

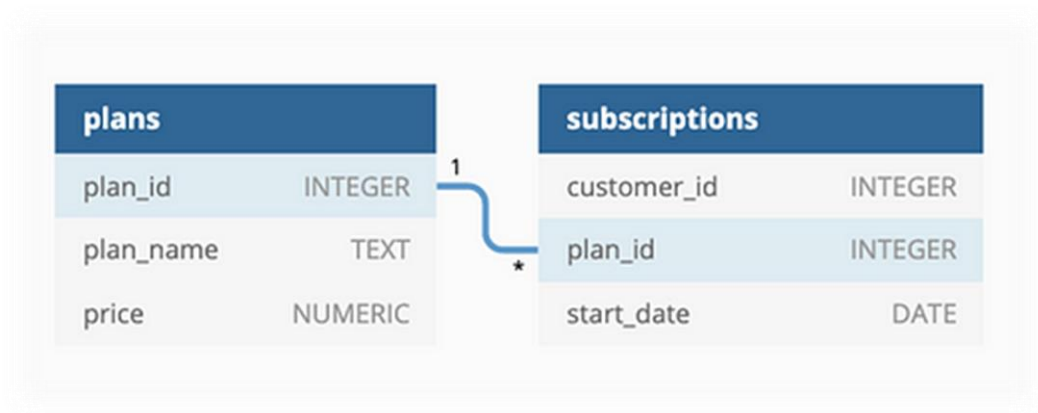
INTRODUCTION:

Foodie-Fi is a website where one can sign up to watch many cooking related shows from all over the world. Danny, who launched Foodie-Fi with few smart friends and started selling monthly and annual subscriptions, giving their customers unlimited on-demand access to exclusive food videos.

Danny created Foodie-Fi with a data driven mindset and wanted to ensure all future investment decisions and new features were decided according to what people desire to watch and what they like.

Therefore needed to analyze the data.

ERD



Data Analysis:

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

```
SELECT COUNT( DISTINCT customer_id) AS total_customer
FROM subscriptions;
```

- As customer_id may be duplicate , use key word distinct to count unique number of customer, total customer used as an alias for customer_id column.

total_customer
1000

- The total number of customer are 1000.
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?

```
SELECT MONTH(start_date) AS start_month,COUNT(*) AS num_of_customer
FROM subscriptions s
JOIN plans p ON s.plan_id=p.plan_id
WHERE p.plan_name='trial'
GROUP BY start_month
ORDER BY start_month;
```

- Extract month from start date column , join tables through plan_id inorder to count number of cutomers in trial plan in each month and ordered month in ascending order.

Result Grid		
Filter Rows: <input type="text"/>		
Export:		
	start_month	num_of_customer
▶	1	88
	2	68
	3	94
	4	81
	5	88
	6	79
	7	89
	8	88
	9	87
	10	79
	11	75
	12	84

- It show that most customer sign up for trial plan in March (3).
 - While in Feburary (2) there are least number of customers.
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(*) AS Number_of_Events
FROM subscriptions s
JOIN plans p ON s.plan_id = p.plan_id
WHERE YEAR(s.start_date) > 2020
GROUP BY p.plan_name;
```

- Join two tables using plan_id column to retrieve plan name and total number of events in each plan_name (using group by)
- In where clause extract those years from start_date column which is greater than 2020

Result Grid		
Filter Rows: <input type="text"/>		
Export:		
	plan_name	Number_of_Events
▶	churn	71
	pro monthly	60
	pro annual	63
	basic monthly	8

- Shows that large number of customer churn after 2020.
- 8, 60, 63 are upgraded to basic, pro monthly and pro annual respectively.

- While there is no one in free trial after 2020.
4. What is the customer count and percentage of customers who have churned rounded to 1 decimal place?

```
SELECT COUNT(customer_id) AS Churned_Customers_Count,
ROUND(COUNT(customer_id) / (SELECT COUNT(DISTINCT customer_id) FROM subscriptions) * 100, 1) AS Churn_Custom_Percentage
FROM subscriptions
WHERE plan_id=(SELECT plan_id FROM plans WHERE plan_name='churn');
```

- Find count of customers and their percentage by dividing number of churn customer to total number of customer and multiply by 100.
- Using inner query to select only that plan-id where plan-name is churn.

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Churned_Customers_Count	Churned_Customers_Percentage		
	307	30.7		

- There are round about 31% customer who have churn the subscription.
5. How many customers have churned straight after their initial free trial - what percentage is this rounded to the nearest whole number?

```
WITH churn_cte AS(
SELECT *,
LAG(plan_id,1) OVER(PARTITION BY customer_id ) AS previous_plan
FROM subscriptions)
SELECT COUNT(previous_plan) AS number_of_churn,
ROUND(COUNT(*)/(SELECT COUNT(DISTINCT customer_id) FROM subscriptions)*100,0) AS churn_percentage
FROM churn_cte
WHERE plan_id=(SELECT plan_id FROM plans WHERE plan_name='churn') AND previous_plan=0;
```

- Use CTE to create a temporary table that can be referred later on.
- LAG function is used to return the value of the expression from the row that precedes the current row by offset number of rows within its partition or result set.
- Then, named the result from the LAG function as previous_plan so can use it outside the CTE set.

Result Grid		
	Filter Rows:	Export:
	number_of_churn	churn_percentage
▶	92	9

- 9% customers have churn straight after their initial free trial.

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

```
WITH cte_for_nextplan AS(
SELECT *,LEAD(plan_id,1) OVER(PARTITION BY customer_id) AS nextplan
FROM subscriptions)
SELECT nextplan,count(*) AS number_of_customer,
round(count(*)/(SELECT count(DISTINCT customer_id) FROM subscriptions)*100,0) AS percentage_of_nextplan
FROM cte_for_nextplan WHERE plan_id=0 AND nextplan IS NOT NULL
GROUP BY nextplan
ORDER BY nextplan;
```

- Use CTE to look forward a number of rows and access data of that row from the current row by using LEAD function and named it as nextplan.
- For calculating the percentage, multiply the number of transactions by 100 and divide it by the number of customer.
- Apply condition where plan_id = 0 because need to find the percentage of the trial plan.



Result Grid			
	Filter Rows:	Export:	Wrap Cell Content:
	nextplan	number_of_customer	percentage_of_nextplan
▶	1	546	55
	2	325	33
	3	37	4
	4	92	9

- 54.6% of customers choose basic monthly or nextplan = 1, 32.5% of customers choose pro monthly or nextplan = 2, 3.7% of customers choose pro annual or nextplan = 3 , 9.2% of customers choose churn or nextplan = 4 after their initial trial.

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

```
SELECT plan_name, COUNT(customer_id) AS number_of_customer,  
       ROUND(COUNT(customer_id)/(SELECT COUNT(DISTINCT customer_id) FROM subscriptions)*100,0) AS percent_of_customer  
FROM subscriptions s  
JOIN plans p ON s.plan_id=p.plan_id  
WHERE s.start_date<='2020-12-31'  
GROUP BY plan_name;
```

- Select plan-name , count of customer and their percentage from subscriptions table.
- Join plans and subscriptions table using plan-id column, in order to get paln-name from plans.
- Select start-date which less than or equal to 2020-12-31



Result Grid  Filter Rows: <input type="text"/> Export: 			
	plan_name	number_of_customer	percent_of_customer
▶	trial	1000	100
	basic monthly	538	54
	pro annual	195	20
	churn	236	24
	pro monthly	479	48

- The highest number of customer sign up for trial plan having 100% customer.

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

```
SELECT COUNT(customer_id) AS upgraded_customers_count  
FROM subscriptions  
WHERE YEAR(start_date) = 2020  
AND plan_id =(SELECT plan_id FROM plans WHERE plan_name='pro annual');
```

- Find number of customer as upgraded_customer_count from subscriptions table.
- Extract only those year from start_date column which is equal to 2020.
- Use inner query to select only plan_id having plan_name equal to pro annual.

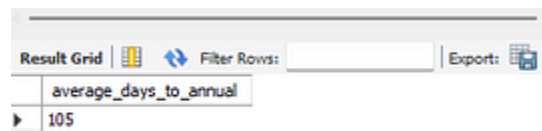
Result Grid  Filter Rows: <input type="text"/> Export: 	
	upgraded_customers_count
▶	195

- There are 195 customers who upgraded to pro annual plan in 2020.

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

```
SELECT round(AVG(DATEDIFF(s.start_date, t.trial_start_date)),0) AS average_days_to_annual
FROM subscriptions s
JOIN (SELECT customer_id,
      MIN(start_date) AS trial_start_date
      FROM subscriptions
      GROUP BY customer_id) AS t ON s.customer_id = t.customer_id
JOIN plans p ON s.plan_id = p.plan_id
WHERE p.plan_name = 'pro annual';
```

- DATEDIFF function calculates the difference in days between the start date of the pro annual plan subscription and the start date of the customer's initial subscription trial, then avg take average and then rounded to nearest whole number.
- Using CTE to find minimum start-date of trial plan using t as alias for temporary table.
- Self join s and t table in customer_id
- Then join with plans on plan_id and filter on pro annual plan.



average_days_to_annual
105

- On average a customer takes 105 days to upgrade to annual plan.

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

```
SELECT
CASE
WHEN DATEDIFF(s.start_date, t.trial_start_date) BETWEEN 0 AND 30 THEN '0-30 days'
WHEN DATEDIFF(s.start_date, t.trial_start_date) BETWEEN 31 AND 60 THEN '31-60 days'
WHEN DATEDIFF(s.start_date, t.trial_start_date) BETWEEN 61 AND 90 THEN '61-90 days'
ELSE '> 90 days'
END AS period,
ROUND(AVG(DATEDIFF(s.start_date, t.trial_start_date)),0) AS average_days_to_annual
FROM subscriptions s
JOIN (SELECT customer_id, MIN(start_date) AS trial_start_date
      FROM subscriptions
      GROUP BY customer_id) AS t ON s.customer_id = t.customer_id
JOIN plans p ON s.plan_id = p.plan_id WHERE plan_name = 'pro annual'
GROUP BY period
ORDER BY period;
```

- Here categorize total average days that get from previous query.
- As if the day between 0 or 30 then named as 0-30 days, if between 31 to 60 named as 31-60 days and if between 61 to 90 named as 61-90 days else greater than 90, categories named as period.

Result Grid			Filter Rows:	Export:
	period	average_days_to_annual		
▶	> 90 days	153		
	0-30 days	10		
	31-60 days	42		
	61-90 days	71		

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

```
WITH cte_for_nextplan AS(
    SELECT *, LEAD(plan_id,1) OVER(PARTITION BY customer_id) AS nextplan
    FROM subscriptions)
SELECT COUNT(customer_id) AS downgraded_customer_number
FROM cte_for_nextplan n
LEFT JOIN plans p ON p.plan_id=n.plan_id
WHERE p.plan_name='basic monthly' and p.plan_name='pro monthly' and year(n.start_date)='2020';
```

- Using LEAD function to find out the customers' nextplans.
- Then count different value of customers using COUNT DISTINCT.
- filter the plan name = 'pro monthly' and basic monthly, and only extract year which is equal to 2020.

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	downgraded_customer_number				
▶	0				

- Show that no customer downgraded from pro monthly to basic monthly plan in 2020.