codecademy

SQL Capstone Project: Churn

Learn SQL from Scratch

Greg Rowsey

Table of Contents

1. Get familiar with the company

1. Get familiar with the company

1. 1 - How many months has the company been operating?

- 4 months total, December 2016 to March 2017

1.2 - For which months can you calculate churn?

- January through March, since there are no licenses active prior to December 1st

1.3 - What segments of users exist?

- There are 2 distinct segments, 87 and 30.

```
SELECT MIN(subscription_start) as [First Month],
 MAX(subscription_end) as [Last Month],
 COUNT(DISTINCT segment)
 FROM subscriptions;

 SELECT * FROM subscriptions LIMIT 100;
```

First Month	Last Month	Number of Segments
2016-12-01	2017-03-31	2

id	subscription_start	subscription_end	Segment
1	2016-12-01	2017-02-01	87
2	2016-12-01	2017-01-24	87

2. What is the overall churn trend?

2. Overall Churn Trend

2.1 - Temporary Months Table

- This sets up a table for our later cross join

first_day	last_day
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

```
WITH months AS
 (SELECT
  '2017-01-01' AS first_day,
  '2017-01-31' AS last_day
      UNION
  SELECT
  '2017-02-01' AS first_day,
  '2017-02-28' AS last_day
      UNION
  SELECT
  '2017-03-01' AS first_day,
  '2017-03-31' AS last_day
 ),
```

2. Overall Churn Trend

2.2 - Crossjoin months with subscriptions

- This enables us later to compare subscription start and end dates with the first and last day of each month, determining if the subscription was active or was canceled in that month

id	subscription_start	subscription_end	Segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2017-01-01	2017-01-31
1	2016-12-01	2017-02-01	87	2017-02-01	2017-02-28
1	2016-12-01	2017-02-01	87	2017-03-01	2017-03-31

```
WITH months AS
 (SELECT
  '2017-01-01' AS first_day,
  '2017-01-31' AS last_day
      UNION
  SELECT
  '2017-02-01' AS first_day,
  '2017-02-28' AS last_day
      UNION
  SELECT
  '2017-03-01' AS first_day,
  '2017-03-31' AS last_day
 ),
```

2. Overall Churn Trend

2.3 - Status counter using CASE

- This creates variables (columns), determining if the subscription was active or was canceled, for each segment
- If the ID was active in that month, the query returns 1, if not it returns 0
- If the ID was canceled in that month, the query returns 1, if not it returns 0

id	month	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	2017-01-01	1	0	0	0
1	2017-02-01	1	0	1	0
1	2017-03-01	0	0	0	0

```
status AS
(SELECT id, first_day AS month,

 CASE
      WHEN segment = 87 AND
            (subscription_start < first_day) AND
     (subscription_end >= first_day OR
subscription_end IS NULL) THEN 1 ELSE 0
      END
 AS is_active_87,

 CASE
      WHEN segment = 30 AND
      (subscription_start < first_day) AND
      (subscription_end >= first_day OR
subscription_end IS NULL) THEN 1 ELSE 0
      END
 AS is_active_30,

 CASE
      WHEN segment = 87 AND
      (subscription_end BETWEEN first_day AND
last_day) THEN 1 ELSE 0
      END
 AS is_canceled_87,

 CASE
      WHEN segment = 30 AND
      (subscription_end BETWEEN first_day AND
last_day) THEN 1 ELSE 0
      END
 AS is_canceled_30

 FROM crossjoin
),
```

2. Overall Churn Trend

2.4 - Aggregating Status counters

- This aggregates (sums) the columns to get the totals for each active/canceled Id for each segment
- It then groups them by month, so we can see the change over time
- Also summed both segments together to get Total Active and Total Canceled, also grouped by month

month	sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30	total_active	total_canceled
2017-01-01	279	291	70	22	570	92
2017-02-01	467	518	148	38	985	186
2017-03-01	541	718	258	84	1259	342

```
status_aggregate AS
(SELECT month,
       SUM(is_active_87) AS sum_active_87,
       SUM(is_active_30) AS sum_active_30,
       SUM(is_canceled_87) AS sum_canceled_87,
       SUM(is_canceled_30) AS sum_canceled_30,
       SUM(is_active_87)+SUM(is_active_30) AS
              total_active,
       SUM(is_canceled_87)+SUM(is_canceled_30) AS
              total_canceled
 FROM status
 GROUP BY month
)
```

2. Overall Churn Trend

2.5 - Calculating Churn by manipulating aggregates

- This calculates churn (#canceled that month / #active at beginning of month) for each month
  - Results are not rounded
- Overall, the churn is INCREASING as time continues

month	Churn 87	Churn 30	Total Churn
2017-01-01	.2508...	.0756...	.1614...
2017-02-01	.3169...	.0733...	.1888...
2017-03-01	.4768...	.1169...	.2716...

```
SELECT month,
1.0*sum_canceled_87/sum_active_87 AS [Churn 87],
1.0*sum_canceled_30/sum_active_30 AS [Churn 30],
1.0*(total_canceled)/(total_active) AS [TOTAL CHURN]
FROM status_aggregate;
```

3. Compare churn rates between segments

3. Churn rates between segments

3.1 -Churn is increasing at a much slower rate for segment 30 compared to segment 87

- THEREFORE the company should focus on segment 30

month	Churn 87	Churn 30	Total Churn
2017-01-01	.2508...	.0756...	.1614...
2017-02-01	.3169...	.0733...	.1888...
2017-03-01	.4768...	.1169...	.2716...

3. Churn rates between segments

3.2.1 - Modifying code to support larger number of segments

- Code can remain partially the same and partially simplified by passing in segment variable into the status temporary table
- Reduces the number of Cases when tabulating/counting status

id	month	Segment	is_active	is_canceled
1	2017-01-01	87	1	0
1	2017-02-01	87	1	1
1	2017-03-01	87	0	0

```
...
status AS
(SELECT id, first_day AS month, segment,

 CASE
      WHEN subscription_start < first_day AND
     (subscription_end >= first_day OR
subscription_end IS NULL) THEN 1 ELSE 0
      END
 AS is_active,

 CASE
      WHEN subscription_end BETWEEN first_day AND
last_day THEN 1 ELSE 0
      END
 AS is_canceled

 FROM crossjoin),
```

3. Churn rates between segments

3.2.2 - Modifying code to support larger number of segments

- Additional rows are added for each segment
- Grouped by month, segment
  - Ordered by segment, month to keep all segment data together
- Can easily support as many segments that may be added

month	Segment	sum_active	sum_canceled
2017-01-01	30	291	22
2017-02-01	30	518	38
2017-03-01	30	718	84
2017-01-01	87	279	70
2017-02-01	87	467	148
2017-03-01	87	541	258

```
...
status_aggregate AS
(SELECT month, segment,
       SUM(is_active) AS sum_active,
       SUM(is_canceled) AS sum_canceled
 FROM status
 GROUP BY month, segment
 ORDER BY segment, month
)
```

3. Churn rates between segments

3.2.3 - Modifying code to support larger number of segments

- Churn is calculated, ordered by segment
- Output is arranged slightly different, but numbers are the same

month	Segment	Churn
2017-01-01	30	.0756...
2017-02-01	30	.0733...
2017-03-01	30	.1169...
2017-01-01	87	.2508...
2017-02-01	87	.3169...
2017-03-01	87	.4768...

```
...
SELECT month, segment, 1.0*sum_canceled/sum_active AS
[Churn] FROM status_aggregate ORDER BY segment;
```

3. Churn rates between segments

3.3 - Using nested queries instead of temporary tables

- # of lines of code is highly reduced, though complexity of each line probably increased
- Columns and rows are pivoted
- Results are the same, just read from left to right

Segment	Churn in Jan	Churn in Feb	Churn in Mar
30	.0756...	.0733...	.1169...
87	.2058...	.3169...	.4768...

```
SELECT segment,

1.0*SUM([Canceled in Jan])/SUM([Active in Jan]) AS
[Churn in Jan],

1.0*SUM([Canceled in Feb])/SUM([Active in Feb]) AS
[Churn in Feb],

1.0*SUM([Canceled in Mar])/SUM([Active in Mar]) AS
[Churn in Mar]

FROM(
  SELECT *,
CASE WHEN subscription_start < '2017-01-01' AND
(subscription_end >= '2017-01-01' OR subscription_end
IS NULL) THEN 1 ELSE 0 END AS [Active in Jan],

CASE WHEN subscription_start < '2017-02-01' AND
(subscription_end >= '2017-02-01' OR subscription_end
IS NULL) THEN 1 ELSE 0 END AS [Active in Feb],

CASE WHEN subscription_start < '2017-03-01' AND
(subscription_end >= '2017-03-01' OR subscription_end
IS NULL) THEN 1 ELSE 0 END AS [Active in Mar],

CASE WHEN subscription_end BETWEEN '2017-01-01' AND
'2017-01-31' THEN 1 ELSE 0 END AS [Canceled in Jan],

CASE WHEN subscription_end BETWEEN '2017-02-01' AND
'2017-02-28' THEN 1 ELSE 0 END AS [Canceled in Feb],

CASE WHEN subscription_end BETWEEN '2017-03-01' AND
'2017-03-31' THEN 1 ELSE 0 END AS [Canceled in Mar]

FROM subscriptions
) GROUP BY segment;
```

Thank You!