8WeekSQLCHALLENGE

Nourishing Growth: A Comprehensive Study on Foodie-Fi’s Subscription Dynamics

Fatih Sahin

https://8weeksqlchallenge.com/case-study-3/

AUGUST 2023

CONTENTS

Introduction

This case study revolves around **Foodie-Fi**, a subscription-based business offering streaming services specifically for food-related content. The business model is similar to Netflix, but is solely focused on cooking shows. The company offers various subscription plans: trial, basic monthly, pro monthly, pro annual, and churn (which represents cancellation).

I wrote this report after working on the "Foodie-Fi" SQL Challenge, which you can find at https://8weeksqlchallenge.com/case-study-3/. This challenge is a fun way to practice SQL using a made-up business.

For this study, I used Microsoft SQL Server Management Studio to write and test my SQL queries. Everything I've written here is based on what I've learned and practiced in SQL.

The purpose of this report is not only to document the findings and insights from the case study but also to serve as a valuable resource for other SQL learners and practitioners. The report demonstrates the power of SQL in deriving actionable business insights from raw data, providing a practical example of data-driven decision-making.

Problem Statement

Danny, the founder of "Foodie-Fi," aims to fine-tune his food streaming service by analyzing key business components:

- Evaluating subscription trends and revenue flow.
- Understanding the customer's journey from trial to paid plans.
- Analyzing churn rates and formulating strategies for customer retention.
- Assessing financial aspects related to upgrades, downgrades, and payment schedules.

The case study is structured into different segments, each probing specific facets of the business:

- **Subscription Metrics**: Understand the various plans chosen by customers, their upgrade patterns, and reasons for downgrading.

- **Customer Journey**: A deep dive into how customers move from a trial plan to a paid subscription or if they choose to opt-out.

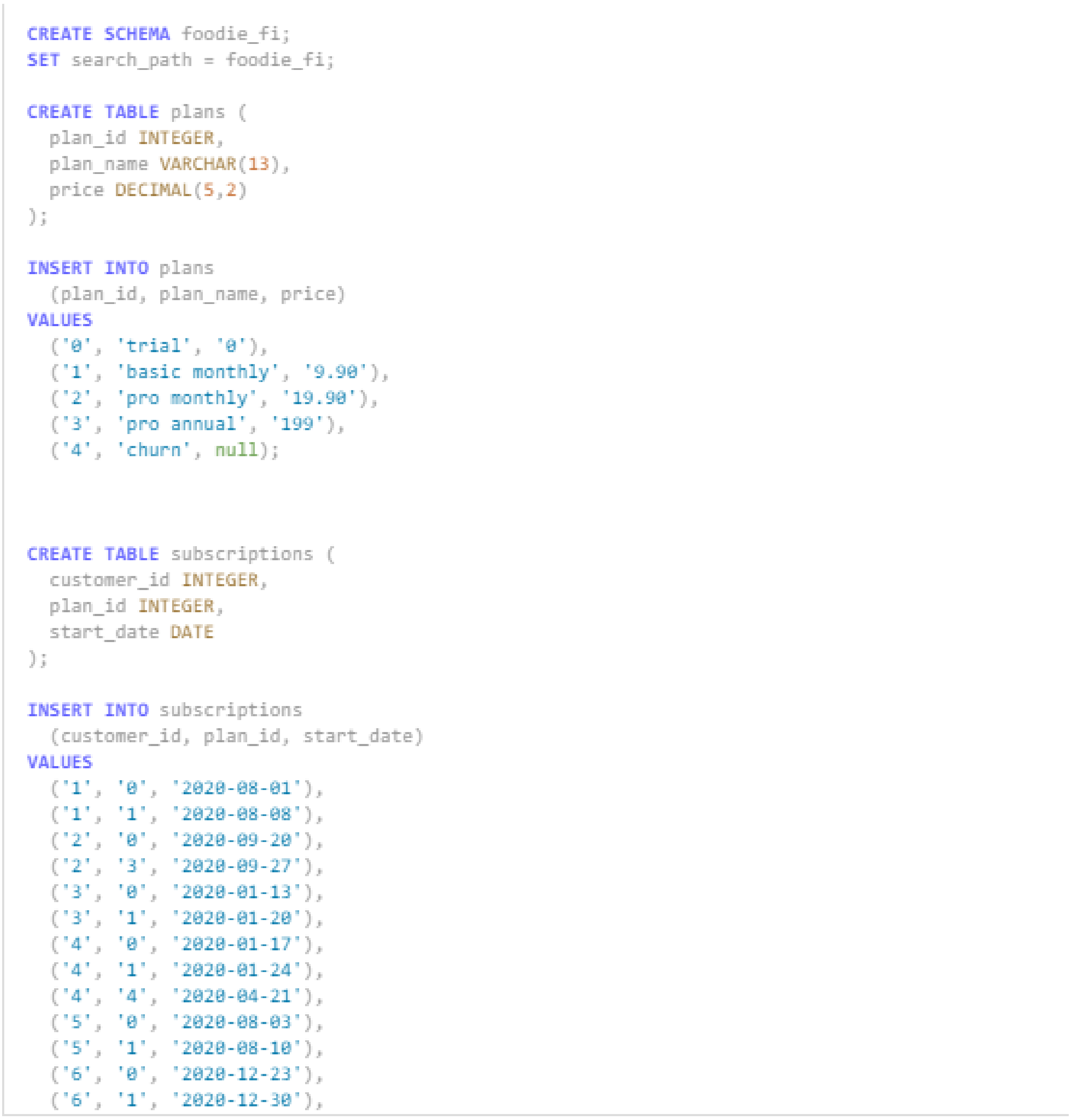
- **Financial Insights**: Examine the revenue from various subscription plans and the implications of plan changes on earnings.

- **Innovative Approaches**: Open-ended questions to explore growth strategies and potential improvements in the service.

The objective is to use SQL to extract meaningful insights from the provided data, offering actionable strategies for "Foodie-Fi" growth and optimization.

Creating Schema and Tables

All datasets exist within the `pizza_runner` database schema.



```
CREATE SCHEMA foodie_fi;
SET search_path = foodie_fi;

CREATE TABLE plans (
  plan_id INTEGER,
  plan_name VARCHAR(13),
  price DECIMAL(5,2)
);

INSERT INTO plans
  (plan_id, plan_name, price)
VALUES
  ('0', 'trial', '0'),
  ('1', 'basic monthly', '9.90'),
  ('2', 'pro monthly', '19.90'),
  ('3', 'pro annual', '199'),
  ('4', 'churn', null);



CREATE TABLE subscriptions (
  customer_id INTEGER,
  plan_id INTEGER,
  start_date DATE
);

INSERT INTO subscriptions
  (customer_id, plan_id, start_date)
VALUES
  ('1', '0', '2020-08-01'),
  ('1', '1', '2020-08-08'),
  ('2', '0', '2020-09-20'),
  ('2', '3', '2020-09-27'),
  ('3', '0', '2020-01-13'),
  ('3', '1', '2020-01-20'),
  ('4', '0', '2020-01-17'),
  ('4', '1', '2020-01-24'),
  ('4', '4', '2020-04-21'),
  ('5', '0', '2020-08-03'),
  ('5', '1', '2020-08-10'),
  ('6', '0', '2020-12-23'),
  ('6', '1', '2020-12-30'),
```

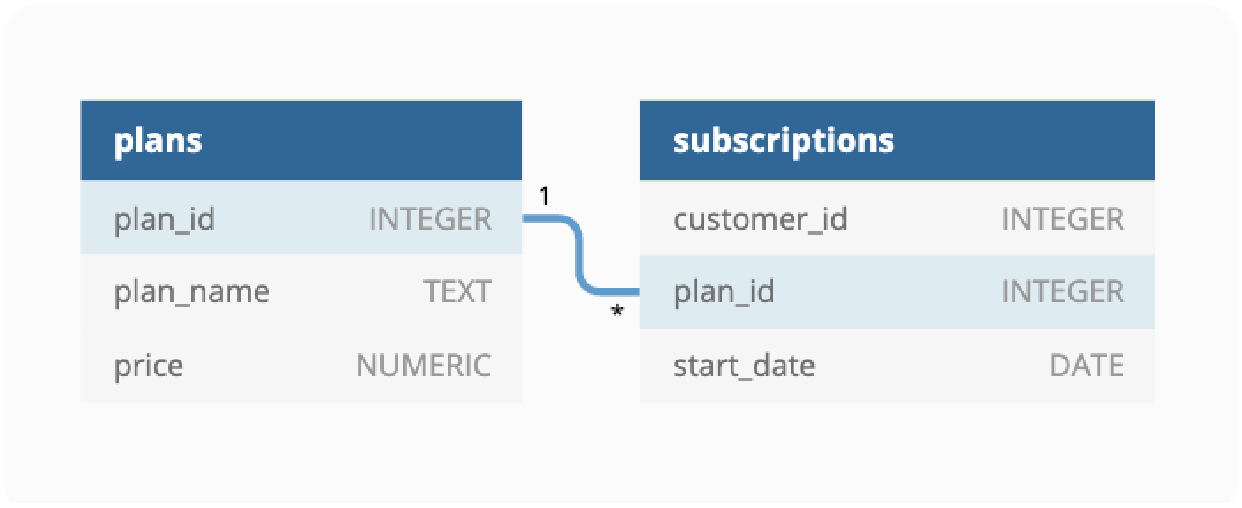
Entity Relationship Diagram

Data Structure

Foodie-Fi keeps track of its customer subscription data in two tables:

> **Plans**: This table includes details about the different types of subscription plans that customers can opt for. Each plan has a unique plan_id, a plan_name, and a price.
>
> **Subscriptions**: This table records every plan that customers subscribe to, with a customer_id to identify each customer, the plan_id that they subscribed to, and the start_date of the plan. If a customer changes their plan, a new record is added to this table.

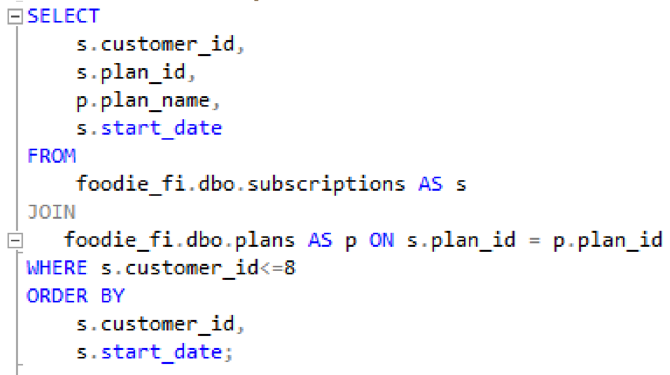
Entity Relationship Diagram



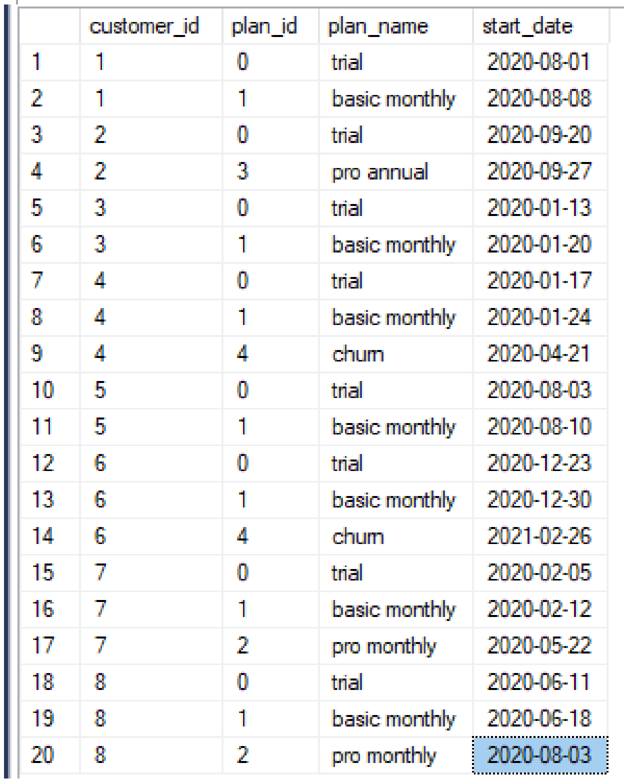
Case Study Questions

A. Customer Journey

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



```
SELECT
    s.customer_id,
    s.plan_id,
    p.plan_name,
    s.start_date
FROM
    foodie_fi.dbo.subscriptions AS s
JOIN
   foodie_fi.dbo.plans AS p ON s.plan_id = p.plan_id
WHERE s.customer_id<=8
ORDER BY
    s.customer_id,
    s.start_date;
```



	customer_id	plan_id	plan_name	start_date
1	1	0	trial	2020-08-01
2	1	1	basic monthly	2020-08-08
3	2	0	trial	2020-09-20
4	2	3	pro annual	2020-09-27
5	3	0	trial	2020-01-13
6	3	1	basic monthly	2020-01-20
7	4	0	trial	2020-01-17
8	4	1	basic monthly	2020-01-24
9	4	4	churn	2020-04-21
10	5	0	trial	2020-08-03
11	5	1	basic monthly	2020-08-10
12	6	0	trial	2020-12-23
13	6	1	basic monthly	2020-12-30
14	6	4	churn	2021-02-26
15	7	0	trial	2020-02-05
16	7	1	basic monthly	2020-02-12
17	7	2	pro monthly	2020-05-22
18	8	0	trial	2020-06-11
19	8	1	basic monthly	2020-06-18
20	8	2	pro monthly	2020-08-03

Customer 1

- Started with a trial on 2020-08-01.
- Upgraded to basic monthly on 2020-08-08.

Summary: Customer 1 tried the service and decided to continue with a basic monthly plan a week later.

Customer 2

- Started with a trial on 2020-09-20.
- Upgraded to pro annual on 2020-09-27.

Summary: Customer 2 started with a trial and opted for the pro annual plan within a week.

Customer 3

- Began with a trial on 2020-01-13.
- Shifted to basic monthly on 2020-01-20.

Summary: Customer 3 tested the service and then continued with a basic monthly subscription after a week.

Customer 4

- Took the trial on 2020-01-17.
- Moved to basic monthly on 2020-01-24.
- Churned (canceled) on 2020-04-21.

Summary: Customer 4 started with a trial, went for the basic monthly plan, but decided to cancel after a few months.

Customer 5

- Started with a trial on 2020-08-03.
- Shifted to basic monthly on 2020-08-10.

Summary: Customer 5 tried the service and then continued with a basic monthly plan.

Customer 6

- Began with a trial on 2020-12-23.
- Upgraded to basic monthly on 2020-12-30.
- Churned (canceled) on 2021-02-26.

Summary: Customer 6 took a trial, upgraded to a basic plan, but then canceled after a couple of months.

Customer 7

- Started with a trial on 2020-02-05.
- Continued with basic monthly on 2020-02-12.
- Upgraded to pro monthly on 2020-05-22.

Summary: Customer 7 began with a trial, chose the basic plan, but later decided to upgrade to the pro monthly plan.

Customer 8

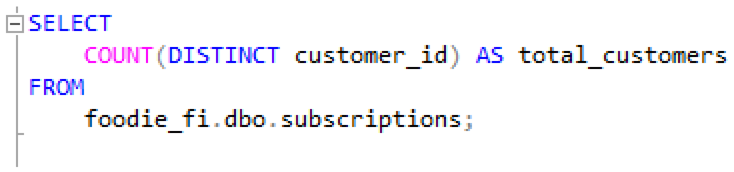
- Took the trial on 2020-06-11.

- Opted for basic monthly on 2020-06-18.
- Upgraded to pro monthly on 2020-08-03.

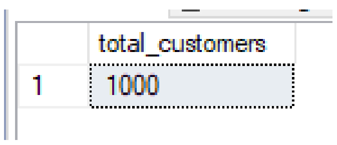
Summary: Customer 8 started with a trial, selected the basic monthly plan, and after a while, decided to upgrade to the pro monthly plan.

B. Data Analysis Questions

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

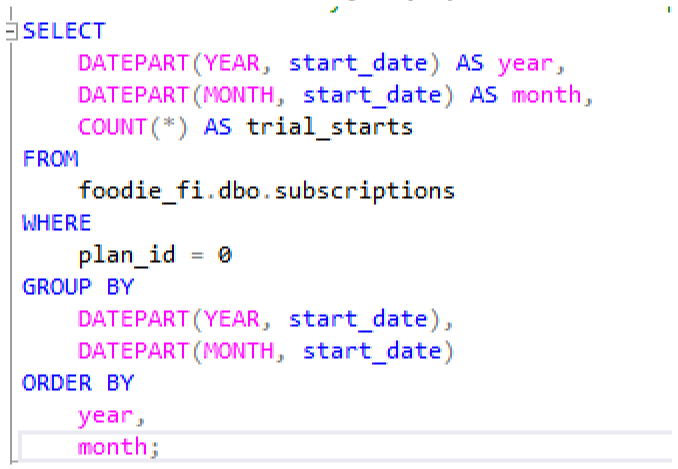


```sql
SELECT
    COUNT(DISTINCT customer_id) AS total_customers
FROM
    foodie_fi.dbo.subscriptions;
```

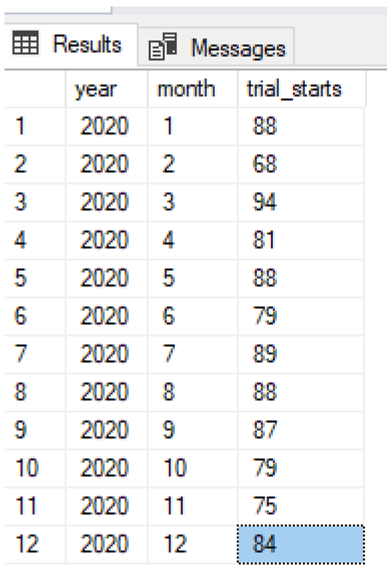


	total_customers
1	1000

3. 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



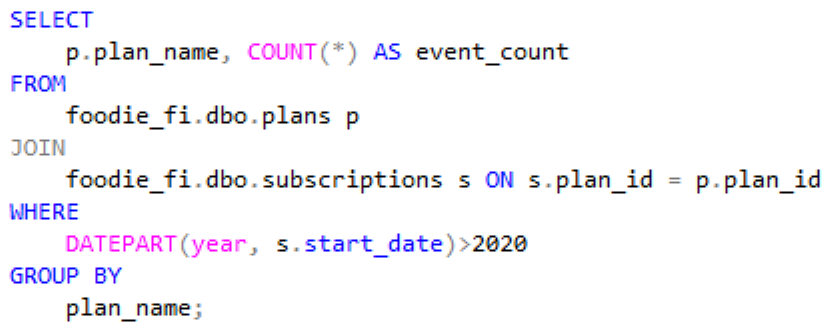
```sql
SELECT
    DATEPART(YEAR, start_date) AS year,
    DATEPART(MONTH, start_date) AS month,
    COUNT(*) AS trial_starts
FROM
    foodie_fi.dbo.subscriptions
WHERE
    plan_id = 0
GROUP BY
    DATEPART(YEAR, start_date),
    DATEPART(MONTH, start_date)
ORDER BY
    year,
    month;
```



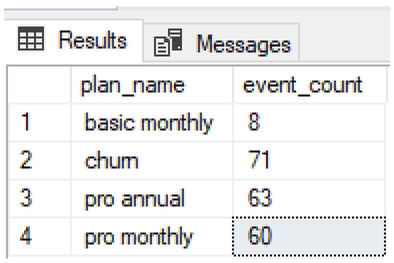
Results | Messages

	year	month	trial_starts
1	2020	1	88
2	2020	2	68
3	2020	3	94
4	2020	4	81
5	2020	5	88
6	2020	6	79
7	2020	7	89
8	2020	8	88
9	2020	9	87
10	2020	10	79
11	2020	11	75
12	2020	12	84

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



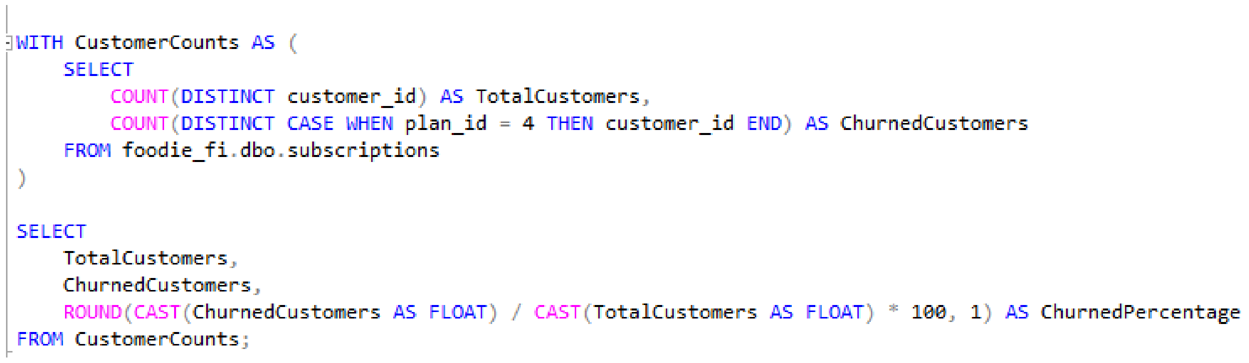
```sql
SELECT
    p.plan_name, COUNT(*) AS event_count
FROM
    foodie_fi.dbo.plans p
JOIN
    foodie_fi.dbo.subscriptions s ON s.plan_id = p.plan_id
WHERE
    DATEPART(year, s.start_date)>2020
GROUP BY
    plan_name;
```



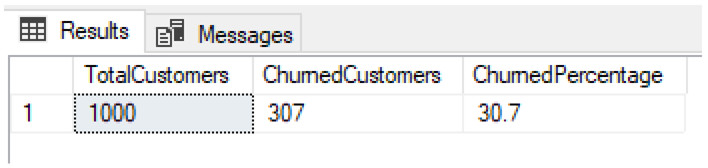
Results | Messages

	plan_name	event_count
1	basic monthly	8
2	churn	71
3	pro annual	63
4	pro monthly	60

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

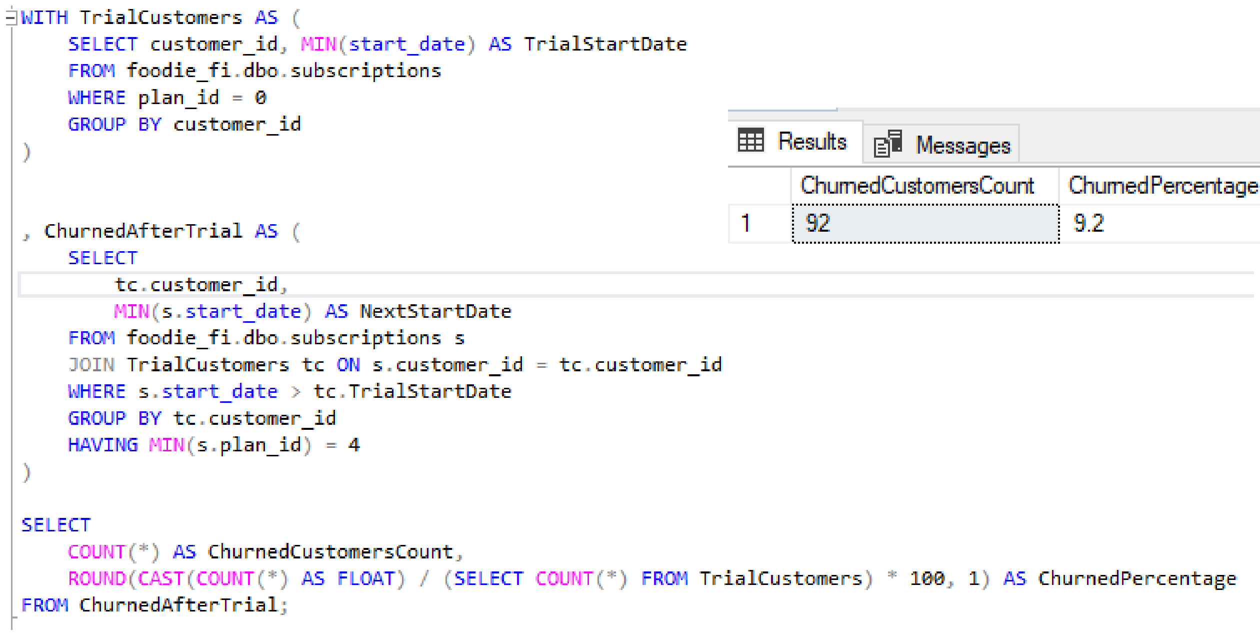


```sql
WITH CustomerCounts AS (
    SELECT
        COUNT(DISTINCT customer_id) AS TotalCustomers,
        COUNT(DISTINCT CASE WHEN plan_id = 4 THEN customer_id END) AS ChurnedCustomers
    FROM foodie_fi.dbo.subscriptions
)

SELECT
    TotalCustomers,
    ChurnedCustomers,
    ROUND(CAST(ChurnedCustomers AS FLOAT) / CAST(TotalCustomers AS FLOAT) * 100, 1) AS ChurnedPercentage
FROM CustomerCounts;
```

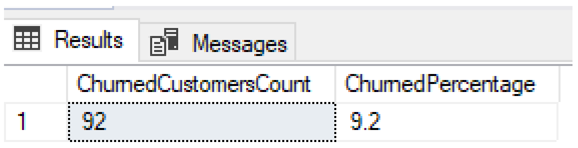


	TotalCustomers	ChurnedCustomers	ChurnedPercentage
1	1000	307	30.7

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

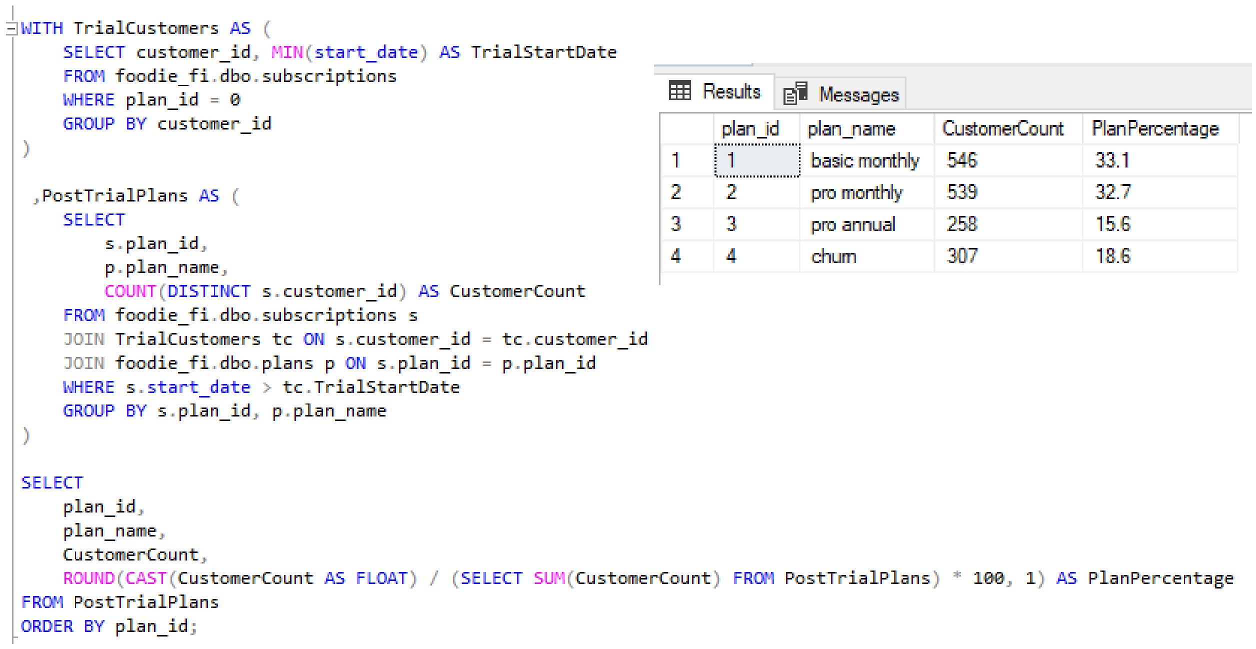


```sql
WITH TrialCustomers AS (
    SELECT customer_id, MIN(start_date) AS TrialStartDate
    FROM foodie_fi.dbo.subscriptions
    WHERE plan_id = 0
    GROUP BY customer_id
)

, ChurnedAfterTrial AS (
    SELECT
        tc.customer_id,
        MIN(s.start_date) AS NextStartDate
    FROM foodie_fi.dbo.subscriptions s
    JOIN TrialCustomers tc ON s.customer_id = tc.customer_id
    WHERE s.start_date > tc.TrialStartDate
    GROUP BY tc.customer_id
    HAVING MIN(s.plan_id) = 4
)

SELECT
    COUNT(*) AS ChurnedCustomersCount,
    ROUND(CAST(COUNT(*) AS FLOAT) / (SELECT COUNT(*) FROM TrialCustomers) * 100, 1) AS ChurnedPercentage
FROM ChurnedAfterTrial;
```

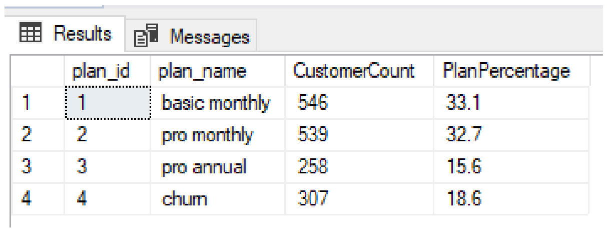


	ChurnedCustomersCount	ChurnedPercentage
1	92	9.2

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



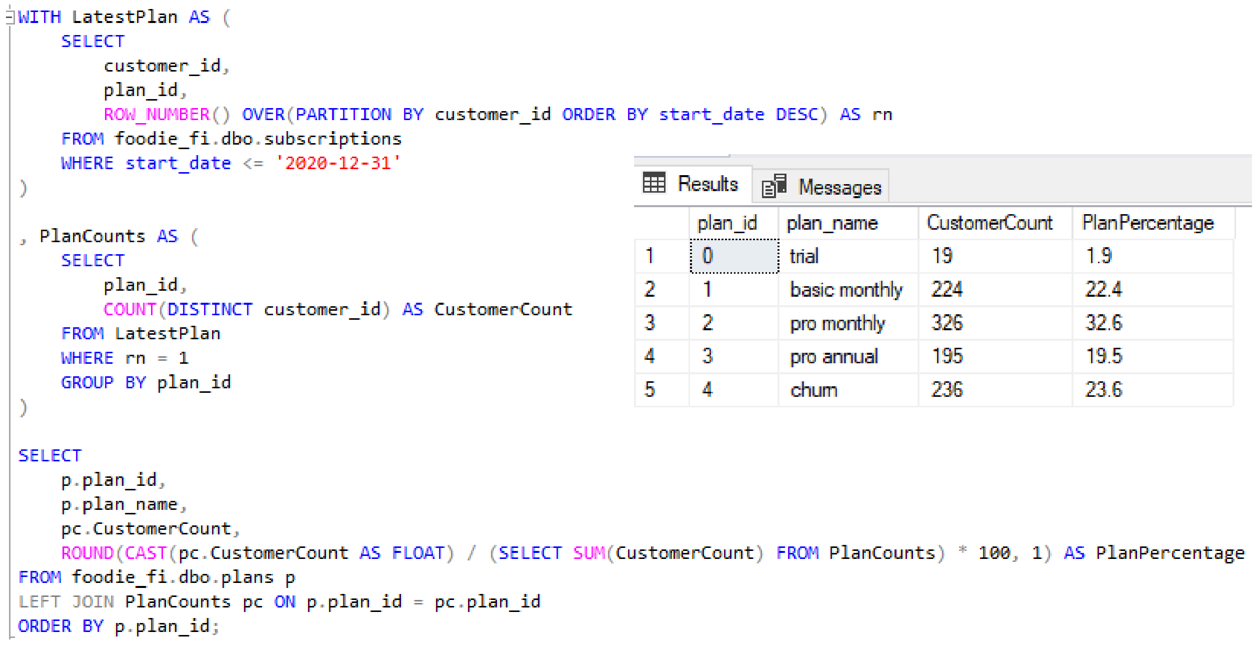
```
WITH TrialCustomers AS (
    SELECT customer_id, MIN(start_date) AS TrialStartDate
    FROM foodie_fi.dbo.subscriptions
    WHERE plan_id = 0
    GROUP BY customer_id
)

 ,PostTrialPlans AS (
    SELECT
        s.plan_id,
        p.plan_name,
        COUNT(DISTINCT s.customer_id) AS CustomerCount
    FROM foodie_fi.dbo.subscriptions s
    JOIN TrialCustomers tc ON s.customer_id = tc.customer_id
    JOIN foodie_fi.dbo.plans p ON s.plan_id = p.plan_id
    WHERE s.start_date > tc.TrialStartDate
    GROUP BY s.plan_id, p.plan_name
)

SELECT
    plan_id,
    plan_name,
    CustomerCount,
    ROUND(CAST(CustomerCount AS FLOAT) / (SELECT SUM(CustomerCount) FROM PostTrialPlans) * 100, 1) AS PlanPercentage
FROM PostTrialPlans
ORDER BY plan_id;
```



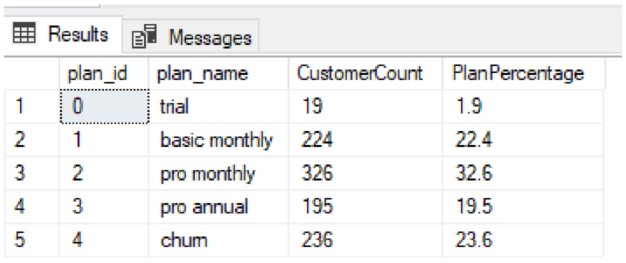
Results | Messages

	plan_id	plan_name	CustomerCount	PlanPercentage
1	1	basic monthly	546	33.1
2	2	pro monthly	539	32.7
3	3	pro annual	258	15.6
4	4	churn	307	18.6

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



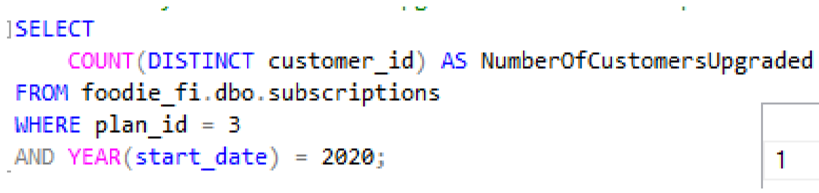
```
WITH LatestPlan AS (
    SELECT
        customer_id,
        plan_id,
        ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY start_date DESC) AS rn
    FROM foodie_fi.dbo.subscriptions
    WHERE start_date <= '2020-12-31'
)

, PlanCounts AS (
    SELECT
        plan_id,
        COUNT(DISTINCT customer_id) AS CustomerCount
    FROM LatestPlan
    WHERE rn = 1
    GROUP BY plan_id
)

SELECT
    p.plan_id,
    p.plan_name,
    pc.CustomerCount,
    ROUND(CAST(pc.CustomerCount AS FLOAT) / (SELECT SUM(CustomerCount) FROM PlanCounts) * 100, 1) AS PlanPercentage
FROM foodie_fi.dbo.plans p
LEFT JOIN PlanCounts pc ON p.plan_id = pc.plan_id
ORDER BY p.plan_id;
```



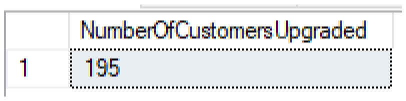
Results | Messages

	plan_id	plan_name	CustomerCount	PlanPercentage
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

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

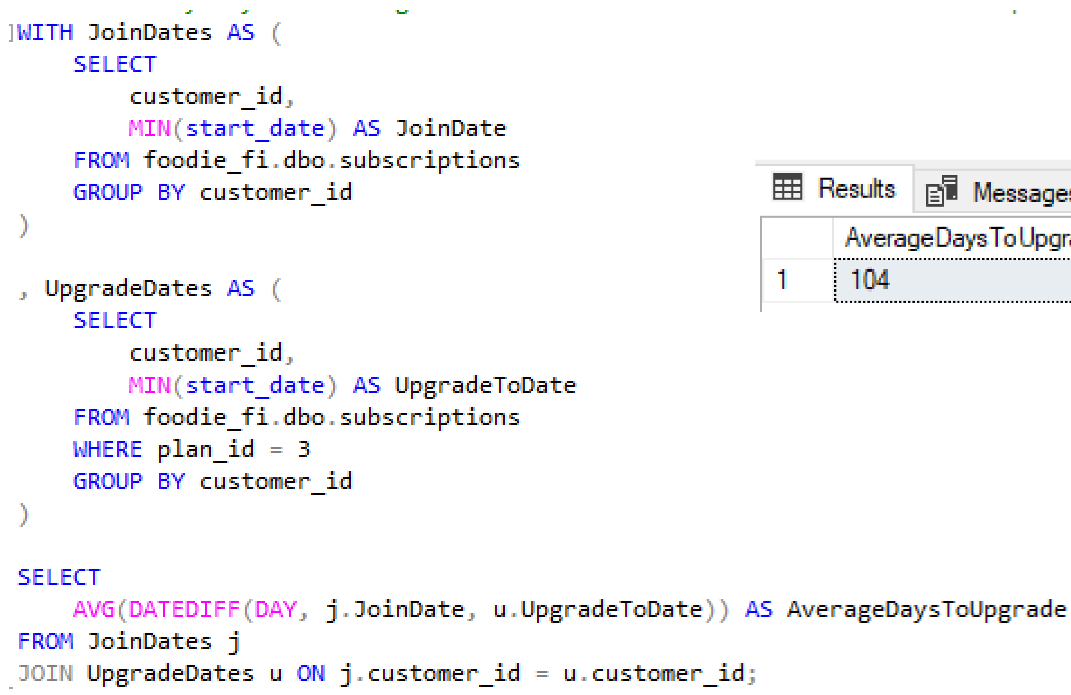


```sql
SELECT
    COUNT(DISTINCT customer_id) AS NumberOfCustomersUpgraded
FROM foodie_fi.dbo.subscriptions
WHERE plan_id = 3
AND YEAR(start_date) = 2020;
```

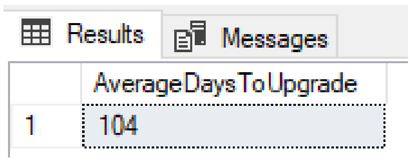


	NumberOfCustomersUpgraded
1	195

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

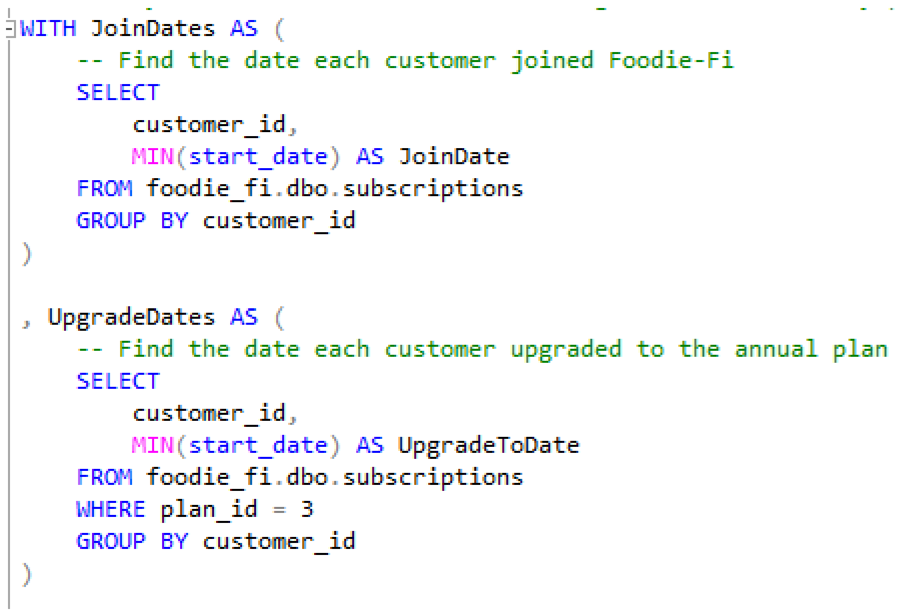


```sql
WITH JoinDates AS (
    SELECT
        customer_id,
        MIN(start_date) AS JoinDate
    FROM foodie_fi.dbo.subscriptions
    GROUP BY customer_id
)

, UpgradeDates AS (
    SELECT
        customer_id,
        MIN(start_date) AS UpgradeToDate
    FROM foodie_fi.dbo.subscriptions
    WHERE plan_id = 3
    GROUP BY customer_id
)

SELECT
    AVG(DATEDIFF(DAY, j.JoinDate, u.UpgradeToDate)) AS AverageDaysToUpgrade
FROM JoinDates j
JOIN UpgradeDates u ON j.customer_id = u.customer_id;
```

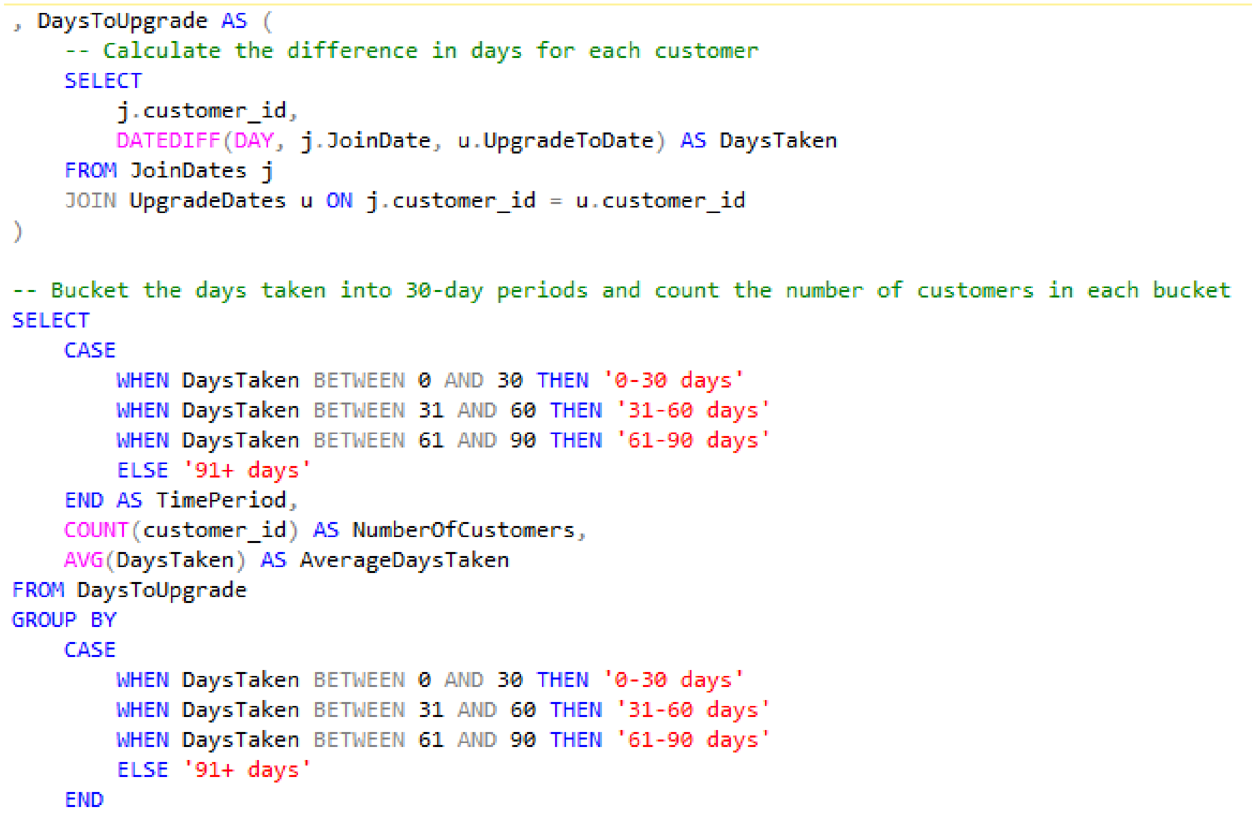


	AverageDaysToUpgrade
1	104

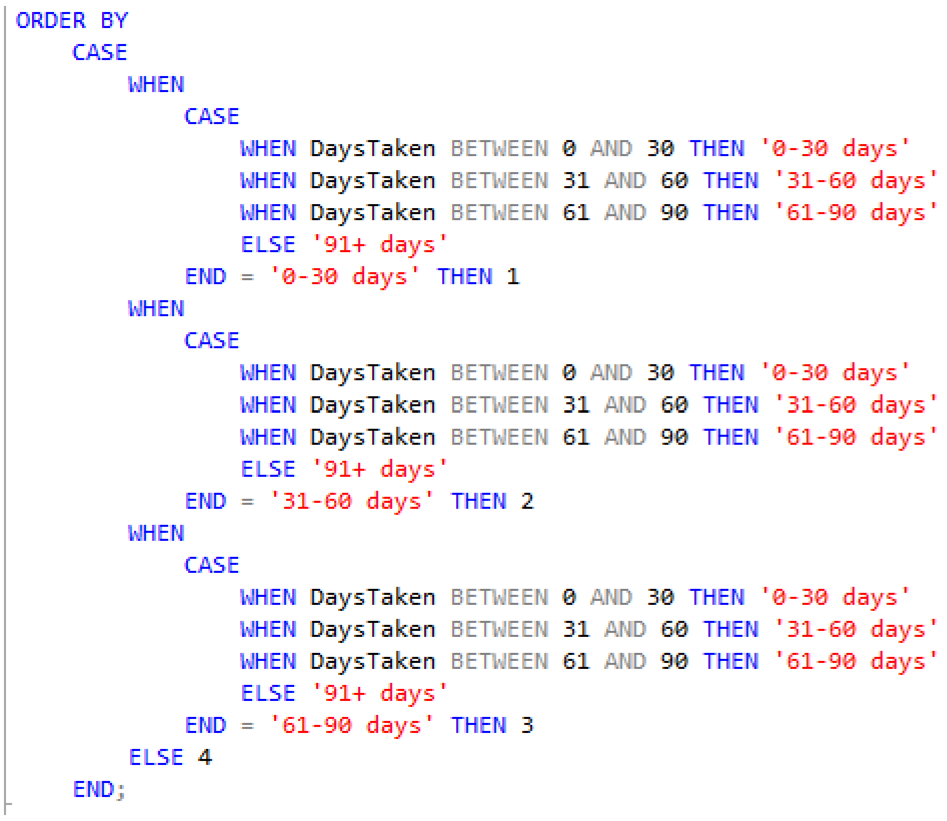
11.Can you further breakdown this average value into 30 day periods (i.e. 0-30 days, 31-60 days etc)



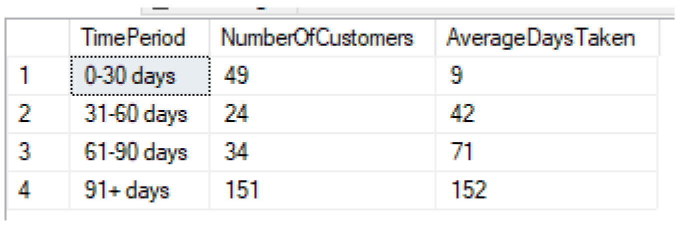
```sql
WITH JoinDates AS (
    -- Find the date each customer joined Foodie-Fi
    SELECT
        customer_id,
        MIN(start_date) AS JoinDate
    FROM foodie_fi.dbo.subscriptions
    GROUP BY customer_id
)

, UpgradeDates AS (
    -- Find the date each customer upgraded to the annual plan
    SELECT
        customer_id,
        MIN(start_date) AS UpgradeToDate
    FROM foodie_fi.dbo.subscriptions
    WHERE plan_id = 3
    GROUP BY customer_id
)
```



```sql
, DaysToUpgrade AS (
    -- Calculate the difference in days for each customer
    SELECT
        j.customer_id,
        DATEDIFF(DAY, j.JoinDate, u.UpgradeToDate) AS DaysTaken
    FROM JoinDates j
    JOIN UpgradeDates u ON j.customer_id = u.customer_id
)

-- Bucket the days taken into 30-day periods and count the number of customers in each bucket
SELECT
    CASE
        WHEN DaysTaken BETWEEN 0 AND 30 THEN '0-30 days'
        WHEN DaysTaken BETWEEN 31 AND 60 THEN '31-60 days'
        WHEN DaysTaken BETWEEN 61 AND 90 THEN '61-90 days'
        ELSE '91+ days'
    END AS TimePeriod,
    COUNT(customer_id) AS NumberOfCustomers,
    AVG(DaysTaken) AS AverageDaysTaken
FROM DaysToUpgrade
GROUP BY
    CASE
        WHEN DaysTaken BETWEEN 0 AND 30 THEN '0-30 days'
        WHEN DaysTaken BETWEEN 31 AND 60 THEN '31-60 days'
        WHEN DaysTaken BETWEEN 61 AND 90 THEN '61-90 days'
        ELSE '91+ days'
    END
```

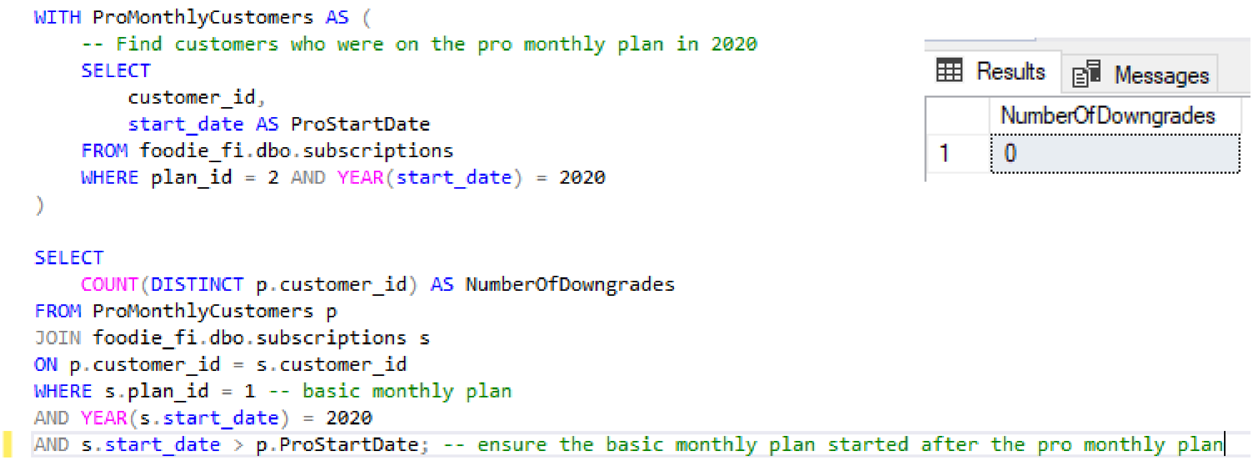


```
ORDER BY
    CASE
        WHEN
            CASE
                WHEN DaysTaken BETWEEN 0 AND 30 THEN '0-30 days'
                WHEN DaysTaken BETWEEN 31 AND 60 THEN '31-60 days'
                WHEN DaysTaken BETWEEN 61 AND 90 THEN '61-90 days'
                ELSE '91+ days'
            END = '0-30 days' THEN 1
        WHEN
            CASE
                WHEN DaysTaken BETWEEN 0 AND 30 THEN '0-30 days'
                WHEN DaysTaken BETWEEN 31 AND 60 THEN '31-60 days'
                WHEN DaysTaken BETWEEN 61 AND 90 THEN '61-90 days'
                ELSE '91+ days'
            END = '31-60 days' THEN 2
        WHEN
            CASE
                WHEN DaysTaken BETWEEN 0 AND 30 THEN '0-30 days'
                WHEN DaysTaken BETWEEN 31 AND 60 THEN '31-60 days'
                WHEN DaysTaken BETWEEN 61 AND 90 THEN '61-90 days'
                ELSE '91+ days'
            END = '61-90 days' THEN 3
        ELSE 4
    END;
```

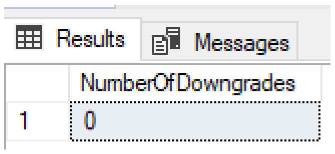


	TimePeriod	NumberOfCustomers	AverageDaysTaken
1	0-30 days	49	9
2	31-60 days	24	42
3	61-90 days	34	71
4	91+ days	151	152

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



```
WITH ProMonthlyCustomers AS (
    -- Find customers who were on the pro monthly plan in 2020
    SELECT
        customer_id,
        start_date AS ProStartDate
    FROM foodie_fi.dbo.subscriptions
    WHERE plan_id = 2 AND YEAR(start_date) = 2020
)

SELECT
    COUNT(DISTINCT p.customer_id) AS NumberOfDowngrades
FROM ProMonthlyCustomers p
JOIN foodie_fi.dbo.subscriptions s
ON p.customer_id = s.customer_id
WHERE s.plan_id = 1 -- basic monthly plan
AND YEAR(s.start_date) = 2020
AND s.start_date > p.ProStartDate; -- ensure the basic monthly plan started after the pro monthly plan
```



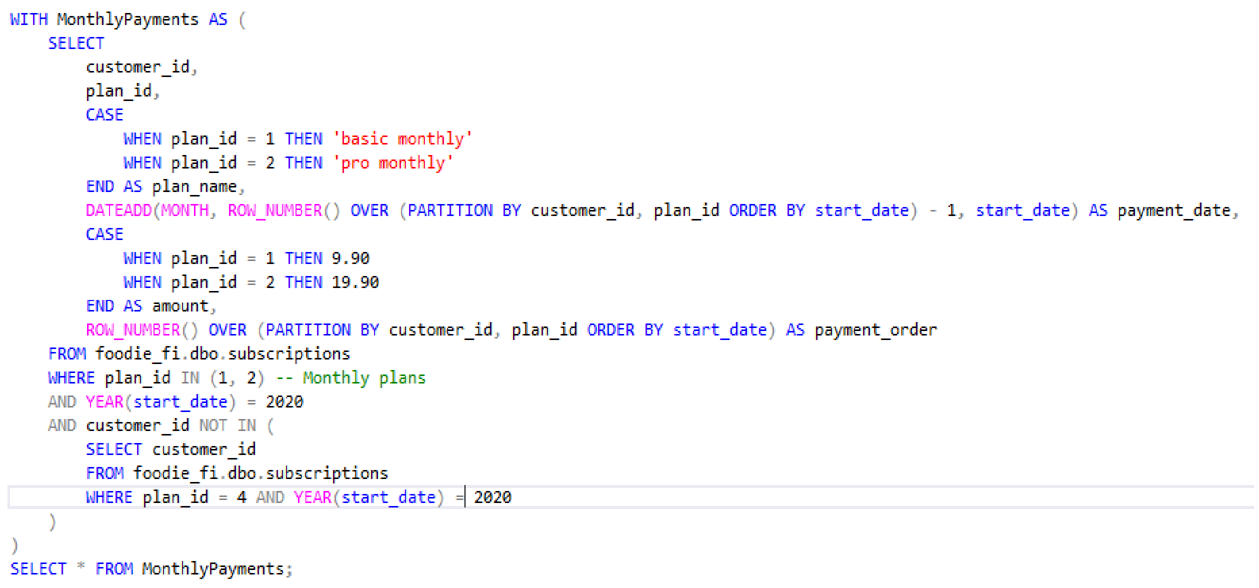
Results   Messages

	NumberOfDowngrades
1	0

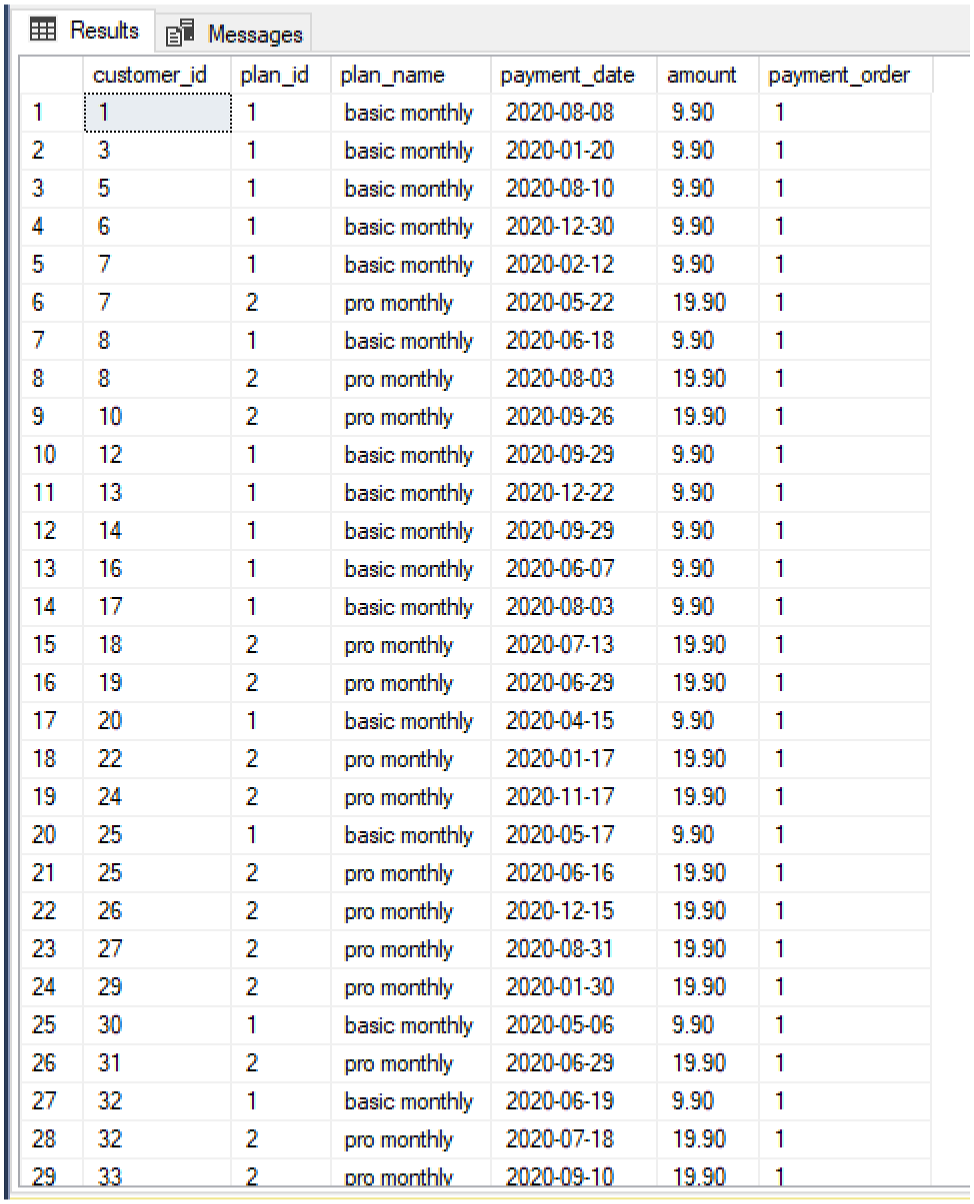
C. Challenge Payment Question

1.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



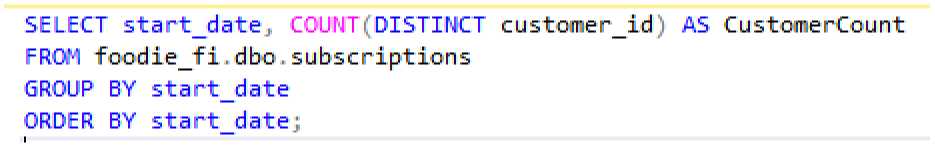
```sql
WITH MonthlyPayments AS (
    SELECT
        customer_id,
        plan_id,
        CASE
            WHEN plan_id = 1 THEN 'basic monthly'
            WHEN plan_id = 2 THEN 'pro monthly'
        END AS plan_name,
        DATEADD(MONTH, ROW_NUMBER() OVER (PARTITION BY customer_id, plan_id ORDER BY start_date) - 1, start_date) AS payment_date,
        CASE
            WHEN plan_id = 1 THEN 9.90
            WHEN plan_id = 2 THEN 19.90
        END AS amount,
        ROW_NUMBER() OVER (PARTITION BY customer_id, plan_id ORDER BY start_date) AS payment_order
    FROM foodie_fi.dbo.subscriptions
    WHERE plan_id IN (1, 2) -- Monthly plans
    AND YEAR(start_date) = 2020
    AND customer_id NOT IN (
        SELECT customer_id
        FROM foodie_fi.dbo.subscriptions
        WHERE plan_id = 4 AND YEAR(start_date) = 2020
    )
)
SELECT * FROM MonthlyPayments;
```


Results Messages

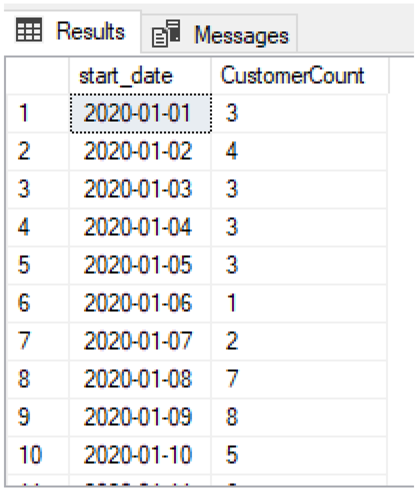
	customer_id	plan_id	plan_name	payment_date	amount	payment_order
1	1	1	basic monthly	2020-08-08	9.90	1
2	3	1	basic monthly	2020-01-20	9.90	1
3	5	1	basic monthly	2020-08-10	9.90	1
4	6	1	basic monthly	2020-12-30	9.90	1
5	7	1	basic monthly	2020-02-12	9.90	1
6	7	2	pro monthly	2020-05-22	19.90	1
7	8	1	basic monthly	2020-06-18	9.90	1
8	8	2	pro monthly	2020-08-03	19.90	1
9	10	2	pro monthly	2020-09-26	19.90	1
10	12	1	basic monthly	2020-09-29	9.90	1
11	13	1	basic monthly	2020-12-22	9.90	1
12	14	1	basic monthly	2020-09-29	9.90	1
13	16	1	basic monthly	2020-06-07	9.90	1
14	17	1	basic monthly	2020-08-03	9.90	1
15	18	2	pro monthly	2020-07-13	19.90	1
16	19	2	pro monthly	2020-06-29	19.90	1
17	20	1	basic monthly	2020-04-15	9.90	1
18	22	2	pro monthly	2020-01-17	19.90	1
19	24	2	pro monthly	2020-11-17	19.90	1
20	25	1	basic monthly	2020-05-17	9.90	1
21	25	2	pro monthly	2020-06-16	19.90	1
22	26	2	pro monthly	2020-12-15	19.90	1
23	27	2	pro monthly	2020-08-31	19.90	1
24	29	2	pro monthly	2020-01-30	19.90	1
25	30	1	basic monthly	2020-05-06	9.90	1
26	31	2	pro monthly	2020-06-29	19.90	1
27	32	1	basic monthly	2020-06-19	9.90	1
28	32	2	pro monthly	2020-07-18	19.90	1
29	33	2	pro monthly	2020-09-10	19.90	1

Key Findings and Observations

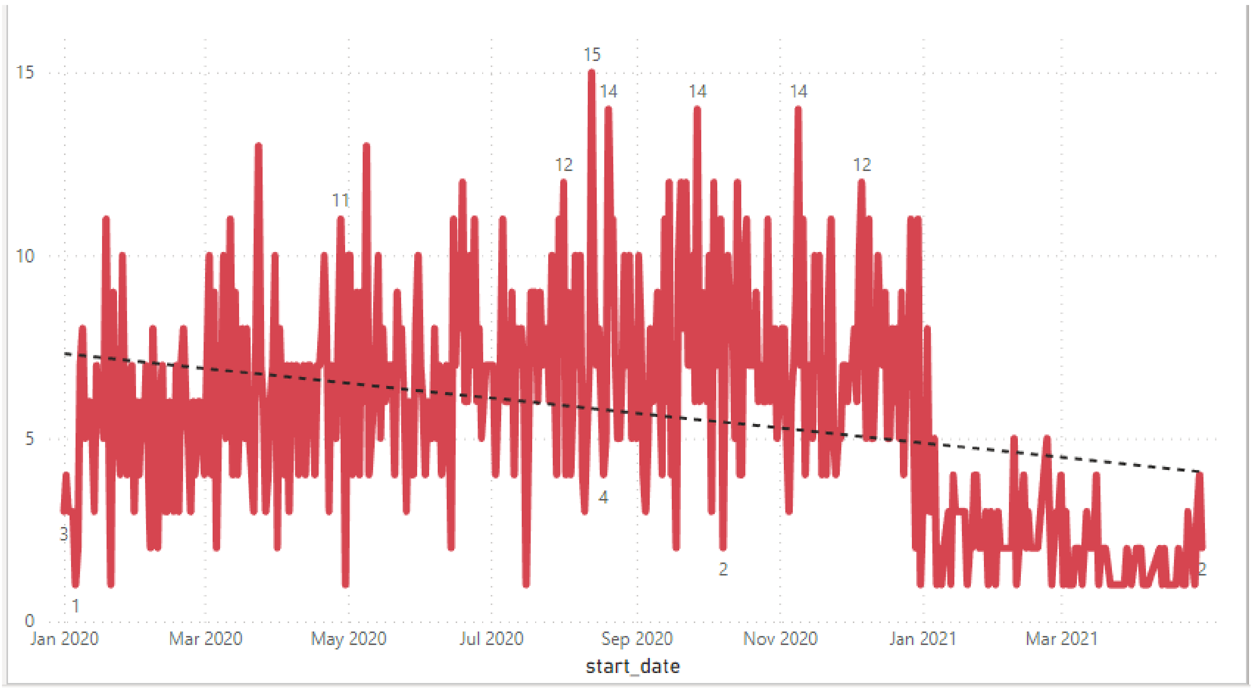
1.How many customers have we acquired over time?



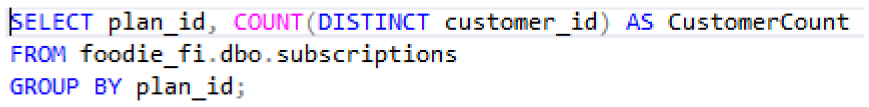
```
SELECT start_date, COUNT(DISTINCT customer_id) AS CustomerCount
FROM foodie_fi.dbo.subscriptions
GROUP BY start_date
ORDER BY start_date;
```


Results
Messages

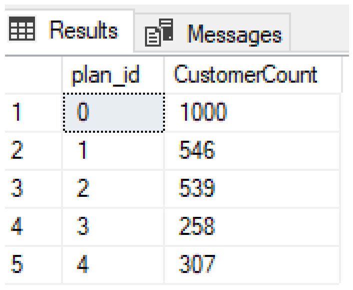
	start_date	CustomerCount
1	2020-01-01	3
2	2020-01-02	4
3	2020-01-03	3
4	2020-01-04	3
5	2020-01-05	3
6	2020-01-06	1
7	2020-01-07	2
8	2020-01-08	7
9	2020-01-09	8
10	2020-01-10	5



2.What's the distribution of our subscription plans?

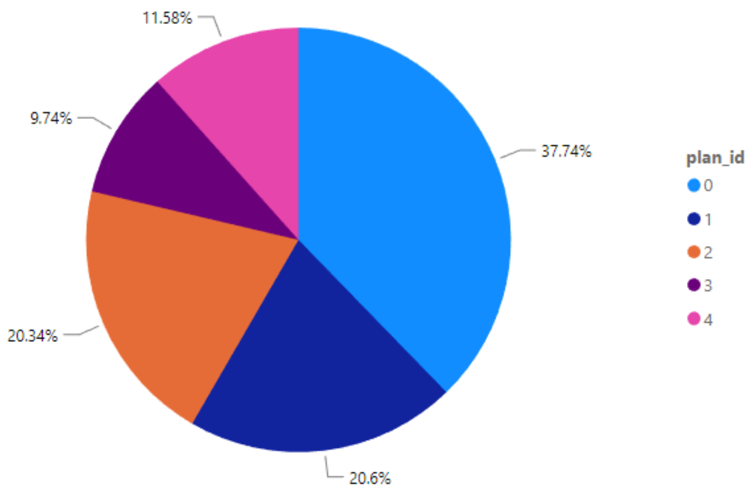


```sql
SELECT plan_id, COUNT(DISTINCT customer_id) AS CustomerCount
FROM foodie_fi.dbo.subscriptions
GROUP BY plan_id;
```

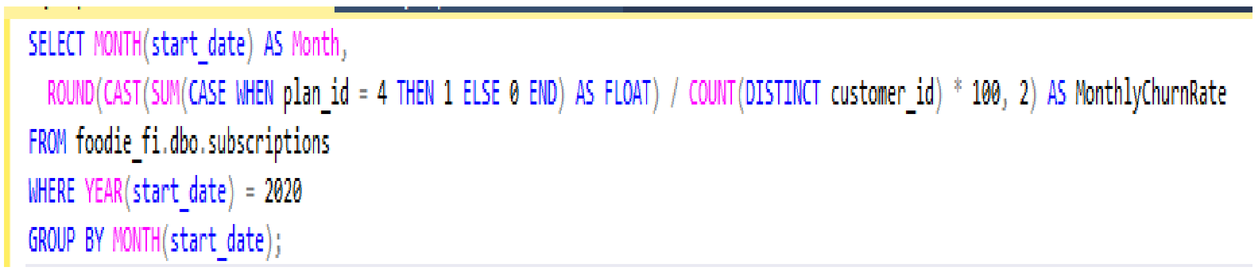


Results
Messages

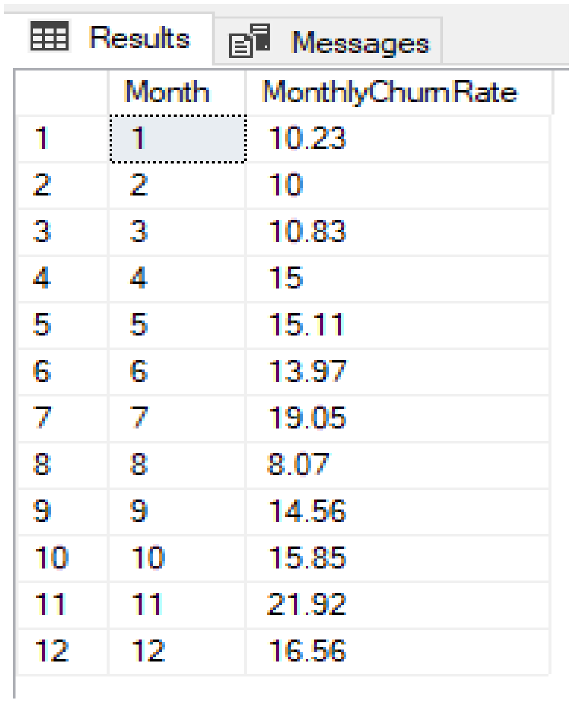
	plan_id	CustomerCount
1	0	1000
2	1	546
3	2	539
4	3	258
5	4	307



3.What's our monthly churn rate?

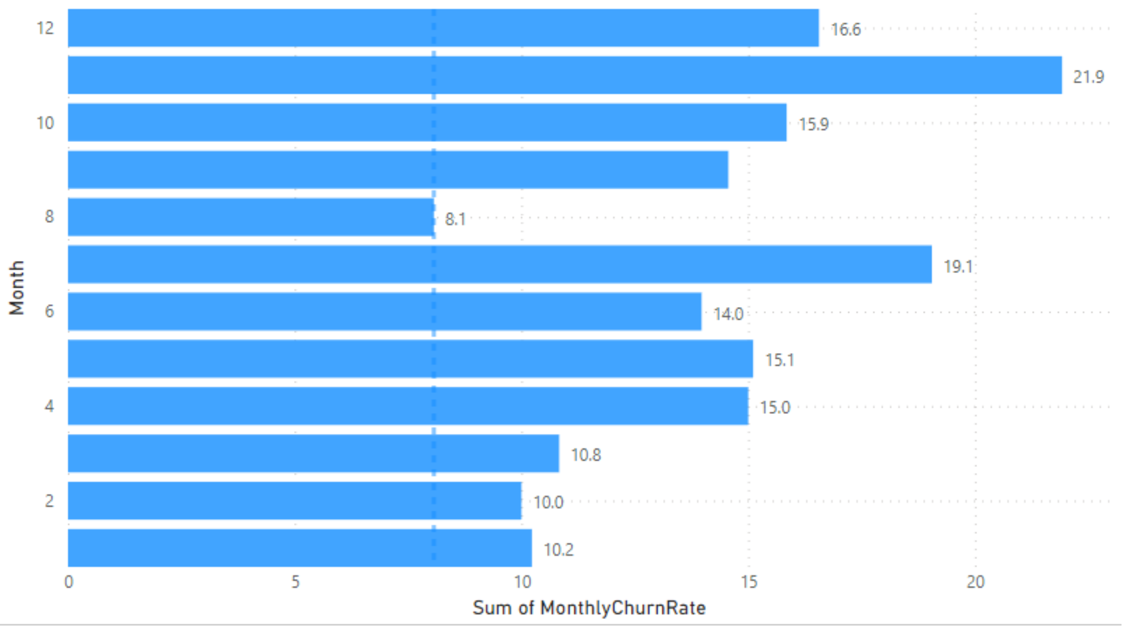


```sql
SELECT MONTH(start_date) AS Month,
  ROUND(CAST(SUM(CASE WHEN plan_id = 4 THEN 1 ELSE 0 END) AS FLOAT) / COUNT(DISTINCT customer_id) * 100, 2) AS MonthlyChurnRate
FROM foodie_fi.dbo.subscriptions
WHERE YEAR(start_date) = 2020
GROUP BY MONTH(start_date);
```

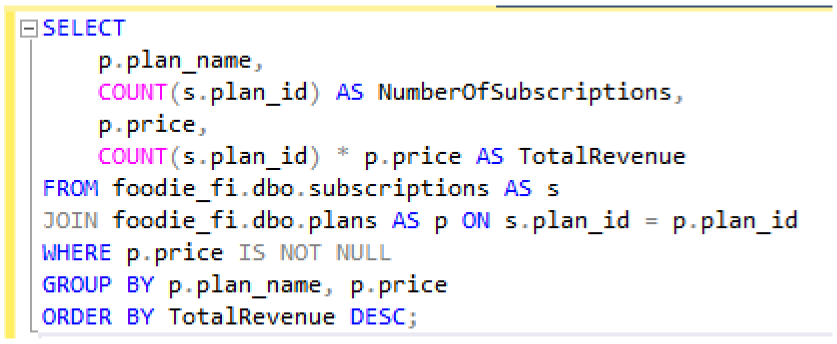


Results | Messages

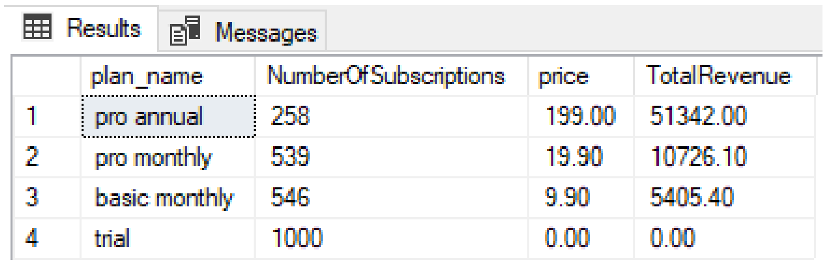
	Month	MonthlyChurnRate
1	1	10.23
2	2	10
3	3	10.83
4	4	15
5	5	15.11
6	6	13.97
7	7	19.05
8	8	8.07
9	9	14.56
10	10	15.85
11	11	21.92
12	12	16.56



4.How much revenue is generated from each plan?

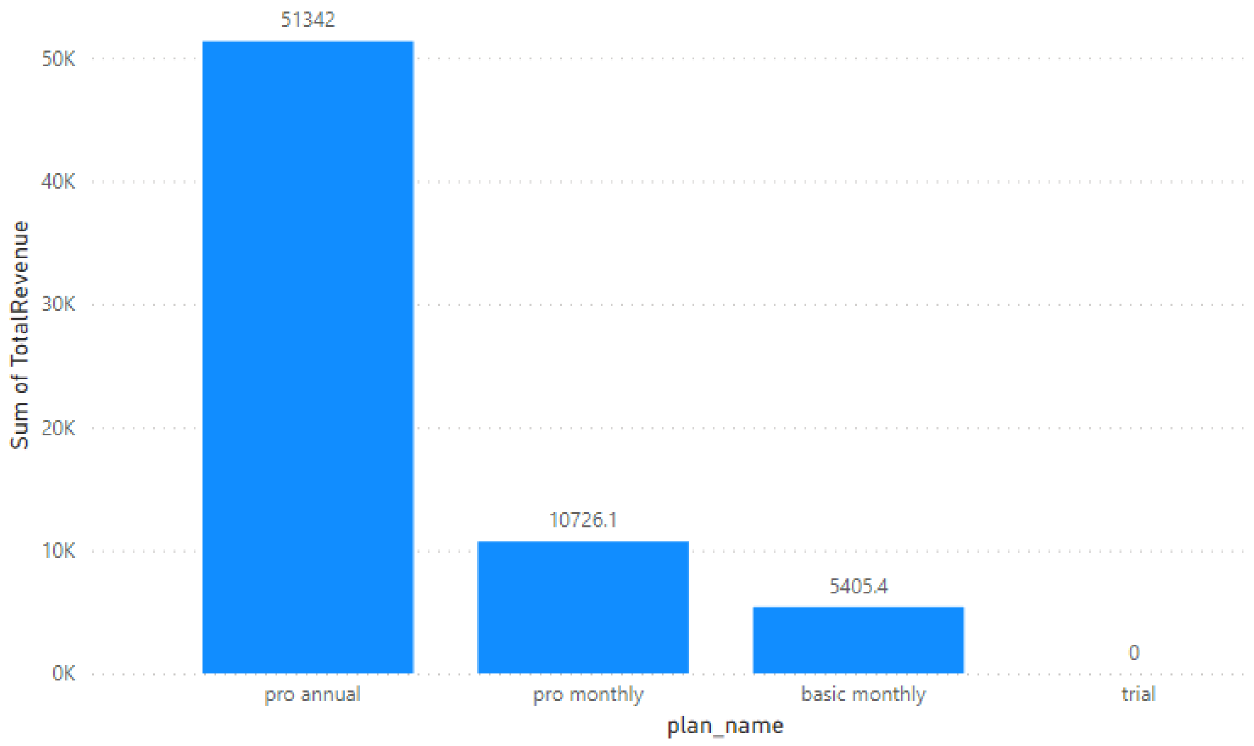


```
SELECT
    p.plan_name,
    COUNT(s.plan_id) AS NumberOfSubscriptions,
    p.price,
    COUNT(s.plan_id) * p.price AS TotalRevenue
FROM foodie_fi.dbo.subscriptions AS s
JOIN foodie_fi.dbo.plans AS p ON s.plan_id = p.plan_id
WHERE p.price IS NOT NULL
GROUP BY p.plan_name, p.price
ORDER BY TotalRevenue DESC;
```

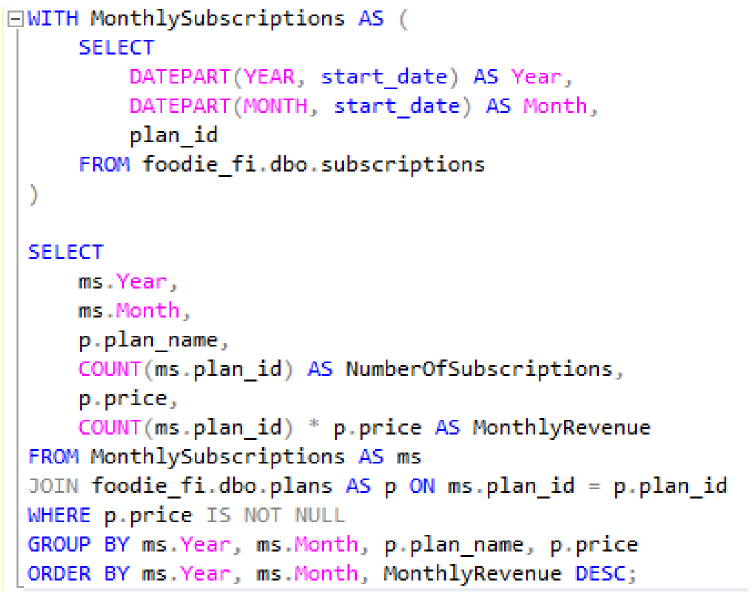


Results Messages

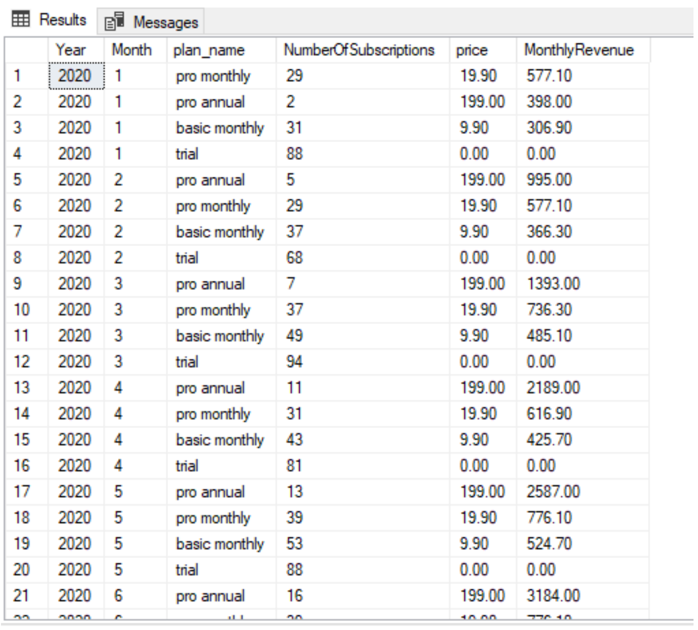
	plan_name	NumberOfSubscriptions	price	TotalRevenue
1	pro annual	258	199.00	51342.00
2	pro monthly	539	19.90	10726.10
3	basic monthly	546	9.90	5405.40
4	trial	1000	0.00	0.00



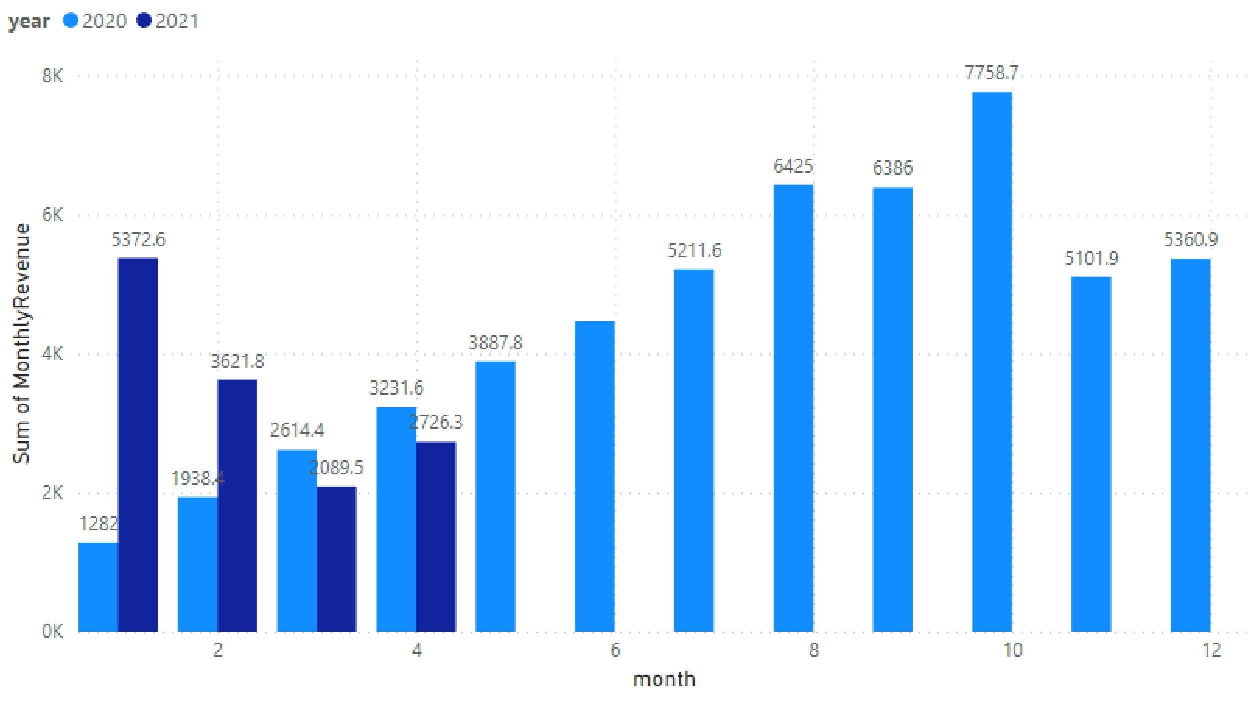
5.Monthly Revenue Trend Over Time



```
WITH MonthlySubscriptions AS (
    SELECT
        DATEPART(YEAR, start_date) AS Year,
        DATEPART(MONTH, start_date) AS Month,
        plan_id
    FROM foodie_fi.dbo.subscriptions
)

SELECT
    ms.Year,
    ms.Month,
    p.plan_name,
    COUNT(ms.plan_id) AS NumberOfSubscriptions,
    p.price,
    COUNT(ms.plan_id) * p.price AS MonthlyRevenue
FROM MonthlySubscriptions AS ms
JOIN foodie_fi.dbo.plans AS p ON ms.plan_id = p.plan_id
WHERE p.price IS NOT NULL
GROUP BY ms.Year, ms.Month, p.plan_name, p.price
ORDER BY ms.Year, ms.Month, MonthlyRevenue DESC;
```



	Year	Month	plan_name	NumberOfSubscriptions	price	MonthlyRevenue
1	2020	1	pro monthly	29	19.90	577.10
2	2020	1	pro annual	2	199.00	398.00
3	2020	1	basic monthly	31	9.90	306.90
4	2020	1	trial	88	0.00	0.00
5	2020	2	pro annual	5	199.00	995.00
6	2020	2	pro monthly	29	19.90	577.10
7	2020	2	basic monthly	37	9.90	366.30
8	2020	2	trial	68	0.00	0.00
9	2020	3	pro annual	7	199.00	1393.00
10	2020	3	pro monthly	37	19.90	736.30
11	2020	3	basic monthly	49	9.90	485.10
12	2020	3	trial	94	0.00	0.00
13	2020	4	pro annual	11	199.00	2189.00
14	2020	4	pro monthly	31	19.90	616.90
15	2020	4	basic monthly	43	9.90	425.70
16	2020	4	trial	81	0.00	0.00
17	2020	5	pro annual	13	199.00	2587.00
18	2020	5	pro monthly	39	19.90	776.10
19	2020	5	basic monthly	53	9.90	524.70
20	2020	5	trial	88	0.00	0.00
21	2020	6	pro annual	16	199.00	3184.00



CONCLUSION

The Foodie-Fi case study was a deep dive into the world of SQL, showing its ability to pull out valuable insights for a business. By tackling this study, I used a mix of SQL tools and techniques. This included not only the basics like JOINs and WHERE clauses but also more advanced features like **WINDOW functions**, **CTEs** (Common Table Expressions), and **aggregate functions** such as **COUNT**, **SUM**, and **AVG**.

All the data manipulation and extraction were done using the **Microsoft SQL Server Management Studio**. It's a great tool for handling complex data tasks and made the work much smoother. After gathering all the necessary data, I used **Excel** to make a clear and **informative dashboard**. This dashboard turned raw numbers into visual stories, making it easier to understand the business's performance.

I also had some great help along the way from **ChatGPT**. Whenever I got stuck or needed a **fresh perspective**, ChatGPT was there to guide and suggest better approaches.

In the end, working on this case study reinforced how powerful SQL can be for making sense of data. I learned a lot about various SQL functions and how they can be used to answer real-world business questions.