codecademy

Codeflix Churn Rates

Learn SQL from Scratch - Erik Nguyen

Prime Focus

1. Codeflix
2. Churn rates
3. Churn rates between segments
4. Outcome

1. Codeflix

1. Codeflix

Codeflix is a monthly video streaming service that has a table called “subscriptions”. To get familiar with the data we need to know all of their columns.

As you can see below the “subscriptions” table contains 4 columns, id, subscription_start, subscription_end and segment.

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

```
-- Select the first 50 rows of the table and show all
--the columns its contains
SELECT *
FROM subscriptions
LIMIT 50;
```

1.1 Codeflix

To start calculating the churn rate of codeflix there is some useful information we need to know. For instance how long has Codeflix been operating? How much information do we have available to calculate churn rate?

Min_start_date	Max_start_date
2016-12-01	2017-03-30

Based on this query we can see that the streaming service has been operational for at least 4 months. But it doesn't necessarily means that we are able to calculate the churn rate with all four months. Since the first subscription start date is on 2016-12-01, it means the first subscription end date will be on the 2017-01-01.

```
-- Calculate the operating time of Codeflix
 SELECT
MIN(subscription_start) AS Min_start_date,
MAX(subscription_start) AS Max_start_date
FROM subscriptions;

-- Showing the missing values
SELECT
Subscription_start,
Subscription_end
FROM subscriptions
WHERE subscription_end IS NULL;
```

1.2 Codeflix

How many customers and segment does exist?

Segment	Customers
30	1000
87	1000

From the query we can see that Codeflix have 2 difference segment, together they contains 1000 customers.

```
-- Number of segment and customers
 SELECT DISTINCT segment , COUNT (*) AS Customer
FROM subscriptions
GROUP BY 1;
```

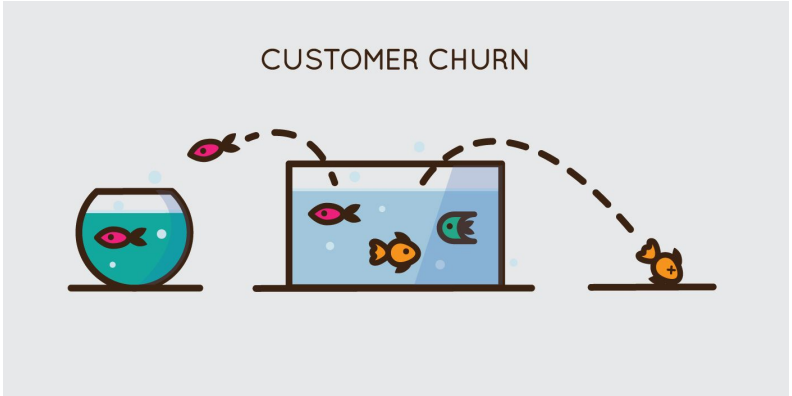
2. Churn rates

2 Churn rates

What are churn rates and how is it useful to Codeflix?

- Churn rate is the percentage of subscribers that have canceled within a certain period of time, in this scenario a month.
- With churn rate we can understand better how the company has increased or decreased based on if the rates are higher or lower.
- We can also see which segment has given the most effect and needs to be focused on.

$$\frac{\text{cancellations}}{\text{total subscribers}}$$



2.1 Calculating churn rates

Which period of time do we want to compute the churn rate?

- Earlier we observed that Codeflix has been operational for 4 months but we only have enough data to calculate the churn rate of 3 months.
- First we need create a new table to store all of the months and cross-join it together with the “subscriptions” table.

```
--Create a temporary table for all the months
--Cross-join the temporary with “Subscriptions” 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 fist_day,
      '2017-03-31' AS last_day
),
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months),
```

2.1 Calculating churn rates

How do we get the number of active and cancelled users?

- To be able to calculate the churn rates we need to know the number of new subscribers and cancelled users. To do that we create a new table called “status” and use “CASE” to measure the number of cancelled/active users.
- With the CASE statement we can with the given conditions then return a value when the first condition is met. Much like an IF-THEN-ELSE statement.

```
--Cross-join previous table with the “status” table
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),
```

2.1 Calculating churn rates

What are the overall churn rates of the streaming service, Codeflix?

- Next we create another table called “status_aggregate” and aggregate the “status” table into it, sort it by month.
- Taking the number of cancel divide it by the number of active users to calculate the overall churn rates of Codeflix

```
--Creating a new table “status_aggregate”
status_aggregate AS
(SELECT
month,
SUM(is_active) AS sum_active,
SUM(is_canceled) AS sum_canceled
FROM status
GROUP BY month)

--Compute the overall churn rate
SELECT month,
1.0* sum_canceled / sum_active AS
'overall_churn_rate'
FROM status_aggregate;
```

Month	overall_churn_rate
2017-01-01	0.161687170474517
2017-02-01	0.189795918367347
2017-03-01	0.274258219727346

2.1 Calculating churn rates

```
--The total query
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 fist_day,
       '2017-03-31' AS last_day
),
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months),
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),
status_aggregate AS
(SELECT
month,
SUM(is_active) AS sum_active,
SUM(is_canceled) AS sum_canceled
FROM status
GROUP BY month)
SELECT month, 1.0* sum_canceled / sum_active AS
'overall_churn_rate'
FROM status_aggregate
;
```

3. Churn rates between segments

3 Churn rates between segments

- The fourth column of Codeflix database contains two segments, 87 and 30. We want to compare them to figure out which segments we need to change or adjust.
- To do that we need to modify our query, and separate segment 87 and 30 from each other.

```
--Previous CASE query for all users
(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
 CASE
       WHEN (subscription_end BETWEEN first_day AND
last_day)
 THEN 1
 ELSE 0
 END as is_canceled
```

```
--Modified query select for cancel/active users of
--segment 87
(SELECT id, first_day AS month, segment,
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_end BETWEEN first_day AND
last_day)
       AND (segment = 87)
    THEN 1
 ELSE 0
 END as is_canceled_87,
```

3 Churn rates between segments

- Now we do the same modification for segment 30 and add it into the CASE statement.

```
(SELECT id, first_day AS month, segment,
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,
 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),
```

3 Churn rates between segments

- Next step is to compute the churn rates for both segments using the same table “status_aggregate” with modification for it to give us the rates for separate segments.
- Rename both segment to “churn_rate_87” and “churn_rate_30”
- Modifying the “status_aggregate” table to compute the churn rate of segment 87 and 30 along with the total churn rates.

Month	overall_churn_rate	churn_rate_30	churn_rate_87
2017-01-01	0.161687170474517	0.0756013745704467	0.251798561151079
2017-02-01	0.189795918367347	0.0733590733590734	0.32034632034632
2017-03-01	0.274258219727346	0.11731843575419	0.485875706214689

```
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) AS sum_active,
SUM(is_canceled) AS sum_canceled
FROM status
GROUP BY month)
SELECT
1.0* sum_canceled_87 / sum_active_87 AS
churn_rate_87,
1.0* sum_canceled_30 / sum_active_30 AS churn_rate_30
1.0* sum_canceled / sum_active AS overall_churn_rate
FROM status_aggregate
;
```

4. Outcome

- Overall has the churn rates been increasing throughout the months. Segment 87 has been consistently higher than segment 30. However to be able to increase sales, Codeflix needs to make changes to the way segment 87 is being handled. Codeflix should also focus on expanding segment 30.