SQL Case Study 1: Dannys Diner

Introduction

Danny seriously loves Japanese food so in the beginning of 2021, he decides to embark upon a risky venture and opens up a cute little restaurant that sells his 3 favourite foods: sushi, curry and ramen.

Danny’s Diner is in need of your assistance to help the restaurant stay afloat - the restaurant has captured some very basic data from their few months of operation but have no idea how to use their data to help them run the business.

Problem Statement

Danny wants to use the data to answer a few simple questions about his customers, especially about their visiting patterns, how much money they’ve spent and also which menu items are their favourite. Having this deeper connection with his customers will help him deliver a better and more personalised experience for his loyal customers.

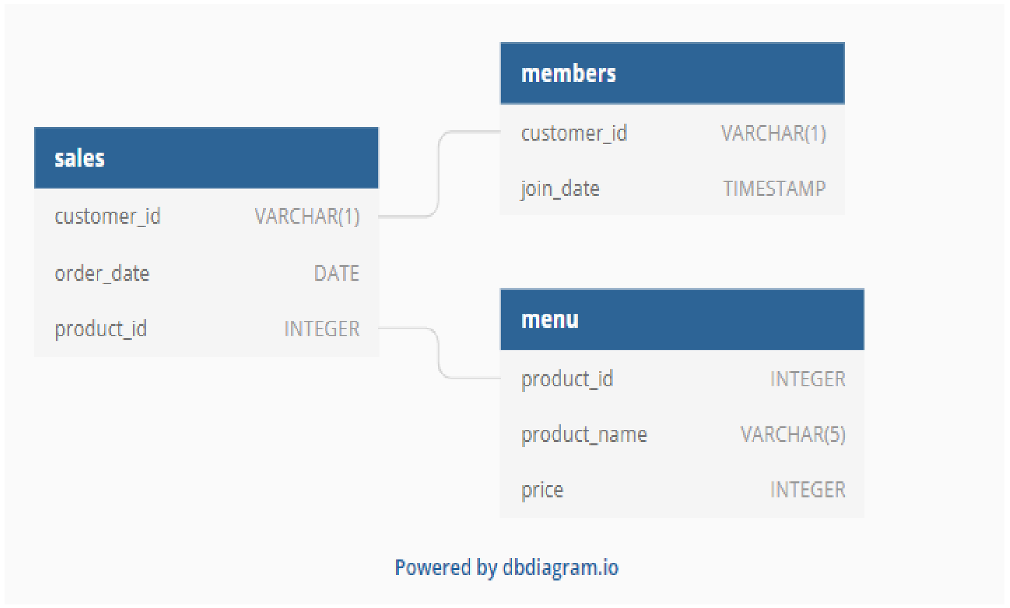
He plans on using these insights to help him decide whether he should expand the existing customer loyalty program - additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

Danny has shared with you 3 key datasets for this case study:

- **sales**
  - customer_id *varchar(1)*
  - order_date *date*
  - product_id *integer*
- **menu**

        - product_id *integer*
        - product_name *varchar(5)*
        - price *integer*
    - **members**
        - customer_id *varchar(1)*
        - join_date *timestamp*



Case Study Questions

1. What is the total amount each customer spent at the restaurant?
2. How many days has each customer visited the restaurant?
3. What was the first item from the menu purchased by each customer?
4. What is the most purchased item on the menu and how many times was it purchased by all customers?
5. Which item was the most popular for each customer?
6. Which item was purchased first by the customer after they became a member?
7. Which item was purchased just before the customer became a member?
8. What is the total items and amount spent for each member 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?
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?

Bonus Questions

1. Create a basic data table that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL. Also, add an aditional field to find-out customers who have taken the membership.
2. Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program. Create a table as per Danny's requirement.

Database Connection

In [1]:
```python
# Importing libraries

import mysql.connector as conn #To make connection with database

import pandas as pd #To load retrival data in the form of pandas dataframe

from tabulate import tabulate #To show the resultant dataframe into table format
```

In [2]:
```python
# Function to connect with the dannys_diner database

def database_connection():
    db = conn.connect(host='localhost',user='root', password='#Datatalks@1',database='dannys_diner')
    global csr
    csr = db.cursor()
    return db

database_connection()
```

Out[2]:
```
<mysql.connector.connection_cext.CMySQLConnection at 0x2c9191b97d0>
```

In [3]:
```python
# Function to execute sql query and output the retrival data

def query_execution(query):
    csr.execute(query)
    output = csr.fetchall()
    df = pd.DataFrame(output, columns = csr.column_names)
    print(tabulate(df, headers='keys', tablefmt='psql', showindex=False))
```

Case Study Solutions

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

In [4]:
```python
query = """

SELECT
        s.customer_id,
        SUM(price) AS total_amount
```

```
FROM sales s
LEFT JOIN menu m
ON s.product_id = m.product_id
GROUP BY s.customer_id;

"""

print('\nTotal amount spent by each customer at the restaurant is ($): \n')
query_execution(query)
```

```
Total amount spent by each customer at the restaurant is ($):

+---------------+----------------+
| customer_id   |   total_amount |
|---------------+----------------|
| A             |             76 |
| B             |             74 |
| C             |             36 |
+---------------+----------------+
```

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

In [5]:
```
query = """

SELECT
        customer_id,
    COUNT(DISTINCT order_date) AS visits_in_days
FROM sales
GROUP BY customer_id;

"""

print('\nEach customer visited the restaurant in days : \n')
query_execution(query)
```

```
Each customer visited the restaurant in days :

+---------------+------------------+
| customer_id   |   visits_in_days |
|---------------+------------------|
| A             |                4 |
| B             |                6 |
| C             |                2 |
+---------------+------------------+
```

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

In [6]:
```
query = """

SELECT
        customer_id,
    product_name as first_item,
    order_date
FROM (
        SELECT
                customer_id,
                order_date,
                product_name,
        ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY order_date) AS rn
        FROM sales s
        LEFT JOIN menu m
        ON s.product_id = m.product_id
) AS sq
WHERE sq.rn = 1;

"""

print('\nFirst item from the menu purchased by each customer : \n')
query_execution(query)
```

```
First item from the menu purchased by each customer :

+---------------+-------------+--------------+
| customer_id   | first_item  | order_date   |
|---------------+-------------+--------------|
| A             | sushi       | 2021-01-01   |
| B             | curry       | 2021-01-01   |
| C             | ramen       | 2021-01-01   |
+---------------+-------------+--------------+
```

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

In [7]:
```
query = """

WITH cte as (
        SELECT
```

```
                product_name,
                count(*) AS number_of_orders
        FROM sales s
        LEFT JOIN menu m
        ON s.product_id = m.product_id
        GROUP BY product_name
)
SELECT
        product_name AS most_purchased_item,
    number_of_orders AS most_purchased_item_frequency
FROM cte
WHERE number_of_orders = (SELECT MAX(number_of_orders) FROM cte);

"""

print('\nMost purchased item & frequency of items purchased by all the customer: \n')
query_execution(query)
```

```
Most purchased item & frequency of items purchased by all the customer:

+------------------------+--------------------------------+
| most_purchased_item    |   most_purchased_item_frequency |
|------------------------+--------------------------------|
| ramen                  |                              8 |
+------------------------+--------------------------------+
```

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

In [8]:
```
query = """

WITH cte AS(
        SELECT
                customer_id,
                product_name,
                COUNT(s.product_id) AS ic
        FROM sales s
        JOIN menu m
        ON s.product_id = m.product_id
        GROUP BY customer_id, product_name
)
SELECT
        customer_id,
    product_name AS popular_items
FROM (
        SELECT *,
                DENSE_RANK() OVER(PARTITION BY customer_id ORDER BY ic DESC) AS rnk
        FROM cte
) AS sq
WHERE sq.rnk = 1;

"""

print('\nMost popular item for each customer: \n')
query_execution(query)
```

```
Most popular item for each customer:

+---------------+-----------------+
| customer_id   | popular_items   |
|---------------+-----------------|
| A             | ramen           |
| B             | curry           |
| B             | sushi           |
| B             | ramen           |
| C             | ramen           |
+---------------+-----------------+
```

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

In [9]:
```
query = """

WITH cte AS (
SELECT
        s.customer_id,
    s.product_id,
    s.order_date,
    m.join_date,
        DENSE_RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date) AS rnk,
    menu.product_name
FROM sales s
JOIN members m ON s.customer_id = m.customer_id
JOIN menu ON s.product_id = menu.product_id
WHERE s.order_date >= m.join_date
)
SELECT
        customer_id,
    product_name as item_name,
    order_date,
    join_date as membership_date
FROM cte
WHERE rnk = 1;

"""

print('\nFirst item purchased by the customer after they became a member : \n')
query_execution(query)
```

```
First item purchased by the customer after they became a member :

+---------------+-------------+--------------+-------------------+
| customer_id   | item_name   | order_date   | membership_date   |
|---------------+-------------+--------------+-------------------|
| A             | curry       | 2021-01-07   | 2021-01-07        |
| B             | sushi       | 2021-01-11   | 2021-01-09        |
+---------------+-------------+--------------+-------------------+
```

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

In [10]:
```
query = """

WITH cte AS (
SELECT
        s.customer_id,
    s.order_date,
    s.product_id,
    m.product_name,
    mb.join_date,
RANK() OVER(PARTITION BY s.customer_id ORDER BY mb.join_date, s.order_date DESC) AS rnk
FROM sales s
JOIN menu m ON m.product_id = s.product_id
JOIN members mb ON mb.customer_id = s.customer_id
WHERE order_date < join_date
)
SELECT
    customer_id,
    product_name as item_name,
    join_date as membership_date,
    order_date
FROM cte
WHERE rnk = 1;

"""
```

```
print('\nPurchased item just before the customer became a member : \n')
query_execution(query)
```

```
Purchased item just before the customer became a member :

+---------------+-------------+-------------------+--------------+
| customer_id   | item_name   | membership_date   | order_date   |
|---------------+-------------+-------------------+--------------|
| A             | sushi       | 2021-01-07        | 2021-01-01   |
| A             | curry       | 2021-01-07        | 2021-01-01   |
| B             | sushi       | 2021-01-09        | 2021-01-04   |
+---------------+-------------+-------------------+--------------+
```

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

In [11]:
```
query = """

WITH cte AS (
SELECT
        s.customer_id,
    s.order_date,
    s.product_id,
    mb.join_date,
    m.product_name,
    m.price
FROM sales s
JOIN menu m ON s.product_id = m.product_id
JOIN members mb ON mb.customer_id = s.customer_id
WHERE order_date < join_date
)
SELECT
        customer_id,
    count(product_name) AS total_items,
    sum(price) AS total_spent
FROM cte
GROUP BY customer_id
ORDER BY customer_id;

"""

print('\nTotal items and total amount spent by each customer before they became a member : \n')
query_execution(query)
```

```
Total items and total amount spent by each customer before they became a member :

+---------------+---------------+---------------+
| customer_id   |   total_items |   total_spent |
|---------------+---------------+---------------|
| A             |             2 |            25 |
| B             |             3 |            40 |
+---------------+---------------+---------------+
```

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

In [12]:

```
query = """

SELECT
        s.customer_id,
    SUM(CASE WHEN s.product_id = 1 THEN m.price*(2*10)
                                ELSE m.price*10 END) AS points
FROM sales s
JOIN menu m ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id;

"""

print('\nPoints of each customer : \n')
query_execution(query)
```

```
Points of each customer :

+---------------+----------+
| customer_id   |   points |
|---------------+----------|
| A             |      860 |
| B             |      940 |
| C             |      360 |
+---------------+----------+
```

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?

```
In [13]: query = """

WITH cte AS (
SELECT
        s.customer_id,
    s.order_date,
    s.product_id,
    mb.join_date,
    m.product_name,
    m.price,
    DATE_ADD(join_date, INTERVAL 6 DAY) AS last_date
FROM sales s
JOIN menu m ON s.product_id = m.product_id
JOIN members mb ON s.customer_id = mb.customer_id
)
SELECT
        customer_id,
    sum(CASE WHEN product_id = 1 THEN price*20
                                WHEN order_date>= join_date AND order_date <= last_date THEN price*20
                ELSE price*10 END) AS total_points
FROM cte
WHERE order_date <= '2021-01-31'
GROUP BY customer_id
ORDER BY customer_id;

"""

print('\nPoints A and B customer have at the end of January : \n')
query_execution(query)
```

```
Points A and B customer have at the end of January :

+---------------+----------------+
| customer_id   |   total_points |
|---------------+----------------|
| A             |           1370 |
| B             |            820 |
+---------------+----------------+
```

Bonus Questions Solutions

1. Create a basic data table that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL. Also, add an aditional field to find-out customers who have taken the membership.

In [14]:
```
query = """

SELECT
        s.customer_id,
    s.order_date,
    m.product_name,
    m.price,
    CASE
                WHEN s.order_date >= mb.join_date THEN 'Y'
        ELSE 'N'
        END AS membership
FROM sales s
LEFT JOIN members mb ON s.customer_id = mb.customer_id
JOIN menu m ON m.product_id = s.product_id
ORDER BY s.customer_id, s.order_date;

"""

print('\nBasic data table : \n')
query_execution(query)
```

```
Basic data table :

+---------------+-------------+----------------+---------+--------------+
| customer_id   | order_date  | product_name   |   price | membership   |
|---------------+-------------+----------------+---------+--------------|
| A             | 2021-01-01  | sushi          |      10 | N            |
| A             | 2021-01-01  | curry          |      15 | N            |
| A             | 2021-01-07  | curry          |      15 | Y            |
| A             | 2021-01-10  | ramen          |      12 | Y            |
| A             | 2021-01-11  | ramen          |      12 | Y            |
| A             | 2021-01-11  | ramen          |      12 | Y            |
| B             | 2021-01-01  | curry          |      15 | N            |
| B             | 2021-01-02  | curry          |      15 | N            |
| B             | 2021-01-04  | sushi          |      10 | N            |
| B             | 2021-01-11  | sushi          |      10 | Y            |
| B             | 2021-01-16  | ramen          |      12 | Y            |
| B             | 2021-02-01  | ramen          |      12 | Y            |
| C             | 2021-01-01  | ramen          |      12 | N            |
| C             | 2021-01-01  | ramen          |      12 | N            |
| C             | 2021-01-07  | ramen          |      12 | N            |
+---------------+-------------+----------------+---------+--------------+
```

2. Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program. Create a table as per Danny's requirement.

In [15]:

```
query = """

WITH cte AS (
SELECT
        s.customer_id,
    s.order_date,
    m.product_name,
    m.price,
    CASE
                WHEN s.order_date >= mb.join_date THEN 'Y'
        ELSE 'N'
        END AS membership
FROM sales s
LEFT JOIN members mb ON s.customer_id = mb.customer_id
JOIN menu m ON m.product_id = s.product_id
ORDER BY s.customer_id, s.order_date
)
SELECT *,
CASE
        WHEN membership = 'Y' THEN RANK() OVER(PARTITION BY customer_id, membership ORDER BY order_date)
    ELSE 'null'
END AS ranking
FROM cte;

"""

print('\nRanking table : \n')
query_execution(query)
```

```
Ranking table :

+---------------+-------------+----------------+---------+--------------+-----------+
| customer_id   | order_date  | product_name   |   price | membership   | ranking   |
|---------------+-------------+----------------+---------+--------------+-----------|
| A             | 2021-01-01  | sushi          |      10 | N            | null      |
| A             | 2021-01-01  | curry          |      15 | N            | null      |
| A             | 2021-01-07  | curry          |      15 | Y            | 1         |
| A             | 2021-01-10  | ramen          |      12 | Y            | 2         |
| A             | 2021-01-11  | ramen          |      12 | Y            | 3         |
| A             | 2021-01-11  | ramen          |      12 | Y            | 3         |
| B             | 2021-01-01  | curry          |      15 | N            | null      |
| B             | 2021-01-02  | curry          |      15 | N            | null      |
| B             | 2021-01-04  | sushi          |      10 | N            | null      |
| B             | 2021-01-11  | sushi          |      10 | Y            | 1         |
| B             | 2021-01-16  | ramen          |      12 | Y            | 2         |
| B             | 2021-02-01  | ramen          |      12 | Y            | 3         |
| C             | 2021-01-01  | ramen          |      12 | N            | null      |
| C             | 2021-01-01  | ramen          |      12 | N            | null      |
| C             | 2021-01-07  | ramen          |      12 | N            | null      |
+---------------+-------------+----------------+---------+--------------+-----------+
```

Thank You

- Nur Hasan