8WEEKSQLCHALLENGE.COM

CASE STUDY #1

DANNY'S DINER

THE TASTE OF SUCCESS

DATAWITHDANNY.COM

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

```
%sql sqlite://
```

```
%%sql
CREATE TABLE sales (
  "customer_id" VARCHAR(1),
  "order_date" DATE,
  "product_id" INTEGER
);

INSERT INTO sales
  ("customer_id", "order_date", "product_id")
VALUES
  ('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');
```

```
 * sqlite://
(sqlite3.OperationalError) table sales already exists
[SQL: CREATE TABLE sales (
  "customer_id" VARCHAR(1),
  "order_date" DATE,
  "product_id" INTEGER
);]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

```
%%sql
SELECT *
FROM sales
```

```
 * sqlite://
Done.
```

```
[('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)]
```

```
%%sql
CREATE TABLE menu (
  "product_id" INTEGER,
  "product_name" VARCHAR(5),
  "price" INTEGER
);

INSERT INTO menu
  ("product_id", "product_name", "price")
VALUES
  ('1', 'sushi', '10'),
  ('2', 'curry', '15'),
  ('3', 'ramen', '12');
```

```
 * sqlite://
(sqlite3.OperationalError) table menu already exists
[SQL: CREATE TABLE menu (
  "product_id" INTEGER,
  "product_name" VARCHAR(5),
  "price" INTEGER
);]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

```
%sql SELECT * FROM menu
```

```
 * sqlite://
Done.
```

```
[(1, 'sushi', 10), (2, 'curry', 15), (3, 'ramen', 12)]
```

```
%%sql
CREATE TABLE members (
  "customer_id" VARCHAR(1),
  "join_date" DATE
);

INSERT INTO members
```

```
  ("customer_id", "join_date")
VALUES
  ('A', '2021-01-07'),
  ('B', '2021-01-09');
```

```
 * sqlite://
(sqlite3.OperationalError) table members already exists
[SQL: CREATE TABLE members (
  "customer_id" VARCHAR(1),
  "join_date" DATE
);]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

```
%%sql
SELECT *
FROM members
```

```
 * sqlite://
Done.
```

```
[('A', '2021-01-07'), ('B', '2021-01-09')]
```

```
#1 What is the total amount each customer spent at the restaurant?
sql_query = """
SELECT customer_id, SUM(price)
FROM sales AS s
JOIN menu AS m
ON s.product_id = m.product_id
GROUP BY customer_id
;
"""

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+------------+
| customer_id | SUM(price) |
+-------------+------------+
|      A      |     76     |
|      B      |     74     |
|      C      |     36     |
+-------------+------------+
```

```
#2 How many days has each customer visited the restaurant?
sql_query = """
SELECT customer_id, COUNT(order_date)
FROM sales
GROUP BY customer_id
;
```

```
"""

result = %sql {sql_query}
print(result)

 * sqlite://
Done.
+-------------+-------------------+
| customer_id | COUNT(order_date) |
+-------------+-------------------+
|      A      |         6         |
|      B      |         6         |
|      C      |         3         |
+-------------+-------------------+

#3 What was the first item from the menu purchased by each customer?
sql_query = '''
WITH cte AS (
SELECT customer_id, product_name, order_date, DENSE_RANK()
OVER(PARTITION BY customer_id ORDER BY order_date) as ranking
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY customer_id, product_name)

SELECT customer_id, product_name, order_date
FROM cte
WHERE ranking = 1
;
'''

result = %sql {sql_query}
print(result)

 * sqlite://
Done.
+-------------+--------------+------------+
| customer_id | product_name | order_date |
+-------------+--------------+------------+
|      A      |    curry     | 2021-01-01 |
|      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?
sql_query = '''
SELECT product_name, COUNT(s.product_id) orders_made
FROM sales s
```

```
JOIN menu m
ON s.product_id = m.product_id
GROUP BY product_name
ORDER BY orders_made DESC
LIMIT 1
;
'''

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+--------------+-------------+
| product_name | orders_made |
+--------------+-------------+
|    ramen     |      8      |
+--------------+-------------+
```

```
#5 Which item was the most popular one for each customer?
sql_query = '''
WITH cte AS (SELECT customer_id, product_name, COUNT(s.product_id)
orders_made, DENSE_RANK() OVER(
PARTITION BY customer_id ORDER BY COUNT (s.product_id) DESC) ranking
FROM sales s
JOIN menu m
ON s.product_id = m.product_id
GROUP BY customer_id, product_name)

SELECT customer_id, product_name, orders_made
FROM cte
WHERE ranking = 1
;
'''

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+--------------+-------------+
| customer_id | product_name | orders_made |
+-------------+--------------+-------------+
|      A      |    ramen     |      3      |
|      B      |    sushi     |      2      |
|      B      |    ramen     |      2      |
|      B      |    curry     |      2      |
|      C      |    ramen     |      3      |
+-------------+--------------+-------------+
#6 Which item was purchased first by the customer after they became a
member?
sql_query = '''
WITH member_sales_cte AS
(
 SELECT s.customer_id, m.join_date, s.order_date, s.product_id,
 DENSE_RANK() OVER(PARTITION BY s.customer_id
 ORDER BY s.order_date) AS rank
 FROM sales AS s
 JOIN members AS m
 ON s.customer_id = m.customer_id
 WHERE s.order_date = m.join_date
)
SELECT s.customer_id, s.order_date, m2.product_name
FROM member_sales_cte AS s
JOIN menu AS m2
 ON s.product_id = m2.product_id
;
'''

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+------------+--------------+
| customer_id | order_date | product_name |
+-------------+------------+--------------+
|      A      | 2021-01-07 |    curry     |
+-------------+------------+--------------+
```

```
#7 Which item was purchased right before the customer became a member?
sql_query = '''WITH member_sales_cte AS
(
 SELECT s.customer_id, m.join_date, s.order_date, s.product_id,
 DENSE_RANK() OVER(PARTITION BY s.customer_id
 ORDER BY s.order_date) AS rank
 FROM sales AS s
 JOIN members AS m
 ON s.customer_id = m.customer_id
 WHERE s.order_date < m.join_date
)
SELECT s.customer_id, s.order_date, m2.product_name
FROM member_sales_cte AS s
JOIN menu AS m2
 ON s.product_id = m2.product_id
;
'''
```

```
result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+------------+--------------+
| customer_id | order_date | product_name |
+-------------+------------+--------------+
|      A      | 2021-01-01 |    sushi     |
|      B      | 2021-01-04 |    sushi     |
|      A      | 2021-01-01 |    curry     |
|      B      | 2021-01-01 |    curry     |
|      B      | 2021-01-02 |    curry     |
+-------------+------------+--------------+
```

```
#8. What is the total number of items and amount spent for each member
before they became a member?
sql_query = '''
SELECT s.customer_id, product_name, COUNT(DISTINCT s.product_id)
orders_made, SUM(m.price) total_spent
FROM sales s
JOIN menu m
ON s.product_id = m.product_id

JOIN members m2
ON s.customer_id = m2.customer_id

WHERE s.order_date < m2.join_date
GROUP BY s.customer_id
;
'''

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+--------------+-------------+-------------+
| customer_id | product_name | orders_made | total_spent |
+-------------+--------------+-------------+-------------+
|      A      |    sushi     |      2      |      25     |
|      B      |    curry     |      2      |      40     |
+-------------+--------------+-------------+-------------+
```

```
#9 If each customers’ $1 spent equates to 10 points and sushi has a 2x
points multiplier, how many points would each customer have?
sql_query = '''
WITH point_cte AS (
SELECT *,
CASE WHEN product_name = 'sushi' THEN price * 20
ELSE price * 10
END AS point
FROM sales
JOIN menu
ON sales. product_id = menu.product_id
)

SELECT customer_id, SUM(point)
FROM point_cte
GROUP BY customer_id
;
'''

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+------------+
| customer_id | SUM(point) |
+-------------+------------+
|      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 Jan21?
sql_query = '''
WITH dates_cte AS ( SELECT *,
DATE(join_date, '+6 day') valid_date,
DATE('2021-01-31') AS last_date
FROM members AS m ),

points_cte AS ( SELECT d.customer_id, s.order_date, d.join_date,
d.valid_date,
d.last_date, m.product_name, m.price,
CASE  WHEN m.product_name = 'sushi' THEN 20 * m.price
WHEN s.order_date BETWEEN d.join_date AND d.valid_date THEN 20 *
m.price
ELSE 10 * m.price  END AS points

FROM dates_cte AS d
JOIN sales AS s
ON d.customer_id = s.customer_id
JOIN menu AS m
ON s.product_id = m.product_id
WHERE s.order_date < d.last_date)

SELECT customer_id, SUM(points) AS total_points
FROM points_cte
GROUP BY customer_id;
'''

result = %sql {sql_query}
print(result)
```

```
 * sqlite://
Done.
+-------------+--------------+
| customer_id | total_points |
+-------------+--------------+
|      A      |     1370     |
|      B      |     820      |
+-------------+--------------+
```