

SQL PORTFOLIO PROJECT

FOODIE – FI CASE STUDY

BY HANIFA IBRAHIM

GITHUB: <https://github.com/Hanifa-Ibrahim/Foodie-Fi>

MEDIUM: <https://medium.com/@honeyfa2002/foodie-fi-case-study-1fd3fa98db53>

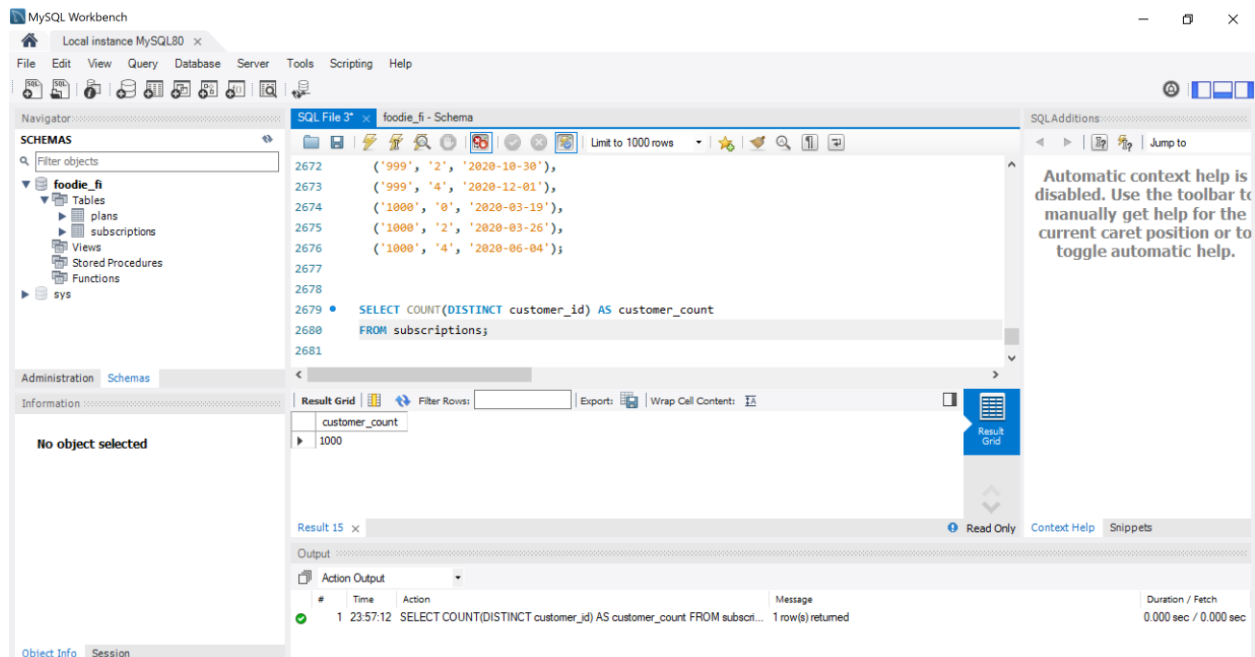
QUESTION: 1

How many customers Foodie-Fi ever had?

QUERY

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

OUTPUT



The screenshot displays the MySQL Workbench interface. The 'Schemas' pane on the left shows the 'foodie-fi' database selected, with 'subscriptions' under the 'Tables' folder. The 'SQL File 3*' editor in the center contains the following query:

```
2672 ('999', '2', '2020-10-30'),  
2673 ('999', '4', '2020-12-01'),  
2674 ('1000', '0', '2020-03-19'),  
2675 ('1000', '2', '2020-03-26'),  
2676 ('1000', '4', '2020-06-04'),  
2677  
2678  
2679 • SELECT COUNT(DISTINCT customer_id) AS customer_count  
2680 FROM subscriptions;  
2681
```

The 'Result Grid' on the right shows the output of the query:

customer_count
1000

The 'Output' pane at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	23:57:12	SELECT COUNT(DISTINCT customer_id) AS customer_count FROM subscri...	1 row(s) returned	0.000 sec / 0.000 sec

QUESTION: 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.

QUERY

```
SELECT plan_id, month(start_date) as month, count(month(start_date)) as count from subscriptions
group by month(start_date), plan_id
having plan_id=0;
```

OUTPUT

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
2676 ('1000', '4', '2020-06-04');
2677
2678 • SELECT COUNT(*) AS Customers
2679 FROM subscriptions;
2680
2681 • SELECT plan_id, month(start_date) as month, count(month(start_date)) as count from subscription
2682 group by month(start_date), plan_id
2683 having plan_id=0;
```

The Results window displays the output of the query in a table format:

plan_id	month	count
0	8	88
0	9	87
0	1	88
0	12	84
0	2	68
0	6	79
0	11	75
0	3	94
0	5	88
0	7	89
0	4	81
0	10	79

The interface also shows the Navigator pane on the left with the 'foodie-fi' schema selected, and the SQL Additions pane on the right with a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

QUESTION: 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.

QUERY

```
SELECT plans.plan_name, sp.plan_id, year(start_date) as year, count(year(start_date))
as Count_of_events from subscriptions as sp
JOIN plans on sp.plan_id=plans.plan_id
where year(start_date)>2020
Group by plans.plan_name, plan_id, year(start_date);
```

OUTPUT

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
2682 group by month(start_date), plan_id
2683 having plan_id=0;
2684
2685 • SELECT plans.plan_name, sp.plan_id, year(start_date) as year, count(year(start_date))
2686 as Count_of_events from subscriptions as sp
2687 JOIN plans on sp.plan_id=plans.plan_id
2688 where year(start_date)>2020
2689 Group by plans.plan_name, plan_id, year(start_date);
```

The Results window displays the following data:

plan_name	plan_id	year	Count_of_events
churn	4	2021	71
pro monthly	2	2021	60
pro annual	3	2021	63
basic monthly	1	2021	8

The Navigator panel on the left shows the database structure, including the 'foodie-fi' schema and its tables: plans, subscriptions, Views, Stored Procedures, Functions, and sys. The Information panel at the bottom left indicates 'No object selected'.

QUESTION: 4

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

QUERY

```
select plan_name, count((select count(distinct customer_id) from subscriptions)) as  
count_of_churned,
```

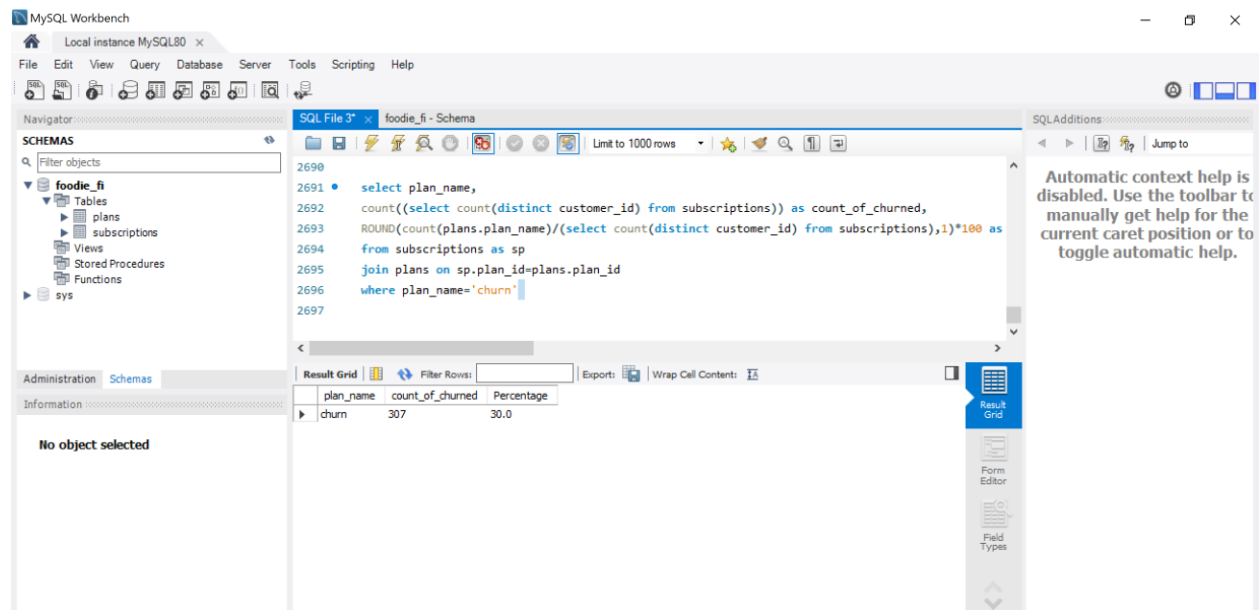
```
ROUND(count(plans.plan_name) / (select count(distinct customer_id) from  
subscriptions),1)*100 as Percentage
```

```
from subscriptions as sp
```

```
join plans on sp.plan_id=plans.plan_id
```

```
where plan_name='churn'
```

OUTPUT



The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
select plan_name,  
count((select count(distinct customer_id) from subscriptions)) as count_of_churned,  
ROUND(count(plans.plan_name)/(select count(distinct customer_id) from subscriptions),1)*100 as  
Percentage  
from subscriptions as sp  
join plans on sp.plan_id=plans.plan_id  
where plan_name='churn'
```

The Results window displays the output in a table with the following columns: plan_name, count_of_churned, and Percentage. The data row shows:

plan_name	count_of_churned	Percentage
churn	307	30.0

The SQL Navigator on the left shows the database structure, including the 'foodie' database and its tables: plans, subscriptions, views, stored procedures, and functions. The SQL Editor toolbar includes options for running the query, saving, and other standard SQL operations. The Results window also includes a 'Result Grid' button and a 'Form Editor' button.

QUESTION: 5

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

QUERY

```
SELECT plan_id, count(customer_id) as count_customer,  
Round(count(customer_id)/(select count(customer_id) from subscriptions),3)*100 as  
Percentage  
from subscriptions where plan_id=4  
and customer_id in(select customer_id from subscriptions where plan_id=0);
```

OUTPUT

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
2693 ROUND(count(plans.plan_name)/(select count(distinct customer_id) from subscriptions),1)*100 as  
2694 from subscriptions as sp  
2695 join plans on sp.plan_id=plans.plan_id  
2696 where plan_name='churn';  
2697  
2698 • SELECT plan_id, count(customer_id) as count_customer,  
2699 Round(count(customer_id)/(select count(customer_id) from subscriptions),3)*100 as Percentage  
2700 from subscriptions where plan_id=4  
2701 and customer_id in(select customer_id from subscriptions where plan_id=0);  
2702
```

The Results window shows the following output:

plan_id	count_customer	Percentage
4	307	11.600

The Output window shows the following action output:

#	Time	Action	Message	Duration / Fetch
4	23:16:42	select plan_name, count(select count(distinct customer_id) from subscrip...	1 row(s) returned	0.015 sec / 0.000 sec
5	23:22:45	SELECT plan_id, count(customer_id) as count_customer, Round(count(cust...	1 row(s) returned	0.015 sec / 0.000 sec

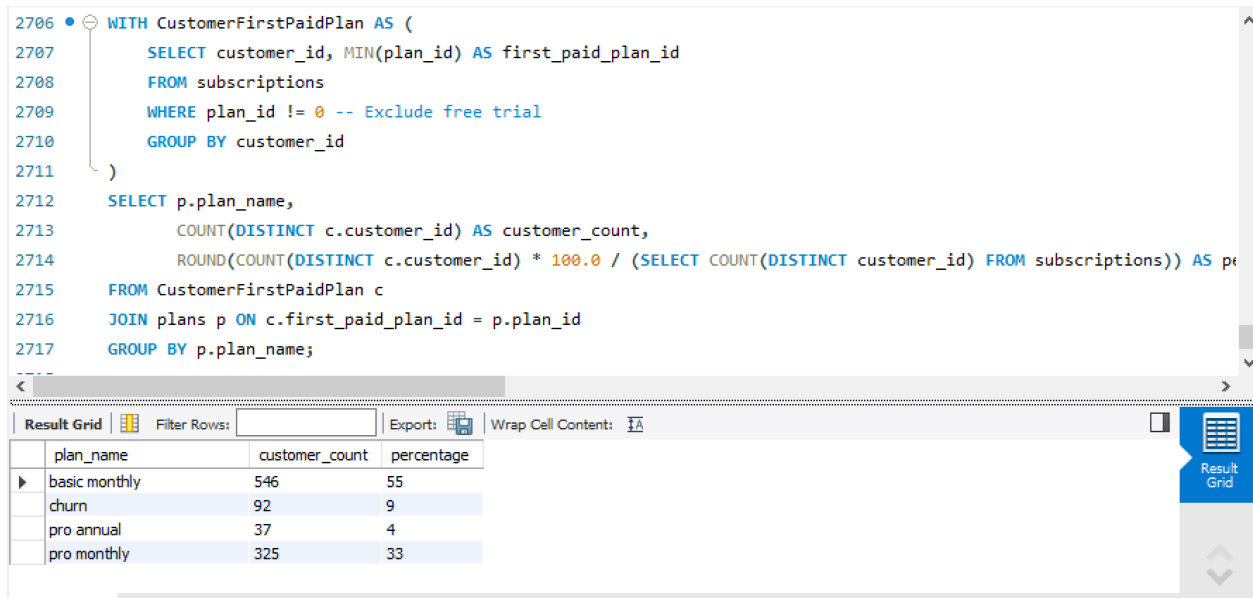
QUESTION: 6

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

QUERY

```
WITH CustomerFirstPaidPlan AS (  
    SELECT customer_id, MIN(plan_id) AS first_paid_plan_id  
    FROM subscriptions  
    WHERE plan_id != 0 -- Exclude free trial  
    GROUP BY customer_id  
)  
SELECT p.plan_name,  
       COUNT(DISTINCT c.customer_id) AS customer_count,  
       ROUND(COUNT(DISTINCT c.customer_id) * 100.0 / (SELECT COUNT(DISTINCT customer_id) FROM  
subscriptions)) AS percentage  
FROM CustomerFirstPaidPlan c  
JOIN plans p ON c.first_paid_plan_id = p.plan_id  
GROUP BY p.plan_name;
```

OUTPUT



```
2706 WITH CustomerFirstPaidPlan AS (  
2707     SELECT customer_id, MIN(plan_id) AS first_paid_plan_id  
2708     FROM subscriptions  
2709     WHERE plan_id != 0 -- Exclude free trial  
2710     GROUP BY customer_id  
2711 )  
2712 SELECT p.plan_name,  
2713        COUNT(DISTINCT c.customer_id) AS customer_count,  
2714        ROUND(COUNT(DISTINCT c.customer_id) * 100.0 / (SELECT COUNT(DISTINCT customer_id) FROM subscriptions)) AS percentage  
2715 FROM CustomerFirstPaidPlan c  
2716 JOIN plans p ON c.first_paid_plan_id = p.plan_id  
2717 GROUP BY p.plan_name;
```

plan_name	customer_count	percentage
basic monthly	546	55
churn	92	9
pro annual	37	4
pro monthly	325	33

QUESTION: 7

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

QUERY

```
WITH cte1 AS (  
  SELECT *, RANK() OVER(PARTITION BY customer_id ORDER BY start_date DESC) AS ranking  
  FROM subscriptions  
  WHERE start_date <= '2020-12-31'  
)  
cte2 AS (  
  SELECT p.plan_name, COUNT(cte1.plan_id) AS num_plans  
  FROM cte1 INNER JOIN plans p  
  ON cte1.plan_id = p.plan_id  
  WHERE cte1.ranking = 1  
  GROUP BY p.plan_name  
  ORDER BY num_plans DESC  
)  
SELECT plan_name, num_plans,  
  ROUND((num_plans/(SUM(num_plans) OVER())*100), 2) AS pct_plans  
FROM cte2;
```

OUTPUT

```
2719 • WITH cte1 AS (  
2720     SELECT *, RANK() OVER(PARTITION BY customer_id ORDER BY start_date DESC) AS ranking  
2721     FROM subscriptions  
2722     WHERE start_date <= '2020-12-31'  
2723 ),  
2724 cte2 AS (  
2725     SELECT p.plan_name, COUNT(cte1.plan_id) AS num_plans  
2726     FROM cte1 INNER JOIN plans p  
2727     ON cte1.plan_id = p.plan_id  
2728     WHERE cte1.ranking = 1  
2729     GROUP BY p.plan_name  
2730     ORDER BY num_plans DESC  
2731 )  
2732 SELECT plan_name, num_plans,  
2733 ROUND((num_plans/(SUM(num_plans) OVER())*100), 2) AS pct_plans
```

plan_name	num_plans	pct_plans
pro monthly	1304	32.60
churn	944	23.60
basic monthly	896	22.40
pro annual	780	19.50
trial	76	1.90

QUESTION: 8



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

QUERY

```
SELECT COUNT(DISTINCT s.customer_id) AS num_customers  
FROM subscriptions s INNER JOIN plans p  
ON s.plan_id = p.plan_id  
WHERE p.plan_name = "pro annual" AND YEAR(s.start_date) = 2020;
```


OUTPUT

```
2736 • SELECT COUNT(DISTINCT s.customer_id) AS num_customers
2737 FROM subscriptions s INNER JOIN plans p
2738 ON s.plan_id = p.plan_id
2739 WHERE p.plan_name = "pro annual" AND YEAR(s.start_date) = 2020;
```

<	
Result Grid	
Filter Rows: <input type="text"/>	
Export: 	
Wrap Cell Content: 	
	num_customers
▶	195

QUESTION: 9


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

QUERY

```
WITH cte1 AS (
  SELECT * FROM subscriptions
  WHERE plan_id = 0
),
cte2 AS (
  SELECT * FROM subscriptions
  WHERE plan_id = 3
)
SELECT ROUND(AVG(DATEDIFF(cte2.start_date, cte1.start_date)), 1) AS avg_days
FROM cte1 JOIN cte2
ON cte1.customer_id = cte2.customer_id;
```

OUTPUT

```
2741 • WITH cte1 AS (  
2742     SELECT * FROM subscriptions  
2743     WHERE plan_id = 0  
2744 ),  
2745 cte2 AS (  
2746     SELECT * FROM subscriptions  
2747     WHERE plan_id = 3  
2748 )  
2749 SELECT ROUND(AVG(DATEDIFF(cte2.start_date, cte1.start_date)), 1) AS avg_days  
2750 FROM cte1 JOIN cte2  
2751 ON cte1.customer_id = cte2.customer_id;  
2752  
2753
```

<	
Result Grid	Filter Rows: <input type="text"/>
Export: 	Wrap Cell Content: 
avg_days	
▶	104.6

QUESTION: 10

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

QUERY

SELECT

CASE

WHEN days_difference >= 0 AND days_difference <= 30 THEN '0-30 days'

WHEN days_difference > 30 AND days_difference <= 60 THEN '31-60 days'

ELSE 'More than 60 days'

END AS period,

COUNT(customer_id) AS No_of_customers,

AVG(days_difference) AS AVG_Days_to_reach_annual_program

```

FROM (

SELECT

    s2.customer_id,

    DATEDIFF(

        (SELECT MIN(start_date) FROM subscriptions AS s1 WHERE s1.customer_id = s2.customer_id AND
plan_id = 3),

        (SELECT MIN(start_date) FROM subscriptions AS s3 WHERE s3.customer_id = s2.customer_id)) AS
days_difference

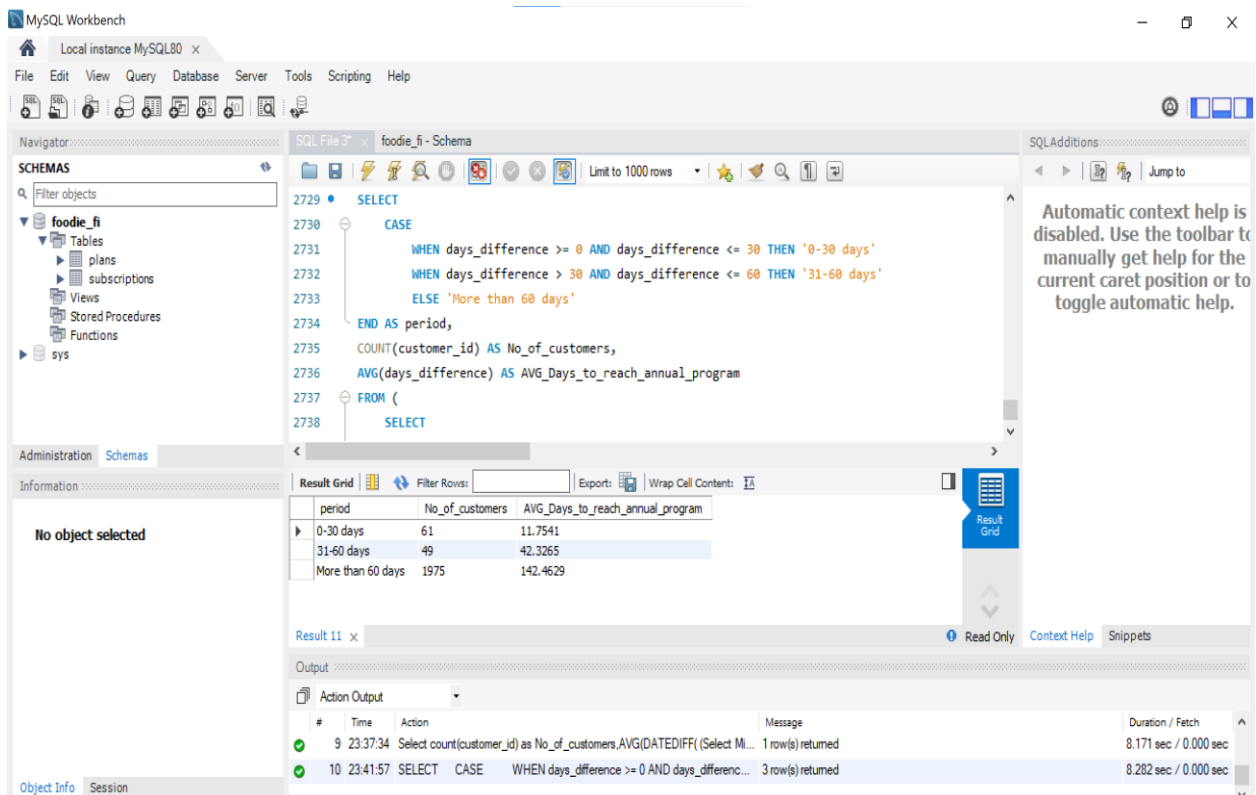
FROM subscriptions AS s2 WHERE plan_id != 3 AND plan_id != 4) AS differences

GROUP BY period

ORDER BY period;

```

OUTPUT



The screenshot shows the MySQL Workbench interface. The SQL Editor contains a query that calculates the average days difference for three periods: 0-30 days, 31-60 days, and more than 60 days. The Results window displays the output of this query.

SQL Query:

```

SELECT
  CASE
    WHEN days_difference >= 0 AND days_difference <= 30 THEN '0-30 days'
    WHEN days_difference > 30 AND days_difference <= 60 THEN '31-60 days'
    ELSE 'More than 60 days'
  END AS period,
  COUNT(customer_id) AS No_of_customers,
  AVG(days_difference) AS AVG_Days_to_reach_annual_program
FROM (
  SELECT

```

Result Grid:

period	No_of_customers	AVG_Days_to_reach_annual_program
0-30 days	61	11.7541
31-60 days	49	42.3265
More than 60 days	1975	142.4629

Action Output:

#	Time	Action	Message	Duration / Fetch
9	23:37:34	Select count(customer_id) as No_of_customers,AVG(DATEDIFF(Select M...	1 row(s) returned	8.171 sec / 0.000 sec
10	23:41:57	SELECT CASE WHEN days_difference >= 0 AND days_differenc...	3 row(s) returned	8.282 sec / 0.000 sec

QUESTION: 11

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

QUERY

select plan_id, count(distinct customer_id) as downgraded_from_annual_to_basic from
subscriptions

where plan_id =1

and customer_id in

(select distinct customer_id from subscriptions where plan_id=2 and year(start_date) = 2020);

OUTPUT

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
ORDER BY period;  
select plan_id, count(distinct customer_id) as downgraded_from_annual_to_basic from subscriptions  
where plan_id =1  
and customer_id in  
(select distinct customer_id from subscriptions where plan_id=2 and year(start_date)=2020);
```

The Results window displays the following output:

plan_id	downgraded_from_annual_to_basic
1	163

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
10	23:41:57	SELECT CASE WHEN days_difference >= 0 AND days_differenc...	3 row(s) returned	8.282 sec / 0.000 sec
11	23:45:01	select plan_id, count(distinct customer_id) as downgraded_from_annual_to_...	1 row(s) returned	0.000 sec / 0.000 sec