Case Study #3 - Foodie-Fi

By Akshay Gupta

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

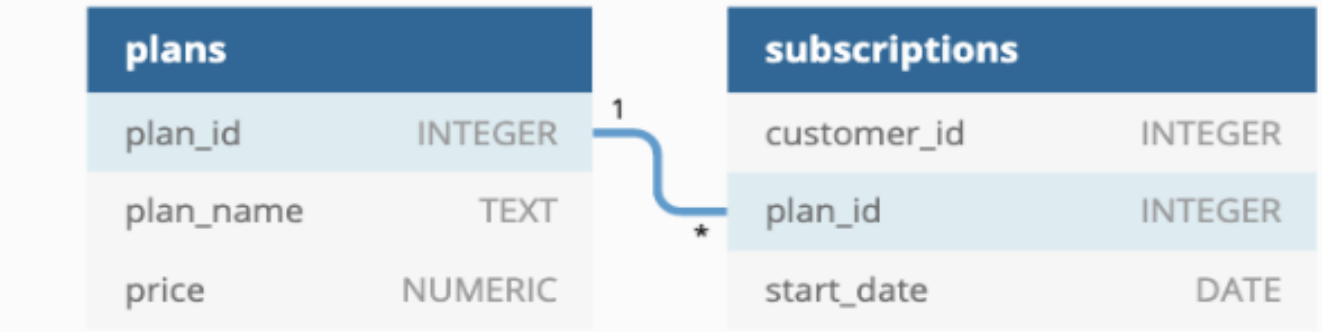


Table 1: plans

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: subscriptions

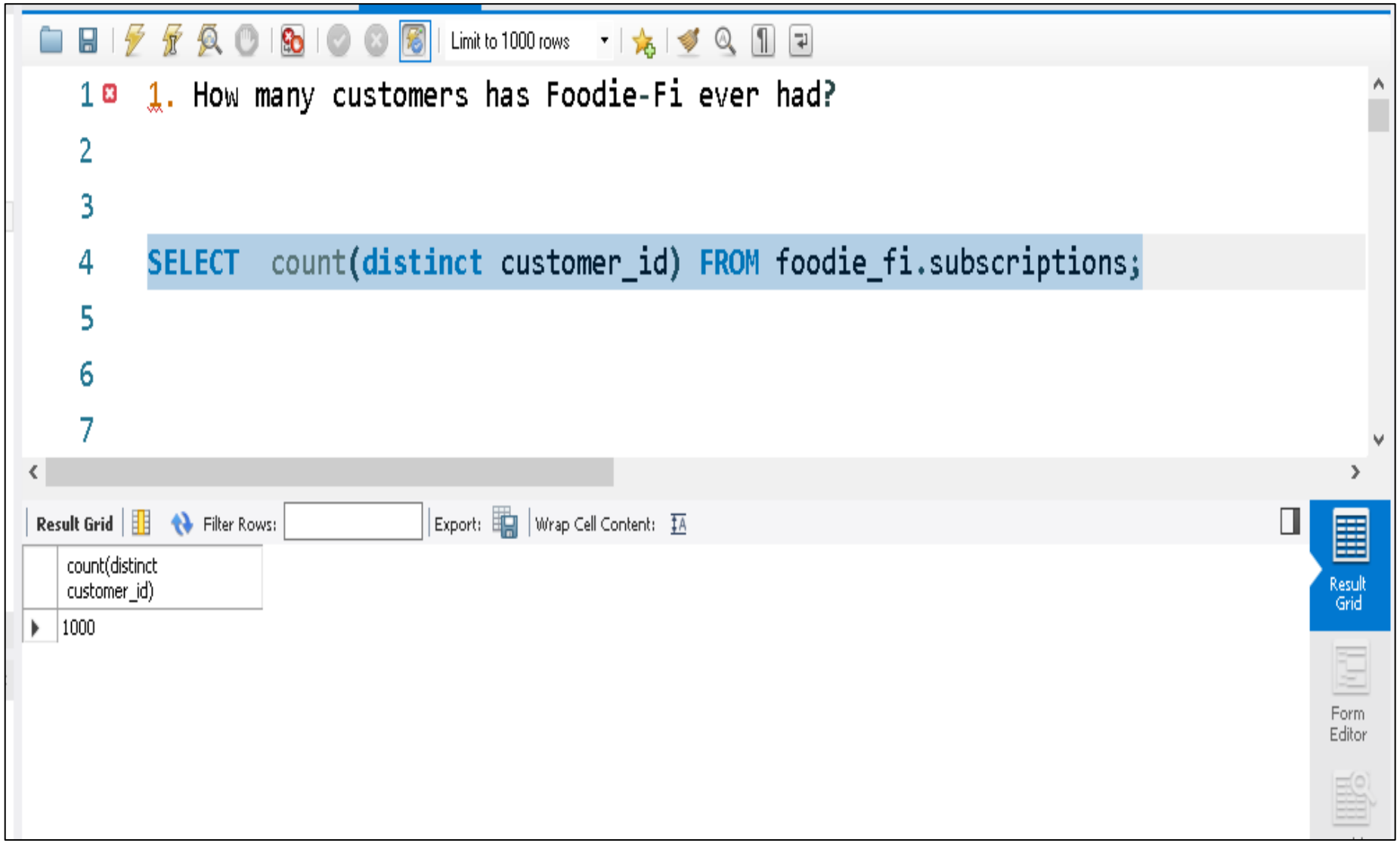
Customer subscriptions show the exact date where their specific `plan_id` starts.

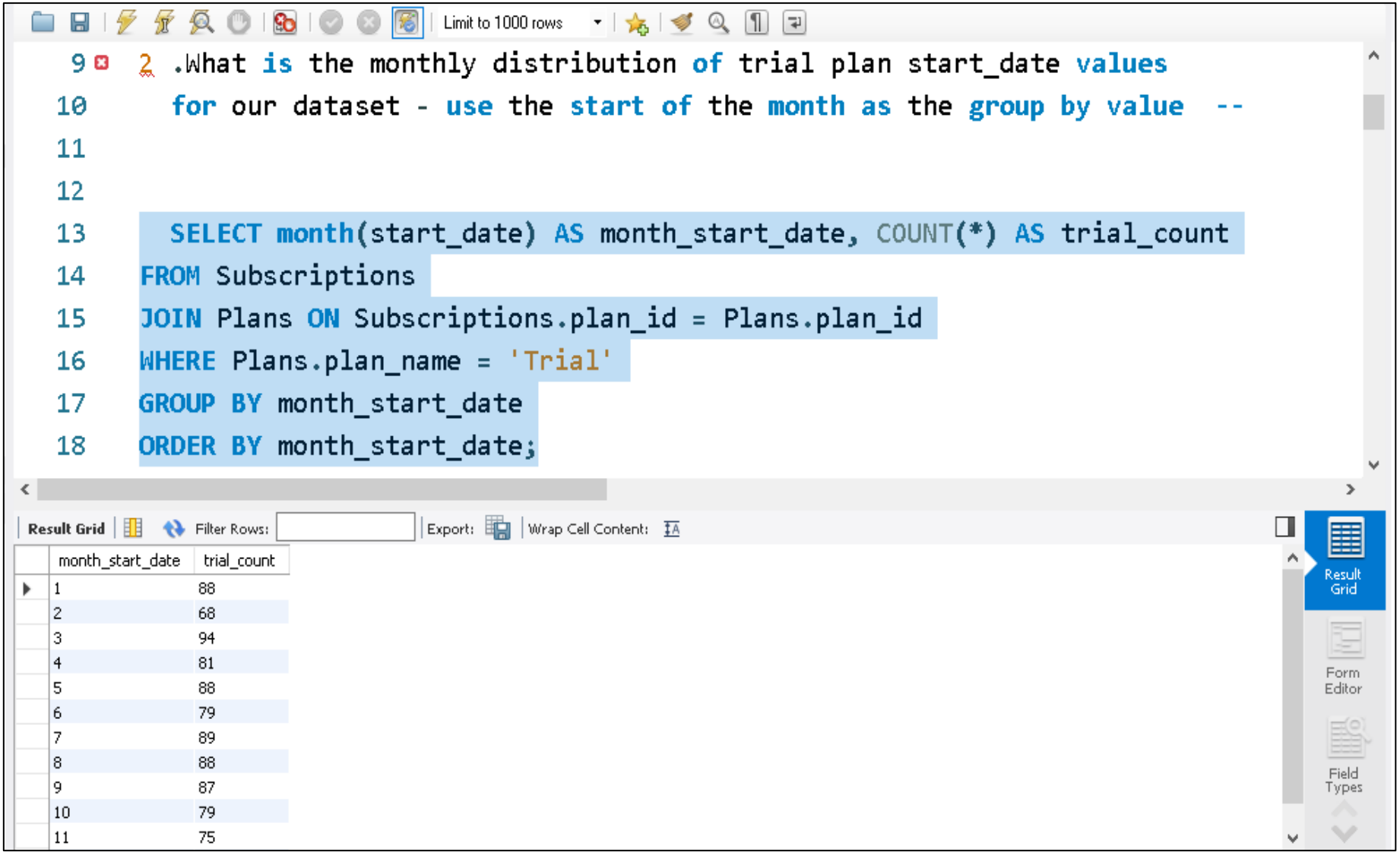
If customers downgrade from a pro plan or cancel their subscription - the higher plan will remain in place until the period is over - the `start_date` in the `subscriptions` table will reflect the date that the actual plan changes.

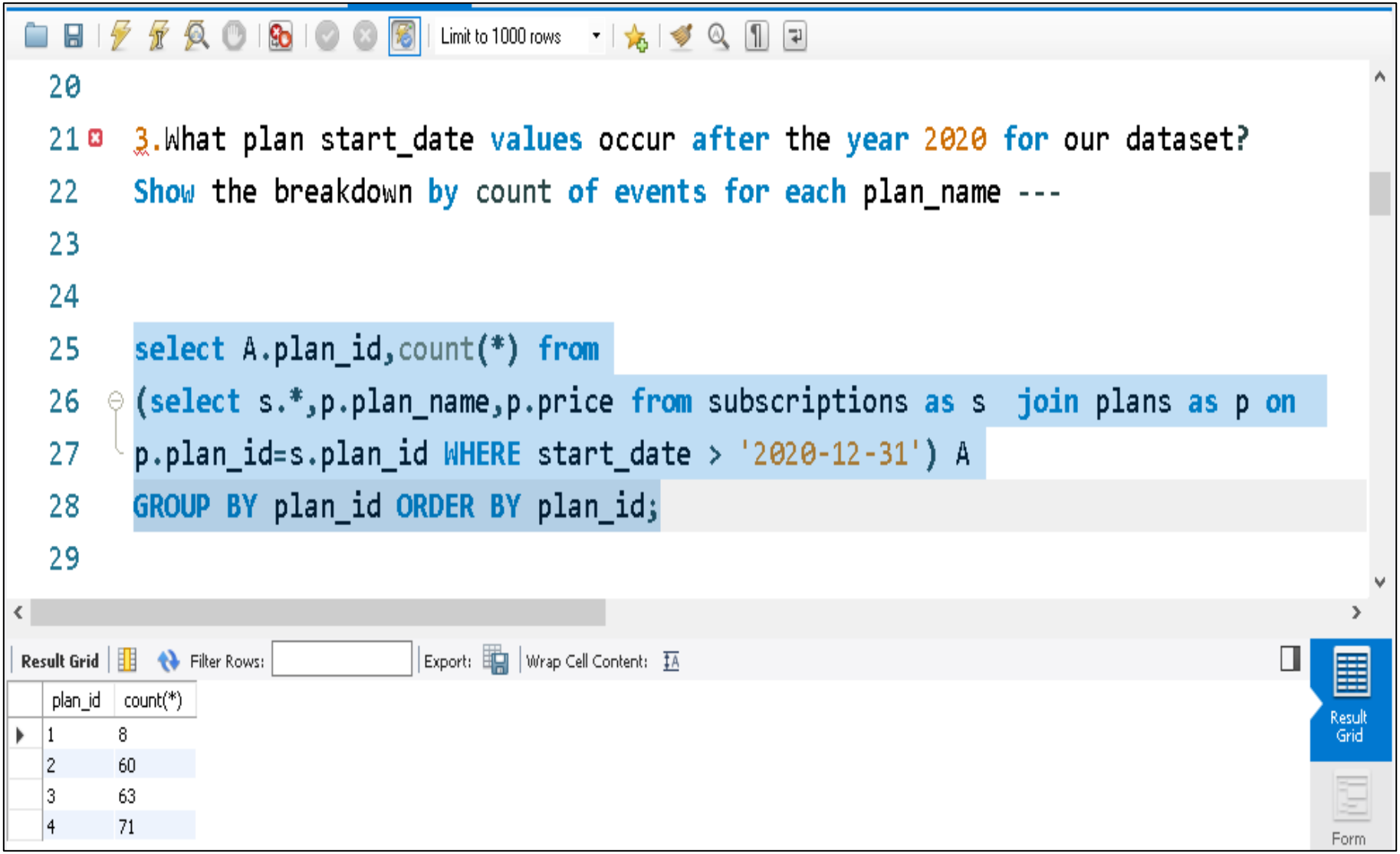
When customers upgrade their account from a basic plan to a pro or annual pro plan - the higher plan will take effect straightaway.

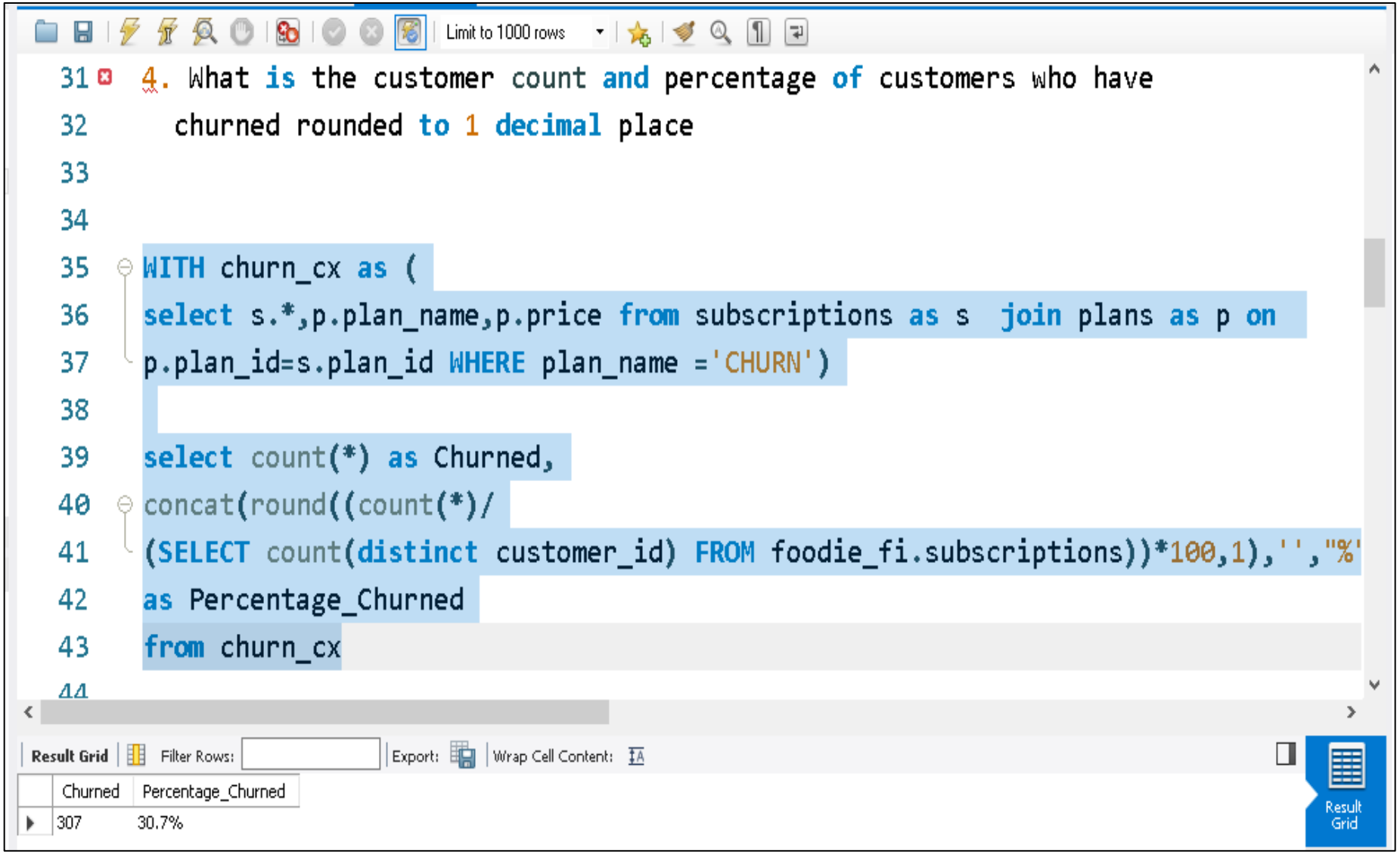
When customers churn - they will keep their access until the end of their current billing period but the `start_date` will be technically the day they decided to cancel their service.

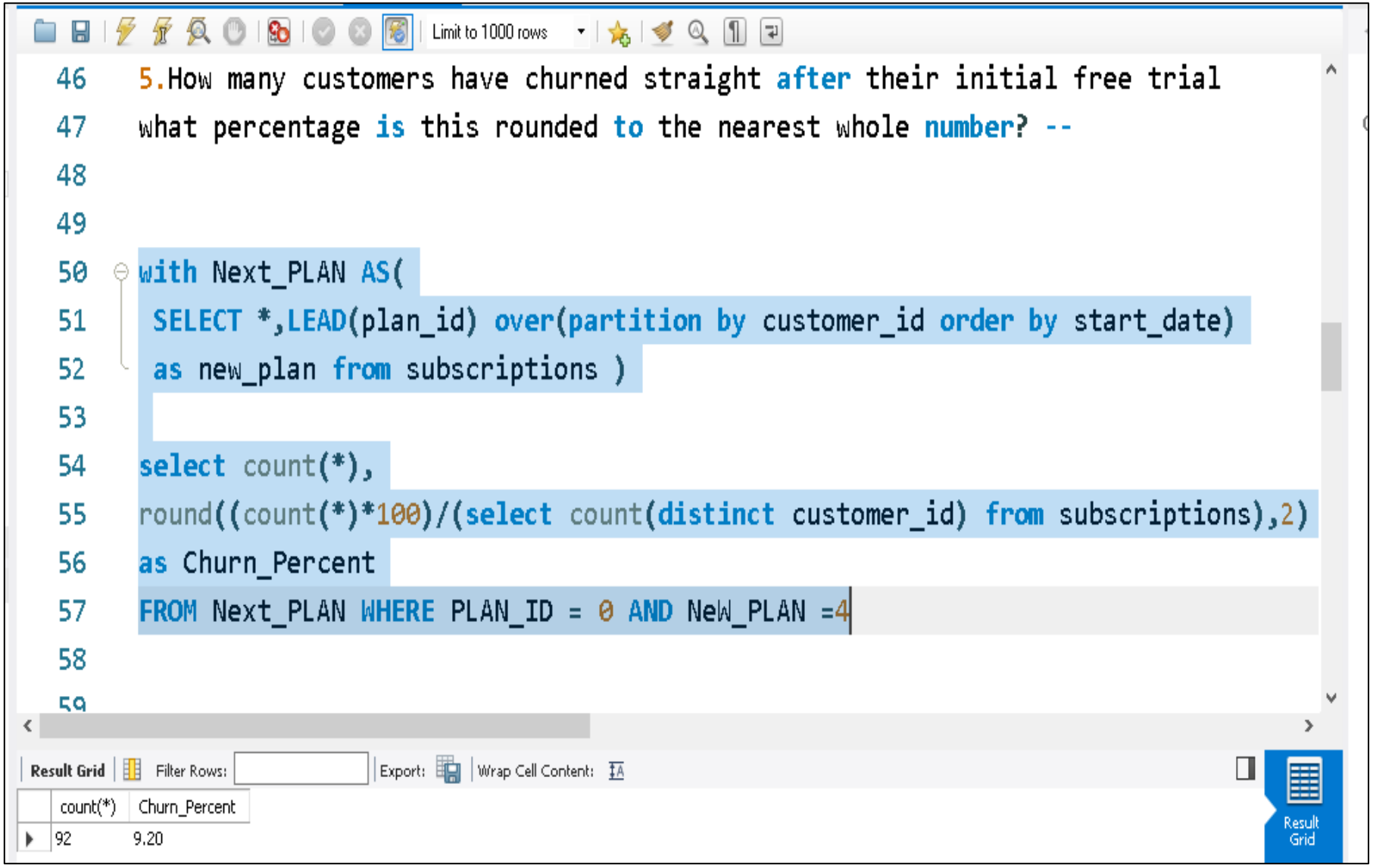
customer_id	plan_id	start_date
1	0	2020-08-01
1	1	2020-08-08
2	0	2020-09-20
2	3	2020-09-27
11	0	2020-11-19
11	4	2020-11-26
13	0	2020-12-15
13	1	2020-12-22
13	2	2021-03-29


Limit to 1000 rows
1. How many customers has Foodie-Fi ever had?
SELECT count(distinct customer_id) FROM foodie_fi.subscriptions;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
count(distinct customer_id)
1000
Result Grid
Form Editor


Limit to 1000 rows
9 2 .What is the monthly distribution of trial plan start_date values
10 for our dataset - use the start of the month as the group by value --
11
12
13 SELECT month(start_date) AS month_start_date, COUNT(*) AS trial_count
14 FROM Subscriptions
15 JOIN Plans ON Subscriptions.plan_id = Plans.plan_id
16 WHERE Plans.plan_name = 'Trial'
17 GROUP BY month_start_date
18 ORDER BY month_start_date;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
month_start_date
trial_count
1 88
2 68
3 94
4 81
5 88
6 79
7 89
8 88
9 87
10 79
11 75
Result Grid
Form Editor
Field Types


Limit to 1000 rows
20
21 3.What plan start_date values occur after the year 2020 for our dataset?
22 Show the breakdown by count of events for each plan_name ---
23
24
25 select A.plan_id,count(*) from
26 (select s.*,p.plan_name,p.price from subscriptions as s join plans as p on
27 p.plan_id=s.plan_id WHERE start_date > '2020-12-31') A
28 GROUP BY plan_id ORDER BY plan_id;
29
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
plan_id count(*)
1 8
2 60
3 63
4 71
Result Grid
Form


Limit to 1000 rows
4. What is the customer count and percentage of customers who have churned rounded to 1 decimal place
WITH churn_cx as (
select s.*,p.plan_name,p.price from subscriptions as s join plans as p on p.plan_id=s.plan_id WHERE plan_name ='CHURN')
select count(*) as Churned,
concat(round((count(*)/
(SELECT count(distinct customer_id) FROM foodie_fi.subscriptions))*100,1),'',"%"
as Percentage_Churned
from churn_cx
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Churned
Percentage_Churned
307
30.7%
Result Grid


Limit to 1000 rows
46 5.How many customers have churned straight after their initial free trial
47 what percentage is this rounded to the nearest whole number? --
48
49
50 with Next_PLAN AS(
51 SELECT *,LEAD(plan_id) over(partition by customer_id order by start_date)
52 as new_plan from subscriptions)
53
54 select count(*),
55 round((count(*)*100)/(select count(distinct customer_id) from subscriptions),2)
56 as Churn_Percent
57 FROM Next_PLAN WHERE PLAN_ID = 0 AND NeW_PLAN =4
58
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
count(*)
Churn_Percent
92
9.20
Result Grid