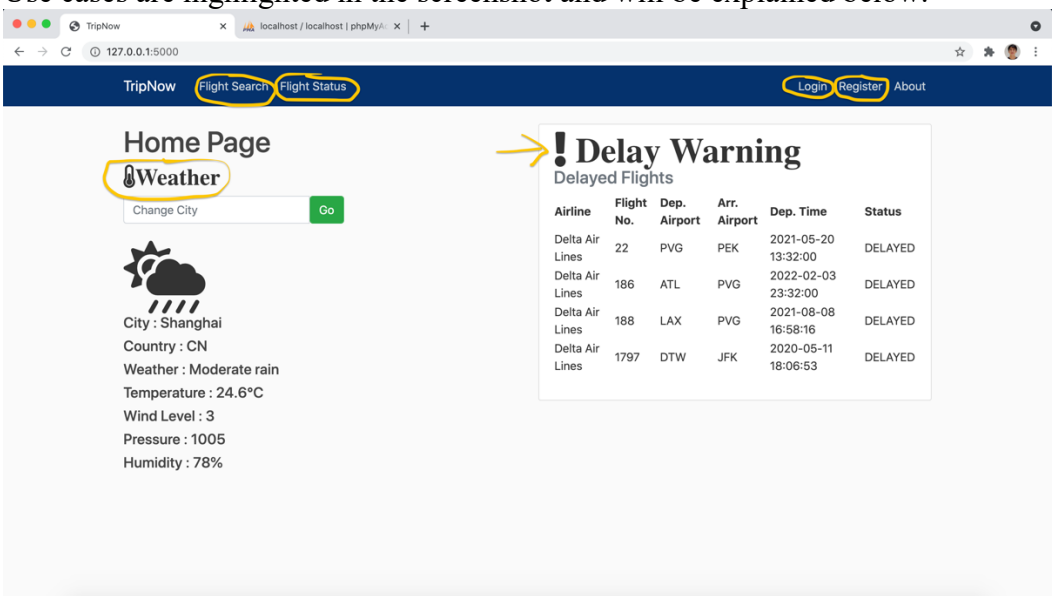
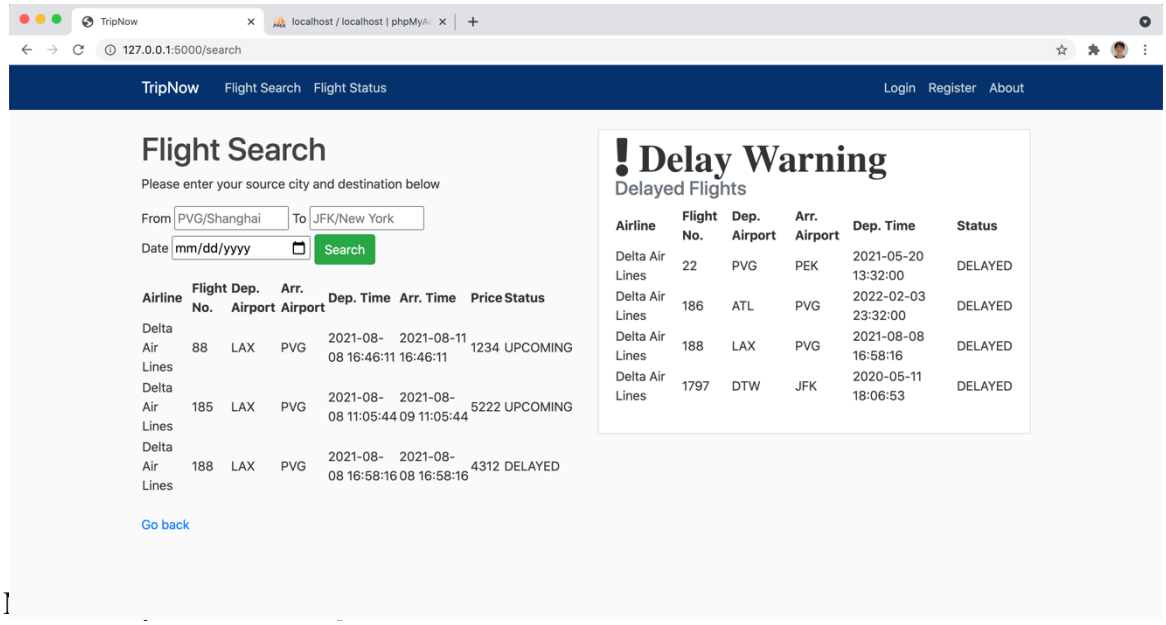
This file may not contain all the details of codes for error checking, but all the important queries are included.

1. General Home Page (Public) app.py + layout.html

Use cases are highlighted in the screenshot and will be explained below.

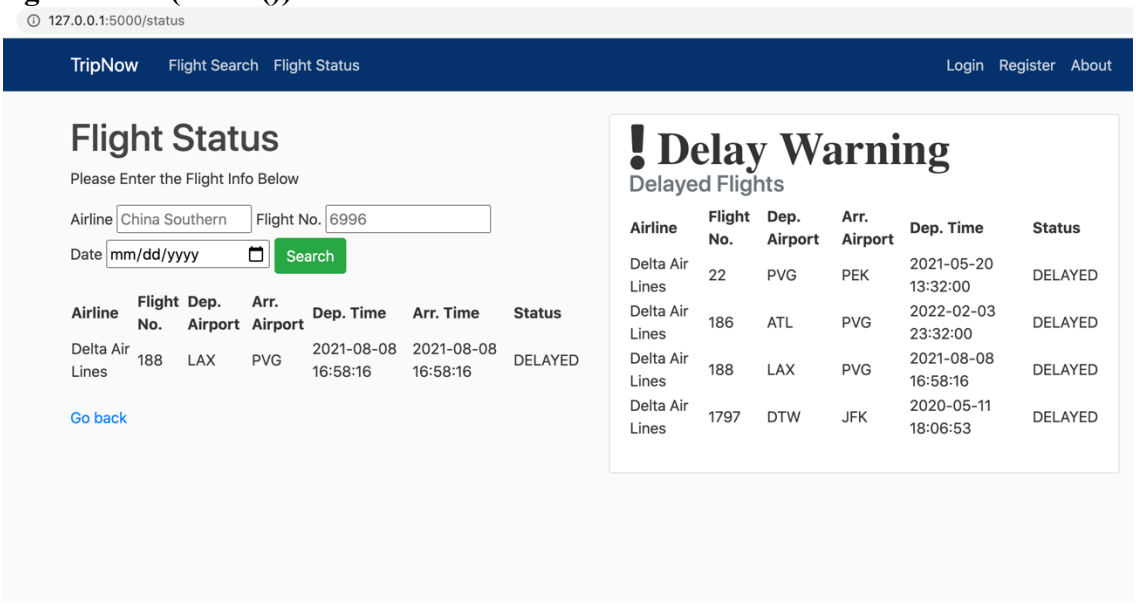


1-1. Flight Search (search())




```
else:
    cursor = conn.cursor()
    query = "SELECT airline_name, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, price, status \
    FROM airport AS a, flight AS b, airport AS c \
    WHERE a.airport_name = b.departure_airport\
    AND b.arrival_airport = c.airport_name\
    AND a.airport_city= \'{}\' \
    AND c.airport_city = \'{}\' \
    AND departure_time LIKE \'{}\'"
    cursor.execute(query.format(departure, arrival, '%'+str(date)+'%'))
    flight = cursor.fetchall()
```

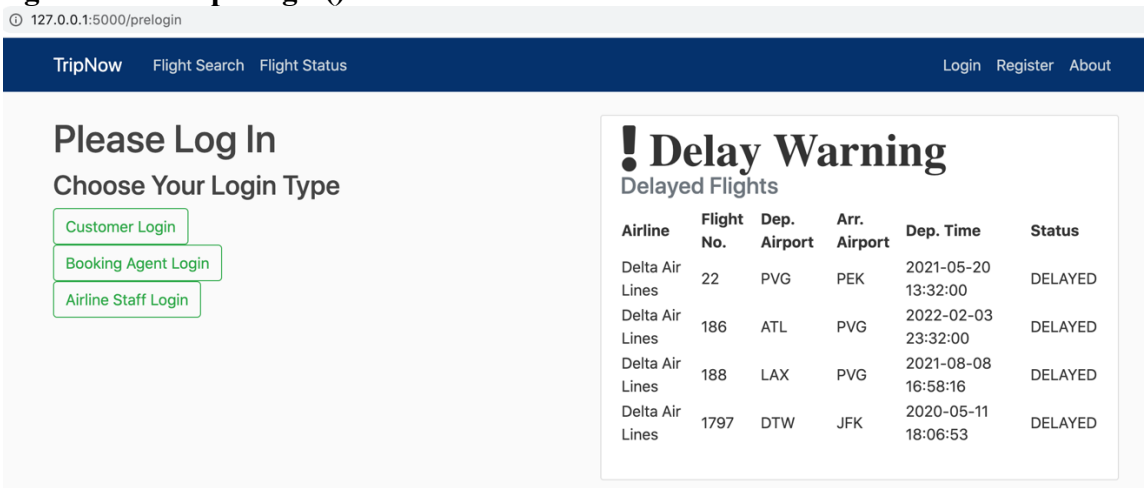
1-2. Flight Status (status())



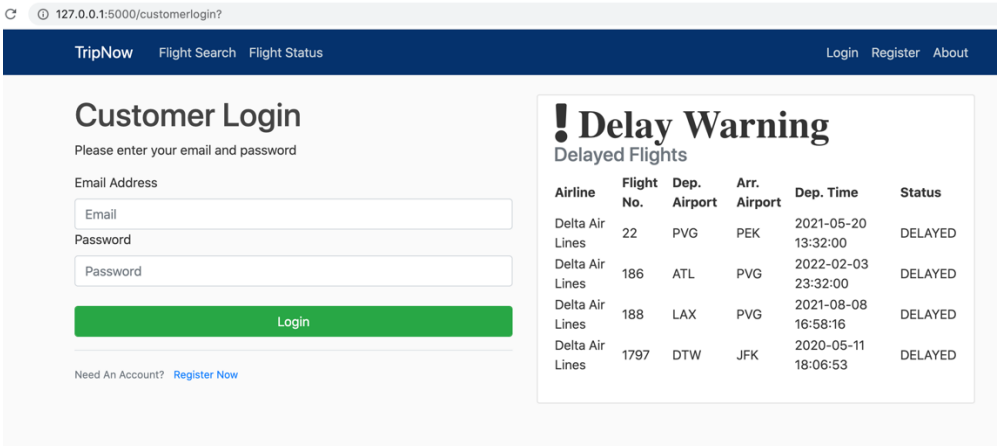
Main query under status():

```
164         cursor = conn.cursor()
165         query = "SELECT airline_name, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, status \
166             FROM flight WHERE airline_name = \'{}\' AND flight_num = \'{}\' AND departure_time LIKE \'{}\'"
167
168         cursor.execute(query.format(airline, flight_num, '%'+str(date)+'%'))
169         flight = cursor.fetchall()
```

1-3. Login → return prelogin()



Choose login type (customerlogin(), booklogin(), stafflogin())
Take customerlogin() for example:

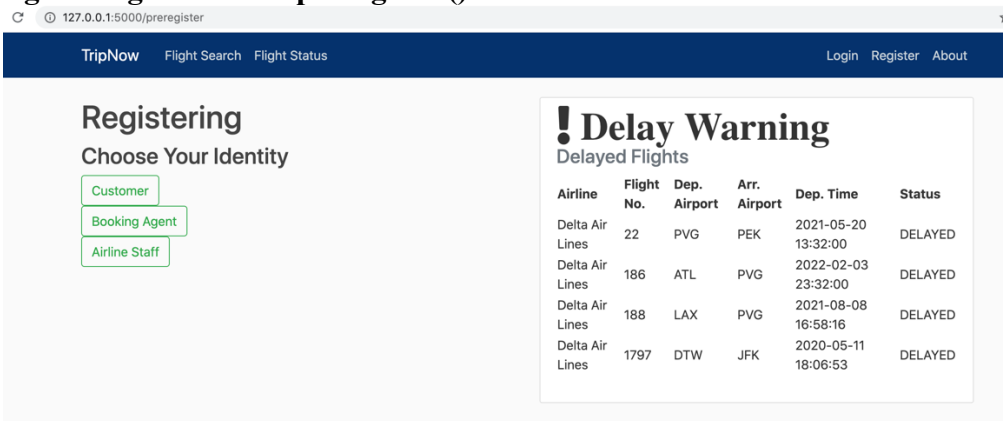


Main Query under customerlogin():

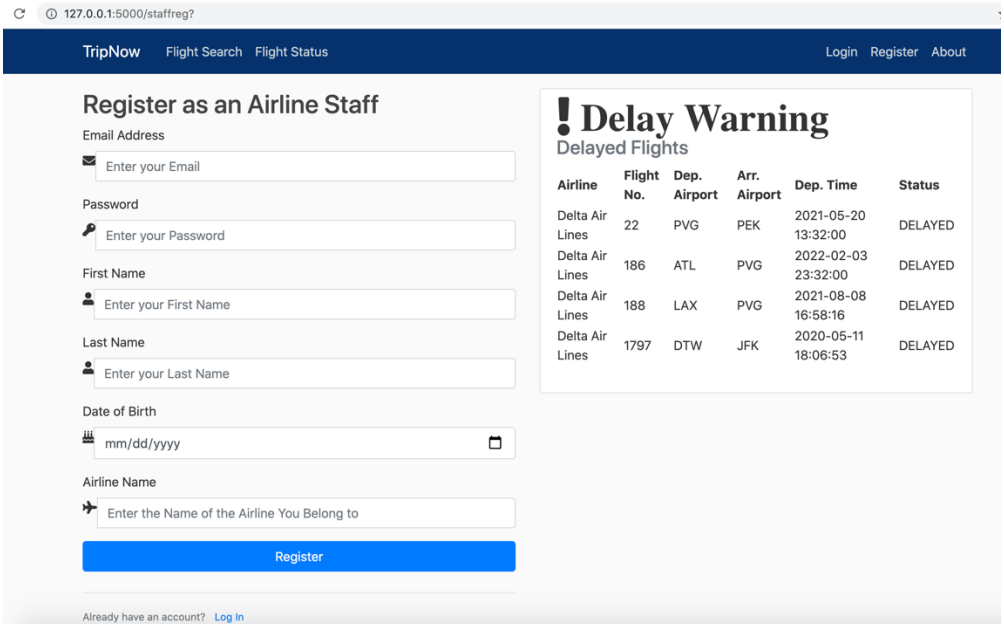
```
if request.method == 'POST':

    # Create variables for easy access
    email = request.form['email']
    password = request.form['password']

    cursor.execute('SELECT * FROM customer WHERE email = %s AND password = %s', (email, password))
    # Fetch one record and return result
    data = cursor.fetchone()

    # If account exists in accounts table in out database
    if data:
        # Create session data, we can access this data in other routes
        #session['name'] = ''.join(name).strip('',)
        name = data[1]
        session['name'] = name.strip('')
        session['cus'] = 'yes'
        session['username'] = email
        # Redirect to home page
        return redirect(url_for('cushome'))
    else:
        # Account doesnt exist or username/password incorrect
        msg = 'The email address or password you entered is incorrect!'

return render_template('logincus.html', msg=msg, data1=data1)
```

Stafflogin() and booklogin() are similar. It will create unique sessiom[‘cus’] for customer, session[‘id’] for booking agent, and session[‘airline name’] for staff to prevent them from accessing other account types.

1-4. Registering → Return preregister()



Choose registering type (customerreg(), bookreg(), staffreg())
Take staffreg() for example:



Main query under staffreg():

```
1762                else:
1763                    # Account doesnt exists and the form data is valid, now insert new account into accounts table
1764                    ins = "INSERT INTO airline_staff VALUES( \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\')"
1765                    cursor.execute(ins.format( email, password, first_name, last_name, date_of_birth, airline_name))
1766                    conn.commit()
1767                    msg = 'You have successfully registered! Log in to view your homepage!'
1768
1769        return render_template('registerstaff.html', msg=msg, data1=data1)
```

customerreg() and bookreg() are similar. We will make sure there's no identical email address beforehand:

```
#Check if account exists using MySQL
query = "SELECT * FROM booking_agent WHERE email = \'{}\'"
cursor.execute(query.format(email))
#cursor.execute('SELECT * FROM accounts WHERE username = %s', (username))
account = cursor.fetchone()
# If account exists show error and validation checks
if account:
    msg = 'Email already exists!'
```

1-5. Weather under home():

Manually convert the icons.

```
# source contain json data from api
try:
    source = urllib.request.urlopen('http://api.openweathermap.org/data/2.5/weather?q=' + city + '&appid=' + api).read()

    list_of_data = json.loads(source)
    icon = {
        "01d": "fas fa-sun",
        "01n": "fas fa-sun",
        "02n": "fas fa-cloud-moon",
        "02d": "fas fa-cloud-sun",
        "03d": "fas fa-cloud-meatball",
        "03n": "fas fa-cloud-meatball",
        "04d": "fas fa-cloud-meatball",
        "04n": "fas fa-cloud-meatball",
        "09d": "fas fa-cloud-showers-heavy",
        "09n": "fas fa-cloud-showers-heavy",
        "10d": "fas fa-cloud-sun-rain",
        "10n": "fas fa-cloud-sun-rain",
        "11d": "fas fa-poo-storm",
        "11n": "fas fa-poo-storm",
        "13d": "fas fa-snowflake",
        "13n": "fas fa-snowflake",
        "50d": "fas fa-smog",
        "50n": "fas fa-smog"
    }

    data = {
        "cityname": str(list_of_data["name"].capitalize()),
        "country_code": str(list_of_data['sys']['country']),
        "coordinate": str(list_of_data['coord']['lon']) + ' '
                    + str(list_of_data['coord']['lat']),
        "weather": str(list_of_data['weather'][0]['description'].capitalize()),
        "temp": str(round(list_of_data['main']['temp']-273, 1)) + '°C',
        "wind": str(list_of_data['wind']['speed']),
        "pressure": str(list_of_data['main']['pressure']),
        "humidity": str(list_of_data['main']['humidity'])+'%',
        "icon": str(icon[list_of_data['weather'][0]['icon']])
    }
```

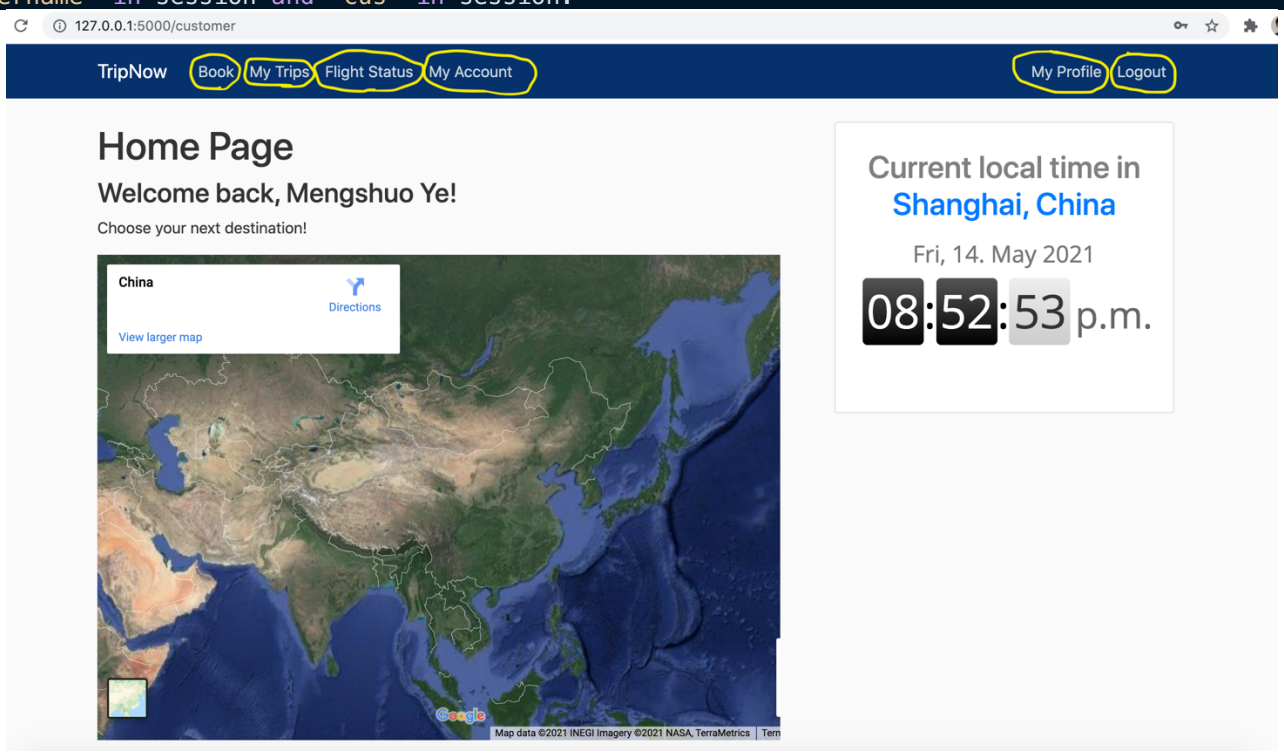
1-6. Delay Warning (embedded in layout.html):

```
cursor = conn.cursor()
cursor.execute('SELECT airline_name, flight_num, departure_airport, arrival_airport, departure_time FROM flight WHERE status = "DELAYED"')
data1 = cursor.fetchall()
msg=''
data = {}
```

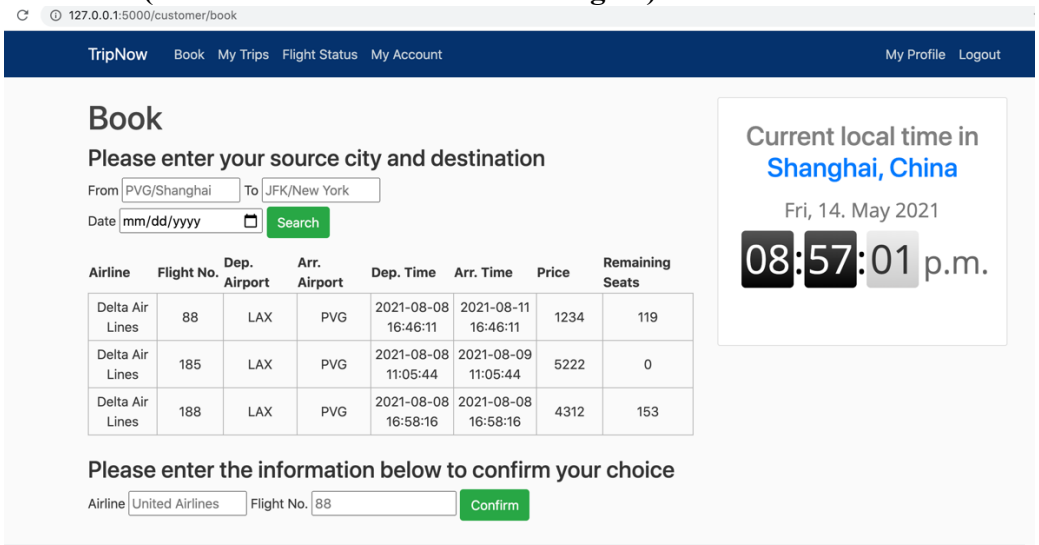
2. Customer Use Cases (app.py + layoutcus.html)

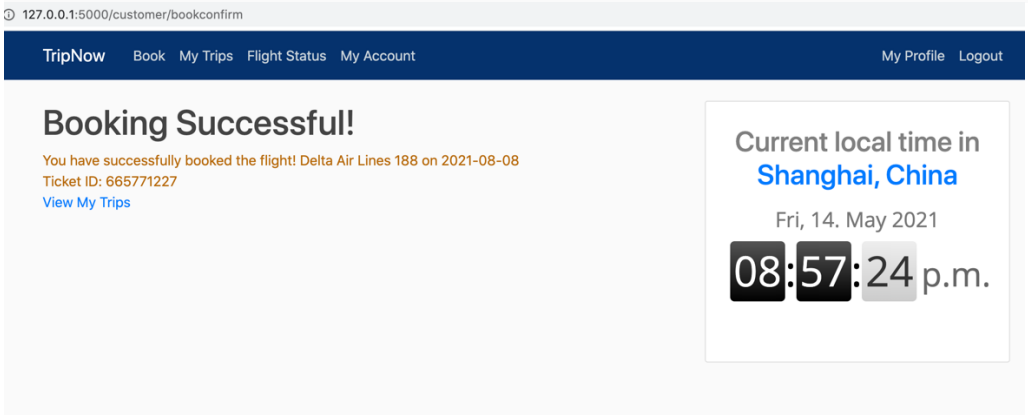
For every use case, check:

```
if 'username' in session and 'cus' in session:
```



2-1. Book (Purchase tickets + Search for flights)





2-1-1. Search and confirm (under bookcus() and bookcusconfirm())

```
if departure.isupper():
                    cursor = conn.cursor()
                    query = "SELECT ALL airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, seats-count(ticket_id) AS seat_left \
                            FROM flight LEFT JOIN ticket USING (airline_name, flight_num) JOIN
airplane USING (airline_name, airplane_id)\
                            WHERE departure_airport = \'{}\' \
                            AND arrival_airport = \'{}\'\
                            AND departure_time LIKE \'{}\' \
                            GROUP BY airline_name, flight_num"

                    cursor.execute(query.format(departure, arrival, '%'+str(flight_date)+'%'))
                    flight = cursor.fetchall()
```

This one above is for searching with airport code.

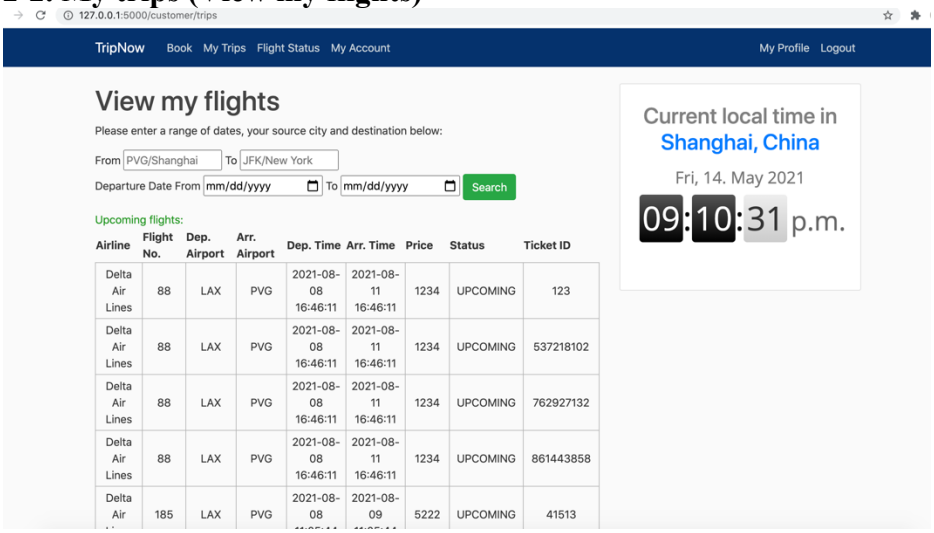
```
query = "SELECT ALL airline_name, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, price,
seats-count(ticket_id) AS seat_left\
                            FROM airport AS a, (flight LEFT JOIN ticket USING (airline_name, flight_num) JOIN airplane
USING (airline_name, airplane_id)), airport AS c \
                            WHERE a.airport_name = departure_airport\
                            AND arrival_airport = c.airport_name\
                            AND a.airport_city= \'{}\'\
                            AND c.airport_city = \'{}\'\
                            AND departure_time LIKE \'{}\'\
                            GROUP BY airline_name, flight_num"
```

This one above is for searching with city name.

2-1-2. Checking available seat number and complete booking Under bookcusconfirm():

```
query = "SELECT seats-count(ticket_id) AS seat_left\
FROM flight JOIN ticket USING (airline_name, flight_num) JOIN airplane USING (airline_name, airplane_id)\
WHERE airline_name = \'{}\' \
AND flight_num = \'{}\'"
cursor.execute(query.format(airline, flight_num))
#stores the results in a variable
data = cursor.fetchone()
if data[0] == None:
    msg = "The flight number does not exist!"
    return render_template('bookcus.html', msg=msg, flight=flight, date=flight_date)
elif data[0] == 0:
    #If the previous query returns data, then user exists
    msg = "This flight has been sold out!"
    return render_template('bookcus.html', msg=msg, flight=flight, date=flight_date)
else:
    a = 0
    while a == 0:
        tkt_id = randint(100000000, 999999999)
        query = "SELECT * FROM ticket WHERE ticket_id = \'{}\'"
        cursor.execute(query.format(tkt_id))
        data = cursor.fetchone()
        if not data:
            a = 1
    try:
        ins1 = "INSERT INTO ticket VALUES(\'{}\', \'{}\', \'{}\')"
        cursor.execute(ins1.format(tkt_id, airline, flight_num))
        today = datetime.today().strftime('%Y-%m-%d')
        ins2 = "INSERT INTO purchases (ticket_id, customer_email, purchase_date) VALUES(\'{}\', \'{}\', \'{}\')"
        cursor.execute(ins2.format(tkt_id, email, today))
        conn.commit()
        cursor.close()
        msg = "You have successfully booked the flight! " + airline + " " + flight_num + " on " + flight_date
        msg1 = "Ticket ID: " + str(tkt_id)
        #flash("Booking Successful! Flight:" + airline + flight_num)
        return render_template('bookcusconfirm.html', msg=msg, msg1=msg1)
```

2-2. My trips (View my flights)



Queries under tripcus():

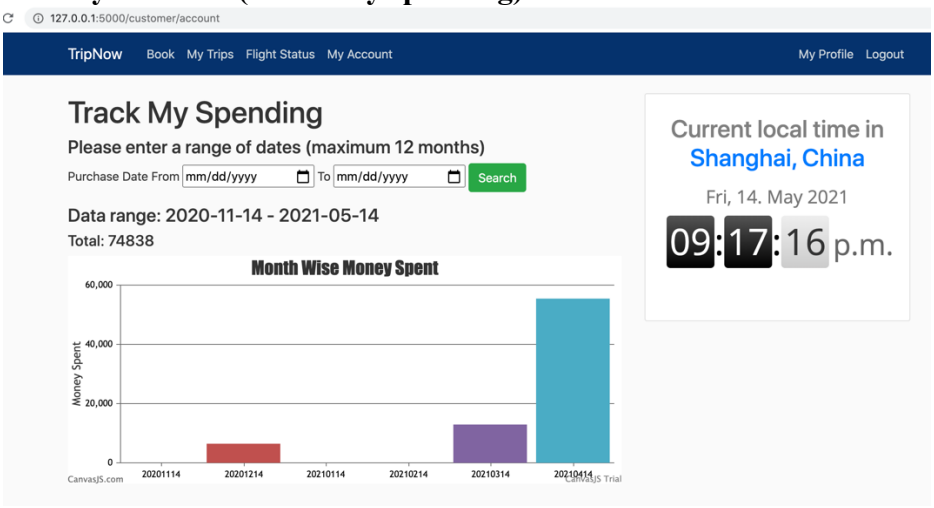
Basic default view

```
q = 'SELECT airline_name, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, price, status, ticket_id \
    FROM flight NATURAL JOIN ticket NATURAL JOIN purchases \
    WHERE departure_time >= \'{}\' AND customer_email = \'{}\''
cursor.execute(q.format(start_date, customer_email))
t = cursor.fetchall()
```

Advanced search:

```
        if request.method == 'POST':
            start_date = request.form['start date']
            end_date = request.form['end date']
            departure = request.form['departure']
            arrival = request.form['arrival']

            if departure.isupper():
                cursor = conn.cursor()
                q = 'SELECT f.airline_name, f.flight_num, f.departure_airport, f.arrival_airport, f.departure_time, f.arrival_time,
f.price, f.status, t.ticket_id \
                    FROM flight f JOIN ticket t ON(f.airline_name = t.airline_name AND f.flight_num = t.flight_num) \
                    JOIN purchases p ON(t.ticket_id = p.ticket_id) JOIN airport a ON(f.departure_airport = a.airport_name) \
                    JOIN airport b ON(f.arrival_airport = b.airport_name) \
                    WHERE p.customer_email = \'{}\' AND p.purchase_date <= \'{}\' AND p.purchase_date >= \'{}\' \
                    AND f.departure_airport = \'{}\' AND f.arrival_airport = \'{}\' '
                cursor.execute(q.format(customer_email, end_date, start_date, departure, arrival))
                t = cursor.fetchall()
            else:
                cursor = conn.cursor()
                q = 'SELECT f.airline_name, f.flight_num, f.departure_airport, f.arrival_airport, f.departure_time, f.arrival_time,
f.price, f.status, t.ticket_id \
                    FROM flight f JOIN ticket t ON(f.airline_name = t.airline_name AND f.flight_num = t.flight_num) \
                    JOIN purchases p ON(t.ticket_id = p.ticket_id) JOIN airport a ON(f.departure_airport = a.airport_name) \
                    JOIN airport b ON(f.arrival_airport = b.airport_name) \
                    WHERE p.customer_email = \'{}\' AND p.purchase_date <= \'{}\' AND p.purchase_date >= \'{}\' \
                    AND a.airport_city = \'{}\' AND b.airport_city = \'{}\''
                cursor.execute(q.format(customer_email, end_date, start_date, departure, arrival))
                t = cursor.fetchall()
```

2-3. Flight Status: Similar to the public flight status. Codes under statuscus().
2-4. My Account (Track my spending): Based on CanvasJS



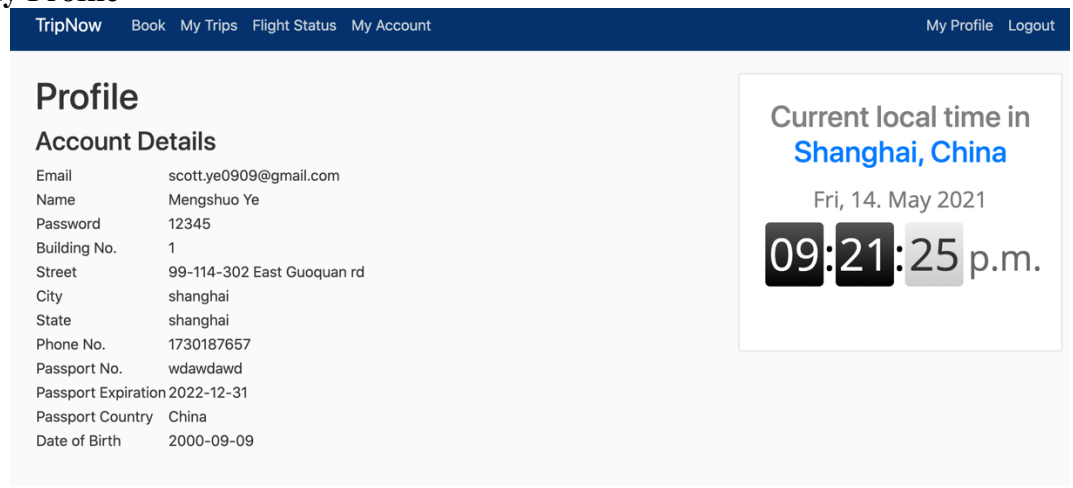
Codes under accountcus():
Default 6 months:

```
curdate = start_date
curdate1 = start_date + relativedelta(months=+1)
i = 0
cursor = conn.cursor()
while curdate < end_date:

    q = "SELECT SUM(price) FROM purchases NATURAL JOIN ticket NATURAL JOIN flight\
        WHERE customer_email = \'{}\'\
        AND purchase_date > \'{}\' AND purchase_date <= \'{}\'"
    cursor.execute(q.format(email, curdate, curdate1))
    result.append(cursor.fetchone()[0])
    i += 1
    curdate = curdate + relativedelta(months=+1)
    curdate1 = curdate + relativedelta(months=+1)
```

Advanced search:

```
else:
    i = 0
    curdate = datetime.strptime(start_date, '%Y-%m-%d')
    #curdate = start_date
    curdate1 = curdate + relativedelta(months=+1)

    cursor = conn.cursor()
    while curdate <= date1:
        q = "SELECT SUM(price) FROM purchases NATURAL JOIN ticket NATURAL JOIN flight\
        WHERE customer_email = \'{}\'\
        AND purchase_date > \'{}\' AND purchase_date <= \'{}\'"
        cursor.execute(q.format(email, str(curdate)[0:10], str(curdate1)[0:10]))
        result_temp.append(cursor.fetchone()[0])

        i += 1
        curdate = curdate + relativedelta(months=+1)
        curdate1 = curdate + relativedelta(months=+1)

        if i == 13:
            msg = 'Please keep the date range within 12 months!'
            return render_template('accountcus.html', msg=msg, tot=tot, ran=ran, num=num)
```

2-5. My Profile



Codes under profilecus():

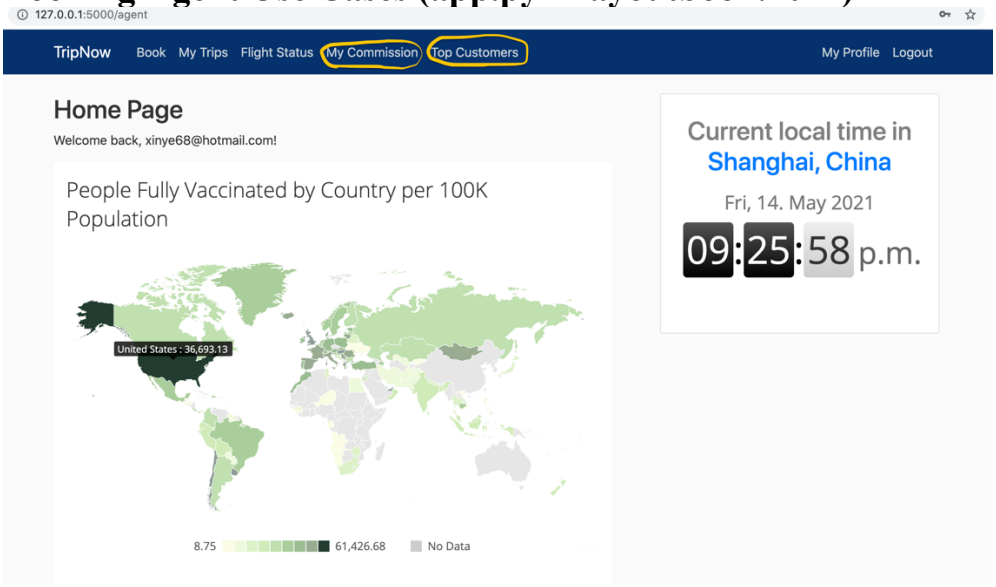
```
506  @app.route('/customer/profile')
507  def profilecus():
508      if 'username' in session and 'cus' in session:
509
510          cursor = conn.cursor()
511          username = session['username']
512          # We need all the account info for the user so we can display it on the profile page
513          q = 'SELECT * FROM customer WHERE email = \'{}\''
514          cursor.execute(q.format(username))
515
516          account = cursor.fetchone()
517          # Show the profile page with account info
518          return render_template('profilecus.html', account=account)
519      # User is not loggedin redirect to login page
520      return redirect(url_for('prelogin'))
```

2-6. Logout

Pop everything possible from session

```
572  @app.route('/logout')
573  def logout():
574      session.pop('username', None)
575      session.pop('name', None)
576      session.pop('airline name', None)
577      session.pop('id', None)
578      session.pop('cus', None)
579
580      # Redirect to login page
581      return redirect(url_for('prelogin'))
```

3. Booking Agent Use Cases (app.py + layoutbook.html)



3-1. My Trips (View My Flights)

Similar to the one in customer. Codes under tripsagent().

```
q = 'SELECT airline_name, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, price, status,
ticket_id \
            FROM flight NATURAL JOIN ticket NATURAL JOIN purchases \
            WHERE departure_time >= \'{}\' AND booking_agent_id = \'{}\''
        cursor.execute(q.format(start_date, booking_agent_id))
        t = cursor.fetchall()
```

```
if request.method == 'POST':
            start_date = request.form['start date']
            end_date = request.form['end date']
            departure = request.form['departure']
            arrival = request.form['arrival']
            if departure.isupper():
                cursor = conn.cursor()
                q = 'SELECT f.airline_name, f.flight_num, f.departure_airport, f.arrival_airport, f.departure_time, f.arrival_time,
f.price, f.status, t.ticket_id \
                    FROM flight f JOIN ticket t ON(f.airline_name = t.airline_name AND f.flight_num = t.flight_num) \
                    JOIN purchases p ON(t.ticket_id = p.ticket_id) JOIN airport a ON(f.departure_airport = a.airport_name) \
                    JOIN airport b ON(f.arrival_airport = b.airport_name) \
                    WHERE p.booking_agent_id = \'{}\' AND p.purchase_date <= \'{}\' AND p.purchase_date >= \'{}\' \
                    AND f.departure_airport = \'{}\' AND f.arrival_airport = \'{}\' '
                cursor.execute(q.format(booking_agent_id, end_date, start_date, departure, arrival))
                t = cursor.fetchall()
            else:
                cursor = conn.cursor()
                q = 'SELECT f.airline_name, f.flight_num, f.departure_airport, f.arrival_airport, f.departure_time, f.arrival_time,
f.price, f.status , t.ticket_id \
                    FROM flight f JOIN ticket t ON(f.airline_name = t.airline_name AND f.flight_num = t.flight_num) \
                    JOIN purchases p ON(t.ticket_id = p.ticket_id) JOIN airport a ON(f.departure_airport = a.airport_name) \
                    JOIN airport b ON(f.arrival_airport = b.airport_name) \
                    WHERE p.booking_agent_id = \'{}\' AND p.purchase_date <= \'{}\' AND p.purchase_date >= \'{}\' \
                    AND a.airport_city = \'{}\' AND b.airport_city = \'{}\''
                cursor.execute(q.format(booking_agent_id, end_date, start_date, departure, arrival))
                t = cursor.fetchall()
```

3-2. Book (Search for flights & Purchase tickets)

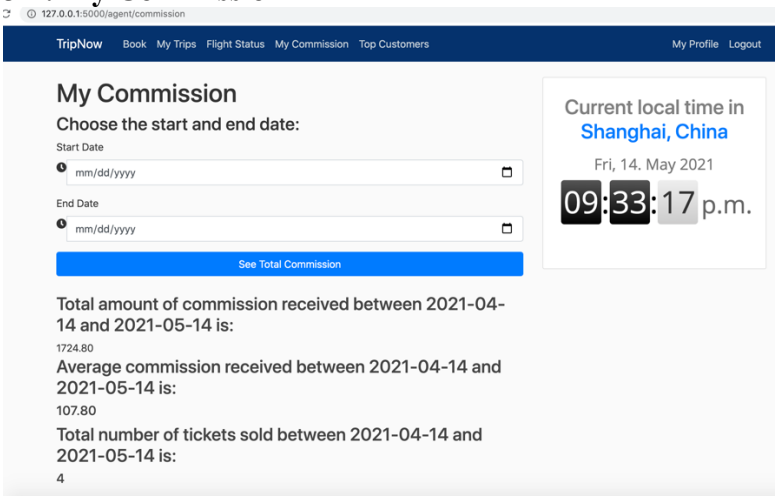
Similar to customer book. Should also enter the email address for customer. Codes under bookagent() and bookagentconfirm().

```
else:
    a = 0
    while a == 0:
        tkt_id = randint(100000000, 999999999)
        query = "SELECT * FROM ticket WHERE ticket_id = \'{}\'"
        cursor.execute(query.format(tkt_id))
        data = cursor.fetchone()
        if not data:
            a = 1
    try:
        ins1 = "INSERT INTO ticket VALUES(\'{}\', \'{}\', \'{}\')"
        cursor.execute(ins1.format(tkt_id, airline, flight_num))

        today = datetime.today().strftime('%Y-%m-%d')

        ins2 = "INSERT INTO purchases VALUES(\'{}\', \'{}\', \'{}\', \'{}\')"
        cursor.execute(ins2.format(tkt_id, cus_email, book_id, today))
        conn.commit()
        cursor.close()

        msg = "You have successfully booked the flight! " + airline + " " + fligh
        msg1 = "Ticket ID: " + str(tkt_id)
        #flash("Booking Successful! Flight:" + airline + flight_num)
        return render_template('bookagentconfirm.html', msg=msg, msg1=msg1)
    except NameError:
        redirect(url_for('bookagent'))
```

3-3. Flight Status

Similar to public flight status. Codes under statusagent().

3-4. My Commission

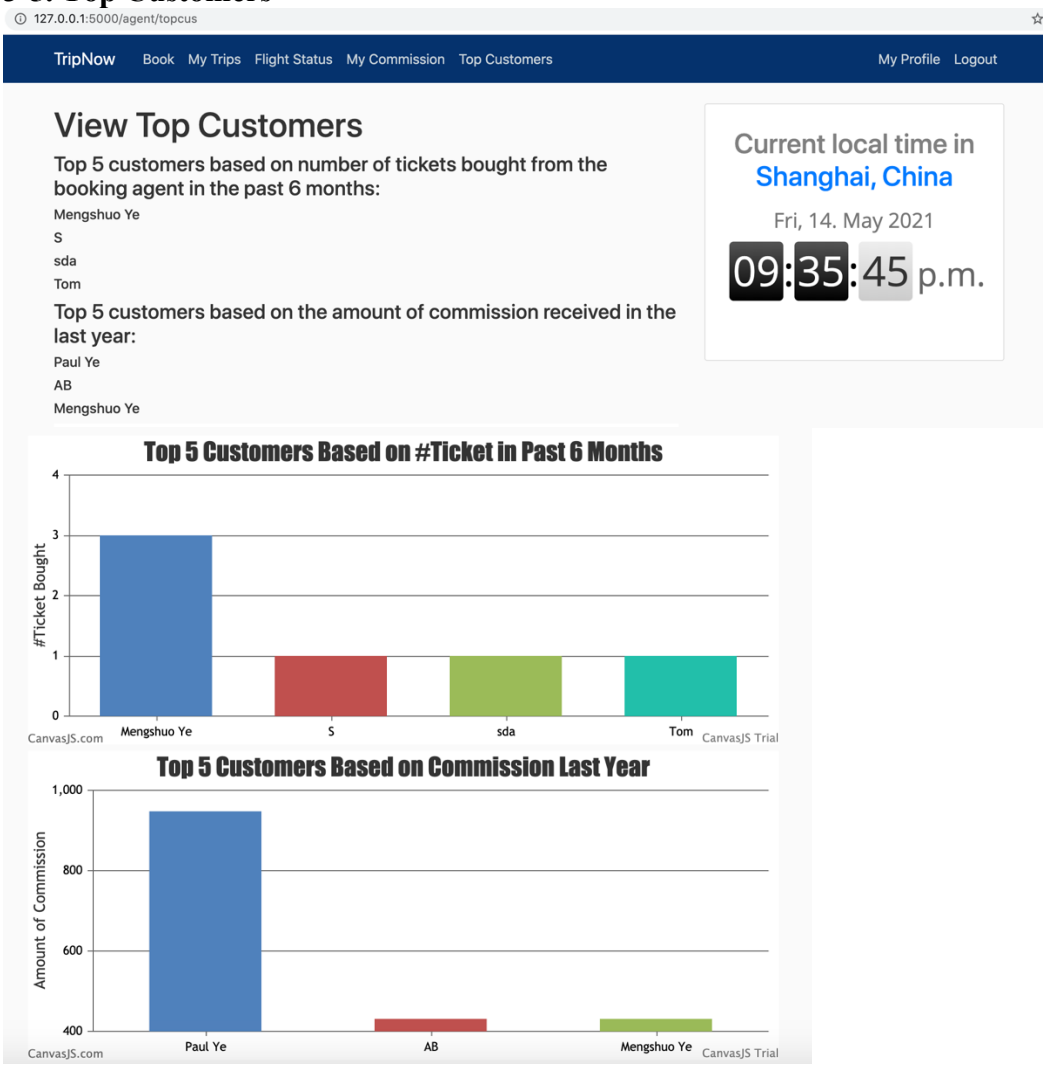


Codes under commissionagent():

Default view and advanced search:

```
q = 'SELECT CAST(0.1*SUM(price) as decimal(10,2)), CAST((0.1*price)/COUNT(*) as decimal(10,2)), COUNT(*) \
    FROM purchases NATURAL JOIN ticket NATURAL JOIN flight \
WHERE booking_agent_id = \'{}\' AND purchase_date <= \'{}\' AND purchase_date >= \'{}\''
cursor.execute(q.format(booking_agent_id, end_date, start_date))
commission = cursor.fetchone()

if request.method == 'POST':
    start_date = request.form['start date']
    end_date = request.form['end date']

    q = 'SELECT CAST(0.1*SUM(price) as decimal(10,2)), CAST((0.1*price)/COUNT(*) as decimal(10,2)), COUNT(*) \
        FROM purchases NATURAL JOIN ticket NATURAL JOIN flight \
    WHERE booking_agent_id = \'{}\' AND purchase_date <= \'{}\' AND purchase_date >= \'{}\''
    cursor.execute(q.format(booking_agent_id, end_date, start_date))
    commission = cursor.fetchone()
```

3-5. Top Customers



Queries under topcus():

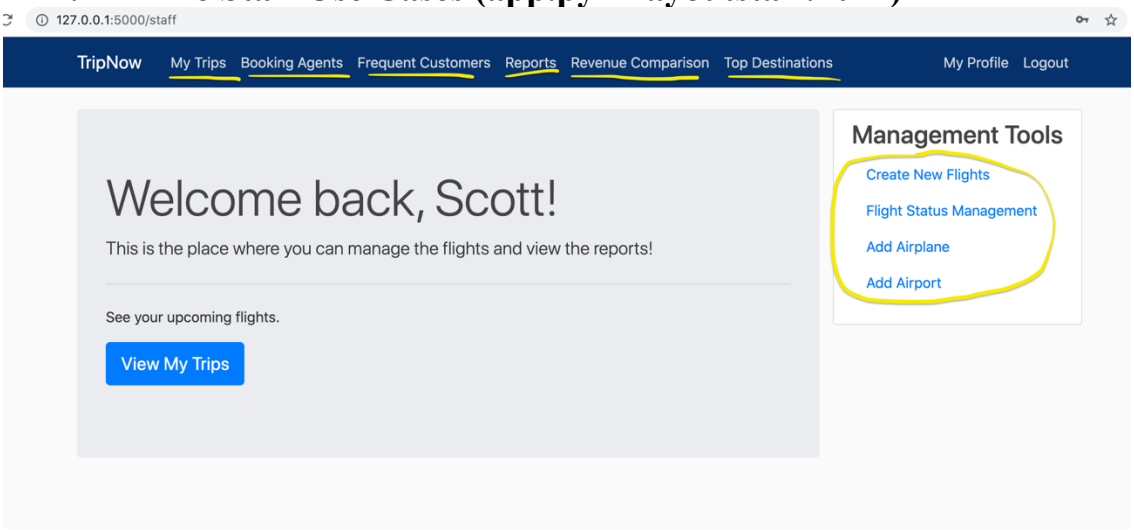
```
q1 = "SELECT name, count(ticket_id) FROM purchases JOIN customer ON (customer.email = purchases.customer_email)\
            WHERE booking_agent_id = \'{}\'\
            AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'\
            GROUP BY customer_email\
            ORDER BY COUNT(ticket_id) DESC LIMIT 5"
```

```
        q2 = "SELECT name, CAST(0.1*SUM(price) as decimal(10,2))\
            FROM (purchases NATURAL JOIN ticket NATURAL JOIN flight) JOIN customer ON (customer.email =
customer_email)\
            WHERE booking_agent_id = \'{}\' AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'\
            GROUP BY customer_email\
            ORDER BY COUNT(ticket_id) DESC LIMIT 5"
```

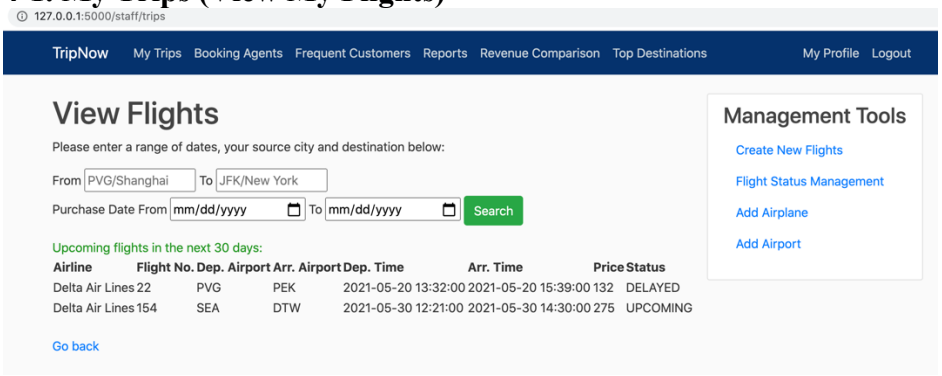
3-6. Logout: logout()

3-7. My Profile: Similar to customer profile. Queries under profileagent().

4. Airline Staff Use Cases (app.py + layoutstaff.html)



4-1. My Trips (View My Flights)



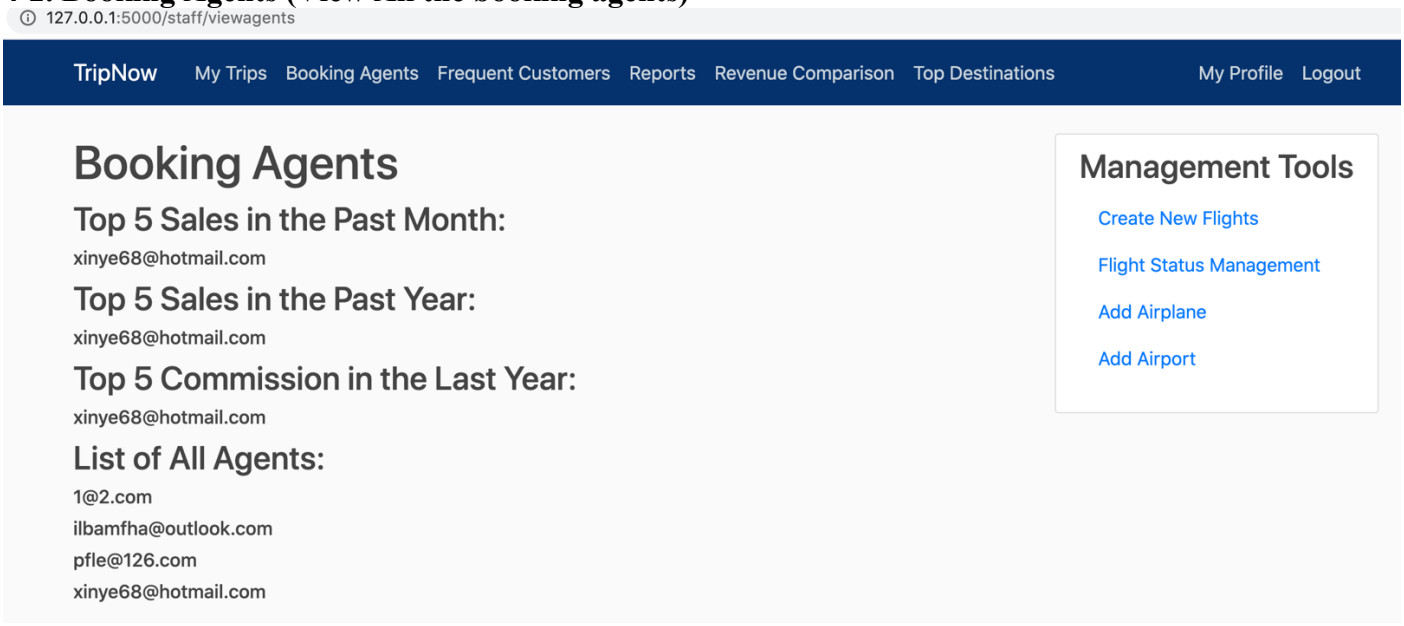
Similar to customer and booking agent. Default 30 days view. Queries under tripsstaff().

```
if 'username' in session and 'airline name' in session:
    cursor = conn.cursor()
    airline_name = session['airline name']
    #default time is showing all the upcoming flights
    start_date = datetime.today()
    end_date = start_date + timedelta(days=30)
    msg = ''
    condition = ''

    q = 'SELECT airline_name, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, price, status \
        FROM flight WHERE departure_time >= \'{}\' AND departure_time <= \'{}\' AND airline_name = \'{}\''
    cursor.execute(q.format(start_date, end_date, airline_name))
    t = cursor.fetchall()
```

Advanced Search:

```
request.method == 'POST':
 start_date = request.form['start date']
 end_date = request.form['end date']
 departure = request.form['departure']
 arrival = request.form['arrival']

 if departure.isupper():
    cursor = conn.cursor()
    q = 'SELECT f.airline_name, f.flight_num, f.departure_airport, f.arrival_airport, f.departure_time, f.arrival_time, f.price, f.status \
        FROM flight f \
        WHERE f.departure_time >= \'{}\' AND f.departure_time <= \'{}\' \
        AND f.departure_airport = \'{}\' AND f.arrival_airport = \'{}\' AND f.airline_name = \'{}\' '
    cursor.execute(q.format(start_date, end_date, departure, arrival, airline_name))
    t = cursor.fetchall()
 else:
    cursor = conn.cursor()
    q = 'SELECT f.airline_name, f.flight_num, f.departure_airport, f.arrival_airport, f.departure_time, f.arrival_time, f.price, f.status \
        FROM flight f JOIN airport a ON(f.departure_airport = a.airport_name) \
        JOIN airport b ON(f.arrival_airport = b.airport_name) \
        WHERE f.departure_time >= \'{}\' AND f.departure_time <= \'{}\' \
        AND a.airport_city = \'{}\' AND b.airport_city = \'{}\' AND f.airline_name = \'{}\''
    cursor.execute(q.format(start_date, end_date, departure, arrival, airline_name))
    t = cursor.fetchall()
```

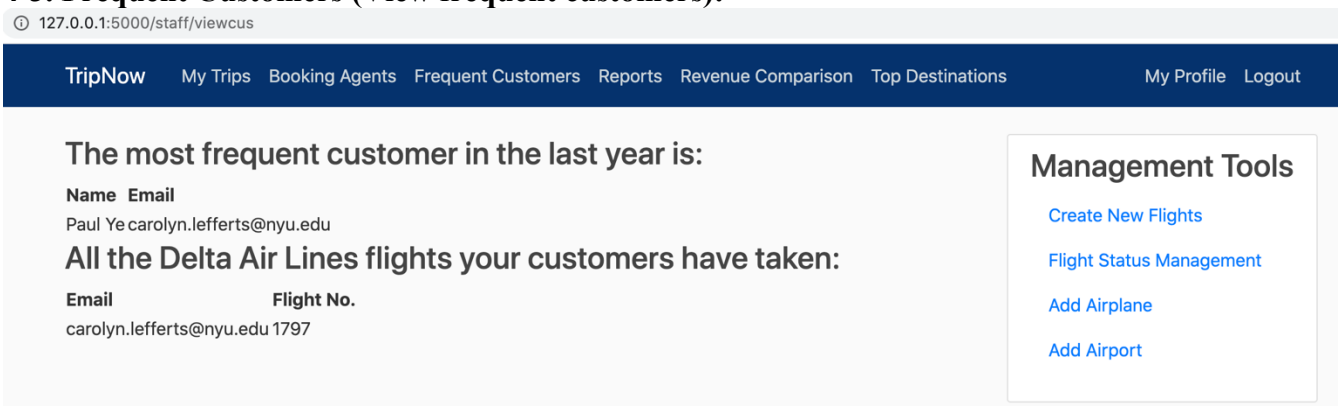
4-2. Booking Agents (View All the booking agents)



Queries under viewagents():

```
889  @app.route('/staff/viewagents')
890  def viewagents():
891   if 'username' in session and 'airline name' in session:
892    end_date = date.today()
893    start_date_month = end_date-timedelta(days=30)
894    start_date_year = end_date-timedelta(days=365)
895    end_date_last = start_date_year.strftime("%Y")+'-12-31'
896    start_date_last = start_date_year.strftime("%Y")+'-01-01'
897
898    cursor = conn.cursor()
899    q = 'SELECT email FROM purchases NATURAL JOIN booking_agent \
900     WHERE purchase_date <= \'{}\' AND purchase_date >= \'{}\' \
901     GROUP BY booking_agent_id ORDER BY COUNT(*) DESC LIMIT 5'
902    cursor.execute(q.format(end_date, start_date_month))
903    s = cursor.fetchall()
904
905
906    q = 'SELECT email FROM purchases NATURAL JOIN booking_agent \
907     WHERE purchase_date <= \'{}\' AND purchase_date >= \'{}\' \
908     GROUP BY booking_agent_id ORDER BY COUNT(*) DESC LIMIT 5'
909    cursor.execute(q.format(end_date, start_date_year))
910    y = cursor.fetchall()
911
912    q = 'SELECT email FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN booking_agent \
913     WHERE purchase_date <= \'{}\' AND purchase_date >= \'{}\' \
914     GROUP BY booking_agent_id ORDER BY SUM(price) DESC LIMIT 5'
915    cursor.execute(q.format(end_date_last, start_date_last))
916    c = cursor.fetchall()
917
918    q = 'SELECT email FROM booking_agent'
919    cursor.execute(q)
920    t = cursor.fetchall()
921
922    return render_template('viewagents.html', s = s, y = y, c = c, t = t)
923   return redirect(url_for('prelogin'))
```

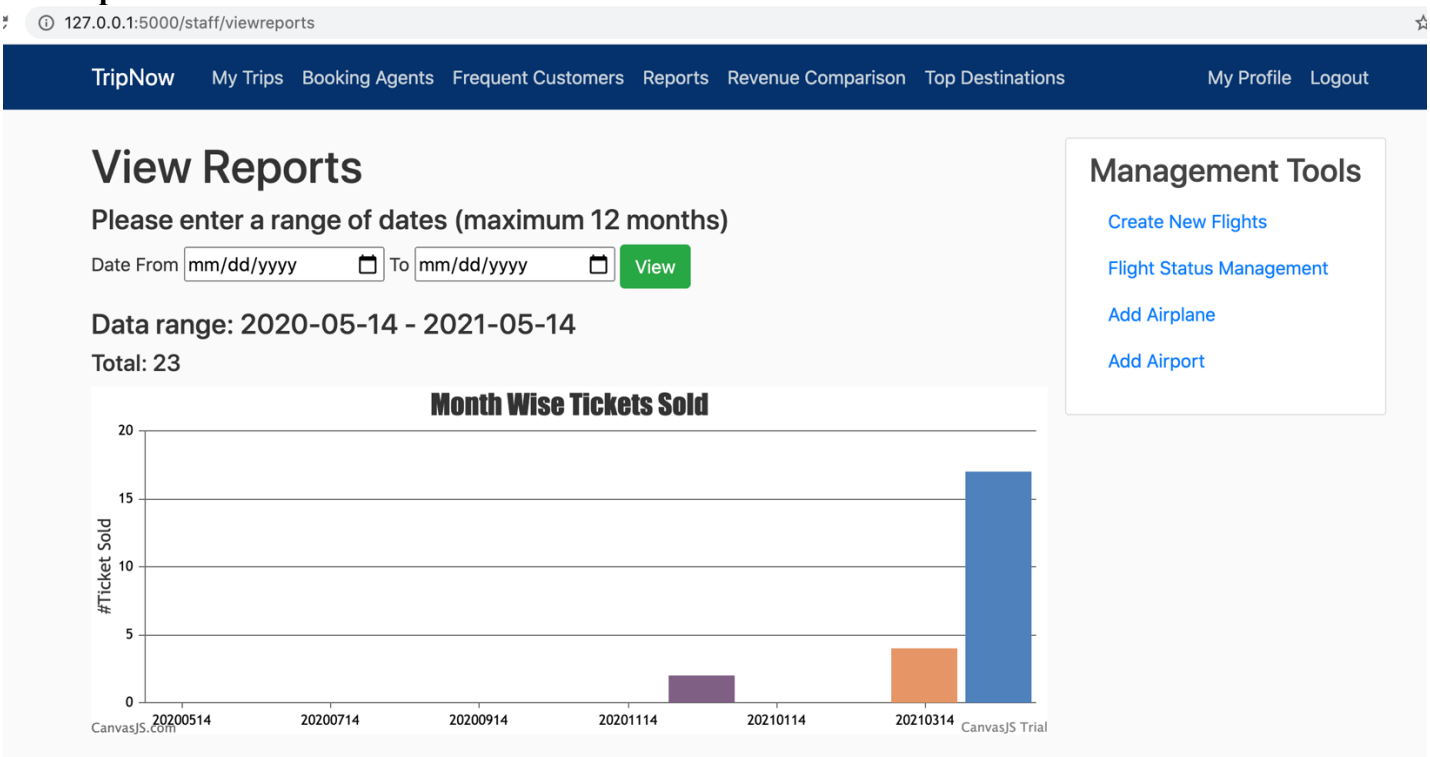
4-3. Frequent Customers (View frequent customers):



Queries under viewcus():

```
q = 'SELECT name, email FROM customer JOIN purchases ON (customer.email = purchases.customer_email) \
WHERE purchase_date <= \'{}\' AND purchase_date >= \'{}\' \
GROUP BY email ORDER BY COUNT(ticket_id) DESC'
cursor.execute(q.format(end_date_last, start_date_last))
cus = cursor.fetchone()

if cus is None:
    msg1 = 'No customer bought ticket last year!'

q = 'SELECT DISTINCT customer_email, flight_num FROM flight NATURAL JOIN ticket NATURAL JOIN purchases \
WHERE airline_name = \'{}\' AND departure_time <= \'{}\' GROUP BY customer_email'
cursor.execute(q.format(airline_name, taken_time))
lst = cursor.fetchall()
```

4-4. Reports



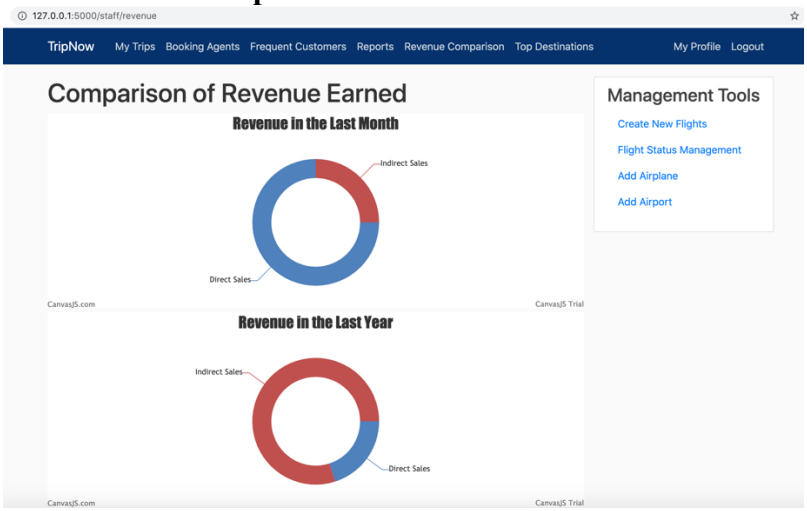
Queries under viewreports():
Default View 12 months

```
q = "SELECT COUNT(ticket_id) FROM purchases NATURAL JOIN ticket\
    WHERE airline_name = \'{}\'\
    AND purchase_date > \'{}\' AND purchase_date <= \'{}\'"

cursor.execute(q.format(airline, curdate, curdate1))
result.append(cursor.fetchone()[0])
```

Customized date range

```
while curdate <= date1:
    q = "SELECT COUNT(ticket_id) FROM purchases NATURAL JOIN ticket\
    WHERE airline_name = \'{}\'\
    AND purchase_date > \'{}\' AND purchase_date <= \'{}\'"
    cursor.execute(q.format(airline, str(curdate)[0:10], str(curdate1)[0:10]))
    result_temp.append(cursor.fetchone()[0])
```

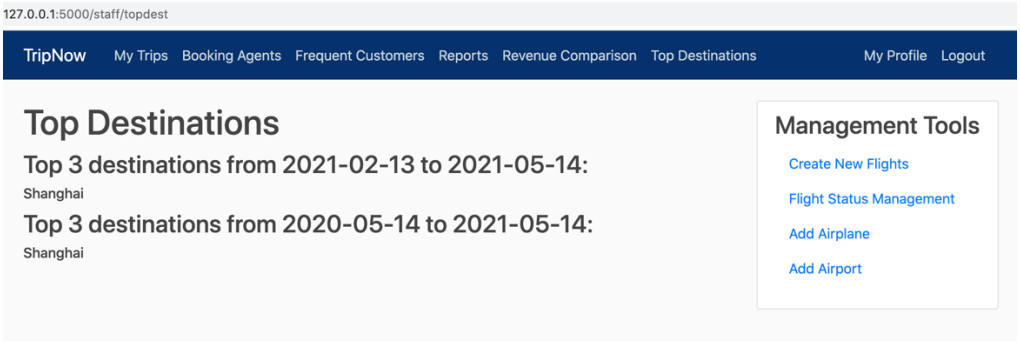
4-5. Revenue Comparison



Queries under revenue(): q1 q2 for last month, q3 q4 for last year

```
q1 = "SELECT count(ticket_id) FROM purchases NATURAL JOIN ticket\
    WHERE booking_agent_id is NULL\
    AND airline_name = \'{}\'\
    AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'"

cursor.execute(q1.format(airline, end_date_last_month, start_date_last_month))
out1 = cursor.fetchall()

q2 = "SELECT count(ticket_id) FROM purchases NATURAL JOIN ticket\
    WHERE booking_agent_id is not NULL\
    AND airline_name = \'{}\'\
    AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'"

cursor.execute(q2.format(airline, end_date_last_month, start_date_last_month))
out2 = cursor.fetchall()

num1["Direct Sales"] = out1[0][0]
num1["Indirect Sales"] = out2[0][0]

q3 = "SELECT count(ticket_id) FROM purchases NATURAL JOIN ticket\
    WHERE booking_agent_id is NULL\
    AND airline_name = \'{}\'\
    AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'"

cursor.execute(q3.format(airline, end_date_last_year, start_date_last_year))
out3 = cursor.fetchall()

q4 = "SELECT count(ticket_id) FROM purchases NATURAL JOIN ticket\
    WHERE booking_agent_id is NOT NULL\
    AND airline_name = \'{}\'\
    AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'"

cursor.execute(q4.format(airline, end_date_last_year, start_date_last_year))
out4 = cursor.fetchall()

num2["Direct Sales"] = out3[0][0]
num2["Indirect Sales"] = out4[0][0]
```

4-6. Top Destinations

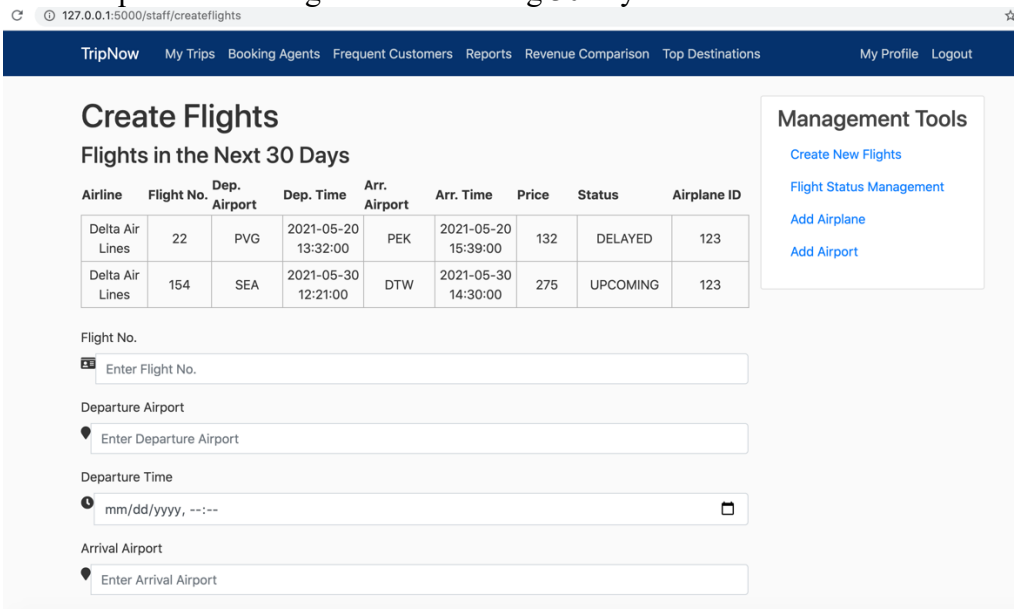


Queries under topdest():

```
q = 'SELECT airport_city FROM flight JOIN airport ON (flight.arrival_airport = airport.airport_name) \
    NATURAL JOIN ticket NATURAL JOIN purchases \
 WHERE purchase_date <= \'{}\' AND purchase_date >= \'{}\' \
 GROUP BY airport_city ORDER BY COUNT(*) DESC LIMIT 3'
cursor.execute(q.format(end_date, start_date_year))
year = cursor.fetchall()

q = 'SELECT airport_city FROM flight JOIN airport ON (flight.arrival_airport = airport.airport_name) \
    NATURAL JOIN ticket NATURAL JOIN purchases \
 WHERE purchase_date <= \'{}\' AND purchase_date >= \'{}\' \
 GROUP BY airport_city ORDER BY COUNT(*) DESC LIMIT 3'
cursor.execute(q.format(end_date, start_date_3month))
month = cursor.fetchall()
```

4-7. Create New Flights

You can preview the flights in the coming 30 days.



Queries under createflights():

Preview coming flights:

```
q = 'SELECT * \
    FROM flight WHERE departure_time >= \'{}\' AND departure_time <= \'{}\' AND airline_name = \'{}\''
cursor.execute(q.format(start_date, end_date, airline_name))
data = cursor.fetchall()
```

Insert after error checking:

```
    msg = 'Number of seats should not exceed 11 characters!'
else:
    # Account doesnt exists and the form data is valid, now insert new account into accounts table
    ins = "INSERT INTO flight VALUES( \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\', \'{}\')"
    cursor.execute(ins.format(airline_name, flight_num, departure_airport, departure_time, arrival_airport, \
        arrival_time, price, status, airplane_id))
    conn.commit()
    msg = 'You have successfully added the flight!'
```

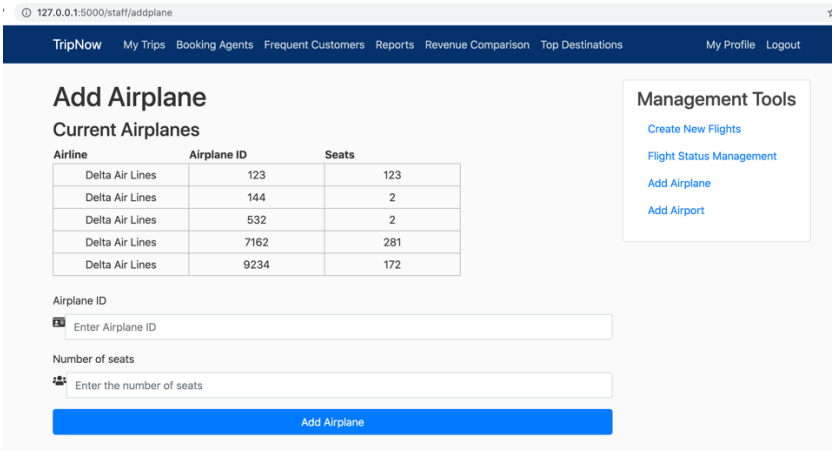
4-8. Flight Status Management (Change status)

Queries under statuschange():

Change status while displaying all the flight status

```
cursor.execute("SELECT * FROM flight")
data = cursor.fetchall()

if request.method == 'POST':
    airline_name = session['airline name']
    flight_num = request.form['flight num']
    changed_status = request.form['status']

    query1 = "SELECT status FROM flight WHERE airline_name = \'{}\' AND flight_num = \'{}\'"
    cursor.execute(query1.format(airline_name, flight_num))
    current_status = cursor.fetchone()

    if (not current_status):
        msg = 'This flight does not exist!'
    elif (current_status[0] == changed_status):
        msg = 'The status does not need to be changed!'
    else:
        up = "UPDATE flight SET status = \'{}\' WHERE airline_name = \'{}\' AND flight_num = \'{}\'"
        cursor.execute(up.format(changed_status, airline_name, flight_num))
        conn.commit()
        msg = 'You have successfully changed the status!'

    cursor.execute("SELECT * FROM flight")
    data = cursor.fetchall()
```

4-9. Add Airplane

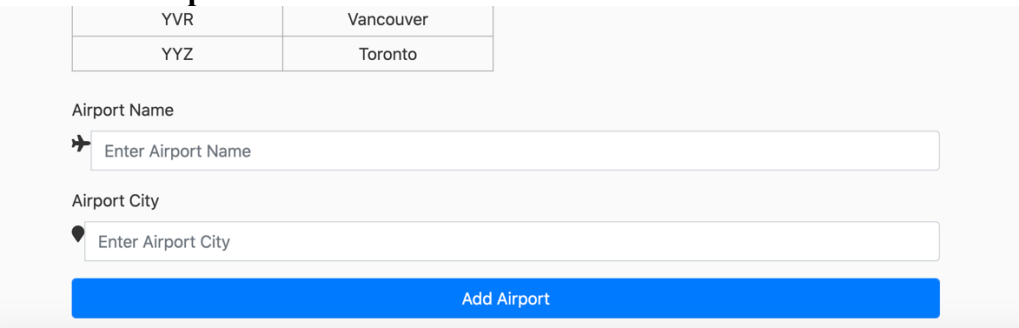


Queries under addplane():

Add plane while displaying all the existing planes

```
cursor.execute("SELECT * FROM airplane")
data = cursor.fetchall()
if request.method == 'POST':
    # Create variables for easy access
    airline_name = session['airline name']
    airplane_id = request.form['airplane id']
    num_of_seats = request.form['seats']

    #Check if airplane exists using MySQL
    query1 = "SELECT * FROM airplane WHERE airline_name = \'{}\' AND airplane_id = \'{}\'"
    cursor.execute(query1.format(airline_name, airplane_id))
    airplane = cursor.fetchone()

    query2 = "SELECT * FROM airline WHERE airline_name = \'{}\'"
    cursor.execute(query2.format(airline_name))
    airline = cursor.fetchone()
    # If airplane exists show error and validation checks
    if airplane:
        msg = 'This airplane already exists!'
    elif (not airline):
        msg = 'This airline does not exist!'
    # elif len(airline_name) >= 50:
    #    msg = 'The Airline Name should be less than 50 characters!'
    elif len(airplane_id) > 11:
        msg = 'The Airplane ID should not exceed 11 characters!'
    #type changed to varchar(11)
    elif len(num_of_seats) > 11:
        msg = 'Number of seats should not exceed 11 characters!'

    else:
        # Account doesnt exists and the form data is valid, now insert new account into accounts table
        ins = "INSERT INTO airplane VALUES( \'{}\', \'{}\', \'{}\')"
        cursor.execute(ins.format(airline_name, airplane_id, num_of_seats))
        conn.commit()
        msg = 'You have successfully added the airplane!'

    cursor.execute("SELECT * FROM airplane")
    data = cursor.fetchall()
```

4-10. Add Airport



Queries under addairport():
Add airport while displaying all the existing airports

```
cursor.execute("SELECT * FROM airport")
data = cursor.fetchall()
if request.method == 'POST':

    # Create variables for easy access
    airport_name = request.form['airport name']
    airport_city = request.form['airport city']

    #Check if airport_name exists using MySQL
    query1 = "SELECT * FROM airport WHERE airport_name = \'{}\'"
    cursor.execute(query1.format(airport_name))
    airport = cursor.fetchone()

    # If airplane exists show error and validation checks
    if airport:
        msg = 'This airport already exists!'
    elif len(airport_name) >= 50:
        msg = 'The Airport Name should be less than 50 characters!'
    elif len(airport_city) >= 50:
        msg = 'The Airport City Name should be less than 50 characters!'
    else:
        # Account doesnt exists and the form data is valid, now insert new account into accounts table
        ins = "INSERT INTO airport VALUES( \'{}\', \'{}\')"
        cursor.execute(ins.format(airport_name, airport_city))
        conn.commit()
        msg = 'You have successfully added the airport!'

    cursor.execute("SELECT * FROM airport")
    data = cursor.fetchall()
```

4-11. My Profile

Similar to booking agent and customer. Queries under profilestaff().

4-12. Logout: logout()