codecademy

Codeflix - Churn Rates

Learn SQL from Scratch

Washington Andrade – 2018.07.10

Table of Contents

codecademy

Get Familiar with Codeflix

1 – Get familiar with Codeflix

Codeflix is a video streaming service that charge a monthly subscription fee for access to their product.

How many months has the company been operating?

- ❑ Find first and last month of operation by using **MIN** and **MAX** to examine range of *subscription_start*.
- ❑ Codeflix has been operating for 4 months. Starting on December 1, 2016 and ending March 30, 2017.

Which months do you have enough information to calculate a churn rate?

- ❑ Find usable data to calculate churn rate by using **MIN** and **MAX** to examine range of *subscription_end*.
- ❑ The month of December has no *subscription_end* values, thus it cannot be calculated. Usable data ranges from January 1, 2017 to March 31, 2017.

What segments of users exist?

- ❑ A **SELECT DISTINCT** query finds distinct values of *segment*.
- ❑ Users segments are split between *87* and *30*.

```sql
--Check how many months Codeflix has been operational
SELECT MIN(subscription_start) AS 'First Month',
   MAX(subscription_start) AS 'Last Month'
FROM subscriptions;
```

Months of Operation	
First Month	**Last Month**
2016-12-01	2017-03-31

```sql
--Find usable data to calculate churn rate
SELECT MIN(subscription_end) AS 'First Sub End',
   MAX(subscription_end) AS 'Last Sub End'
FROM subscriptions;
```

Months with Churn	
First Sub End	**Last Sub End**
2017-01-01	2017-03-31

```sql
--Find existing consumer segment
SELECT DISTINCT segment AS 'Segment'
FROM subscriptions;
```

Segments Available	
87	**30**

codecademy

Codeflix’s Overall Churn Trend

2 – Codeflix's overall churn trend

What is the overall churn trend since the company started?

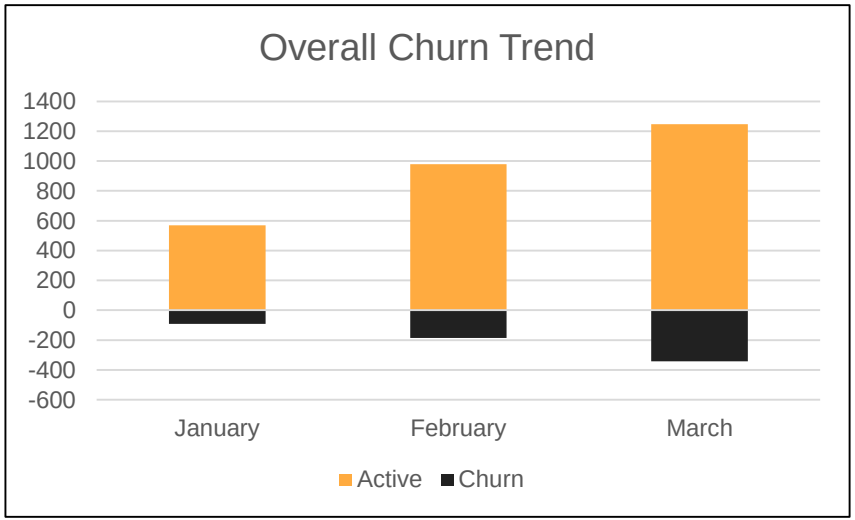
- ❑ Create temporary table of months using **WITH.**
- ❑ Create temporary table cross_join from subscription and months using **CROSS JOIN.**
- ❑ Create temporary table, status, from cross_join containing id selected from cross_join, *month* as an alias of first_day, *is_active*, using **CASE.**
- ❑ Add an *is_cancelled* column to status temporary table using **CASE** and **BETWEEN** statement.
- ❑ Create status_aggregate temporary table that is a **SUM** of the active and cancelled subscriptions, for each month.
- ❑ Divide canceled subscriptions by active subscriptions for each month (churn_rate = canceled / active), multiplied by 1.0 to cast the result to a float.

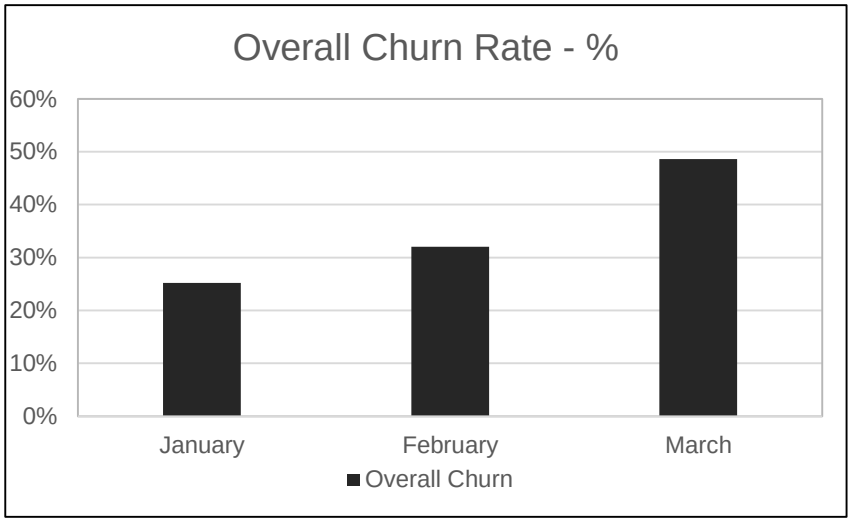
```
37  --Create status_aggregate temporary table that is the SUM of the overall
    active and cancelled subscriptions, for each month
38  status_aggregate AS
39    (SELECT
40      month,
41      SUM(is_active) AS active,
42      SUM(is_canceled) AS canceled
43    FROM status
44    GROUP BY month)
45  --Calculate overall churn rate over the three month period
46  SELECT
47    month,
48    1.0 * canceled / active AS churn_rate
49  FROM status_aggregate;
```

Codeflix Overall Churn Trend			
Status	Active	Churn	Net
January	569	-92	**661**
February	980	-186	**1166**
March	1247	-342	**1589**

Codeflix Churn Rates - %	
January	16%
February	19%
March	27%

2 – Codeflix’s overall churn trend – Cont.





codecademy

Codeflix’s Churn Trend Per Segment

3 – Codeflix's Churn Trend per Segment

Compare the churn rates between user segments

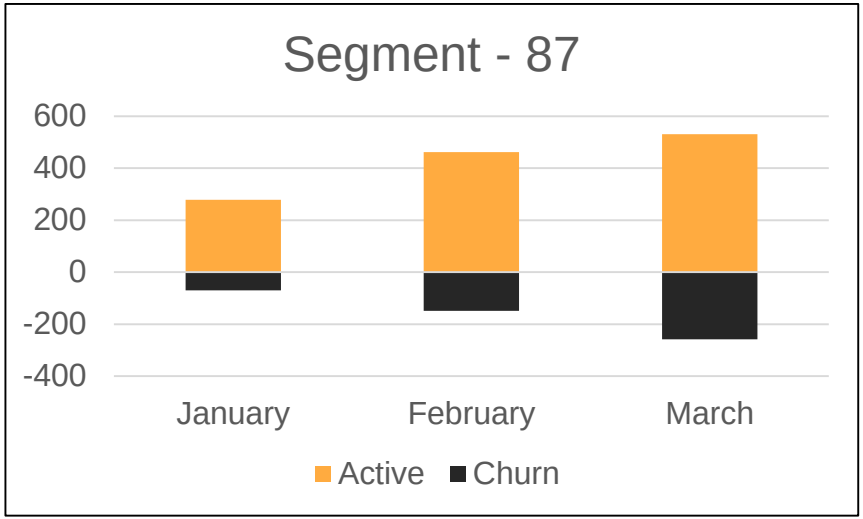
- ❑ Create temporary table of months using **WITH.**
- ❑ Create temporary table cross_join from subscription and months using **CROSS JOIN.**
- ❑ Create temporary table, status, from cross_join containing id selected from cross_join, *month* as an alias of first_day, *is_active_87*, *is_active_30* using **CASE.**
- ❑ Add an *is_cancelled_87* and an *is_cancelled_30* column to status temporary table using **CASE**.
- ❑ Create status_aggregate temporary table that is a **SUM** of the active and cancelled subscriptions for each segment, for each month. Result columns should be: *sum_active_87*, *sum_active_30*, *sum_canceled_87*, *sum_canceled_30*.

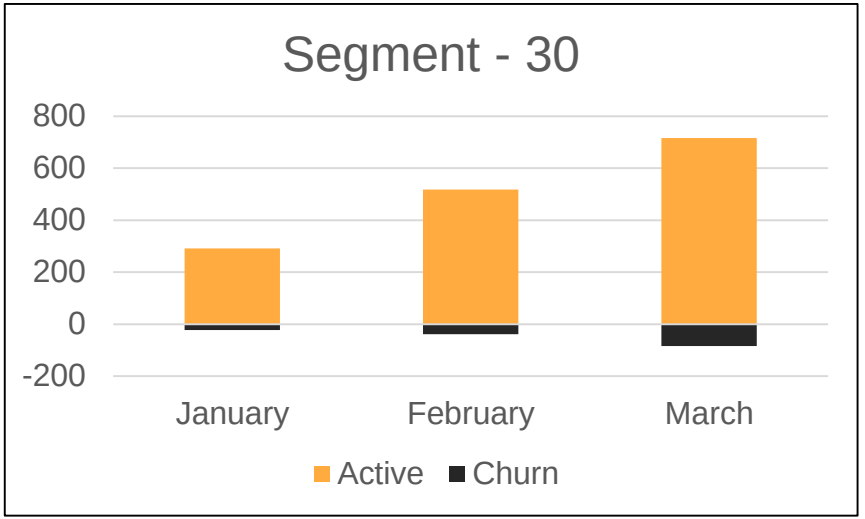
```
59  --Create status_aggregate temporary table that is a SUM of the active and cancelled subscriptions for
    each segment, for each month
60  status_aggregate AS
61  (SELECT month,
62    SUM(is_active_87) AS sum_active_87,
63    SUM(is_active_30) AS sum_active_30,
64    SUM(is_canceled_87) AS sum_canceled_87,
65    SUM(is_canceled_30) AS sum_canceled_30
66    FROM status
67  GROUP BY month)
```

Churn Trend per Segment

Segment	87			30		
Status	Active	Churn	Net	Active	Churn	Net
January	278	70	**208**	291	22	**269**
February	462	148	**314**	518	38	**480**
March	531	258	**273**	716	84	**632**

3 – Codeflix’s Churn Trend per Segment – Cont.





3 – Codeflix's Churn Trend per Segment – Cont.

Compare the churn rates between user segments

- ❑ Using SELECT statement we can calculate the churn rate within SQL. To find churn rate, divide canceled subscriptions by active subscriptions for each month and segment (churn_rate = canceled / active), multiplied by 1.0 to cast the result to a float.
- ❑ BONUS – You can easily round whole numbers into percentages by exporting them to o excel and formatting them into percent.

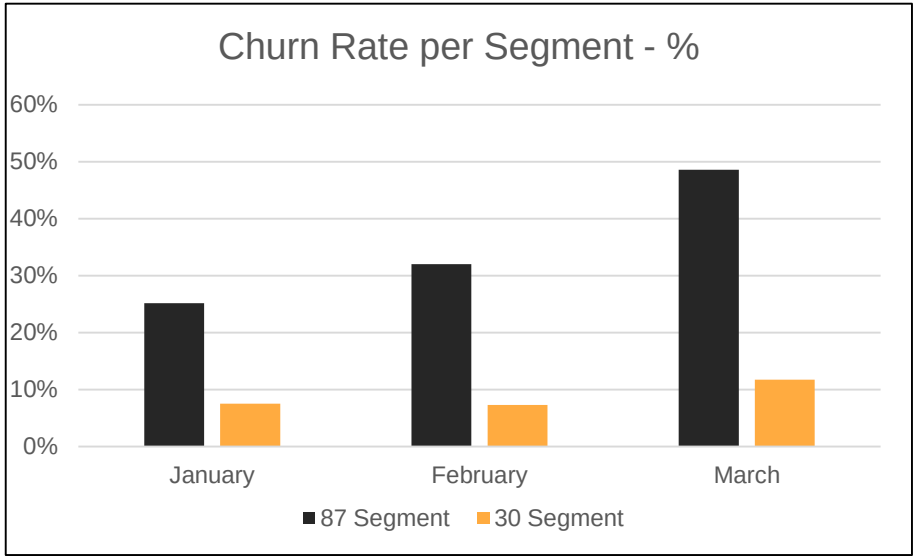
```sql
--Calculate churn rate for the two segments over the three month period
SELECT
  month,
  1.0 * sum_canceled_87 / sum_active_87 AS churn_rate_87,
  1.0 * sum_canceled_30 / sum_active_30 AS churn_rate_30
FROM status_aggregate;
```

Churn Rates per Segment - %		
Segment	**87**	**30**
January	25%	8%
February	32%	7%
March	49%	12%

codecademy

Conclusion

4 – Which segment of users should be focused?



- Considering the higher churn rates for **segment 87**, it is suggested that the company focus on acquiring **segment 30** customers since this target is showing better retention rates in the long term.
- Alternatively, providing more appealing offers and/or a loyalty program for costumers on **segment 87** could help improve retention rates on that segment.
- Consider compiling churn trend analysis on a 3 month cadence considering new offers, segments and integrated marketing initiatives.