by course_id
```

▼ [준비2] course_id별 인원을 구해보기!

👉 orders 테이블을 course_id로 group by 하면 되겠죠! 너무 쉽다!

	course_id	cnt_total
1	5f0ae408765dae0006002817	153
2	5f0ae408765dae0006002816	133

▼ [코드스니펫] course_id별 인원

```
select course_id, count(*) as cnt_total from orders
group by course_id
```

▼ [진짜 하고 싶은 것] course_id별 like 개수에 전체 인원을 붙이기

👉 쉽죠? 준비1과 준비2를 inner join 하면 됩니다!

	course_id	cnt_checkins	cnt_total
1	5f0ae408765dae0006002817	47	153
2	5f0ae408765dae0006002816	40	133

▼ [코드스니펫] course_id별 like 개수, 전체

```
select a.course_id, b.cnt_checkins, a.cnt_total from
(
  select course_id, count(*) as cnt_total from orders
  group by course_id
) a
inner join (
  select course_id, count(distinct(user_id)) as cnt_checkins from checkins
  group by course_id
) b
on a.course_id = b.course_id
```

▼ [한 걸음 더] 퍼센트를 나타내기

👉 전체 중 얼마나 like를 하는지 알아보면 좋겠죠?

	course_id	cnt_checkins	cnt_total	ratio
1	5f0ae408765dae0006002817	47	153	0.3072
2	5f0ae408765dae0006002816	40	133	0.3008

▼ **[코드스니펫] course_id별 like 개수, 전체, 비율**

```
select a.course_id, b.cnt_checkins, a.cnt_total, (b.cnt_checkins/a.cnt_total) as ratio from
(
  select course_id, count(*) as cnt_total from orders
  group by course_id
) a
inner join (
  select course_id, count(distinct(user_id)) as cnt_checkins from checkins
  group by course_id
) b
on a.course_id = b.course_id
```

▼ [반 걸음 더] 앗, 강의 제목도 나타나면 좋겠네요!

👉 courses 테이블과 조인을 하면 되겠군요! 그러면 완벽하겠네요!

	title	cnt_checkins	cnt_total	ratio
1	앱개발 종합반	47	153	0.3072
2	웹개발 종합반	40	133	0.3008

▼ **[코드스니펫] 코스제목별 like 개수, 전체, 비율**

```
select c.title,
       a.cnt_checkins,
       b.cnt_total,
       (a.cnt_checkins/b.cnt_total) as ratio
from
(
  select course_id, count(distinct(user_id)) as cnt_checkins from checkins
  group by course_id
) a
inner join
(
  select course_id, count(*) as cnt_total from orders
  group by course_id
) b on a.course_id = b.course_id
inner join courses c on a.course_id = c.course_id
```

06. with절 연습하기

▼ 12) with 절로 더 깔끔하게 쿼리문을 정리하기

▼ **[코드스니펫] 코스제목별 like 개수, 전체, 비율**

```
select c.title,
       a.cnt_checkins,
       b.cnt_total,
       (a.cnt_checkins/b.cnt_total) as ratio
from
```

```
(
  select course_id, count(distinct(user_id)) as cnt_checkins from checkins
  group by course_id
) a
inner join
(
  select course_id, count(*) as cnt_total from orders
  group by course_id
) b on a.course_id = b.course_id
inner join courses c on a.course_id = c.course_id
```

- 다시 위의 코드를 볼까요?

```
select c.title,
       a.cnt_checkins,
       b.cnt_total,
       (a.cnt_checkins/b.cnt_total) as ratio
from
(
  select course_id, count(distinct(user_id)) as cnt_checkins from checkins
  group by course_id
) a
inner join
(
  select course_id, count(*) as cnt_total from orders
  group by course_id
) b on a.course_id = b.course_id
inner join courses c on a.course_id = c.course_id
```

- 이렇게 계속 서브쿼리가 붙으면, inner join 안쪽이 너무 헷갈리겠죠!

  → 그 때 쓰는 것이 with 절! 결과는 같은데 훨씬 보기가 좋죠?

```
with table1 as (
  select course_id, count(distinct(user_id)) as cnt_checkins from checkins
  group by course_id
), table2 as (
  select course_id, count(*) as cnt_total from orders
  group by course_id
)
select c.title,
       a.cnt_checkins,
       b.cnt_total,
       (a.cnt_checkins/b.cnt_total) as ratio
from table1 a inner join table2 b on a.course_id = b.course_id
inner join courses c on a.course_id = c.course_id
```

07. 실전에서 유용한 SQL 문법 (문자열)

▼ 13) 문자열 데이터 다뤄보기

> 👉 실제 업무에서는, 문자열 데이터를 원하는 형태로 한번 정리해야 하는 경우가 많습니다.

▼ 문자열 쪼개보기

👉 이메일 주소에서 @앞의 아이디만 가져오거나, @뒤의 이메일 도메인을 가져오고 싶어요!

- SUBSTRING_INDEX 라는 문법을 사용하면 됩니다. 바로 쿼리를 볼까요?

▼ [코드스니펫] 이메일에서 아이디만 가져와보기

```
select user_id, email, SUBSTRING_INDEX(email, '@', 1) from users
```

👉 @를 기준으로 텍스트를 쪼개고, 그 중 첫 번째 조각을 가져오라는 뜻!

▼ [코드스니펫] 이메일에서 이메일 도메인만 가져와보기

```
select user_id, email, SUBSTRING_INDEX(email, '@', -1) from users
```

👉 @를 기준으로 텍스트를 쪼개고, 그 중 마지막 조각을 가져오라는 뜻!

▼ 문자열 일부만 출력하기

👉 orders 테이블에서 created_at을 날짜까지만 출력하게 해봅시다!

- SUBSTRING 이라는 문법을 사용하면 됩니다. 바로 쿼리를 보러 가시죠!

▼ [코드스니펫] orders 테이블에서 날짜까지 출력하게 해보기

```
select order_no, created_at, substring(created_at,1,10) as date from orders
```

- SUBSTRING(문자열, 출력을 하고싶은 첫 글자의 위치, 몇개의 글자를 출력하고 싶은지)

▼ [코드스니펫] 일별로 몇 개씩 주문이 일어났는지 살펴보기

```
select substring(created_at,1,10) as date, count(*) as cnt_date from orders
group by date
```

08. 실전에서 유용한 SQL 문법 (Case)

▼ 14) CASE: 경우에 따라 원하는 값을 새 필드에 출력해보기

> 👉 특정 조건에 따라, 데이터를 구분해서 정리해주고 싶을 때가 있겠죠?
> 이런 경우에 CASE 라는 문법이 사용됩니다.

- 자, 바로 예시로 배워볼까요?

> 👉 10000점보다 높은 포인트를 가지고 있으면 '잘 하고 있어요!', 평균보다 낮으면 '조금 더 달려주세요!' 라고 표시해 주려면 어떻게 해야할까요?

▼ [코드스니펫] 포인트 보유액에 따라 다르게 표시해주기

```
select pu.point_user_id, pu.point,
case
when pu.point > 10000 then '잘 하고 있어요!'
else '조금 더 달려주세요!'
END as '구분'
from point_users pu;
```

▼ 15) CASE: 실전을 위한 트릭!

> 👉 Subquery를 이용하면 이런 통계도 낼 수 있어요!

1. 우선 몇 가지로 구분을 나누고,

	point_user_id	point	level
1	5f1fce3ce28fa10009b46244	8,477	5천 이상
2	5f20e3549441d800091eab5e	14,759	1만 이상
3	5f211faa9441d800091ead4f	380	5천 미만
4	5f21264a9441d800091eae34	44,113	1만 이상
5	5f222d25bde57f00096c1b4c	23,962	1만 이상
6	5f225cdca4a6870009d8c96e	20,330	1만 이상
7	5f23d7eb385fc70009fdff75	22,476	1만 이상
8	5f268d4fe462a000096e751d	10,688	1만 이상
9	5f26a7d1b81d800009afde82	5,983	5천 이상
10	5f26c7818ff89200093fc2e4	150	5천 미만
11	5f26d615d3943c0009e065af	2,241	5천 미만
12	5f26d6268ff89200093fc345	12,694	1만 이상
13	5f26d67ad3943c0009e065b2	8,537	5천 이상
14	5f26d6d5d3943c0009e065b4	7,353	5천 이상

Save Cancel Script

```
select pu.point_user_id, pu.point,
case
```

```
when pu.point >= 10000 then '1만 이상'
when pu.point >= 5000 then '5천 이상'
else '5천 미만'
END as level
from point_users pu
```

2. 서브쿼리를 이용해서 group by로 통계를 낼 수 있습니다.

	level	cnt
1	5천 미만	146
2	5천 이상	94
3	1만 이상	31

```
select level, count(*) as cnt from (
  select pu.point_user_id, pu.point,
  case
  when pu.point >= 10000 then '1만 이상'
  when pu.point >= 5000 then '5천 이상'
  else '5천 미만'
  END as level
  from point_users pu
) a
group by level
```

3. with 절과 함께하면 금상첨화죠!

```
with table1 as (
  select pu.point_user_id, pu.point,
  case
  when pu.point >= 10000 then '1만 이상'
  when pu.point >= 5000 then '5천 이상'
  else '5천 미만'
  END as level
  from point_users pu
)
select level, count(*) as cnt from table1
group by level
```

09. SQL 문법 복습. 또 복습! (초급)

▼ 16) [퀴즈] 평균 이상 포인트를 가지고 있으면 '잘 하고 있어요' / 낮으면 '열심히 합시다!' 표시하기!

👉 **[힌트!]** CASE 문법 사용, CASE 안에서 Subquery로 평균을 계산하여 비교!

	point_user_id	point	msg
1	5f1fce3ce28fa10009b46244	8,477	잘 하고 있어요!
2	5f20e3549441d800091eab5e	14,759	잘 하고 있어요!
3	5f211faa9441d800091ead4f	380	열심히 합시다!
4	5f21264a9441d800091eae34	44,113	잘 하고 있어요!
5	5f222d25bde57f00096c1b4c	23,962	잘 하고 있어요!
6	5f225cdca4a6870009d8c96e	20,330	잘 하고 있어요!
7	5f23d7eb385fc70009fdff75	22,476	잘 하고 있어요!
8	5f268d4fe462a000096e751d	10,688	잘 하고 있어요!
9	5f26a7d1b81d800009afde82	5,983	잘 하고 있어요!
10	5f26c7818ff89200093fc2e4	150	열심히 합시다!
11	5f26d615d3943c0009e065af	2,241	열심히 합시다!
12	5f26d6268ff89200093fc345	12,694	잘 하고 있어요!
13	5f26d67ad3943c0009e065b2	8,537	잘 하고 있어요!
14	5f26d6d5d3943c0009e065b4	7,353	잘 하고 있어요!

▼ 정답 같이 풀어보기

```
select pu.point_user_id, pu.point,
case
when pu.point > (select avg(pu2.point) from point_users pu2) then '잘 하고 있어요!'
else '열심히 합시다!'
end as 'msg'
from point_users pu
```

▼ 17) [퀴즈] 이메일 도메인별 유저의 수 세어보기

👉 **[힌트!]** SUBSTRING_INDEX와 Group by를 잘 사용하면 끝!

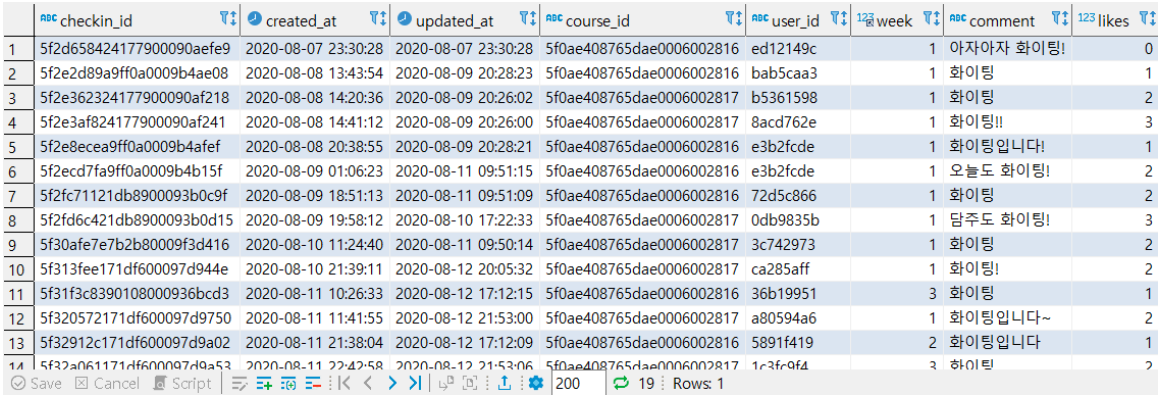
	domain	cnt_domain
1	hanmail.com	46
2	naver.com	164
3	teamsparta.co	47
4	gmail.com	139
5	yahoo.com	46
6	daum.net	56

▼ 정답 살펴보기!

```
select domain, count(*) as cnt from (
  select SUBSTRING_INDEX(email,'@',-1) as domain from users
) a
group by domain
```

▼ 18) [퀴즈] '화이팅'이 포함된 오늘의 다짐만 출력해보기

👉 이번 건 쉽죠? 잊어버리셨을까봐! `like`를 어떻게 썼더라~



	checkin_id	created_at	updated_at	course_id	user_id	week	comment	likes
1	5f2d658424177900090aefe9	2020-08-07 23:30:28	2020-08-07 23:30:28	5f0ae408765dae0006002816	ed12149c	1	아자아자 화이팅!	0
2	5f2e2d89a9ff0a0009b4ae08	2020-08-08 13:43:54	2020-08-09 20:28:23	5f0ae408765dae0006002816	bab5caa3	1	화이팅	1
3	5f2e362324177900090af218	2020-08-08 14:20:36	2020-08-09 20:26:02	5f0ae408765dae0006002817	b5361598	1	화이팅	2
4	5f2e3af824177900090af241	2020-08-08 14:41:12	2020-08-09 20:26:00	5f0ae408765dae0006002817	8acd762e	1	화이팅!!	3
5	5f2e8ecea9ff0a0009b4afef	2020-08-08 20:38:55	2020-08-09 20:28:21	5f0ae408765dae0006002816	e3b2fcde	1	화이팅입니다!	1
6	5f2ecd7fa9ff0a0009b4b15f	2020-08-09 01:06:23	2020-08-11 09:51:15	5f0ae408765dae0006002816	e3b2fcde	1	오늘도 화이팅!	2
7	5f2fc71121db8900093b0c9f	2020-08-09 18:51:13	2020-08-11 09:51:09	5f0ae408765dae0006002816	72d5c866	1	화이팅	2
8	5f2fd6c421db8900093b0d15	2020-08-09 19:58:12	2020-08-10 17:22:33	5f0ae408765dae0006002817	0db9835b	1	담주도 화이팅!	3
9	5f30afe7e7b2b80009f3d416	2020-08-10 11:24:40	2020-08-11 09:50:14	5f0ae408765dae0006002817	3c742973	1	화이팅	2
10	5f313fee171df600097d944e	2020-08-10 21:39:11	2020-08-12 20:05:32	5f0ae408765dae0006002817	ca285aff	1	화이팅!	2
11	5f31f3c8390108000936bcd3	2020-08-11 10:26:33	2020-08-12 17:12:15	5f0ae408765dae0006002816	36b19951	3	화이팅	1
12	5f320572171df600097d9750	2020-08-11 11:41:55	2020-08-12 21:53:00	5f0ae408765dae0006002817	a80594a6	1	화이팅입니다~	2
13	5f32912c171df600097d9a02	2020-08-11 21:38:04	2020-08-12 17:12:09	5f0ae408765dae0006002816	5891f419	2	화이팅입니다	1
14	5f32a061171df600097d9a53	2020-08-11 22:42:58	2020-08-12 21:53:06	5f0ae408765dae0006002817	1c3fc9f4	3	화이팅	2

Save Cancel Script 200 19 Rows: 1

▼ 정답 쿼리 살펴보기!

```
select * from checkins c
where c.comment like '%화이팅%'
```

10. SQL 문법 복습. 또 복습! (중급)

▼ 19) [퀴즈] 수강등록정보(enrolled_id)별 전체 강의 수와 들은 강의의 수 출력해보기

👉 [힌트!] subquery 두 개를 만들어놓고, inner join!

살펴볼 테이블: `enrolled_details`

done_cnt는 들은 강의의 수(done=1),
total_cnt는 전체 강의의 수

	enrolled_id	done_cnt	total_cnt
1	5f0c41073ebf8500096844f0	9	65
2	5f0c622f3606e60009a128d9	1	71
3	5f0c64503606e60009a1293c	6	65
4	5f0d1e3d3606e60009a129b5	9	65
5	5f0d2dab3606e60009a12a40	5	65
6	5f0d33063606e60009a12aba	29	71
7	5f0d33b83606e60009a12b1a	29	71
8	5f0d342e3606e60009a12b7a	13	65
9	5f0d350f3606e60009a12be7	17	65
10	5f0d35173606e60009a12c4f	10	65
11	5f0d352e3606e60009a12cb7	17	71
12	5f0d4b4e3606e60009a12d2b	11	65
13	5f0d55a43606e60009a12d9d	9	65

Save Cancel Script

▼ 정답 쿼리 살펴보기!

```
with lecture_done as (
  select enrolled_id, count(*) as cnt_done from enrolleds_detail ed
  where done = 1
  group by enrolled_id
), lecture_total as (
  select enrolled_id, count(*) as cnt_total from enrolleds_detail ed
  group by enrolled_id
)


select a.enrolled_id, a.cnt_done, b.cnt_total from lecture_done a
inner join lecture_total b on a.enrolled_id = b.enrolled_id
```

😎 이해가 안 되신다고요? 저랑 같이 해봐요!

▼ 20) [퀴즈] 수강등록정보(enrolled_id)별 전체 강의 수와 들은 강의의 수, 그리고 진도율 출력해보기

👉 [힌트!] 진도율 = (들은 강의의 수 / 전체 강의 수)

	enrolled_id	done_cnt	total_cnt	ratio
1	5f0c41073ebf8500096844f0	9	65	0.14
2	5f0c622f3606e60009a128d9	1	71	0.01
3	5f0c64503606e60009a1293c	6	65	0.09
4	5f0d1e3d3606e60009a129b5	9	65	0.14
5	5f0d2dab3606e60009a12a40	5	65	0.08
6	5f0d33063606e60009a12aba	29	71	0.41
7	5f0d33b83606e60009a12b1a	29	71	0.41
8	5f0d342e3606e60009a12b7a	13	65	0.2
9	5f0d350f3606e60009a12be7	17	65	0.26
10	5f0d35173606e60009a12c4f	10	65	0.15
11	5f0d352e3606e60009a12cb7	17	71	0.24
12	5f0d4b4e3606e60009a12d2b	11	65	0.17
13	5f0d55a43606e60009a12d9d	9	65	0.14

Save Cancel Script 200

▼ 정답 쿼리 살펴보기!

```
with table1 as (
  select enrolled_id, count(*) as done_cnt from enrolleds_detail
  where done = 1
  group by enrolled_id
), table2 as (
  select enrolled_id, count(*) as total_cnt from enrolleds_detail
  group by enrolled_id
)

select a.enrolled_id,
       a.done_cnt,
       b.total_cnt,
       round(a.done_cnt/b.total_cnt,2) as ratio
  from table1 a
 inner join table2 b on a.enrolled_id = b.enrolled_id
```

😎 이해가 안 되신다고요? 저랑 같이 해봐요!

▼ 21) [함께] 그러나, 더 간단하게 만들 수 있지 않을까!

👉 아래와 같이 써도 같은 결과겠죠? 이렇게, 가끔 멀리서 보면 더 나은 쿼리를 만들 수 있어요!

```
select enrolled_id,
       sum(done) as cnt_done,
```

```
       count(*) as cnt_total
from enrolleds_detail ed
group by enrolled_id
```

11. 끝 & 숙제 설명

4주 동안 고생 많았습니다! 숙제로 유종의 미를 같이 거둬볼까요?
마지막 숙제는, 문제를 푸는 것이 아니라, `문법을 정리` 해보는 것이에요!

SQL 쿼리문은 문법 종류가 다양하지 않아서,
한번 정리해두면 정말 요긴하게 써먹을 수 있습니다!

블로그를 하나 만들고 문법을 정리한 다음, 제출해주세요!
(이미 있는 분들은 쓰던 것을 사용하셔도 무방합니다)

블로그 만들 수 있는 곳들

- 네이버블로그 만들기: (링크)
- Velog: (링크)
- .. 아무데나!