



**SQL**

**Portfolio Project**

**Course Instructor**

Miss Mahnoor Fatima

**Teaching Assistant**

Miss Nimra

**Batch-02 DS**

**Submitted By: Muhammad Hamza**

**Data Science & AI Boot Camp**

Table of Contents

[Introduction 1](#_Toc166361089)

[SQL Analysis 1](#_Toc166361090)

[1. Total Customers 1](#_Toc166361091)

[2. Monthly Distribution of Trial Plan Start Dates 1](#_Toc166361092)

[3. Plan Start Dates After 2020 2](#_Toc166361093)

[4. Customer Churn 2](#_Toc166361094)

[5. Churn After Free Trial 2](#_Toc166361095)

[6. Customer Plans After Free Trial 3](#_Toc166361096)

[7. Plan Breakdown at End of 2020 3](#_Toc166361097)

[8. Upgrades to Annual Plan in 2020 4](#_Toc166361098)

[9. Average Time to Upgrade to Annual Plan 4](#_Toc166361099)

[10. Breakdown of Upgrade Time into 30-Day Periods 5](#_Toc166361100)

[11. Downgrades from Pro Monthly to Basic Monthly Plan in 2020 6](#_Toc166361101)

[Conclusion 6](#_Toc166361102)

Project Report: Analysis of Foodie-Fi Subscription Data

## Introduction

This report presents an analysis of the subscription data for Foodie-Fi, a food delivery service. The analysis is based on SQL queries that extract insights from the data. The SQL code used for the analysis is included in the report.

## SQL Analysis

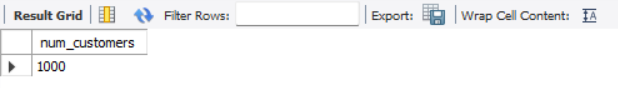
### 1. Total Customers

The total number of customers that Foodie-Fi has ever had is determined using the following SQL query:

***SELECT***

***COUNT(DISTINCT customer\_id) AS num\_customers***

***FROM subscriptions;***



The result is 1000 customers.

### 2. Monthly Distribution of Trial Plan Start Dates

The monthly distribution of trial plan start dates is determined using the following SQL query:

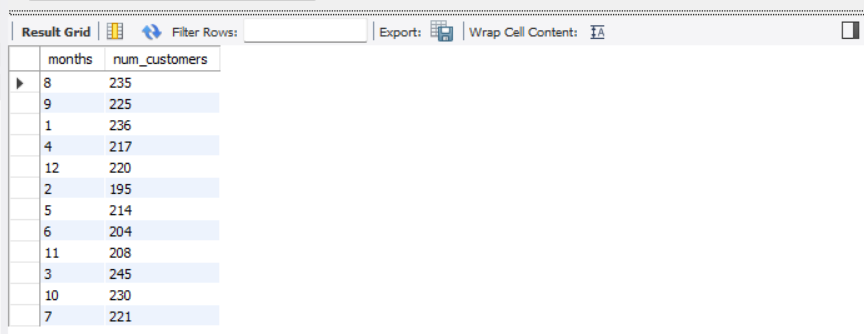
SELECT

MONTH(start\_date) AS months,

COUNT(customer\_id) AS num\_customers

FROM subscriptions

GROUP BY months;



### 3. Plan Start Dates After 2020

The plan start dates that occur after the year 2020 are determined using the following SQL query:

SELECT

 p.plan\_name,

 p.plan\_id,

 COUNT(\*) AS cnt\_event

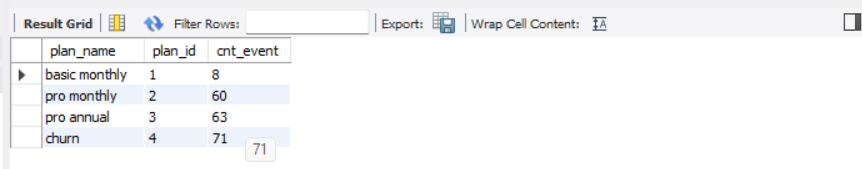
FROM subscriptions s

INNER JOIN plans p ON p.plan\_id = s.plan\_id

WHERE s.start\_date >= '2021-01-01'

GROUP BY p.plan\_id,p.plan\_name

ORDER BY p.plan\_id;



### 4. Customer Churn

The customer count and percentage of customers who have churned is determined using the following SQL query:

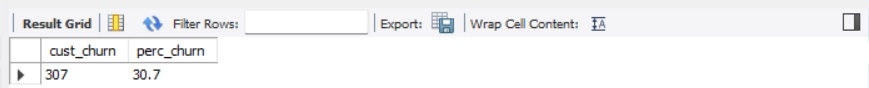
SELECT

  COUNT(\*) AS cust\_churn,

  ROUND(COUNT(\*) \* 100 / (SELECT COUNT(DISTINCT customer\_id) FROM subscriptions),1) AS perc\_churn

FROM subscriptions

WHERE plan\_id = 4;



### 5. Churn After Free Trial

The number of customers who churned straight after their initial free trial is determined using the following SQL query:

WITH cte\_churn AS (

SELECT \*, LAG(plan\_id, 1) OVER(PARTITION BY customer\_id ORDER BY plan\_id) AS prev\_plan

FROM subscriptions)

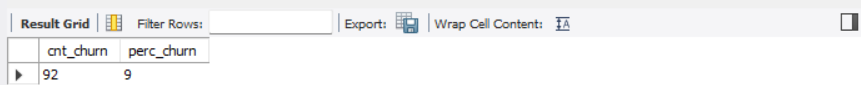
SELECT

COUNT(prev\_plan) AS cnt\_churn,

   ROUND(COUNT(\*) \* 100/(SELECT COUNT(DISTINCT customer\_id) FROM subscriptions),0) AS perc\_churn

FROM cte\_churn

WHERE plan\_id = 4 and prev\_plan = 0;



The result is a churn count of 92, which is 9% of the total customers.

### 6. Customer Plans After Free Trial

The number and percentage of customer plans after their initial free trial is determined using the following SQL query:

WITH cte\_next\_plan AS (

SELECT

\*,

LEAD(plan\_id, 1) OVER(PARTITION BY customer\_id ORDER BY plan\_id) AS next\_plan

FROM subscriptions)

SELECT

next\_plan,

COUNT(\*) AS num\_cust,

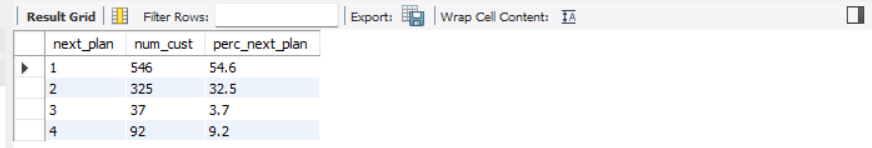
   ROUND(COUNT(\*) \* 100/(SELECT COUNT(DISTINCT customer\_id) FROM subscriptions),1) AS perc\_next\_plan

FROM cte\_next\_plan

WHERE next\_plan is not null and plan\_id = 0

GROUP BY next\_plan

ORDER BY next\_plan;



### 7. Plan Breakdown at End of 2020

The customer count and percentage breakdown of all plan names at the end of 2020 is determined using the following SQL query:

WITH cte\_next\_date AS (

SELECT

\*,

LEAD(start\_date, 1) OVER(PARTITION BY customer\_id ORDER BY start\_date) AS next\_date

FROM subscriptions

  WHERE start\_date <= '2020-12-31'),

plans\_breakdown AS(

SELECT

plan\_id,

  COUNT(DISTINCT customer\_id) AS num\_customer

FROM cte\_next\_date

WHERE (next\_date IS NOT NULL AND (start\_date < '2020-12-31' AND next\_date > '2020-12-31'))

   OR (next\_date IS NULL AND start\_date < '2020-12-31')

GROUP BY plan\_id)

SELECT

plan\_id,

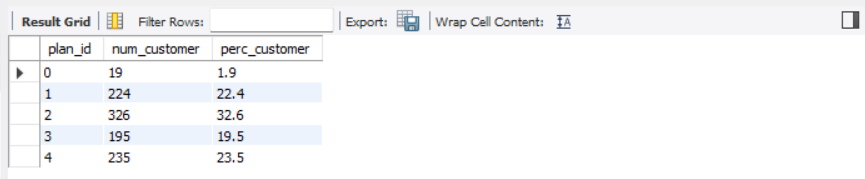
num\_customer,

  ROUND(num\_customer \* 100/(SELECT COUNT(DISTINCT customer\_id) FROM subscriptions),1) AS perc\_customer

FROM plans\_breakdown

GROUP BY plan\_id, num\_customer

ORDER BY plan\_id;



### 8. Upgrades to Annual Plan in 2020

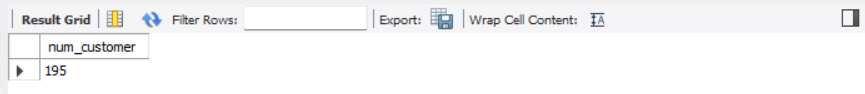
The number of customers who upgraded to an annual plan in 2020 is determined using the following SQL query:

SELECT

COUNT(customer\_id) AS num\_customer

FROM subscriptions

WHERE plan\_id = 3 AND start\_date <= '2020-12-31';



The result is 195 customers.

### 9. Average Time to Upgrade to Annual Plan

The average number of days it takes for a customer to upgrade to an annual plan from the day they join Foodie-Fi is determined using the following SQL query:

WITH annual\_plan AS (

SELECT

customer\_id,

    start\_date AS annual\_date

FROM subscriptions

   WHERE plan\_id = 3),

trial\_plan AS (

SELECT

customer\_id,

    start\_date AS trial\_date

FROM subscriptions

  WHERE plan\_id = 0

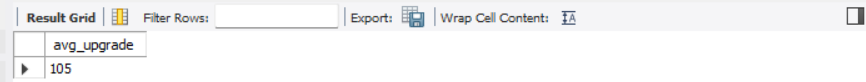
)

SELECT

ROUND(AVG(DATEDIFF(annual\_date, trial\_date)),0) AS avg\_upgrade

FROM annual\_plan ap

JOIN trial\_plan tp ON ap.customer\_id = tp.customer\_id;



The result is an average of 105 days.

### 10. Breakdown of Upgrade Time into 30-Day Periods

The breakdown of the average upgrade time into 30-day periods is determined using the following SQL query:

WITH annual\_plan AS (

SELECT

customer\_id,

    start\_date AS annual\_date

FROM subscriptions

  WHERE plan\_id = 3),

trial\_plan AS (

SELECT

customer\_id,

    start\_date AS trial\_date

FROM subscriptions

  WHERE plan\_id = 0

),

day\_period AS (

SELECT

DATEDIFF(annual\_date, trial\_date) AS diff

FROM trial\_plan tp

LEFT JOIN annual\_plan ap ON tp.customer\_id = ap.customer\_id

WHERE annual\_date is not null

),

bins AS (

SELECT

\*, FLOOR(diff/30) AS bins

FROM day\_period)

SELECT

CONCAT((bins \* 30) + 1, ' - ', (bins + 1) \* 30, ' days ') AS days,

COUNT(diff) AS total

FROM bins

GROUP BY bins;

A screenshot of a computer

Description automatically generated

### 11. Downgrades from Pro Monthly to Basic Monthly Plan in 2020

The number of customers who downgraded from a pro monthly to a basic monthly plan in 2020 is determined using the following SQL query:

WITH next\_plan AS (

SELECT

\*,

LEAD(plan\_id, 1) OVER(PARTITION BY customer\_id ORDER BY start\_date, plan\_id) AS plan

FROM subscriptions)

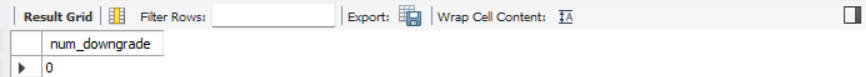
SELECT

COUNT(DISTINCT customer\_id) AS num\_downgrade

FROM next\_plan np

LEFT JOIN plans p ON p.plan\_id = np.plan\_id

WHERE p.plan\_name = 'pro monthly' AND np.plan = 1 AND start\_date <= '2020-12-31';



The result is 0 customers.

## Conclusion

The SQL analysis provides valuable insights into the customer behavior and subscription trends of Foodie-Fi. These insights can be used to inform business decisions and improve customer retention strategies. The SQL code used for the analysis is robust and can be adapted for future analyses as the subscription data grows and evolves.