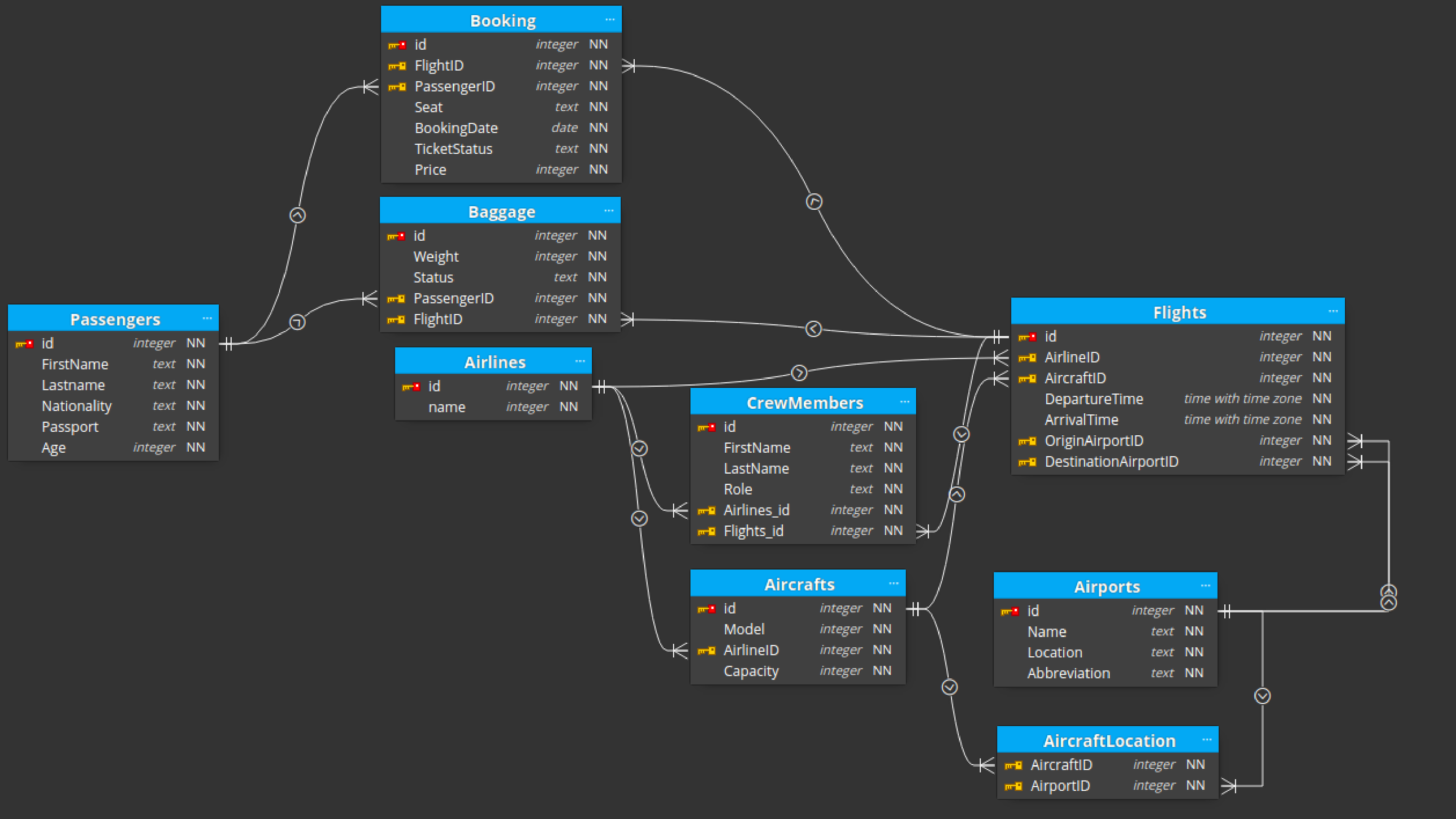
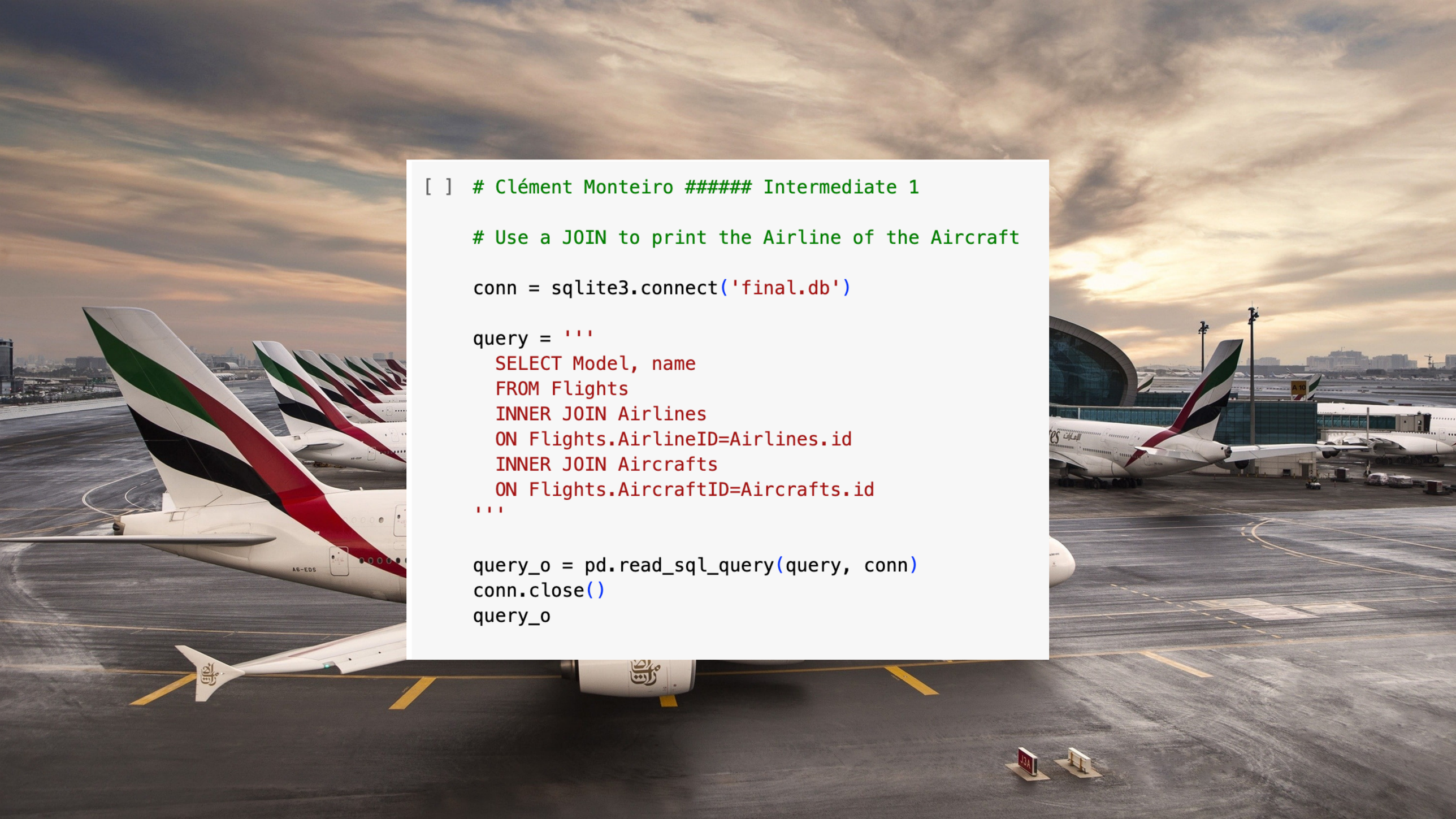
Database Managment

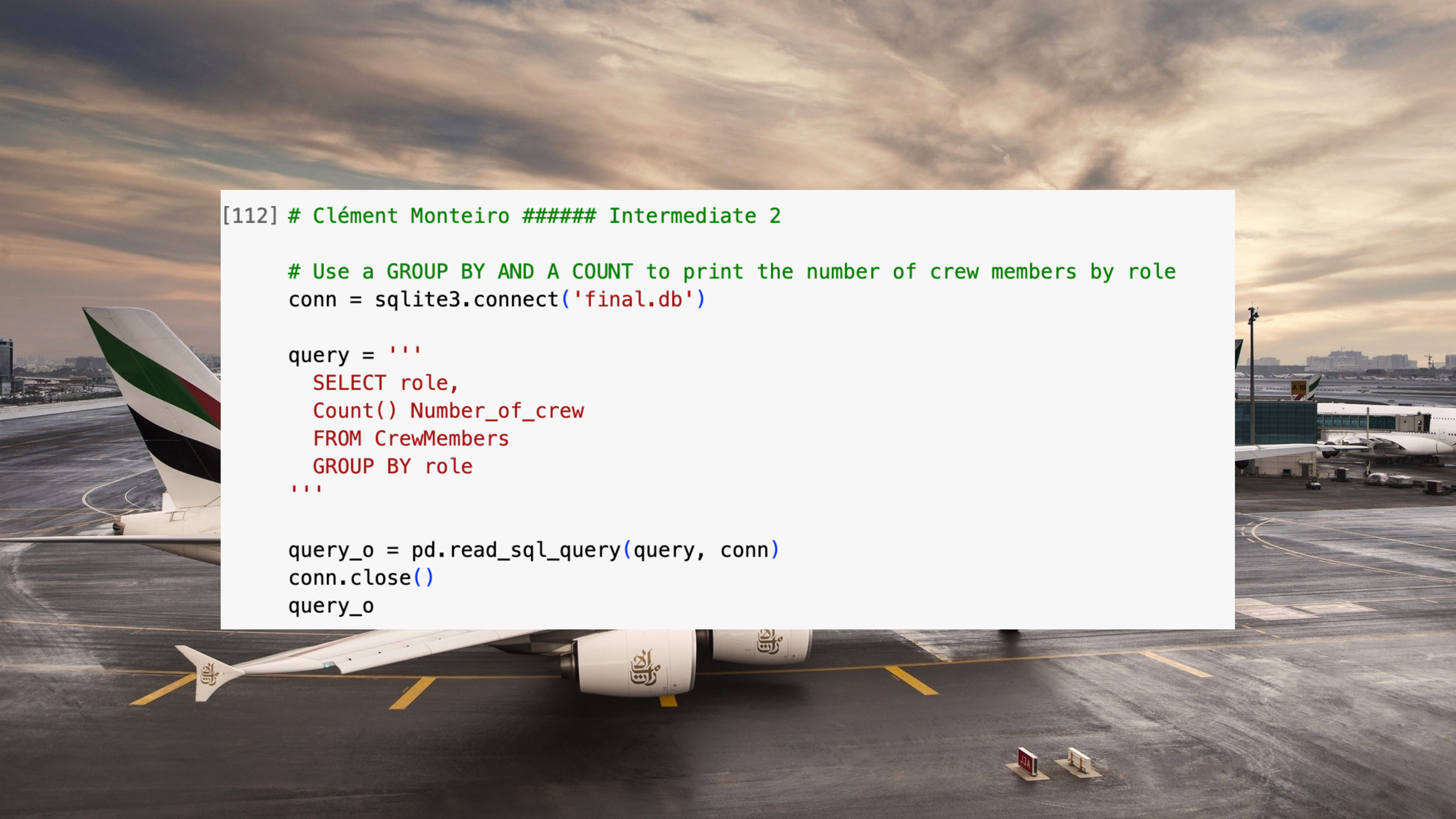
Clement Monteiro, Mahe Faure,
Antoine Van Gorp, Cyprien Singez


Booking
id integer NN
FlightID integer NN
PassengerID integer NN
Seat text NN
BookingDate date NN
TicketStatus text NN
Price integer NN
Baggage
id integer NN
Weight integer NN
Status text NN
PassengerID integer NN
FlightID integer NN
Passengers
id integer NN
FirstName text NN
Lastname text NN
Nationality text NN
Passport text NN
Age integer NN
Airlines
id integer NN
name integer NN
CrewMembers
id integer NN
FirstName text NN
LastName text NN
Role text NN
Airlines_id integer NN
Flights_id integer NN
Flights
id integer NN
AirlineID integer NN
AircraftID integer NN
DepartureTime time with time zone NN
ArrivalTime time with time zone NN
OriginAirportID integer NN
DestinationAirportID integer NN
Aircrafts
id integer NN
Model integer NN
AirlineID integer NN
Capacity integer NN
Airports
id integer NN
Name text NN
Location text NN
Abbreviation text NN
AircraftLocation
AircraftID integer NN
AirportID integer NN

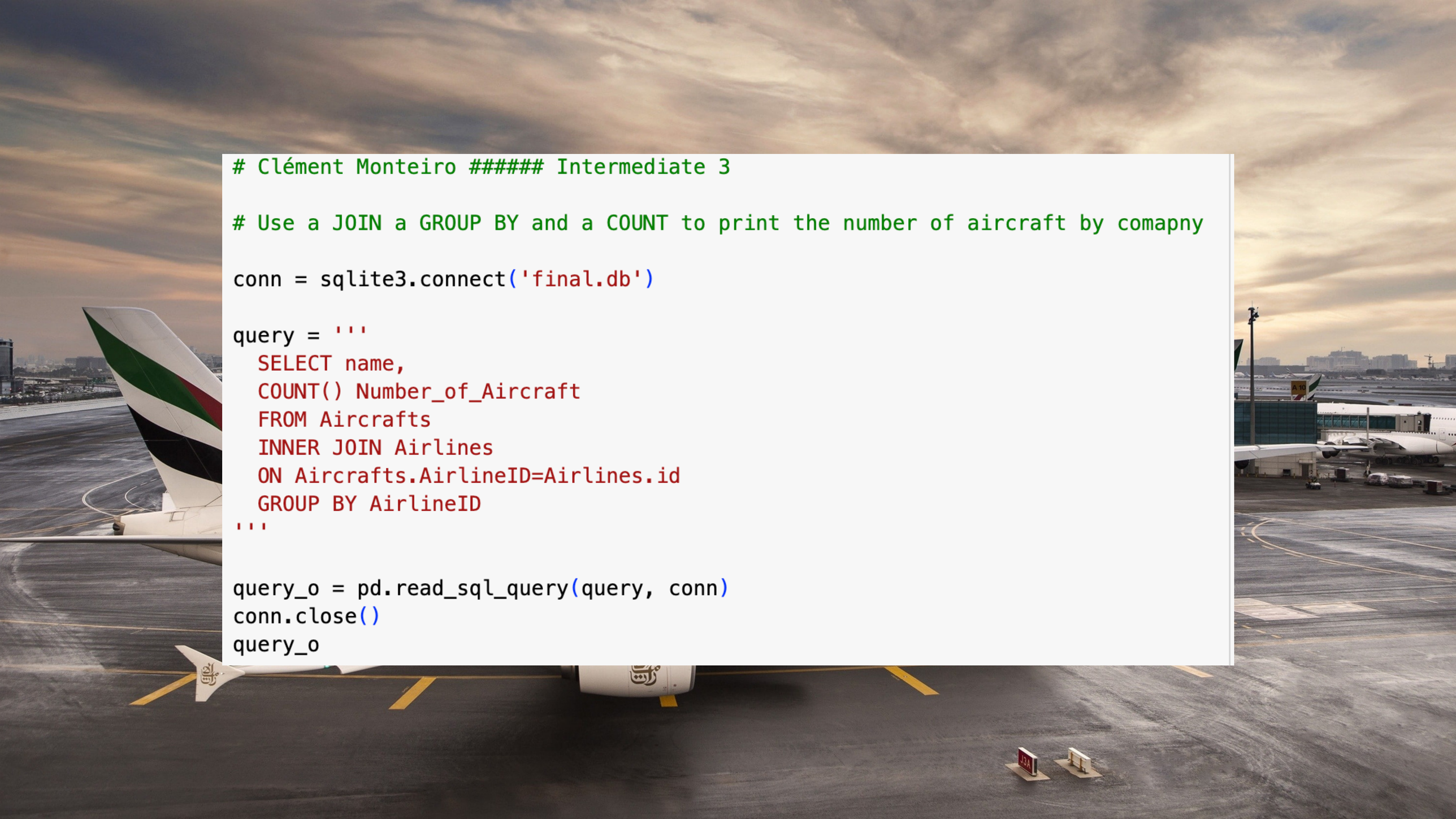
Clément Monteiro



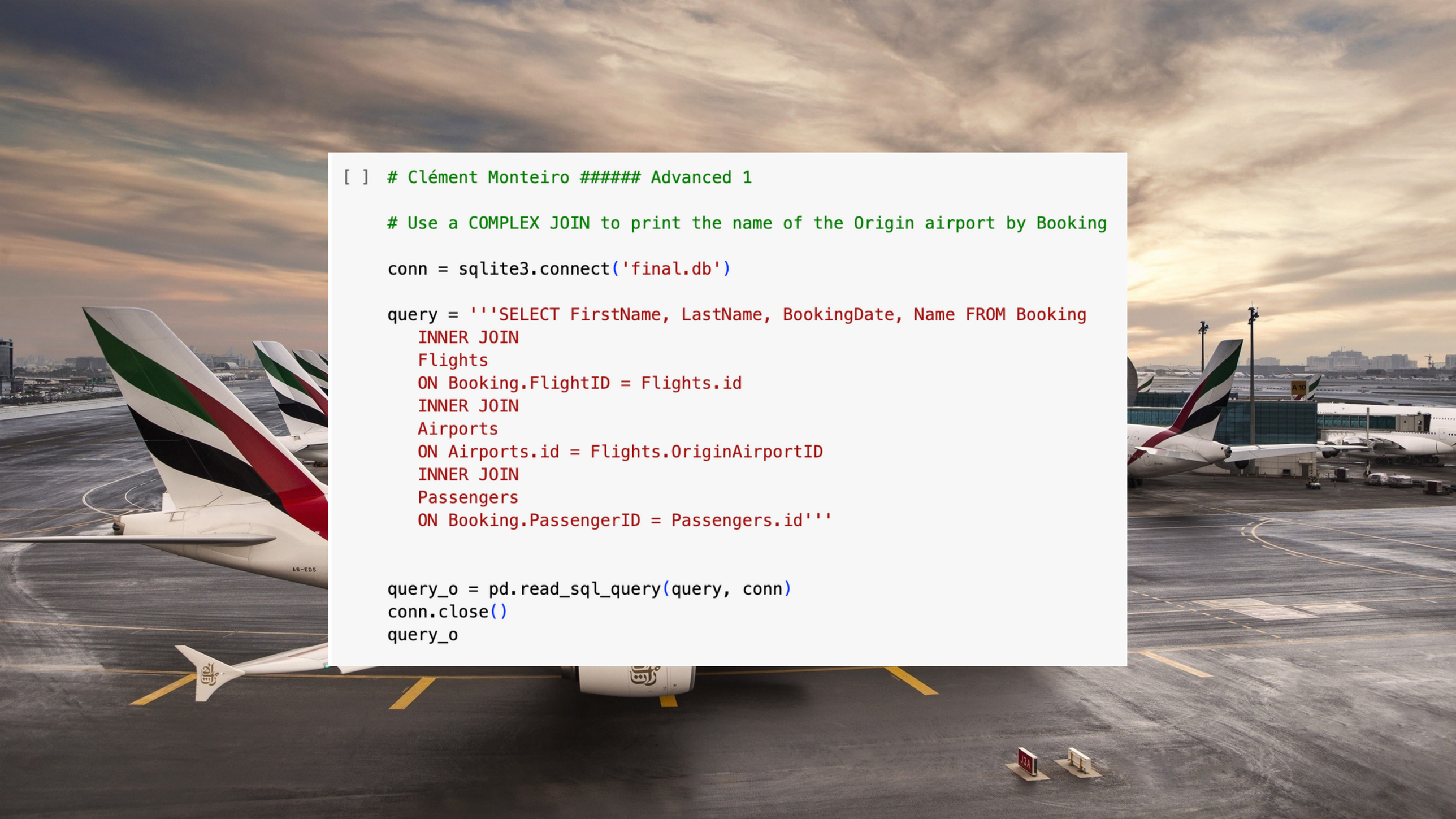
```
[ ]  # Clément Monteiro ###### Intermediate 1

     # Use a JOIN to print the Airline of the Aircraft

     conn = sqlite3.connect('final.db')

     query = '''
       SELECT Model, name
       FROM Flights
       INNER JOIN Airlines
       ON Flights.AirlineID=Airlines.id
       INNER JOIN Aircrafts
       ON Flights.AircraftID=Aircrafts.id
     '''

     query_o = pd.read_sql_query(query, conn)
     conn.close()
     query_o
```



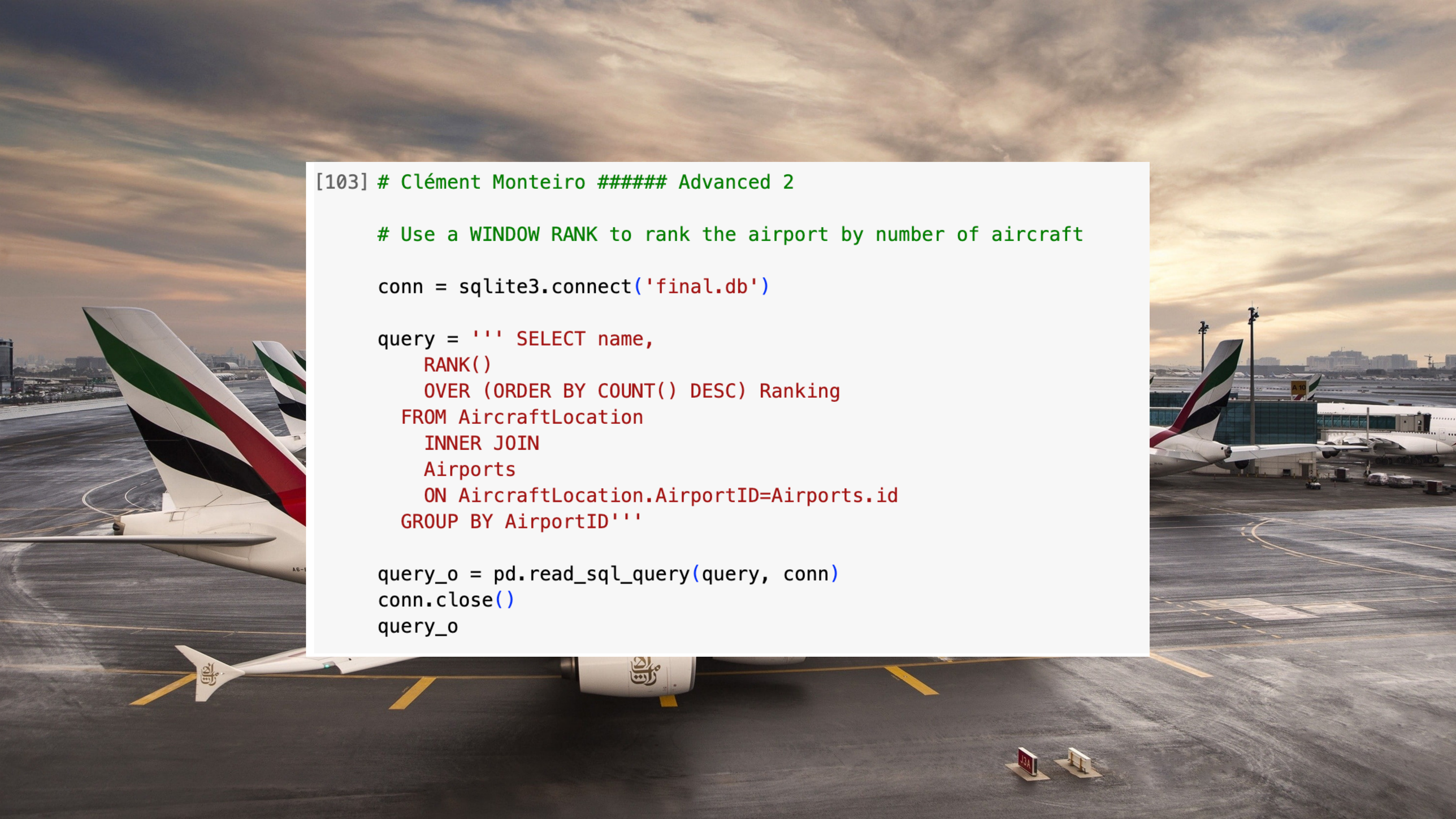
```
[112] # Clément Monteiro ###### Intermediate 2

      # Use a GROUP BY AND A COUNT to print the number of crew members by role
      conn = sqlite3.connect('final.db')

      query = '''
        SELECT role,
        Count() Number_of_crew
        FROM CrewMembers
        GROUP BY role
      '''

      query_o = pd.read_sql_query(query, conn)
      conn.close()
      query_o
```



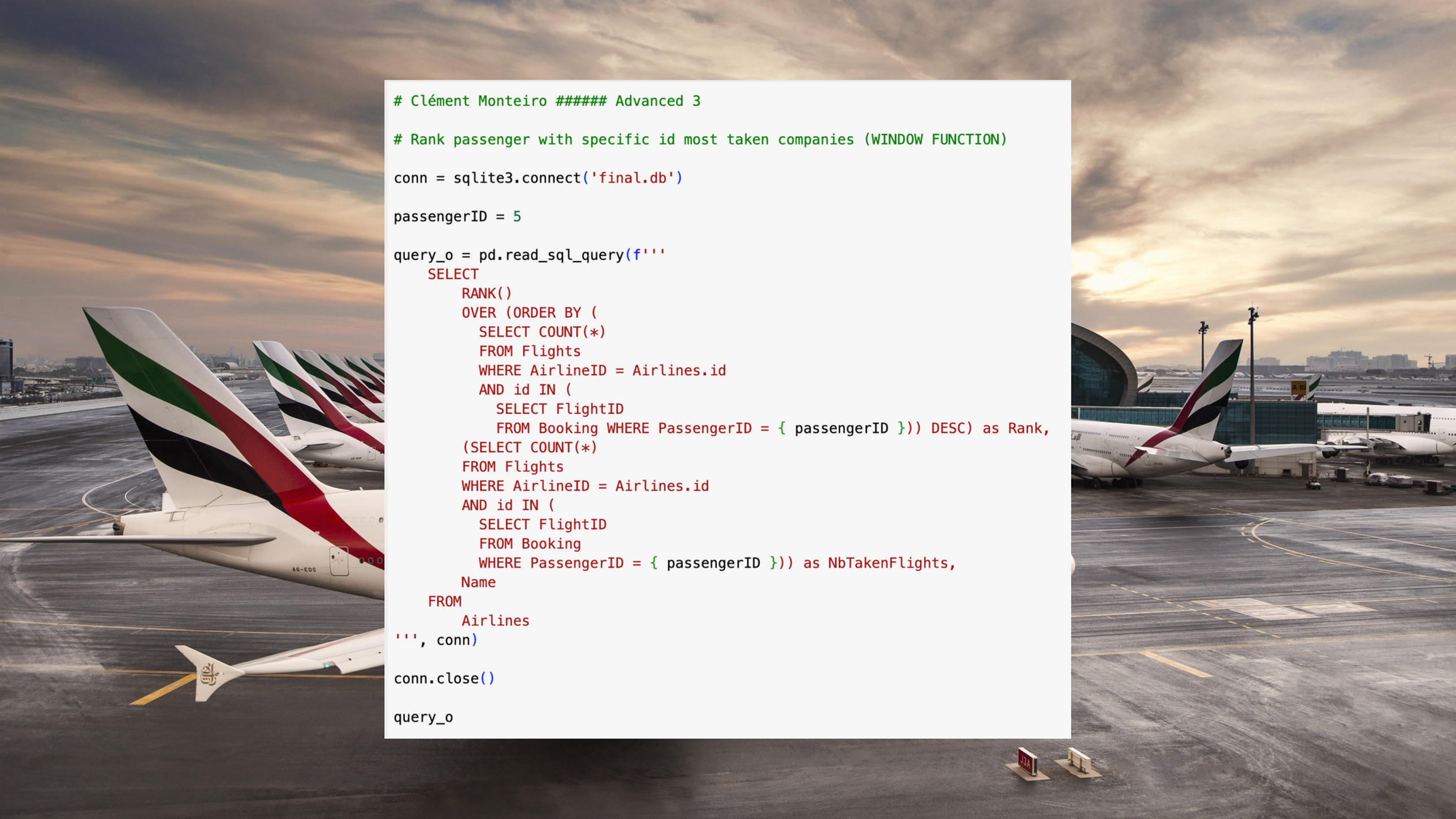
```
# Clément Monteiro ###### Intermediate 3

# Use a JOIN a GROUP BY and a COUNT to print the number of aircraft by comapny

conn = sqlite3.connect('final.db')

query = '''
  SELECT name,
  COUNT() Number_of_Aircraft
  FROM Aircrafts
  INNER JOIN Airlines
  ON Aircrafts.AirlineID=Airlines.id
  GROUP BY AirlineID
'''

query_o = pd.read_sql_query(query, conn)
conn.close()
query_o
```



```
[ ]  # Clément Monteiro ###### Advanced 1

     # Use a COMPLEX JOIN to print the name of the Origin airport by Booking

     conn = sqlite3.connect('final.db')

     query = '''SELECT FirstName, LastName, BookingDate, Name FROM Booking
        INNER JOIN
        Flights
        ON Booking.FlightID = Flights.id
        INNER JOIN
        Airports
        ON Airports.id = Flights.OriginAirportID
        INNER JOIN
        Passengers
        ON Booking.PassengerID = Passengers.id'''


     query_o = pd.read_sql_query(query, conn)
     conn.close()
     query_o
```



```
[103] # Clément Monteiro ###### Advanced 2

      # Use a WINDOW RANK to rank the airport by number of aircraft

      conn = sqlite3.connect('final.db')

      query = ''' SELECT name,
          RANK()
          OVER (ORDER BY COUNT() DESC) Ranking
        FROM AircraftLocation
          INNER JOIN
          Airports
          ON AircraftLocation.AirportID=Airports.id
        GROUP BY AirportID'''

      query_o = pd.read_sql_query(query, conn)
      conn.close()
      query_o
```



```
# Clément Monteiro ###### Advanced 3

# Rank passenger with specific id most taken companies (WINDOW FUNCTION)

conn = sqlite3.connect('final.db')

passengerID = 5

query_o = pd.read_sql_query(f'''
    SELECT
        RANK()
        OVER (ORDER BY (
          SELECT COUNT(*)
          FROM Flights
          WHERE AirlineID = Airlines.id
          AND id IN (
            SELECT FlightID
            FROM Booking WHERE PassengerID = { passengerID })) DESC) as Rank,
        (SELECT COUNT(*)
        FROM Flights
        WHERE AirlineID = Airlines.id
        AND id IN (
          SELECT FlightID
          FROM Booking
          WHERE PassengerID = { passengerID })) as NbTakenFlights,
        Name
    FROM
        Airlines
''', conn)

conn.close()

query_o
```


Cyprien SINGEZ
الإمارات
Emirates
www.emirates.com
EXPO 2020
DUBAI
UAE
A6-EDS

```python
# Cyprien SINGEZ

# Intermediate °1
# Select all distinct different airlines that the passenger with specific ID has flown with (SUBQUERY)

import pandas as pd

conn = sqlite3.connect('final.db')

passengerID = 5

df = pd.read_sql_query(f'''
    SELECT
        Name
    FROM
        Airlines
    WHERE
        id IN (SELECT AirlineID FROM Flights WHERE id IN (SELECT FlightID FROM Booking WHERE PassengerID
= { passengerID }))
    ORDER BY Name ASC
''', conn)

conn.close()

df
```

```python
# Cyprien SINGEZ

# Intermediate °2
# Print the number of flights for each company (JOIN)

import pandas as pd

conn = sqlite3.connect('final.db')

df = pd.read_sql_query('''
    SELECT
        Name,
        NbFlights
    FROM
        Airlines
    INNER JOIN
        (SELECT
            AirlineID,
            COUNT(*) as NbFlights
        FROM
            Flights
        GROUP BY
            AirlineID) as FlightsPerAirline
        ON Airlines.id = FlightsPerAirline.AirlineID
    ORDER BY
        NbFlights DESC
''', conn)

conn.close()

df
```

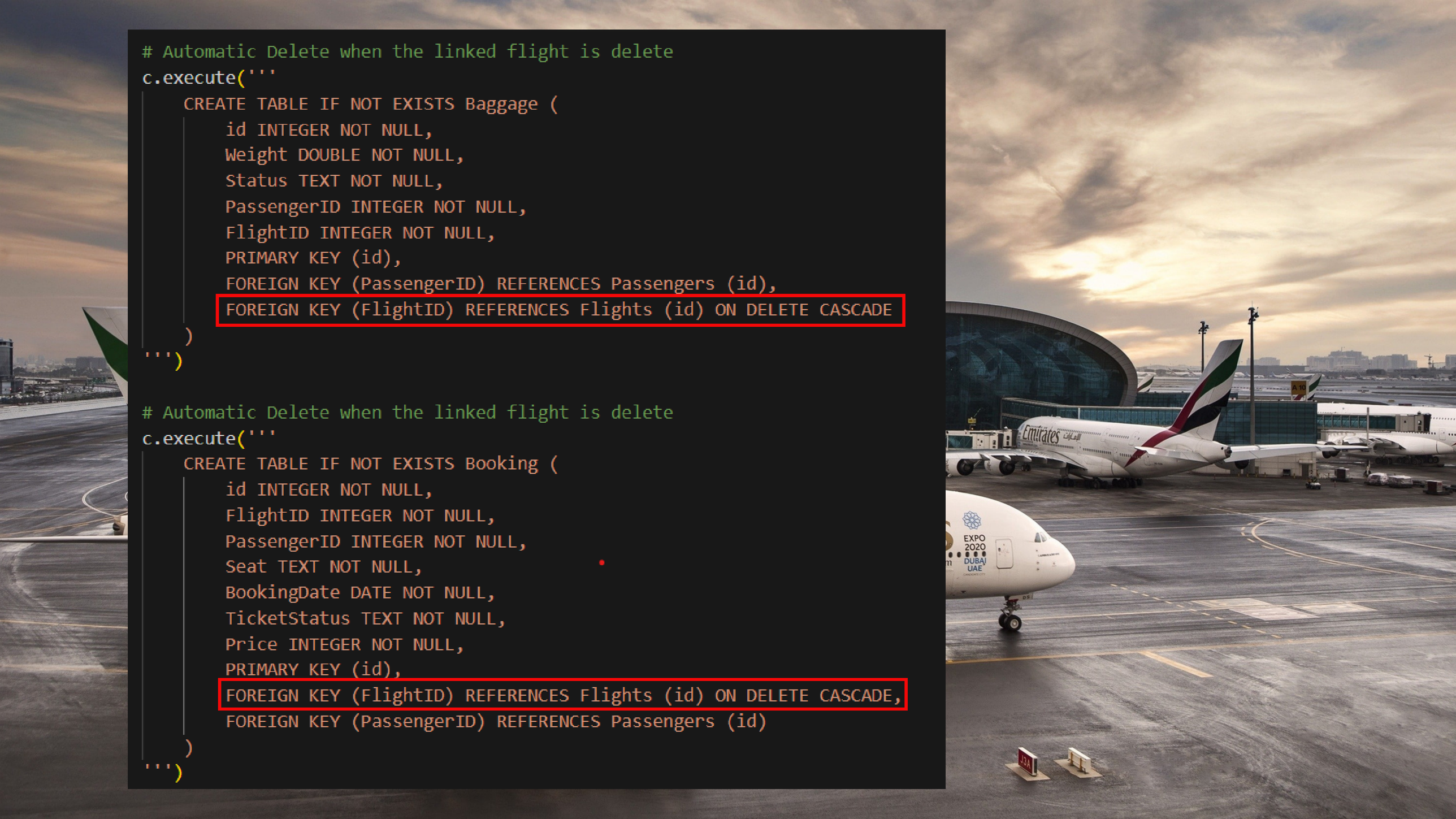
```python
# Cyprien SINGEZ

# Intermediate °3
# Print the total amount of money spent per each passenger in descending order (SUM + SUBQUERY)

import pandas as pd

conn = sqlite3.connect('final.db')

df = pd.read_sql_query('''
    SELECT
        (SELECT FirstName FROM Passengers WHERE id = PassengerID) as FirstName,
        (SELECT LastName FROM Passengers WHERE id = PassengerID) as FirstName,
        SUM(Price) as TotalMoneySpent
    FROM
        Booking
    GROUP BY
        PassengerID
    ORDER BY
        SUM(Price) DESC
''', conn)

conn.close()

df
```

```python
# Cyprien SINGEZ

# Advanced °1
# Print all the adult passengers (+18) and order them by number of flights taken (CTE)

import pandas as pd

conn = sqlite3.connect('final.db')

df = pd.read_sql_query('''
    WITH cte AS (
        SELECT
            id,
            FirstName,
            LastName,
            Nationality
        FROM
            Passengers
        WHERE
            Age >= 18
    )
    SELECT
        cte.FirstName,
        cte.LastName,
        cte.Nationality,
        COUNT(*) AS TotalFlights
    FROM
        cte
    JOIN
        Booking ON cte.id = Booking.PassengerID
    GROUP BY
        cte.id, cte.FirstName, cte.LastName, cte.Nationality
    ORDER BY
        TotalFlights DESC;
''', conn)

conn.close()

df
```

```python
# Cyprien SINGEZ

# Advanced °2
# Print the final destination of a bagage (COMPLEX JOINS)

import pandas as pd

conn = sqlite3.connect('final.db')

df = pd.read_sql_query('''
    SELECT
        FirstName,
        LastName,
        Weight,
        Name Destination_Airport
    FROM Baggage
    INNER JOIN
        Flights
            ON Baggage.FlightID = Flights.id
    INNER JOIN
        Airports
            ON Airports.id =
Flights.DestinationAirportID
        Passengers
            ON Baggage.PassengerID = Passengers.id
''', conn)

conn.close()

df
```

```
# Cyprien SINGEZ

# Advanced °3
# Rank companies by number of aircrafts (WINDOW FUNCTION)

import pandas as pd

conn = sqlite3.connect('final.db')

df = pd.read_sql_query('''
    SELECT
        RANK() OVER (ORDER BY (SELECT COUNT(*) FROM Aircrafts WHERE AirlineID = Airlines.id) DESC) as
Rank,
        (SELECT COUNT(*) FROM Aircrafts WHERE AirlineID = Airlines.id) as NbAircraft,
        Name
    FROM Airlines
''', conn)

conn.close()

df
```

Mahe Faure



```
# Automatic Delete when the linked flight is delete
c.execute('''
    CREATE TABLE IF NOT EXISTS Baggage (
        id INTEGER NOT NULL,
        Weight DOUBLE NOT NULL,
        Status TEXT NOT NULL,
        PassengerID INTEGER NOT NULL,
        FlightID INTEGER NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (PassengerID) REFERENCES Passengers (id),
        FOREIGN KEY (FlightID) REFERENCES Flights (id) ON DELETE CASCADE
    )
''')

# Automatic Delete when the linked flight is delete
c.execute('''
    CREATE TABLE IF NOT EXISTS Booking (
        id INTEGER NOT NULL,
        FlightID INTEGER NOT NULL,
        PassengerID INTEGER NOT NULL,
        Seat TEXT NOT NULL,
        BookingDate DATE NOT NULL,
        TicketStatus TEXT NOT NULL,
        Price INTEGER NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (FlightID) REFERENCES Flights (id) ON DELETE CASCADE,
        FOREIGN KEY (PassengerID) REFERENCES Passengers (id)
    )
''')
```

```
#by Mahe

# Connect to the SQLite database
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL commands
cursor = conn.cursor()

# Rank flights by price using WINDOW FUNCTION RANK
ranked_flight_query = """
    SELECT
        FlightID,
        Price,
        RANK() OVER (ORDER BY price ASC) as flight_rank
    FROM
        Booking
"""

ranked_flight_df = pd.read_sql_query(ranked_flight_query, conn)

# Close the database connection
conn.close()
# Display the result
ranked_flight_df
```

	FlightID	Price	flight_rank
0	860	600	1
1	423	600	1
2	756	600	1
3	820	600	1
4	872	600	1
...	...	...	...
99995	918	1000	99745
99996	229	1000	99745
99997	715	1000	99745
99998	987	1000	99745
99999	22	1000	99745

100000 rows × 3 columns

```python
#by Mahe

# Connect to the SQLite database
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL commands
cursor = conn.cursor()

# Calculate the average price for each company using the ranked flights
average_company_query = """
    SELECT
        c.id,
        c.name,
        AVG(b_ranked.Price) as average_price
    FROM
        Airlines c
    JOIN
        Flights f ON c.id = f.AirlineID
    JOIN
        (SELECT FlightId, Price FROM Booking ORDER BY Price ASC) b_ranked ON f.id = b_ranked.FlightId
    GROUP BY
        c.id;
"""

average_company_df = pd.read_sql_query(average_company_query, conn)

# Close the database connection
conn.close()
# Display the result
average_company_df
```

	id	name	average_price
0	1	Malone, Brooks and Mcpherson	801.494374
1	2	Morrow, Brown and Hughes	799.829147
2	3	Holt, Jefferson and Proctor	799.577068
3	4	Sharp-Jackson	801.563722
4	5	Swanson-Anderson	801.618280
5	6	Ortiz-Byrd	797.656537
6	7	Palmer-Thompson	799.811920
7	8	Mitchell, Lynch and Smith	801.579347
8	9	Frost-Foster	802.448643
9	10	Wilson LLC	798.107732
10	11	Sharp, Freeman and Hogan	801.844171
11	12	Williams-Parker	799.275579
12	13	Welch-Nolan	800.639111
13	14	Navarro, Fox and Reynolds	801.016243
14	15	Eaton-Padilla	800.526804

```python
#by Mahe

# Connect to the SQLite database
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL commands
cursor = conn.cursor()

# Calculate the average price for each company using the ranked flights
baggage_count_query = """
    SELECT
        b.FlightId,
        COUNT(b.id) as baggage_count
    FROM
        Baggage b
    GROUP BY
        b.FlightId
"""

baggage_count_df = pd.read_sql_query(baggage_count_query, conn)


# Close the database connection
conn.close()
# Display the result
baggage_count_df
```

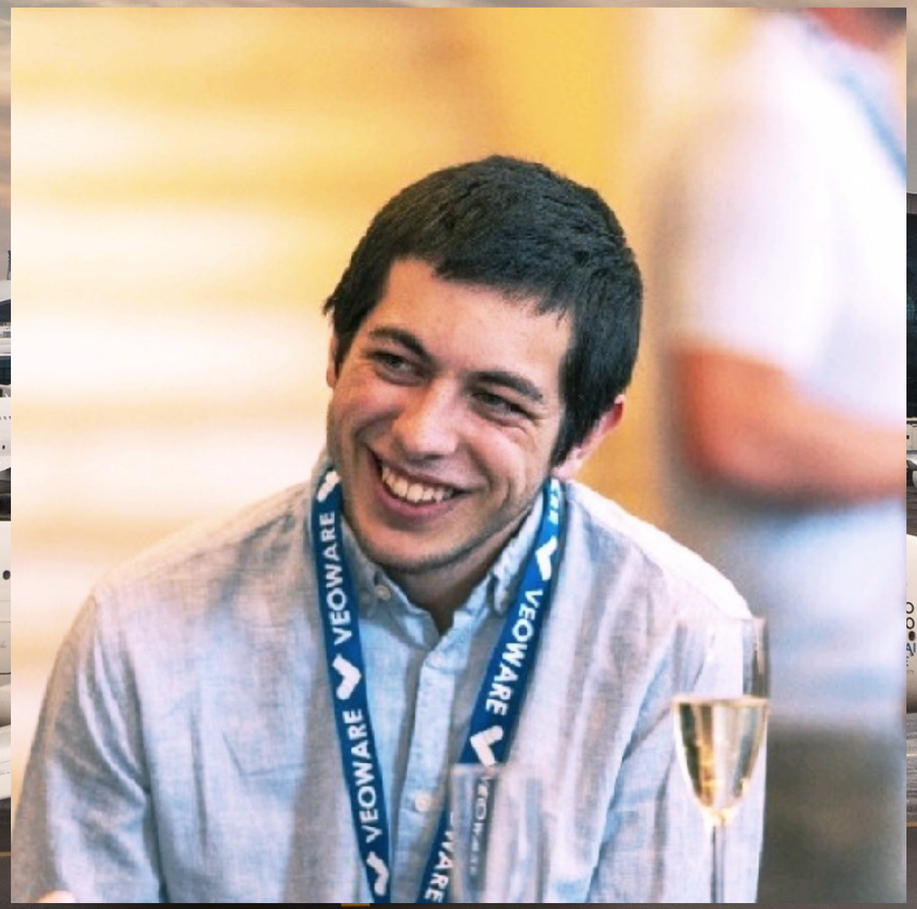
	FlightID	baggage_count
0	1	94
1	2	93
2	3	106
3	4	110
4	5	106
...	...	...
995	996	101
996	997	85
997	998	93
998	999	81
999	1000	104

1000 rows × 2 columns

```sql
# SQL query to retrieve flight information, including average price, baggage count, depa
flight_info_query = """
    SELECT
        f.id,
        dep_airport.name as departure_airport,
        arr_airport.name as arrival_airport,
        f.DepartureTime,
        f.ArrivalTime,
        c.name,
        AVG(b.Price) as average_price,
        COUNT(DISTINCT bg.id) as baggage_count,
        RANK() OVER (ORDER BY AVG(b.price) ASC) as flight_rank
    FROM
        Flights f
    JOIN
        Airlines c ON f.AirlineID = c.id
    JOIN
        Airports dep_airport ON f.OriginAirportID = dep_airport.id
    JOIN
        Airports arr_airport ON f.DestinationAirportID = arr_airport.id
    LEFT JOIN
        Booking b ON f.id = b.FlightId
    LEFT JOIN
        Baggage bg ON f.id = bg.FlightId
    GROUP BY
        f.id
"""

# Use pandas to execute the SQL query and read the result into a DataFrame
flight_info_df = pd.read_sql_query(flight_info_query, conn)

# Close the database connection
conn.close()

# Display the result (the flight information DataFrame)
flight_info_df
```

	id	departure_airport	arrival_airport	DepartureTime	ArrivalTime	name	average_price	baggage_count	flight_rank
0	791	Valeriefort	Lake Levi	2023-05-06 20:30:39	2024-05-13 15:21:31	Wilson LLC	761.849462	99	1
1	845	Port Susanville	Port Susanville	2024-01-27 10:44:36	2024-06-29 20:02:54	Mitchell, Lynch and Smith	763.257732	115	2
2	308	South Rebekahmouth	Lake Levi	2024-11-23 22:26:34	2023-08-07 01:05:43	Palmer-Thompson	765.670000	92	3
3	470	South Rebekahmouth	South Ginatown	2024-03-12 16:26:47	2023-09-07 13:36:38	Malone, Brooks and Mcpherson	766.263158	90	4
4	894	Lake Levi	North Danielleton	2022-12-12 12:08:33	2023-01-08 01:37:04	Holt, Jefferson and Proctor	766.337500	114	5
...	...	...	...	...	...	...	...	...	...
995	200	Valeriefort	Lake Levi	2024-08-09 09:08:15	2023-01-10 22:01:38	Morrow, Brown and Hughes	831.927711	137	996
996	636	Lake Amanda	Benderport	2024-09-10 15:16:15	2023-05-22 17:54:06	Swanson-Anderson	832.822917	106	997
997	174	Chapmanfort	Benderport	2023-02-02 14:11:40	2023-11-09 01:51:16	Holt, Jefferson and Proctor	832.838095	108	998
998	90	Valeriefort	South Rebekahmouth	2024-11-23 15:06:53	2023-08-02 08:23:20	Navarro, Fox and Reynolds	834.233645	93	999
999	487	Benderport	Port Susanville	2024-05-09 17:00:01	2024-08-14 02:50:21	Sharp-Jackson	836.822917	120	1000

Antoine Van Gorp



```python
# Done by Antoine Van Gorp
# Connect to the SQLite database named 'final.db'
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Define the SQL query to retrieve information about aircraft and their correspond
aircraft_airport = """
    SELECT
        Airports.name,
        Aircrafts.Model
    FROM
        AircraftLocation
    JOIN
        Airports on Airports.id = AircraftLocation.AirportID,
        Aircrafts on Aircrafts.id = AircraftLocation.AircraftID
    ORDER BY
        Airports.name
"""

# Execute the SQL query and store the result in a Pandas DataFrame
aircraft_airport_df = pd.read_sql_query(aircraft_airport, conn)

# Close the database connection since the query has been executed
conn.close()

# Display the resulting DataFrame
aircraft_airport_df
```

Python

```python
# Done by Antoine Van Gorp

# Connect to the SQLite database named 'final.db'
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Define the SQL query to retrieve information about passengers,
# and the departure location of the flights they booked
pass_loc = """
    SELECT
        Passengers.FirstName,
        Passengers.Lastname,
        Airports.Location AS DepartureLocation
    FROM
        Passengers
    JOIN
        Booking ON Passengers.id = Booking.PassengerID,
        Flights ON Booking.FlightID = Flights.id,
        Airports ON Flights.OriginAirportID = Airports.id
"""

# Execute the SQL query and store the result in a Pandas DataFrame
pass_loc_df = pd.read_sql_query(pass_loc, conn)

# Close the database connection since the query has been executed
conn.close()

# Display the resulting DataFrame
pass_loc_df
```

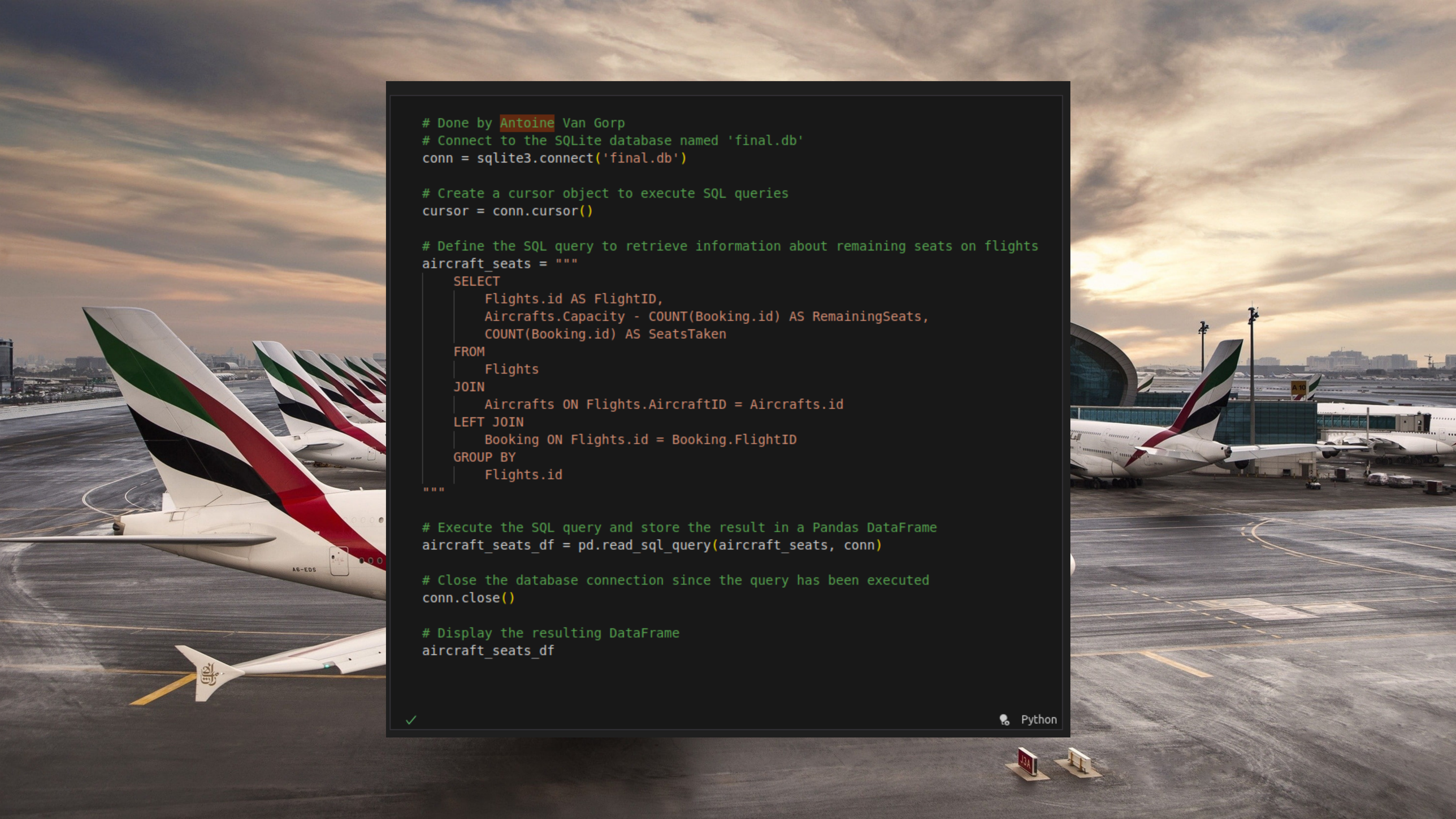
Python

```python
# Done by Antoine Van Gorp

# Connect to the SQLite database named 'final.db'
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Define the SQL query to retrieve information about remaining seats on flights
aircraft_seats = """
    SELECT
        Flights.id AS FlightID,
        Aircrafts.Capacity - COUNT(Booking.id) AS RemainingSeats,
        COUNT(Booking.id) AS SeatsTaken
    FROM
        Flights
    JOIN
        Aircrafts ON Flights.AircraftID = Aircrafts.id
    LEFT JOIN
        Booking ON Flights.id = Booking.FlightID
    GROUP BY
        Flights.id
"""

# Execute the SQL query and store the result in a Pandas DataFrame
aircraft_seats_df = pd.read_sql_query(aircraft_seats, conn)

# Close the database connection since the query has been executed
conn.close()

# Display the resulting DataFrame
aircraft_seats_df
```

Python

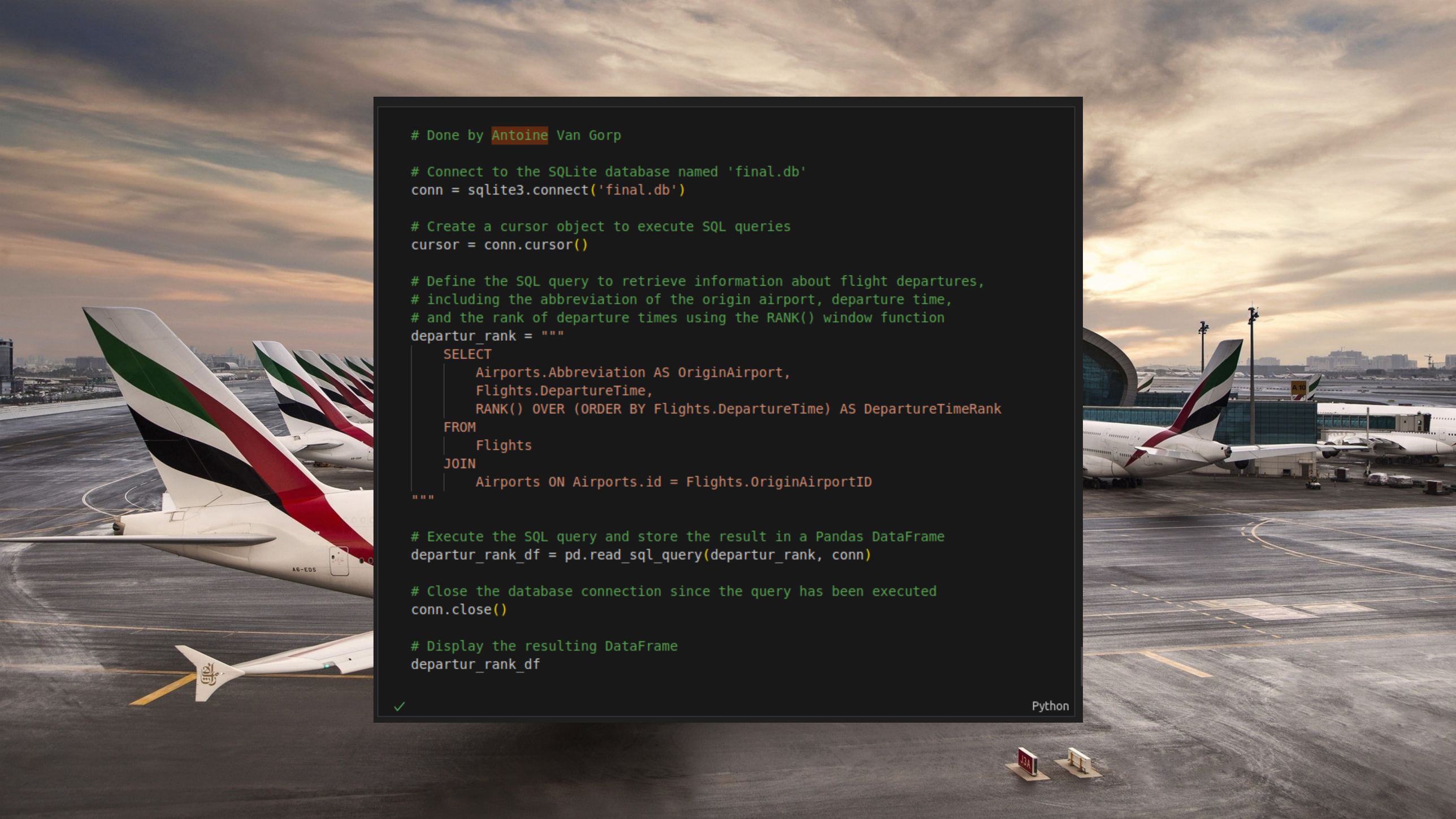
```python
# Done by Antoine Van Gorp

# Connect to the SQLite database named 'final.db'
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Define the SQL query to retrieve information about passengers, their baggage wei
# and assign a rank to each row based on baggage weight in descending order
pass_bag_waight = """
    SELECT
        Passengers.FirstName,
        Passengers.Lastname,
        Baggage.Weight,
        RANK() OVER (ORDER BY Baggage.Weight DESC) AS BaggageRank
    FROM
        Passengers
    JOIN
        Baggage ON Passengers.id = Baggage.PassengerID
"""

# Execute the SQL query and store the result in a Pandas DataFrame
pass_bag_waight_df = pd.read_sql_query(pass_bag_waight, conn)

# Close the database connection since the query has been executed
conn.close()

# Display the resulting DataFrame
pass_bag_waight_df
```

✓ Python



```python
# Done by Antoine Van Gorp
# Connect to the SQLite database named 'final.db'
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Define the SQL query to retrieve information about remaining seats on flights
aircraft_seats = """
    SELECT
        Flights.id AS FlightID,
        Aircrafts.Capacity - COUNT(Booking.id) AS RemainingSeats,
        COUNT(Booking.id) AS SeatsTaken
    FROM
        Flights
    JOIN
        Aircrafts ON Flights.AircraftID = Aircrafts.id
    LEFT JOIN
        Booking ON Flights.id = Booking.FlightID
    GROUP BY
        Flights.id
"""

# Execute the SQL query and store the result in a Pandas DataFrame
aircraft_seats_df = pd.read_sql_query(aircraft_seats, conn)

# Close the database connection since the query has been executed
conn.close()

# Display the resulting DataFrame
aircraft_seats_df
```

✓ Python



```python
# Done by Antoine Van Gorp

# Connect to the SQLite database named 'final.db'
conn = sqlite3.connect('final.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Define the SQL query to retrieve information about flight departures,
# including the abbreviation of the origin airport, departure time,
# and the rank of departure times using the RANK() window function
departur_rank = """
    SELECT
        Airports.Abbreviation AS OriginAirport,
        Flights.DepartureTime,
        RANK() OVER (ORDER BY Flights.DepartureTime) AS DepartureTimeRank
    FROM
        Flights
    JOIN
        Airports ON Airports.id = Flights.OriginAirportID
"""

# Execute the SQL query and store the result in a Pandas DataFrame
departur_rank_df = pd.read_sql_query(departur_rank, conn)

# Close the database connection since the query has been executed
conn.close()

# Display the resulting DataFrame
departur_rank_df
```

✓ Python

Thank you for listening