Case Study - Foodie-Fi

Introduction

Subscription-based businesses are super popular and Danny realized that there was a large gap in the market - he wanted to create a new streaming service that only had food-related content - something like Netflix but with only cooking shows!

Danny finds a few smart friends to launch his new startup Foodie-Fi in 2020 and started selling monthly and annual subscriptions, giving their customers unlimited on-demand access to exclusive food videos from around the world!

Danny created Foodie-Fi with a data-driven mindset and wanted to ensure all future investment decisions and new features were decided using data. This case study focuses on using subscription-style digital data to answer important business questions.

Available Data

Danny has shared the data design for Foodie-Fi and short descriptions on each of the database tables - our case study focuses on only 2 tables but there will be challenging to create a new table for the Foodie-Fi team.

Entity Relationship Diagram

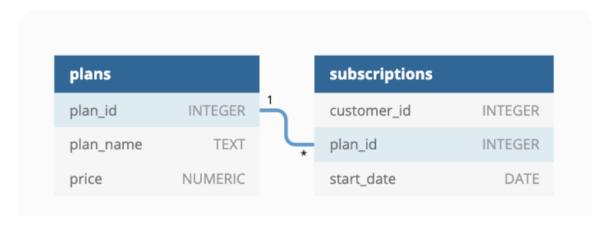


Table 1: plans

Customers can choose which plans to join Foodie-Fi when they first sign up.

Basic plan customers have limited access and can only stream their videos and are only available monthly at \$9.90

Pro plan customers have no watch time limits and are able to download videos for offline viewing. Pro plans start at \$19.90 a month or \$199 for an annual subscription.

Customers can sign up for an initial 7-day free trial and will automatically continue with the pro monthly subscription plan unless they cancel, downgrade to basic, or upgrade to an annual pro plan at any point during the trial.

When customers cancel their Foodie-Fi service - they will have a churn plan record with a null price but their plan will continue until the end of the billing period.

Table 2: subscriptions

Customer subscriptions show the exact date when their specific plan_id starts. If customers downgrade from a pro plan or cancel their subscription - the higher plan will remain in place until the period is over - the start date in the subscriptions table will reflect the date that the actual plan changes.

When customers upgrade their account from a basic plan to a pro or annual pro plan - the higher plan will take effect straight away.

When customers churn - they will keep their access until the end of their current billing period but the start date will be technically the day they decided to cancel their service

```
CREATE DATABASE foodie_fi;
CREATE OR REPLACE TABLE plans (
 plan_id INTEGER,
 plan_name VARCHAR(13),
 price DECIMAL(5,2)
);
INSERT INTO plans
 (plan_id, plan_name, price)
VALUES
 ('0', 'trial', '0'),
 ('1', 'basic monthly', '9.90'),
 ('2', 'pro monthly', '19.90'),
 ('3', 'pro annual', '199'),
 ('4', 'churn', null);
CREATE OR REPLACE TABLE subscriptions (
 customer_id INTEGER,
 plan_id INTEGER,
 start_date DATE
);
```

INSERT INTO subscriptions

(customer_id, plan_id, start_date)

VALUES

- ('1', '0', '2020-08-01'),
- ('1', '1', '2020-08-08'),
- ('2', '0', '2020-09-20'),
- ('2', '3', '2020-09-27'),
- ('3', '0', '2020-01-13'),
- ('3', '1', '2020-01-20'),
- ('4', '0', '2020-01-17'),
- ('4', '1', '2020-01-24'),
- ('4', '4', '2020-04-21'),
- ('5', '0', '2020-08-03'),
- ('5', '1', '2020-08-10'),
- ('6', '0', '2020-12-23'),
- ('6', '1', '2020-12-30'),
- ('6', '4', '2021-02-26'),
- ('7', '0', '2020-02-05'),
- ('7', '1', '2020-02-12'),
- ('7', '2', '2020-05-22'),
- ('8', '0', '2020-06-11'),
- ('8', '1', '2020-06-18'),
- ('8', '2', '2020-08-03'),
- ('9', '0', '2020-12-07'),
- ('9', '3', '2020-12-14'),
- ('10', '0', '2020-09-19'),
- ('10', '2', '2020-09-26'),
- ('11', '0', '2020-11-19'),
- ('11', '4', '2020-11-26'),

```
('12', '0', '2020-09-22'),
```

```
('23', '3', '2020-05-20'),
```

```
('36', '0', '2020-02-25'),
```

```
('46', '3', '2020-08-06'),
```

```
('58', '1', '2020-07-11'),
```

```
('69', '0', '2020-03-07'),
```

```
('79', '2', '2020-08-06'),
```

```
('90', '3', '2021-04-28'),
```

```
('102', '0', '2020-06-02'),
```

```
('113', '2', '2020-09-13'),
```

```
('124', '1', '2020-03-24'),
```

```
('137', '2', '2020-08-19'),
```

```
('149', '1', '2020-12-26'),
```

```
('162', '4', '2020-03-01'),
```

```
('174', '3', '2020-07-10'),
```

```
('187', '0', '2020-09-19'),
```

```
('199', '0', '2020-12-09'),
```

```
('211', '1', '2020-10-17'),
```

```
('223', '0', '2020-08-01'),
```

```
('234', '0', '2020-01-19'),
```

```
('246', '0', '2020-01-27'),
```

```
('257', '4', '2021-04-22'),
```

```
('268', '4', '2020-11-07'),
```

```
('280', '2', '2020-10-28'),
```

```
('292', '1', '2020-08-21'),
```

```
('302', '2', '2020-01-16'),
```

```
('313', '1', '2020-01-22'),
```

```
('325', '4', '2020-05-25'),
```

```
('337', '1', '2020-11-23'),
```

```
('348', '4', '2020-09-28'),
```

```
('360', '2', '2020-09-03'),
```

```
('371', '0', '2020-05-06'),
```

```
('381', '1', '2020-07-28'),
```

```
('393', '2', '2020-05-05'),
```

```
('405', '4', '2020-03-09'),
```

```
('417', '2', '2020-02-04'),
```

```
('429', '4', '2020-07-30'),
```

```
('441', '0', '2020-09-27'),
```

```
('453', '0', '2020-02-15'),
```

```
('465', '0', '2020-10-24'),
```

```
('476', '2', '2020-08-29'),
```

```
('488', '0', '2020-02-15'),
```

```
('499', '0', '2020-02-17'),
```

```
('510', '3', '2020-06-19'),
```

```
('522', '1', '2020-09-02'),
```

```
('533', '0', '2020-02-01'),
```

```
('545', '0', '2020-03-05'),
```

```
('557', '0', '2020-03-02'),
```

```
('570', '0', '2020-10-22'),
```

```
('581', '3', '2020-06-01'),
```

```
('594', '0', '2020-01-15'),
```

```
('607', '0', '2020-01-02'),
```

```
('618', '4', '2020-08-27'),
```

```
('632', '1', '2020-05-01'),
```

```
('645', '0', '2020-05-07'),
```

```
('655', '1', '2020-04-29'),
```

```
('668', '0', '2020-02-13'),
```

```
('680', '0', '2020-04-11'),
```

```
('690', '2', '2020-06-13'),
```

```
('703', '0', '2020-11-02'),
```

```
('713', '2', '2020-09-22'),
```

```
('726', '4', '2020-03-09'),
```

```
('738', '0', '2020-01-22'),
```

```
('750', '0', '2020-07-03'),
```

```
('762', '2', '2020-10-14'),
```

```
('774', '0', '2020-12-04'),
```

```
('786', '0', '2020-05-10'),
```

```
('798', '2', '2020-12-05'),
```

```
('810', '1', '2020-11-29'),
```

```
('821', '2', '2020-10-06'),
```

```
('833', '0', '2020-10-08'),
```

```
('844', '0', '2020-10-14'),
```

```
('855', '1', '2020-06-24'),
```

```
('866', '4', '2021-03-25'),
```

```
('878', '1', '2020-08-03'),
```

```
('889', '2', '2020-09-13'),
```

```
('900', '2', '2020-10-04'),
```

```
('912', '2', '2020-12-23'),
```

```
('923', '2', '2020-09-04'),
```

```
('934', '0', '2020-01-07'),
```

```
('945', '1', '2020-01-14'),
```

```
('957', '4', '2020-05-05'),
```

```
('968', '2', '2020-11-29'),
```

```
('981', '0', '2020-02-16'),
```

```
('994', '1', '2020-08-01'),
 ('994', '2', '2020-08-27'),
 ('995', '0', '2020-06-11'),
 ('995', '1', '2020-06-18'),
 ('995', '2', '2020-12-06'),
 ('996', '0', '2020-11-11'),
 ('996', '1', '2020-11-18'),
 ('996', '4', '2020-12-07'),
 ('997', '0', '2020-07-27'),
 ('997', '1', '2020-08-03'),
 ('997', '2', '2020-08-26'),
 ('997', '4', '2020-11-14'),
 ('998', '0', '2020-10-12'),
 ('998', '2', '2020-10-19'),
 ('999', '0', '2020-10-23'),
 ('999', '2', '2020-10-30'),
 ('999', '4', '2020-12-01'),
 ('1000', '0', '2020-03-19'),
 ('1000', '2', '2020-03-26'),
 ('1000', '4', '2020-06-04');
SELECT * FROM PLANS;
SELECT * FROM subscriptions;
```

--1. How many customers has Foodie-Fi ever had?

SELECT

COUNT(DISTINCT customer_id) AS unique_customer FROM subscriptions;

--2. What is the monthly distribution of trial plan start_date values for our dataset — use the start of the month as the group by value.

```
SELECT
```

```
DATE_PART('month',start_date) AS month_date, -- Cast month as integer

TO_CHAR(start_date, 'MMMM') AS month_name, -- Cast month as string

COUNT(*) AS trial_subscriptions

FROM subscriptions s

JOIN plans p

ON s.plan_id = p.plan_id

WHERE s.plan_id = 0
```

GROUP BY 1, 2

ORDER BY month_date;

- --March has the highest number of trial plans, whereas February has the lowest number of trial
- --3. What plan start_date values occur after the year 2020 for our dataset? Show the breakdown by count of events for each plan_name.

```
SELECT
```

plans.

```
p.plan_id,
p.plan_name,

COUNT(*) AS events

FROM subscriptions s

JOIN plans p
  ON s.plan_id = p.plan_id

WHERE s.start_date >= '2021-01-01'
```

```
GROUP BY p.plan_id, p.plan_name
ORDER BY p.plan_id;
--4. What is the customer count and percentage of customers who have churned (rounded to 1
decimal place)?
SELECT
COUNT(*) AS churn_count,
 ROUND(100 * COUNT(*)::NUMERIC / (
  SELECT COUNT(DISTINCT customer_id)
  FROM subscriptions),1) AS churn_percentage
FROM subscriptions s
JOIN plans p
ON s.plan_id = p.plan_id
WHERE s.plan_id = 4;
--There are 307 customers who churned, which is 30.7% of Foodie-Fi customer base.
--5. How many customers have churned straight after their initial free trial.
-- Find ranking of plans by customer and plan type
WITH ranking AS (
SELECT
s.customer_id,
s.plan_id,
 p.plan_name,
-- Run a ROW_NUMBER() to rank plans from 0 to 4
 ROW_NUMBER() OVER (
```

```
PARTITION BY s.customer_id
  ORDER BY s.plan_id) AS plan_rank
FROM subscriptions s
JOIN plans p
ON s.plan_id = p.plan_id)
SELECT
COUNT(*) AS churn_count,
 ROUND(100 * COUNT(*) / (
  SELECT COUNT(DISTINCT customer_id)
  FROM subscriptions),0) AS churn_percentage
FROM ranking
WHERE plan_id = 4 -- Filter to churn plan
AND plan_rank = 2;-- Filter to rank 2 as customers who churned immediately after trial have churn
plan ranked as 2
--92 customers churned straight after the initial free trial which is 9% of entire customer base.
--6. What is the number and percentage of customer plans after their initial free trial?
-- To retrieve next plan's start date located in the next row based on current row
WITH next_plan_cte AS (
SELECT
customer_id,
 plan_id,
 LEAD(plan_id, 1) OVER( -- Offset by 1 to retrieve the immediate row's value below
  PARTITION BY customer_id
  ORDER BY plan_id) as next_plan
FROM subscriptions)
```

SELECT

```
next_plan,
 COUNT(*) AS conversions,
 ROUND(100 * COUNT(*)::NUMERIC / (
  SELECT COUNT(DISTINCT customer_id)
  FROM subscriptions),1) AS conversion_percentage
FROM next_plan_cte
WHERE next_plan IS NOT NULL
AND plan_id = 0
GROUP BY next_plan
ORDER BY next_plan;
--7. What is the customer count and percentage breakdown of all 5 plan_name values at 2020-12-
31?
--Retrieve next plan's start date located in the next row based on current row
WITH next_plan AS(
SELECT
customer_id,
plan_id,
start_date,
LEAD(start_date, 1) OVER(PARTITION BY customer_id ORDER BY start_date) as next_date
FROM subscriptions
WHERE start_date <= '2020-12-31'
),
-- Find customer breakdown with existing plans on or after 31 Dec 2020
customer_breakdown AS (
SELECT
  plan_id,
  COUNT(DISTINCT customer_id) AS customers
 FROM next_plan
```

```
WHERE
  (next_date IS NOT NULL AND (start_date < '2020-12-31'
   AND next_date > '2020-12-31'))
  OR (next_date IS NULL AND start_date < '2020-12-31')
 GROUP BY plan_id)
SELECT plan_id, customers,
 ROUND(100 * customers::NUMERIC / (
  SELECT COUNT(DISTINCT customer_id)
  FROM subscriptions),1) AS percentage
FROM customer_breakdown
GROUP BY plan_id, customers
ORDER BY plan_id;
--8. How many customers have upgraded to an annual plan in 2020?
SELECT
COUNT(DISTINCT customer_id) AS unique_customer
FROM subscriptions
WHERE plan_id = 3
AND start_date <= '2020-12-31';
--9. How many days on average does it take for a customer to an annual plan from the day they join
Foodie-Fi?
-- Filter results to customers at trial plan = 0
WITH trial_plan AS
 (SELECT
  customer_id,
  start_date AS trial_date
 FROM subscriptions
 WHERE plan_id = 0
```

```
),
-- Filter results to customers at pro annual plan = 3
annual_plan AS
(SELECT
  customer_id,
  start_date AS annual_date
 FROM subscriptions
WHERE plan_id = 3
)
SELECT
 ROUND(AVG(annual_date - trial_date),0) AS avg_days_to_upgrade
FROM trial_plan tp
JOIN annual_plan ap
ON tp.customer_id = ap.customer_id;
--10. How many customers downgraded from a pro monthly to a basic monthly plan in 2020?
-- Retrieve next plan's start date located in the next row based on current row
WITH next_plan_cte AS (
SELECT
  customer_id,
  plan_id,
  start_date,
  LEAD(plan_id, 1) OVER(
   PARTITION BY customer_id
   ORDER BY plan_id) as next_plan
 FROM subscriptions)
```

COUNT(*) AS downgraded

FROM next_plan_cte

WHERE start_date <= '2020-12-31'

AND plan_id = 2

AND next_plan = 1;