extended_case_3

May 1, 2020

1 How are different customer cohorts affecting our NPS ratings over time?

1.1 Introduction

Business Context. You are a data scientist for the same fast-growing startup in the previous case. You presented your analysis to the executive team of the startup, and now they want to dig deeper into various customer segments.
They want to zoom in on specific interesting customers, such as those who have left a wide range of scores over time.

Business Problem. Your task is to write and apply more advanced SQL queries to the same NPS dataset and cloud database that we set up in the previous case to answer the specific questions the client has, which are mentioned below.

Analytical Context. The startup has now given you more information about what the `is_spam` and `is_premier` columns mean in the `customer` table that we ignored before. They would like to know the answers to the following:

1. Do spam customers give lower scores?
2. Do premier customers give higher scores?
3. How does removing spam accounts affect our NPS ratings?
4. Do some customers leave a wide range of scores?
5. What is the growth rate in total customers by month and is this related to NPS?

1.2 Premier and spam customers

Take a look at the `customer` table again:

```
SELECT * FROM customer LIMIT 10;
```

```
 id | created_at | is_premier | is_spam
----+------------+------------+---------
  1 | 2018-01-01 | f          | f
  2 | 2018-01-01 | f          | f
  3 | 2018-01-01 | t          | f
  4 | 2018-01-01 | f          | f
  5 | 2018-01-01 | f          | f
  6 | 2018-01-01 | f          | f
```

```
  7 | 2018-01-01 | f          | f
  8 | 2018-01-01 | f          | f
  9 | 2018-01-01 | f          | f
 10 | 2018-01-01 | f          | f
```

You'll notice that there are two boolean columns, `is_premier` and `is_spam`. Premier customers pay a little bit extra for priority support and some extra features. The startup hypothesizes that these customers will give higher satisfaction scores.

The startup also suspects that a competitor is creating fake customer accounts to sign up and leave negative responses, but this has not been tested.

Use these columns to answer the following questions. For each question: (1) give the SQL query or queries that you needed to calculate the answer; (2) show example output from your queries (truncated if necessary, and with numbers rounded to 2 decimal points); and finally (3) provide a written response that directly answers the question.

1.2.1 Exercise 1:

The startup hypothesizes that spam accounts are giving artificially low scores. Is this true? How many spam accounts exist?

Answer. The average scores for all customers, non-spam customers, and spam customers is as follows:

```
SELECT AVG(score) from score INNER JOIN customer ON customer_id = customer.id
```

```
        avg
--------------------
 8.1928418119421037
(1 row)
```

```
SELECT AVG(score) from score INNER JOIN customer ON customer_id = customer.id
WHERE is_spam = False
```

```
        avg
--------------------
 8.2769821605708720
(1 row)
```

```
SELECT AVG(score) from score INNER JOIN customer ON customer_id = customer.id
WHERE is_spam = True
```

```
            avg
----------------------------
 0.000000000000000000000000
(1 row)
```

There are 188323 customers recorded in total. Of these, 1941 (around 1%) are flagged as spam. The average score left across all customers is 8.2, but this goes up to 8.3 when the spam scores are filtered out. Notably, the average of scores left by spam accounts is 0.0, indicating that spam accounts only ever leave 0 as a response.

1.2.2 Exercise 2:

The startup hypothesizes that premier customers give higher scores. Is this true? How many premier customers are there?

Answer. One possible solution is given below:

```
SELECT avg(score), count(score) FROM score INNER JOIN customer on customer_id = customer.id
WHERE is_premier = True;
```

```
        avg         | count
--------------------+--------
 9.2664041085100350 | 158695
```

```
SELECT avg(score), count(score) FROM score INNER JOIN customer on customer_id = customer.id
WHERE is_premier = False;
```

```
        avg         |  count
--------------------+---------
 8.0727692135292339 | 1418883
```

The average score left by premier customers is 9.3, while the average score left by non-premier customers is 8.0. On average, premier customers leave scores 1.3 points higher than non-premier customers. Around 10% of responses left are left by premier customers.

1.2.3 Exercise 3:

Since spam accounts are artificially lowering our scores, it is natural to determine how much they were affecting our weekly NPS ratings. Please do the following:

1. Modify the final query from Case 12.2 to remove spam accounts before calculating the NPS scores
2. Modify the query again to summarize the NPS results by month instead of by week

How much higher are scores in the first five weeks with spam accounts filtered out?

Answer. One possible solution is given below:

NPS with spam filtering:

```
SELECT *, ROUND(((CAST(promoter AS DECIMAL) / total) - (CAST(detractor AS DECIMAL) / total)) *
(SELECT week,
SUM(CASE WHEN nps_class = 'promoter' THEN 1 ELSE 0 END) AS "promoter",
SUM(CASE WHEN nps_class = 'passive' THEN 1 ELSE 0 END) AS "passive",
SUM(CASE WHEN nps_class = 'detractor' THEN 1 ELSE 0 END) AS "detractor",
    COUNT(*) AS "total" FROM
(SELECT CASE
    WHEN avg_week_score > 8 THEN 'promoter'
    WHEN avg_week_score > 6 THEN 'passive'
    ELSE 'detractor'
END AS nps_class, week FROM
(SELECT TO_CHAR(score.created_at, 'IYYY-IW') AS week, customer_id, AVG(score) as avg_week_score
```

```
INNER JOIN customer on customer_id = customer.id
WHERE is_spam = False
GROUP BY week, customer_id) a) b
GROUP BY week
ORDER BY week) c
limit 100;
```

```
  week   | promoter | passive | detractor | total | nps
---------+----------+---------+-----------+-------+-----
 2018-01 |       26 |       0 |        38 |    64 | -19
 2018-02 |       65 |       2 |        62 |   129 |   2
 2018-03 |       71 |       7 |        68 |   146 |   2
 2018-04 |       76 |       6 |        82 |   164 |  -4
 2018-05 |      186 |      23 |       132 |   341 |  16
 ...
```

NPS by month with spam filtering:

```
SELECT *, ROUND(((CAST(promoter AS DECIMAL) / total) - (CAST(detractor AS DECIMAL) / total)) *
(SELECT year_month,
SUM(CASE WHEN nps_class = 'promoter' THEN 1 ELSE 0 END) AS "promoter",
SUM(CASE WHEN nps_class = 'passive' THEN 1 ELSE 0 END) AS "passive",
SUM(CASE WHEN nps_class = 'detractor' THEN 1 ELSE 0 END) AS "detractor",
    COUNT(*) AS "total" FROM
(SELECT CASE
    WHEN avg_month_score > 8 THEN 'promoter'
    WHEN avg_month_score > 6 THEN 'passive'
    ELSE 'detractor'
END AS nps_class, year_month FROM
(SELECT TO_CHAR(score.created_at, 'YYYY-MM') AS year_month, customer_id, AVG(score) as avg_mon
INNER JOIN customer ON customer_id = customer.id
WHERE is_spam = False
GROUP BY year_month, customer_id) a) b
GROUP BY year_month
ORDER BY year_month) c
limit 100;
```

```
 year_month | promoter | passive | detractor | total  | nps
------------+----------+---------+-----------+--------+-----
 2018-01    |      138 |       8 |       143 |    289 |  -2
 2018-02    |      972 |     144 |       476 |   1592 |  31
 2018-03    |     3008 |     395 |       889 |   4292 |  49
 2018-04    |     5882 |     767 |      1487 |   8136 |  54
 2018-05    |     9583 |    1322 |      2355 |  13260 |  55
 2018-06    |    15879 |    1724 |      2990 |  20593 |  63
 2018-07    |    24038 |    2671 |      4508 |  31217 |  63
 2018-08    |    37301 |    3607 |      6173 |  47081 |  66
 2018-09    |    42627 |    5085 |     14332 |  62044 |  46
 2018-10    |    57964 |    6910 |     15589 |  80463 |  53
 2018-11    |    78351 |    8265 |     17539 | 104155 |  58
```

```
 2018-12    |   122236 |   11649 |     23114 | 156999 |  63
```

Filtering out the spam accounts does not have a large effect on the final NPS score. After spam filtering, it is one point higher in the first week (-19 vs. -20); the same in the second week (2); one point higher in the third week (2 vs. 1); the same in the fourth week (-4) and one point higher in the fifth week (16 vs. 15).

1.2.4 Exercise 4:

The marketing department is very interested in customers who leave a wide range of scores over time. Customers who started out leaving `0`s as responses and more recently left `10`s as responses are great customers to reach out to and ask for public testimonials on the product, as they are likely very happy with a specific new feature or a specific recent interaction to the point that it made them completely change their minds.

Similarly, customers who were initially happy (leaving `10`s as a response) and more recently started leaving `0`s, are interesting because the startup can attempt to "win them back". We know that they were very happy with us before, so if we find out what specifically went wrong, they may become promoters again.

Your job is to find out if these customers exist, and who they are. This will require a few steps:

1. For each customer, their score range (let's denote it `score_range` in code) is the difference between the highest and lowest scores they have ever left. For example, if one customer has left 5 responses, and the scores were 1, 4, 3, 6, and 2 respectively, then that customer's score range is $6 - 1 = 5$. Create a nested `SELECT` query to calculate how many premier, non-spam customers fall into each `score_range` bucket. For example, how many premier, non-spam customers have a score range of 10, 9, 8, etc.? Your output should have two columns: one for `score_range` and one showing how many customers have that `score_range`, with the most frequent `score_ranges` first. Note that Postgres has `min` and `max` functions built-in.
2. The marketing department would like to do a case study on the 10 premier customers with the largest range in response. In case of ties, they want to focus on customers who have left the most responses over time. Write a query which allows them to find these top 10 customers.
3. What are the customer IDs of the top 10 customers by score range and number of responses that the marketing department should contact for their case study? What are their score ranges and number of responses?

What are the most frequent score ranges across all premier, non-spam customers? Give a possible explanation as to why these ranges are more frequent.

Answer. One possible solution is given below:

Get the number of premier, non-spam customers with each score range:

```
SELECT score_range, count(score_range) FROM
(SELECT customer_id,
max(score) - min(score) as score_range
FROM score INNER JOIN customer on customer_id = customer.id
WHERE is_premier = True and is_spam = False
GROUP BY customer_id
) a
```

```
GROUP BY score_range
ORDER BY count desc
```

```
 score_range | count
-------------+-------
           0 | 14328
           9 |  1535
           1 |   796
           2 |   531
           3 |   395
           8 |   314
           4 |   259
           7 |   236
           6 |   169
           5 |   166
```

Get the top 10 premier, non-spam customers by `score_range`, with ties broken by number of responses:

```
SELECT customer_id, score_range, num_responses FROM
(SELECT customer_id,
min(score) as min_score,
max(score) as max_score,
max(score) - min(score) as score_range,
count(score) as num_responses
FROM score
INNER JOIN customer ON customer_id = customer.id
WHERE is_spam = False and is_premier = True
GROUP BY customer_id
) a
GROUP BY customer_id, score_range, num_responses
ORDER BY score_range desc, num_responses desc
LIMIT 10;
```

```
 customer_id | score_range | num_responses
-------------+-------------+---------------
        1253 |           9 |            37
       12343 |           9 |            32
        7495 |           9 |            31
        9516 |           9 |            31
        2475 |           9 |            30
        5222 |           9 |            30
        1774 |           9 |            30
        6203 |           9 |            29
        8163 |           9 |            29
        8418 |           9 |            29
```

There are no premier customers with a score range of 10, but 1535 premier, non-spam customers who have a score range of 9. The marketing department can contact the customers with IDs: (1253, 12343, 7495, 9516, 2475, 5222, 1774, 6203, 8163, 8418) as these have all left 29 or more responses

and have a score range of 9.

The most frequent score range for premier, non-spam customers is `0`. This is likely because many customers only leave one score, and that satisfaction is overall more dependent on the customer than on the product. However, the second most frequent score range is `9`, probably because surprising events (such as a bad experience with customer support, or a good experience in discovering a new feature) is likely to make customers very happy or very unhappy for a specific period of time. The third most common score range is `1`, again probably because some customers (excluding the ones who have surprising events occur) leave a score slightly above or below their "usual" score if they are in a better or worse mood that day.

1.2.5 Exercise 5:

The startup also hypothesizes that monthly growth rates in the customer base are lower when customers are less satisfied. For example, we already know that NPS dropped in September, but was this mirrored by a drop in growth rates? We'll do this in a few steps:

1. Write a query that calculates the growth rate each month. The growth rate is defined as the growth percentage from the previous month. For example, in January, 291 customers signed up. In February, 1383 customers signed up, so the growth rate for February was $(1383 - 291)/291 * 100 = 375\%$. (Hint: you can use the `lag()` function built into Postgres to use the previous row in a `GROUP BY` clause for a calculation in the current row.)
2. Write a query that shows three columns: `year_month`, `growth_percent`, and `nps` to see if these are related. (Hint: Instead of working out how to combine the two different and complicated queries we have built for NPS and Growth calculation, you can use the `CREATE VIEW` functionality of Postgres to turn each query into its own view, similar to a temporary table, and then simply join the two views and select the columns you need).

Which three months had the fastest growth? Which three months had the slowest growth? Do you see a relation between growth and NPS?

Answer. One possible solution is given below:

Get the growth rate per month:

```
SELECT year_month, new_customers, ROUND((new_customers - prev_new_customers) / cast (prev_new_
(SELECT year_month, new_customers, lag(new_customers, 1) OVER (ORDER BY year_month) prev_new_cu
(SELECT TO_CHAR(created_at, 'YYYY-MM') AS year_month, COUNT(id) AS new_customers
FROM customer
GROUP BY year_month
ORDER BY year_month) a
) b;
```

```
 year_month | new_customers | growth_percent
------------+---------------+----------------
 2018-01    |           291 |
 2018-02    |          1383 |            375
 2018-03    |          3005 |            117
 2018-04    |          4558 |             52
 2018-05    |          6313 |             39
```

```
 2018-06    |          8780 |             39
 2018-07    |         12422 |             41
 2018-08    |         19124 |             54
 2018-09    |         19632 |              3
 2018-10    |         22176 |             13
 2018-11    |         30288 |             37
 2018-12    |         60351 |             99
```

Create the views and combine the results:

```
CREATE VIEW month_nps AS
SELECT *, ROUND(((CAST(promoter AS DECIMAL) / total) - (CAST(detractor AS DECIMAL) / total)) *
(SELECT year_month,
SUM(CASE WHEN nps_class = 'promoter' THEN 1 ELSE 0 END) AS "promoter",
SUM(CASE WHEN nps_class = 'passive' THEN 1 ELSE 0 END) AS "passive",
SUM(CASE WHEN nps_class = 'detractor' THEN 1 ELSE 0 END) AS "detractor",
    COUNT(*) AS "total" FROM
(SELECT CASE
    WHEN avg_month_score > 8 THEN 'promoter'
    WHEN avg_month_score > 6 THEN 'passive'
    ELSE 'detractor'
END AS nps_class, year_month FROM
(SELECT TO_CHAR(score.created_at, 'YYYY-MM') AS year_month, customer_id, AVG(score) as avg_mon
GROUP BY year_month, customer_id) a) b
GROUP BY year_month
ORDER BY year_month) c
```

```
CREATE VIEW month_growth AS
SELECT year_month, new_customers, ROUND((new_customers - prev_new_customers) / cast (prev_new_
(SELECT year_month, new_customers, lag(new_customers, 1) OVER (ORDER BY year_month) prev_new_cu
(SELECT TO_CHAR(created_at, 'YYYY-MM') AS year_month, COUNT(id) AS new_customers
FROM customer
GROUP BY year_month
ORDER BY year_month) a
) b;
```

```
SELECT month_nps.year_month, nps, growth_percent
FROM month_nps
INNER JOIN month_growth ON month_nps.year_month = month_growth.year_month
LIMIT 100;
```

February, March, and December had the fastest growth rates of 375%, 117%, and 99% respectively. September, October, and November had the lowest growth rates of 3%, 13%, and 37% respectively. It appears as though the product issues occurred in September, as this is the only month where we observe the NPS score dropping (from 66 to 46). This appears to have had a large effect on customer growth, as September had the only single-figure growth rate (3%), which took 2 - 3 months to recover.