Martin Thai

Data Analytics Project: Danny's Diner

In this project, I use Danny Ma's case study here. Thank you Danny and if you are interested in more case studies, I suggest you give him a follow on LinkedIn. In this project, we will be using SQL, specifically MySql in VSCode to analyze the relevant datasets.

Introduction:

In this case study, I take a look at the data gathered by Danny's Diner, a Japanese restaurant that sells ramen, sushi, and curry, in the sole month of January. The goal of this project is to simulate analyzing data from a real world restaurant (by understanding visiting patterns, money spent, and popular items) and to use this data to increase sales and customer satisfaction. The problem at hand in Danny's Diner is that the diner wants to see if they should expand their existing customer loyalty program or simply resume/cut it off.

What I did in a nutshell:

- Leveraged MySQL's aggregate/join functions and CTES, analyzing customer data, extracting actionable insights on visiting patterns, spending habits, and menu favorites, to optimize menu offerings and maximize profitability
- Evaluated effectiveness of restaurant's loyalty program using data-driven analyses, and suggested recommendations for its expansion, influencing strategic decisions and driving revenue growth for the establishment

Datasets:

There are 3 datasets that I will be using:

- sales
- menu
- members

Below is the **Entity Relationship Diagram**:

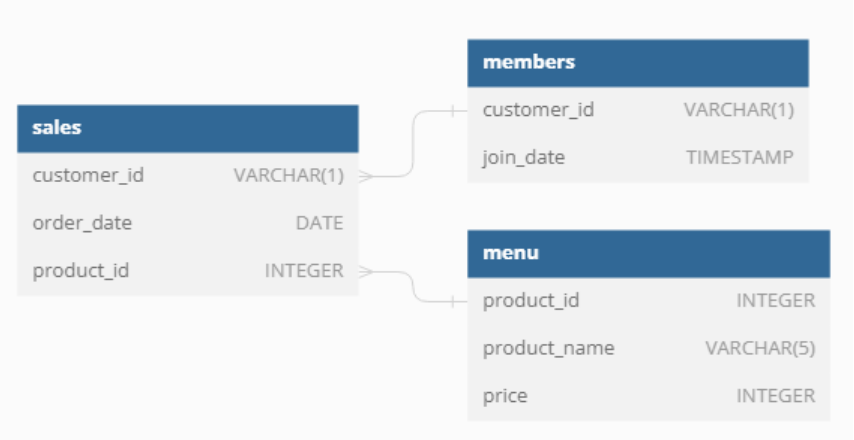

Table 1: sales

The sales table contains customer_id, order_date, and product_id (refers to what menu item was ordered).

customer_id	order_date	product_id
A	2021-01-01	1
A	2021-01-01	2
A	2021-01-07	2
A	2021-01-10	3
A	2021-01-11	3
A	2021-01-11	3
B	2021-01-01	2
B	2021-01-02	2
B	2021-01-04	1
B	2021-01-11	1
B	2021-01-16	3
B	2021-02-01	3
C	2021-01-01	3
C	2021-01-01	3
C	2021-01-07	3

Table 2: menu

The menu table has product_id, product_name, and price.

product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12

Table 3: members

The members table has customer_id, and join_date (the date when the customer joined the loyalty program).

customer_id	join_date
A	2021-01-07
B	2021-01-09

Questions to Answer:

1. What is the total amount each customer spent at the restaurant?

I used GROUP BY and SUM functions to find out how much each customer spent at the restaurant. In addition, I used JOIN to relate the sales and menu tables.

```
SELECT sales.customer_id, SUM(menu.price) AS total_amt_spent
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
GROUP BY sales.customer_id
```

Query Result:

customer_id	total_amt_spent
A	76
B	74
C	36

Answer:

- Customer A spent $76.
- Customer B spent $74.
- Customer C spent $36.

2. How many days has each customer visited the restaurant?

I used GROUP BY and COUNT to find out how many days each customer came to the diner.

```
SELECT customer_id, COUNT(DISTINCT(order_date)) AS days_visited
FROM `martintthai`.`sales` sales
GROUP BY customer_id
```

Query Result:

customer_id	days_visited
A	4
B	6
C	2

Answer:

- Customer A visited 4 times.
- Customer B visited 6 times.
- Customer C visited 2 times.

3. What was the first item on the menu purchased by each customer?

I used GROUP BY and MIN(order_date)-> (this chooses the earliest date) to find the first item purchased by each customer.

Query Result:

customer_id	date_ordered	first_item_purchased
A	Fri Jan 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	sushi
B	Fri Jan 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	curry
C	Fri Jan 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	ramen

Answer:

- Customer A first ordered sushi.
- Customer B first ordered curry.
- Customer C first ordered ramen.

4. What is the most purchased item on the menu and how many times was it purchased by all customers?

I used GROUP BY product_name and COUNT to find out how many times each item was purchased. I also used JOIN on sales and menu.

```
SELECT product_name, COUNT(product_name) AS times_purchased
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
GROUP BY product_name
```

Query Result:

product_name	times_purchased
sushi	3
curry	4
ramen	8

Answer:

- Ramen was the most purchased item (8 times).

5. Which item was the most popular for each customer?

I did the same as above but instead of GROUP BY just product_name, I also grouped by customer_id.

```
SELECT customer_id, product_name, COUNT(product_name) AS times_purchased
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
GROUP BY customer_id, product_name
ORDER BY times_purchased DESC
```

Query Result:

customer_id	product_name	times_purchased
A	ramen	3
C	ramen	3
A	curry	2
B	curry	2
B	sushi	2
B	ramen	2
A	sushi	1

Answer:

- Customer A's most popular item was ramen.
- Customer B's most popular item was a tie between curry, sushi, and ramen.
- Customer C's most popular item was ramen.

6. Which item was purchased first by each customer after they became a member?

I used JOIN twice to relate sales, menu, and members as we needed all of the data on every table. I did order_date - join_date to find the time passed until their first

purchase. I then filtered using WHERE to only display the rows where join_date is before the order_date to only see orders AFTER they became a member.

```
SELECT *, order_date-join_date AS time_passed_til_first_purchase
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
JOIN `martintthai`.`members` members ON sales.customer_id =
members.customer_id
WHERE join_date < order_date
ORDER BY time_passed_til_first_purchase ASC
```

Query Result:

customer_id	order_date	product_id	product_name	price	join_date	time_passed_til_first_purchase
B	Mon Jan 11 2021 00:00:00 GMT-0800 (Pacific Standard Time)	1	sushi	10	Sat Jan 09 2021 00:00:00 GMT-0800 (Pacific Standard Time)	2
A	Sun Jan 10 2021 00:00:00 GMT-0800 (Pacific Standard Time)	3	ramen	12	Thu Jan 07 2021 00:00:00 GMT-0800 (Pacific Standard Time)	3
A	Mon Jan 11 2021 00:00:00 GMT-0800 (Pacific Standard Time)	3	ramen	12	Thu Jan 07 2021 00:00:00 GMT-0800 (Pacific Standard Time)	4
A	Mon Jan 11 2021 00:00:00 GMT-0800 (Pacific Standard Time)	3	ramen	12	Thu Jan 07 2021 00:00:00 GMT-0800 (Pacific Standard Time)	4
B	Sat Jan 16 2021 00:00:00 GMT-0800 (Pacific Standard Time)	3	ramen	12	Sat Jan 09 2021 00:00:00 GMT-0800 (Pacific Standard Time)	7
B	Mon Feb 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	3	ramen	12	Sat Jan 09 2021 00:00:00 GMT-0800 (Pacific Standard Time)	92

Answer:

- Customer A's first purchase since membership is ramen.
- Customer B's first purchase since membership is sushi.
- Customer C is not a member.

7. Which item was purchased just before the customer became a member?

I did the same as the previous query except I filtered to only select rows where join_date was after order_date to only see orders that came before they were a

member. I also did join_date - order_date to find the time passed until each customer became a member.

```
SELECT *, join_date - order_date AS time_passed_til_became_member
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
JOIN `martintthai`.`members` members ON sales.customer_id =
members.customer_id
WHERE join_date > order_date
ORDER BY time_passed_til_became_member ASC
```

Query Result:

customer_id	order_date	product_id	product_name	price	join_date	time_passed_til_became_member
B	Mon Jan 04 2021 00:00:00 GMT-0800 (Pacific Standard Time)	1	sushi	10	Sat Jan 09 2021 00:00:00 GMT-0800 (Pacific Standard Time)	5
A	Fri Jan 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	1	sushi	10	Thu Jan 07 2021 00:00:00 GMT-0800 (Pacific Standard Time)	6
A	Fri Jan 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	2	curry	15	Thu Jan 07 2021 00:00:00 GMT-0800 (Pacific Standard Time)	6
B	Sat Jan 02 2021 00:00:00 GMT-0800 (Pacific Standard Time)	2	curry	15	Sat Jan 09 2021 00:00:00 GMT-0800 (Pacific Standard Time)	7
B	Fri Jan 01 2021 00:00:00 GMT-0800 (Pacific Standard Time)	2	curry	15	Sat Jan 09 2021 00:00:00 GMT-0800 (Pacific Standard Time)	8

Answer:

- Customer A last ordered sushi and curry before they became a member.
- Customer B last ordered sushi before they became a member.
- Customer C is not a member.

8. What is the total items and amount spent for each member before they became a member?

I used JOIN to relate all the tables. I filtered using WHERE to only find orders that occurred before membership. I also used GROUP BY, COUNT, and SUM (two different queries for COUNT (total items) and SUM (amount spent)).

```
SELECT sales.customer_id, COUNT(sales.product_id) AS total_items_bought
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
JOIN `martintthai`.`members` members ON sales.customer_id =
members.customer_id
WHERE join_date > order_date
GROUP BY sales.customer_id
```

Query Result:

customer_id	total_items_bought
B	3
A	2

```
SELECT sales.customer_id, SUM(menu.price) AS total_amt_spent
FROM `martintthai`.`sales` sales
JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
JOIN `martintthai`.`members` members ON sales.customer_id =
members.customer_id
WHERE join_date > order_date
GROUP BY sales.customer_id
```

Query Result:

customer_id	total_amt_spent
B	40
A	25

Answer:

- Customer A bought 2 items and spent $25 before they became a member.
- Customer B bought 3 items and spent $40 before they became a member.

9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?

I used LEFT JOIN to relate sales, members, and menu and used CASE WHEN to convert price to points. I then used GROUP BY and SUM to find out the total points of each customer.

For the CASE WHEN statement, when product_id = 1 (which is sushi) then the total points is price * 2 * 10 (because each dollar is worth 10 points and sushi points are doubled). The rest of the product ids (2 and 3) are just price * 10.

```
SELECT sales.customer_id,
SUM(CASE WHEN sales.product_id = 1 THEN price * 20
WHEN sales.product_id = 2 THEN price * 10
WHEN sales.product_id = 3 THEN price * 10
END) AS total_points
FROM `martintthai`.`sales` sales
LEFT JOIN `martintthai`.`members` members ON sales.customer_id =
members.customer_id
LEFT JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
GROUP BY sales.customer_id
```

Query Result:

customer_id	total_points
A	860
B	940
C	360

Answer:

- Customer A has 860 total points.
- Customer B has 940 total points.
- Customer C has 360 total points.

10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi - how many points do customer A and B have at the end of January?

Step 1. I first created a temporary table with all the data including a new column (end_double_pts_week) which finds the last day where the double points interval applies.

Step 2. I then used this table and a CASE WHEN statement to double all points on orders between join_date and end_double_pts_week. If the purchase wasn't in this time interval, then apply the double points only if it is sushi.

Step 3. I used GROUP BY to group by customer and filtered using WHERE to only select orders in the month of January.

```
WITH sales_with_membership_wk AS
    (SELECT sales.customer_id,
    sales.order_date,
```

```
    members.join_date,
    DATE_ADD(join_date, INTERVAL 6 DAY) AS end_double_pts_week,
    menu.product_name,
    menu.price
    FROM `martintthai`.`sales` sales
    JOIN `martintthai`.`menu` menu ON sales.product_id = menu.product_id
    JOIN `martintthai`.`members` members ON sales.customer_id =
members.customer_id)

SELECT *,
SUM(CASE WHEN order_date BETWEEN join_date AND end_double_pts_week THEN
price * 20
WHEN (order_date NOT BETWEEN join_date AND end_double_pts_week) AND
product_name = 'sushi' THEN price * 20
ELSE price * 10
END) points
FROM sales_with_membership_wk
WHERE order_date < '2021-02-01'
GROUP BY customer_id
```

Query Result:

customer_id	points
B	820
A	1370

Answer:

- Customer A has 1370 total points.
- Customer B has 820 total points

Conclusions and Final Thoughts:

1. Customer A and Customer B spent the most amount of money at the diner, both spending more than Customer C. Customer A and B are also members of the loyalty program, which is a good sign that the loyalty program is bringing in more revenue.
2. The most frequent visitor is Customer B (6 times).
3. The most popular item by far is ramen. All the customers seem to have enjoyed it.

4. Assuming that in the first week a customer joins the program they earn 2x points on all items, not just sushi - Customer A would have 1370, B would have 820, and C would have 360.
5. Customer A made the most of their membership, ordering a lot from the diner in the week after they became a member.
6. There is good indication that the loyalty program is successful in bringing in the diner more revenue, so they should keep it around for now and see how if it brings in more revenue/customers in the month of February.