First and Last Touch Attributions for CoolTShirts campaigns.

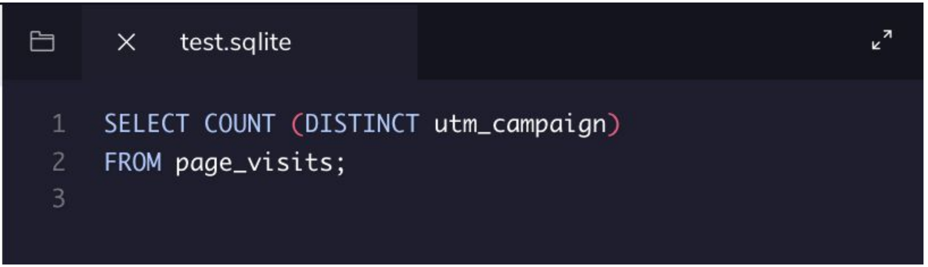
Gabriel Pena

(Fictional Data / Codecademy project)

CoolTShirts Campaign count and Campaigns

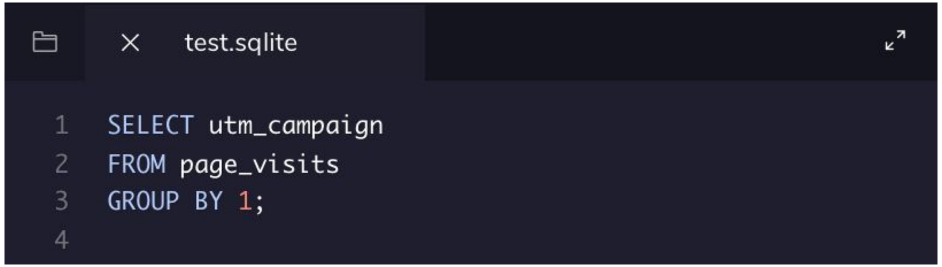
CoolTShirts has a total of 8 campaigns. To get this we use a COUNT statement but don't want the duplicate campaigns counted so we include a DISTINCT clause for the utm_column, counting unique campaigns only.

We can also list the 8 campaigns by doing a GROUP BY on utm_campaign column, and referencing that column as 1.



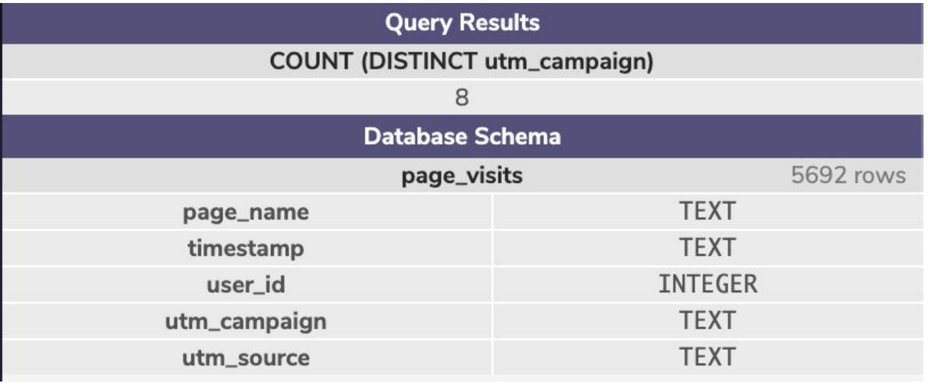
test.sqlite

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



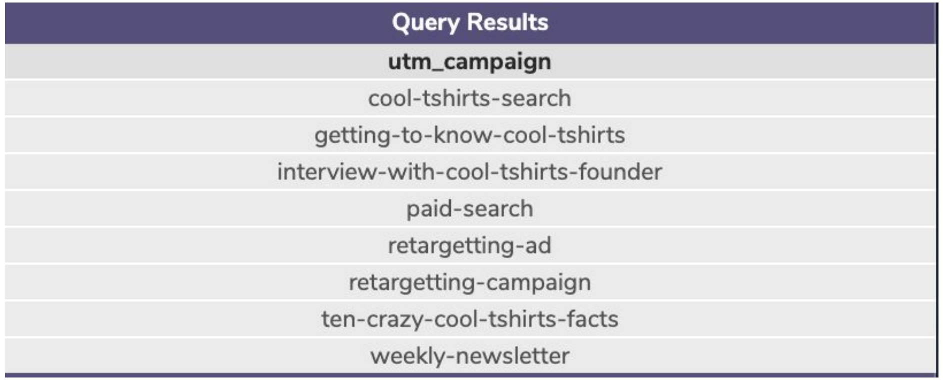
test.sqlite

```sql
SELECT utm_campaign
FROM page_visits
GROUP BY 1;
```



Query Results
COUNT (DISTINCT utm_campaign)
8

Database Schema	
page_visits	5692 rows
page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT

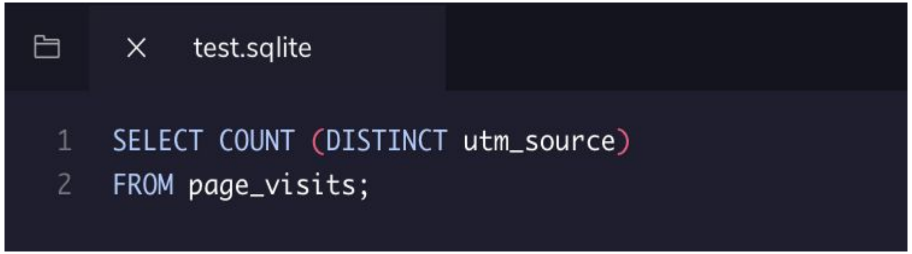


Query Results
utm_campaign
cool-tshirts-search
getting-to-know-cool-tshirts
interview-with-cool-tshirts-founder
paid-search
retargetting-ad
retargetting-campaign
ten-crazy-cool-tshirts-facts
weekly-newsletter

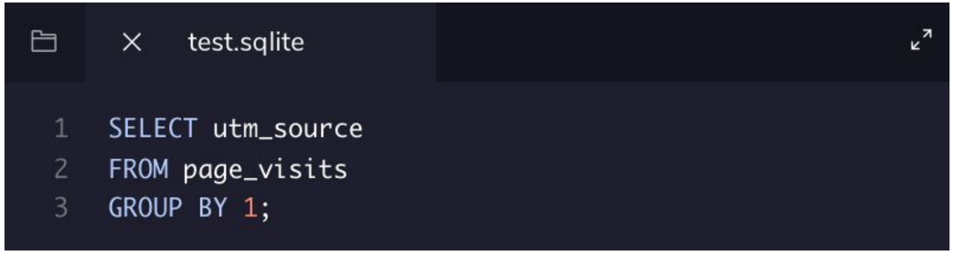
CoolTShirts Source count and Sources

CoolTShirts utilize 6 sources to reach out to users and can be found using another COUNT statement of unique sources. To make sure that only unique sources are counted, we need to use a DISTINCT clause for the utm_column.

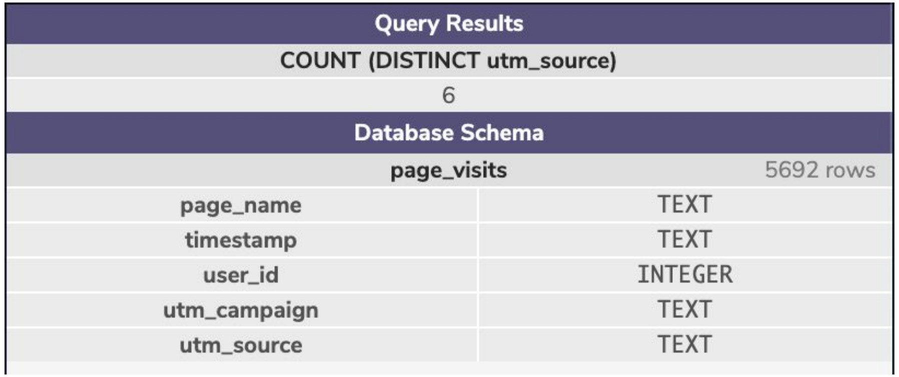
We can also display the 6 sources by using a GROUP BY on the utm_source column, putting the all repeated sources into groups.


test.sqlite

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


test.sqlite

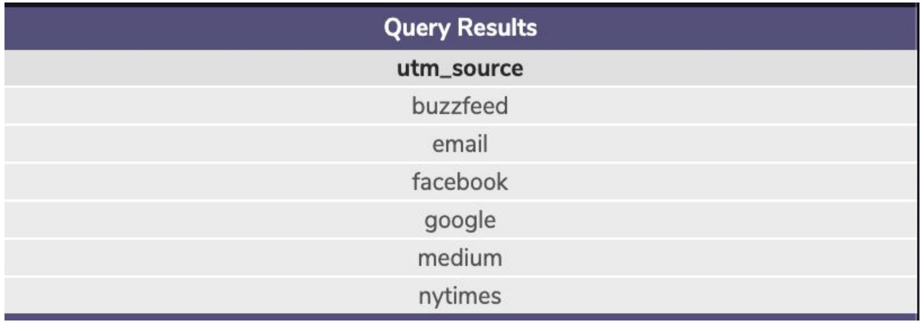
```sql
SELECT utm_source
FROM page_visits
GROUP BY 1;
```


Query Results

COUNT (DISTINCT utm_source)
6

Database Schema

page_visits	5692 rows
page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT


Query Results

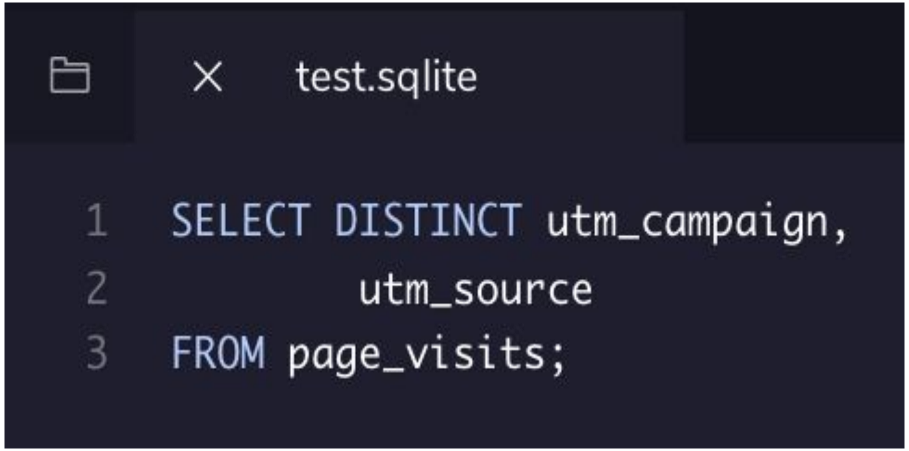
utm_source
buzzfeed
email
facebook
google
medium
nytimes

utm _source and utm_campaign

utm_campaign : the column that identifies the ad, email blast, or pop-up. Also known as traffic

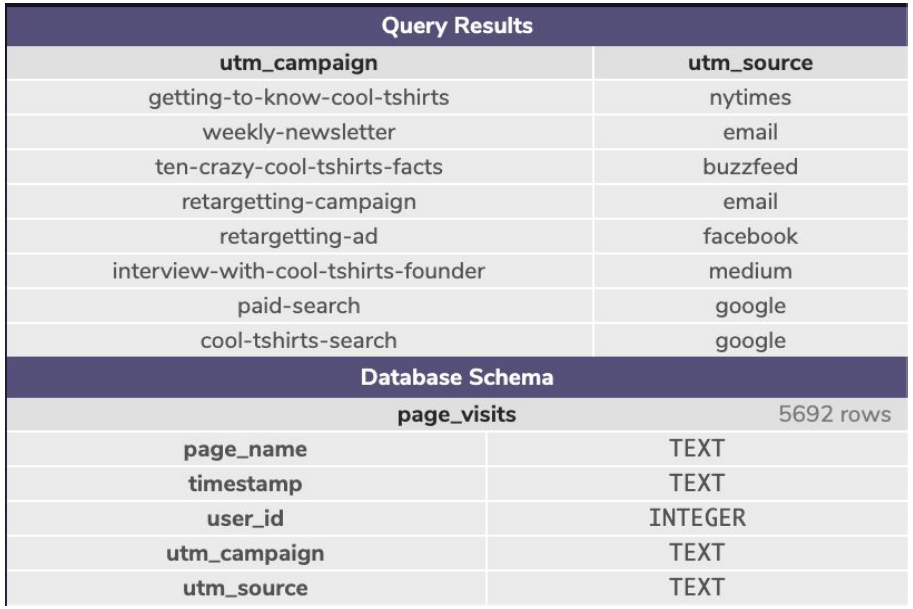
utm_source : the column that identifies what site sent out the traffic.

Together they can allow us to figure out how each campaign performs from the source it is used in allowing us to make a better decision of what investments to make for campaigns.



test.sqlite

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



Query Results

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

Database Schema

page_visits 5692 rows

page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT

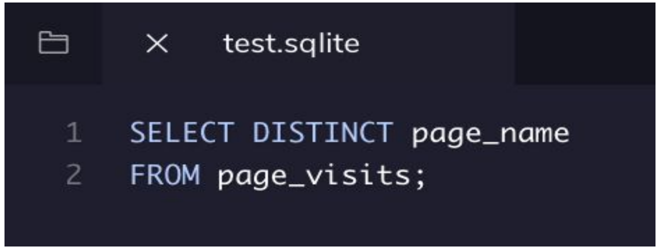
CoolTShirts Website Pages

To find the pages users navigate through on the CoolTShirts website, we can use two types of queries that will return 4 pages:

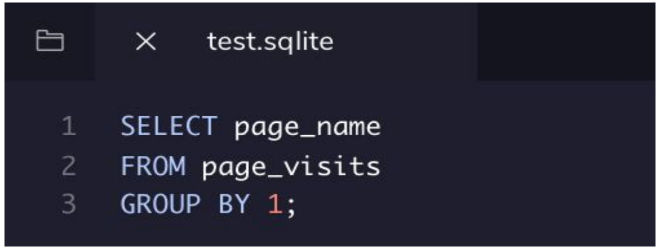
1- landing_page // 2 - shopping_cart // 3 - checkout // 4 - purchase.

We can SELECT all unique values under the page_name column using the DISTINCT clause.

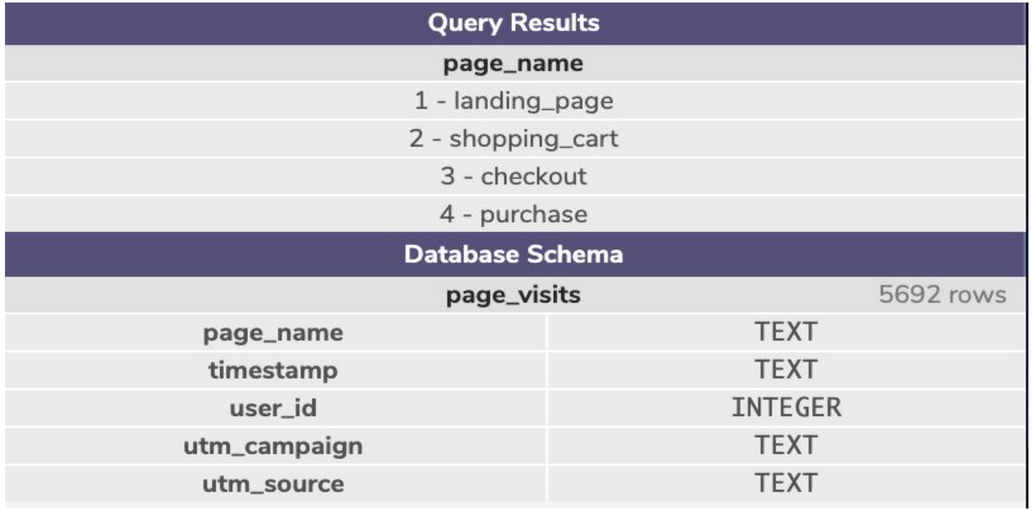
We can GROUP BY the unique values in the page_name column and reference the column as 1.



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



```sql
SELECT page_name
FROM page_visits
GROUP BY 1;
```



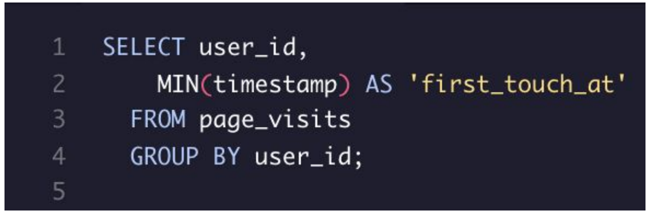
Query Results
page_name
1 - landing_page
2 - shopping_cart
3 - checkout
4 - purchase

Database Schema

page_visits 5692 rows

page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT

Count of first touches for each campaign



```sql
SELECT user_id,
    MIN(timestamp) AS 'first_touch_at'
  FROM page_visits
  GROUP BY user_id;
```

To find how many touches each campaign is responsible for we first need to find each users earliest timestamp, their first touch, whos query SELECTs user_id column from page_visits table and a column that aggregates the earliest timestamp for each user, but will only tell us the first time each user visited CoolTShirt using their earliest timestamp, but, not how they got there.

To find how users got to the site, UTM parameter will need to be established using a JOIN clause to combine the first touch results, now a temporary table as 'first_touch', with the original 'page_visits' table, ON the condition that:

First_touch column user_id equals to page_visits column user_id

And

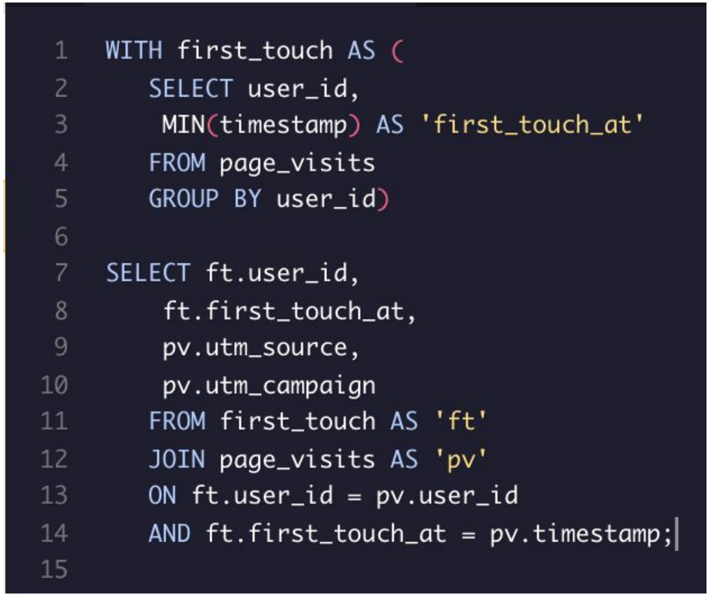
First_touch column first_touch_at equals to page_visits column timestamp, the earliest timestamp

SELECT columns user_id and first_touch_at from temporary table, first_touch, and columns utm_source and utm_campaign from page_visits table to display results that meet the JOIN condition.

Both tables will be renamed when being joined for clearer read:

First_touch is 'ft'

Page_visits is 'pv'



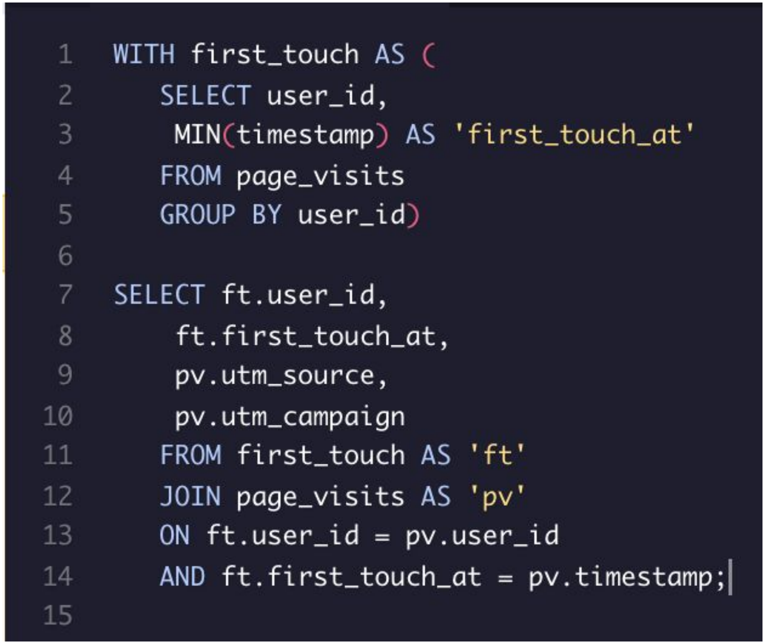
```sql
WITH first_touch AS (
   SELECT user_id,
    MIN(timestamp) AS 'first_touch_at'
   FROM page_visits
   GROUP BY user_id)

SELECT ft.user_id,
    ft.first_touch_at,
    pv.utm_source,
    pv.utm_campaign
   FROM first_touch AS 'ft'
   JOIN page_visits AS 'pv'
   ON ft.user_id = pv.user_id
   AND ft.first_touch_at = pv.timestamp;
```

Count of first touches for each campaign

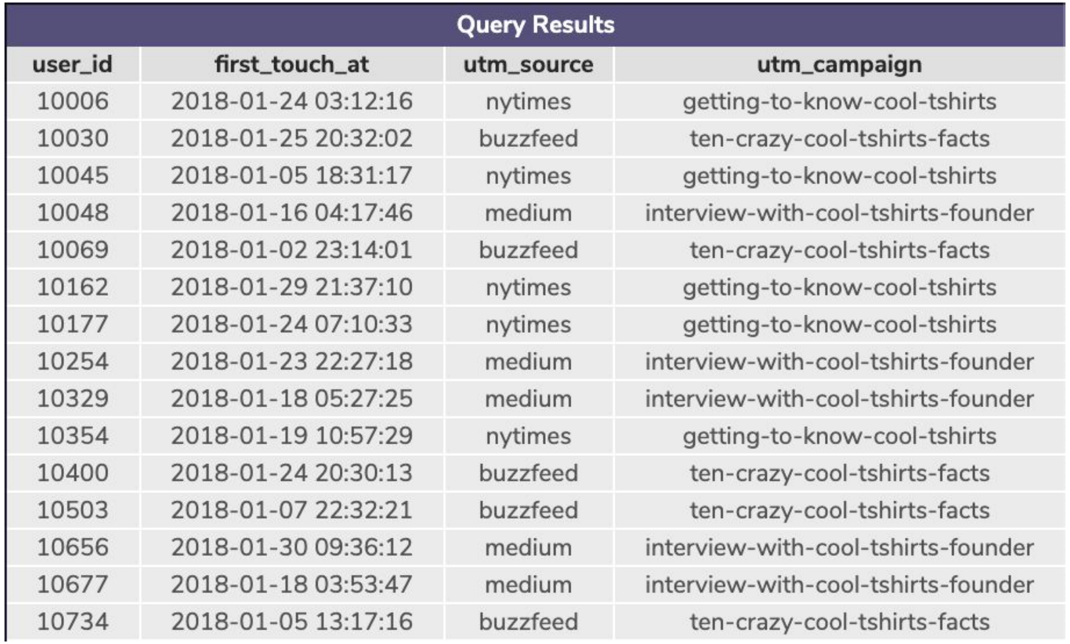
This query from the previous slide will show the all the users first touch timestamp as well as the source and campaign that caused a first visit to CoolTShirt website.

The results for this query, though, contain too many rows, and would take some time to count how many each campaign has.



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

SELECT ft.user_id,
    ft.first_touch_at,
    pv.utm_source,
    pv.utm_campaign
   FROM first_touch AS 'ft'
   JOIN page_visits AS 'pv'
   ON ft.user_id = pv.user_id
   AND ft.first_touch_at = pv.timestamp;
```



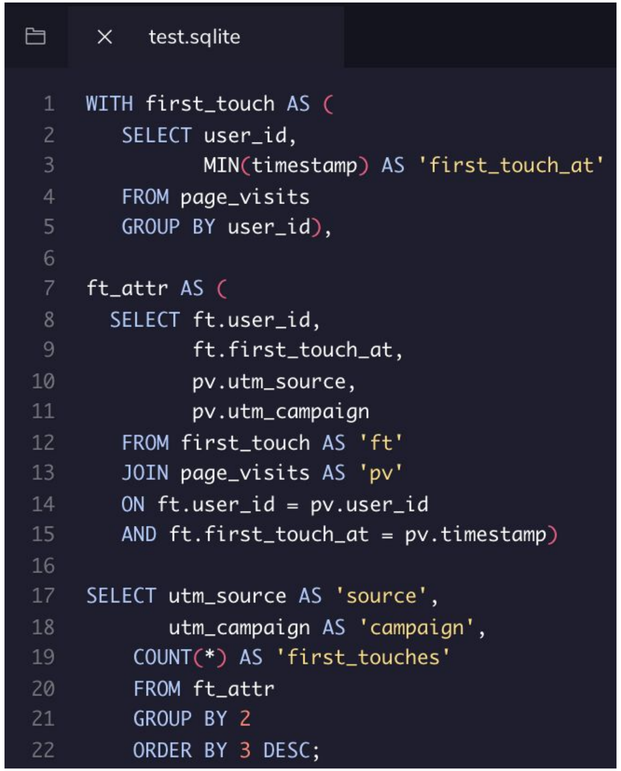
Query Results

user_id	first_touch_at	utm_source	utm_campaign
10006	2018-01-24 03:12:16	nytimes	getting-to-know-cool-tshirts
10030	2018-01-25 20:32:02	buzzfeed	ten-crazy-cool-tshirts-facts
10045	2018-01-05 18:31:17	nytimes	getting-to-know-cool-tshirts
10048	2018-01-16 04:17:46	medium	interview-with-cool-tshirts-founder
10069	2018-01-02 23:14:01	buzzfeed	ten-crazy-cool-tshirts-facts
10162	2018-01-29 21:37:10	nytimes	getting-to-know-cool-tshirts
10177	2018-01-24 07:10:33	nytimes	getting-to-know-cool-tshirts
10254	2018-01-23 22:27:18	medium	interview-with-cool-tshirts-founder
10329	2018-01-18 05:27:25	medium	interview-with-cool-tshirts-founder
10354	2018-01-19 10:57:29	nytimes	getting-to-know-cool-tshirts
10400	2018-01-24 20:30:13	buzzfeed	ten-crazy-cool-tshirts-facts
10503	2018-01-07 22:32:21	buzzfeed	ten-crazy-cool-tshirts-facts
10656	2018-01-30 09:36:12	medium	interview-with-cool-tshirts-founder
10677	2018-01-18 03:53:47	medium	interview-with-cool-tshirts-founder
10734	2018-01-05 13:17:16	buzzfeed	ten-crazy-cool-tshirts-facts

Count of first touches for each campaign

-To count how many first touches each campaign is responsible for a temporary table out of the previous query is created, which contains each users first touch and the sites and campaigns that led them. To do this, we will use an AS command which will have parentheses.
-We insert the query from the previous slide into the parentheses and put a comma after the first temporary table, first_touch, since we are working with 2 tables instead of 1, and name it ft_attr.
All info that was produced in the previous query is now temporary table ft_attr.

-Finally, SELECT the utm_source column, because of its relation to campaign, renamed as ‘source’ and utm.campaign renamed as ‘campaign’ from ft_attr table, and a column that aggregates the number all first touches for each ‘source’ and ‘campaign’ renamed ‘users’.
-We finish by grouping column 2, campaign, and order column 3, ‘users’, DESCing to show us the campaigns responsible for most first touches and what source those campaigns were on.

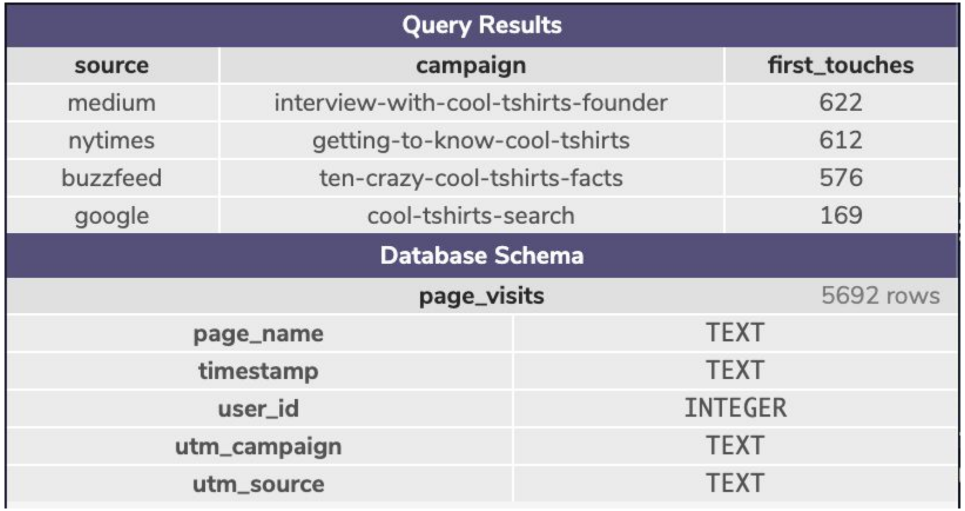


test.sqlite

```
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 AS 'ft'
   JOIN page_visits AS 'pv'
   ON ft.user_id = pv.user_id
   AND ft.first_touch_at = pv.timestamp)

SELECT utm_source AS 'source',
       utm_campaign AS 'campaign',
     COUNT(*) AS 'first_touches'
     FROM ft_attr
     GROUP BY 2
     ORDER BY 3 DESC;
```



Query Results

source	campaign	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

Database Schema

page_visits 5692 rows

page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT

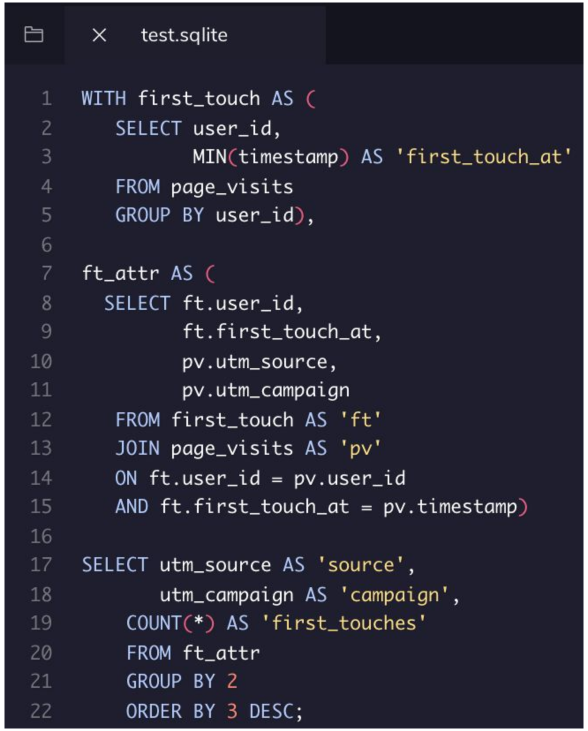
Count of last touches for each campaign

this part will be simple because the code to use will be similar to the code used to find first touch users for each campaign, with a few adjustments made.

Change aggregated column on the first temporary table to ‘MAX(timestamp)’ renamed as ‘last_touch_at’ so it aggregates each users last time on the site and change the name of this temporary table to ‘last_touch’.

Because the table name changes to ‘last_touch’, so will all the other references to this table.

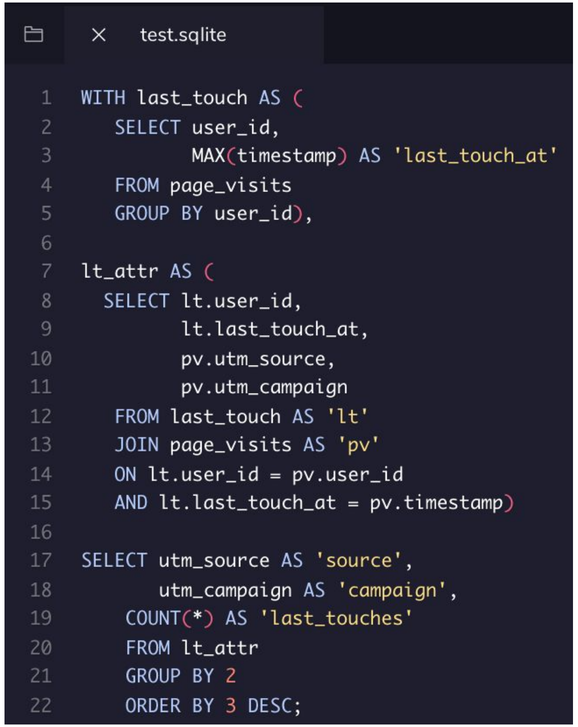
All ‘ft’ combinations will now be ‘lt’ and anything listed as ‘first_touch’ will now be ‘last_touch’.



```sql
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 AS 'ft'
   JOIN page_visits AS 'pv'
   ON ft.user_id = pv.user_id
   AND ft.first_touch_at = pv.timestamp)

SELECT utm_source AS 'source',
       utm_campaign AS 'campaign',
    COUNT(*) AS 'first_touches'
    FROM ft_attr
    GROUP BY 2
    ORDER BY 3 DESC;
```



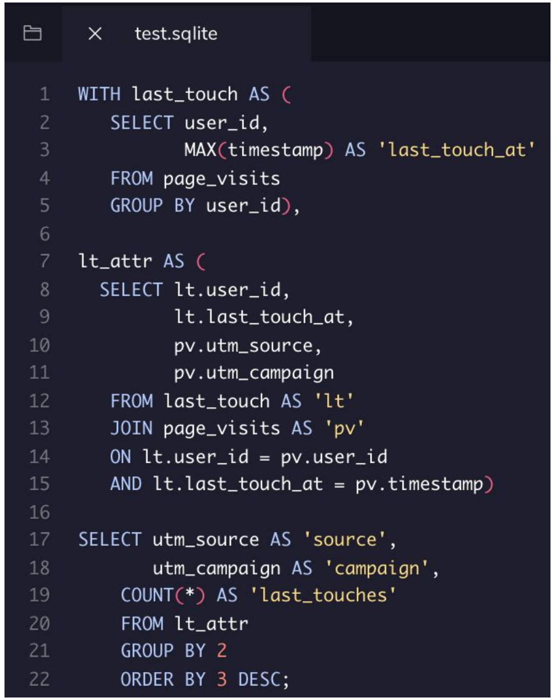
```sql
WITH last_touch AS (
   SELECT user_id,
          MAX(timestamp) AS 'last_touch_at'
   FROM page_visits
   GROUP BY user_id),

lt_attr AS (
  SELECT lt.user_id,
         lt.last_touch_at,
         pv.utm_source,
         pv.utm_campaign
   FROM last_touch AS 'lt'
   JOIN page_visits AS 'pv'
   ON lt.user_id = pv.user_id
   AND lt.last_touch_at = pv.timestamp)

SELECT utm_source AS 'source',
       utm_campaign AS 'campaign',
    COUNT(*) AS 'last_touches'
    FROM lt_attr
    GROUP BY 2
    ORDER BY 3 DESC;
```

Count of last touches for each campaign

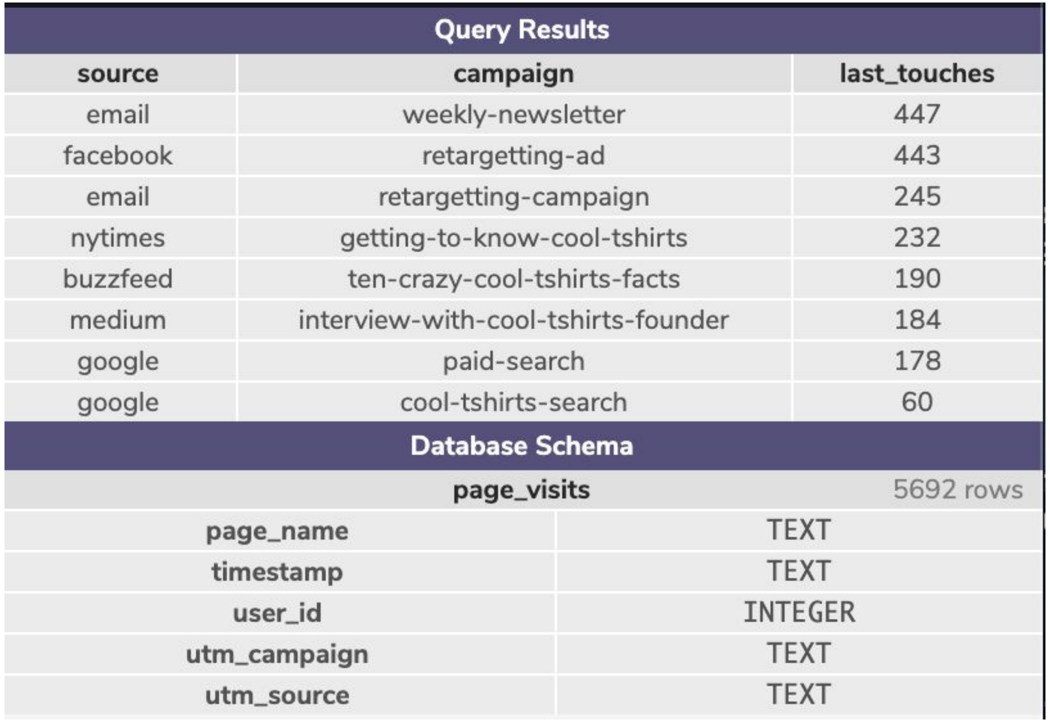
Once we have made the changes explained in the previous slide, we can see that it returns how many last touches each campaign is responsible for bringing back to the site and the sources the campaigns where on.


test.sqlite

```
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 AS 'lt'
   JOIN page_visits AS 'pv'
   ON lt.user_id = pv.user_id
   AND lt.last_touch_at = pv.timestamp)

SELECT utm_source AS 'source',
       utm_campaign AS 'campaign',
    COUNT(*) AS 'last_touches'
    FROM lt_attr
    GROUP BY 2
    ORDER BY 3 DESC;
```


Query Results

source	campaign	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

Database Schema

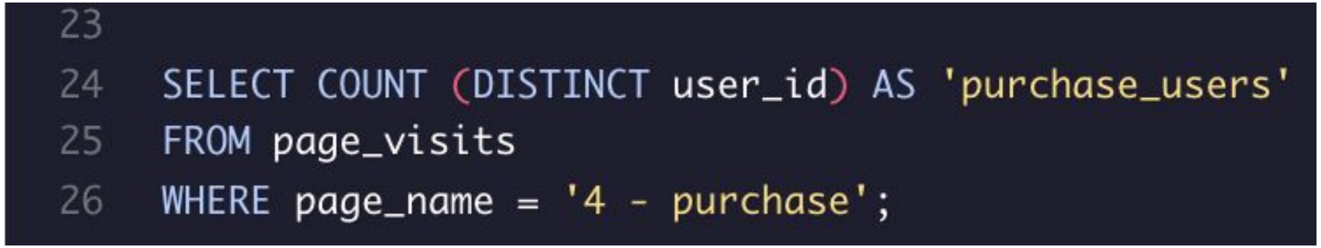
page_visits 5692 rows

page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT

Finding all website visitors who make a purchase

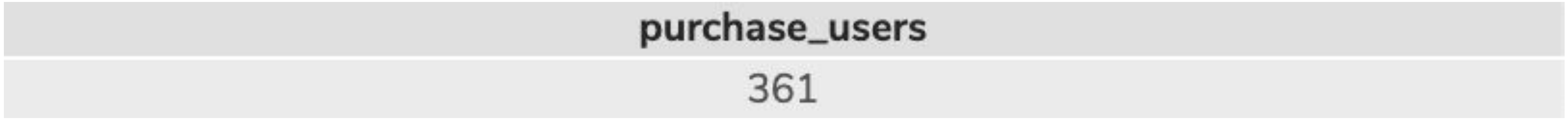
Using this query we can find that 361 users made a purchase on the site.

We SELECT the distinct users who have a value of ‘4 - purchase’ in the page_name column, and COUNT them.
‘4 - purchase’ is the last page on the site where a user makes a purchase.



```sql

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

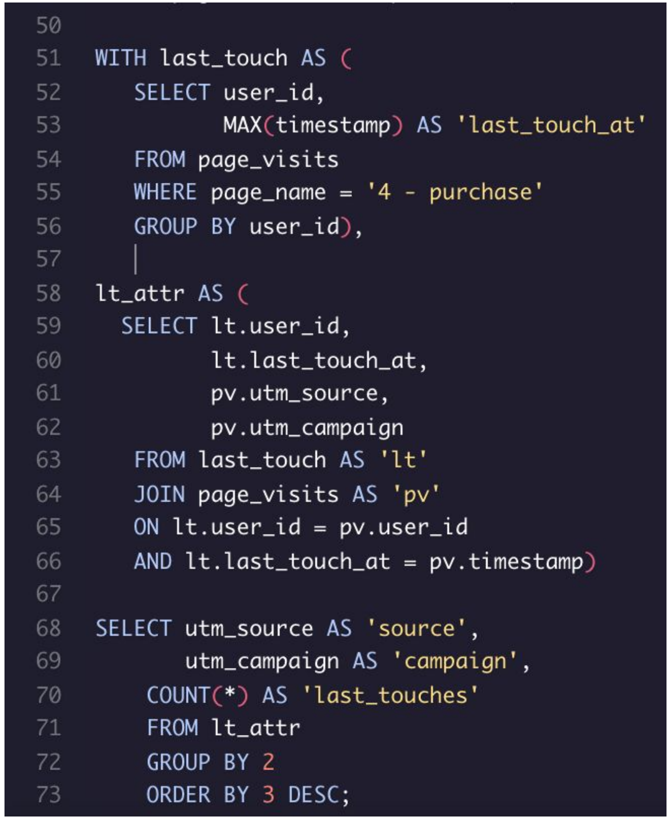


purchase_users
361

Count of last touches *on the purchase page* for each campaign

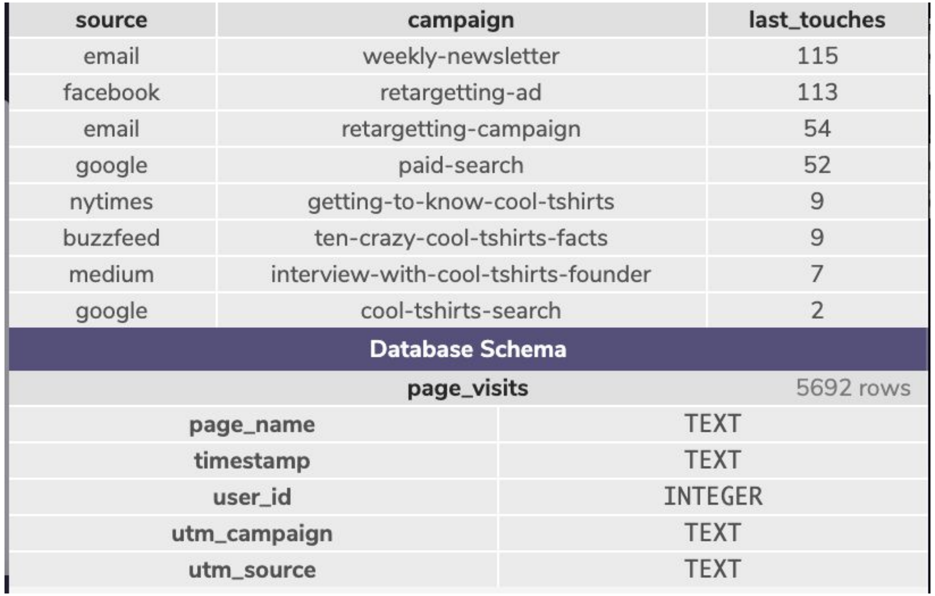
To find the amount of last touches on the purchase page of the site, ‘4 - purchase’, we make an edit to the ‘last_touch’ table by adding a WHERE clause for the column ‘page_name’.
This will obtain users who have the latest time value and who have the value of ‘4 - purchase’ for the ‘page_name’ column.

our full query will now return which campaigns caused the most last touches on the purchase page, and not all last touches made in all pages



```
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 AS 'lt'
   JOIN page_visits AS 'pv'
   ON lt.user_id = pv.user_id
   AND lt.last_touch_at = pv.timestamp)

SELECT utm_source AS 'source',
       utm_campaign AS 'campaign',
   COUNT(*) AS 'last_touches'
   FROM lt_attr
   GROUP BY 2
   ORDER BY 3 DESC;
```



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

Database Schema

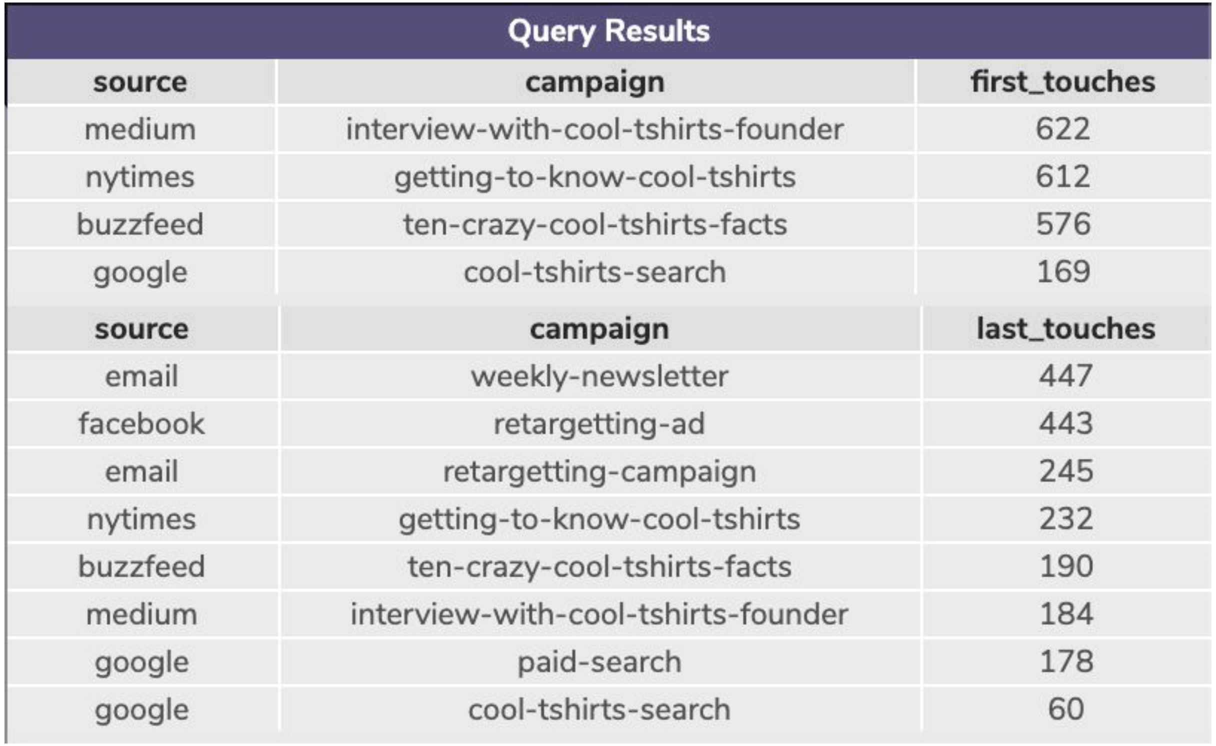
page_visits 5692 rows

page_name	TEXT
timestamp	TEXT
user_id	INTEGER
utm_campaign	TEXT
utm_source	TEXT

Analyzing users journey through CoolTShirts website

Based on the data that is given by both first touches for each campaign table and last touches for each campaign table, we can see the users first visit the page through more personal type campaigns such as an interview with the founder and a history of cooltshirts .

Their last visit to the page was impacted more through campaigns that focused on the specific user, such as a weekly newsletter via personal email, and retargeting methods using personal emails and ad's on the users facebook profile.

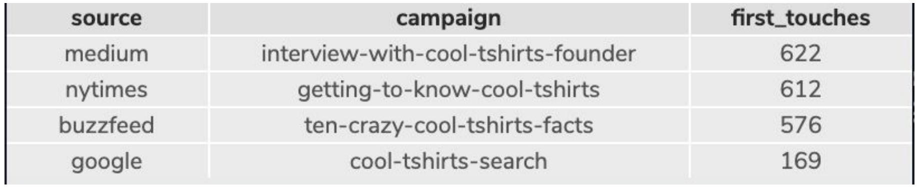


Query Results

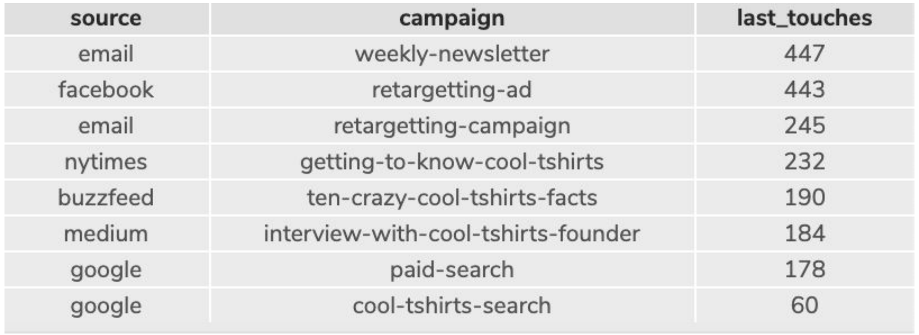
source	campaign	first_touches
medium	interview-with-cool-tshirts-founder	622
nytimes	getting-to-know-cool-tshirts	612
buzzfeed	ten-crazy-cool-tshirts-facts	576
google	cool-tshirts-search	169

source	campaign	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

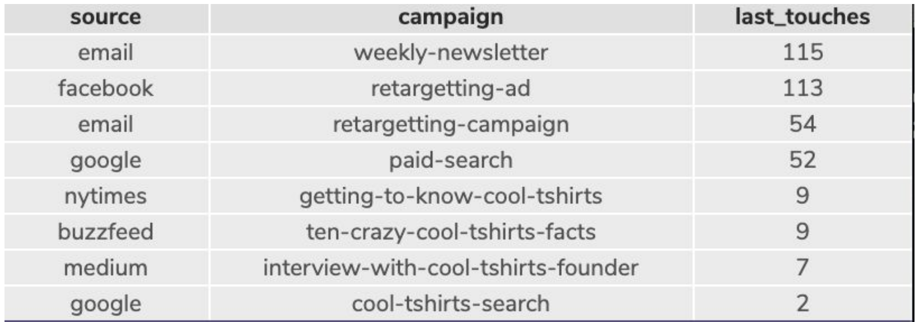
Top 5 campaigns that deserve re-investment



source	campaign	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



source	campaign	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



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

After reviewing the results for each campaign and the impact they had on the users first touches and last touches, i would recommend you invest in:

-interview-with-cool-tshirts-founder
-getting-to-know-cooltshirts
-ten-crazy-cool-tshirts-facts
-weekly-newsletter
-retargetting-ad

The first three listed show a large number of new visitors coming to the site for each campaign.

The last two are the campaigns that genereted the most last touches from the site visitors.

When using the results that show how many last touches on the purchase page each campaign is responsible for, it better shows why to invest in ten-crazy-cool-tshirts over retargeting campaign. The facts campaign generated more visitors than the retargeting-campaign has recorded on the purchase page, and all last touches made.