```sql
-- Querying plans table
SELECT *
FROM plans;
```

plan_id	plan_name	price
0	trial	0.00
1	basic monthly	9.90
2	pro monthly	19.90
3	pro annual	199.00
4	churn	*NULL*

```sql
-- Querying subscriptions table
SELECT *
FROM subscriptions
LIMIT 5;
```

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
3	0	2020-01-13

```sql
-- A. CUSTOMER JOURNEY

-- I'm providing the brief of a sample of subscriptions customers journey on the app.

SELECT S.*,
                P.plan_name
FROM subscriptions AS S
INNER JOIN plans AS P
ON S.plan_id = P.plan_id
WHERE S.customer_id IN (1, 2, 11, 13, 15, 16, 18, 19)
ORDER BY S.customer_id, S.start_date ASC;
```

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

customer_id 1: customer subscribed to the trial plan and after 7 days subscribed to the basic_monthly plan.

customer_id 2: customer subscribed to the trial plan and after 7 days subscribed to the pro_annual plan.

customer_id 11: customer subscribed to the trial plan and after 7 days churned.

customer_id 13: customer subscribed to the trial plan and after 7 days subscribed to the basic_monthly plan. After 3 months, subscribed to the pro_montly plan.

customer_id 15: customer subscribed to the trial plan and after 7 days subscribed to the pro_montly plan. After using more than 30 days, customer churned.

customer_id 16: customer subscribed to the trial plan and after 7 days subscribed to the basic_monthly plan. After using more than 4 months, customer subscribed to the pro_annual plan.

customer_id 18: customer subscribed to the trial plan and after 7 days subscribed to the pro_monthly plan.

customer_id 19: customer subscribed to the trial plan and after 7 days subscribed to the pro_monthly plan. After using it for more than 2 months, customer subscribed to the pro_annual plan.

```sql
-- B. DATA ANALYSIS QUESTIONS

-- 1. How many customers has Foodie-Fi ever had?
SELECT COUNT(DISTINCT(customer_id)) AS total_customers
FROM subscriptions;
```

total_customers
1000

```sql
-- 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
WITH trial_months AS (
    SELECT EXTRACT(month FROM start_date) AS month,
                        TO_CHAR(start_date, 'Month') AS month_name
    FROM subscriptions
    WHERE plan_id = 0
            )

SELECT month,
       month_name,
        COUNT(month) AS num_of_trials
FROM trial_months
GROUP BY month, month_name
ORDER BY month ASC;
```

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

```sql
-- 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(S.plan_id) AS count_of_events
FROM subscriptions AS S
INNER JOIN plans AS P
ON S.plan_id = P.plan_id
WHERE EXTRACT(YEAR FROM start_date) > 2020
GROUP BY P.plan_name;
```

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

```sql
-- 4. What is the customer count and percentage of customers who have churned rounded to 1
decimal place?
SELECT COUNT(DISTINCT(customer_id)) AS customer_count,
        ROUND((SELECT CAST(COUNT(DISTINCT(customer_id)) AS NUMERIC)
                FROM subscriptions
                WHERE plan_id = (SELECT plan_id
                        FROM plans
                        WHERE plan_name = 'churn')) / COUNT(DISTINCT(customer_id)) * 100,
1) AS percentage_of_churned_customer
FROM subscriptions;
```

customer_count	percentage_of_churned_customer
1000	30.7

```sql
-- 5. How many customers have churned straight after their initial free trial - what
percentage is this rounded to the nearest whole number?
SELECT COUNT(DISTINCT(customer_id)) AS customers_churned,
        CEILING(CAST(COUNT(DISTINCT(customer_id)) AS NUMERIC) /
        (SELECT COUNT(DISTINCT(customer_id)) AS total_customers FROM subscriptions)) AS
percent_of_customer_straight_after_trial
FROM (SELECT customer_id,
        plan_id,
        LEAD(plan_id) OVER(ORDER BY customer_id ASC, start_date ASC) AS next_plan
FROM subscriptions) AS churned_after_trial
WHERE plan_id = 0 AND next_plan = 4;
```

customers_churned	percent_of_customer_straight_after_trial
92	1

```sql
-- 6. What is the number and percentage of customer plans after their initial free trial?
WITH plan_after_trial AS (
        SELECT customer_id,
                plan_id,
                LEAD(plan_id) OVER(ORDER BY customer_id ASC, start_date ASC) AS next_plan
        FROM subscriptions
        ),

        next_plan_count AS (
        SELECT next_plan, COUNT(customer_id) AS plan_count
        FROM plan_after_trial
        WHERE plan_id = 0 AND next_plan != 4
        GROUP BY next_plan
        )

SELECT next_plan,
        P.plan_name,
        plan_count,
        ROUND(CAST(plan_count AS NUMERIC) / (SELECT SUM(plan_count) FROM next_plan_count) *
100, 2) AS percent_of_plans
FROM next_plan_count AS NPC
INNER JOIN plans AS P
ON NPC.next_plan = P.plan_id
ORDER BY next_plan;
```

next_plan	plan_name	plan_count	percent_of_plans
1	basic monthly	546	60.13
2	pro monthly	325	35.79
3	pro annual	37	4.07

```sql
-- 7. What is the customer count and percentage breakdown of all 5 plan_name values at
2020-12-31?
WITH next_plan_count AS (
        SELECT customer_id,
                plan_id,
                LEAD(plan_id) OVER(ORDER BY customer_id ASC, start_date ASC) AS next_plan
        FROM subscriptions
        WHERE start_date <= '2020-12-31'
        ),

        plans_2020 AS (
        SELECT customer_id,
                MAX(next_plan) AS customer_current_plan
        FROM next_plan_count
        GROUP BY customer_id
        ),

        plan_breakdown AS (
        SELECT customer_current_plan,
                P.plan_name,
                COUNT(customer_current_plan) AS customer_count
        FROM plans_2020 AS p_2020
        INNER JOIN plans AS P
        ON p_2020.customer_current_plan = P.plan_id
        GROUP BY customer_current_plan, plan_name
        )

SELECT *,
        ROUND(CAST(customer_count AS NUMERIC) / (SELECT SUM(customer_count) FROM
plan_breakdown) * 100, 2) AS percentage_of_plans
FROM plan_breakdown
ORDER BY customer_current_plan;
```

customer_current_plan	plan_name	customer_count	percentage_of_plans
0	trial	19	1.90
1	basic monthly	224	22.40
2	pro monthly	326	32.60
3	pro annual	195	19.50
4	churn	236	23.60

```sql
-- 8. How many customers have upgraded to an annual plan in 2020?
SELECT COUNT(DISTINCT(customer_id)) AS customers_upgraded_to_annual_plan
FROM subscriptions
WHERE plan_id IN (SELECT plan_id FROM plans WHERE plan_name = 'pro annual')
AND start_date < '2020-12-31';
```

customers_upgraded_to_annual_plan
195

```sql
-- 9. How many days on average does it take for a customer to an annual plan FROM the day
they join Foodie-Fi?
DROP VIEW IF EXISTS days_to_upgrade_to_annual_plan;
```

```sql
CREATE VIEW days_to_upgrade_to_annual_plan AS
WITH annual_plans AS (
        SELECT *
        FROM subscriptions
        WHERE plan_id != 4
        ),

        initial_plan_date AS (
        SELECT customer_id,
                min(start_date) AS initial_date
        FROM annual_plans
        GROUP BY customer_id
        ),

        annual_plan_date AS (
        SELECT customer_id,
                start_date AS upgrade_date
        FROM annual_plans
        WHERE plan_id = 3
        )

SELECT upgrade_date - initial_date AS days_to_upgrade
FROM initial_plan_date AS I
INNER JOIN annual_plan_date AS A ON I.customer_id = A.customer_id;

-- Querying the view to get the answer
SELECT CEILING(AVG(days_to_upgrade)) AS
average_days_customer_takes_to_upgrade_to_annual_plan
FROM days_to_upgrade_to_annual_plan;
```

average_days_customer_takes_to_upgrade_to_annual_plan
105

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

-- Querying the min and max values of days_to_upgrade
SELECT MIN(days_to_upgrade),
        MAX(days_to_upgrade)
FROM days_to_upgrade_to_annual_plan;
```

min	max
7	346

```sql
-- Querying for the final answer
WITH periods_of_upgrade AS (
        SELECT *,
                CASE
                        WHEN days_to_upgrade BETWEEN 0 AND 30 THEN '0-30 days'
                        WHEN days_to_upgrade BETWEEN 30 AND 60 THEN '30-60 days'
```

```
                        WHEN days_to_upgrade BETWEEN 60 AND 90 THEN '60-90 days'
                        WHEN days_to_upgrade BETWEEN 90 AND 120 THEN '90-120 days'
                        WHEN days_to_upgrade BETWEEN 120 AND 150 THEN '120-150 days'
                        WHEN days_to_upgrade BETWEEN 150 AND 180 THEN '150-180 days'
                        WHEN days_to_upgrade BETWEEN 180 AND 210 THEN '180-210 days'
                        WHEN days_to_upgrade BETWEEN 210 AND 240 THEN '210-240 days'
                        WHEN days_to_upgrade BETWEEN 240 AND 270 THEN '240-270 days'
                        WHEN days_to_upgrade BETWEEN 270 AND 300 THEN '270-300 days'
                        WHEN days_to_upgrade BETWEEN 300 AND 330 THEN '300-330 days'
                        ELSE '330-360 days'
                END AS upgrade_periods,
                CASE
                        WHEN days_to_upgrade BETWEEN 0 AND 30 THEN 1
                        WHEN days_to_upgrade BETWEEN 30 AND 60 THEN 2
                        WHEN days_to_upgrade BETWEEN 60 AND 90 THEN 3
                        WHEN days_to_upgrade BETWEEN 90 AND 120 THEN 4
                        WHEN days_to_upgrade BETWEEN 120 AND 150 THEN 5
                        WHEN days_to_upgrade BETWEEN 150 AND 180 THEN 6
                        WHEN days_to_upgrade BETWEEN 180 AND 210 THEN 7
                        WHEN days_to_upgrade BETWEEN 210 AND 240 THEN 8
                        WHEN days_to_upgrade BETWEEN 240 AND 270 THEN 9
                        WHEN days_to_upgrade BETWEEN 270 AND 300 THEN 10
                        WHEN days_to_upgrade BETWEEN 300 AND 330 THEN 11
                        ELSE 12
                END AS rank
        FROM days_to_upgrade_to_annual_plan
        )

SELECT upgrade_periods,
        CEILING(AVG(days_to_upgrade)) AS average_days_to_upgrade
FROM periods_of_upgrade
GROUP BY upgrade_periods, rank
ORDER BY rank ASC;
```

upgrade_periods	average_days_to_upgrade
0-30 days	10
30-60 days	43
60-90 days	72
90-120 days	101
120-150 days	134
150-180 days	163
180-210 days	191
210-240 days	225
240-270 days	258
270-300 days	285
300-330 days	327
330-360 days	346

```sql
-- How many customers downgraded from a pro monthly to a basic monthly plan in 2020?
WITH customers_next_plan AS (
        SELECT customer_id,
                plan_id,
                LEAD(plan_id) OVER(ORDER BY customer_id ASC, start_date ASC) AS next_plan
        FROM subscriptions
        WHERE start_date < '2020-12-31'
        )

SELECT COUNT(customer_id) AS customers_downgraded_from_pro_monthly_to_basic
FROM customers_next_plan
WHERE next_plan = 1 AND plan_id = 2;
```

customers_downgraded_from_pro_monthly_to_basic
0