Challenge-3

FOODIE-FI

By: Harshita Aswani

INTRODUCTION

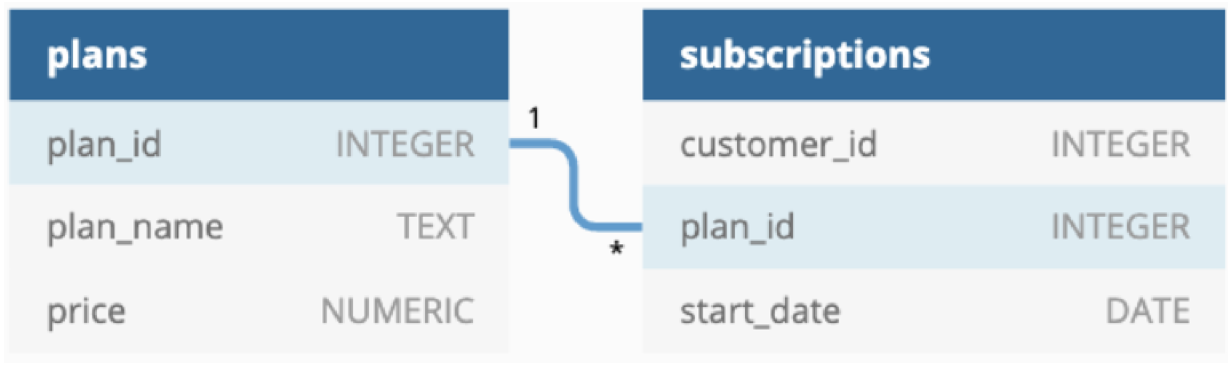
Subscription-based businesses are super popular and Danny realized that there was a large gap in the market - he wanted to create a new streaming service that only had food-related content - something like Netflix but with only cooking shows!

Danny found 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.

Entity Relationship Diagram

Danny has shared with me 2 key datasets for this case study whose entity relationship diagram is given below:



Tables

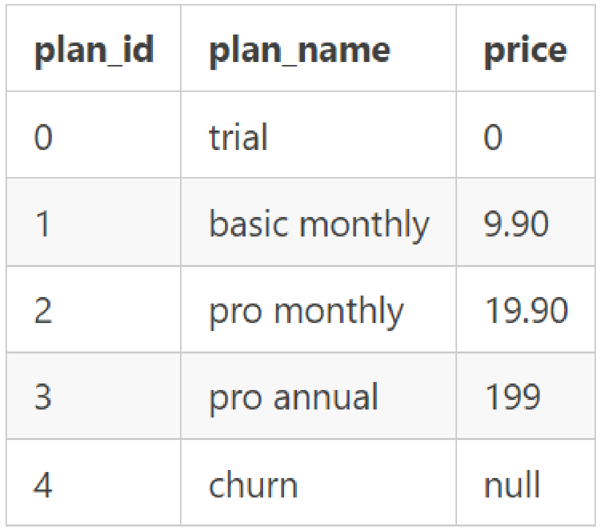
Customers can choose which plans to join Foodie-Fi when they first sign up.

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

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

Customers can sign up to an initial 7 day free trial 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.



plan_id	plan_name	price
0	trial	0
1	basic monthly	9.90
2	pro monthly	19.90
3	pro annual	199
4	churn	null

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 decide to cancel their service.

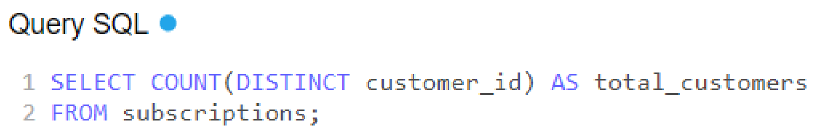
customer_id	plan_id	start_date
1	0	2020-08-01
1	1	2020-08-08
2	0	2020-09-20
2	3	2020-09-27
11	0	2020-11-19
11	4	2020-11-26
13	0	2020-12-15
13	1	2020-12-22
13	2	2021-03-29
15	0	2020-03-17
15	2	2020-03-24
15	4	2020-04-29
16	0	2020-05-31
16	1	2020-06-07
16	3	2020-10-21

Data Cleaning

```
-- Handling NULL values in plans table
UPDATE plans
SET price = 0  -- Or any default value
WHERE price IS NULL;
```

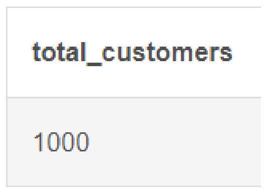
Case Study Questions

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

a. 

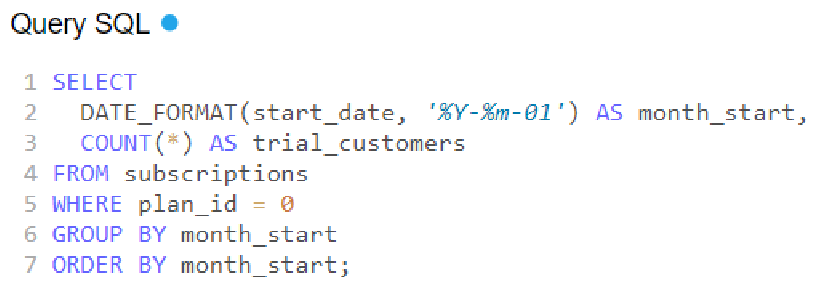
Query SQL ●

```sql
SELECT COUNT(DISTINCT customer_id) AS total_customers
FROM subscriptions;
```

b. 

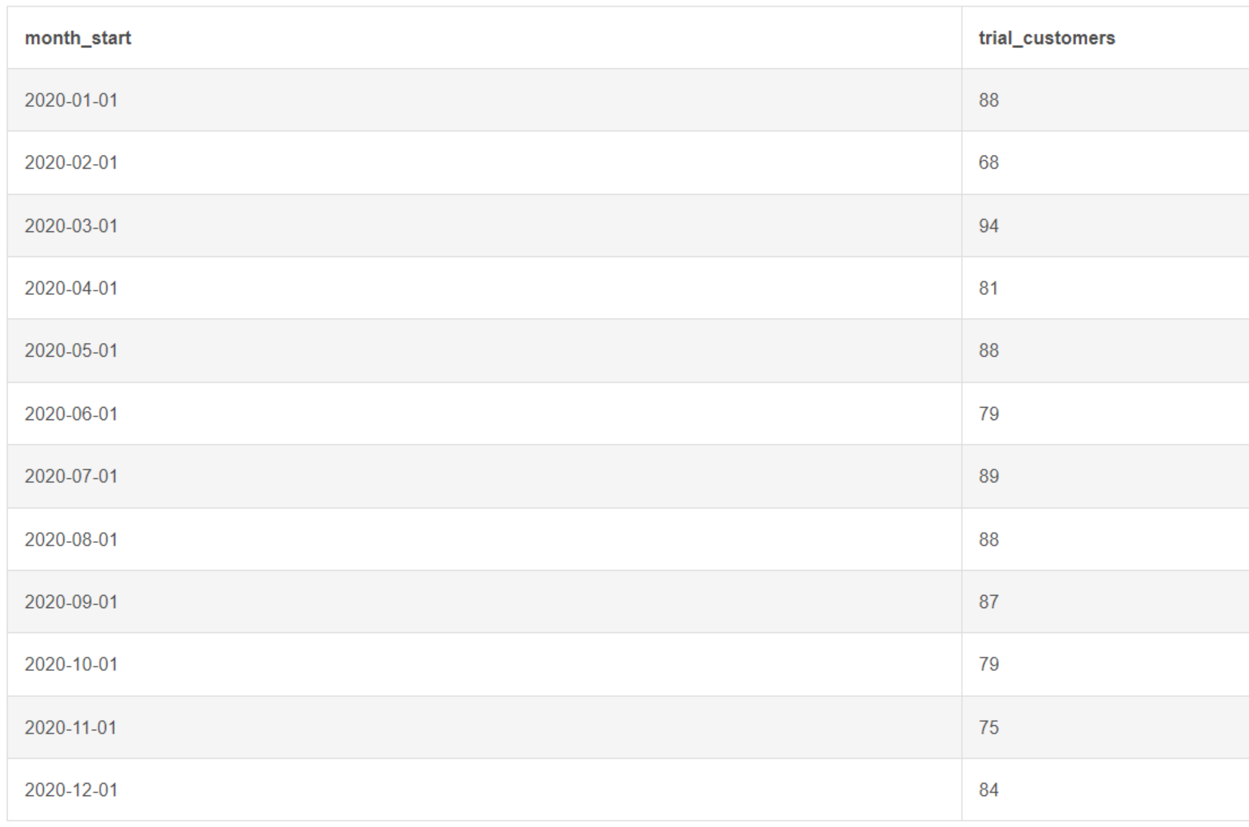
total_customers
1000

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

a. 

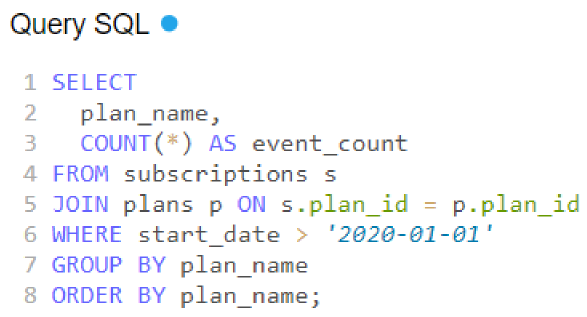
Query SQL ●

```sql
SELECT
  DATE_FORMAT(start_date, '%Y-%m-01') AS month_start,
  COUNT(*) AS trial_customers
FROM subscriptions
WHERE plan_id = 0
GROUP BY month_start
ORDER BY month_start;
```

b. 

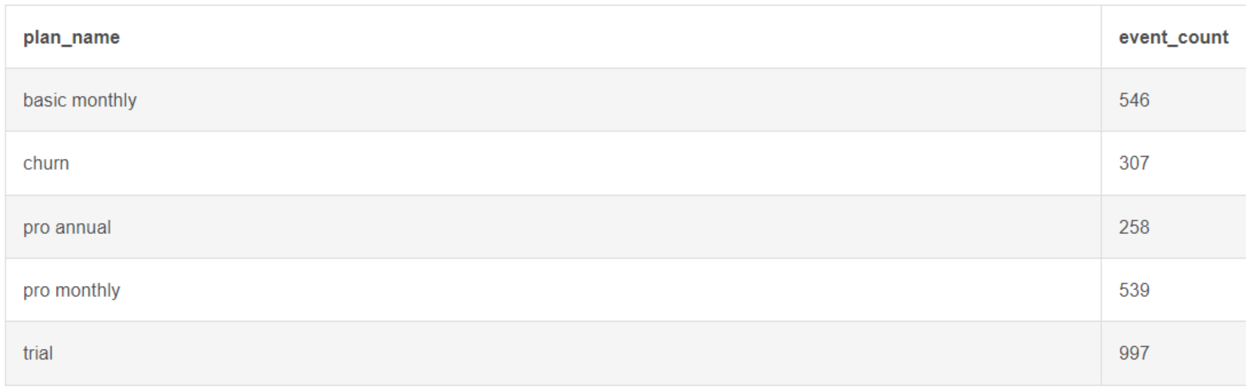
month_start	trial_customers
2020-01-01	88
2020-02-01	68
2020-03-01	94
2020-04-01	81
2020-05-01	88
2020-06-01	79
2020-07-01	89
2020-08-01	88
2020-09-01	87
2020-10-01	79
2020-11-01	75
2020-12-01	84

III. What plan start_date values occur after the year 2020 for our dataset? Show the breakdown by count of events for each plan_name?

a. 

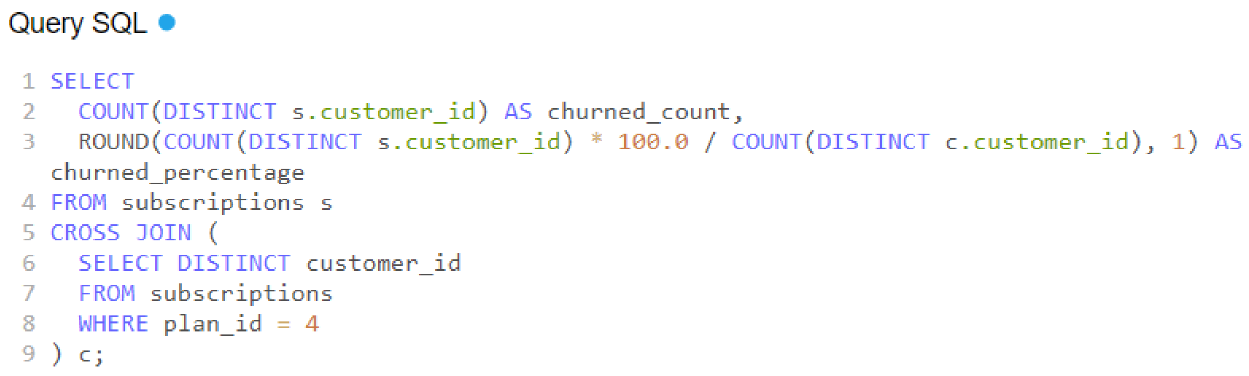
Query SQL

```sql
SELECT
  plan_name,
  COUNT(*) AS event_count
FROM subscriptions s
JOIN plans p ON s.plan_id = p.plan_id
WHERE start_date > '2020-01-01'
GROUP BY plan_name
ORDER BY plan_name;
```

b. 

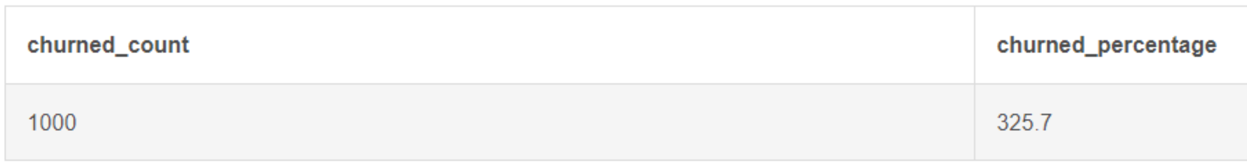
plan_name	event_count
basic monthly	546
churn	307
pro annual	258
pro monthly	539
trial	997

IV. What is the customer count and percentage of customers who have churned rounded to 1 decimal place?

a. 

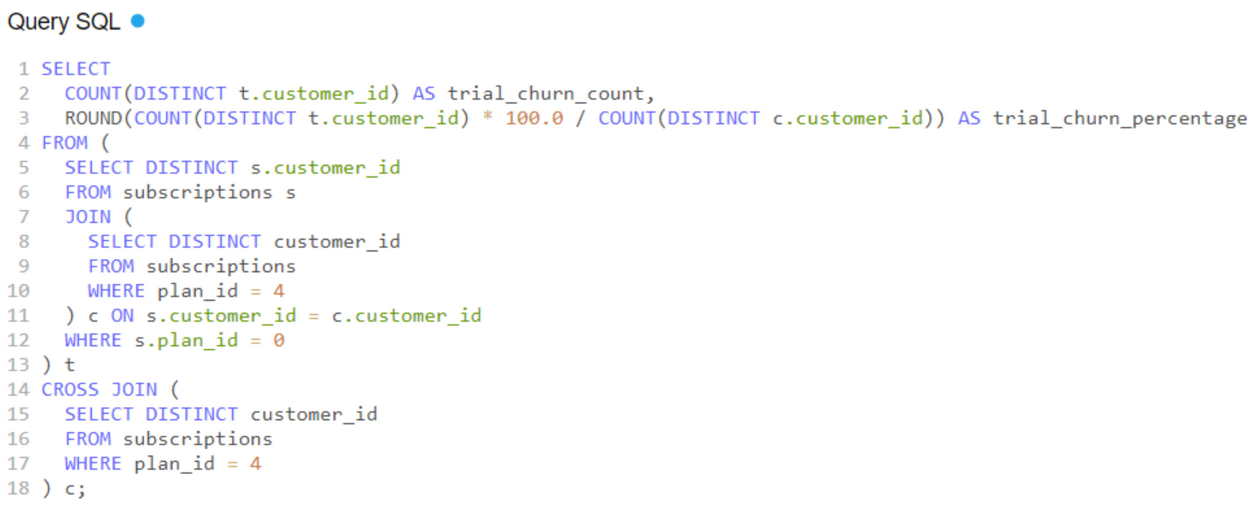
Query SQL

```sql
SELECT
  COUNT(DISTINCT s.customer_id) AS churned_count,
  ROUND(COUNT(DISTINCT s.customer_id) * 100.0 / COUNT(DISTINCT c.customer_id), 1) AS
churned_percentage
FROM subscriptions s
CROSS JOIN (
  SELECT DISTINCT customer_id
  FROM subscriptions
  WHERE plan_id = 4
) c;
```

b. 

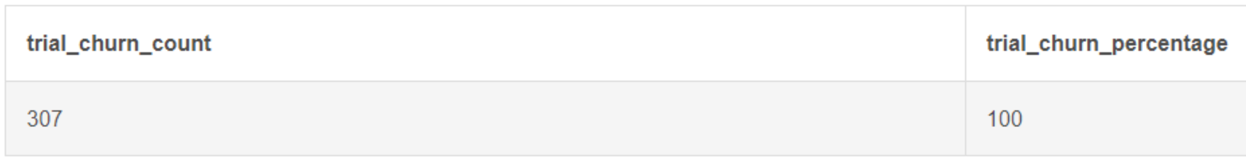
churned_count	churned_percentage
1000	325.7

V. How many customers have churned straight after their initial free trial - what percentage is this rounded to the nearest whole number?


Query SQL ●

```sql
SELECT
  COUNT(DISTINCT t.customer_id) AS trial_churn_count,
  ROUND(COUNT(DISTINCT t.customer_id) * 100.0 / COUNT(DISTINCT c.customer_id)) AS trial_churn_percentage
FROM (
  SELECT DISTINCT s.customer_id
  FROM subscriptions s
  JOIN (
    SELECT DISTINCT customer_id
    FROM subscriptions
    WHERE plan_id = 4
  ) c ON s.customer_id = c.customer_id
  WHERE s.plan_id = 0
) t
CROSS JOIN (
  SELECT DISTINCT customer_id
  FROM subscriptions
  WHERE plan_id = 4
) c;
```

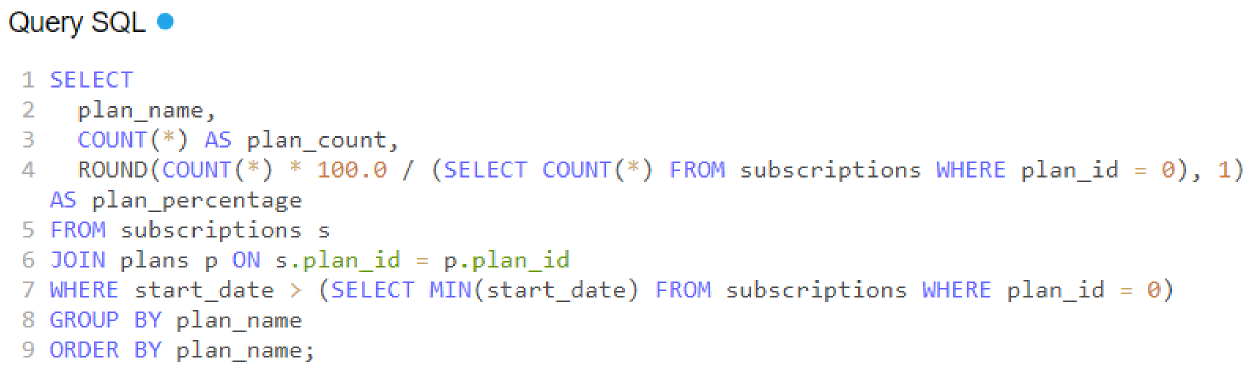
a.



trial_churn_count	trial_churn_percentage
307	100

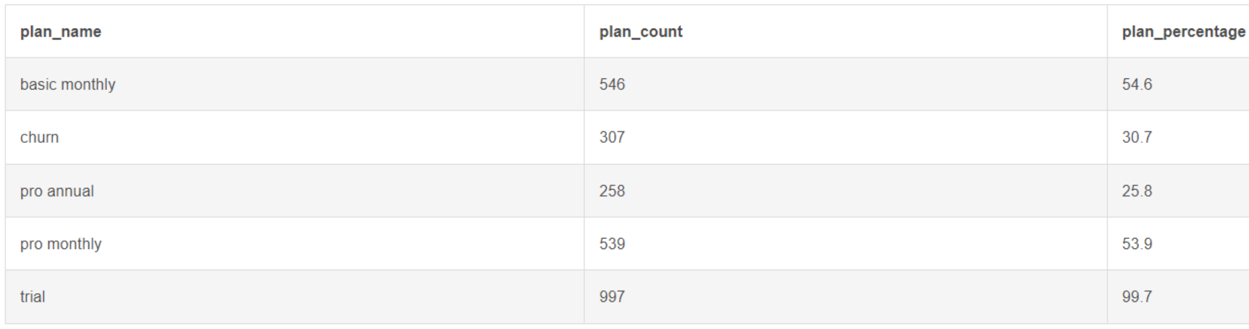
b.

VI. What is the number and percentage of customer plans after their initial free trial?


Query SQL ●

```sql
SELECT
  plan_name,
  COUNT(*) AS plan_count,
  ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM subscriptions WHERE plan_id = 0), 1)
AS plan_percentage
FROM subscriptions s
JOIN plans p ON s.plan_id = p.plan_id
WHERE start_date > (SELECT MIN(start_date) FROM subscriptions WHERE plan_id = 0)
GROUP BY plan_name
ORDER BY plan_name;
```

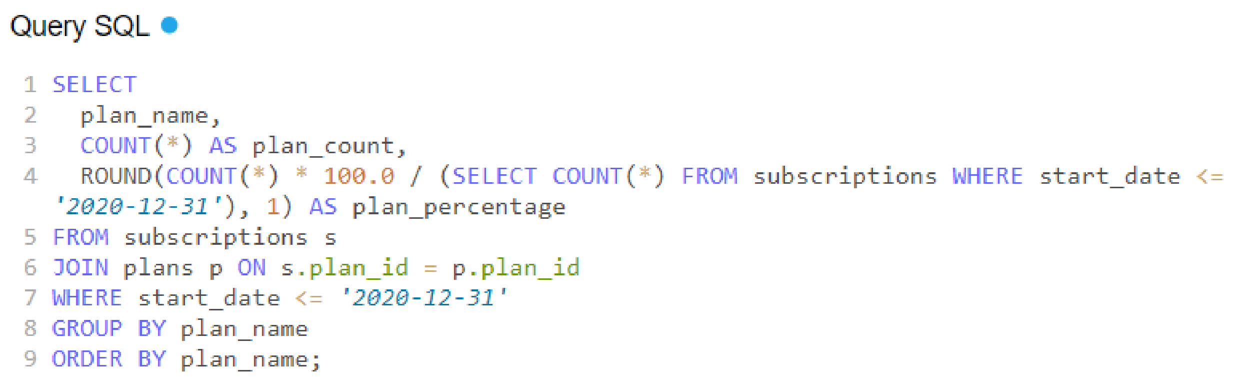
a.



plan_name	plan_count	plan_percentage
basic monthly	546	54.6
churn	307	30.7
pro annual	258	25.8
pro monthly	539	53.9
trial	997	99.7

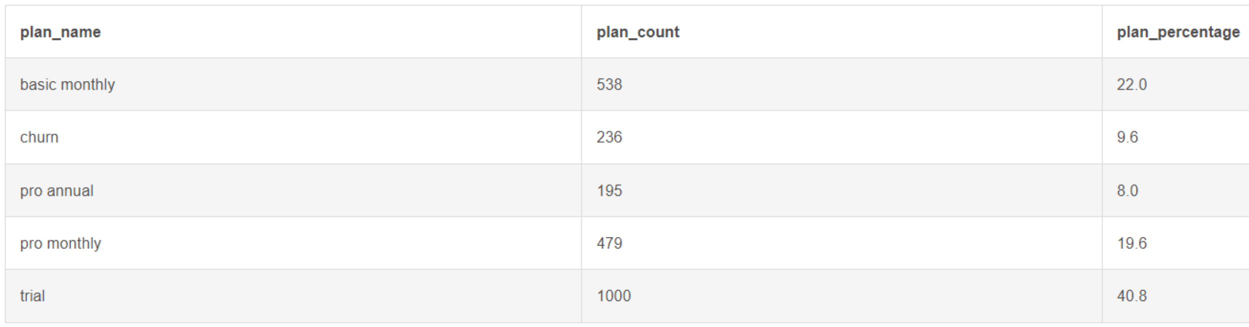
b.

VII. What is the customer count and percentage breakdown of all 5 plan_name values at 2020-12-31?

a. 

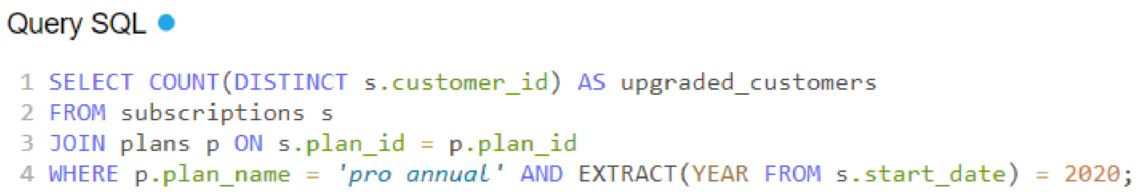
Query SQL ●

```sql
SELECT
  plan_name,
  COUNT(*) AS plan_count,
  ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM subscriptions WHERE start_date <=
'2020-12-31'), 1) AS plan_percentage
FROM subscriptions s
JOIN plans p ON s.plan_id = p.plan_id
WHERE start_date <= '2020-12-31'
GROUP BY plan_name
ORDER BY plan_name;
```

b. 

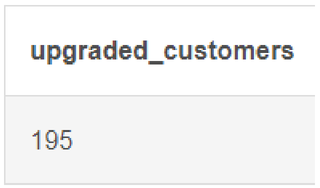
plan_name	plan_count	plan_percentage
basic monthly	538	22.0
churn	236	9.6
pro annual	195	8.0
pro monthly	479	19.6
trial	1000	40.8

VIII. How many customers have upgraded to an annual plan in 2020?

a. 

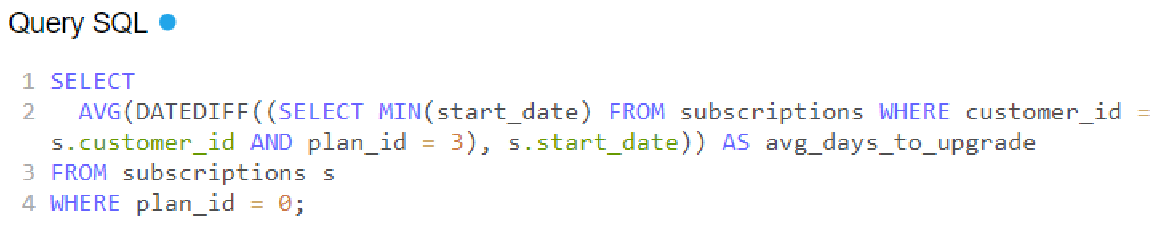
Query SQL ●

```sql
SELECT COUNT(DISTINCT s.customer_id) AS upgraded_customers
FROM subscriptions s
JOIN plans p ON s.plan_id = p.plan_id
WHERE p.plan_name = 'pro annual' AND EXTRACT(YEAR FROM s.start_date) = 2020;
```

b. 

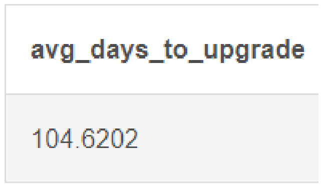
upgraded_customers
195

IX. How many days on average does it take for a customer to an annual plan from the day they join Foodie-Fi?

a. 

Query SQL ●

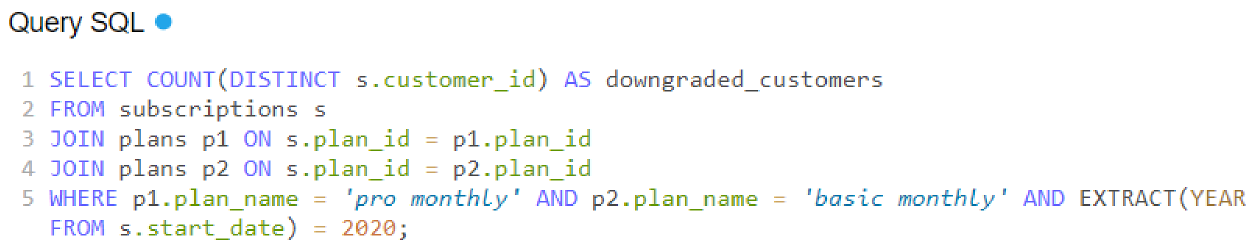
```sql
SELECT
  AVG(DATEDIFF((SELECT MIN(start_date) FROM subscriptions WHERE customer_id =
s.customer_id AND plan_id = 3), s.start_date)) AS avg_days_to_upgrade
FROM subscriptions s
WHERE plan_id = 0;
```

b. 

avg_days_to_upgrade
104.6202

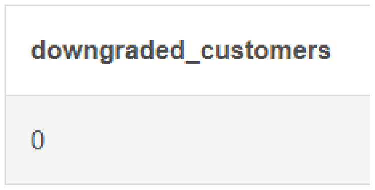
X. How many customers downgraded from a pro monthly to a basic monthly plan in 2020?

a.


Query SQL ●

```sql
SELECT COUNT(DISTINCT s.customer_id) AS downgraded_customers
FROM subscriptions s
JOIN plans p1 ON s.plan_id = p1.plan_id
JOIN plans p2 ON s.plan_id = p2.plan_id
WHERE p1.plan_name = 'pro monthly' AND p2.plan_name = 'basic monthly' AND EXTRACT(YEAR
FROM s.start_date) = 2020;
```

b.



downgraded_customers
0