

/*

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:

	customer_id integer	plan_name character varying (13)	price numeric (5,2)	start_date date
1	1	trial	0.00	2020-08-01
2	1	basic monthly	9.90	2020-08-08
3	2	trial	0.00	2020-09-20
4	2	pro annual	199.00	2020-09-27
5	11	trial	0.00	2020-11-19
6	11	churn	[null]	2020-11-26
7	13	trial	0.00	2020-12-15
8	13	basic monthly	9.90	2020-12-22
9	13	pro monthly	19.90	2021-03-29
10	15	trial	0.00	2020-03-17
11	15	pro monthly	19.90	2020-03-24
12	15	churn	[null]	2020-04-29
13	16	trial	0.00	2020-05-31
14	16	basic monthly	9.90	2020-06-07
15	16	pro annual	199.00	2020-10-21
16	18	trial	0.00	2020-07-06
17	18	pro monthly	19.90	2020-07-13
18	19	trial	0.00	2020-06-22

Total rows: 20 of 20

Query complete 00:00:00.293

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.

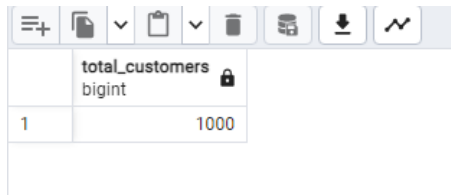


B. Data Analysis Questions

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

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

RESULT:



	total_customers bigint
1	1000

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:

	month_start timestamp with time zone	trial_count bigint
1	2020-01-01 00:00:00-08	88
2	2020-02-01 00:00:00-08	68
3	2020-03-01 00:00:00-08	94
4	2020-04-01 00:00:00-07	81
5	2020-05-01 00:00:00-07	88
6	2020-06-01 00:00:00-07	79
7	2020-07-01 00:00:00-07	89
8	2020-08-01 00:00:00-07	88
9	2020-09-01 00:00:00-07	87
10	2020-10-01 00:00:00-07	79
11	2020-11-01 00:00:00-07	75
12	2020-12-01 00:00:00-08	84

Total rows: 12 of 12 Query complete 00:00:01.120

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:

	plan_name character varying (13)	start_count bigint
1	churn	71
2	pro annual	63
3	pro monthly	60
4	basic monthly	8

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

Data Output				Messages	Notifications
	churned_count bigint		churn_percentage numeric		
1	307		30.7		

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:

	churned_after_trial bigint	percent_churn_after_trial numeric
1	92	9

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:

	plan_id integer	plan_name character varying (13)	convert_count bigint	convert_percent numeric
1	1	basic monthly	546	54.6
2	2	pro monthly	325	32.5
3	3	pro annual	37	3.7
4	4	churn	92	9.2

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:

	next_plan_id integer	converted_customers bigint	conversion_percentage numeric
1	1	546	54.6
2	2	325	32.5
3	3	37	3.7
4	4	92	9.2

- 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:

	plan_id integer	plan_name character varying (13)	customer_count bigint	percent_of_customers numeric
1	0	trial	19	1.9
2	1	basic monthly	224	22.4
3	2	pro monthly	326	32.6
4	3	pro annual	195	19.5
5	4	churn	236	23.6
Total rows: 5 of 5		Query complete 00:00:00.194		

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:

	plan_id integer	customers bigint	percentage numeric
1	0	19	1.9
2	1	224	22.4
3	2	326	32.6
4	3	195	19.5
5	4	236	23.6
Total rows: 5 of 5 Query complete 00:00:00			

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:

	plan_name character varying (13) 🔒	customer_count bigint 🔒
1	pro annual	195

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:

	avg_days_to_upgrade numeric
1	105

• 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:

	sort_order integer	bin text	customer_count bigint	average_days_to_annual numeric
1	1	0-30	49	10.0
2	2	31-60	24	42.3
3	3	61-90	34	71.4
4	4	91-120	35	100.7
5	5	121-150	42	133.4
6	6	151-180	36	162.1
7	7	181-210	26	190.7
8	8	211-240	4	224.3
9	9	241-270	5	257.2
10	10	271-300	1	285.0
11	11	301-330	1	327.0
12	12	331-360	1	346.0

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:

	bucket integer	bin text	customer_count bigint	avg_days_to_upgrade numeric
1	1	1 - 30	49	10.0
2	2	31 - 60	24	42.3
3	3	61 - 90	35	72.0
4	4	91 - 120	35	101.5
5	5	121 - 150	43	134.5
6	6	151 - 180	37	164.2
7	7	181 - 210	24	192.8
8	8	211 - 240	4	231.8
9	9	241 - 270	4	261.3
10	10	271 - 300	1	285.0
11	11	301 - 330	1	327.0
12	12	331 - 360	1	346.0

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:

	downgrade_count bigint
1	0

--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:

	downgrade_count bigint
1	0

-- 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:

downgraded_customers		
bigint		
1		0

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,

	customer_id integer 🔒	plan_id integer 🔒	plan_name character varying (13) 🔒	payment_date timestamp without time zone 🔒	amount numeric (5,2) 🔒	payment_order bigint 🔒
1	1	0	trial	2020-08-01 00:00:00	0.00	1
2	1	1	basic monthly	2020-08-08 00:00:00	9.90	2
3	2	0	trial	2020-09-20 00:00:00	0.00	1
4	2	3	pro annual	2020-09-27 00:00:00	199.00	2
5	3	0	trial	2020-01-13 00:00:00	0.00	1
6	3	1	basic monthly	2020-01-20 00:00:00	9.90	2
7	4	0	trial	2020-01-17 00:00:00	0.00	1
8	4	1	basic monthly	2020-01-24 00:00:00	9.90	2
9	5	0	trial	2020-08-03 00:00:00	0.00	1
10	5	1	basic monthly	2020-08-10 00:00:00	9.90	2
11	6	0	trial	2020-12-23 00:00:00	0.00	1
12	6	1	basic monthly	2020-12-30 00:00:00	9.90	2
13	7	2	pro monthly	2020-06-22 00:00:00	19.90	1
14	7	0	trial	2020-02-05 00:00:00	0.00	1
15	7	2	pro monthly	2020-07-22 00:00:00	19.90	2
16	7	1	basic monthly	2020-02-12 00:00:00	9.90	2
17	7	2	pro monthly	2020-05-22 00:00:00	19.90	3
18	7	2	pro monthly	2020-08-22 00:00:00	19.90	3
Total rows: 1000 of 4020		Query complete 00:00:00.324				