SQL CASE STUDY - FOODIE-FI

ANALYZING DATA FOR FOOD FI COMPANY USING SQL

BY ABDULLAH ZUNORAIN

5 April 2024

INTRODUCTION TO FOOD-FI COMPANY

Brief Overview of Foodie-Fi:

- Foodie-Fi: A subscription-based streaming service launched in 2020.
- Industry: Positioned within the streaming media sector, specializing in culinary content.
- Unique Selling Point: Offers exclusive access to a diverse range of cooking shows and food-related content.
- Target Audience: Caters to people who love food and cooking and want specialized shows.
- Competition: Operates in a market dominated by general entertainment streaming platforms like Netflix, Hulu, and Amazon Prime.
- Market Differentiation: Stands out by focusing solely on culinary content, providing subscribers with unlimited on-demand access to food-related videos from around the world.
- Business Model: Generates revenue through monthly and annual subscription plans, offering subscribers unrestricted access to premium culinary content.

INTRODUCTION TO THE CASE STUDY AND ITS OBJECTIVES

Background: Food Fi is a subscription-based streaming service launched in 2020, specializing in food-related content.

Objective: The case study aims to showcase how Food Fi utilizes data analysis using SQL to make informed business decisions.

Data-Driven Approach: Food Fi operates with a data-driven mindset, ensuring that all decisions, including investment choices and feature development, are based on data insights.

Purpose: Through this case study, we aim to demonstrate how SQL queries are used to analyze Food Fi's subscription data and derive valuable insights for business growth and decision-making.

Key Goals: The primary objectives include understanding customer preferences, optimizing pricing strategies, analyzing cohort behavior, optimizing customer acquisition, and reducing churn through data-driven strategies.

Value Proposition: By harnessing SQL for data analysis, Food Fi can gain actionable insights to enhance customer satisfaction, improve retention, and drive business success in the competitive streaming media industry.

DATA DESCRIPTION

The dataset used in the case study consists of two main tables within the Food Fi database schema. Below are the details of each table along with their columns and meanings:

Table 1: plans

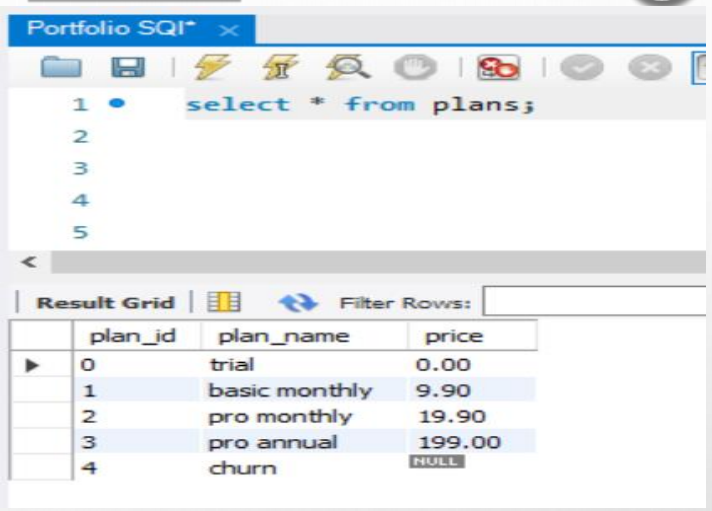
- **plan_id**: Unique identifier for each subscription plan.
- **plan_name**: Name of the subscription plan (e.g., trial, basic monthly, pro monthly, pro annual, churn).
- **price**: Price associated with each subscription plan. For some plans like the trial and churn, the price may be null.

Table 2: subscriptions

- **customer_id**: Unique identifier for each customer.
- **plan_id**: Identifier referencing the subscription plan chosen by the customer.
- **start_date**: Date when the subscription plan started for the customer.

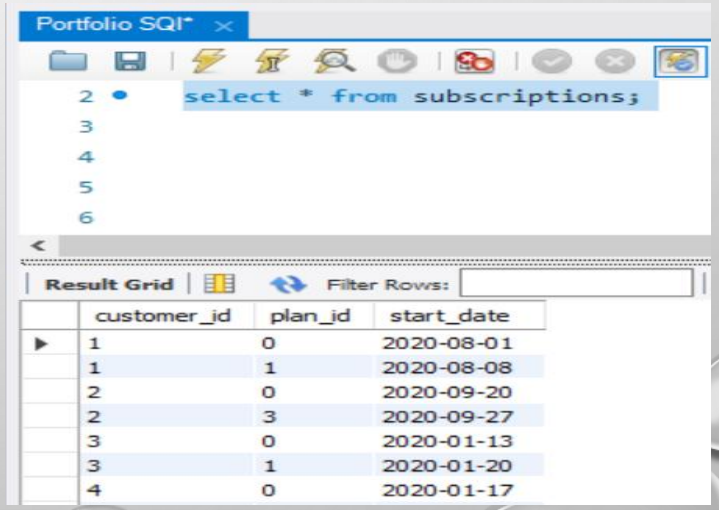
These tables provide essential data for analyzing customer subscription patterns, understanding plan preferences, and tracking customer engagement over time. The plan_id column serves as a foreign key linking the subscriptions table with the plans table, allowing for comprehensive analysis of subscription data.

Plans table-1



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

Subscriptions table-2



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
3	1	2020-01-20
4	0	2020-01-17

- *The dataset is available at the following link:* https://tinyurl.com/food-fi

BUSINESS PROBLEMS AND OBJECTIVES

Business Problems:

- **Subscription Retention:** Food Fi faces challenges in retaining subscribers and reducing churn rates.
- **Understanding Customer Preferences:** Lack of insights into customer preferences regarding subscription plans and content preferences.
- **Optimizing Pricing Strategy:** Need to determine the most effective pricing strategy to attract and retain customers while maximizing revenue.

Objectives of Using SQL for Data Analysis:

- **Identifying Patterns:** Utilize SQL to analyze subscription data and identify patterns in customer behavior, such as subscription trends and viewing habits.
- **Optimizing Plans:** Use SQL to analyze subscription data and tailor subscription plans to match customer preferences, thereby improving retention rates.
- **Price Optimization:** Employ SQL to analyze pricing data and determine optimal price points for subscription plans, balancing revenue generation with customer satisfaction and retention.

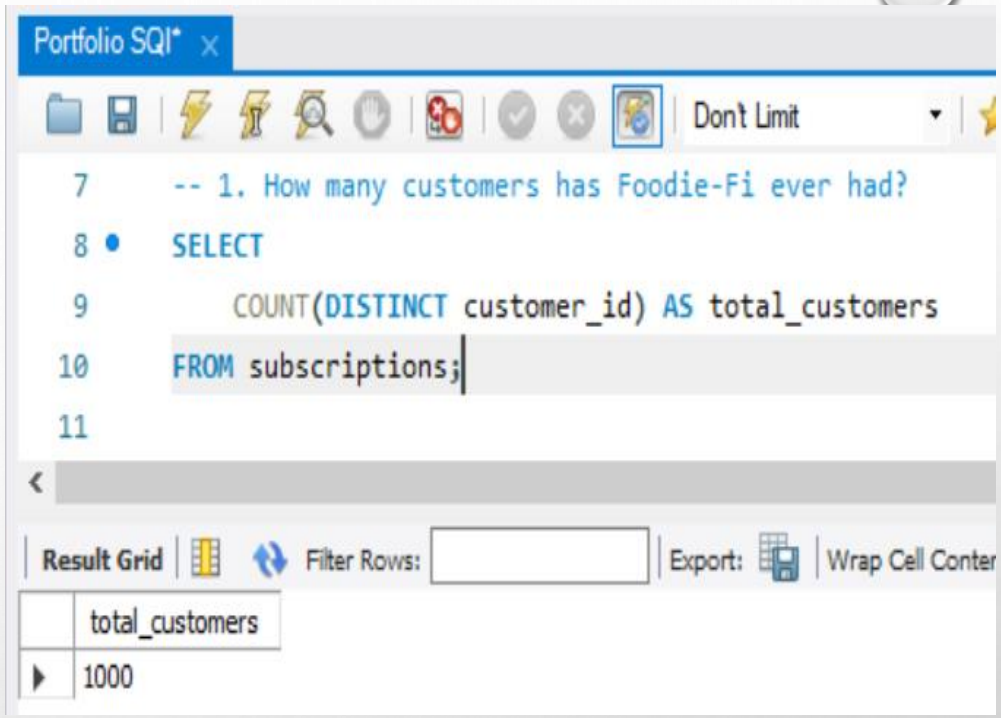
SQL QUERIES AND ANALYSIS

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

- *SQL CODE:*

```
SELECT    COUNT(DISTINCT CUSTOMER_ID) AS
TOTAL_CUSTOMERS

FROM SUBSCRIPTIONS;
```



Explanation:

The first query calculates the total number of unique customers that Foodie-Fi has ever had by counting the distinct customer IDs from the subscriptions table. In this case, the result is 1000, indicating that Foodie-Fi has had 1000 unique customers over the specified time period.

CONTINUE...

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?

- *SQL CODE:*

```
SELECT
    COUNT(PLAN_ID) AS COUNTS_OF_PLANS,
    MONTH(START_DATE) AS MONTHS
FROM SUBSCRIPTIONS
GROUP BY MONTHS, PLAN_ID
HAVING PLAN_ID=0;
```

Result Grid | Filter Rows:

counts_of_plans	months
88	8
87	9
88	1
84	12
68	2
79	6
75	11
94	3
88	5
89	7
81	4
79	10

Explanation:

The above query calculates the monthly distribution of trial plan start_date values for the dataset. It groups the data by the start month of each trial plan and counts the occurrences of trial plan start dates within each month. The `plan_id` column is filtered to only include trial plans (where `plan_id` equals 0). Finally, the results are displayed with the count of trial plans and the corresponding month of their start_date values.

CONTINUE...

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?

- *SQL CODE:*

```
SELECT
    S.PLAN_ID,
    P.PLAN_NAME,
    COUNT(P.PLAN_NAME) AS EVENT_COUNT
FROM SUBSCRIPTIONS S INNER JOIN PLANS P
ON S.PLAN_ID = P.PLAN_ID
WHERE YEAR(S.START_DATE) > 2020
GROUP BY S.PLAN_ID, P.PLAN_NAME;
```

Result Grid | Filter Rows:

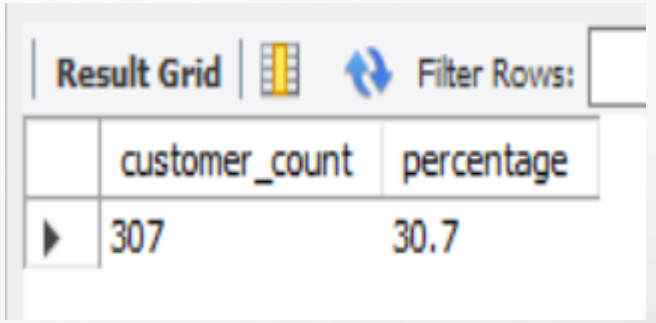
plan_id	plan_name	event_count
4	churn	71
2	pro monthly	60
3	pro annual	63
1	basic monthly	8

Explanation:

The above query groups the data by plan_id and plan_name, counting the occurrences of each plan_name. Finally, it presents the breakdown of events by plan_name along with the corresponding event_count.

CONTINUE...

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



- *SQL CODE:*

```
SELECT
    COUNT(DISTINCT CUSTOMER_ID) AS CUSTOMER_COUNT,
    ROUND((COUNT(DISTINCT CUSTOMER_ID)/(SELECT COUNT(DISTINCT CUSTOMER_ID) FROM SUBSCRIPTIONS)) * 100, 1) AS PERCENTAGE
FROM SUBSCRIPTIONS
WHERE PLAN_ID=4;
```

Explanation:

This query calculates the customer count and percentage of customers who have churned, rounded to 1 decimal place. It first counts the number of distinct customer IDs where the plan_id is equal to 4 (indicating churn). Then, it divides this count by the total count of distinct customer IDs in the subscriptions table, multiplying by 100 to get the percentage. The result shows that 307 customers have churned, which is approximately 30.7% of the total customer base.

CONTINUE...

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

- *SQL CODE:*

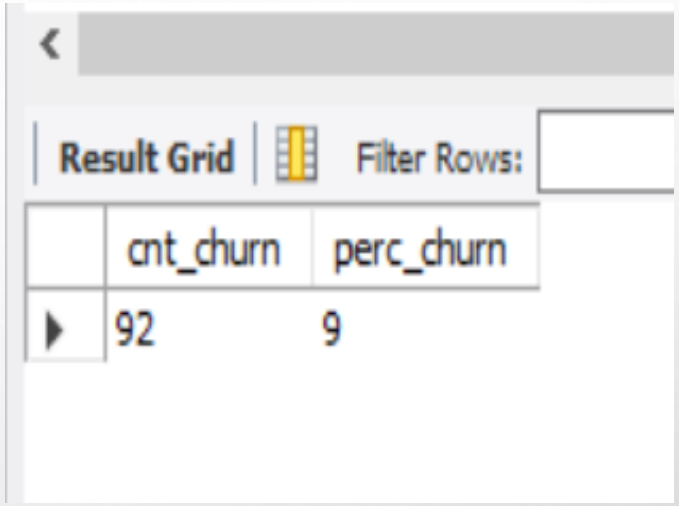
```
WITH CTE_CHURN AS (
        SELECT *, LAG(PLAN_ID, 1)  OVER(PARTITION BY CUSTOMER_ID) AS PREV_PLAN
        FROM SUBSCRIPTIONS )

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;
```



Explanation:

The SQL code identifies customers who churned immediately after their initial free trial. It calculates the count and percentage of such customers, which are 92 and approximately 9%, respectively.

CONTINUE...

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

- *SQL CODE:*

```
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;
```

Result Grid | Filter Rows:

next_plan	num_cust	perc_next_plan
1	546	54.6
2	325	32.5
3	37	3.7
4	92	9.2

CONTINUE...

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

- *SQL CODE:*

```
WITH MY_CTE AS (
        SELECT  *, ROW_NUMBER() OVER(PARTITION BY CUSTOMER_ID ORDER BY START_DATE DESC) AS RWNMBR
        FROM SUBSCRIPTIONS
        WHERE START_DATE <= '2020-12-31')

SELECT
        PLAN_NAME,
        COUNT(CUSTOMER_ID) AS CUSTOMER_COUNT,
        ROUND((COUNT(CUSTOMER_ID)/(SELECT COUNT(DISTINCT CUSTOMER_ID) FROM MY_CTE))*100,1) AS PERCENT_OF_CUSTOMERS
FROM MY_CTE MC INNER JOIN PLANS AS P
ON MC.PLAN_ID = P.PLAN_ID
WHERE RWNMBR = 1
GROUP BY PLAN_NAME;
```

Result Grid | Filter Rows: | Export:

plan_name	customer_count	percent_of_customers
trial	19	1.9
basic monthly	224	22.4
pro monthly	326	32.6
pro annual	195	19.5
churn	236	23.6

CONTINUE...

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

- *SQL CODE:*

```
SELECT COUNT(*)

FROM SUBSCRIPTIONS

WHERE YEAR(START_DATE) = 2020 AND PLAN_ID = 3;
```

```
247
248 •   select count(*) as Upgraded_cust_count
249     from subscriptions
250     where Year(start_date) = 2020 and plan_id = 3;
251
```

Result Grid | Filter Rows: | Export: | Wrap C

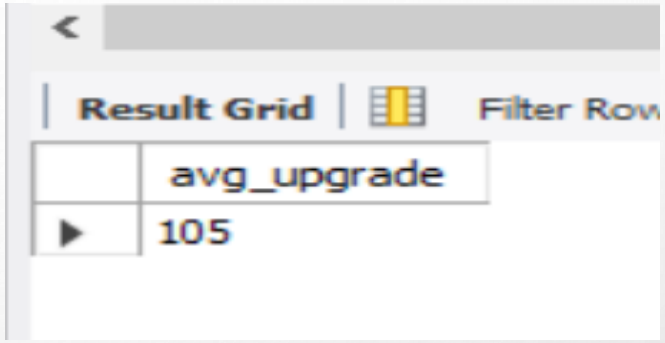
Upgraded_cust_count
195

Explanation:

The above SQL code counts the number of customers who upgraded to an annual plan in the year 2020. It filters the subscriptions table to include only records where the start_date occurred in 2020 (`Year(start_date) = 2020`) and the plan_id matches the identifier for the annual plan (`plan_id = 3`). The result indicates that 195 customers upgraded to an annual plan in 2020.

CONTINUE...

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



- *SQL CODE:*

```
WITH
        TRAIL_PLAN AS (
                SELECT  CUSTOMER_ID, START_DATE AS TRAIL_DATES    FROM SUBSCRIPTIONS    WHERE PLAN_ID=0 ),
       ANNUAL_PLAN AS (
                SELECT  CUSTOMER_ID, START_DATE AS ANNUAL_DATES    FROM SUBSCRIPTIONS    WHERE PLAN_ID=3 )
SELECT ROUND(AVG(DATEDIFF(ANNUAL_DATES, TRAIL_DATES)),0) AS AVG_UPGRADE
FROM ANNUAL_PLAN AP JOIN TRAIL_PLAN TP
ON AP.CUSTOMER_ID = TP.CUSTOMER_ID;
```

CONTINUE...

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

- *SQL CODE:*

```
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 BINSFROM DAY_PERIOD)

SELECT CONCAT((BINS * 30) + 1, ' - ', (BINS + 1) * 30, ' DAYS ') AS DAYS,

COUNT(DIFF) AS TOTAL

FROM BINS

GROUP BY BINS;
```

Result Grid | Filter Rows:

days	total
1 - 30 days	48
121 - 150 days	43
61 - 90 days	33
31 - 60 days	25
151 - 180 days	35
91 - 120 days	35
181 - 210 days	27
331 - 360 days	1
241 - 270 days	5
211 - 240 days	4
271 - 300 days	1
301 - 330 days	1

CONTINUE...

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

- *SQL CODE:*

```
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';
```