Analyze Data With SQL: Calculating Churn Rates

Heidi (Yeonwoo) Seo

codecademy

1. Context

codecademy

Context

Client: Codeflix

- Streaming video startup
- Four months since launch date

Task

- Measure Codeflix users' subscription churn rate
- Compare churn rates between two segments of users

2. Dataset

Dataset

Provided SQL table: `subscriptions`

- `id` : the subscription ID
- `subscription_start`: the start date of the subscription
- `subscription_end`: the end date of the subscription
- `segment` : identifies which segment the subscription owner belongs to

Database Schema subscriptions	
name	**type**
id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER
Rows: 2000	

* Codeflix requires a minimum subscription length of 31 days (users cannot start and end their subscription in the same month)

Dataset

```
--Query to check out data
SELECT *
FROM subscriptions
LIMIT 100;

--Query to identify the
range of months represented
SELECT
   MIN(subscription_start),
   MAX(subscription_end)
FROM subscriptions;
```

Query Results			
id	**subscription_start**	**subscription_end**	**segment**
1	2016-12-01	2017-02-01	87
2	2016-12-01	2017-01-24	87
3	2016-12-01	2017-03-07	87
4	2016-12-01	2017-02-12	87
5	2016-12-01	2017-03-09	87
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87

Query Results	
MIN(subscription_start)	**MAX(subscription_end)**
2016-12-01	2017-03-31

3. Calculate Churn Rates

codecademy

Calculate Churn Rates

Step 1: Create a temporary table `months` representing the first and last day of each month from January to March, 2017

Step 2: Create a temporary table `cross_join` from `subscriptions` and `months`

```
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
),
cross_join AS (
   SELECT *
   FROM subscriptions
   CROSS JOIN months
),
```

Calculate Churn Rates

Step 3: Create a temporary table `status` from the `cross_join` table containing

- `id` selected from `cross_join`
- `month` as an alias of `first_day`
- `is_active_87` created using a `CASE WHEN` to find any users from segment 87 who existed prior to the beginning of the month (`1` if true and `0` otherwise)
- `is_active_30` created using a `CASE WHEN` to find any users from segment 30 who existed prior to the beginning of the month (`1` if true and `0` otherwise)

```
status AS (
   SELECT
      id,
      first_day AS month,
      CASE
         WHEN (subscription_start <
         first_day)
            AND (subscription_end >
            first_day
            OR subscription_end IS
            NULL)
            AND (segment = 87) THEN 1
         ELSE 0
      END AS is_active_87,
      CASE
         WHEN (subscription_start <
         first_day)
            AND (subscription_end >
            first_day
            OR subscription_end IS
            NULL)
            AND (segment = 30) THEN 1
         ELSE 0
      END AS is_active_30,
)
```

Calculate Churn Rates

Step 4: Add an `is_canceled_87` and an `is_canceled_30` column to `status` (`1` if subscription is canceled during the month, `0` otherwise)

```
...
      CASE
         WHEN (subscription_end
         BETWEEN first_day AND
         last_day)
            AND (segment = 87) THEN 1
         ELSE 0
      END AS is_canceled_87,
      CASE
         WHEN (subscription_end
         BETWEEN first_day AND
         last_day)
            AND (segment = 30) THEN 1
         ELSE 0
      END AS is_canceled_30
FROM cross_join
),
```

Calculate Churn Rates

Step 5: Create a `status_aggregate` temporary table that is a `SUM` of the active and canceled subscriptions for each segment, for each month. The table should include

- `sum_active_87`
- `sum_active_30`
- `sum_canceled_87`
- `sum_canceled_30`

```
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
)
```

Calculate Churn Rates

Step 6: Calculate the churn rates for the two segments over the three month period

```
SELECT
   month,
   100.0 * sum_canceled_87 /
   sum_active_87 AS churn_rate_87,
   100.0 * sum_canceled_30 /
   sum_active_30 AS churn_rate_30
FROM status_aggregate;
```

4. Results & Conclusion

codecademy

Results

Maximum churn rate over the months: 48.6% from segment 87 in March 2017
Minimum churn rate over the months: 7.3% from segment 30 in February 2017

Average churn rate for segment 87 = 35.3%
Average churn rate for segment 30 = 8.9%

Query Results		
month	**churn_rate_87**	**churn_rate_30**
2017-01-01	25.1798561151079	7.56013745704467
2017-02-01	32.034632034632	7.33590733590734
2017-03-01	48.5875706214689	11.731843575419

Conclusion

- Segment 87's average churn rate is almost **4 times** the average churn rate of Segment 30, implying that Segment 87 is losing subscribers at 4 times the rate at which Segment 30 is losing subscribers.

  Codeflix must identify the underlying cause (such as advertising location, graphics, target demographics, user experience, fees, etc.) of this discrepancy to reverse this trend.