codecademy

B. Parry – Churn Analysis

Learn SQL from Scratch – Capstone Project

Table of Contents

1.1 Overview of *subscriptions*

This project analyzed the ‘subscriptions’ database for the fictional firm Codeflix. This contained Codeflix’s subscription data over a 4 month period 12/2016 – 3/2017. The table below contains some examples from this data displaying its format and giving some example values. Notes:

- *id* is the unique identifier for each subscription.
- *subscription_start* is when this subscription began, this dataset only contains subscriptions from 2016-12-01 onwards.
- *subscription_end* is when the subscription ended, it can contain null values for subscriptions that have not yet ended.
- *segment* has two values depending on how the customer was acquired and can be either 87 or 30.

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

1.1 Churn Rate By Month

To determine churn rate by month we

1. Created a cross join table with a new row of data for each original subscription for every month in question (Jan, Feb, Mar).
2. Used CASE WHEN to mark subscriptions active or inactive based on subscription status for each month.
3. Summed this data and calculated churn for each month as:

   cancelled subscriptions / active subscriptions

As we can see Codeflix's churn rate increased over time.
* Note SQL code to the right formatted to fit on page.

month	churn_rate
2017-01-01	0.16
2017-02-01	0.19
2017-03-01	0.27

```
WITH months AS (
              SELECT
              '2017-01-01' AS first_day,
              '2017-01-31' AS last_day
#… Repeat for remaining months …
),

cross_join AS (
#… cross join subscriptions and 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_cancelled
FROM cross_join
), status_aggregate AS (
SELECT month, SUM(is_active) AS sum_active,
SUM(is_cancelled) AS sum_cancelled
FROM status
GROUP BY month
)

SELECT month, ROUND(1.0 * sum_cancelled / sum_active,
2) AS churn_rate
FROM status_aggregate;
```

1.1 Churn Rate By Segment

- To find the churn rate by segment we used a similar process to that for calculating by month except added an AND statement to our cases to find subscription states for the two different segments.
- We can see from the output shown below that the churn rate for subscriptions gained through the 87 segment are consistently higher than the churn rate for the 30 segment by at least a factor of 3.

*Note that associated SQL code is available in the attached file.

month	churn_rate_87	churn_rate_30
2017-01-01	0.25	0.08
2017-02-01	0.32	0.07
2017-03-01	0.49	0.12