Claire Chour
May 11, 2020

Happy Money BI Intern Data Challenge

Part 1: Marketing Campaign Analysis

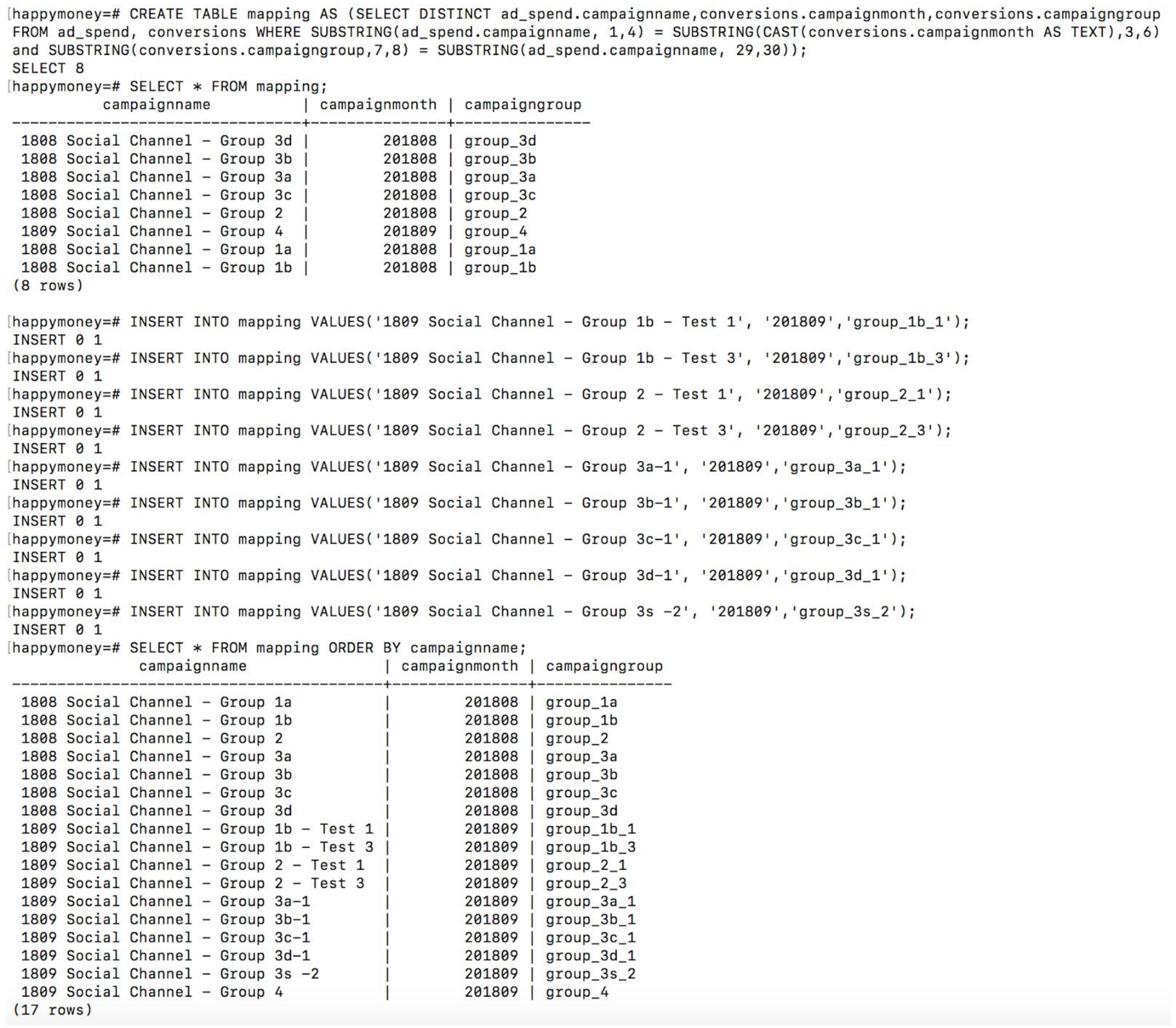
Q1: *To link two tables together, we need a mapping table.*
(1) Use your judgment based on the similarity of descriptions in campaign_name in SocialChannelAdSpend and <campaign_month, campaign_group> in SocialChannelConversion to come up with a mapping table. Your mapping table will have 3 columns, < campaign_name, campaign_month, campaign_group >.
(2) Write the mapping table to the database.

Since the format for campaign_name in SocialChannelAdSpend and campaign_group in SocialChannelConversion is different, I assumed that Group 1b - Test 1 in campaign_name was the same group as group_1b_1 in campaign_group, Group 3a-1 was the same as group_3a_1, etc. I did not want to change the CSV file in case this information is not correct, so I used the function INSERT INTO to add these rows into the mapping table. I also added photos of my code and its output for your reference. I did not use any underscores in my database when naming each column, but I will add them into the query below so it is easier to read.

SQL Queries:

```
CREATE TABLE mapping AS (
SELECT DISTINCT ad_spend.campaign_name, conversions.campaign_month,
conversions.campaign_group FROM ad_spend, conversions
WHERE SUBSTRING(ad_spend.campaign_name, 1,4) =
SUBSTRING(CAST(conversions.campaign_month AS TEXT), 3, 6) AND
SUBSTRING(conversions.campaign_group, 7, 8) = SUBSTRING(ad_spend.campaign_name, 29,
30)
);
INSERT INTO mapping VALUES('1809 Social Channel - Group 1b - Test 1', '201809',
'group_1b_1');
INSERT INTO mapping VALUES('1809 Social Channel - Group 1b - Test 3', '201809',
'group_1b_3');
INSERT INTO mapping VALUES('1809 Social Channel - Group 2 - Test 1', '201809','group_2_1');
INSERT INTO mapping VALUES('1809 Social Channel - Group 2 - Test 3', '201809','group_2_3');
INSERT INTO mapping VALUES('1809 Social Channel - Group 3a-1', '201809','group_3a_1');
INSERT INTO mapping VALUES('1809 Social Channel - Group 3b-1', '201809','group_3b_1');
```

INSERT INTO mapping VALUES('1809 Social Channel - Group 3c-1', '201809','group_3c_1');
INSERT INTO mapping VALUES('1809 Social Channel - Group 3d-1', '201809','group_3d_1');
INSERT INTO mapping VALUES('1809 Social Channel - Group 3s -2', '201809','group_3s_2');



```
[happymoney=# CREATE TABLE mapping AS (SELECT DISTINCT ad_spend.campaignname,conversions.campaignmonth,conversions.campaigngroup
 FROM ad_spend, conversions WHERE SUBSTRING(ad_spend.campaignname, 1,4) = SUBSTRING(CAST(conversions.campaignmonth AS TEXT),3,6)
 and SUBSTRING(conversions.campaigngroup,7,8) = SUBSTRING(ad_spend.campaignname, 29,30));
 SELECT 8
[happymoney=# SELECT * FROM mapping;
          campaignname          | campaignmonth | campaigngroup
--------------------------------+---------------+---------------
 1808 Social Channel - Group 3d |        201808 | group_3d
 1808 Social Channel - Group 3b |        201808 | group_3b
 1808 Social Channel - Group 3a |        201808 | group_3a
 1808 Social Channel - Group 3c |        201808 | group_3c
 1808 Social Channel - Group 2  |        201808 | group_2
 1809 Social Channel - Group 4  |        201809 | group_4
 1808 Social Channel - Group 1a |        201808 | group_1a
 1808 Social Channel - Group 1b |        201808 | group_1b
(8 rows)

[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 1b - Test 1', '201809','group_1b_1');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 1b - Test 3', '201809','group_1b_3');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 2 - Test 1', '201809','group_2_1');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 2 - Test 3', '201809','group_2_3');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 3a-1', '201809','group_3a_1');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 3b-1', '201809','group_3b_1');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 3c-1', '201809','group_3c_1');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 3d-1', '201809','group_3d_1');
 INSERT 0 1
[happymoney=# INSERT INTO mapping VALUES('1809 Social Channel - Group 3s -2', '201809','group_3s_2');
 INSERT 0 1
[happymoney=# SELECT * FROM mapping ORDER BY campaignname;
              campaignname               | campaignmonth | campaigngroup
-----------------------------------------+---------------+---------------
 1808 Social Channel - Group 1a          |        201808 | group_1a
 1808 Social Channel - Group 1b          |        201808 | group_1b
 1808 Social Channel - Group 2           |        201808 | group_2
 1808 Social Channel - Group 3a          |        201808 | group_3a
 1808 Social Channel - Group 3b          |        201808 | group_3b
 1808 Social Channel - Group 3c          |        201808 | group_3c
 1808 Social Channel - Group 3d          |        201808 | group_3d
 1809 Social Channel - Group 1b - Test 1 |        201809 | group_1b_1
 1809 Social Channel - Group 1b - Test 3 |        201809 | group_1b_3
 1809 Social Channel - Group 2 - Test 1  |        201809 | group_2_1
 1809 Social Channel - Group 2 - Test 3  |        201809 | group_2_3
 1809 Social Channel - Group 3a-1        |        201809 | group_3a_1
 1809 Social Channel - Group 3b-1        |        201809 | group_3b_1
 1809 Social Channel - Group 3c-1        |        201809 | group_3c_1
 1809 Social Channel - Group 3d-1        |        201809 | group_3d_1
 1809 Social Channel - Group 3s -2       |        201809 | group_3s_2
 1809 Social Channel - Group 4           |        201809 | group_4
(17 rows)
```

Q2: *Suppose this is an ongoing process where new marketing campaigns are added every day. What would your method be to keep the mapping table updated with new marketing campaigns? Should we allow the marketing team to own it and edit it themselves?*

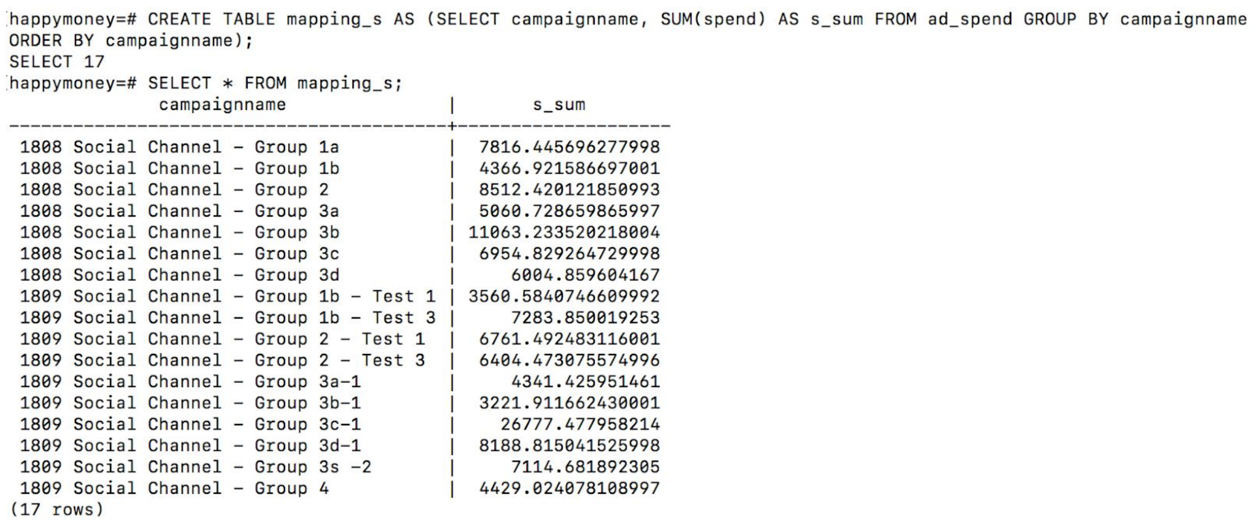
It is definitely realistic that there will be new marketing campaigns added on a daily basis. To keep the mapping table updated, I would continue using the INSERT INTO function considering we do not change the format for campaign_name and campaign_group. I should not have to change the query too much if not at all unless something is altered in the way the Excel sheet is organized.
The marketing team should definitely have access to the mapping table for reference, but I believe that they should not receive access to edit it because they may not be familiar with how the queries are written or with the language; there could be a risk of losing the mapping table. I personally like keeping my queries in a structured, organized format so that they are clear and easy to read.

Q3: *With the help of the mapping table you saved to the database, calculate cost per applied, cost per offered, cost per offer accepted, and cost per funded loan at the campaign level. Please write down the SQL queries you used. The goal here is to generate a summary table with campaign_name and all of our cpc metrics: cpa, cpo, cpoa, and cpfl.*

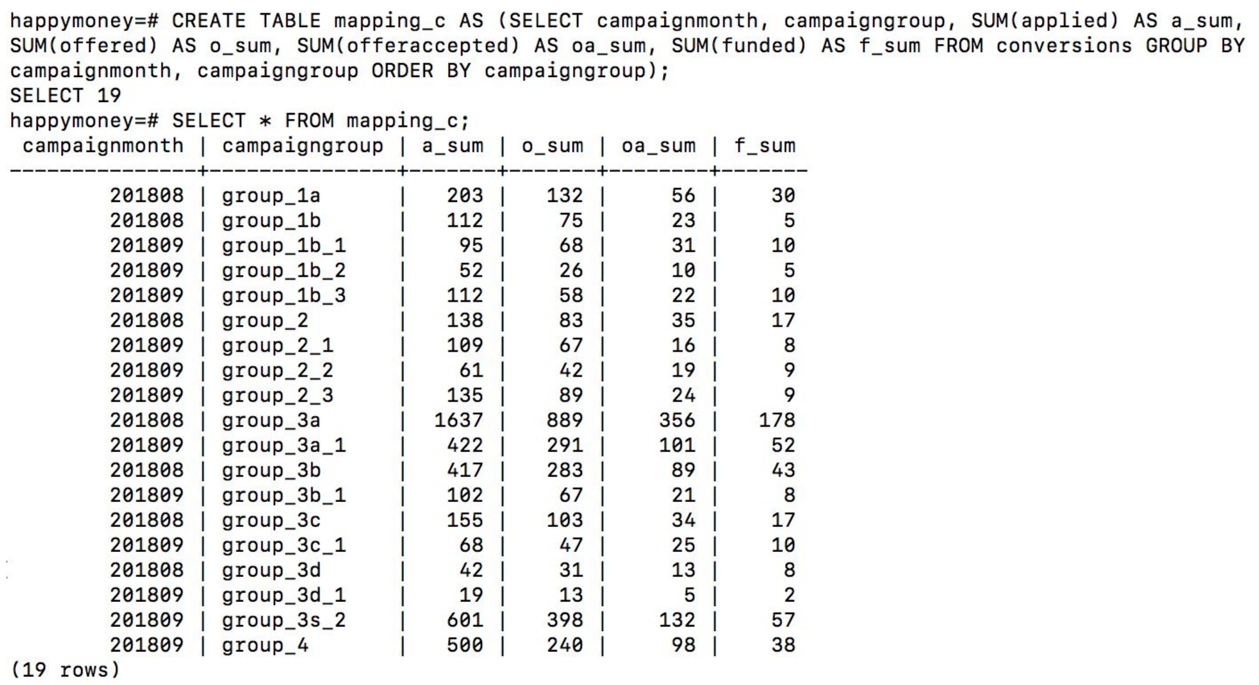
Like Q1, I also added photos of my database for your reference. I created two tables (one each for SocialChannelAdSpend and SocialChannelConversions) because I hope to join them in the end.

SQL Queries:

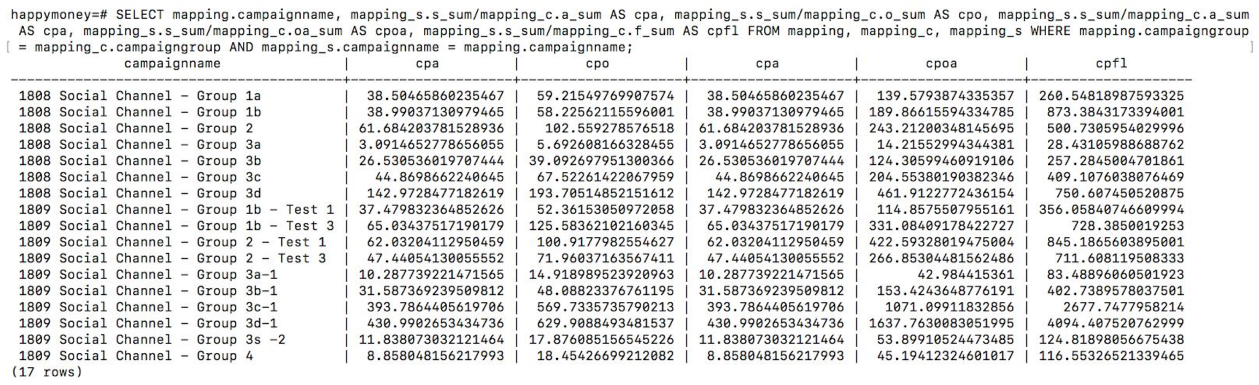
```
CREATE TABLE mapping_s AS (
SELECT campaign_name, SUM(spend) AS s_sum FROM ad_spend GROUP BY campaign_name
ORDER BY campaign_name
);
CREATE TABLE mapping_c AS (
SELECT campaign_month, campaign_group, SUM(applied) AS a_sum, SUM(offered) AS o_sum,
SUM(offer_accepted) AS oa_sum, SUM(funded) AS f_sum FROM conversions GROUP BY
campaign_month, campaign_group ORDER BY campaign_group
);
SELECT mapping.campaign_name, mapping_s.s_sum/mapping_c.a_sum AS cpa,
mapping_s.s_sum/mapping_c.o_sum AS cpo, mapping_s.s_sum/mapping_c.a_sum AS cpa,
mapping_s.s_sum/mapping_c.oa_sum AS cpoa, mapping_s.s_sum/mapping_c.f_sum AS cpfl
FROM mapping, mapping_c, mapping_s WHERE mapping.campaign_group =
mapping_c.campaign_group AND mapping_s.campaign_name = mapping.campaign_name;
```



```
happymoney=# CREATE TABLE mapping_s AS (SELECT campaignname, SUM(spend) AS s_sum FROM ad_spend GROUP BY campaignname
ORDER BY campaignname);
SELECT 17
happymoney=# SELECT * FROM mapping_s;
              campaignname              |       s_sum
----------------------------------------+--------------------
 1808 Social Channel - Group 1a         |  7816.445696277998
 1808 Social Channel - Group 1b         |  4366.921586697001
 1808 Social Channel - Group 2          |  8512.420121850993
 1808 Social Channel - Group 3a         |  5060.728659865997
 1808 Social Channel - Group 3b         | 11063.233520218004
 1808 Social Channel - Group 3c         |  6954.829264729998
 1808 Social Channel - Group 3d         |     6004.859604167
 1809 Social Channel - Group 1b - Test 1 | 3560.5840746609992
 1809 Social Channel - Group 1b - Test 3 |     7283.850019253
 1809 Social Channel - Group 2 - Test 1  |  6761.492483116001
 1809 Social Channel - Group 2 - Test 3  |  6404.473075574996
 1809 Social Channel - Group 3a-1       |     4341.425951461
 1809 Social Channel - Group 3b-1       |  3221.911662430001
 1809 Social Channel - Group 3c-1       |    26777.477958214
 1809 Social Channel - Group 3d-1       |  8188.815041525998
 1809 Social Channel - Group 3s -2      |     7114.681892305
 1809 Social Channel - Group 4          |  4429.024078108997
(17 rows)
```



```
happymoney=# CREATE TABLE mapping_c AS (SELECT campaignmonth, campaigngroup, SUM(applied) AS a_sum,
SUM(offered) AS o_sum, SUM(offeraccepted) AS oa_sum, SUM(funded) AS f_sum FROM conversions GROUP BY
campaignmonth, campaigngroup ORDER BY campaigngroup);
SELECT 19
happymoney=# SELECT * FROM mapping_c;
 campaignmonth | campaigngroup | a_sum | o_sum | oa_sum | f_sum
---------------+---------------+-------+-------+--------+-------
        201808 | group_1a      |   203 |   132 |     56 |    30
        201808 | group_1b      |   112 |    75 |     23 |     5
        201809 | group_1b_1    |    95 |    68 |     31 |    10
        201809 | group_1b_2    |    52 |    26 |     10 |     5
        201809 | group_1b_3    |   112 |    58 |     22 |    10
        201808 | group_2       |   138 |    83 |     35 |    17
        201809 | group_2_1     |   109 |    67 |     16 |     8
        201809 | group_2_2     |    61 |    42 |     19 |     9
        201809 | group_2_3     |   135 |    89 |     24 |     9
        201808 | group_3a      |  1637 |   889 |    356 |   178
        201809 | group_3a_1    |   422 |   291 |    101 |    52
        201808 | group_3b      |   417 |   283 |     89 |    43
        201809 | group_3b_1    |   102 |    67 |     21 |     8
        201808 | group_3c      |   155 |   103 |     34 |    17
        201809 | group_3c_1    |    68 |    47 |     25 |    10
        201808 | group_3d      |    42 |    31 |     13 |     8
        201809 | group_3d_1    |    19 |    13 |      5 |     2
        201809 | group_3s_2    |   601 |   398 |    132 |    57
        201809 | group_4       |   500 |   240 |     98 |    38
(19 rows)
```



```
happymoney=# SELECT mapping.campaignname, mapping_s.s_sum/mapping_c.a_sum AS cpa, mapping_s.s_sum/mapping_c.o_sum AS cpo, mapping_s.s_sum/mapping_c.a_sum
 AS cpa, mapping_s.s_sum/mapping_c.oa_sum AS cpoa, mapping_s.s_sum/mapping_c.f_sum AS cpfl FROM mapping, mapping_c, mapping_s WHERE mapping.campaigngroup
[ = mapping_c.campaigngroup AND mapping_s.campaignname = mapping.campaignname;                                                                              ]
              campaignname              |        cpa         |        cpo         |        cpa         |        cpoa        |        cpfl
----------------------------------------+--------------------+--------------------+--------------------+--------------------+--------------------
 1808 Social Channel - Group 1a         |  38.50465860235467 |  59.21549769907574 |  38.50465860235467 |  139.5793874335357 | 260.54818987593325
 1808 Social Channel - Group 1b         |  38.99037130979465 |  58.22562115596001 |  38.99037130979465 | 189.86615594334785 |  873.3843173394001
 1808 Social Channel - Group 2          | 61.684203781528936 |   102.559278576518 | 61.684203781528936 | 243.21200348145695 |  500.7305954029996
 1808 Social Channel - Group 3a         | 3.0914652778656055 |   5.69260816632845 | 3.0914652778656055 |  14.21552994344381 |  28.43105988688762
 1808 Social Channel - Group 3b         | 26.530536019707444 | 39.092697951300366 | 26.530536019707444 | 124.30599460919106 |  257.2845004701861
 1808 Social Channel - Group 3c         |   44.8698662240645 |  67.52261422067959 |   44.8698662240645 | 204.55380190382346 |  409.1076038076469
 1808 Social Channel - Group 3d         |  142.9728477182619 | 193.70514852151612 |  142.9728477182619 |  461.9122772436154 |    750.60745052087 5
 1809 Social Channel - Group 1b - Test 1 | 37.479832364852626 |  52.36153050972058 | 37.479832364852626 |  114.8575507955161 | 356.05840746609994
 1809 Social Channel - Group 1b - Test 3 |  65.03437517190179 | 125.58362102160345 |  65.03437517190179 | 331.08409178422727 |     728.3850019253
 1809 Social Channel - Group 2 - Test 1  |  62.03204112950459 |  100.9177982554627 |  62.03204112950459 | 422.59328019475004 |  845.1865603895001
 1809 Social Channel - Group 2 - Test 3  |  47.44054130055552 |  71.96037163567411 |  47.44054130055552 | 266.85304481562486 |    711.60811950833
 1809 Social Channel - Group 3a-1       | 10.287739221471565 | 14.918989523920963 | 10.287739221471565 |       42.984415361 |  83.48896060501923
 1809 Social Channel - Group 3b-1       | 31.587369239509812 |  48.08823376761195 | 31.587369239509812 |  153.4243648776191 |  402.7389578037501
 1809 Social Channel - Group 3c-1       |  393.7864405619706 |  569.7335735790213 |  393.7864405619706 |   1071.09911832856 |    2677.7477958214
 1809 Social Channel - Group 3d-1       |  430.9902653434736 |  629.9088493481537 |  430.9902653434736 | 1637.7630083051995 |  4094.407520762999
 1809 Social Channel - Group 3s -2      | 11.838073032121464 | 17.876085156545226 | 11.838073032121464 |  53.89910524473485 | 124.81898056675438
 1809 Social Channel - Group 4          |  8.858048156217993 |  18.45426699212082 |  8.858048156217993 |  45.19412324601017 | 116.55326521339465
(17 rows)
```

Q4: *Now that we have key CPC metrics like CPFL, we would like to use a BI platform and visualize data. At minimum, the dashboard we want consists of summary table of cpc metrics we just generated, including cpa, cpo, cpoa, and cpfl by Campaign. Extra point: plots of the daily cumulative spend by campaign (either faceted by campaign or sharing the same plot, where each campaign has a trace)*

Please create a prototype for your dashboard using one of the following methods:

1) *Tableau Dashboard*
2) *Use R/Shiny and something like https://rstudio.github.io/shinydashboard/*
3) ***Create elements (tables, graphs, etc) of your dashboard in R or Python, and paste everything together in a lucidchart or powerpoint slide, or whatever prototyping software you like to use***

See Jupyter Notebook and powerpoint slide for visualizations (in Python).

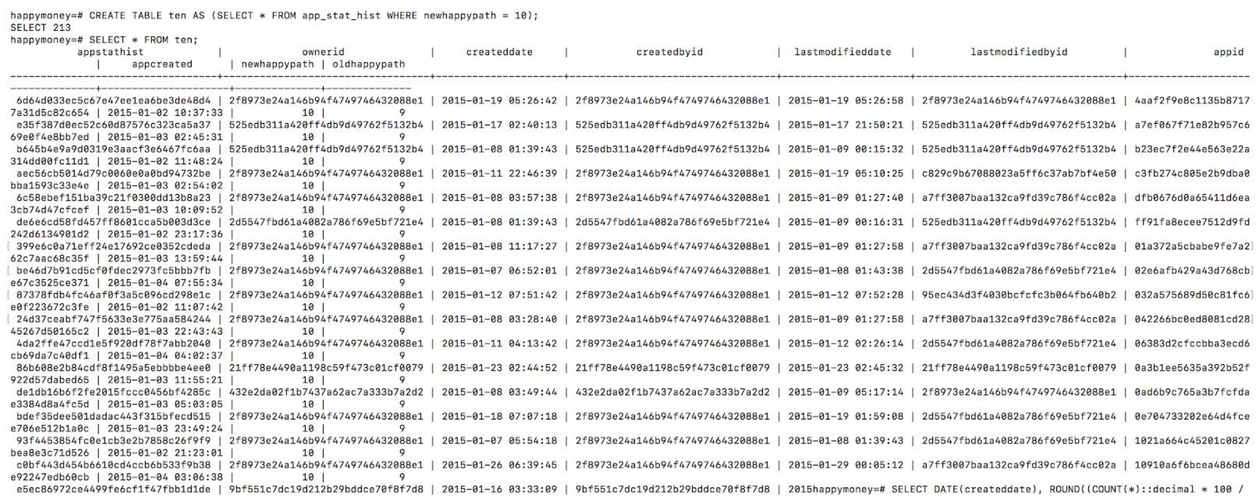
Part 2: Application Funnel Analysis

Q1. *Which application create date has the highest pull-through rate? Pull-through rate is defined as the number of people who reach the last status which is HappyPath 10 divided by the total number of applications. What is the overall conversion rate of each Happypath (rank from highest to lowest)? Please write SQL queries below.*

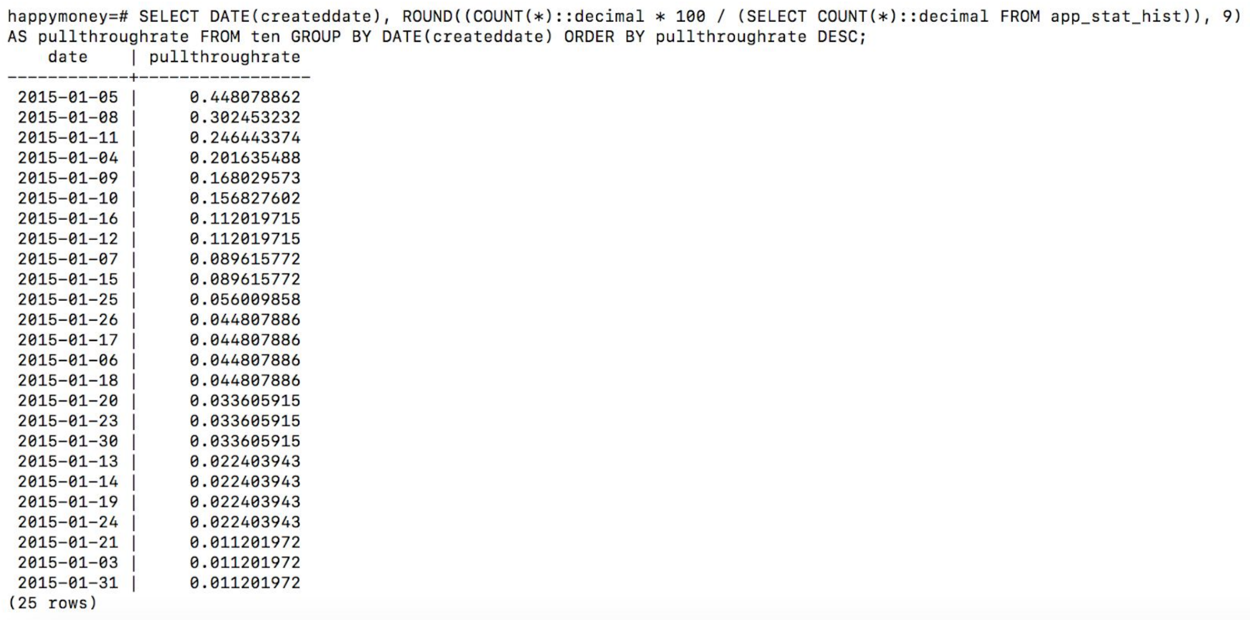
The highest pull-through rate was on 2015-01-05, and the pull-through rate was about 0.45. I ranked the rate of each Happypath of 10 from highest to lowest, and you can see the output below in the photos.

SQL Queries:

CREATE TABLE ten AS (
SELECT * FROM app_stat_hist WHERE newhappypath = 10
);
SELECT DATE(createddate), ROUND((COUNT(*)::decimal * 100 / (SELECT COUNT(*)::decimal FROM app_stat_hist)), 9) AS pullthroughrate FROM ten GROUP BY DATE(createddate) ORDER BY pullthroughrate DESC;



```
happymoney=# CREATE TABLE ten AS (SELECT * FROM app_stat_hist WHERE newhappypath = 10);
SELECT 213
happymoney=# SELECT * FROM ten;
           appstathist            |             ownerid              |     createddate     |            createdbyid           |   lastmodifieddate  |          lastmodifiedbyid        |            appid
             |      appcreated     | newhappypath | oldhappypath
----------------------------------+----------------------------------+---------------------+----------------------------------+---------------------+----------------------------------+--------------------
-------------+---------------------+--------------+--------------
 6d64d033ec5c67e47ee1ea6be3de48d4 | 2f8973e24a146b94f4749746432088e1 | 2015-01-19 05:26:42 | 2f8973e24a146b94f4749746432088e1 | 2015-01-19 05:26:58 | 2f8973e24a146b94f4749746432088e1 | 4aaf2f9e8c1135b8717
7a31d5c82c654 | 2015-01-02 10:37:33 |           10 |            9
 e35f387d0ec52c60d87576c323ca5a37 | 525edb311a420ff4db9d49762f5132b4 | 2015-01-17 02:40:13 | 525edb311a420ff4db9d49762f5132b4 | 2015-01-17 21:50:21 | 525edb311a420ff4db9d49762f5132b4 | a7ef067f71e82b957c6
69e0f4e8bb7ed | 2015-01-03 02:45:31 |           10 |            9
 b645b4e9a9d0319e3aacf3e6467fc6aa | 525edb311a420ff4db9d49762f5132b4 | 2015-01-08 01:39:43 | 525edb311a420ff4db9d49762f5132b4 | 2015-01-09 00:15:32 | 525edb311a420ff4db9d49762f5132b4 | b23ec7f2e44e563e22a
314dd00fc11d1 | 2015-01-02 11:48:24 |           10 |            9
 aec56cb5014d79c0060e0a0bd94732be | 2f8973e24a146b94f4749746432088e1 | 2015-01-11 22:46:39 | 2f8973e24a146b94f4749746432088e1 | 2015-01-19 05:10:25 | c829c9b67088023a5ff6c37ab7bf4e50 | c3fb274c805e2b9dba0
bba1593c33e4e | 2015-01-03 02:54:02 |           10 |            9
 6c58ebef151ba39c21f0300dd13b8a23 | 2f8973e24a146b94f4749746432088e1 | 2015-01-08 03:57:38 | 2f8973e24a146b94f4749746432088e1 | 2015-01-09 01:27:40 | a7ff3007baa132ca9fd39c786f4cc02a | dfb0676d0a65411d6ea
3cb74d47cfcef | 2015-01-03 10:09:52 |           10 |            9
 de6e6cd58fd457ff8601cca5b003d3ce | 2d5547fbd61a4082a786f69e5bf721e4 | 2015-01-08 01:39:43 | 2d5547fbd61a4082a786f69e5bf721e4 | 2015-01-09 00:16:31 | 525edb311a420ff4db9d49762f5132b4 | ff91fa8ecee7512d9fd
242d6134901d2 | 2015-01-02 23:17:36 |           10 |            9
 399e6c0a71eff24e17692ce0352cdeda | 2f8973e24a146b94f4749746432088e1 | 2015-01-08 11:17:27 | 2f8973e24a146b94f4749746432088e1 | 2015-01-09 01:27:58 | a7ff3007baa132ca9fd39c786f4cc02a | 01a372a5cbabe9fe7a2
62c7aac68c35f | 2015-01-03 13:59:44 |           10 |            9
 be46d7b91cd5cf0fdec2973fc5bbb7fb | 2f8973e24a146b94f4749746432088e1 | 2015-01-07 06:52:01 | 2f8973e24a146b94f4749746432088e1 | 2015-01-08 01:43:38 | 2d5547fbd61a4082a786f69e5bf721e4 | 02e6afb429a43d768cb
e67c3525ce371 | 2015-01-04 07:55:34 |           10 |            9
 87378fdb4fc46af0f3a5c096cd298e1c | 2f8973e24a146b94f4749746432088e1 | 2015-01-12 07:51:42 | 2f8973e24a146b94f4749746432088e1 | 2015-01-12 07:52:28 | 95ec434d3f4030bcfcfc3b064fb640b2 | 032a575689d50c81fc6
e0f223672c3fe | 2015-01-02 11:07:42 |           10 |            9
 24d37ceabf747f5633e3e775aa584244 | 2f8973e24a146b94f4749746432088e1 | 2015-01-08 03:28:40 | 2f8973e24a146b94f4749746432088e1 | 2015-01-09 01:27:58 | a7ff3007baa132ca9fd39c786f4cc02a | 042266bc0ed8081cd28
45267d50165c2 | 2015-01-03 22:43:43 |           10 |            9
 4da2ffe47ccd1e5f920df78f7abb2040 | 2f8973e24a146b94f4749746432088e1 | 2015-01-11 04:13:42 | 2f8973e24a146b94f4749746432088e1 | 2015-01-12 02:26:14 | 2d5547fbd61a4082a786f69e5bf721e4 | 06383d2cfccbba3ecd6
cb69da7c40df1 | 2015-01-04 04:02:37 |           10 |            9
 86b608e2b84cdf8f1495a5ebbbbe4ee0 | 21ff78e4490a1198c59f473c01cf0079 | 2015-01-23 02:44:52 | 21ff78e4490a1198c59f473c01cf0079 | 2015-01-23 02:45:32 | 21ff78e4490a1198c59f473c01cf0079 | 0a3b1ee5635a392b52f
922d57dabed65 | 2015-01-03 11:55:21 |           10 |            9
 de1db16b6f2fe2015fccc0456bf4285c | 432e2da02f1b7437a62ac7a333b7a2d2 | 2015-01-08 03:49:44 | 432e2da02f1b7437a62ac7a333b7a2d2 | 2015-01-09 05:17:14 | 2f8973e24a146b94f4749746432088e1 | 0ad6b9c765a3b7fcfda
e3384d8a4fc5d | 2015-01-03 05:03:05 |           10 |            9
 bdef35dee501dadac443f315bfecd515 | 2f8973e24a146b94f4749746432088e1 | 2015-01-18 07:07:18 | 2f8973e24a146b94f4749746432088e1 | 2015-01-19 01:59:08 | 2d5547fbd61a4082a786f69e5bf721e4 | 0e704733202e64d4fce
e706e512b1a0c | 2015-01-03 23:49:24 |           10 |            9
 93f4453854fc0e1cb3e2b7858c26f9f9 | 2f8973e24a146b94f4749746432088e1 | 2015-01-07 05:54:18 | 2f8973e24a146b94f4749746432088e1 | 2015-01-08 01:39:43 | 2d5547fbd61a4082a786f69e5bf721e4 | 1021a664c45201c0827
bea8e3c71d526 | 2015-01-02 21:23:01 |           10 |            9
 c0bf443d454b6610cd4ccb6b533f9b38 | 2f8973e24a146b94f4749746432088e1 | 2015-01-26 06:39:45 | 2f8973e24a146b94f4749746432088e1 | 2015-01-29 00:05:12 | a7ff3007baa132ca9fd39c786f4cc02a | 10910a6f6bcea48680d
e92247edb60cb | 2015-01-04 03:06:38 |           10 |            9
 e5ec86972ce4499fe6cf1f47fbb1d1de | 9bf551c7dc19d212b29bddce70f8f7d8 | 2015-01-16 03:33:09 | 9bf551c7dc19d212b29bddce70f8f7d8 | 2015happymoney=# SELECT DATE(createddate), ROUND((COUNT(*)::decimal * 100 /
```



```
happymoney=# SELECT DATE(createddate), ROUND((COUNT(*)::decimal * 100 / (SELECT COUNT(*)::decimal FROM app_stat_hist)), 9)
AS pullthroughrate FROM ten GROUP BY DATE(createddate) ORDER BY pullthroughrate DESC;
    date    | pullthroughrate
------------+-----------------
 2015-01-05 |     0.448078862
 2015-01-08 |     0.302453232
 2015-01-11 |     0.246443374
 2015-01-04 |     0.201635488
 2015-01-09 |     0.168029573
 2015-01-10 |     0.156827602
 2015-01-16 |     0.112019715
 2015-01-12 |     0.112019715
 2015-01-07 |     0.089615772
 2015-01-15 |     0.089615772
 2015-01-25 |     0.056009858
 2015-01-26 |     0.044807886
 2015-01-17 |     0.044807886
 2015-01-06 |     0.044807886
 2015-01-18 |     0.044807886
 2015-01-20 |     0.033605915
 2015-01-23 |     0.033605915
 2015-01-30 |     0.033605915
 2015-01-13 |     0.022403943
 2015-01-14 |     0.022403943
 2015-01-19 |     0.022403943
 2015-01-24 |     0.022403943
 2015-01-21 |     0.011201972
 2015-01-03 |     0.011201972
 2015-01-31 |     0.011201972
(25 rows)
```

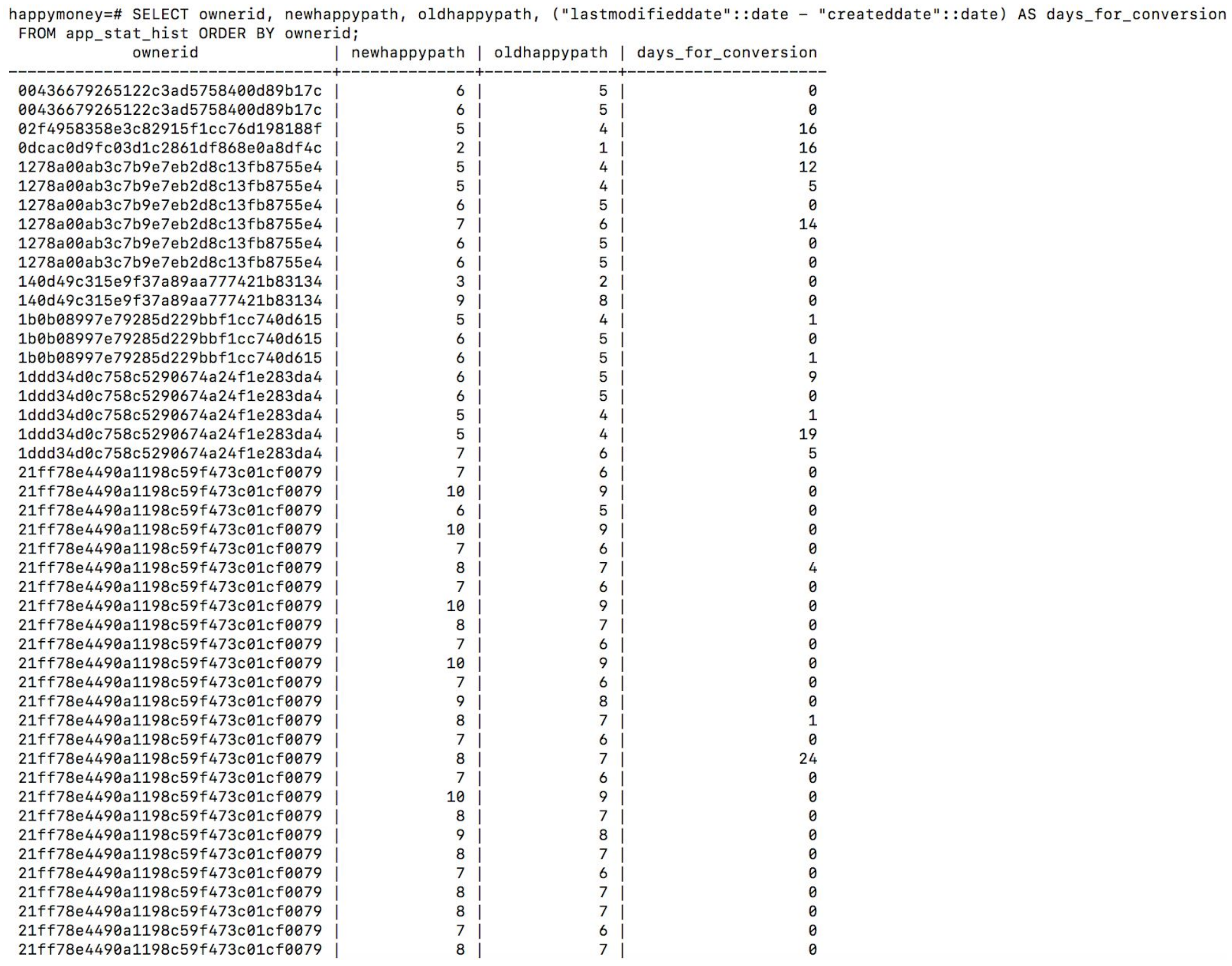
Q2. *How long does it take from one status to the next status? When the conversion gets mature? How to visualize it? Use SQL, R or Python to answer the questions. (hint: one of the approaches is the maturity curve. It calculates the cumulative percentage of applicants goes from one status to the next over time)*

Since created_date represents the start time of a status change and last_modified_date represents the end time of the status change, I am taking the difference of the days of the two dates to output how long it took for a user to upgrade from one status to the next. I ordered by the ownerid because I had the motive of looking at how each user was doing in terms of status changes, but it is also an option to order by the Happy Path and analyze if it usually takes longer for one to upgrade to the next status if one is at a higher status. Assuming that each status update has the same level of "difficulty" when upgrading to the next one, I took the average of the column days_for_conversion and it takes about four days for one to upgrade to one's next status. However, later on, I realize this is not the case (by the line graph I created). The conversion becomes mature when one gets closer to a Happy Path of 10.

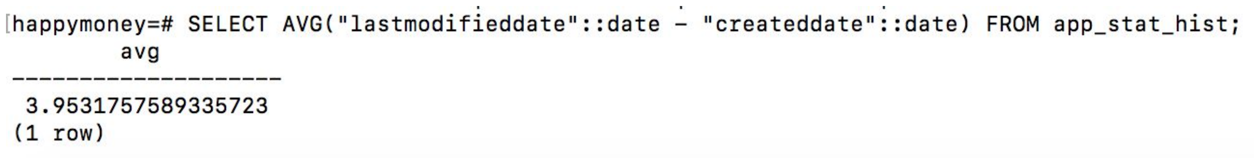
I would visualize the conversion via a line graph of the Happy Path as the x-axis and the days of conversion to the next status as the y-axis. I envisioned the line to appear as a maturity curve based on my observations of the data, but it is difficult to determine a set pattern for how long it takes for one to convert to the next status. There are definitely other factors that contribute to why a user may take longer to upgrade, but I would need more information. According to the graph, path 2 takes the longest; this is definitely because people have not yet adapted to Happy Money, and this "curve" goes down from there.

SQL Queries:

SELECT ownerid, newhappypath, oldhappypath, ("lastmodifieddate"::date - "createddate"::date) AS days_for_conversion FROM app_stat_hist ORDER BY ownerid;
SELECT AVG("lastmodifieddate"::date - "createddate"::date) FROM app_stat_hist;



```
happymoney=# SELECT ownerid, newhappypath, oldhappypath, ("lastmodifieddate"::date - "createddate"::date) AS days_for_conversion
 FROM app_stat_hist ORDER BY ownerid;
             ownerid              | newhappypath | oldhappypath | days_for_conversion
----------------------------------+--------------+--------------+---------------------
 00436679265122c3ad5758400d89b17c |            6 |            5 |                   0
 00436679265122c3ad5758400d89b17c |            6 |            5 |                   0
 02f4958358e3c82915f1cc76d198188f |            5 |            4 |                  16
 0dcac0d9fc03d1c2861df868e0a8df4c |            2 |            1 |                  16
 1278a00ab3c7b9e7eb2d8c13fb8755e4 |            5 |            4 |                  12
 1278a00ab3c7b9e7eb2d8c13fb8755e4 |            5 |            4 |                   5
 1278a00ab3c7b9e7eb2d8c13fb8755e4 |            6 |            5 |                   0
 1278a00ab3c7b9e7eb2d8c13fb8755e4 |            7 |            6 |                  14
 1278a00ab3c7b9e7eb2d8c13fb8755e4 |            6 |            5 |                   0
 1278a00ab3c7b9e7eb2d8c13fb8755e4 |            6 |            5 |                   0
 140d49c315e9f37a89aa777421b83134 |            3 |            2 |                   0
 140d49c315e9f37a89aa777421b83134 |            9 |            8 |                   0
 1b0b08997e79285d229bbf1cc740d615 |            5 |            4 |                   1
 1b0b08997e79285d229bbf1cc740d615 |            6 |            5 |                   0
 1b0b08997e79285d229bbf1cc740d615 |            6 |            5 |                   1
 1ddd34d0c758c5290674a24f1e283da4 |            6 |            5 |                   9
 1ddd34d0c758c5290674a24f1e283da4 |            6 |            5 |                   0
 1ddd34d0c758c5290674a24f1e283da4 |            5 |            4 |                   1
 1ddd34d0c758c5290674a24f1e283da4 |            5 |            4 |                  19
 1ddd34d0c758c5290674a24f1e283da4 |            7 |            6 |                   5
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |           10 |            9 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            6 |            5 |                   0
 21ff78e4490a1198c59f473c01cf0079 |           10 |            9 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   4
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |           10 |            9 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |           10 |            9 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            9 |            8 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   1
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                  24
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |           10 |            9 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            9 |            8 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            7 |            6 |                   0
 21ff78e4490a1198c59f473c01cf0079 |            8 |            7 |                   0
```



```
[happymoney=# SELECT AVG("lastmodifieddate"::date - "createddate"::date) FROM app_stat_hist;
        avg
--------------------
 3.9531757589335723
(1 row)
```

Q3. *Build a BI dashboard to visualize Q1 and Q2. Transform the current table to help generate plots if needed.*

Please create a prototype for your dashboard using one of the following methods:

1) Tableau Dashboard

2) Use R/Shiny and something like https://rstudio.github.io/shinydashboard/

3) ***Create elements (tables, graphs, etc) of your dashboard in R or Python, and paste everything together in a lucidchart or powerpoint slide, or whatever prototyping software you like to use***

See Jupyter Notebook and powerpoint slide for visualizations (in Python).