codecademy

Calculating Churn Rates

Learn SQL from Scratch

Josh Le Tissier | 01/10/2018

Table of Contents

1. Get familiar with Codeflix

1.1 Get familiar with Codeflix

In order to calculate the churn rates for Codeflix I will needed to become familiar with the subscriptions table (see Query A and Table A)

How many months has the company been operating?

- The company has been operating for 4 months - this can be proven by running Query B and viewing results in Table B

What segments of users exist?

- There are 2 segments of users - this can be proven by running Query C and viewing results in Table C

Query A

```sql
/* Query to view the first 100 records of the subscriptions table */
SELECT *
FROM subscriptions
LIMIT 100;
```

Query B

```sql
/* Query with 2 aggregate functions to find the minimum subscription_start (earliest subscription start date) and maximum subscription_end (most recent subscription end date) of the subscriptions table */
SELECT MIN(subscription_start) AS 'Earliest Subscription Start Date',
       MAX(subscription_end) AS 'Most Recent Subscription End Date'
FROM subscriptions;
```

Query C

```sql
/* Query with a combination of the COUNT and DISTINCT aggregate functions to find the number of user segments in the subscriptions table */
SELECT COUNT(DISTINCT segment) AS 'No of User Segments'
FROM subscriptions;
```

Table A

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

Table B

Earliest Subscription Start Date	Most Recent Subscription End Date
2016-12-01	2017-03-31

Table C

No of User Segments
2

2. What is the overall churn trend since Codeflix started?

2. What is the overall churn trend since Codeflix started?

- I first created a ‘months’ temporary table that has ‘first_day’ and ‘last_day’ for each month using the ‘UNION’ function to add multiple rows in the table (Query A)
- I then used ‘CROSS JOIN’ to join this temporary table with the ‘subscriptions’ table (Query B)
- Next I created another temporary table, ‘status’, by using the ‘CASE’ function I was able to create a column for each status, ‘is_active’ and ‘is_canceled’ (Query C)

Query A

```
/* Query that creates the months temporary table */
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
),
```

Query B

```
/* Query that creates the cross_join temporary table by using CROSS JOIN on the subscriptions table and the months temporary table */
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months
),
```

Query C

```
/* Query that creates the status temporary table using CASE statements to determine whether the record id is active or canceled */
status AS
(SELECT id,
  first_day AS month,
  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 cross_join),
```

2. What is the overall churn trend since Codeflix started? (cont'd)

- Using the 'status' temporary table I created the 'status_aggregate' temporary table that has two columns; 1 for the SUM of the active statuses and 1 for the SUM of the canceled statuses (Query D)
- Having the SUM of each status I was able to use a SELECT query to calculate the churn rate since the company started (Query E)
- Since Codeflix started on 1st December 2016 the overall churn rate (up to and including 31st March 2017) has been 22.17% - see Table A.

Table A

Churn Rate (%)
22.17

Query D

```
52  /* Query that creates the status_aggregate temporary table by using the SUM function to find
    out how many records are either active or canceled */
53  status_aggregate AS
54  (SELECT SUM(is_active) AS 'sum_active',
55    SUM(is_canceled) AS 'sum_canceled'
56   FROM status
57   )
```

Query E

```
58  /* Query that calculates the churn rate. The calculations multiplied by 100 to produce the
    results as a percentage and are rounded to 2 decimal places for better readability */
59   SELECT ROUND(100.0 * sum_canceled / sum_active,2) AS 'Churn Rate (%)'
60   FROM status_aggregate;
```

2. What is the overall churn trend since Codeflix started? (cont'd)

In order to calculate the churn rate for each month I had to:

- Amend the temporary table 'status_aggregate' to SELECT 'month' and GROUP BY 'month' as well (Query F)
- Amend the SELECT query that calculates the churn rate to include 'month' as a column - (Query G)

We can see from Table B that the churn rate is increasing each month, however, the churn rate was drastically higher in March that it was in February. The rate at which the churn rate is increasing, is increasing!

Query F

```
52  /* Query that creates the status_aggregate temporary table by using the SUM function to find
    out how many records are either active or canceled for each month*/
53  status_aggregate AS
54  (SELECT month,
55    SUM(is_active) AS 'sum_active',
56    SUM(is_canceled) AS 'sum_canceled'
57   FROM status
58   GROUP BY month
59   )
```

Query G

```
60  /* Query that calculates the churn rate each month. The calculations multiplied by 100 to
    produce the results as a percentage and are rounded to 2 decimal places for better
    readability */
61   SELECT month,
62    ROUND(100.0 * sum_canceled / sum_active,2) AS 'Churn Rate (%)'
63   FROM status_aggregate;
```

Table B

Month	Churn Rate (%)
2017-01-01	16.17
2017-02-01	18.98
2017-03-01	27.43

3. Compare the churn rates between user segments

3. Compare the churn rates between user segments

In order to calculate the churn rate for each month I had to:

- Amend the CASE functions in the ‘status’ temporary table as well as select 2 new CASE functions to include user segment as part of the WHEN clause (Query A)
- Amend the SUM functions in the ‘status_aggregate’ temporary table and add 2 additional SUM functions to include the user segments (Query B)

Steps continued on the next slide...

Query A

```
/* Query that creates the status temporary table using CASE statements to determine whether the record id is active or canceled and of which user segment*/
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 cross_join),
```

Query B

```
 /* Query that creates the status_aggregate temporary table by using the SUM function to find out how many records are either active or canceled and of which user segment */
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'
 FROM status
 GROUP BY month
 )
```

3. Compare the churn rates between user segments (cont'd)

- Amend the SELECT query that calculates the churn rate to calculate each churn rate per user segment - (Query C)

This enabled me to answer the question, "Which segment of users should the company focus on expanding?"

- We can see from Table A that the churn rate is much higher for the '87' user segment than the '30' user segment each month (January: 25.18% vs 7.56%), (February: 32.03% vs 7.34%), (March: 48.59% vs 11.73%), therefore Codeflix should focus on expanding on that user segment to increase user retention.
- It is important to remember to put some focus on the '30' user segment as well because we can see from Table A that the churn rate for both user segments is increasing each month.

Conclusion: Nearly half of the '87' user segment canceled their subscription in March which is alarmingly high. Codeflix must act fast in order to increase retention of this user segment.

Query C

```
78   /* Query that calculates the churn rate of each user segment for each month. The
     calculations multiplied by 100 to produce the results as a percentage and are rounded to 2
     decimal places for better readability */
79   SELECT month,
80    ROUND(100.0 * sum_canceled_87 / sum_active_87,2) AS '87 Churn Rate (%)',
81    ROUND(100.0 * sum_canceled_30 / sum_active_30,2) AS '30 Churn Rate (%)'
82   FROM status_aggregate;
```

Table A

Month	87 Churn Rate (%)	30 Churn Rate (%)
2017-01-01	25.18	7.56
2017-02-01	32.03	7.34
2017-03-01	48.59	11.73