SQL을 이용한 데이터 분석
3. GROUP BY와 CTAS

한기용

keeyonghan@hotmail.com

Contents

GROUP BY와 AGGREGATE 함수

◆ 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, LISTAGG, …
    - 보통 필드 이름을 지정하는 것이 일반적 (alias)

◆ GROUP BY & Aggregate 함수 (2)

❖ 월별 세션수를 계산하는 SQL

- raw_data.session_timestamp를 사용 (sessionId와 ts 필드)

```
SELECT
    LEFT(ts, 7) AS mon,
    COUNT(1) AS session_count
FROM raw_data.session_timestamp
GROUP BY 1  -- GROUP BY mon, GROUP BY LEFT(ts, 7)
ORDER BY 1;
```

◆ GROUP BY로 다음 문제를 풀어보자

❖ 앞서 설명한 raw_data.session_timestamp와 raw_data.user_session_channel 테이블들을 사용

❖ 다음을 계산하는 SQL을 만들어 보자

- ~~월별 총 세션 수~~ (이미 풀어봤음)
- 가장 많은 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!
  - **user_session_channel**?
  - session_timestamp?
  - 혹은 이 2개의 테이블을 조인해야하나?

◆가장 많이 사용된 채널은 무엇인가?

```
SELECT
    channel,
    COUNT(1) AS session_count,
    COUNT(DISTINCT userId) AS user_count
FROM raw_data.user_session_channel
GROUP BY 1                          -- GROUP BY channel
ORDER BY 2 DESC;                    -- ORDER BY session_count DESC
```

◆가장 많은 세션을 만들어낸 사용자 ID는 무엇인가? (1)

- ❖ 필요한 정보 - 세션 정보, 사용자 정보
- ❖ 먼저 어느 테이블을 사용해야하는지 생각!
  - **user_session_channel**?
  - session_timestamp?
  - 혹은 이 2개의 테이블을 조인해야하나?

◆가장 많은 세션을 만들어낸 사용자 ID는 무엇인가? (2)

```
SELECT
    userId,
    COUNT(1) AS count
FROM raw_data.user_session_channel
GROUP BY 1                    -- GROUP BY userId
ORDER BY 2 DESC               -- ORDER BY count DESC
LIMIT 1;
```

◆월별 유니크한 사용자 수 (1)

- ❖ 이게 바로 **MAU(Monthly Active User)**에 해당
- ❖ 필요한 정보 - 시간 정보, 사용자 정보
- ❖ 먼저 어느 테이블을 사용해야하는지 생각!
  - user_session_channel (userId, **sessionId**, channel)?
  - session_timestamp (**sessionId**, ts)?
  - 혹은 이 2개의 테이블을 조인해야하나?

userId	sessionId	channel	ts
779	7cdace91c487558e27ce54df7cdb299c	Instagram	2019-05-01 0:36:00
230	94f192dee566b018e0acf31e1f99a2d9	Naver	2019-05-01 2:53:49
369	7ed2d3454c5eea71148b11d0c25104ff	Youtube	2019-05-01 12:18:27
248	f1daf122cde863010844459363cd31db	Naver	2019-05-01 13:41:29

월별 유니크한 사용자 수 (2)

```
SELECT
  TO_CHAR(A.ts, 'YYYY-MM') AS month,
  COUNT(DISTINCT B.userid) AS mau
FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid
GROUP BY 1
ORDER BY 1 DESC;
```

TO_CHAR (A.ts, ‘YYYY-MM’)

- LEFT(A.ts, 7)
- DATE_TRUNC(‘month’, A.ts)
- SUBSTRING(A.ts, 1, 7)

장단점?

◆월별 유니크한 사용자 수 (3)

```
SELECT
  TO_CHAR(A.ts, 'YYYY-MM') AS month,
  COUNT(DISTINCT B.userid) AS mau
FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid
GROUP BY 1
ORDER BY 1 DESC;
```

COUNT의 동작을 잘 이해하는 것이
중요!
DISTINCT와 연동

월별 유니크한 사용자 수 (4)

```
SELECT
  TO_CHAR(A.ts, 'YYYY-MM') AS month,
  COUNT(DISTINCT B.userid) AS mau
FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid
GROUP BY 1
ORDER BY 1 DESC;
```

INNER JOIN vs. LEFT JOIN

월별 유니크한 사용자 수 (5)

```
SELECT
  TO_CHAR(A.ts, 'YYYY-MM') AS month,
  COUNT(DISTINCT B.userid) AS mau
FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid
GROUP BY 1
ORDER BY 1 DESC;
```

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

월별 유니크한 사용자 수 (6)

```
SELECT
  TO_CHAR(A.ts, 'YYYY-MM') AS month,
  COUNT(DISTINCT B.userid) AS mau
FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid
GROUP BY 1
ORDER BY 1 DESC;  -- ASC
```

ORDER BY와 GROUP BY:

- 포지션 번호 vs. 필드 이름

GROUP BY 1 == GROUP BY month == GROUP BY TO_CHAR(A.ts, 'YYYY-MM')

◆월별 채널별 유니크한 사용자 수 (1)

❖ 필요한 정보 - 시간 정보, 사용자 정보, 채널 정보

❖ 먼저 어느 테이블을 사용해야하는지 생각!

- user_session_channel (userId, **sessionId**, **channel**)?
- session_timestamp (**sessionId**, ts)?
- 혹은 이 2개의 테이블을 조인해야하나?

userId	sessionId	channel	ts
779	7cdace91c487558e27ce54df7cdb299c	Instagram	2019-05-01 0:36:00
230	94f192dee566b018e0acf31e1f99a2d9	Naver	2019-05-01 2:53:49
369	7ed2d3454c5eea71148b11d0c25104ff	Youtube	2019-05-01 12:18:27
248	f1daf122cde863010844459363cd31db	Naver	2019-05-01 13:41:29

월별 채널별 유니크한 사용자 수 (2)

```
SELECT
  TO_CHAR(A.ts, 'YYYY-MM') AS month,
  channel,
  COUNT(DISTINCT B.userid) AS mau
FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid
GROUP BY 1, 2
ORDER BY 1 DESC, 2;
```

◆ SQL 실습

❖ 구글 Colab 실습 링크:

- https://colab.research.google.com/drive/1ig4v_NMel4rD3a9MHYNCkjNgVp-izJul#scrollTo=wOS9-QY1amnt

CTAS와 CTE 소개

◆CTAS: SELECT를 가지고 테이블 생성

- ❖ 간단하게 새로운 테이블을 만드는 방법
- ❖ 자주 조인하는 테이블들이 있다면 이를 CTAS를 사용해서 조인해두면 편리해짐

```
DROP TABLE IF EXISTS adhoc.keeyong_session_summary;
CREATE TABLE adhoc.keeyong_session_summary AS
SELECT B.*, A.ts FROM raw_data.session_timestamp A
JOIN raw_data.user_session_channel B ON A.sessionid = B.sessionid;
```

◆월별 유니크한 사용자 수를 다시 풀어보기

```
SELECT
  TO_CHAR(ts, 'YYYY-MM') AS month,
  COUNT(DISTINCT userid) AS mau
FROM adhoc.keeyong_session_summary
GROUP BY 1
ORDER BY 1 DESC;
```

◆ 항상 시도해봐야하는 데이터 품질 확인 방법들

- ❖ 중복된 레코드들 체크하기
- ❖ 최근 데이터의 존재 여부 체크하기 (freshness)
- ❖ Primary key uniqueness가 지켜지는지 체크하기
- ❖ 값이 비어있는 컬럼들이 있는지 체크하기

◆ 중복된 레코드들 체크하기 (1)

❖ 다음 두 개의 카운트를 비교

```
SELECT COUNT(1)
FROM adhoc.keeyong_session_summary;


SELECT COUNT(1)
FROM (
    SELECT DISTINCT userId, sessionId, ts, channel
    FROM adhoc.keeyong_session_summary
);
```

◆ 중복된 레코드들 체크하기 (2)

❖ CTE를 사용해서 중복 제거 후 카운트 해보기

```
With ds AS (
    SELECT DISTINCT userId, sessionId, ts, channel
    FROM adhoc.keeyong_session_summary
)
SELECT COUNT(1)
FROM ds;
```

◆ 최근 데이터의 존재 여부 체크하기 (freshness)

```
SELECT MIN(ts), MAX(ts)
FROM adhoc.keeyong_session_summary;
```

◆ Primary key uniqueness가 지켜지는지 체크하기

```
SELECT sessionId, COUNT(1)
FROM adhoc.keeyong_session_summary
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

◆ 값이 비어있는 컬럼들이 있는지 체크하기

```
SELECT
  COUNT(CASE WHEN sessionId is NULL THEN 1 END) sessionid_null_count,
  COUNT(CASE WHEN userId is NULL THEN 1 END) userid_null_count,
  COUNT(CASE WHEN ts is NULL THEN 1 END) ts_null_count,
  COUNT(CASE WHEN channel is NULL THEN 1 END) channel_null_count
FROM adhoc.keeyong_session_summary;
```

◆ SQL 실습

❖ 구글 Colab 실습 링크:

- https://colab.research.google.com/drive/1ig4v_NMel4rD3a9MHYNCkjNgVp-izJul#scrollTo=pll9oUgD0UQu

◆ 값이 비어있는 컬럼들이 있는지 체크하기

```
SELECT
  COUNT(CASE WHEN sessionId is NULL THEN 1 END) sessionid_null_count,
  COUNT(CASE WHEN userId is NULL THEN 1 END) userid_null_count,
  COUNT(CASE WHEN ts is NULL THEN 1 END) ts_null_count,
  COUNT(CASE WHEN channel is NULL THEN 1 END) channel_null_count
FROM adhoc.keeyong_session_summary;
```

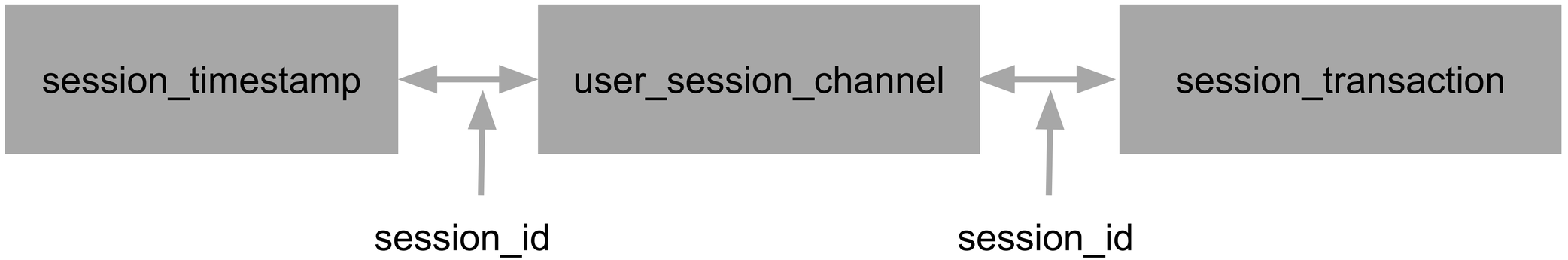
숙제

◆ 2개의 새 테이블 소개 (1)

❖ 지금까지 session_timestamp와 user_session_channel을 사용

```
CREATE TABLE raw_data.session_transaction (
    sessionid varchar(32),
    refunded boolean,
    amount int
);
```

```
CREATE TABLE raw_data.channel (
    channelName varchar(32)
);
```



◆ 2개의 새 테이블 소개 (2)

❖ 총 4개의 테이블 모두 소개

Table	Fields
session_timestamp	sessionid (string), ts (timestamp)
user_session_channel	userid (integer), sessionid (string), channel (string)
session_transaction	**sessionid (string), refunded (boolean), amount (integer)**
channel	**channelname (string)**

◆ 숙제 #1

❖ 채널별 월별 매출액 테이블 만들기 (adhoc 밑에 CTAS로 본인이름을 포함한 테이블로 만들기)

- session_timestamp,user_session_channel,session_transaction 사용
- 아래와 같은 필드로 구성
  - month
  - channel
  - uniqueUsers (총방문 사용자)
  - paidUsers (구매 사용자: refund한 경우도 판매로 고려)
  - conversionRate (구매사용자 / 총방문 사용자)
  - grossRevenue (refund 포함)
  - netRevenue (refund 제외)