programmers

실리콘밸리에서 날아온 데이터베이스

3. SQL 기본다지기 (SELECT와 GROUP BY)

keeyonghan@hotmail.com

한기용

Harmonize, Inc

programmers

Contents

programmers

실습환경 소개

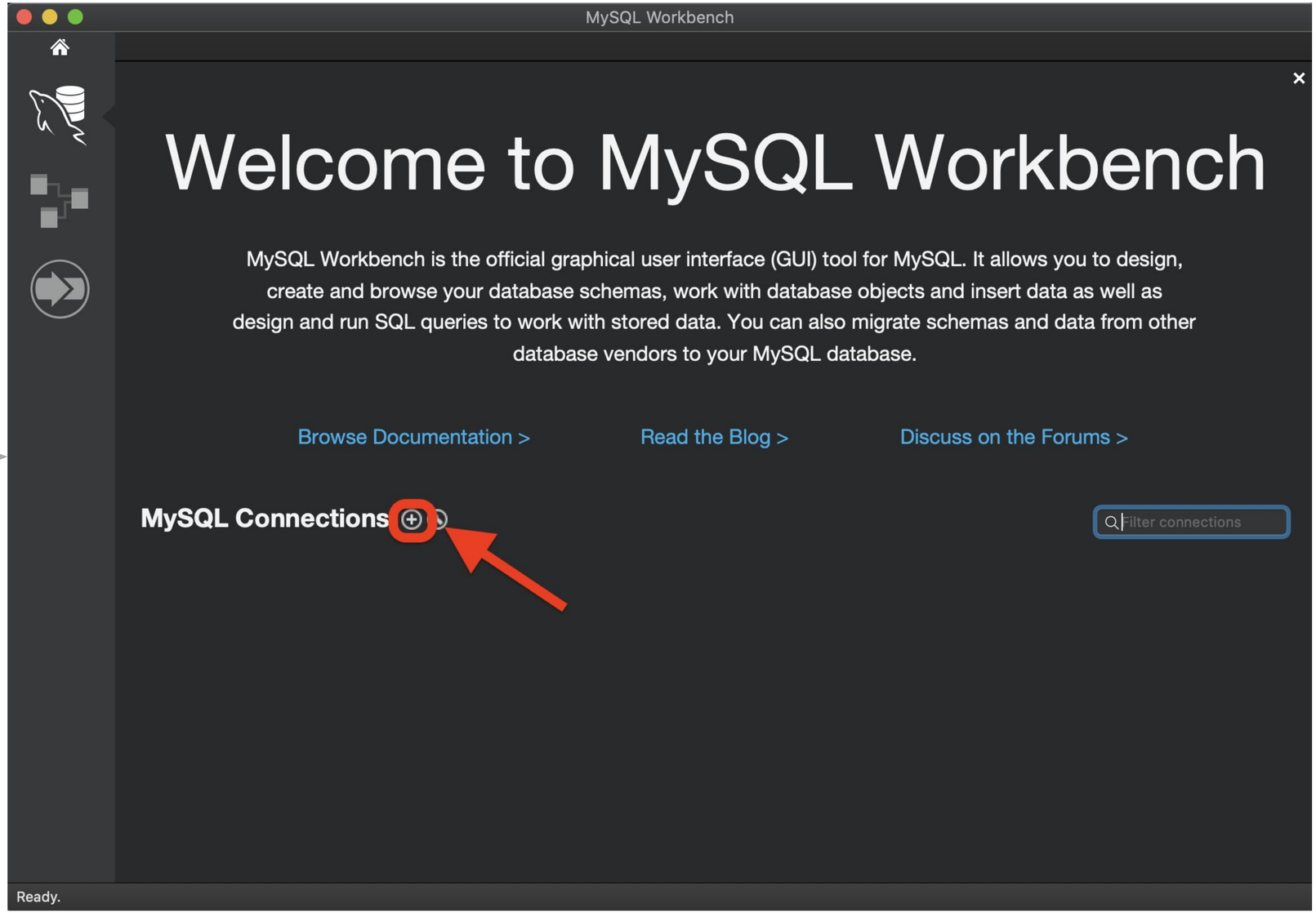
앞서 설치한 AWS RDS의 MySQL을 사용
SQL Workbench를 사용하여 SQL 실습

◆ MySQL Workbench를 사용 (1)

- ❖ MySQL 사이트에서 제공해주는 무료 클라이언트 SQL 에디터
- ❖ Download MySQL Workbench - MySQL
  - https://dev.mysql.com/downloads/workbench/
  - 윈도우와 맥과 리눅스 버전 제공
  - 뒤 슬라이드에 제공되는 정보로 실습 MySQL 서버에 연결

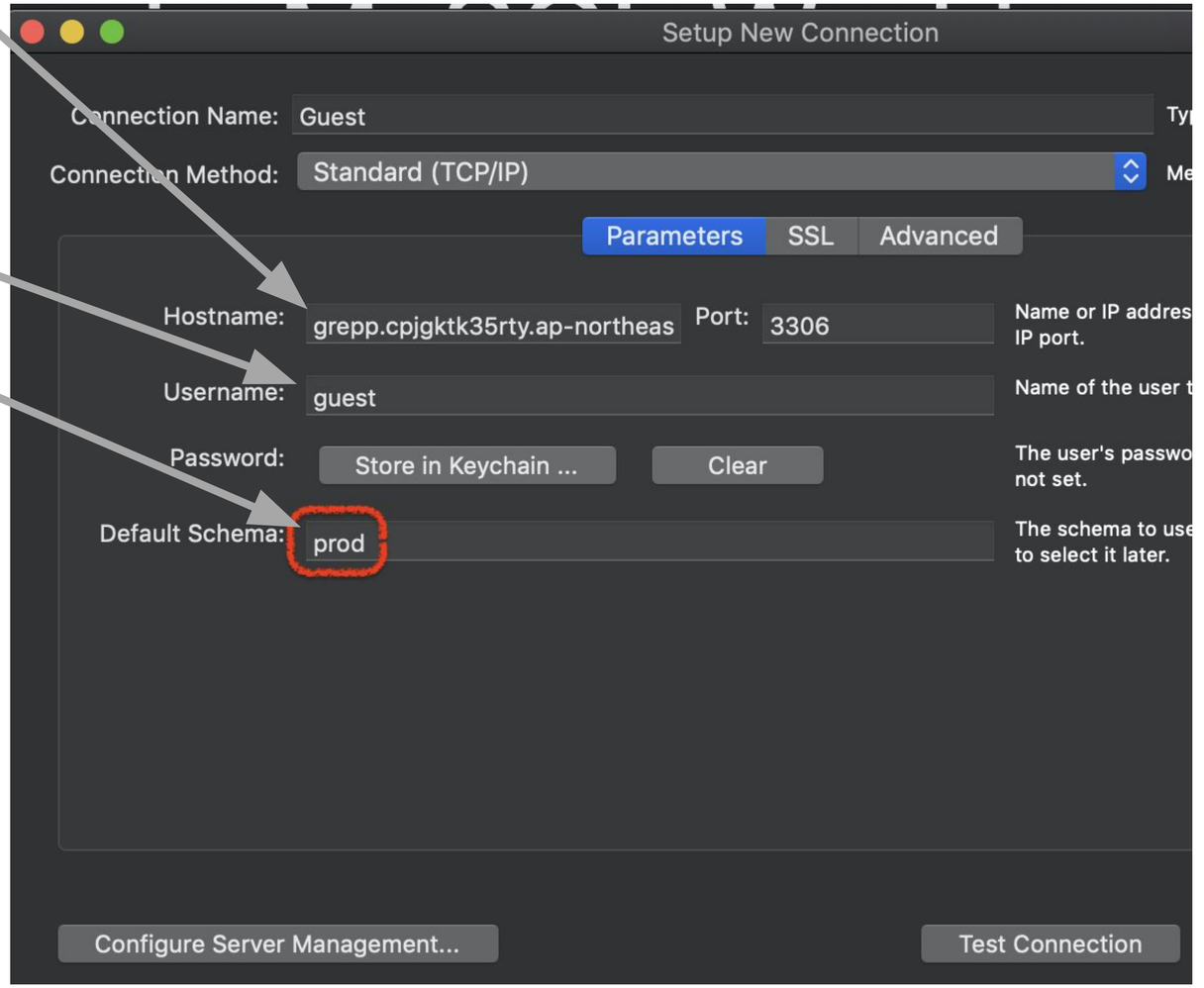
◆ MySQL Workbench를 사용 (2)

❖ 설치 과정



◆ 실습 MySQL 정보

- ❖ 호스트: grepp.cpjgktk35rty.ap-northeast-2.rds.amazonaws.com
- ❖ 포트번호: 3306
- ❖ 사용자 ID: guest
- ❖ 패스워드: Guest1!*
- ❖ 디폴트 데이터베이스: prod



programmers

SELECT 살펴보기

SQL DML의 기본이 되는 SELECT를 살펴보자
앞서 만든 session과 channel 테이블을 가지고 실습 예정

◆관계형 데이터베이스 예제: channel 테이블

id	channel
1	Instagram
2	Naver
3	Youtube
4	Google
5	Facebook
6	Tiktok
7	Unknown

◆관계형 데이터베이스 예제: session 테이블

id	user_id	created	channel_id
1	779	2019-05-01 00:36:00	5
2	230	2019-05-01 02:53:49	3
3	369	2019-05-01 12:18:27	6
4	248	2019-05-01 13:41:29	5
5	676	2019-05-01 14:17:54	2
6	40	2019-05-01 14:42:50	3
7	468	2019-05-01 15:08:16	1
8	69	2019-05-01 15:20:27	4

◆ SELECT를 사용하기 전에

- ❖ SHOW DATABASES;
- ❖ USE prod; -- 데이터베이스이름
- ❖ SHOW TABLES;

◆ SELECT (1)

- ❖ 테이블(들)에서 레코드들(혹은 레코드수)을 읽어오는데 사용
- ❖ WHERE를 사용해 조건을 만족하는 레코드

```
SELECT 필드이름1, 필드이름2, ...
FROM 테이블이름
WHERE 선택조건
GROUP BY 필드이름1, 필드이름2, ...
ORDER BY 필드이름 [ASC|DESC] -- 필드 이름 대신에 숫자 사용 가능
LIMIT N;
```

◆ SELECT (2)

```
SELECT *                    -- *는 모든 필드를 지칭하는 표현
FROM prod.session;     -- 앞서 USE prod;를 수행했다면 FROM session도 사용 가능
```

id	user_id	created	channel_id
1	571	2019-01-01 0:06:48	5
2	143	2019-01-01 0:08:31	3
3	39	2019-01-01 0:16:38	6
4	707	2019-01-01 0:21:47	5
5	435	2019-01-01 0:25:20	2
6	89	2019-01-01 0:29:37	3

◆ SELECT (3)

```
SELECT id, user_id, channel_id
FROM prod.session;
```

```
SELECT *
FROM prod.session
LIMIT 10;
```

◆ SELECT (4)

```
SELECT DISTINCT channel_id        -- 유일한 채널 ID를 알고 싶은 경우
FROM prod.session;


SELECT channel_id, COUNT(1)       -- 채널 ID별 카운트를 하려면 GROUP BY/COUNT 함수!!
FROM prod.session
GROUP BY 1;
```

◆ SELECT (5)

```
SELECT COUNT(1)                    -- 테이블의 모든 레코드 수 카운트. COUNT(*). 하나의 레코드
FROM prod.session;

SELECT *                           -- channel 테이블의 모든 레코드들을 표시
FROM prod.channel;

SELECT COUNT(1)
FROM prod.session
WHERE channel_id = 5;   -- channel이 Facebook경우만 레코드수 카운트
```

◆ CASE WHEN

❖ 필드 값의 변환을 위해 사용 가능

- CASE WHEN 조건 THEN 참일때 값 ELSE 거짓일때 값 END 필드이름

❖ 여러 조건을 사용하여 변환하는 것도 가능

```
CASE
   WHEN 조건1 THEN 값1
   WHEN 조건2 THEN 값2
   ELSE 값3
END 필드이름
```

```
SELECT channel_id, CASE
    WHEN channel_id in (1, 5, 6) THEN 'Social-Media'
    WHEN channel_id in (2, 4) THEN 'Search-Engine'
    ELSE 'Something-Else'
END channel_type
FROM prod.session;
```

◆ NULL이란?

- ❖ 값이 존재하지 않음을 나타내는 상수. 0 혹은 ""과는 다름
- ❖ 필드 지정시 값이 없는 경우 NULL로 지정 가능
  - 테이블 정의시 디폴트 값으로도 지정 가능
- ❖ 어떤 필드의 값이 NULL인지 아닌지 비교는 특수한 문법을 필요로 함
  - field1 is NULL 혹은 field1 is not NULL
- ❖ NULL이 사칙연산에 사용되면 그 결과는?
  - SELECT 0 + NULL, 0 - NULL, 0 * NULL, 0/NULL

◆ COUNT 함수 제대로 이해하기

value
NULL
1
1
0
0
4
3

테이블: **prod.count_test**

SELECT COUNT(1) FROM prod.count_test -> ??

SELECT COUNT(0) FROM prod.count_test -> ??

SELECT COUNT(NULL) FROM prod.count_test -> ??

SELECT COUNT(value) FROM prod.count_test -> ??

SELECT COUNT(DISTINCT value) FROM prod.count_test -> ??

WHERE

- IN
  - WHERE channel_id in (3, 4)
    - WHERE channel_id = 3 OR channel_id = 4
  - NOT IN
- LIKE
  - LIKE: 대소문자 구별 없이 문자열 매칭 기능을 제공해줌
  - WHERE channel LIKE 'G%' -> 'G*'
  - WHERE channel LIKE '%o%' -> '*o*'
  - NOT LIKE
- BETWEEN
  - 날짜 범위에 사용 가능
- 위의 오퍼레이터들은 CASE WHEN 사이에서도 사용가능

IN & LIKE/ILIKE

```
SELECT COUNT(1)
FROM prod.session
WHERE channel_id IN (4, 5);
```

```
SELECT COUNT(1)
FROM prod.channel
WHERE channel LIKE '%G%';
```

```
SELECT DISTINCT channel
FROM prod.channel
WHERE channel LIKE '%o%';
```

```
SELECT DISTINCT channel
FROM prod.channel
WHERE channel NOT LIKE '%o%';
```

STRING Functions

- LEFT(str, N)
- REPLACE(str, exp1, exp2)
- UPPER(str)
- LOWER(str)
- LENGTH(str)
- LPAD, RPAD
- SUBSTRING
- CONCAT

SQL 연습:

```
SELECT
    LENGTH(channel),
    UPPER(channel),
    LOWER(channel),
    LEFT(channel, 4),
    RPAD(channel, 15, '-'),
    LPAD(channel, 15, '-')
FROM prod.channel;
```

ORDER BY

- 디폴트 순서는 오름차순 (작은 값이 먼저 나옴)
  - ORDER BY 1 ASC
- 내림차순(Descending)을 원하면 “DESC”
  - ORDER BY 1 DESC
- 여러 개의 필드를 사용해서 정렬하려면
  - ORDER BY 1 DESC, 2, 3
- NULL 값 순서는?
  - NULL 값들은 오름차순 일 경우 (ASC), 처음에 위치함
  - NULL 값들은 내림차순 일 경우 (DESC) 마지막에 위치함

```
SELECT value
FROM prod.count_test
ORDER BY value DESC;

SELECT value
FROM prod.count_test
ORDER BY value ASC;
```

타입 변환 (1)

- DATE Conversion:
  - NOW
  - 타임존 관련 변환
    - CONVERT_TZ(now(), 'GMT', 'Asia/Seoul')
  - DATE, WEEK, MONTH, YEAR, HOUR, MINUTE, SECOND, QUARTER, MONTHNAME
  - DATEDIFF
  - DATE_ADD
  - ...
- STR_TO_DATE, DATE_FORMAT

타입 변환 (2)

```
SELECT
  created, CONVERT_TZ(created, 'GMT', 'Asia/Seoul') seoul_time,
  YEAR(created) y, QUARTER(created) q, MONTH(created) m, MONTHNAME(created) mnn,
  DATE(created) d, HOUR(created) h, MINUTE(created) m, SECOND(created) s
FROM session
LIMIT 10;
```

created	seoul_time	y	q	m	mn	d	h	m	s
2019-01-01 00:06:48	2019-01-01 09:06:48	2019	1	1	January	2019-01-01	0	6	48

타입 변환 (3)

```
SELECT created,
    DATEDIFF(now(), created) gap_in_days,
    DATE_ADD(created,  INTERVAL 10 DAY) ten_days_after_created
FROM session
LIMIT 10;
```

created	gap_in_days	ten_days_after_created
2019-01-01 00:06:48	936	2019-01-11 00:06:48

```
SELECT STR_TO_DATE('01,5,2013','%d,%m,%Y');  -- 2013-05-01
```

Type Casting

- 1/2의 결과는?
  - 0이 됨. 정수간의 연산은 정수가 되어야하기 때문
    - 분자나 분모 중의 하나를 float로 캐스팅해야 0.5가 나옴
    - 이는 프로그래밍 언어에서도 일반적으로 동일하게 동작함
  - 뒤에서 예제를 살펴볼 예정
- cast 함수를 사용
  - cast(category as float)
  - convert(expression, float)

```
SELECT cast('100.0' as float), convert('100.0', float);
```

programmers

SELECT 실습

앞서 배운 내용을 MySQL Workbench로 실습해보자

◆ SQL 실습

❖ MySQL Workbench로 실습

- 실습 SQL 링크

programmers

GROUP BY 살펴보기

SELECT시 실행시 결과를 특정 그룹별로 묶는 방법을 찾아보자

◆ GROUP BY & Aggregate 함수 (1)

❖ 테이블의 레코드를 그룹핑하여 그룹별로 다양한 정보를 계산

❖ 이는 두 단계로 이뤄짐

- 먼저 그룹핑을 할 필드를 결정 (하나 이상의 필드가 될 수 있음)
  - GROUP BY로 지정 (필드 이름을 사용하거나 필드 일련번호를 사용)
- 다음 그룹별로 계산할 내용을 결정
  - 여기서 Aggregate함수를 사용
  - COUNT, SUM, AVG, MIN, MAX, GROUP_CONCAT …
    - 보통 필드 이름을 지정하는 것이 일반적 (alias)

◆ GROUP BY & Aggregate 함수 (2)

❖ 월별 세션수를 계산하는 SQL

- prod.session을 사용 (id와 created 필드)

```
SELECT
    LEFT(created, 7) AS mon,
    COUNT(1) AS session_count
FROM prod.session
GROUP BY 1  -- GROUP BY mon, GROUP BY LEFT(created, 7)
ORDER BY 1;
```

◆ GROUP BY로 다음 문제를 풀어보자

❖ 앞서 설명한 prod.session와 prod.channel 테이블들을 사용

❖ 다음을 계산하는 SQL을 만들어 보자

- ~~월별 총 세션 수~~ (이미 풀어봤음)
- 가장 많이 사용된 채널은 무엇인가?
- 가장 많은 세션을 만들어낸 사용자 ID는 무엇인가?
- 월별 유니크한 사용자 수 (MAU - Monthly Active User)
  - 한 사용자는 한번만 카운트되어야 함
- 월별 채널별 유니크한 사용자 수

◆가장 많이 사용된 채널은 무엇인가? (1)

❖ 가장 많이 사용되었다는 정의는?

- 사용자 기반 아니면 세션 기반?

❖ 필요한 정보 - 채널 정보, 사용자 정보 혹은 세션 정보

❖ 먼저 어느 테이블을 사용해야하는지 생각!

- **prod.session?**
- prod.channel?
- 혹은 이 2개의 테이블을 조인해야하나?
  - channel_id로만 충분하다면 조인이 필요 없고 session만으로 충분

◆가장 많이 사용된 채널은 무엇인가? (2)

```
SELECT
    channel_id,
    COUNT(1) AS session_count,
    COUNT(DISTINCT user_id) AS user_count
FROM prod.session
GROUP BY 1                    -- GROUP BY channel_id
ORDER BY 2 DESC;              -- ORDER BY session_count DESC
```

◆가장 많은 세션을 만들어낸 사용자 ID는 무엇인가?

```
SELECT
    user_id,
    COUNT(1) AS count
FROM prod.session
GROUP BY 1                    -- GROUP BY user_id
ORDER BY 2 DESC               -- ORDER BY count DESC
LIMIT 1;
```

◆월별 유니크한 사용자 수

- ❖ 이게 바로 MAU(Monthly Active User)에 해당
- ❖ 필요한 정보 - 시간 정보, 사용자 정보

```
SELECT
    LEFT(created, 7) AS mon,
    COUNT(DISTINCT user_id) AS user_count
FROM prod.session
GROUP BY 1  -- GROUP BY mon, GROUP BY LEFT(created, 7)
ORDER BY 1;
```

◆월별 채널별 유니크한 사용자 수 (1)

- ❖ 필요한 정보 - 시간 정보, 사용자 정보, 채널 정보
- ❖ 먼저 어느 테이블을 사용해야하는지 생각!
- ❖ 이번에는 channel_id가 아닌 channel 이름으로 계산해보자
  - session, channel의 조인이 필요

월별 채널별 유니크한 사용자 수 (2): 일단 2 테이블의 조인부터 해결

```
SELECT s.id, s.user_id, s.created, s.channel_id, c.channel
FROM session s
JOIN channel c ON c.id = s.channel_id;
```

- Alias를 먼저 이해 (s, c)
- JOIN이란 결국 서로 다른 테이블에 존재하는 레코드들은 특정 조건을 바탕으로 병합하는 작업
  - 4일차에 심화학습 수행

id	user_id	created	channel_id	channel
7	363	2019-01-01 0:34:20	1	Instagram
9	333	2019-01-01 1:02:40	1	Instagram
40	642	2019-01-01 6:15:28	1	Instagram
44	229	2019-01-01 6:42:11	1	Instagram
48	9	2019-01-01 7:29:40	1	Instagram

월별 채널별 유니크한 사용자 수 (3)

```
SELECT
  LEFT(s.created, 7) AS mon,
  c.channel,
  COUNT(DISTINCT user_id) AS mau
 FROM session s
 JOIN channel c ON c.id = s.channel_id
GROUP BY 1, 2
ORDER BY 1 DESC, 2;
```

◆월별 채널별 유니크한 사용자 수 (4)

```
SELECT
  LEFT(s.created, 7) AS mon,
  c.channel,
  COUNT(DISTINCT user_id) AS mau
 FROM session s
 JOIN channel c ON c.id = s.channel_id
GROUP BY 1, 2
ORDER BY 1 DESC, 2;
```

COUNT의 동작을 잘 이해하는 것이 중요!
DISTINCT와 연동

월별 채널별 유니크한 사용자 수 (6)

```
SELECT
  LEFT(s.created, 7) AS mon,
  c.channel,
  COUNT(DISTINCT user_id) AS mau
 FROM session s
 JOIN channel c ON c.id = s.channel_id
GROUP BY 1, 2
ORDER BY 1 DESC, 2;
```

필드/테이블 이름에 Alias 사용. AS는 필수가 아님.
COUNT(DISTINCT s.user_id) AS mau와
COUNT(DISTINCT s.user_id) mau는 동일

월별 채널별 유니크한 사용자 수 (7)

```
SELECT
  LEFT(s.created, 7) AS mon,
  c.channel,
  COUNT(DISTINCT user_id) AS mau
 FROM session s
 JOIN channel c ON c.id = s.channel_id
GROUP BY 1, 2
ORDER BY 1 DESC, 2;
```

ORDER BY와 GROUP BY:

- 포지션 번호 vs. 필드 이름

GROUP BY 1 == GROUP BY mon == GROUP BY LEFT(s.created, 7)

programmers

GROUP BY 실습

◆ SQL 실습

❖ MySQL Workbench로 실습

- 실습 SQL 링크

강의 요약

실습환경 (AWS RDS)
SELECT
GROUP BY