SQL – CASE STUDY CHALLENGE

FODDIE - FI

By – Deepak Londhe

INTRODUCTION

Subscription based businesses are **super popular** and Danny realised 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 also short descriptions on each of the database tables - our case study focuses on only 2 tables but there will be a challenge to create a new table for the Foodie-Fi team.

All datasets exist within the foodie fi database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

Entity Relationship Diagram

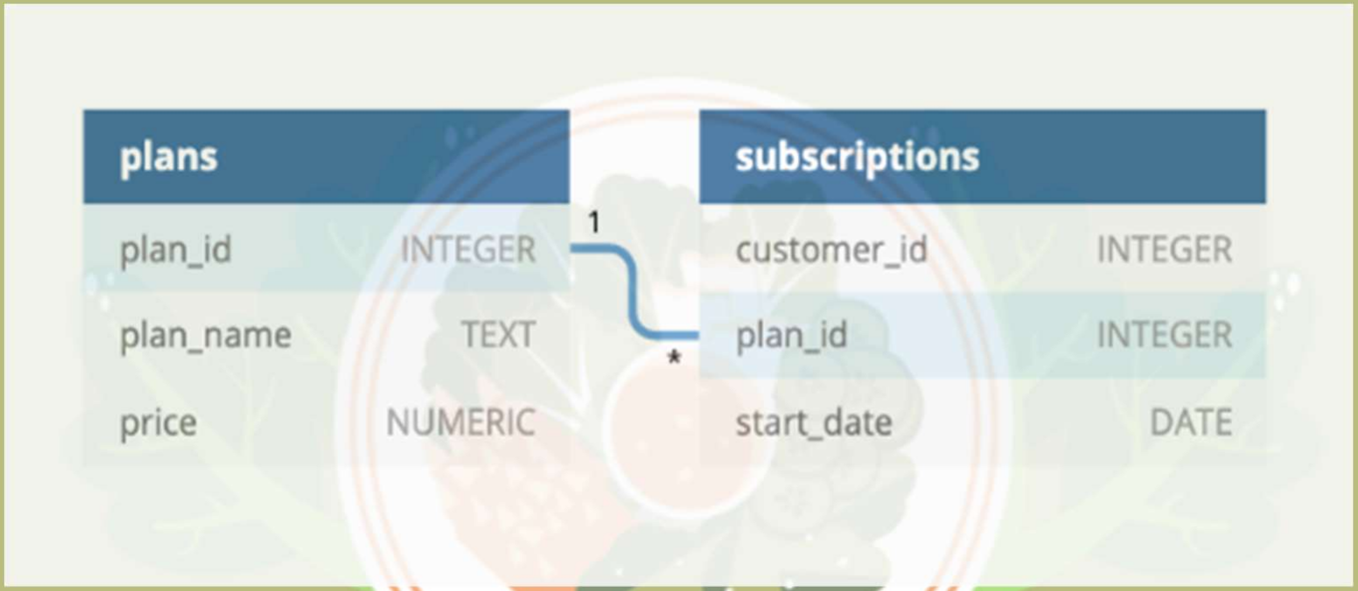


Table – 1: Plans

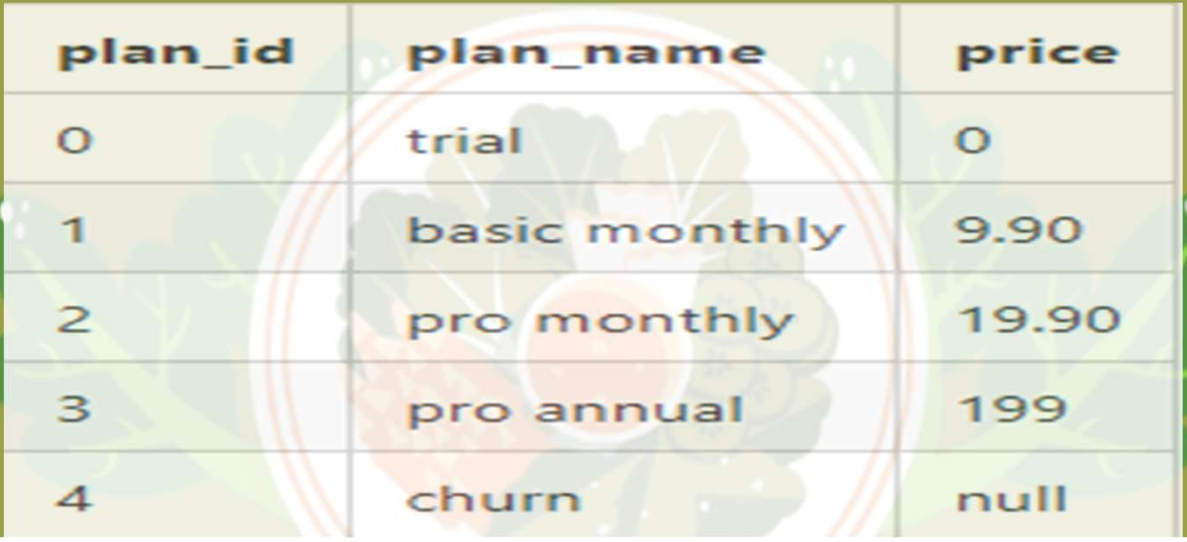
Customers can choose which plans to join Foodie-Fi when they first sign up.

Basic plan customers have limited access and can only stream their videos and is 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 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

Table – 2: Subscription

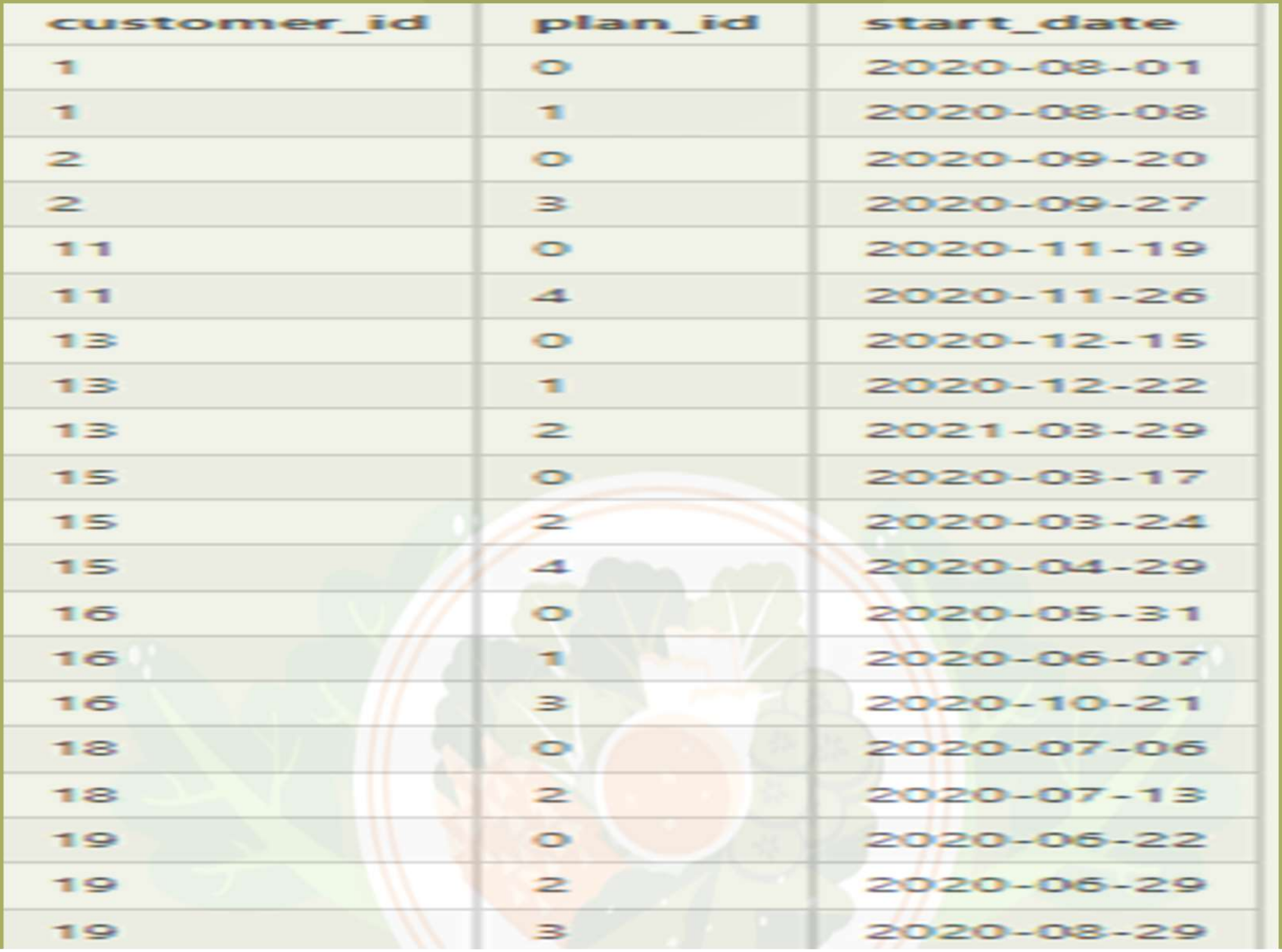
Customer subscriptions show the exact date where their specific plan id starts.

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 straightaway.

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

Table – 2: Subscription Table

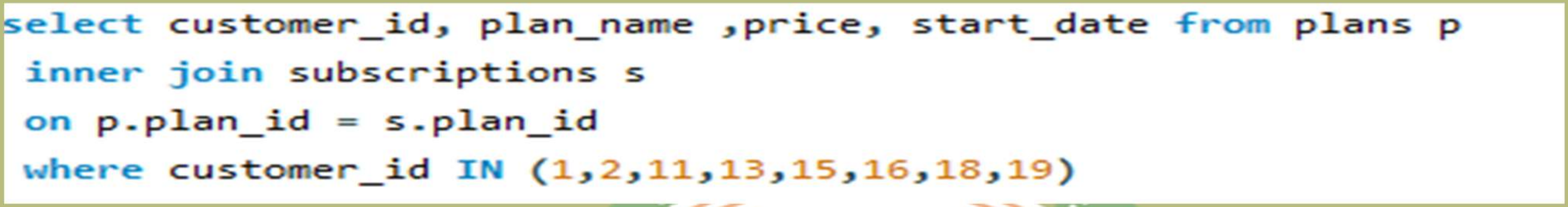


customer_id	plan_id	start_date
1	0	2020-08-01
1	1	2020-08-08
2	0	2020-09-20
2	3	2020-09-27
11	0	2020-11-19
11	4	2020-11-26
13	0	2020-12-15
13	1	2020-12-22
13	2	2021-03-29
15	0	2020-03-17
15	2	2020-03-24
15	4	2020-04-29
16	0	2020-05-31
16	1	2020-06-07
16	3	2020-10-21
18	0	2020-07-06
18	2	2020-07-13
19	0	2020-06-22
19	2	2020-06-29
19	3	2020-08-29

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.

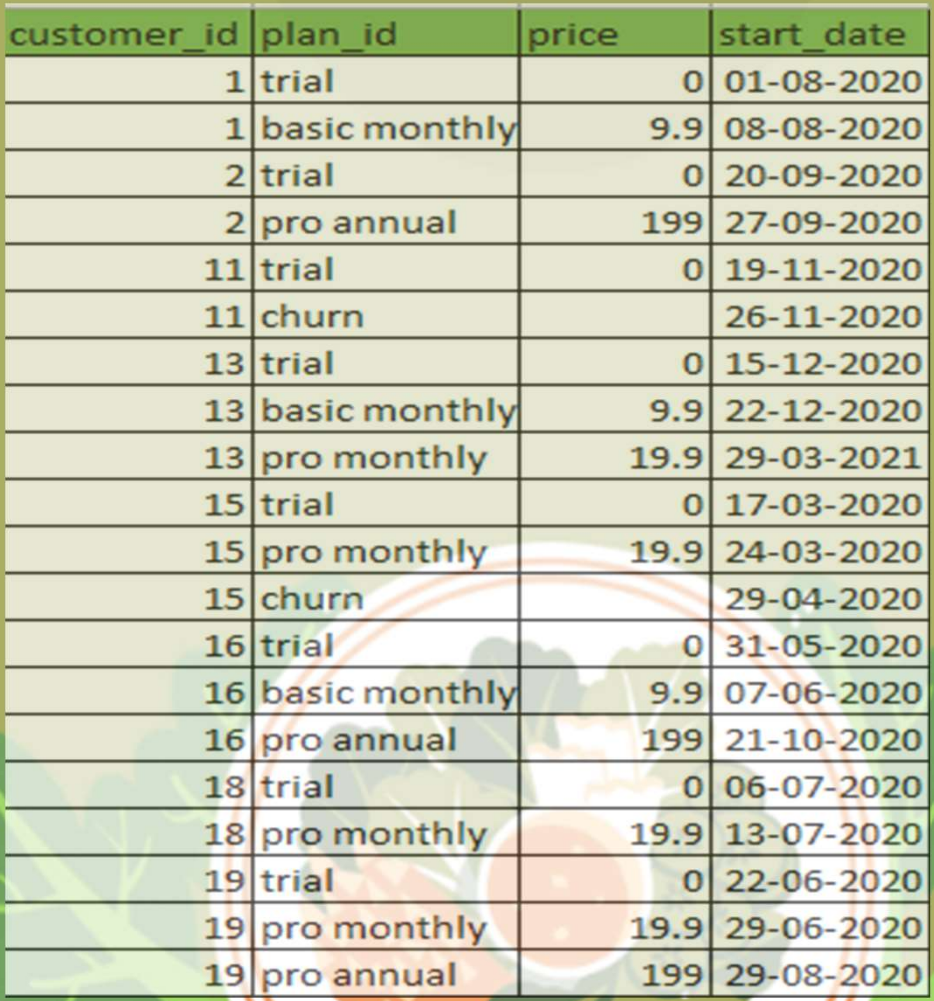
Approach – To get the description on customers journey . I had joined the PLANS table with SUBSCRIPTION table .



```
select customer_id, plan_name ,price, start_date from plans p
 inner join subscriptions s
 on p.plan_id = s.plan_id
 where customer_id IN (1,2,11,13,15,16,18,19)
```

Customer Journey

Query output



customer_id	plan_id	price	start_date
1	trial	0	01-08-2020
1	basic monthly	9.9	08-08-2020
2	trial	0	20-09-2020
2	pro annual	199	27-09-2020
11	trial	0	19-11-2020
11	churn		26-11-2020
13	trial	0	15-12-2020
13	basic monthly	9.9	22-12-2020
13	pro monthly	19.9	29-03-2021
15	trial	0	17-03-2020
15	pro monthly	19.9	24-03-2020
15	churn		29-04-2020
16	trial	0	31-05-2020
16	basic monthly	9.9	07-06-2020
16	pro annual	199	21-10-2020
18	trial	0	06-07-2020
18	pro monthly	19.9	13-07-2020
19	trial	0	22-06-2020
19	pro monthly	19.9	29-06-2020
19	pro annual	199	29-08-2020

Customer 1 : - Signed up for free trail of Foddie-Fi on August 1 2020, and after trial period end they subscribed to basic monthly plan on August 8 2020

Customer 2 : - Signed up for free trail of Foddie-Fi on September 20 2020, and after trial period end they subscribed to pro annual plan on September 27 2020

Customer 11 : - Signed up for free trail of Foddie-Fi on November 19 2020, and cancelled the service on November 26 2020 right after the end of 7 - days trial
the service on November 26 2020 right after the end of 7 - days trial

Customer 13 : - Signed up for free trail of Foddie-Fi on December 15 2020,after the trial period end they subscribed to basic monthly plan on December 22 2020 and later upgrade to pro monthly plan on March 29 2021

Customer 15 : - Signed up for free trail of Foddie-Fi on March 17 2020,after the trial period end they subscribed to pro monthly plan on March 24 2020 . However they cancelled the service on April 29 2020.

Customer 16 : - Signed up for free trail of Foddie-Fi on May 31 2020,after the trial period end they subscribed to basic monthly plan on June 7 202 and later upgrade to pro annual plan on October 21 2020

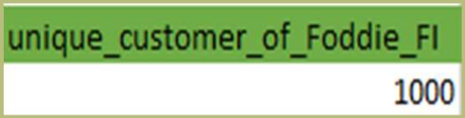
Customer 18 : - Signed up for free trail of Foddie-Fi on July 7 2020,after the trial period end they subscribed to pro monthly plan on June 13 2020

Customer 19 :- Signed up for free trail of Foddie-Fi on June 22 2020,after the trial period end they subscribed to pro monthly plan on June 29 2020 and later upgrade to pro annual plan on August 29 2020

Data Analysis Question

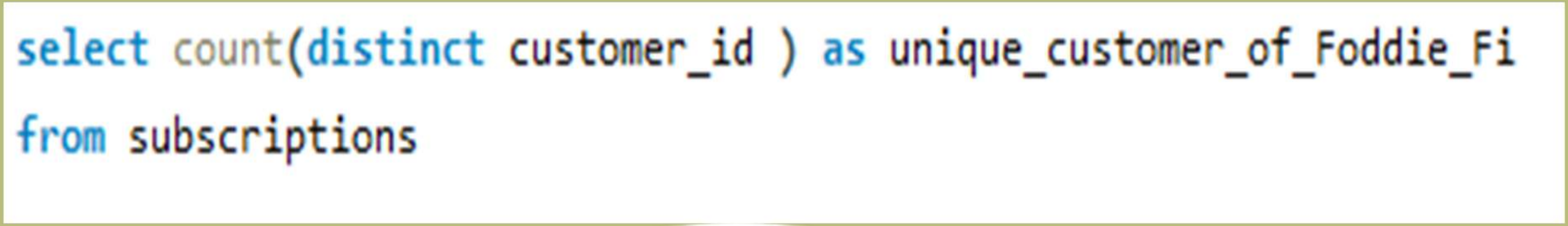
1.How many customers has Foodie-Fi ever had?

Solution



unique_customer_of_Foddie_FI
1000

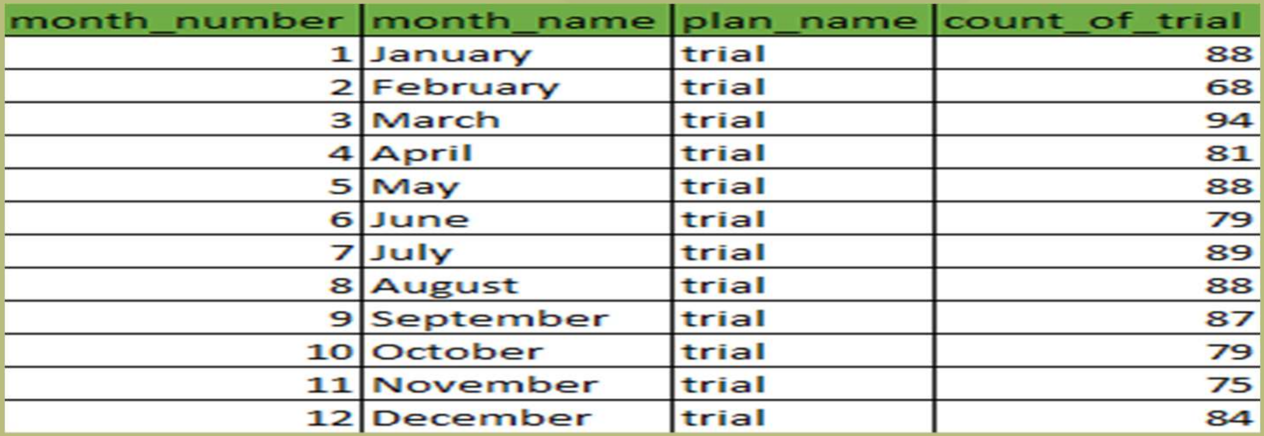
Query



```sql
select count(distinct customer_id ) as unique_customer_of_Foddie_Fi
from subscriptions
```

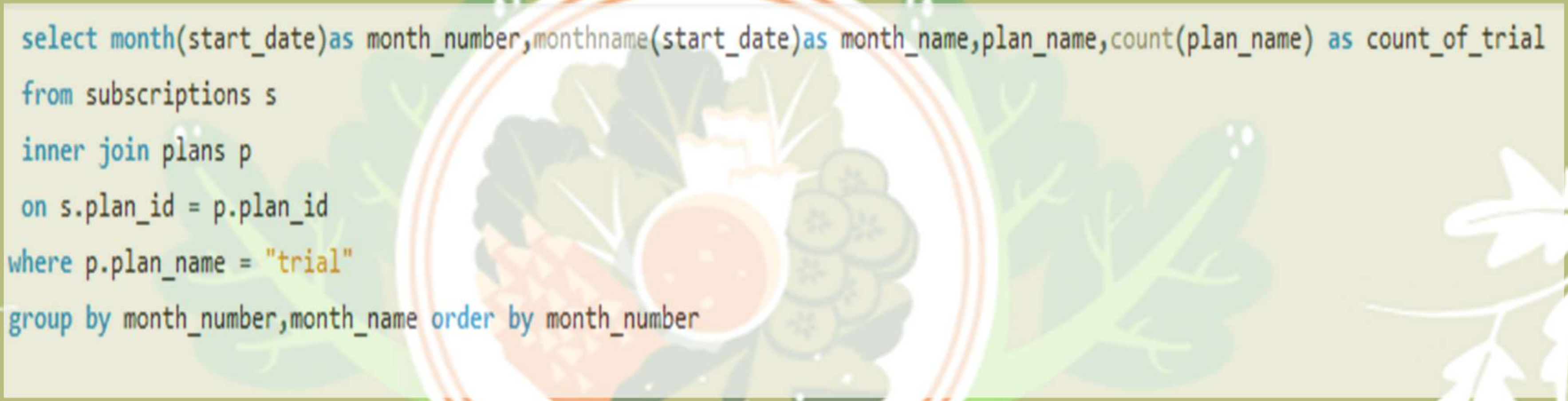
2) What is the monthly distribution of trial plan start_date values for our data set

Solution



month_number	month_name	plan_name	count_of_trial
1	January	trial	88
2	February	trial	68
3	March	trial	94
4	April	trial	81
5	May	trial	88
6	June	trial	79
7	July	trial	89
8	August	trial	88
9	September	trial	87
10	October	trial	79
11	November	trial	75
12	December	trial	84

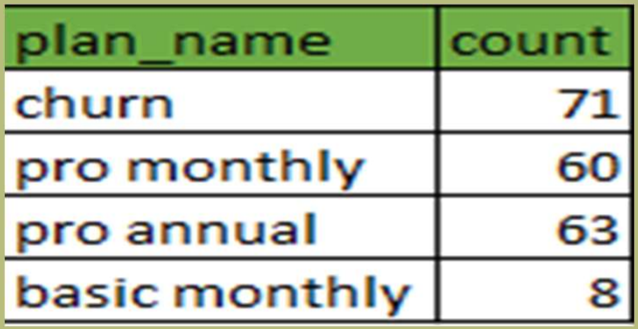
Query



```
select month(start_date)as month_number,monthname(start_date)as month_name,plan_name,count(plan_name) as count_of_trial
from subscriptions s
inner join plans p
on s.plan_id = p.plan_id
where p.plan_name = "trial"
group by month_number,month_name order by month_number
```

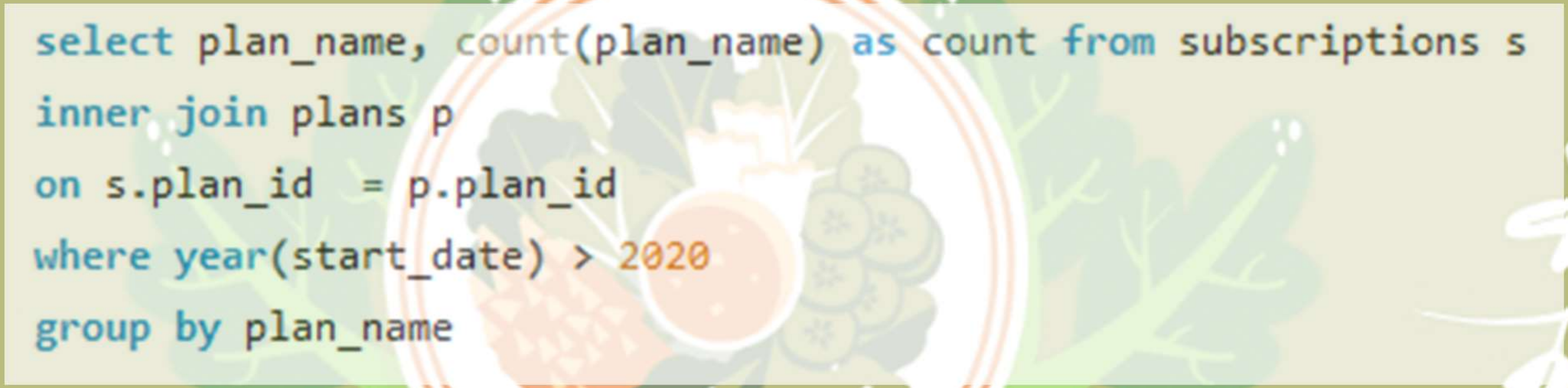
3) What plan start_date value occurs after the year 2020 for our datasets ? Show the breakdown by count of events for each plan_name

Solution



plan_name	count
churn	71
pro monthly	60
pro annual	63
basic monthly	8

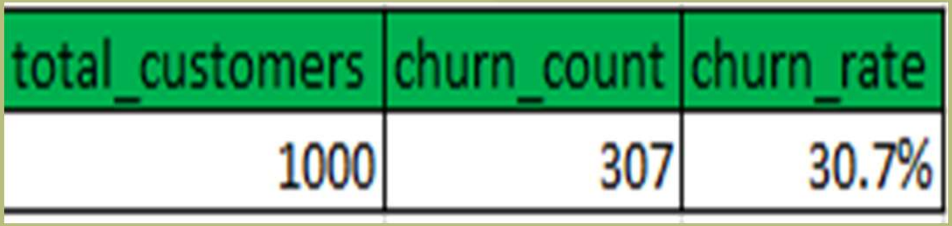
Query



```
select plan_name, count(plan_name) as count from subscriptions s
inner join plans p
on s.plan_id  = p.plan_id
where year(start_date) > 2020
group by plan_name
```

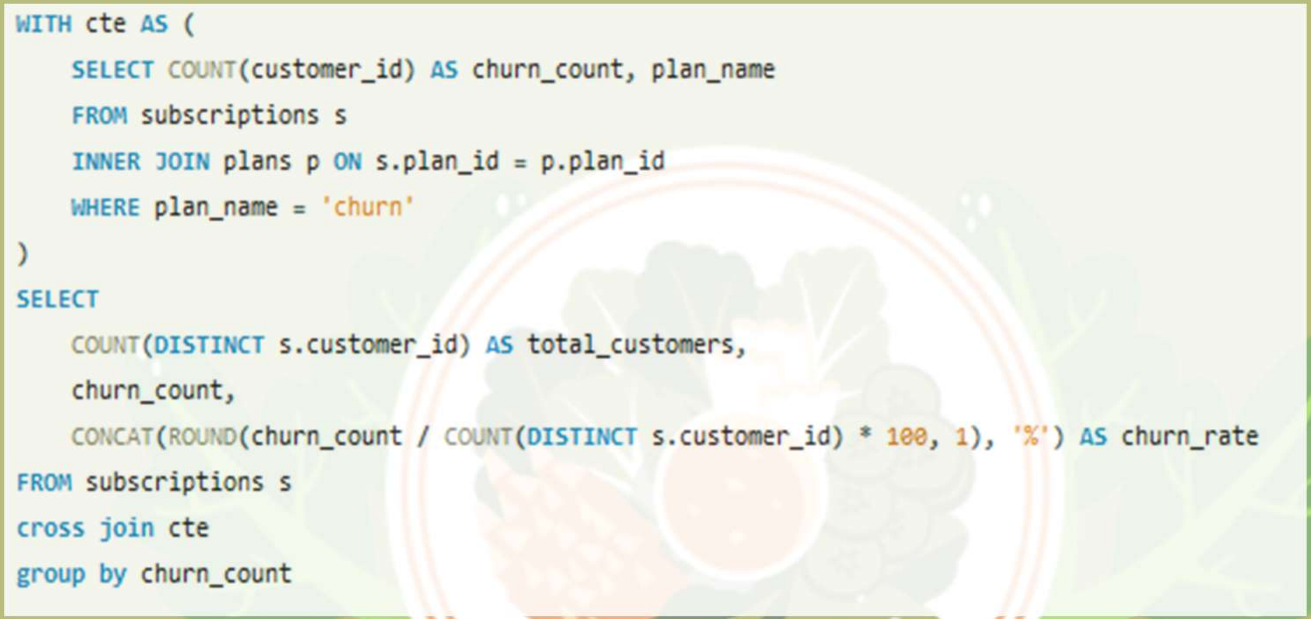
4) What is the customer count and percentage of customers who have churned rounded to 1 decimal place

Solution



total_customers	churn_count	churn_rate
1000	307	30.7%

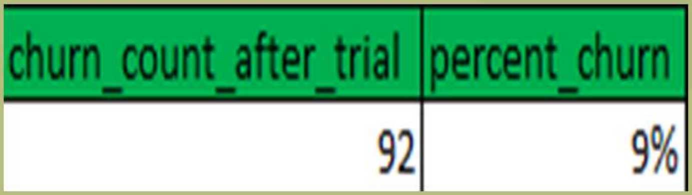
Query



```
WITH cte AS (
    SELECT COUNT(customer_id) AS churn_count, plan_name
    FROM subscriptions s
    INNER JOIN plans p ON s.plan_id = p.plan_id
    WHERE plan_name = 'churn'
)
SELECT
    COUNT(DISTINCT s.customer_id) AS total_customers,
    churn_count,
    CONCAT(ROUND(churn_count / COUNT(DISTINCT s.customer_id) * 100, 1), '%') AS churn_rate
FROM subscriptions s
cross join cte
group by churn_count
```

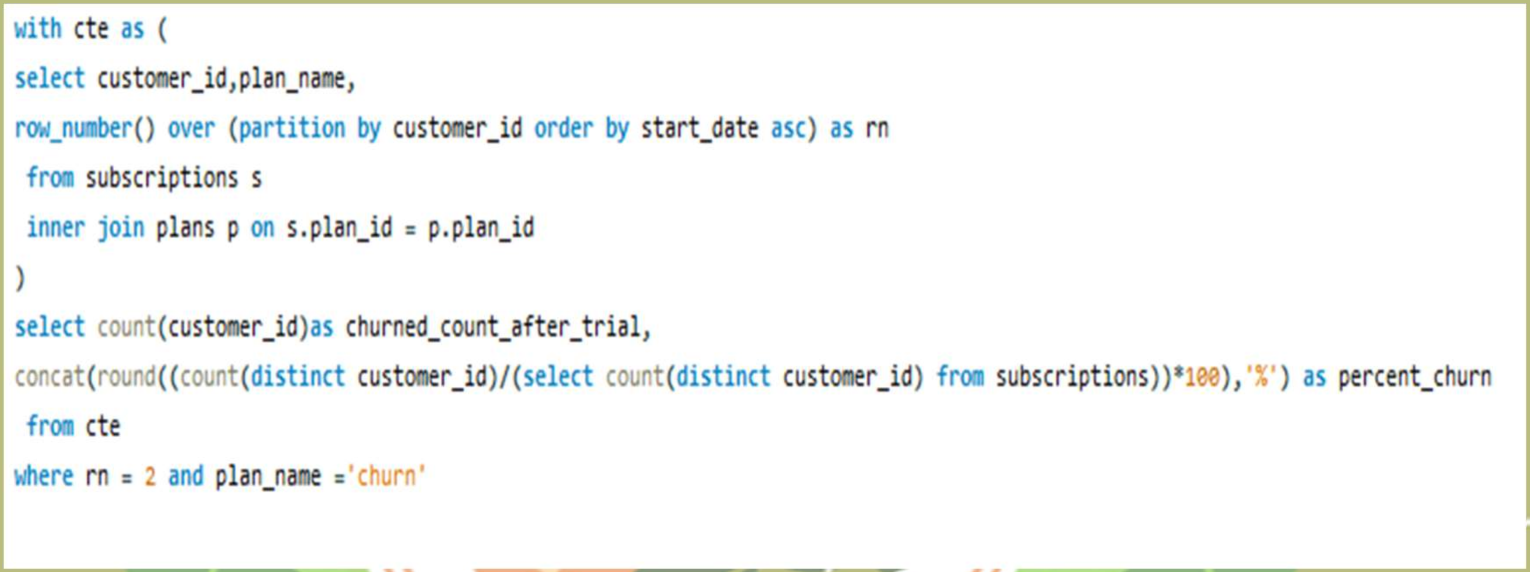
5) How many customer have churned straight after their initial free trial – what percentage is this rounded to nearest whole number ?

Solution



churn_count_after_trial	percent_churn
92	9%

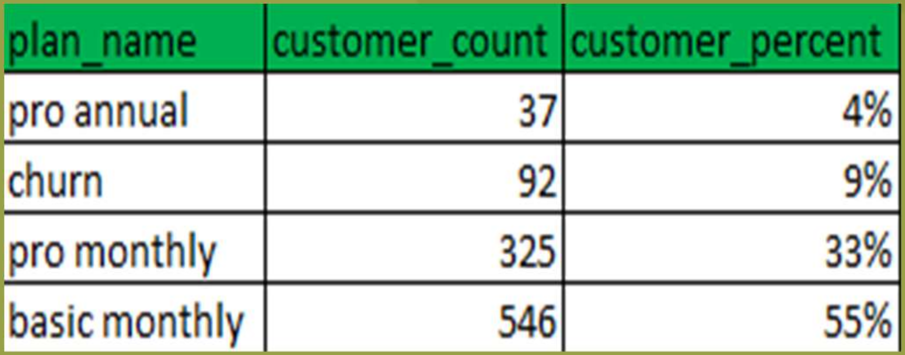
Query



```
with cte as (
select customer_id,plan_name,
row_number() over (partition by customer_id order by start_date asc) as rn
 from subscriptions s
 inner join plans p on s.plan_id = p.plan_id
)
select count(customer_id)as churned_count_after_trial,
concat(round((count(distinct customer_id)/(select count(distinct customer_id) from subscriptions))*100),'%') as percent_churn
 from cte
where rn = 2 and plan_name ='churn'
```

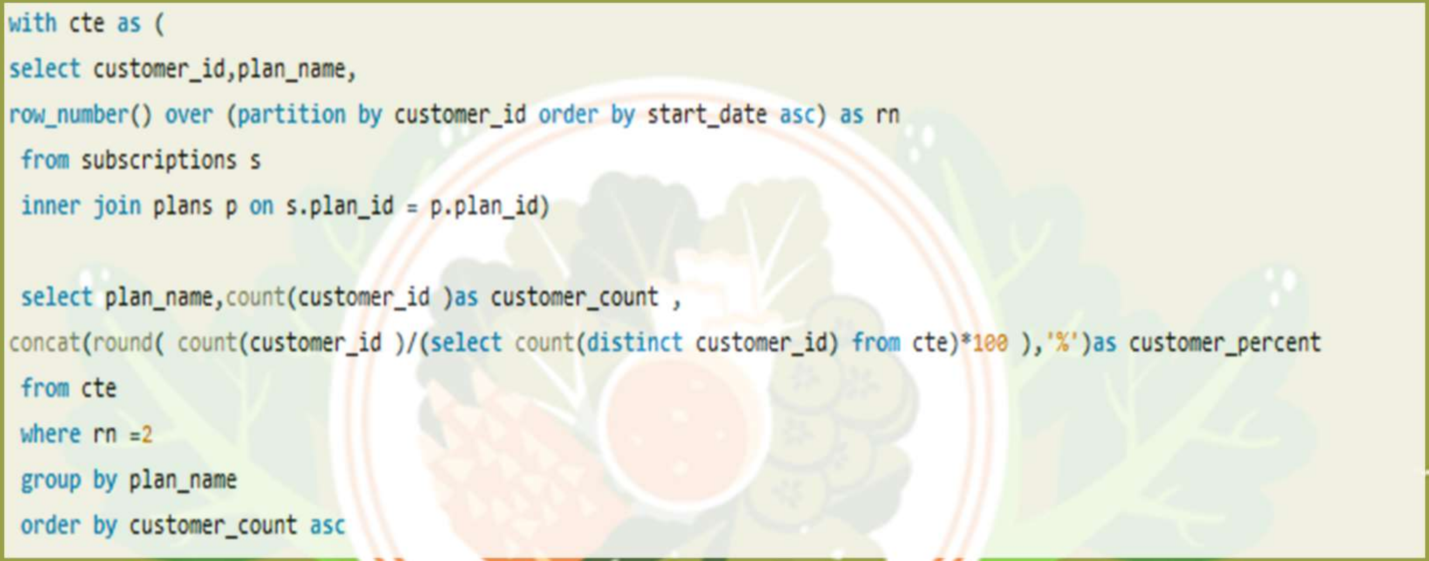
6) What is the number and percentage of customers plan after their initial free trial ?

Solution



plan_name	customer_count	customer_percent
pro annual	37	4%
churn	92	9%
pro monthly	325	33%
basic monthly	546	55%

Query

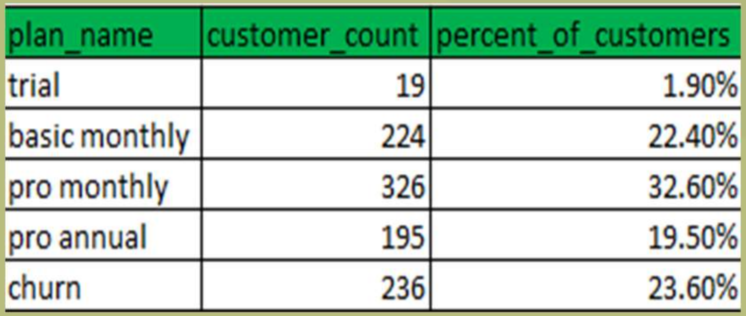


```
with cte as (
select customer_id,plan_name,
row_number() over (partition by customer_id order by start_date asc) as rn
 from subscriptions s
 inner join plans p on s.plan_id = p.plan_id)

 select plan_name,count(customer_id )as customer_count ,
concat(round( count(customer_id )/(select count(distinct customer_id) from cte)*100 ),'%')as customer_percent
 from cte
 where rn =2
 group by plan_name
 order by customer_count asc
```

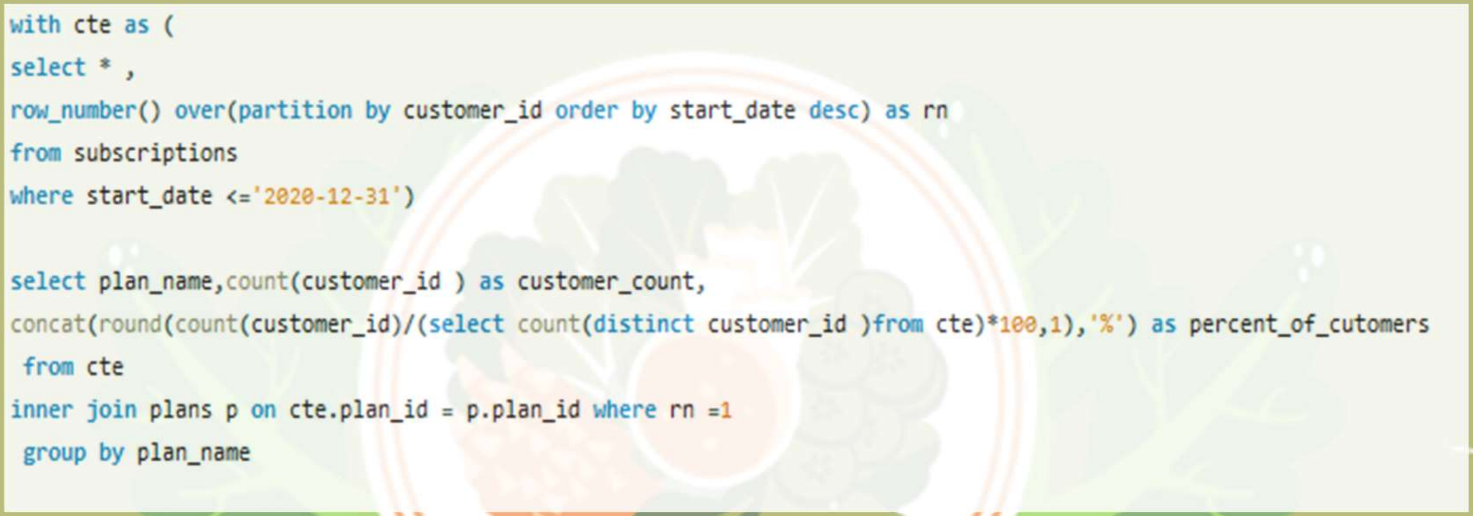
7) What is the customer count and percentage breakdown of all 5 plan_name values at 2020-12-31 ?

Solution



plan_name	customer_count	percent_of_customers
trial	19	1.90%
basic monthly	224	22.40%
pro monthly	326	32.60%
pro annual	195	19.50%
churn	236	23.60%

Query

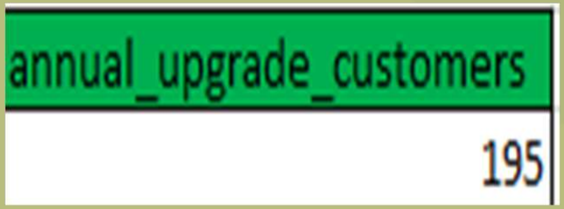


```
with cte as (
select * ,
row_number() over(partition by customer_id order by start_date desc) as rn
from subscriptions
where start_date <='2020-12-31')

select plan_name,count(customer_id ) as customer_count,
concat(round(count(customer_id)/(select count(distinct customer_id )from cte)*100,1),'%') as percent_of_cutomers
 from cte
inner join plans p on cte.plan_id = p.plan_id where rn =1
 group by plan_name
```

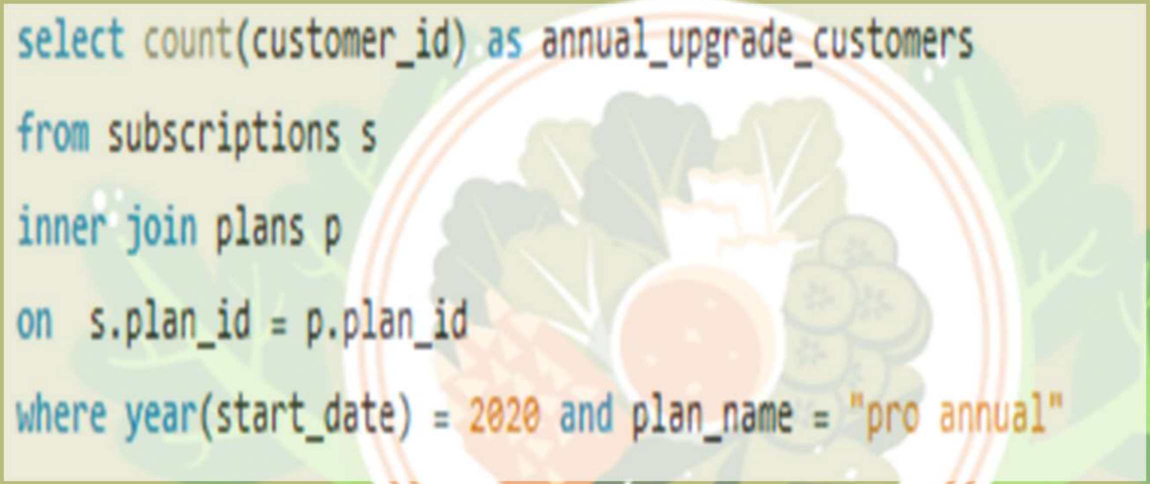
8) How many customers have upgraded to an annual plan in 2020 ?

Solution



annual_upgrade_customers
195

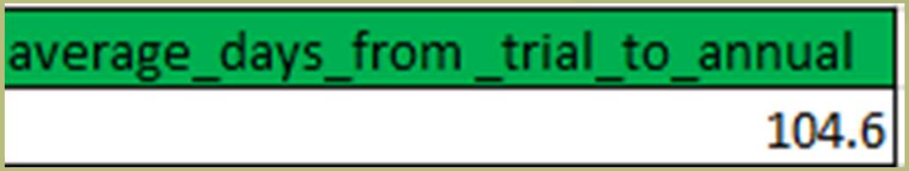
Query



```
select count(customer_id) as annual_upgrade_customers
from subscriptions s
inner join plans p
on  s.plan_id = p.plan_id
where year(start_date) = 2020 and plan_name = "pro annual"
```

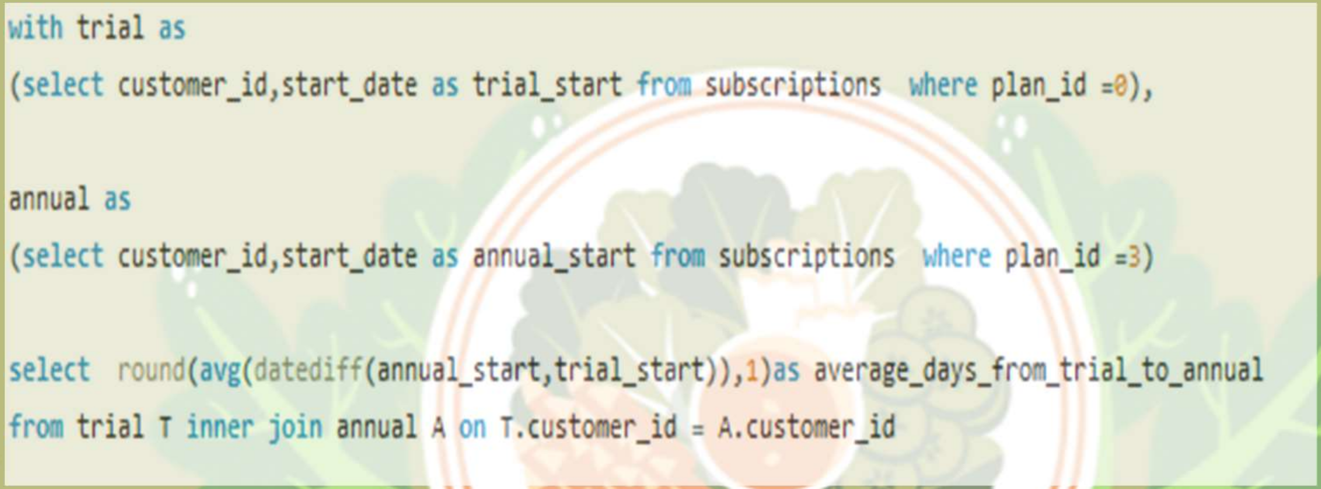
9) How many days on an average does it take for a customer to an annual plan from the day they join Foddie-FI ?

Solution



average_days_from_trial_to_annual
104.6

Query



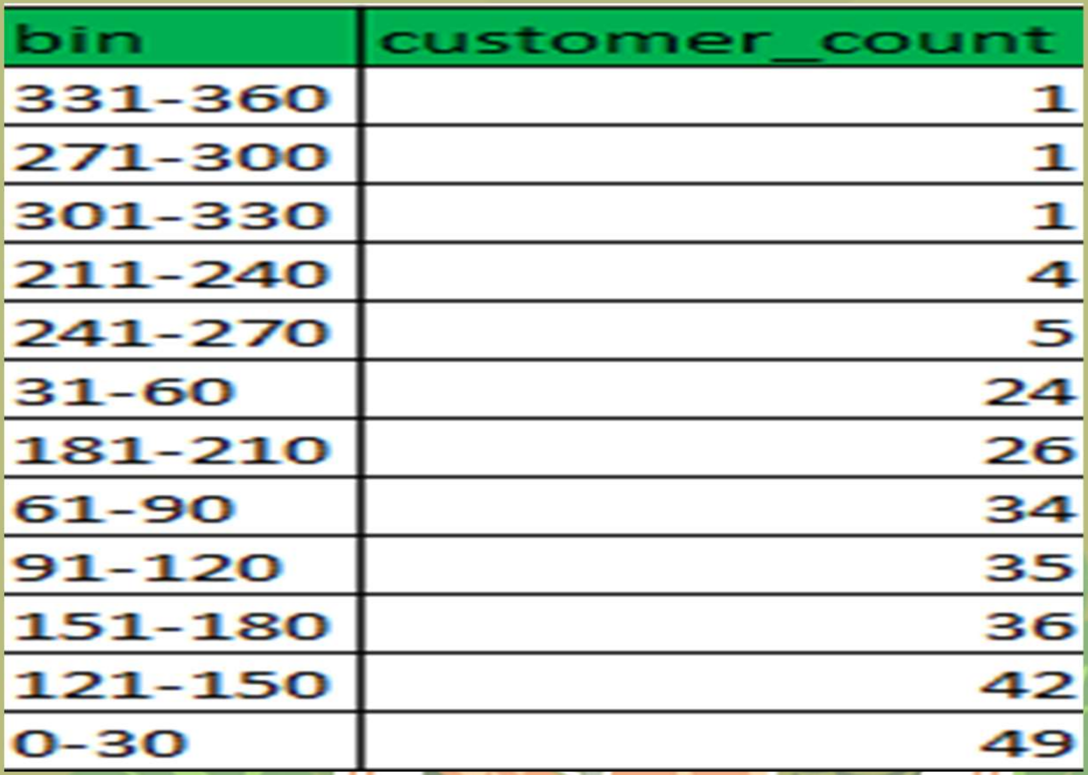
```
with trial as
(select customer_id,start_date as trial_start from subscriptions  where plan_id =0),

annual as
(select customer_id,start_date as annual_start from subscriptions  where plan_id =3)

select  round(avg(datediff(annual_start,trial_start)),1)as average_days_from_trial_to_annual
from trial T inner join annual A on T.customer_id = A.customer_id
```

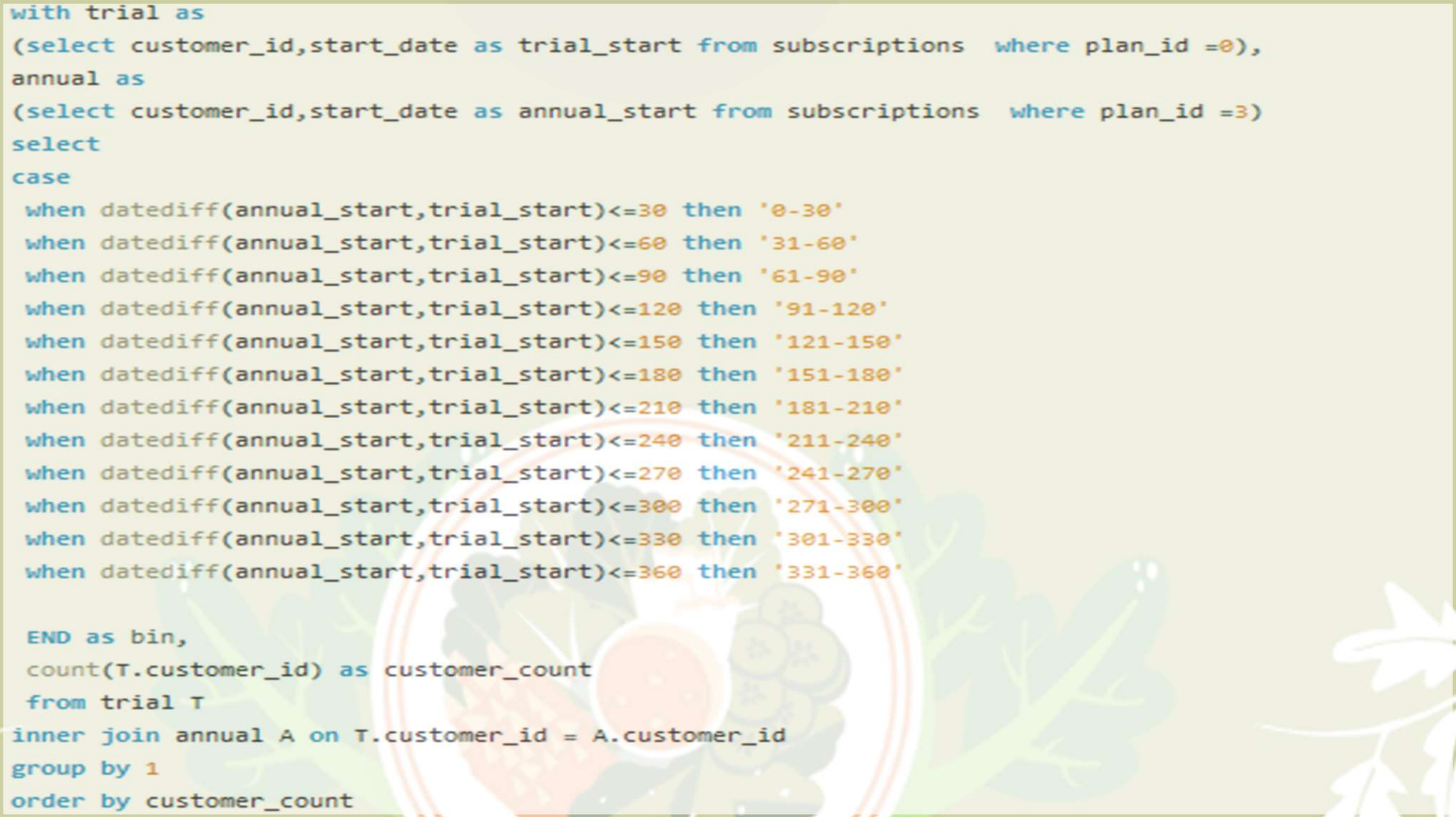
10) Can you further breakdown this average value into 30 days period (ie 0-30 days,31-60 days etc. ?

Solution



bin	customer_count
331-360	1
271-300	1
301-330	1
211-240	4
241-270	5
31-60	24
181-210	26
61-90	34
91-120	35
151-180	36
121-150	42
0-30	49

Query

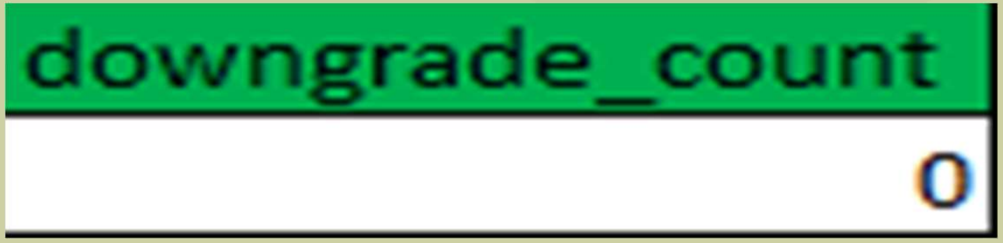


```
with trial as
(select customer_id,start_date as trial_start from subscriptions  where plan_id =0),
annual as
(select customer_id,start_date as annual_start from subscriptions  where plan_id =3)
select
case
 when datediff(annual_start,trial_start)<=30 then '0-30'
 when datediff(annual_start,trial_start)<=60 then '31-60'
 when datediff(annual_start,trial_start)<=90 then '61-90'
 when datediff(annual_start,trial_start)<=120 then '91-120'
 when datediff(annual_start,trial_start)<=150 then '121-150'
 when datediff(annual_start,trial_start)<=180 then '151-180'
 when datediff(annual_start,trial_start)<=210 then '181-210'
 when datediff(annual_start,trial_start)<=240 then '211-240'
 when datediff(annual_start,trial_start)<=270 then '241-270'
 when datediff(annual_start,trial_start)<=300 then '271-300'
 when datediff(annual_start,trial_start)<=330 then '301-330'
 when datediff(annual_start,trial_start)<=360 then '331-360'

 END as bin,
 count(T.customer_id) as customer_count
 from trial T
inner join annual A on T.customer_id = A.customer_id
group by 1
order by customer_count
```

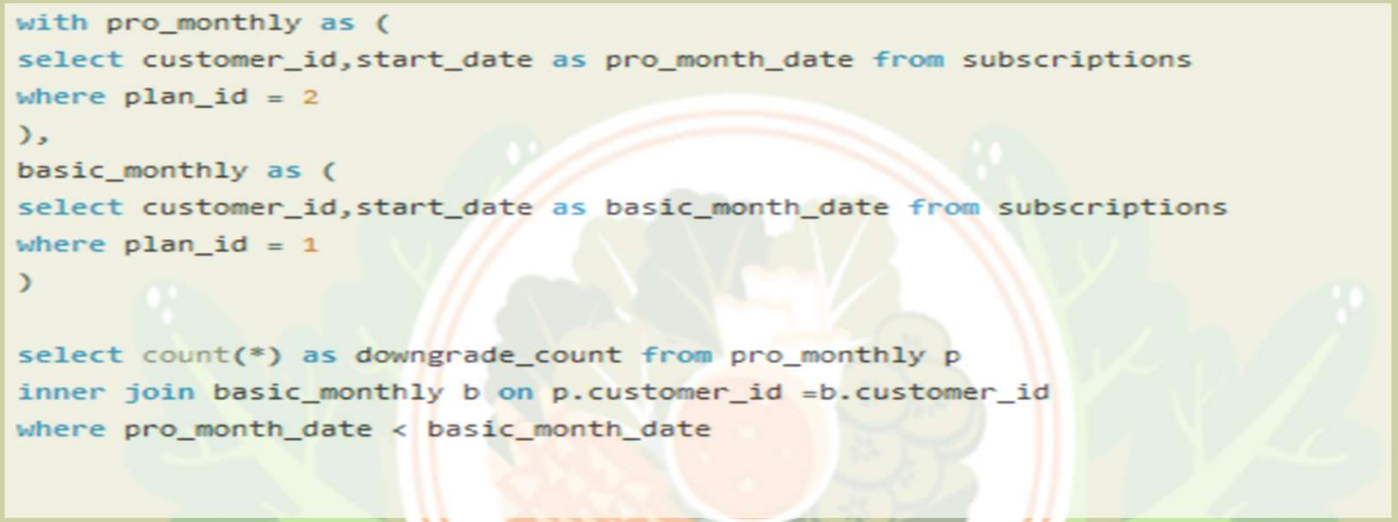
11) How many customers downgraded from a pro monthly to basic monthly in 2020 ?

Solution



downgrade_count
0

Query



```
with pro_monthly as (
select customer_id,start_date as pro_month_date from subscriptions
where plan_id = 2
),
basic_monthly as (
select customer_id,start_date as basic_month_date from subscriptions
where plan_id = 1
)

select count(*) as downgrade_count from pro_monthly p
inner join basic_monthly b on p.customer_id =b.customer_id
where pro_month_date < basic_month_date
```

INSIGHTS

1) FOODIE-Fi ever had 1000 customers.

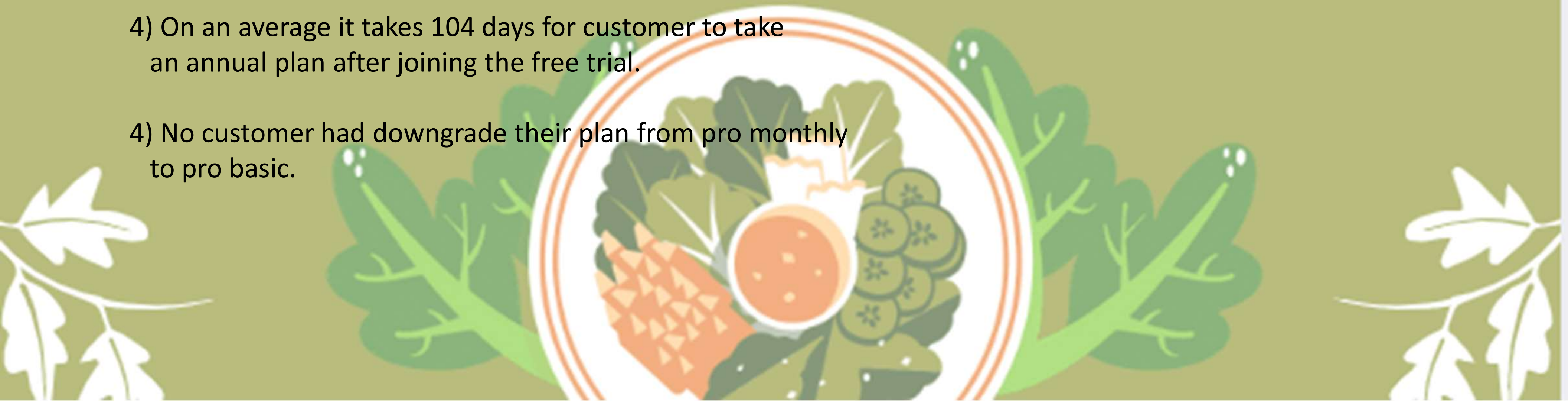
2) The churn rate of customers is 30.70% .

3) Around 9% customer churn after their initial free trial.

4) After the initial free trail around 55 % customer go
for the basic monthly and 4% for pro annual plan.

4) On an average it takes 104 days for customer to take
an annual plan after joining the free trial.

4) No customer had downgrade their plan from pro monthly
to pro basic.



Challenge Payment Questions

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:

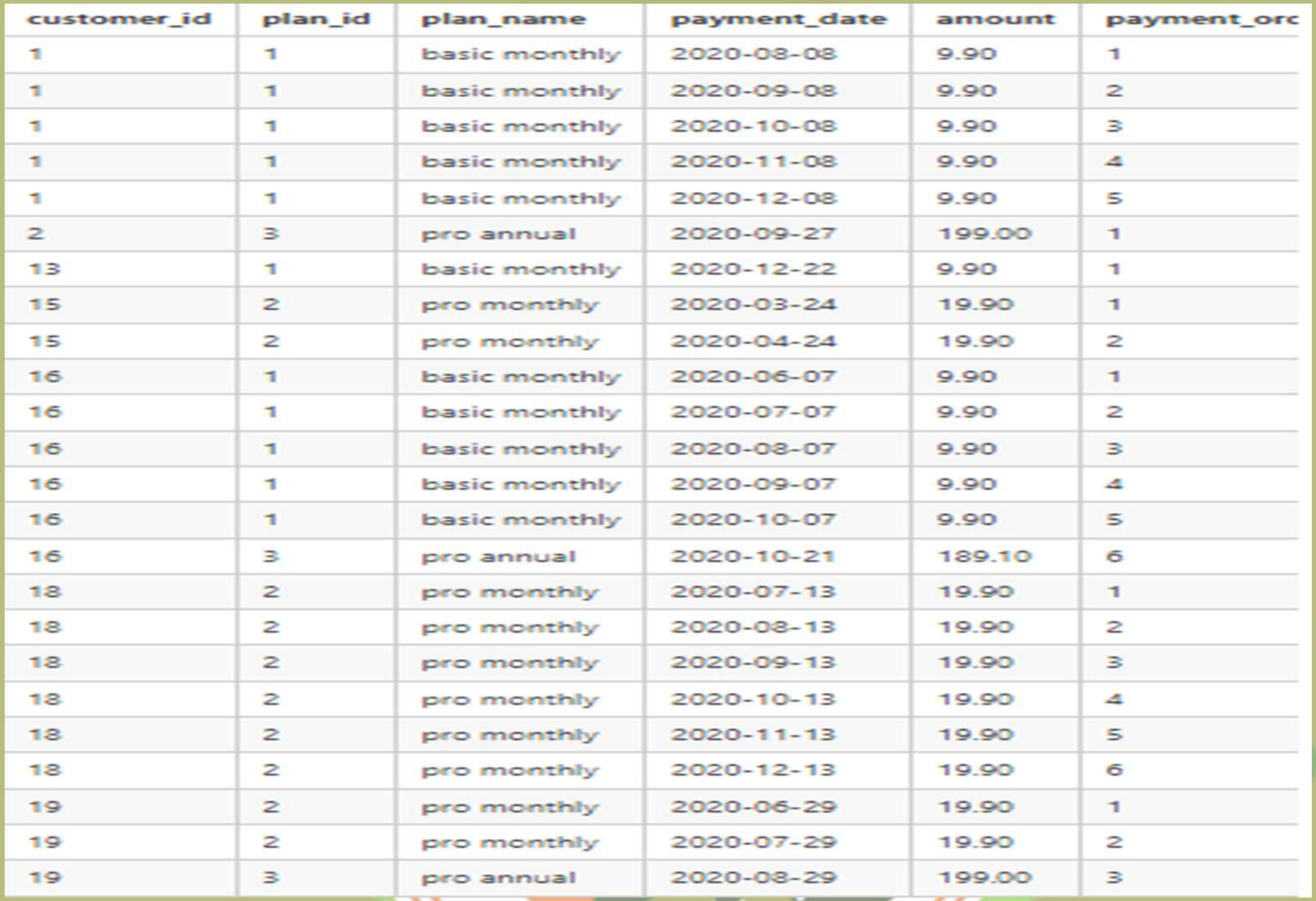
monthly payments always occur on the same day of month as the original start date of any monthly paid plan

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:

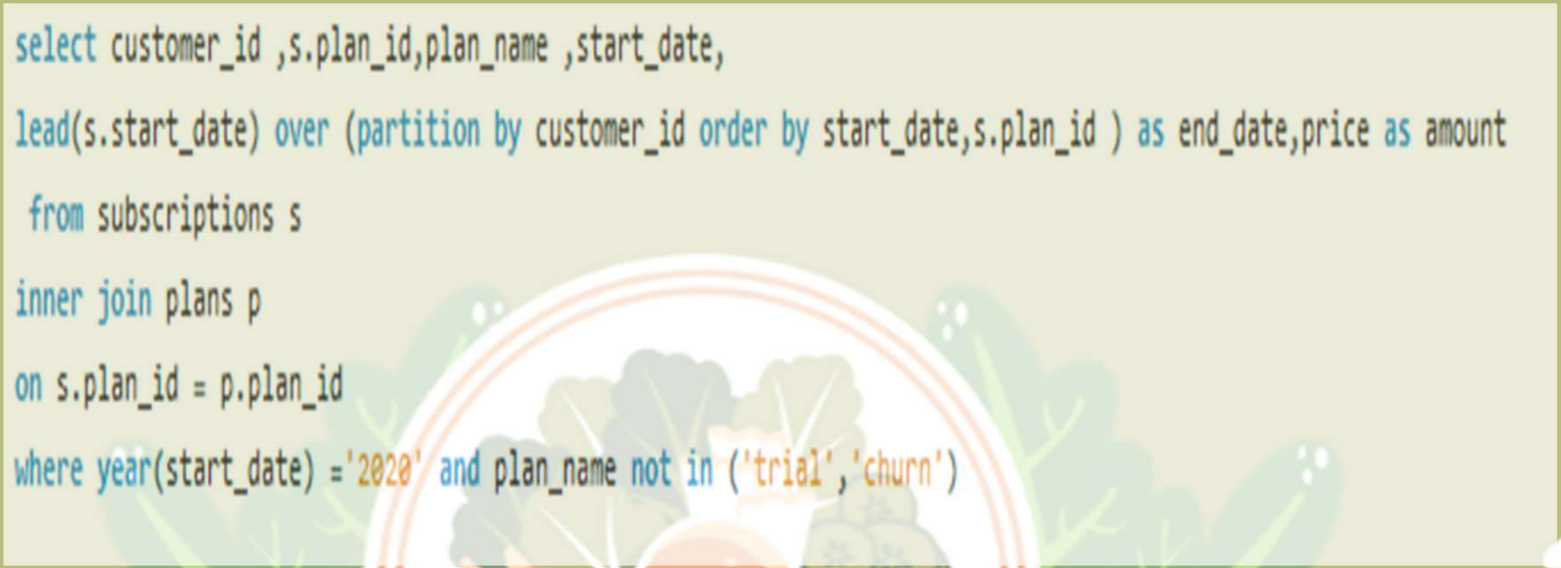


customer_id	plan_id	plan_name	payment_date	amount	payment_orc
1	1	basic monthly	2020-08-08	9.90	1
1	1	basic monthly	2020-09-08	9.90	2
1	1	basic monthly	2020-10-08	9.90	3
1	1	basic monthly	2020-11-08	9.90	4
1	1	basic monthly	2020-12-08	9.90	5
2	3	pro annual	2020-09-27	199.00	1
13	1	basic monthly	2020-12-22	9.90	1
15	2	pro monthly	2020-03-24	19.90	1
15	2	pro monthly	2020-04-24	19.90	2
16	1	basic monthly	2020-06-07	9.90	1
16	1	basic monthly	2020-07-07	9.90	2
16	1	basic monthly	2020-08-07	9.90	3
16	1	basic monthly	2020-09-07	9.90	4
16	1	basic monthly	2020-10-07	9.90	5
16	3	pro annual	2020-10-21	189.10	6
18	2	pro monthly	2020-07-13	19.90	1
18	2	pro monthly	2020-08-13	19.90	2
18	2	pro monthly	2020-09-13	19.90	3
18	2	pro monthly	2020-10-13	19.90	4
18	2	pro monthly	2020-11-13	19.90	5
18	2	pro monthly	2020-12-13	19.90	6
19	2	pro monthly	2020-06-29	19.90	1
19	2	pro monthly	2020-07-29	19.90	2
19	3	pro annual	2020-08-29	199.00	3

We will do the solution in steps

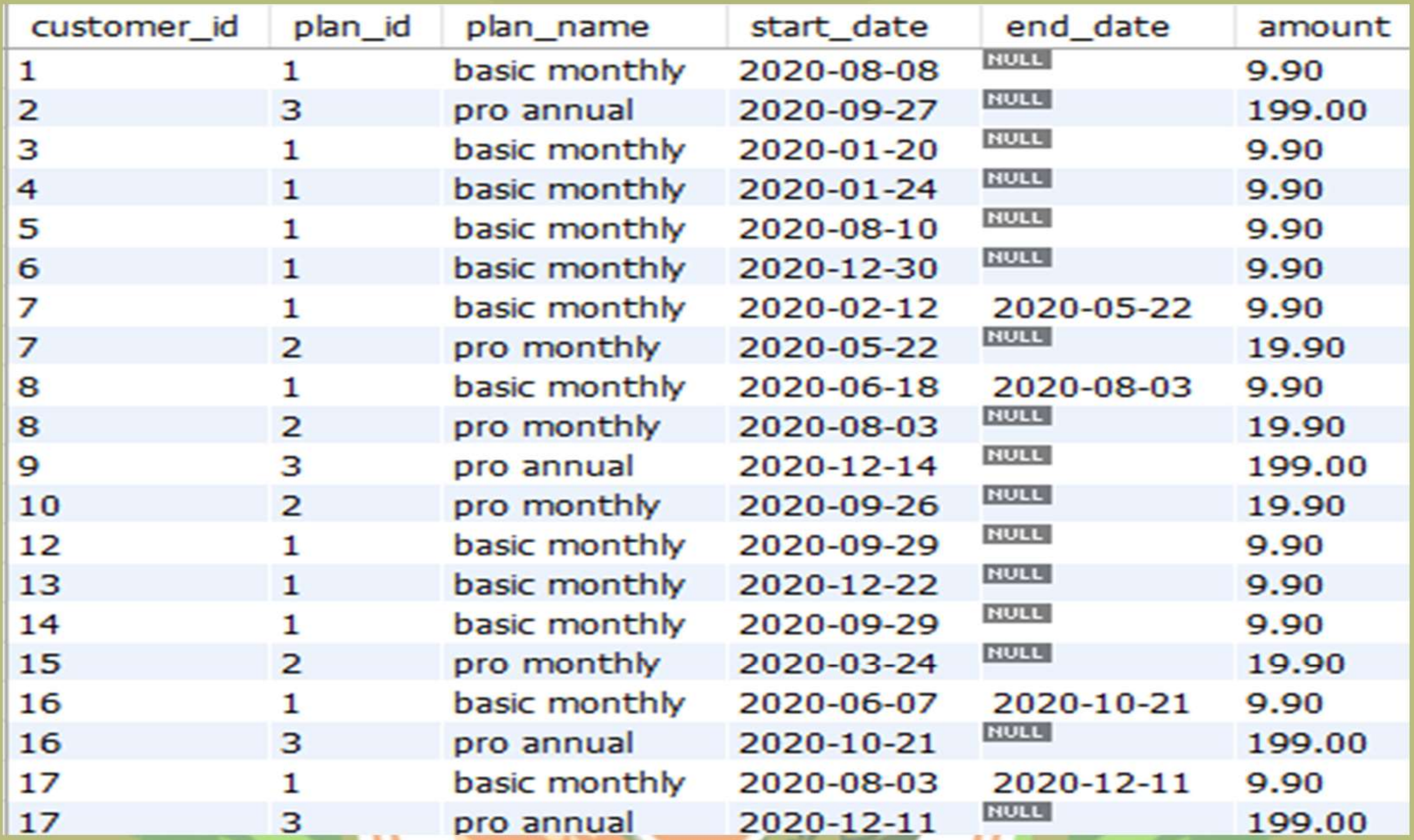
Step 1 : - Filter the subscription to only include the year 2020 . Use lead window function to calculate the end date for next plan (if available) . Exclude and plans as “trial” or “churn” since no payment are made on these

Query of step 1 :



```
select customer_id ,s.plan_id,plan_name ,start_date,
lead(s.start_date) over (partition by customer_id order by start_date,s.plan_id ) as end_date,price as amount
 from subscriptions s
inner join plans p
on s.plan_id = p.plan_id
where year(start_date) ='2020' and plan_name not in ('trial','churn')
```

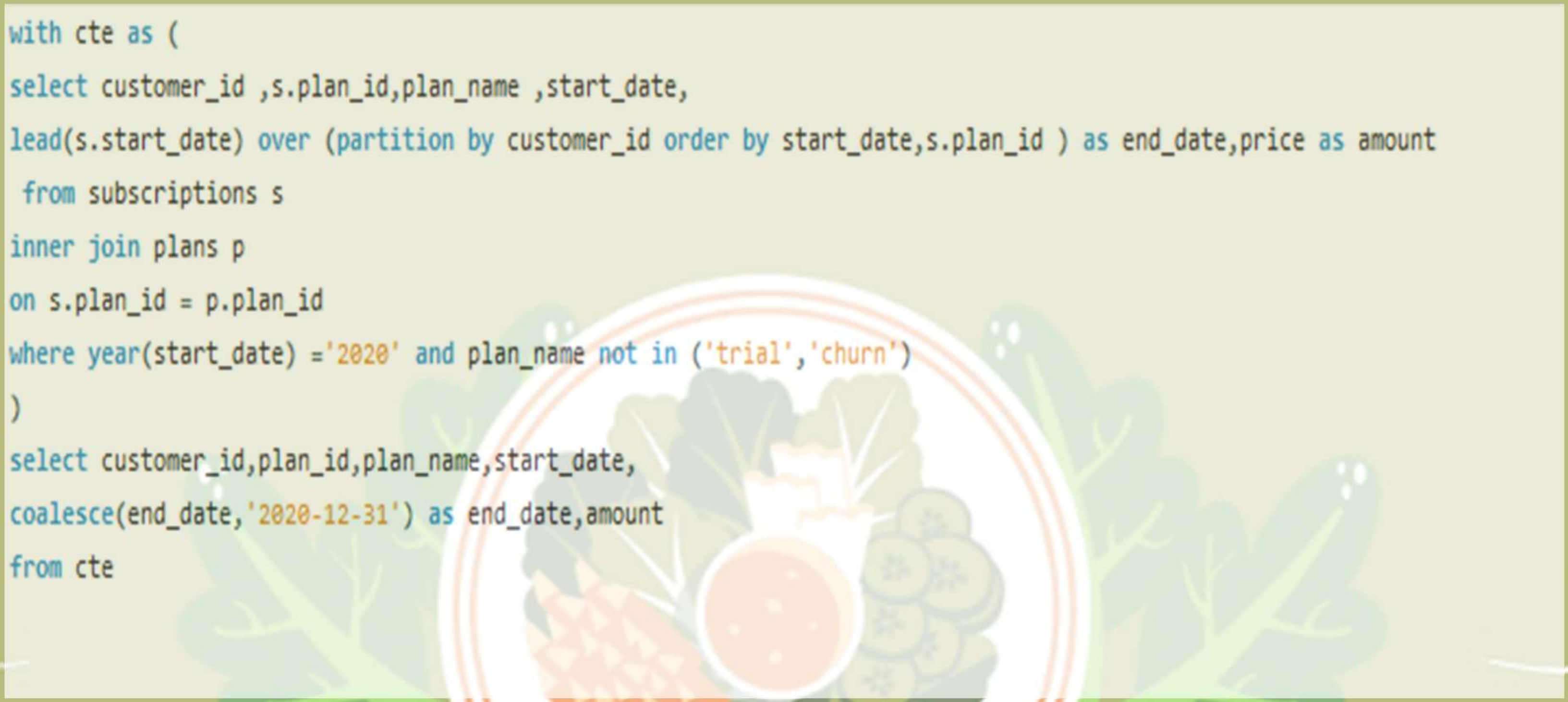
Output of step 1 :



customer_id	plan_id	plan_name	start_date	end_date	amount
1	1	basic monthly	2020-08-08	NULL	9.90
2	3	pro annual	2020-09-27	NULL	199.00
3	1	basic monthly	2020-01-20	NULL	9.90
4	1	basic monthly	2020-01-24	NULL	9.90
5	1	basic monthly	2020-08-10	NULL	9.90
6	1	basic monthly	2020-12-30	NULL	9.90
7	1	basic monthly	2020-02-12	2020-05-22	9.90
7	2	pro monthly	2020-05-22	NULL	19.90
8	1	basic monthly	2020-06-18	2020-08-03	9.90
8	2	pro monthly	2020-08-03	NULL	19.90
9	3	pro annual	2020-12-14	NULL	199.00
10	2	pro monthly	2020-09-26	NULL	19.90
12	1	basic monthly	2020-09-29	NULL	9.90
13	1	basic monthly	2020-12-22	NULL	9.90
14	1	basic monthly	2020-09-29	NULL	9.90
15	2	pro monthly	2020-03-24	NULL	19.90
16	1	basic monthly	2020-06-07	2020-10-21	9.90
16	3	pro annual	2020-10-21	NULL	199.00
17	1	basic monthly	2020-08-03	2020-12-11	9.90
17	3	pro annual	2020-12-11	NULL	199.00

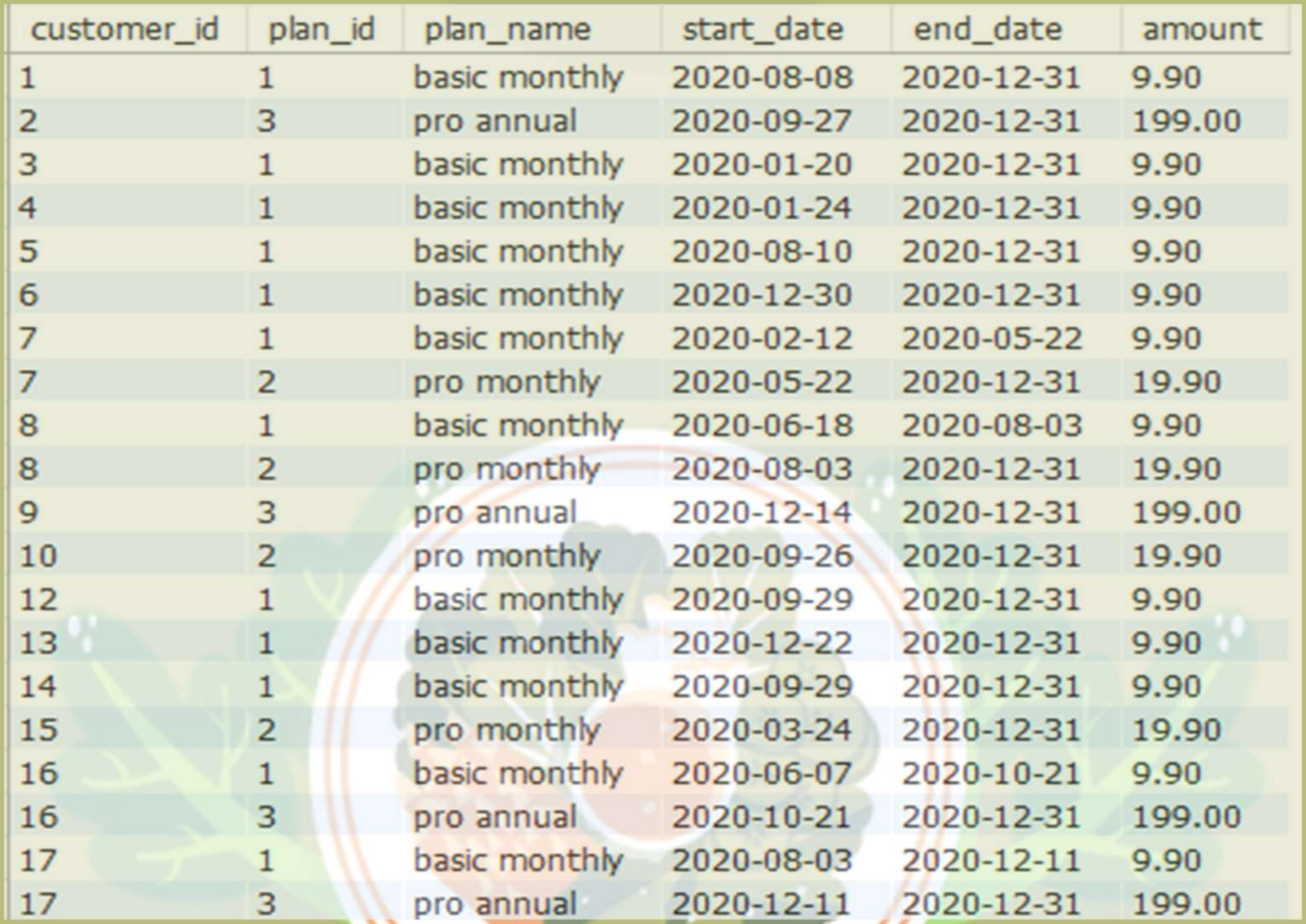
Step 2 : - Replaced the “null” values in the end date column with the last day of year 2020 indicating that it was the last plan user had for that year

Query of step 2 :



```
with cte as (
select customer_id ,s.plan_id,plan_name ,start_date,
lead(s.start_date) over (partition by customer_id order by start_date,s.plan_id ) as end_date,price as amount
 from subscriptions s
inner join plans p
on s.plan_id = p.plan_id
where year(start_date) ='2020' and plan_name not in ('trial','churn')
)
select customer_id,plan_id,plan_name,start_date,
coalesce(end_date,'2020-12-31') as end_date,amount
from cte
```

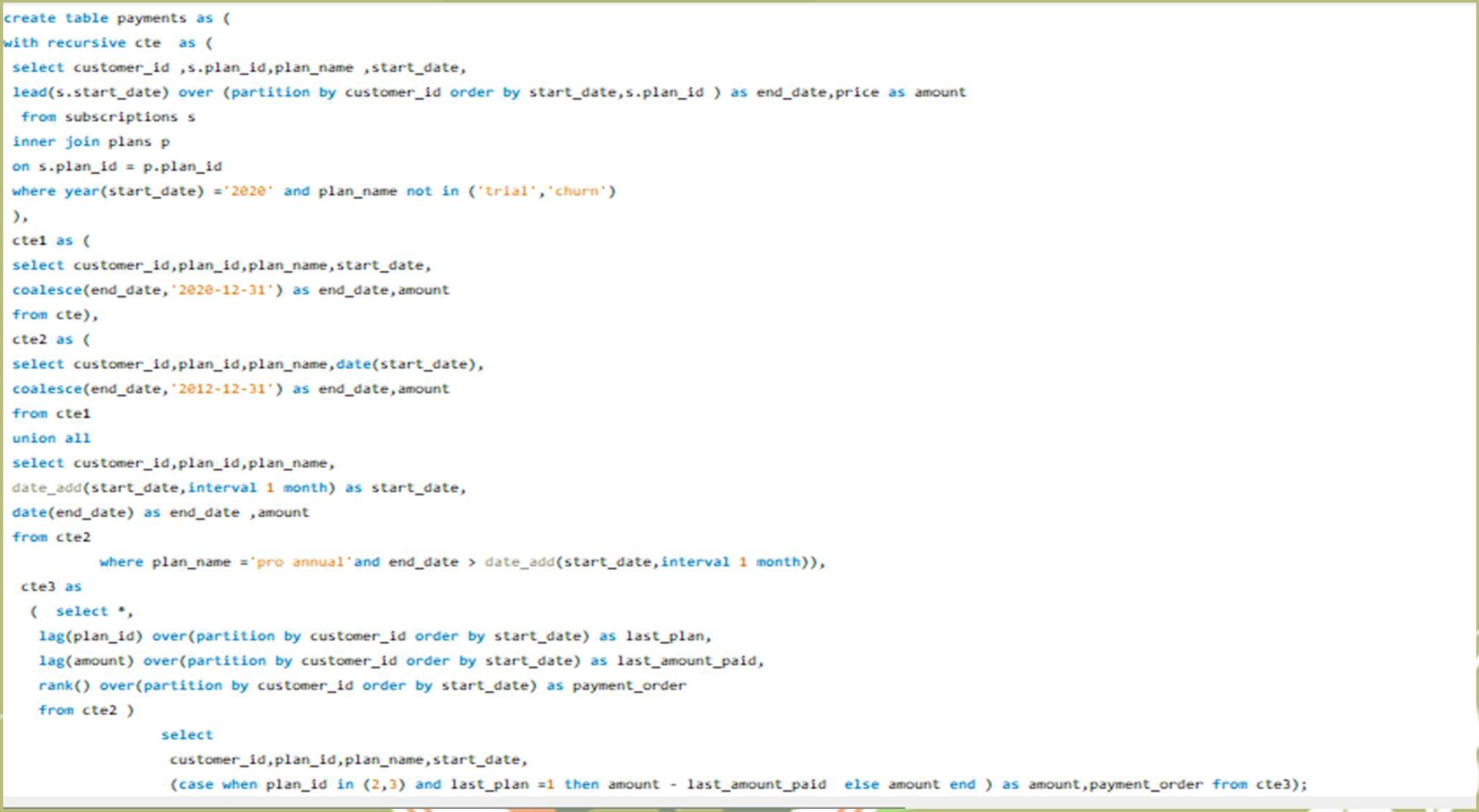
Output of step 2 :



customer_id	plan_id	plan_name	start_date	end_date	amount
1	1	basic monthly	2020-08-08	2020-12-31	9.90
2	3	pro annual	2020-09-27	2020-12-31	199.00
3	1	basic monthly	2020-01-20	2020-12-31	9.90
4	1	basic monthly	2020-01-24	2020-12-31	9.90
5	1	basic monthly	2020-08-10	2020-12-31	9.90
6	1	basic monthly	2020-12-30	2020-12-31	9.90
7	1	basic monthly	2020-02-12	2020-05-22	9.90
7	2	pro monthly	2020-05-22	2020-12-31	19.90
8	1	basic monthly	2020-06-18	2020-08-03	9.90
8	2	pro monthly	2020-08-03	2020-12-31	19.90
9	3	pro annual	2020-12-14	2020-12-31	199.00
10	2	pro monthly	2020-09-26	2020-12-31	19.90
12	1	basic monthly	2020-09-29	2020-12-31	9.90
13	1	basic monthly	2020-12-22	2020-12-31	9.90
14	1	basic monthly	2020-09-29	2020-12-31	9.90
15	2	pro monthly	2020-03-24	2020-12-31	19.90
16	1	basic monthly	2020-06-07	2020-10-21	9.90
16	3	pro annual	2020-10-21	2020-12-31	199.00
17	1	basic monthly	2020-08-03	2020-12-11	9.90
17	3	pro annual	2020-12-11	2020-12-31	199.00

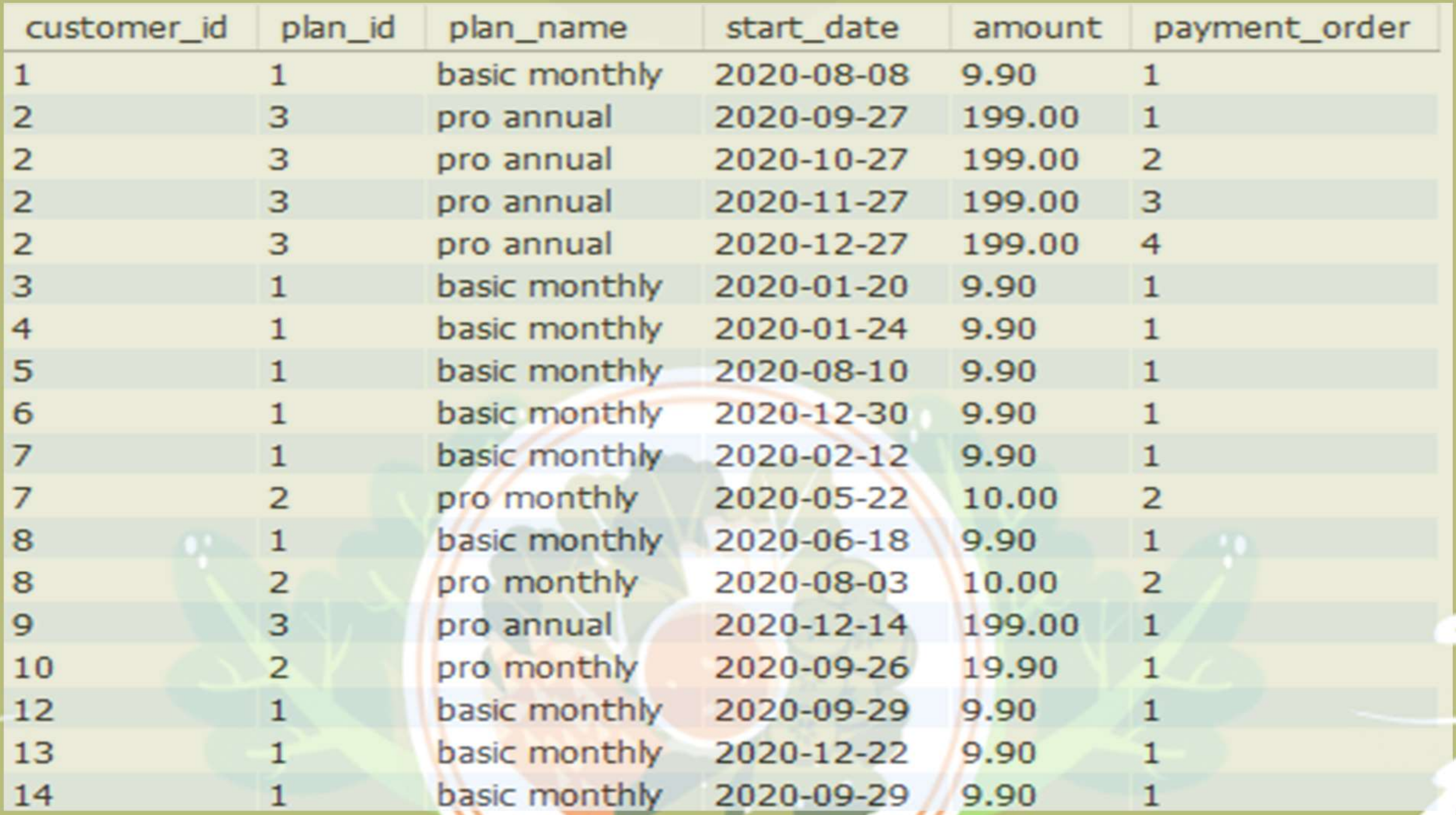
Step 3 – created a recursive CTE query that generates new rows for new users on monthly plans by incrementing the start date by a month until it reaches the end date .deduct the money paid for the basic plan from pro plans when user upgrade from basic plan .The result were ranked by start date for each customer and from these result “payments” table created

Full query



```
create table payments as (
with recursive cte  as (
 select customer_id ,s.plan_id,plan_name ,start_date,
 lead(s.start_date) over (partition by customer_id order by start_date,s.plan_id ) as end_date,price as amount
  from subscriptions s
 inner join plans p
 on s.plan_id = p.plan_id
 where year(start_date) ='2020' and plan_name not in ('trial','churn')
 ),
 cte1 as (
 select customer_id,plan_id,plan_name,start_date,
 coalesce(end_date,'2020-12-31') as end_date,amount
 from cte),
 cte2 as (
 select customer_id,plan_id,plan_name,date(start_date),
 coalesce(end_date,'2012-12-31') as end_date,amount
 from cte1
 union all
 select customer_id,plan_id,plan_name,
 date_add(start_date,interval 1 month) as start_date,
 date(end_date) as end_date ,amount
 from cte2
           where plan_name ='pro annual'and end_date > date_add(start_date,interval 1 month)),
  cte3 as
   (  select *,
   lag(plan_id) over(partition by customer_id order by start_date) as last_plan,
   lag(amount) over(partition by customer_id order by start_date) as last_amount_paid,
   rank() over(partition by customer_id order by start_date) as payment_order
   from cte2 )
                  select
                   customer_id,plan_id,plan_name,start_date,
                   (case when plan_id in (2,3) and last_plan =1 then amount - last_amount_paid  else amount end ) as amount,payment_order from cte3);
```

New generated payments table



customer_id	plan_id	plan_name	start_date	amount	payment_order
1	1	basic monthly	2020-08-08	9.90	1
2	3	pro annual	2020-09-27	199.00	1
2	3	pro annual	2020-10-27	199.00	2
2	3	pro annual	2020-11-27	199.00	3
2	3	pro annual	2020-12-27	199.00	4
3	1	basic monthly	2020-01-20	9.90	1
4	1	basic monthly	2020-01-24	9.90	1
5	1	basic monthly	2020-08-10	9.90	1
6	1	basic monthly	2020-12-30	9.90	1
7	1	basic monthly	2020-02-12	9.90	1
7	2	pro monthly	2020-05-22	10.00	2
8	1	basic monthly	2020-06-18	9.90	1
8	2	pro monthly	2020-08-03	10.00	2
9	3	pro annual	2020-12-14	199.00	1
10	2	pro monthly	2020-09-26	19.90	1
12	1	basic monthly	2020-09-29	9.90	1
13	1	basic monthly	2020-12-22	9.90	1
14	1	basic monthly	2020-09-29	9.90	1

THANK
YOU