Databases Project

Group Members:

- Andrew Asseily aa7342@nyu.edu
- Leisha Murthy lm4684@nyu.edu
- Gabe Lora gl1367@nyu.edu

Use cases:

All the use cases prevent unauthorized users from performing specific tasks successfully.

Customer:

1. View My flights: Provide various ways for the user to see flight information which he/she purchased. The default should be showing for the future flights. Optionally you may include a way for the user to specify a range of dates, specify destination and/or source airport name or city name etc.

```python
@app.route('/customer/view/myflights')
def customerViewmyFlights(msg=None):
    customer = cus_check_session()
    if customer:
        cursor = conn.cursor()
        query = 'SELECT * FROM Flight, buys WHERE buys.flight_num = flight.flight_num and buys.email = %s'
        cursor.execute(query, (customer[0]['email']))
        data = cursor.fetchall()
        cursor.close()
        return render_template('cus-view-flight.html',data=data,msg=msg)
    return redirect('/login/customer')
```

2. Search for flights: Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or return).

```python
@app.route('/searchFlights', methods=['GET', 'POST'])
def flightSearch():
    deptairport = request.form['deptairport']
    arrairport = request.form['arrairport']
    deptdate = request.form['deptdate']
    current_date = get_format_date()
    cursor = conn.cursor()

    query = 'SELECT  f.stats, f.flight_num, f.dept_airport, f.arr_airport, f.dept_date, f.arr_date FROM flight as f, manage as m WHERE (f.dept_date = %s OR f.dept_date >= %s) and f.dept_date >= %s and f.dept_airport = (SELECT name_airport FROM airport WHER
    cursor.execute(query, (
        deptdate,
        deptdate,
        current_date,
        deptairport,
        deptairport,
        arrairport,
        arrairport
    ))
    flight = cursor.fetchall()
    cursor.close()
    return render_template('flights.html', flight=flight)
```

.

3. Purchase tickets: Customer chooses a flight and purchases a ticket for this flight, providing all the needed data, via forms. You may find it easier to implement this along with a use case to search for flights.

```python
@app.route('/purchaseTicket', methods=['GET', 'POST'])
def purchaseTicket():
    return render_template('purchase-ticket.html')

@app.route('/purchaseTicketsAuth', methods=['GET', 'POST'])
def purchaseTicketAuth():
    cursor = conn.cursor()
    cus_email = request.form['cus_email']
    cus_ticket_id = request.form['cus_ticket_id']
    deptdate = request.form['deptdate']
    depttime = request.form['depttime']
    flightnum = request.form['flightnum']
    price = request.form['price']
    cardtype = request.form['cardtype']
    cardnumber = request.form['cardnumber']
    nameoncard = request.form['nameoncard']
    expmonth = request.form['expmonth']
    expyear = request.form['expyear']
    ins = 'INSERT INTO buys(`email`, `ticket_id`, `dept_date`, `dept_time`, `flight_num`, `sold_price`, `card_type`, `card_num`, `name_on_card`, `exp_month`, `exp_year`) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)'
    cursor.execute(ins, (
        cus_email,
        cus_ticket_id,
        deptdate,
        depttime,
        flightnum,
        price,
        cardtype,
        cardnumber,
        nameoncard,
        expmonth,
        expyear
        ))
    conn.commit()
    data = cursor.fetchall()
    cursor.close()
    return render_template('flights.html',buys = data)
```

4. Cancel Trip: Customer chooses a purchased ticket for a flight that will take place more than 24 hours in the future and cancels the purchase. After cancellation, the ticket will no longer belong to the customer. The ticket will be available again in the system and purchasable by other customers.

```python
@app.route('/cusCancelFlight', methods=['GET', 'POST'])
def cusCancelFlight():
    currentdate = get_date()
    customer = cus_check_session()
    if customer:
        flightnum = request.form['flight_num']
        cur = conn.cursor()
        query = 'SELECT ticket_id, dept_date, dept_time FROM buys WHERE flight_num = %s and email = %s'
        cur.execute(query, (
            flightnum,
            customer[0]['email']
        ))
        data = cur.fetchall()
        ticketid = data[0]['ticket_id']
        deptdate = data[0]['dept_date']
        depttime = data[0]['dept_time']
        year = deptdate.year
        month = deptdate.month
        day = deptdate.day
        strtime = str(depttime)
        mylist = strtime.split(':')
        hour = int(mylist[0])
        minute = int(mylist[1])
        d1 = datetime(year,month,day,hour,minute,0)
        d2 = datetime.strptime(currentdate, "%d/%m/%Y %H:%M:%S")
        delta = d1-d2
        seconds = delta.total_seconds()
        hours = seconds // 3600
        error = None
#ERR MSGS PRINT IN TERMINAL BUT NOT ON PAGE
        if hours < 24:
            error =  'You cannot cancel your flight'
            print(error)
            return customerViewmyFlights(error)
        else:
            success = 'your flight has been cancelled'
            cancel = True
            cursor = conn.cursor()
            query = 'DELETE FROM buys WHERE ticket_id = %s and dept_date = %s and dept_time = %s'
            cursor.execute(query, (ticketid,deptdate,depttime))
            print(success)
            seat = 'UPDATE `manage` SET `total_seats`= total_seats - 1 WHERE total_seats > 0'
            cursor.execute(seat)
            conn.commit()
            cur.close()
            print('the seat has been removed')
            return customerViewmyFlights(success)
    return redirect('/login/customer')
```

5. Give Ratings and Comment on previous flights: Customers will be able to rate and comment on their previous flights (for which he/she purchased tickets and already took that flight) for the airline they logged in.

```
@app.route('/customer/review')
def reviews():
    name = cus_check_session()
    date = get_format_date()
    print(date)
    if (name):
        cursor = conn.cursor()
        query = 'SELECT flight_num, ticket_id, dept_date FROM buys where dept_date <= %s and email = %s'
        cursor.execute(query, (
            date,
            name[0]['email']
        ))
        data = cursor.fetchall()
        cursor.close()
        return render_template('cus-review.html', data=data)
    return redirect('/login/customer')
@app.route('/customer/addReview',methods=['GET', 'POST'])
def addReview():
    customer = cus_check_session()
    if (customer):
        ticket = request.form['ticket_id']
        rating = request.form['rating']
        comment = request.form['comment']
        cursor = conn.cursor()
        query = 'UPDATE buys set ratings=%s, comments=%s WHERE ticket_id =%s and email = %s'
        cursor.execute(query, (
            rating,
            comment,
            ticket,
            customer[0]['email']
        ))
        conn.commit()
        cursor.close()
        return redirect('/customer/home')
    return redirect('/login/customer')
```

6.Track My Spending: Default view will be the total amount of money spent in the past year and a bar chart/table showing month wise money spent for the last 6 months. He/she will also have the option to specify a range of dates to view the total amount of money spent within that range and a bar chart/table showing month wise money spent within that range.

```
@app.route('/customer/spending')
def viewSpending():
    customer = cus_check_session()
    if (customer):
        cursor = conn.cursor()
        query = 'SELECT sum(sold_price) as total, MONTH(purchase_timestamp) as m, YEAR(purchase_timestamp) as y FROM `buys` WHERE email = %s GROUP BY m,y'
        cursor.execute(query, customer[0]['email'])
        data = cursor.fetchall()

        labels = [(str(line['m']) + '/'+ str(line['y'])) for line in data]
        values = [float(line['total']) for line in data]
        values.append(0)
        cursor.close()
        return render_template('cus-track-spending.html', total=data, labels = labels, values=values)
    return redirect('/login/customer')
@app.route('/customer/spending/search', methods=['GET', 'POST'])
```

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

```
@app.route('/logout/customer')
def logout_customer():
    session.pop('email')
    return redirect('/')
```

Staff:

1. View flights: Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days. He/she will be able to see all the current/future/past flights operated by the airline he/she works for based on the range of dates, source/destination airports/city etc. He/she will be able to see all the customers of a particular flight.

```
@app.route('/staff/viewflights')
def staffViewFlights():
    air_data = staff_check_session()
    if (air_data):
        cursor = conn.cursor()
        query = 'SELECT f.dept_date, f.dept_time, f.flight_num, f.arr_time, f.arr_airport, f.arr_date, f.base_price, f.id_airplane, f.dept_airport, f.stats FROM flight as f, manage as m, air_staff as s WHERE
        cursor.execute(query)
        data = cursor.fetchall()
        cursor.close()
        return render_template('staff-view-flights.html', flight = data, airline_name=air_data)
```

```
WHERE m.airline_name = s.airline_name AND m.flight_num = f.flight_num'
```

2. Create new flights: He or she creates a new flight, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days.

```python
@app.route('/addFlightAuth', methods=['GET', 'POST'])
def CreateNewFligh Atuth():
    staff = staff_check_session()
    if staff:
        deptdate = request.form['deptdate']
        depttime = request.form['depttime']
        flightnum = request.form['flightnum']
        arrtime = request.form['arrtime']
        arrairport = request.form['arrairport']
        arrdate = request.form['arrdate']
        baseprice = request.form['baseprice']
        idairplane = request.form['idairplane']
        deptairport = request.form['deptairport']
        states = request.form['states']
        cursor = conn.cursor()
        ins = 'INSERT INTO flight(`dept_date`, `dept_time`, `flight_num`, `arr_time`,`arr_airport`, `arr_date`, `base_price`, `id_airplane`, `dept_airport`, `stats`) VALUES( %s, %s, %s, %s,%s, %s,%s
        cursor.execute(ins, (
            deptdate,
            depttime,
            flightnum,
            arrtime,
            arrairport,
            arrdate,
            baseprice,
            idairplane,
            deptairport,
            states
            ))
        conn.commit()
        cur = conn.cursor()

        data = flight_helper(idairplane)
        tracks = 'INSERT INTO manage(`airline_name`, `dept_date`, `dept_time`, `flight_num`, `total_seats`, `max_seats`) VALUES (%s,%s,%s,%s, %s, %s)'
        cur.execute(tracks,(
            staff[0]['airline_name'],
            deptdate,
            depttime,
            flightnum,
            0,
            data[0]['num_of_seats'],

        ))
        conn.commit()
        cur.close()
        cursor.close()
        return redirect('/staff/home')
    return redirect('/login/staff')
```

3. Change Status of flights: He or she changes a flight status (from on-time to delayed or vice versa) via forms.

```python
@app.route('/changeStatus', methods=['GET', 'POST'])
def changeFlightStatus():
    airline = staff_check_session()
    if airline:
        cursor = conn.cursor()
        query = 'SELECT m.flight_num FROM manage as m, flight as f WHERE f.flight_num = m.flight_num and m.airline_name = %s'
        cursor.execute(query, airline[0]['airline_name'])
        data = cursor.fetchall()
        cursor.close()
        return render_template('changeFlightStatus.html', flight = data)
    return redirect('/staff/home')

@app.route('/changeStatusAuth', methods=['GET', 'POST'])
def changeFlightStatusAuth():
    if (staff_check_session):
        flightnum = request.form['flightNum']
    #airplaneid = request.form['airplaneid']
        status = request.form['status']
        cursor = conn.cursor()
        ins = 'UPDATE flight SET stats = %s WHERE flight_num = %s'# and id_airplane = %s'
        cursor.execute(ins, (
            status,
            flightnum,
        #airplaneid
            ))
        conn.commit()

        cursor.close()
```

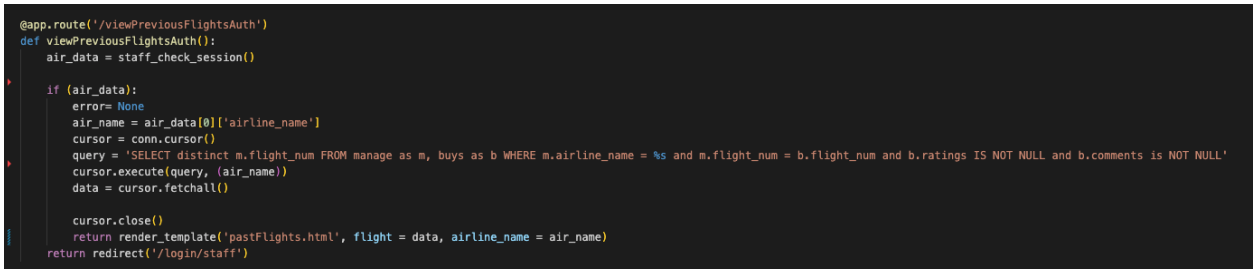
4. Add airplane in the system: He or she adds a new airplane, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. In the confirmation page, she/he will be able to see all the airplanes owned by the airline he/she works for.

```python
@app.route('/addAirplane', methods=['GET', 'POST'])
def addAirplane():
    if (staff_check_session):
        return render_template('addAirplane.html')
    return redirect('/login/staff')

@app.route('/addAirplaneAuth', methods=['GET', 'POST'])
def addAirplaneAuth():
    airline = staff_check_session()
    if (airline):
        ID = request.form['idAirplane']
        numSeats = request.form['numSeats']
        manufact = request.form['manuCompany']
        age = request.form['age']
        cursor = conn.cursor()
        query = 'INSERT INTO `airplane`(`id_airplane`, `airline_name`, `num_of_seats`, `manufacturing_company`, `age`) VALUES (%s,%s,%s,%s,%s)'
        cursor.execute(query, (
            ID,
            airline[0]['airline_name'],
            numSeats,
            manufact,
            age
        ))
        conn.commit()
        cursor.close()
        return redirect('/viewAirplane')
    return redirect('/login/staff')
```

5. 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.

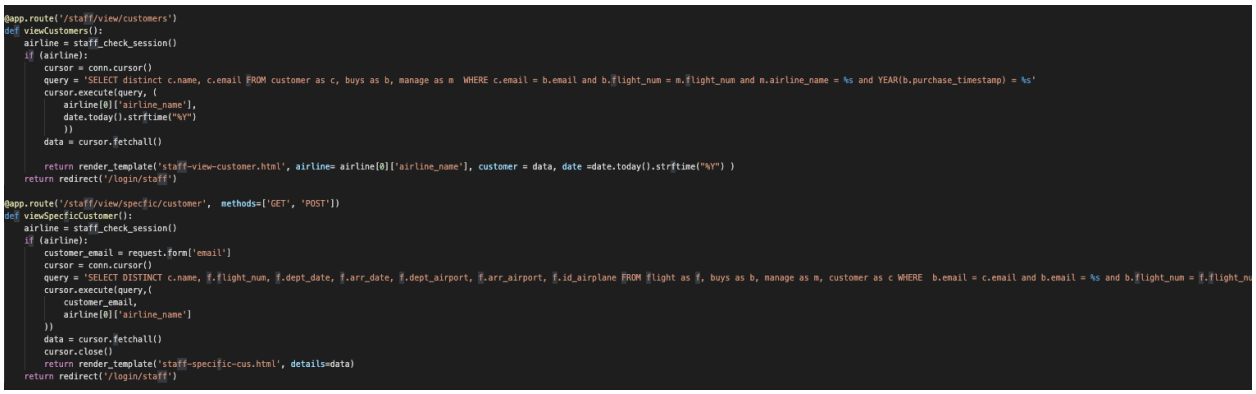
```python
@app.route('/addAirport', methods=['GET', 'POST'])
def addAirport():
    if (staff_check_session):
        return render_template('addAirport.html')
    return redirect('/login/staff')

@app.route('/addAirportAuth', methods=['GET', 'POST'])
def addAirportAuth():
    airplane = staff_check_session()
    if (airplane):
        airportName = request.form['airportName']
        city = request.form['city']
        country = request.form['country']
        airport_type =  request.form['airportType']
        cursor = conn.cursor()
        check = 'SELECT * FROM airport WHERE name_airport = %s'
        cursor.execute(check, airportName)
        data = cursor.fetchall()
        query = 'INSERT INTO `airport`(`name_airport`, `city`, `country`, `airport_type`) VALUES (%s,%s,%s,%s)'
        error = None
        if (data):
            error = "Airport already exists"
            return render_template('addAirport.html', error=error)
        else:
            cursor.execute(query, (
                airportName,
                city,
                country,
                airport_type
            ))
            conn.commit()
            cursor.close()
            return redirect('/staff/home')
    return redirect('/login/staff')
```

6. View flight ratings: Airline Staff will be able to see each flight's average ratings and all the comments and ratings of that flight given by the customers.



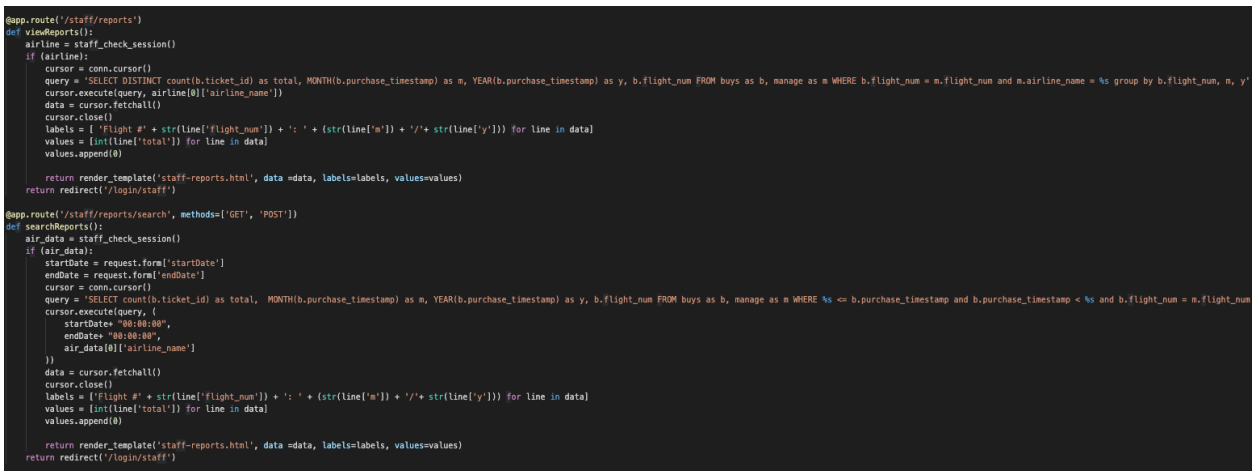
```
@app.route('/viewPreviousFlightsAuth')
def viewPreviousFlightsAuth():
    air_data = staff_check_session()

    if (air_data):
        error= None
        air_name = air_data[0]['airline_name']
        cursor = conn.cursor()
        query = 'SELECT distinct m.flight_num FROM manage as m, buys as b WHERE m.airline_name = %s and m.flight_num = b.flight_num and b.ratings IS NOT NULL and b.comments is NOT NULL'
        cursor.execute(query, (air_name))
        data = cursor.fetchall()

        cursor.close()
        return render_template('pastFlights.html', flight = data, airline_name = air_name)
    return redirect('/login/staff')
```

7. View frequent customers: Airline Staff will also be able to see the most frequent customer within the last year. In addition, Airline Staff will be able to see a list of all flights a particular Customer has taken only on that particular airline.



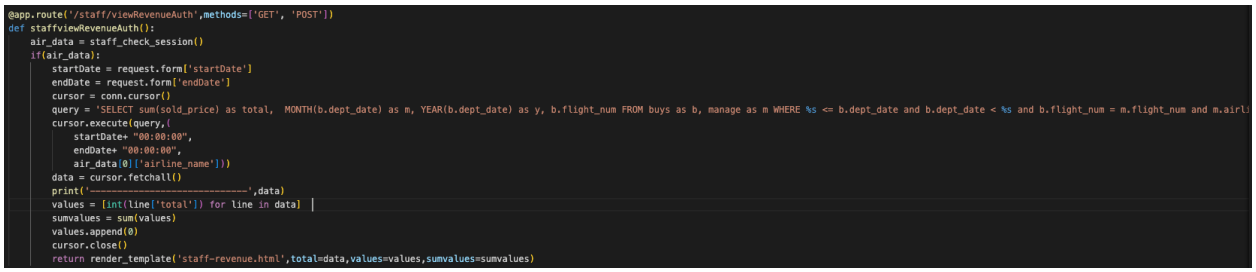
```
@app.route('/staff/view/customers')
def viewCustomers():
    airline = staff_check_session()
    if (airline):
        cursor = conn.cursor()
        query = 'SELECT distinct c.name, c.email FROM customer as c, buys as b, manage as m  WHERE c.email = b.email and b.flight_num = m.flight_num and m.airline_name = %s and YEAR(b.purchase_timestamp) = %s'
        cursor.execute(query, (
            airline[0]['airline_name'],
            date.today().strftime("%Y")
            ))
        data = cursor.fetchall()

        return render_template('staff-view-customer.html', airline= airline[0]['airline_name'], customer = data, date =date.today().strftime("%Y") )
    return redirect('/login/staff')

@app.route('/staff/view/specfic/customer',  methods=['GET', 'POST'])
def viewSpecficCustomer():
    airline = staff_check_session()
    if (airline):
        customer_email = request.form['email']
        cursor = conn.cursor()
        query = 'SELECT DISTINCT c.name, f.flight_num, f.dept_date, f.arr_date, f.dept_airport, f.arr_airport, f.id_airplane FROM flight as f, buys as b, manage as m, customer as c WHERE  b.email = c.email and b.email = %s and b.flight_num = f.flight_nu
        cursor.execute(query,(
            customer_email,
            airline[0]['airline_name']
        ))
        data = cursor.fetchall()
        cursor.close()
        return render_template('staff-specific-cus.html', details=data)
    return redirect('/login/staff')
```

8. View reports: Total amounts of ticket sold based on range of dates/last year/last month etc. Month wise tickets sold in a bar chart/table.



```
@app.route('/staff/reports')
def viewReports():
    airline = staff_check_session()
    if (airline):
        cursor = conn.cursor()
        query = 'SELECT DISTINCT count(b.ticket_id) as total, MONTH(b.purchase_timestamp) as m, YEAR(b.purchase_timestamp) as y, b.flight_num FROM buys as b, manage as m WHERE b.flight_num = m.flight_num and m.airline_name = %s group by b.flight_num, m, y'
        cursor.execute(query, airline[0]['airline_name'])
        data = cursor.fetchall()
        cursor.close()
        labels = [ 'Flight #' + str(line['flight_num']) + ': ' + (str(line['m']) + '/'+ str(line['y'])) for line in data]
        values = [int(line['total']) for line in data]
        values.append(0)

        return render_template('staff-reports.html', data =data, labels=labels, values=values)
    return redirect('/login/staff')

@app.route('/staff/reports/search', methods=['GET', 'POST'])
def searchReports():
    air_data = staff_check_session()
    if (air_data):
        startDate = request.form['startDate']
        endDate = request.form['endDate']
        cursor = conn.cursor()
        query = 'SELECT count(b.ticket_id) as total,  MONTH(b.purchase_timestamp) as m, YEAR(b.purchase_timestamp) as y, b.flight_num FROM buys as b, manage as m WHERE %s <= b.purchase_timestamp and b.purchase_timestamp < %s and b.flight_num = m.flight_num
        cursor.execute(query, (
            startDate+ "00:00:00",
            endDate+ "00:00:00",
            air_data[0]['airline_name']
        ))
        data = cursor.fetchall()
        cursor.close()
        labels = ['Flight #' + str(line['flight_num']) + ': ' + (str(line['m']) + '/'+ str(line['y'])) for line in data]
        values = [int(line['total']) for line in data]
        values.append(0)

        return render_template('staff-reports.html', data =data, labels=labels, values=values)
    return redirect('/login/staff')
```

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



```
@app.route('/staff/viewRevenueAuth',methods=['GET', 'POST'])
def staffviewRevenueAuth():
    air_data = staff_check_session()
    if(air_data):
        startDate = request.form['startDate']
        endDate = request.form['endDate']
        cursor = conn.cursor()
        query = 'SELECT sum(sold_price) as total,  MONTH(b.dept_date) as m, YEAR(b.dept_date) as y, b.flight_num FROM buys as b, manage as m WHERE %s <= b.dept_date and b.dept_date < %s and b.flight_num = m.flight_num and m.airl
        cursor.execute(query,(
            startDate+ "00:00:00",
            endDate+ "00:00:00",
            air_data[0]['airline_name']))
        data = cursor.fetchall()
        print('----------------------------',data)
        values = [int(line['total']) for line in data]
        sumvalues = sum(values)
        values.append(0)
        cursor.close()
        return render_template('staff-revenue.html',total=data,values=values,sumvalues=sumvalues)
```

```
and m.airline_name = %s group by b.flight_num, m,y'
```

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

```
@app.route('/logout/staff')
def logout_staff():
    session.pop('username')
    return redirect('/')
```