/\*

### A. Customer Journey

***Based off the 8 sample customers provided in the sample from the subscriptions table, write a brief description about each customer’s onboarding journey.***

***Try to keep it as short as possible - you may also want to run some sort of join to make your explanations a bit easier!***

\*/

SELECT

s.customer\_id,

p.plan\_name,

p.price,

s.start\_date

FROM subscriptions s

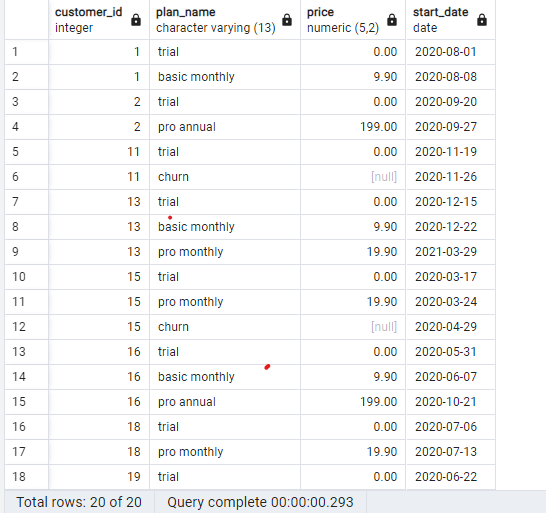
JOIN plans p

ON s.plan\_id = p.plan\_id

WHERE s.customer\_id<=8

ORDER BY s.customer\_id, s.start\_date;

**RESULT:**



brief descriptions of each customer's onboarding journey:

Customer 1:

Journey: Started with a trial on August 1, 2020, and upgraded to the Basic Monthly plan on August 8, 2020.

Customer 2:

Journey: Began with a trial on September 20, 2020, and quickly switched to the Pro Annual plan on September 27, 2020.

Customer 3:

Journey: Initiated with a trial on January 13, 2020, and moved to Basic Monthly a week later, on January 20, 2020.

Customer 4:

Journey: Started a trial on January 17, 2020, upgraded to Basic Monthly on January 24, 2020, but churned on April 21, 2020.

Customer 5:

Journey: Began with a trial on August 3, 2020, and switched to the Basic Monthly plan a week later, on August 10, 2020.

Customer 6:

Journey: Started a trial on December 23, 2020, upgraded to Basic Monthly on December 30, 2020, and churned on February 26, 2021.

Customer 7:

Journey: Began with a trial on February 5, 2020, quickly moved to Basic Monthly on February 12, 2020, and later upgraded to Pro Monthly on May 22, 2020.

Customer 8:

Journey: Started a trial on June 11, 2020, moved to Basic Monthly on June 18, 2020, and upgraded to Pro Monthly on August 3, 2020.

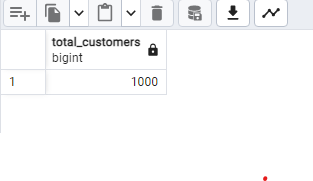
**B. Data Analysis Questions**

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

SELECT COUNT(DISTINCT customer\_id) AS total\_customers

FROM subscriptions;

**RESULT:**



There were 1000 customers

***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\_TRUNC('month', start\_date) AS month\_start,

COUNT(\*) AS trial\_count

FROM

subscriptions

WHERE

plan\_id = 0 -- Filter for trial plan (plan\_id = 0)

GROUP BY

DATE\_TRUNC('month', start\_date)

ORDER BY

month\_start;

**RESULT:**



***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

p.plan\_name,

COUNT(\*) AS start\_count

FROM subscriptions s

JOIN plans p

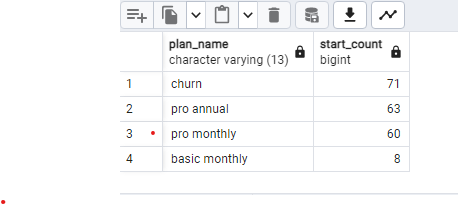
ON s.plan\_id = p.plan\_id

WHERE EXTRACT(YEAR FROM start\_date)>2020 --OR WHERE start\_date>'2020-12-31'

GROUP BY p.plan\_name

ORDER BY start\_count DESC;

**RESULT:**



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

WITH total\_customers AS (

SELECT COUNT(DISTINCT customer\_id) AS total\_count

FROM subscriptions

),

churned\_customers AS (

SELECT COUNT(DISTINCT customer\_id) AS churned\_count

FROM subscriptions

WHERE plan\_id = 4

)

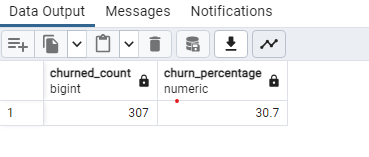
SELECT

c.churned\_count,

ROUND((c.churned\_count::decimal / t.total\_count) \* 100, 1) AS churn\_percentage

FROM churned\_customers c, total\_customers t;

**RESULT**



Out of the total customer base of Foodie-Fi, 307 customers have churned. This represents approximately 30.7% of the overall customer count

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

WITH plan\_cte AS

(

SELECT customer\_id,

plan\_name,

ROW\_NUMBER()OVER (PARTITION BY customer\_id ORDER BY start\_date)AS rnk

FROM subscriptions s

INNER JOIN plans p

ON s.plan\_id=p.plan\_id

)

SELECT COUNT(DISTINCT customer\_id) AS churned\_after\_trial,

ROUND(100.0\*

COUNT(DISTINCT customer\_id)/

(SELECT COUNT(DISTINCT customer\_id )FROM subscriptions)

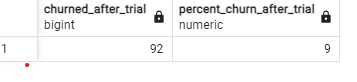
) percent\_churn\_after\_trial

FROM plan\_cte

WHERE rnk=2

AND plan\_name='churn';

**RESULT:**



A total of 92 customers churned immediately after the initial free trial period, representing approximately 9% of the entire customer base.

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

Sol1:

WITH plan\_cte AS

(

SELECT s.customer\_id,s.plan\_id,

p.plan\_name,

ROW\_NUMBER()OVER (PARTITION BY s.customer\_id ORDER BY s.start\_date)AS rnk

FROM subscriptions s

INNER JOIN plans p

ON s.plan\_id=p.plan\_id

)

SELECT

plan\_id,plan\_name,

COUNT(DISTINCT customer\_id) convert\_count,

ROUND(100\*

COUNT(customer\_id)::NUMERIC/

(SELECT COUNT(DISTINCT customer\_id )FROM subscriptions),1

) convert\_percent

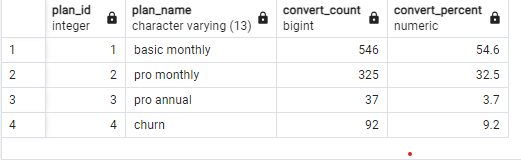
FROM plan\_cte

WHERE rnk=2

GROUP BY plan\_name,plan\_id

ORDER BY plan\_id;

**RESULT:**



Sol2:using LEAD()

WITH next\_plans AS (

SELECT

customer\_id,

plan\_id,

LEAD(plan\_id) OVER(

PARTITION BY customer\_id

ORDER BY plan\_id) as next\_plan\_id

FROM subscriptions

)

SELECT

n.next\_plan\_id ,

COUNT(n.customer\_id) AS converted\_customers,

ROUND(100 \*

COUNT(n.customer\_id)::NUMERIC

/ (SELECT COUNT(DISTINCT customer\_id)

FROM subscriptions)

,1) AS conversion\_percentage

FROM next\_plans n

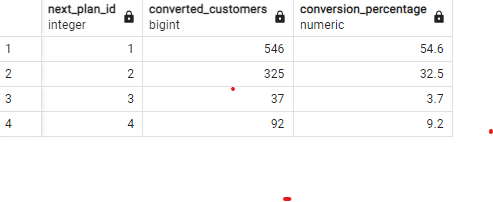
WHERE next\_plan\_id IS NOT NULL

AND plan\_id=0

GROUP BY n.plan\_id,next\_plan\_id

ORDER BY next\_plan\_id;

**RESULT:**



* More than 80% of Foodie-Fi's customers are on paid plans with a majority opting for Plans 1 and 2.
* There is potential for improvement in customer acquisition for Plan 3 as only a small percentage of customers are choosing this higher-priced plan.

***7. What is the customer count and percentage breakdown of all 5 plan\_name values at 2020-12-31?***

WITH CTE AS (

SELECT \*

,ROW\_NUMBER() OVER(PARTITION BY customer\_id ORDER BY start\_date DESC) as rn

FROM subscriptions

WHERE start\_date <= '2020-12-31'

)

SELECT

cte.plan\_id,

plan\_name,

COUNT(customer\_id) as customer\_count,

ROUND((COUNT(customer\_id)::NUMERIC/(SELECT COUNT(DISTINCT customer\_id) FROM CTE))\*100,1) as percent\_of\_customers

FROM CTE

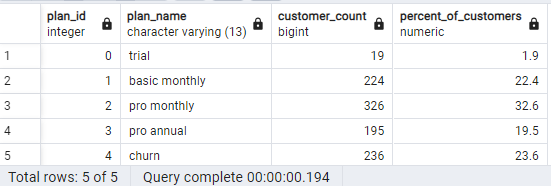
INNER JOIN plans as P on CTE.plan\_id = P.plan\_id

WHERE rn = 1

GROUP BY cte.plan\_id,plan\_name

ORDER BY cte.plan\_id;

**RESULT:**



Sol2:

WITH next\_dates AS (

SELECT

customer\_id,

plan\_id,

start\_date,

LEAD(start\_date) OVER (

PARTITION BY customer\_id

ORDER BY start\_date

) AS next\_date

FROM subscriptions

WHERE start\_date <= '2020-12-31'

)

SELECT

plan\_id,

COUNT(DISTINCT customer\_id) AS customers,

ROUND(100.0 \*

COUNT(DISTINCT customer\_id)

/ (SELECT COUNT(DISTINCT customer\_id)

FROM subscriptions)

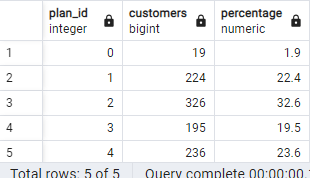
,1) AS percentage

FROM next\_dates

WHERE next\_date IS NULL

GROUP BY plan\_id;

RESULT:



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

SELECT p.plan\_name,COUNT(s.customer\_id)AS count\_convert\_to\_annual

FROM subscriptions s

JOIN plans p

USING(plan\_id)

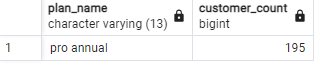
WHERE plan\_id=3

AND

EXTRACT(YEAR FROM start\_date)=2020

GROUP BY p.plan\_name;

**RESULT:**



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

WITH trial\_cte AS

(

SELECT customer\_id,start\_date AS trial\_date

FROM subscriptions

WHERE plan\_id=0

),

annual\_cte AS

(

SELECT customer\_id,start\_date AS annual\_date

FROM subscriptions

WHERE plan\_id=3

)

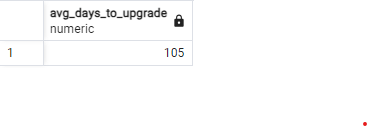
SELECT

ROUND(AVG(a.annual\_date-t.trial\_date),0) AS avg\_days\_to\_upgrade

FROM trial\_cte t JOIN annual\_cte a

USING(customer\_id);

**RESULT:**

On average, customers take approximately 105 days from the day they join Foodie-Fi to upgrade to an annual plan.

***10. Can you further breakdown this average value into 30 day periods (i.e. 0-30 days, 31-60 days etc)***

-- Sol1

WITH trial\_cte AS

(

SELECT

customer\_id,

MIN(start\_date) AS trial\_start

FROM subscriptions

WHERE plan\_id = 0

GROUP BY customer\_id

), annual\_cte AS

(

SELECT

s.customer\_id,

s.start\_date AS annual\_start,

(s.start\_date - t.trial\_start) AS days\_to\_annual -- Calculate days between trial and annual plan

FROM subscriptions s

JOIN trial\_cte t

ON s.customer\_id = t.customer\_id

WHERE s.plan\_id = 3

)

SELECT

(CASE

WHEN days\_to\_annual <= 30 THEN 1

WHEN days\_to\_annual <= 60 THEN 2

WHEN days\_to\_annual <= 90 THEN 3

WHEN days\_to\_annual <= 120 THEN 4

WHEN days\_to\_annual <= 150 THEN 5

WHEN days\_to\_annual <= 180 THEN 6

WHEN days\_to\_annual <= 210 THEN 7

WHEN days\_to\_annual <= 240 THEN 8

WHEN days\_to\_annual <= 270 THEN 9

WHEN days\_to\_annual <= 300 THEN 10

WHEN days\_to\_annual <= 330 THEN 11

WHEN days\_to\_annual <= 360 THEN 12

ELSE 13

END) AS sort\_order,

(CASE

WHEN days\_to\_annual <= 30 THEN '0-30'

WHEN days\_to\_annual <= 60 THEN '31-60'

WHEN days\_to\_annual <= 90 THEN '61-90'

WHEN days\_to\_annual <= 120 THEN '91-120'

WHEN days\_to\_annual <= 150 THEN '121-150'

WHEN days\_to\_annual <= 180 THEN '151-180'

WHEN days\_to\_annual <= 210 THEN '181-210'

WHEN days\_to\_annual <= 240 THEN '211-240'

WHEN days\_to\_annual <= 270 THEN '241-270'

WHEN days\_to\_annual <= 300 THEN '271-300'

WHEN days\_to\_annual <= 330 THEN '301-330'

WHEN days\_to\_annual <= 360 THEN '331-360'

ELSE '360+' -- Handles cases where days are more than 360

END) AS bin,

COUNT(customer\_id) AS customer\_count,

ROUND(AVG(days\_to\_annual), 1) AS average\_days\_to\_annual

FROM

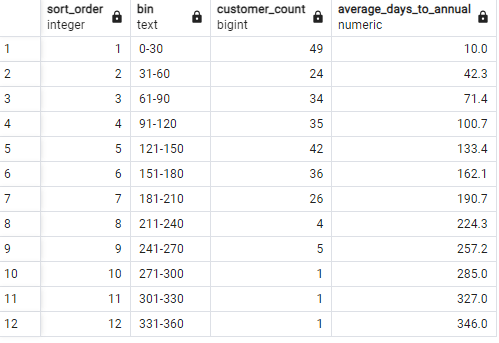
annual\_cte

GROUP BY bin,sort\_order

ORDER BY sort\_order

;

RESULT:



Sol2:

WITH trial\_cte AS

(

SELECT customer\_id,

MIN(start\_date) AS trial\_start\_date

FROM subscriptions s

WHERE plan\_id=0

GROUP BY customer\_id

),

annual\_cte AS

(

SELECT s.customer\_id,

s.start\_date AS annual\_start\_date,

(s.start\_date-t.trial\_start\_date) AS days\_to\_upgrade

FROM subscriptions s

JOIN trial\_cte t

USING(customer\_id)

WHERE plan\_id=3

)

SELECT

WIDTH\_BUCKET(days\_to\_upgrade,0,365,12) AS bucket,

((WIDTH\_BUCKET(days\_to\_upgrade,0,365,12)-1)\*30+1||' - '||WIDTH\_BUCKET(days\_to\_upgrade,0,365,12)\*30) AS bin,

COUNT(\*) AS customer\_count,

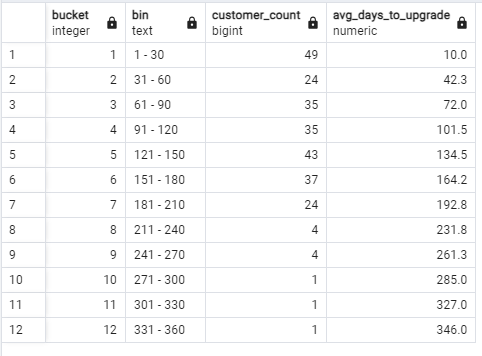
ROUND(AVG(days\_to\_upgrade),1) AS avg\_days\_to\_upgrade

FROM annual\_cte a

GROUP BY bin,bucket

ORDER BY Bucket ;

**RESULT:**



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

--Sol1:

WITH pro\_monthly\_cte As

(

SELECT

customer\_id,

start\_date AS pro\_month\_start

FROM subscriptions

WHERE plan\_id=2

),

basic\_monthly\_cte As

(

SELECT

customer\_id,

start\_date AS basic\_month\_start

FROM subscriptions

WHERE plan\_id=1

)

SELECT

COUNT(DISTINCT p.customer\_id) downgrade\_count

FROM

pro\_monthly\_cte p JOIN basic\_monthly\_cte b

USING(customer\_id)

WHERE

p.pro\_month\_start<b.basic\_month\_start

AND EXTRACT(YEAR FROM b.basic\_month\_start)=2020;

**RESULT:**



--sol2:

WITH pro\_monthly\_to\_basic AS (

-- Find customers who switched from pro monthly to basic monthly

SELECT

s1.customer\_id,

s1.start\_date AS pro\_monthly\_start,

s2.start\_date AS basic\_monthly\_start

FROM subscriptions s1

JOIN subscriptions s2

ON s1.customer\_id = s2.customer\_id

WHERE

s1.plan\_id = 2 -- Pro monthly

AND s2.plan\_id = 1 -- Basic monthly

AND s1.start\_date < s2.start\_date -- Downgrade event: pro monthly must occur before basic monthly

AND EXTRACT(YEAR FROM s2.start\_date) = 2020 -- Limit to downgrades in 2020

)

SELECT

COUNT(DISTINCT customer\_id) AS downgrade\_count

FROM pro\_monthly\_to\_basic;

**RESULT:**



-- sol3

WITH pro\_to\_basic\_cte AS (

SELECT

s.customer\_id,

p.plan\_id,

p.plan\_name,

LEAD(p.plan\_id) OVER (

PARTITION BY s.customer\_id

ORDER BY s.start\_date) AS next\_plan\_id

FROM subscriptions AS s

JOIN plans AS p

ON s.plan\_id = p.plan\_id

WHERE EXTRACT(YEAR FROM s.start\_date) = 2020

)

SELECT

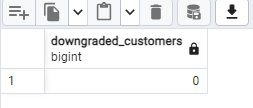
COUNT(customer\_id) AS downgraded\_customers

FROM pro\_to\_basic\_cte

WHERE plan\_id = 2

AND next\_plan\_id = 1;

RESULT:



In 2020, there were no instances where customers downgraded from a pro monthly plan to a basic monthly plan.

**/\*C. Challenge Payment Question**

***The Foodie-Fi team wants you to create a new payments table for the year 2020***

***that includes amounts paid by each customer in the subscriptions table with***

***the following requirements:***

***monthly payments always occur on the same day of month as the original start\_date of any monthly paid plan***

***upgrades from basic to monthly or pro plans are reduced by the current paid amount in that month and start immediately***

***upgrades from pro monthly to pro annual are paid at the end of the current billing***

***period and also starts at the end of the month period once a customer churns they will no longer make payments***

***Example outputs for this table might look like the following:***

***customer\_id plan\_id plan\_name payment\_date amount payment\_order\*/***

-- Create the payments table for 2020

WITH payment\_schedule AS -- Identify all subscription changes for customers in 2020

(

SELECT s.customer\_id,

s.plan\_id,

p.plan\_name,

p.price,

s.start\_date,

LEAD(s.start\_date) OVER (PARTITION BY s.customer\_id ORDER BY s.start\_date) AS next\_plan\_start\_date,

p.plan\_name='churn' AS has\_churned

FROM subscriptions s

JOIN plans p

USING(plan\_id)

WHERE EXTRACT(YEAR FROM s.start\_date)=2020

),

payment\_calender AS -- Generate the payment calendar for all customers, taking into account their plan changes

(

SELECT

ps.customer\_id,

ps.plan\_id,

ps.plan\_name,

ps.start\_date AS payment\_date,

ps.price AS amount,

ROW\_NUMBER() OVER(PARTITION BY ps.customer\_id ORDER BY ps.start\_date) AS payment\_order

FROM payment\_schedule ps

WHERE NOT ps.has\_churned --exclude churned

UNION ALL -- Handle monthly payments until the customer changes plans or churns

SELECT ps.customer\_id,

ps.plan\_id,

ps.plan\_name,

ps.start\_date + INTERVAL '1 month' \* generate\_series(1, CAST(DATE\_PART('month', AGE(COALESCE(next\_plan\_start\_date, '2020-12-31'), ps.start\_date)) AS INTEGER)) AS payment\_date,

ps.price AS amount,

ROW\_NUMBER() OVER(PARTITION BY ps.customer\_id ORDER BY ps.start\_date + INTERVAL '1 month' \* generate\_series(1, CAST(DATE\_PART('month', AGE(COALESCE(next\_plan\_start\_date, '2020-12-31'), ps.start\_date)) AS INTEGER)))AS payment\_order

FROM payment\_schedule ps

WHERE ps.plan\_name IN('basix monthly','pro monthly')

)

-- create the payments table for 2020

SELECT

customer\_id,

plan\_id,

plan\_name,

payment\_date,

amount,

payment\_order

FROM payment\_calender

WHERE

EXTRACT(YEAR FROM payment\_date) = 2020

ORDER BY customer\_id, payment\_order;

**RESULT:**

