Query Explanations

1. **View Public Info:**

```
else:
    query = "SELECT * FROM flights WHERE "
    inserts = []
    if departureAirport != "":
        query += "departureAirport = %s AND "
        inserts.append(departureAirport)
    if arrivalAirport != "":
        query += "arrivalAirport = %s AND "
        inserts.append(arrivalAirport)
    if departureDate != "":
        query += "departureDateTime = %s AND "
        convertedDateTime = datetime.strptime(departureDate, '%Y-%m-%d')
        inserts.append(convertedDateTime)
    if arrivalDate != "":
        query += "arrivalDateTime = %s AND "
        convertedDateTime = datetime.strptime(arrivalDate, '%Y-%m-%d')
        inserts.append(convertedDateTime)
```

Building the query to search for all possible flights given the what the user decides to include as parameters

2. **Register**: 2 types of user registrations (Customer, and Airline Staff) option via forms.

```
customerQuery = 'SELECT * FROM Customer WHERE email = %s'
cursor.execute(customerQuery, (email))
customerData = cursor.fetchone()
print(customerData)
if(customerData):
    flash("This user already exists")
    return redirect(url_for("registerCustomer"))
else:
    ins = 'INSERT INTO Customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'
    encoded = password.encode()
    hashedPW = hashlib.sha256(encoded).hexdigest()
    cursor.execute(ins, (name,email, hashedPW,buildingNum,street,city,state,PhoneNumber,PassportNumber,
    passportExDate,dob,passportCountry))
```

Checking to see if the user's email exists in the customer database if not then insert the new account into Customer database

```
airlineQuery = 'SELECT * FROM AirlineStaff WHERE username = %s'
cursor.execute(airlineQuery, (username))
airlineData = cursor.fetchone()

if(airlineData):
    flash("This user already exists")
    return redirect(url_for("registerStaff"))
else:
    ins = 'INSERT INTO AirlineStaff VALUES(%s, %s, %s, %s, %s, %s, %s)'
    encoded = password.encode()
    hashedPW = hashlib.sha256(encoded).hexdigest()
    cursor.execute(ins, (username, hashedPW,firstName,lastName,dob,phoneNumber,airline
```

Checking to see if the username of the registering airline staff already exists in Staff database if not then insert the new account into the DB.

3. **Login: 2 types of user login (Customer, and Airline Staff).**

```
        customerQuery = 'SELECT * FROM Customer WHERE email = %s and password = %s'
        cursor.execute(customerQuery, (username, hashedPW))
        customerData = cursor.fetchone()

        airlineQuery = 'SELECT * FROM AirlineStaff WHERE username = %s and password = %s'
        cursor.execute(airlineQuery, (username, hashedPW))
        airlineData = cursor.fetchone()
```

Checks to see if the username and password exists in either Customer or Airline Staff database and if it does allows the user to log in successfully otherwise sends an error message

Customer use cases:

4. **View My flights**

```
customerQuery = 'SELECT DISTINCT f.* from Ticket t join Flights f on t.flightNumber = f.flightNumber
where customerEmail = %s and departureDateTime >= now()'
```

Selecting all the future flights (by making sure the DepartureDateTime is greater than the current time) the customer has coming up (with the where customeremail = %s)

5. Search for flights

```
else:
    query = "SELECT * FROM flights WHERE "
    inserts = []
    if departureAirport != "":
        query += "departureAirport = %s AND "
        inserts.append(departureAirport)
    if arrivalAirport != "":
        query += "arrivalAirport = %s AND "
        inserts.append(arrivalAirport)
    if departureDate != "":
        query += "departureDateTime = %s AND "
        convertedDateTime = datetime.strptime(departureDate, '%Y-%m-%d')
        inserts.append(convertedDateTime)
    if arrivalDate != "":
        query += "arrivalDateTime = %s AND "
        convertedDateTime = datetime.strptime(arrivalDate, '%Y-%m-%d')
        inserts.append(convertedDateTime)
```

Building the query to search for all possible flights given the what the user decides to include as parameters

6. **Purchase tickets**

```
numSeatsQuery = 'SELECT a.seats, f.basePrice, f.airline from Flights f join Airplane a on a.airplaneId
= f.airplaneId where f.flightNumber = %s'
cursor.execute(numSeatsQuery, (flightNumber))
data = cursor.fetchall()
```

query to check number of seats remaining on flight as well as getting the base price and airline

```
if (numSeats == 0):
    flash("No More Seats Avail")
    return redirect(url_for("purchaseTicket"))
elif (numSeats < 15):
    basePrice = basePrice*1.15 #scam customer
#insert into ticket table + get credit card info
purchaseTicketQuery = 'INSERT into Ticket VALUES(%s, %s, %s, %s, %s, now(), %s, %s, %s, %s)'
```

query to allow purchases by inserting into the ticket database

6. **Give Ratings and Comment on previous flights:**

```
checkFlightQuery = 'SELECT ticketID from Ticket where flightNumber = %s and customerEmail = %s'
cursor.execute(checkFlightQuery, (flightNumber,session['username']))
customerTicket = cursor.fetchone()
if (customerTicket is None):
    flash("User didn't take flight")
    return redirect(url_for('rate'))
else:
    ins = 'INSERT INTO ViewPreviousFlights VALUES(%s, %s, %s, %s)'
    cursor.execute(ins, (session['username'],flightNumber,rate,comment))
```

Checks to see if the user has taken the flight then inserts their rate and comment if true.

7.**Track My Spending**:

```
#option to specify a range of dates and look at charts for specified time
spendingQuery = 'SELECT sum(soldPrice) from Ticket where customerEmail = %s and purchaseDateTime > date_add
(now(), INTERVAL -1 YEAR)'
cursor.execute(spendingQuery, (session['username']))
totalSpentOneYear = cursor.fetchone()
spendingSixMonthsQuery = 'SELECT MONTH(purchaseDateTime) AS Purchase_Month, sum(soldPrice) AS monthlySpent
from Ticket where customerEmail = %s and purchaseDateTime > (SELECT CURDATE() - INTERVAL 6 MONTH) GROUP BY
MONTH(purchaseDateTime) ORDER BY MONTH(purchaseDateTime);'
cursor.execute(spendingSixMonthsQuery, (session['username']))
```

Spending Query : sums up the sold price for all tickets sold in the past year

SpendingSixMonthsQuery : does a group by month and for every month calculates the total sold price for a specific customer within a 6 month interval

```
    end = request.form.get("end")
    cursor.execute('''SELECT MONTH(purchaseDateTime) AS Purchase_Month, sum(soldPrice) AS monthlySpent
    from Ticket where customerEmail = %s and (purchaseDateTime BETWEEN %s and %s) ORDER BY MONTH
    (purchaseDateTime);''', (session['username'], start, end))
    spendingData = cursor.fetchall()
```

Same as SpendingSixMonthsQuery only the user can specify between which dates

8.**Logout:** The session is destroyed and a “goodbye” page or the login page is displayed. **Airline Staff use cases:**

```
def logout():
    del session['username']
    if session['user_type'] == "AirlineStaff":
        del session['airline']
    del session['user_type']
    return redirect(url_for("index"))
```

Log out destroys the session and returns the user to start page

4. **View flights:**

```
    cursor = db.cursor()
    staffQuery = 'SELECT * FROM Flights WHERE airline = %s'
    cursor.execute(staffQuery, (session['airline']))
    airlineFlights = cursor.fetchall()
```

Selects all flights from the airline the staff works at

5**. Create new flights:**

```
  staffQuery = 'SELECT * FROM Flights WHERE airline = %s'
  cursor.execute(staffQuery, (session['airline']))
```

```
cursor.execute("INSERT INTO Flights VALUES (%s, %s, %s, %s, %s, %s, %s, %s)", (flightNumber,
departureDateTimeSQL, airline, arrivalAirport, basePrice, arrivalDateTimeSQL, departureAirport,
airplaneId))
```

checks if the user is a staff, if yes inserts the query to create a new flight with form elements

6**. Change Status of flights:**

```
 if "username" in session and session['user_type'] == "AirlineStaff":
    if request.method == "POST":
        flightNumber = request.form.get("flightNumber")
        status = request.form.get("status")

        cursor = db.cursor()
        cursor.execute("INSERT INTO FlightStatus VALUES (%s, %s, %s)", (flightNumber, session['username'],
        status))
```

checks if the user is a staff, if yes inserts the query to create a change the status of a flight with form elements

7**. Add airplane in the system:** He or she adds a new airplane, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. In the confirmation page, she/he will be able to see all the airplanes owned by the airline he/she works for.

```
staffQuery = 'SELECT * FROM Airplane WHERE airline = %s'
cursor.execute(staffQuery, (session['airline']))
airlinePlanes = cursor.fetchall()

if request.method == "POST":
    airplaneId = request.form.get("airplaneId")
    seats = request.form.get("seats")
    airline = request.form.get("airline")

    cursor.execute("INSERT INTO Airplane VALUES (%s, %s, %s)", (airplaneId, seats, airline))
    db.commit()
```

checks if the user is a staff, if yes inserts the query to create a add an airplane in the system with form elements

8**. Add new airport in the system:** He or she adds a new airport, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action.

```
def addAirport():
    if "username" in session and session['user_type'] == "AirlineStaff":
        if request.method == "POST":
            airportID = request.form.get("airportID")
            name = request.form.get("name")
            city = request.form.get("city")

            cursor = db.cursor()
            cursor.execute("INSERT INTO Airport VALUES (%s, %s, %s)", (airportID, name, city))
```

checks if the user is a staff, if yes inserts the query to create a add an airport in the system with form elements

9**. View flight ratings:**

```
cursor = db.cursor()
cursor.execute("SELECT AVG(rate) FROM ViewPreviousFlights WHERE FlightNumber = %s GROUP BY
FlightNumber", (flightID))
avg = cursor.fetchone()

cursor.execute("SELECT comment FROM ViewPreviousFlights WHERE FlightNumber = %s", (flightID))
comments = cursor.fetchall()
```

To get the avg rating for a flight, group by flight number and take the aggregate function avg of the rate then a separate query to get all the comments for the flight

11**. View frequent customers:**

```
cursor.execute("SELECT customerEmail, count(customerEmail) from Ticket ORDER BY count(customerEmail)
LIMIT 1")
most = cursor.fetchone()

cursor.execute("SELECT flightNumber FROM Ticket WHERE customerEmail = %s AND airlineName = (SELECT
airline FROM AirlineStaff WHERE username = %s)", (email, session['username']))
flights = cursor.fetchall()
```

Gets the most frequent customers by selecting the top customer with the most counts of tickets
Gets all the flights of a particular customer on a specific airline with where clauses

12**. View reports:**

```
cursor = db.cursor()
cursor.execute("SELECT count(ticketID) FROM Ticket WHERE (purchaseDateTime BETWEEN %s AND %s);",
(start, end))
total = cursor.fetchall()

cursor.execute('''SELECT MONTH(purchaseDateTime) AS
Purchase_Month ,count(ticketID) AS Ticket_Count
FROM Ticket
WHERE airlineName = %s and (purchaseDateTime BETWEEN %s AND %s)
GROUP BY MONTH(purchaseDateTime)
ORDER BY MONTH(purchaseDateTime);
''', (session['airline'], start, end))
spending = cursor.fetchall()
```

GEts the total count of tickets using unique identifier ticketID between specified dates

Does a group by month and for every month calculates the total count of tickets within a specified range for the airline the staff works for

13. **View Earned Revenue:** Show total amount of revenue earned from ticket sales in the last month and last year.

```
cursor.execute("SELECT sum(soldPrice) FROM Ticket WHERE airline = %s AND purchaseDateTime > date_add(now()
, INTERVAL -1 MONTH);", (session['airline']))
monthly = cursor.fetchall()
cursor.execute("SELECT sum(soldPrice) FROM Ticket WHERE airline = %s AND purchaseDateTime > date_add(now()
, INTERVAL -1 YEAR);", (session['airline']))
```

Two queries that add up the sold prices of all the tickets within the purchase date/time of a month ago, and a year ago respectively.

14. **View Top destinations:** Find the top 3 most popular destinations for last 3 months and last year (based on tickets already sold).

```
        cursor.execute('''
        SELECT distinct a.city
        FROM Ticket t
        join Flights f on t.flightNumber = f.flightNumber
        join Airport a on f.arrivalAirport = a.name
        where purchaseDateTime > date_add(now(), INTERVAL -3 MONTH)
        limit 3;''')
        monthly = cursor.fetchall()
        cursor.execute('''
        SELECT distinct a.city
        FROM Ticket t
        join Flights f on t.flightNumber = f.flightNumber
        join Airport a on f.arrivalAirport = a.name
        where purchaseDateTime > date_add(now(), INTERVAL -1 YEAR)
        limit 3;''')
```

First query grabs the city from the airport table, and compares the airport ID to the arrival airport on certain flights, which are identified by the flight ID given on tickets, with a purchase datetime limit of 3 months ago. The second one does the same but for a year.

15. **Logout:** The session is destroyed and a “goodbye” page or the login page is displayed.