codecademy

Attribution Queries

Learn SQL from Scratch

Jenessa Bettencourt

September 4th, 2018

Table of Contents

1. Introduction

In this project, I will be assisting **CoolTshirts** with their advertising campaigns.

To do this, I will be analyzing site traffic data to determine which of their advertising campaigns brought in the most traffic, and from there, which campaigns they should reinvest in.

To begin, let’s see how many campaigns I’m working with.

What this does, is count how many unique values are in the column named ‘utm_campaign’, which would give me the amount of campaigns that **CoolTshirts** has active.

```
SELECT COUNT (DISTINCT
utm_campaign)
FROM page_visits;
```

COUNT(DISTINCT utm_campaign)
8

Next, I need to look at where the traffic is coming from. To do this, I used:

This accomplishes the same thing as the slide before, except that it’s counting the values from the column ‘utm_source’

```
SELECT COUNT (DISTINCT utm_source)
FROM page_visits;
```

COUNT(DISTINCT utm_source)
6

Lets see how those two columns relate to each other.

utm_source	utm_campaign
nytimes	Getting-to-know-cool-tshirts
email	weekly-newsletter
buzzfeed	ten-crazy-cool-tshirts-facts
email	retargetting-campaign
facebook	retargetting-ad
medium	interview-with-cool-tshirts-founder
google	paid-search
google	cool-tshirts-search

```
SELECT DISTINCT utm_source,
utm_campaign
FROM page_visits;
```

Now **CoolTshirts** doesn't just have one single page on their website. Users move around to different pages depending on what step of the shopping process they are in.

CoolTshirts has 4 different pages.

utm_campaign
1 - landing_page
2 - shopping_cart
3 - checkout
4 - purchase

```
SELECT DISTINCT page_name
FROM page_visits;
```

2. User Journey

Now that I know more about **CoolTshirts** and the data they collect, I can analyze how a user navigates to and through the **CoolTshirts** website. This is called the ‘user journey’.

First, I’ll focus on how the user got to the website.

When someone clicks on a link to **CoolTshirts** from one of the advertising campaigns, and arrives at the landing page for the first time, this is called a user’s ‘first touch’.

I want to know how many first touches each source and campaign is responsible for:

source	campaign	Total first touches
medium	interview-with-cool-tshirts-founder	622
nytimes	getting-to-know-cool-tshirts	612
buzzfeed	ten-crazy-cool-tshirts-facts	576
google	cool-tshirts-search	169

To get this result, first, I created two temporary tables with columns from the main table page_visits.

In the first table ‘first_touch’, I took the id of the user, and the first timestamp that user is associated with using the MIN() function. That function picks out the smallest value in the column I give it.

```
WITH first_touch AS (
        SELECT user_id,
           MIN(timestamp) AS first_touch_at
        FROM page_visits
        GROUP BY user_id),

      ft_attr AS (
        SELECT ft.user_id,
         ft.first_touch_at,
         pv.utm_source,
         pv.utm_campaign
      FROM first_touch ft

      JOIN page_visits pv
      ON ft.user_id = pv.user_id
        AND ft.first_touch_at = pv.timestamp)
      SELECT ft_attr.utm_source AS ‘source’,
        Ft_attr.utm_campaign AS ‘campaign’,
        COUNT(*) AS ‘total first touches’
      FROM ft_attr
      GROUP BY 1, 2
      ORDER BY 3 desc;
```

To get this result, first, I created two temporary tables with columns from the main table page_visits.

In the first table ‘first_touch’, I took the id of the user, and the first timestamp that user is associated with using the MIN() function. That function picks out the smallest value in the column I give it.

The second table, ‘ft_attr’ (first touch attribute), was created with the columns from the first temporary table, user_id and first_touch_at, and the columns umt_source and utm_campaign from the main table ‘page_visits’.

```
WITH first_touch AS (
        SELECT user_id,
           MIN(timestamp) AS first_touch_at
        FROM page_visits
        GROUP BY user_id),

      ft_attr AS (
        SELECT ft.user_id,
         ft.first_touch_at,
         pv.utm_source,
         pv.utm_campaign
      FROM first_touch ft

      JOIN page_visits pv
      ON ft.user_id = pv.user_id
        AND ft.first_touch_at = pv.timestamp)
      SELECT ft_attr.utm_source AS ‘source’,
        Ft_attr.utm_campaign AS ‘campaign’,
        COUNT(*) AS ‘total first touches’
      FROM ft_attr
      GROUP BY 1, 2
      ORDER BY 3 desc;
```

To get this result, first, I created two temporary tables with columns from the main table page_visits.

In the first table ‘first_touch’, I took the id of the user, and the first timestamp that user is associated with using the MIN() function. That function picks out the smallest value in the column I give it.

The second table, ‘ft_attr’ (first touch attribute), was created with the columns from the first temporary table, user_id and first_touch_at, and the columns umt_source and utm_campaign from the main table ‘page_visits’.

Then I joined these two tables and joined them back into the page_visits table, counted the rows, and grouped it by each source and campaign, then sorted that result to show me the results in descending order.

```
WITH first_touch AS (
        SELECT user_id,
           MIN(timestamp) AS first_touch_at
        FROM page_visits
        GROUP BY user_id),

      ft_attr AS (
        SELECT ft.user_id,
         ft.first_touch_at,
         pv.utm_source,
         pv.utm_campaign
      FROM first_touch ft

      JOIN page_visits pv
      ON ft.user_id = pv.user_id
        AND ft.first_touch_at = pv.timestamp)
      SELECT ft_attr.utm_source AS ‘source’,
        Ft_attr.utm_campaign AS ‘campaign’,
        COUNT(*) AS ‘total first touches’
      FROM ft_attr
      GROUP BY 1, 2
      ORDER BY 3 desc;
```

Now I know how much new traffic **CoolTshirts** has received from their advertising campaigns, I'm going to move on to how many purchases each campaign led to. This is called 'last_touch'.

source	campaign	Total last touches
email	weekly-newsletter	447
facebook	retargetting-ad	443
email	retargetting-campaign	245
nytimes	getting-to-know-cool-tshirts	232
buzzfeed	ten-crazy-cool-tshirts-facts	190
medium	interview-with-cool-tshirts-founder	184
google	paid-search	178
google	cool-tshirts-search	60

The last_touch code is pretty much the same as the first_touch code, except I have changed the MIN() function with the MAX() function. That results in the highest timestamp associated with a user.

```
WITH last_touch AS (
       SELECT user_id,
        MAX(timestamp) AS last_touch_at
       FROM page_visits
       GROUP BY user_id),

      lt_attr AS (
       SELECT lt.user_id,
        lt.last_touch_at,
        pv.utm_source,
        pv.utm_campaign
       FROM last_touch lt

       JOIN page_visits pv
       ON lt.user_id = pv.user_id
        AND lt.last_touch_at = pv.timestamp)
        SELECT lt_attr.utm_source AS 'source',
        lt_attr.utm_campaign AS 'campaign',
        count(*) AS ' total last touches'
       FROM lt_attr
       GROUP BY 1, 2
       ORDER BY 3 desc;
```

Now I want to know the amount of purchases made.

This selects all the rows where a user made it to the purchase page and counts them.

```
SELECT COUNT(*) AS 'purchases'
FROM page_visits
WHERE page_name = '4 - purchase';
```

purchases
361

Now I will combine the two codes to find out how many purchases each campaign is associated with.

source	campaign	Total purchases
email	weekly-newsletter	115
facebook	retargetting-ad	113
email	retargetting-campaign	54
google	paid-search	52
buzzfeed	ten-crazy-cool-tshirts-facts	9
nytimes	getting-to-know-cool-tshirts-founder	9
medium	interview-with-cool-tshirts-founder	7
google	cool-tshirts-search	2

```
WITH last_touch AS (
       SELECT user_id,
        MAX(timestamp) AS last_touch_at
       FROM page_visits
       WHERE page_name = "4 - purchase"
       GROUP BY user_id),

      lt_attr AS (
       SELECT lt.user_id,
        lt.last_touch_at,
        pv.utm_source,
        pv.utm_campaign
       FROM last_touch lt

       JOIN page_visits pv
       ON lt.user_id = pv.user_id
        AND lt.last_touch_at =
pv.timestamp)
        SELECT lt_attr.utm_source AS
'source',
        lt_attr.utm_campaign AS
'campaign',
        COUNT(*) AS 'total purchases'
       FROM lt_attr
       GROUP BY 1, 2
       ORDER BY 3 desc;
```

3. Conclusion

From this data, I can give **CoolTshirts** five campaigns I think they should reinvest in.

1. Interview-with-cool-tshirts-founder from Medium
2. Getting-to-know-cool-tshirts from New York Times
3. Weekly-newsletter from email
4. Retargeting-campaign from email
5. Retargeting-ad from facebook
6.

Interview-with-cool-tshirts from Medium and getting-to-know-cool-tshirts from New York Times brought in the most new traffic to the site. They provided **CoolTshirts** with the most new potential customers out of all of the other campaigns, and I think that is worth investing in.

The weekly-newsletter from email, and the two retargeting campaigns from email and facebook brought in the most purchases, and also should be invested in.