codecademy

First- and Last-Touch Attribution with CoolTShirts.com

Learn SQL from Scratch

Elena Pellín Carcelén

July 10th

Contents

1. Get familiar with the company
2. What is the user journey?
3. Optimize the campaign budget

1. Get familiar with the company

1.1 utm_campaign vs utm_source

How many campaigns and sources does CoolTShirts use and how are they related? Be sure to explain the difference between utm_campaignand utm_source.

- We use the first query to know how many different campaigns there are on the table page_visits. We just need to use the COUNT() aggregated function.
- The purpose of the second query is to know how many different sources are on the table page_visits. It is the same as the first one, we just change utm_campaign for utm_source.
- In the third query we select just the two columns that we are interested in from the table. This way we can see which source is related to each campaign.

The difference between campaign and source is that the first one identifies the specific ad or email blast, and the second one refers to which site sent the traffic.

```
SELECT COUNT(DISTINCT utm_campaign) AS 'Number of
campaigns'
FROM page_visits;

SELECT COUNT(DISTINCT utm_source) AS 'Number of
sources'
FROM page_visits;

SELECT DISTINCT utm_campaign, utm_source
FROM page_visits;
```

Query Results

Number of campaigns
8

Number of sources
6

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

1.2 What pages are on their website?

We select just the column page_name from the table page_visits, but we need to write DISTINCT first in order to discard repetitions of the same values. In conclusion we have 4 different pages.

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

Query Results
page_name
1 - landing_page
2 - shopping_cart
3 - checkout
4 - purchase

2. What is the user journey?

2.1 How many first touches is each campaign responsible for?

First of all, we have created a temporary table using the WITH clause. In this table we store the timestamp of the first touch for each user, which we obtain with their smallest timestamp.
Next we use an INNER JOIN to match each row of the original table with rows of the temporary one based on common columns, with the corresponding relations (ON… AND…).
This query let us group the contents by campaign (just those which are in both tables), and count how many rows we have for each campaign with the COUNT() aggregation function. Bearing in mind that we only have rows corresponding to the first touch of each user, the “followings touches” of each user do not appear in this inner join.

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

SELECT pv.utm_campaign, COUNT(ft.first_touch_at) AS
    'Number of first-touch'
FROM first_touch ft
JOIN page_visits pv
    ON ft.user_id = pv.user_id
    AND ft.first_touch_at = pv.timestamp
GROUP BY 1
ORDER BY 2 DESC;
```

Query Results

utm_campaign	Number of first-touch
interview-with-cool-tshirts-founder	622
getting-to-know-cool-tshirts	612
ten-crazy-cool-tshirts-facts	576
cool-tshirts-search	169

2.2 How many last touches is each campaign responsible for?

First of all, we have created a temporary table using the WITH clause. In this table we store the timestamp of the last touch for each user, which we obtain with their biggest timestamp.
Next we use an INNER JOIN to match each row of the original table with rows of the temporary one based on common columns, with the corresponding relations (ON… AND…).
This query let us group the contents by campaign (just those which are in both tables, in this case, all of them), and count how many rows we have for each campaign with the COUNT() aggregation function. Bearing in mind that we only have rows corresponding to the last touch of each user, the “previous touches” of each user do not appear in this inner join.

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

SELECT pv.utm_campaign, COUNT(lt.last_touch_at) AS
    'Number of last-touch'
FROM last_touch lt
JOIN page_visits pv
    ON lt.user_id = pv.user_id
    AND lt.last_touch_at = pv.timestamp
GROUP BY 1
ORDER BY 2 DESC;
```

Query Results

utm_campaign	Number of last-touch
weekly-newsletter	447
retargetting-ad	443
retargetting-campaign	245
getting-to-know-cool-tshirts	232
ten-crazy-cool-tshirts-facts	190
interview-with-cool-tshirts-founder	184
paid-search	178
cool-tshirts-search	60

2.3 How many visitors make a purchase?

This query will show the amount of different users who arrived to the page_name ‘4 - purchase’, in other words, users who have made a purchase.
To make sure we are not going to repeat users we use the DISTINCT command before the name of the column. Moreover we use WHERE to filter just the rows where the page_name is ‘4 - purchase’.

```
SELECT COUNT(DISTINCT user_id) AS 'Purchases'
FROM page_visits
WHERE page_name = '4 - purchase';
```

Query Results
Purchases
361

2.4 How many last touches *on the purchase page* is each campaign responsible for?

First of all, we have created a temporary table using the WITH clause. In this table we store the timestamp of the last touch for each user, which we obtain with their biggest timestamp, and we add the filter where the page_name is '4 - purchase'.
Next we use an INNER JOIN to match each row of the original table with rows of the temporary one based on common columns, with the corresponding relations (ON… AND…).
This query let us group the contents by campaign, and count how many rows we have for each campaign with the COUNT() aggregation function. Bearing in mind that we only have rows corresponding to the page_name '4 - purchase' and the last touch of each user, the "previous touches" of each user do not appear in this inner join.

```
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
)

SELECT pv.utm_campaign, COUNT(lt.last_touch_at) AS
    'Last-touch on Purchase'
FROM last_touch lt
JOIN page_visits pv
    ON lt.user_id = pv.user_id
    AND lt.last_touch_at = pv.timestamp
GROUP BY 1
ORDER BY 2 DESC;
```

Query Results

utm_campaign	Last-touch on Purchase
weekly-newsletter	115
retargetting-ad	113
retargetting-campaign	54
paid-search	52
getting-to-know-cool-tshirts	9
ten-crazy-cool-tshirts-facts	9
interview-with-cool-tshirts-founder	7
cool-tshirts-search	2

2.5 What is the typical user journey?

We want to get the funnel of CoolTShirts, analyzing how many users complete a series of steps and which steps have the most number of users giving up.
With this query we are going to group the content by page_name, so we can calculate how many users have complete each step.
If we divide the number of people completing each step by the number of people completing the previous step:
1 - 100%
2 - 95%
3 - 76%
4 - 25%
We see that the two first pages have a high completion rates, and the two last ones a lower rate, specially the purchase page.

```
SELECT page_name, COUNT(DISTINCT user_id) AS 'Number
   of users'
FROM page_visits
GROUP BY 1;
```

Query Results	
page_name	**Number of users**
1 - landing_page	1979
2 - shopping_cart	1881
3 - checkout	1431
4 - purchase	361

3. Optimize the campaign budget

3.1 CoolTShirts can re-invest in 5 campaigns. Which should they pick and why?

First of all, we have created two temporary tables using the WITH clause. The first one store the timestamp of the first touch for each user, which we obtain with their smallest timestamp. The second one store the timestamp of the last touch for each user, which we obtain with their biggest timestamp.

Next we use two LEFT JOIN to match each row of the original table with rows of the temporary ones based on common columns, with the corresponding relations (ON… AND…). The resulting table will only contain all data from the original table, and additional data from the second tables.

The most important touches are the first and last one for each user. So we calculate the sum of both corresponding to each campaign. We sort all rows in descending order to see in first place the campaign with more first and last touches. Finally, we limit the result to 5 rows, because we are going to re-invest just in 5 campaigns, and those are the most relevant.

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

SELECT pv.utm_campaign,
    (IFNULL(COUNT(ft.first_touch_at), 0) +
    COUNT(lt.last_touch_at)) AS 'Sum of users'
FROM page_visits pv
LEFT JOIN first_touch ft
    ON pv.user_id = ft.user_id
    AND pv.timestamp = ft.first_touch_at
LEFT JOIN last_touch lt
    ON pv.user_id = lt.user_id
    AND pv.timestamp = lt.last_touch_at
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

Query Results

utm_campaign	Sum of users
getting-to-know-cool-tshirts	844
interview-with-cool-tshirts-founder	806
ten-crazy-cool-tshirts-facts	766
weekly-newsletter	447
retargetting-ad	443