PostgreSQL Integration with Jupyter Notebook

In [1]:
```
%load_ext sql
from sqlalchemy import create_engine
```

In [2]:
```
import os
print('Get current working directory : ', os.getcwd())
```

```
Get current working directory :  C:\Users[redacted]\Documents
```

Connecting to Postgres Database

In [5]:
```
%sql postgresql://postgres[redacted]@localhost/postgres
```

In [6]:
```
engine = create_engine('postgresql://postgres:[redacted]@localhost/postgres')
```

Writing SQL Commands in Jupyter Notebook

In [10]:
```
%%sql
-- 1. What is the total amount each customer spent at the restaurant?
SELECT
s.customer_id, SUM(price)
FROM sales s
INNER JOIN menu m ON s.product_id = m.product_id
GROUP BY s.customer_id
ORDER BY s.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
3 rows affected.
```

Out[10]:

customer_id	sum
A	76
B	74
C	36

In [11]:
```
%%sql
--2. How many days has each customer visited the restaurant?
SELECT
  customer_id
, COUNT(order_date)
FROM sales
GROUP BY customer_id
ORDER BY customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
3 rows affected.
```

Out[11]:

customer_id	count
A	6
B	6
C	3

In [21]:
```
%%sql
-- 3.What was the first item from the menu purchased by each customer?
SELECT
  x.customer_id
, menu.product_name
FROM (
        SELECT
          customer_id
        , product_id
        , DENSE_RANK() OVER (PARTITION BY customer_id ORDER BY order_date) as rnum
        FROM sales
) x
INNER JOIN menu ON menu.product_id = x.product_id
WHERE rnum = 1
ORDER BY x.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
5 rows affected.
```

Out[21]:

customer_id	product_name
A	sushi
A	curry
B	curry
C	ramen
C	ramen

In [12]:
```
%%sql
-- 4.What is the most purchased item on the menu and how many times was it purchase

SELECT
  product_id
, COUNT(product_id) as number_of_times_ordered
FROM sales
GROUP BY product_id
ORDER BY COUNT(product_id) DESC
LIMIT 1
```

```
 * postgresql://postgres:***@localhost/postgres
1 rows affected.
```

Out[12]:

product_id	number_of_times_ordered
3	8

In [14]:
```
%%sql
-- 5.Which item was the most popular for each customer?
```

```
SELECT
  customer_id
, product_name
FROM
(
        SELECT
          customer_id
        , product_id
        , DENSE_RANK() OVER (PARTITION BY customer_id ORDER BY COUNT(product_id) DE
        FROM sales
        GROUP BY product_id, customer_id
)x
JOIN menu ON menu.product_id = x.product_id
WHERE x.rnum = 1
ORDER BY x.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
5 rows affected.
```

Out[14]:

customer_id	product_name
A	ramen
B	sushi
B	curry
B	ramen
C	ramen

In [22]:

```
%%sql
--6. Which item was purchased first by the customer after they became a member?
SELECT
  x.customer_id
, menu.product_name
FROM (
        SELECT
          sales.customer_id
        , sales.product_id
        , sales.order_date
        , DENSE_RANK() OVER (PARTITION BY sales.customer_id ORDER BY sales.order_da
        FROM sales
        INNER JOIN members ON members.customer_id = sales.customer_id
        WHERE sales.order_date >= members.join_date
) x
INNER JOIN menu ON menu.product_id = x.product_id
WHERE rnum = 1
ORDER BY x.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
2 rows affected.
```

Out[22]:

customer_id	product_name
A	curry
B	sushi

In [23]:
```
%%sql
--7.Which item was purchased just before the customer became a member? max(order_da
SELECT
  x.customer_id
, menu.product_name
FROM (
        SELECT
          sales.customer_id
        , sales.product_id
        , sales.order_date
        , DENSE_RANK() OVER (PARTITION BY sales.customer_id ORDER BY sales.order_da
        FROM sales
        INNER JOIN members ON members.customer_id = sales.customer_id
        WHERE sales.order_date < members.join_date
)x
INNER JOIN menu ON menu.product_id = x.product_id
WHERE rnum = 1
ORDER BY customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
3 rows affected.
```

Out[23]:

customer_id	product_name
A	sushi
A	curry
B	sushi

In [24]:
```
%%sql
--8.What is the total items and amount spent for each member before they became a m
SELECT
s.customer_id, COUNT(s.product_id),SUM(price)
FROM sales s
INNER JOIN menu m ON s.product_id = m.product_id
INNER JOIN members mem ON s.customer_id = mem.customer_id
WHERE s.order_date NOT IN (join_date, CURRENT_DATE)
GROUP BY s.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
2 rows affected.
```

Out[24]:

customer_id	count	sum
A	5	61
B	6	74

In [20]:
```
%%sql
-- 9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier -
SELECT
  sales.customer_id
, SUM(CASE
                WHEN menu.product_name = 'sushi' THEN menu.price*20
                ELSE menu.price *10
          END) AS customer_wise_product_wise_total_points
FROM sales
```

```
INNER JOIN menu ON sales.product_id = menu.product_id
GROUP BY sales.customer_id
ORDER BY sales.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
3 rows affected.
```

Out[20]:

customer_id	customer_wise_product_wise_total_points
A	860
B	940
C	360

In [19]:
```
%%sql
-- 10.In the first week after a customer joins the program (including their join da
SELECT
  sales.customer_id
, SUM(CASE
                WHEN sales.order_date BETWEEN members.join_date AND members.join_da
                WHEN menu.product_name = 'sushi' THEN menu.price*20
                ELSE menu.price *10
          END) AS customer_wise_product_wise_total_points
FROM sales
INNER JOIN menu ON sales.product_id = menu.product_id
INNER JOIN members ON sales.customer_id = members.customer_id
WHERE sales.order_date <= '2021-01-31'
GROUP BY sales.customer_id
ORDER BY sales.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
2 rows affected.
```

Out[19]:

customer_id	customer_wise_product_wise_total_points
A	1370
B	940

Bonus Questions

In [26]:
```
%%sql
-- Recreate table with customer_id, order_date, product_name, price, and whether cu
SELECT
  sales.customer_id
, sales.order_date
, menu.product_name
, menu.price
, CASE
        WHEN sales.order_date>= members.join_date THEN 'Y'
        ELSE 'N'
  END AS member
FROM sales
INNER JOIN menu ON sales.product_id =menu.product_id
LEFT JOIN members ON sales.customer_id = members.customer_id
```

```
 * postgresql://postgres:***@localhost/postgres
15 rows affected.
```

Out[26]:

customer_id	order_date	product_name	price	member
A	2021-01-07	curry	15	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-10	ramen	12	Y
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
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