Assignment 3 Individual Report

SID:

520514492

(Do not edit the format of this cover sheet!)

Table of Contents

Part A:

Note: changes made to scaffold code are highlighted yellow.

Required functionality 1: show all airports known to database

database.py:

```
def list_airports(limit=None, offset=None):
    """
    Retrieve all airports from the Airports table.
    """
    # Get the database connection and set up the cursor
    conn = database_connect()
    if(conn is None):
        # If a connection cannot be established, send an Null object
        return None
    # Set up the rows as a dictionary
    cur = conn.cursor()
    returndict = None

    try:
        sql = """
            SELECT * FROM Airports
            LIMIT %s OFFSET %s;
        """

        # Retrieve all the information we need from the query
        returndict = dictfetchall(cur, sql, (limit, offset))

        # report to the console what we recieved
        print(returndict)
    except:
        # If there are any errors, we print something nice and return a null value
        import traceback
        traceback.print_exc()
        print("Error Fetching Airports", sys.exc_info()[0])

    # Close our connections to prevent saturation
    cur.close()
    conn.close()

    # return our struct
    return returndict
```

routes.py:

```python
@app.route('/airports')
def list_airports():
    '''
    List all airports by calling the relevant database functions and rendering the template.
    '''
    page = request.args.get('page', 1, type=int)
    items_per_page = 15
    offset = (page - 1) * items_per_page

    # Generate airport summaries with item per page limit and offset to paginate
    airports_listdict = []
    airports_listdict =database.list_airports(limit=items_per_page, offset=offset)

    # Set the next URL if the page limit is reached
    next_url = url_for('list_airports', page=page + 1) if len(airports_listdict) == items_per_page else None
    prev_url = url_for('list_airports', page=page - 1) if page > 1 else None

    # create a page data dict
    page_data = {
        'title': 'List of Airports',
        'stats': airports_listdict,
        'next_url': next_url,
        'prev_url': prev_url
    }

    # Handle the null condition
    if (airports_listdict is None):
        # Create an empty list and show error message
        airports_listdict = []
        flash('Error, there are no rows in users')
    return render_template('list_airports.html', page=page_data, session=session, airports=airports_listdict)
```

html:

```jinja
{% include 'top.html' %}

<div id="content" class="container my-4">
    <h1 class="page-title">{{page.get('title', 'Airport')}}</h1>
    <table class="table table-striped">
        <thead>
            <tr>
                <th>AirportID</th>
                <th>Name</th>
                <th>IATA Code</th>
                <th>City</th>
                <th>Country</th>
                {% if session.isadmin %}
                <th>Actions</th>
                {% endif %}
            </tr>
        </thead>
        <tbody>
            {% for item in page.stats %}
            <tr class="align-items-center">
                <td class="align-middle">{{ item['airportid'] }}</td>
                <td><a href="{{ url_for('list_single_airports', airportid=item['airportid']) }}">{{ item['name'] }}</a></td>
                <td class="align-middle">{{ item['iatacode'] }}</td>
                <td class="align-middle">{{ item['city'] }}</td>
                <td class="align-middle">{{ item['country'] }}</td>
                {% if session.isadmin %}
                <td>
                    <a href="{{ url_for('edit_airport', airportid=item['airportid']) }}">Edit</a> |
                    <a href="{{ url_for('delete_airport_route', airportid=item['airportid']) }}"
                    onclick="return confirm('Are you sure you want to delete this airport?');">Delete</a>
                </td>
                {% endif %}
            </tr>
            {% endfor %}
        </tbody>
    </table>
```

```
    <nav aria-label="Page navigation">
        <ul class="pagination justify-content-center">
            {% if page.prev_url %}
            <li class="page-item">
                <a class="page-link" href="{{ page.prev_url }}" aria-label="Previous">
                    <span aria-hidden="true">&laquo; Previous</span>
                </a>
            </li>
            {% else %}
            <li class="page-item disabled">
                <span class="page-link">&laquo; Previous</span>
            </li>
            {% endif %}

            {% if page.next_url %}
            <li class="page-item">
                <a class="page-link" href="{{ page.next_url }}" aria-label="Next">
                    <span aria-hidden="true">Next &raquo;</span>
                </a>
            </li>
            {% else %}
            <li class="page-item disabled">
                <span class="page-link">Next &raquo;</span>
            </li>
            {% endif %}
        </ul>
    </nav>
</div>
{% include 'end.html' %}
```

Output:

List of Airports

AirportID	Name	IATA Code	City	Country	Actions
3	Dubai International Airport	DXB	Dubai	United Arab Emirates	Edit | Delete
4	Los Angeles International Airport	LAX	Los Angeles	United States	Edit | Delete
5	O'Hare International Airport	ORD	Chicago	United States	Edit | Delete
6	Heathrow Airport	LHR	London	United Kingdom	Edit | Delete
7	Haneda Airport	HND	Tokyo	Japan	Edit | Delete
8	Hong Kong International Airport	HKG	Hong Kong	Hong Kong	Edit | Delete
9	Shanghai Pudong International Airport	PVG	Shanghai	China	Edit | Delete
10	Charles de Gaulle International Airport	CDG	Paris	France	Edit | Delete
11	Amsterdam Airport Schiphol	AMS	Amsterdam	Netherlands	Edit | Delete
12	Dallas-Fort Worth International Airport	DFW	Dallas	United States	Edit | Delete
13	Guangzhou Baiyun International Airport	CAN	Guangzhou	China	Edit | Delete
14	Frankfurt am Main International Airport	FRA	Frankfurt	Germany	Edit | Delete
15	Istanbul Atatürk Airport	IST	Istanbul	Turkey	Edit | Delete
16	Indira Gandhi International Airport	DEL	Delhi	India	Edit | Delete
17	Soekarno-Hatta International Airport	CGK	Jakarta	Indonesia	Edit | Delete

« Previous Next »

Required functionality 2: show airport with given AirportID.

database.py:

```
def list_airports_equifilter(attributename, filterval):
    """
    Retrieve airports where a specific attribute matches a given value.
    """
    # Get the database connection and set up the cursor
    conn = database_connect()
    if(conn is None):
        # If a connection cannot be established, send an Null object
        return None
    # Set up the rows as a dictionary
    cur = conn.cursor()
    val = None

    try:
        # Retrieve all the information we need from the query
        sql = f"SELECT * FROM Airports WHERE {attributename} = %s;"
        val = dictfetchall(cur, sql, (filterval,))
    except:
        # If there are any errors, we print something nice and return a null value
        import traceback
        traceback.print_exc()
        print("Error Fetching from Airports", sys.exc_info()[0])

    # Close our connections to prevent saturation
    cur.close()
    conn.close()

    # return our struct
    return val
```

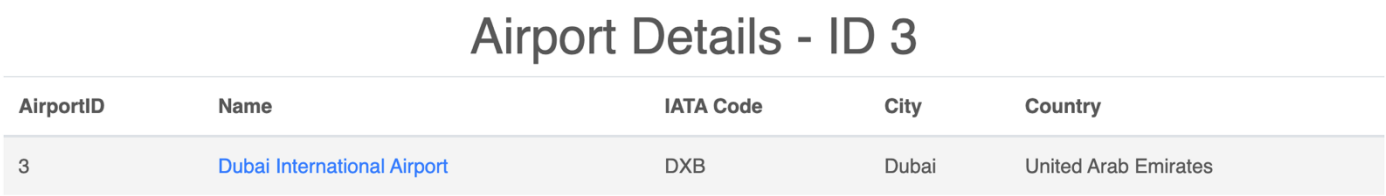
routes.py:

```python
@app.route('/airports/<airportid>')
def list_single_airports(airportid):
    '''
    List all rows in airports that match a particular id attribute airportid by calling the
    relevant database calls and pushing to the appropriate template
    '''

    # connect to the database and call the relevant function
    airports_listdict = None
    airports_listdict = database.list_airports_equifilter("AirportID", airportid)

    # Handle the null condition
    if (airports_listdict is None or len(airports_listdict) == 0):
        # Create an empty list and show error message
        airports_listdict = []
        flash(f'Error: No airport found with AirportID {airportid}.')

    page['title'] = f'Airport Details - ID {airportid}'
    return render_template('list_single_airports.html', page=page, session=session, airports=airports_listdict)
```

Html:

```html
{% include 'top.html' %}

<div id="content" class="container my-4">
    <h1 class="page-title">{{page.get('title', 'Airport')}}</h1>
    <table class="table table-striped">
        <thead>
            <tr>
                <th>AirportID</th>
                <th>Name</th>
                <th>IATA Code</th>
                <th>City</th>
                <th>Country</th>
                {% if session.isadmin %}
                <th>Actions</th>
                {% endif %}
            </tr>
        </thead>
        <tbody>
            {% for item in airports %}
            <tr class="align-items-center">
                <td class="align-middle">{{ item['airportid'] }}</td>
                <td><a href="{{ url_for('list_single_airports', airportid=item['airportid']) }}">{{ item['name'] }}</a></td>
                <td class="align-middle">{{ item['iatacode'] }}</td>
                <td class="align-middle">{{ item['city'] }}</td>
                <td class="align-middle">{{ item['country'] }}</td>
                {% if session.isadmin %}
                <td>
                    <a href="{{ url_for('edit_airport', airportid=item['airportid']) }}">Edit</a> |
                    <a href="{{ url_for('delete_airport_route', airportid=item['airportid']) }}"
                    onclick="return confirm('Are you sure you want to delete this airport?');">Delete</a>
                </td>
                {% endif %}
            </tr>
            {% endfor %}
        </tbody>
    </table>
</div>
{% include 'end.html' %}
```

Output:



Airport Details - ID 3

AirportID	Name	IATA Code	City	Country
3	Dubai International Airport	DXB	Dubai	United Arab Emirates

Required functionality 3: add a new airport to the database.

database.py:

```
def add_airport_insert(airportid, name, iata_code, city, country):
    conn = database_connect()
    if(conn is None):
        return None
    cur = conn.cursor()
    sql = """
        INSERT INTO Airports(AirportID, Name, IATACode, City, Country)
        VALUES (%s, %s, %s, %s, %s);
    """
    print_sql_string(sql, (airportid, name, iata_code, city, country))
    try:
        # Try executing the SQL and get from the database
        cur.execute(sql, (airportid, name, iata_code, city, country))
        conn.commit()  # Commit the transaction
        cur.close()   # Close the cursor
        conn.close() # Close the connection to the db
        return []
    except:
        # If there were any errors, return a NULL row printing an error to the debug
        print("Unexpected error adding an airport:", sys.exc_info()[0])
        cur.close()
        conn.close()
        raise
```

routes.py:

```
def add_airport():
    """
    Add a new Airport.
    """
    # # Check if the user is logged in, if not: back to login.
    if('logged_in' not in session or not session['logged_in']):
        return redirect(url_for('login'))

    # admin priveleges are in top.html

    page['title'] = 'Add New Airport'

    # Check your incoming parameters
    if request.method == 'POST':
        try:
            airportid = int(request.form['airportid'])
        except ValueError:
            flash('AirportID must be an integer.')
            return redirect(url_for('add_airport'))

        # obtain variables from html form
        name = request.form.get('name', '').strip()
        iata_code = request.form.get('iata_code', '').strip().upper()
        city = request.form.get('city', '').strip()
        country = request.form.get('country', '').strip()

        # Check null condition
        if not name or not iata_code or not city or not country:
            flash("you didn't input all fields")
            return redirect(url_for('add_airport'))

        if len(iata_code) != 3:
            flash('IATA Code must be 3 characters.')
            return redirect(url_for('add_airport'))

        try:
            # add new airport to database
            database.add_airport_insert(airportid, name, iata_code, city, country)
            return redirect(url_for('list_airports'))
        except Exception as e:
            flash(f'Error adding airport: {e}')
            return redirect(url_for('add_airport'))
    else:
        return render_template('add_airport.html', page=page, session=session)
```

html:

```
{% include 'top.html' %}
<div class="content">
    <div class="container my-4">

        <h2 class="title"> Add Airport </h2>

        <form class="needs-validation" method="POST" action="{{url_for('add_airport')}}" novalidate>

            <div class="form-group">
                <label for="airportid">Airport ID:</label>
                <input type="text" class="form-control" id="airportid" name="airportid" placeholder="Enter Airport ID here" required>
                <div class="invalid-feedback">Please enter an Airport ID.</div>
            </div>

            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" class="form-control" id="name" name="name" placeholder="Enter name here" required>
                <div class="invalid-feedback">Please enter a name.</div>
            </div>

            <div class="form-group">
                <label for="iata_code">IATA Code:</label>
                <input type="text" class="form-control" id="iata_code" name="iata_code" placeholder="Enter IATA code here" required>
                <div class="invalid-feedback">Please enter an IATA code.</div>
            </div>

            <div class="form-group">
                <label for="city">City:</label>
                <input type="text" class="form-control" id="city" name="city" placeholder="Enter a city here" required>
                <div class="invalid-feedback">Please enter a city.</div>
            </div>

            <div class="form-group">
                <label for="country">Country:</label>
                <input type="text" class="form-control" id="country" name="country" placeholder="Enter a country here" required>
                <div class="invalid-feedback">Please enter a country.</div>
            </div>

            <button class="btn btn-primary" type="submit">Add</button>
        </form>

    </div>
</div>
{% include 'end.html' %}
```

Output:

Required functionality 4: update fields for a particular airport

database.py:

```
def update_single_airport(airportid, name, iata_code, city, country):
    # Get the database connection and set up the cursor
    conn = database_connect()
    if(conn is None):
        # If a connection cannot be established, send an Null object
        return None

    # Set up the rows as a dictionary
    cur = conn.cursor()
    val = None

    # progressively concatenate attributes to update
    try:
        setitems = ""
        attcounter = 0
        if name is not None:
            setitems += "name = %s\n"
            attcounter += 1
        if iata_code is not None:
            if attcounter != 0:
                setitems += ","
            setitems += "iatacode = %s\n"
            attcounter += 1
        if city is not None:
            if attcounter != 0:
                setitems += ","
            setitems += "city = %s\n"
            attcounter += 1
        if country is not None:
            if attcounter != 0:
                setitems += ","
            setitems += "country = %s\n"
            attcounter += 1

        sql = f"""UPDATE airports
                    SET {setitems}
                    WHERE airportid = '{airportid}';"""
        print_sql_string(sql, (name, iata_code, city, country))
        val = dictfetchone(cur, sql, (name, iata_code, city, country))
        conn.commit()

    except:
        # If there are any errors, we print something nice and return a nu
        print("Unexpected error updating an airport:", sys.exc_info()[0])
        cur.close()
        conn.close()
        raise

    cur.close()
    conn.close()

    return val
```

routes.py:

```
def edit_airport(airportid):
    """
    Edit an Airport.
    """
    # # Check if the user is logged in, if not: back to login.
    if('logged_in' not in session or not session['logged_in']):
        return redirect(url_for('login'))

    page['title'] = 'Edit Airport details'

    if request.method == 'POST':
        name = request.form.get('name', '').strip()
        iata_code = request.form.get('iata_code', '').strip().upper()
        city = request.form.get('city', '').strip()
        country = request.form.get('country', '').strip()

        # Set fields to None if they are empty
        name = name if name else None
        iata_code = iata_code if iata_code else None
        city = city if city else None
        country = country if country else None

        # Validate IATA Code
        if iata_code and len(iata_code) != 3:
            flash('IATA Code must be 3 characters.')
            return redirect(url_for('edit_airport', airportid=airportid))

        # Update the database
        try:
            database.update_single_airport(airportid, name, iata_code, city, country)
            return redirect(url_for('list_single_airports', airportid=airportid))
        except Exception as e:
            flash(f'Error updating airport: {e}')
            return redirect(url_for('edit_airport', airportid=airportid))
    else:
        airports_listdict = database.list_airports_equifilter("AirportID", airportid)

        if (airports_listdict is None or len(airports_listdict) == 0):
            flash(f'Error: No airport found with AirportID {airportid}.')
            return redirect(url_for('list_airports'))

        airport = airports_listdict[0]
        return render_template('edit_airport.html', page=page, session=session, airport=airport)
```

html:

```
{% include 'top.html' %}
<div class="content">
    <div class="container my-4">

        <h2 class="title"> Update Airport </h2>

        <form class="needs-validation" method="POST" action="{{url_for('edit_airport', airportid=airport.airportid)}}" novalidate>

            <div class="form-group">
                <label for="airportid">Airport ID:</label>
                <input type="text" class="form-control" id="airportid" name="airportid" value="{{ airport.airportid }}" required>
                <div class="invalid-feedback">Please enter an Airport ID.</div>
            </div>

            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" class="form-control" id="name" name="name" value="{{ airport.name }}" required>
                <div class="invalid-feedback">Please enter a name.</div>
            </div>

            <div class="form-group">
                <label for="iata_code">IATA Code:</label>
                <input type="text" class="form-control" id="iata_code" name="iata_code"  value="{{ airport.iatacode }}" required>
                <div class="invalid-feedback">Please enter an IATA code.</div>
            </div>

            <div class="form-group">
                <label for="city">City:</label>
                <input type="text" class="form-control" id="city" name="city" value="{{ airport.city }}" required>
                <div class="invalid-feedback">Please enter a city.</div>
            </div>

            <div class="form-group">
                <label for="country">Country:</label>
                <input type="text" class="form-control" id="country" name="country" value="{{ airport.country }}" required>
                <div class="invalid-feedback">Please enter a country.</div>
            </div>

            <button class="btn btn-primary" type="submit">Update</button>

        </form>

    </div>
</div>
{% include 'end.html' %}
```

Output:

Required functionality 5: remove an airport that is in the database

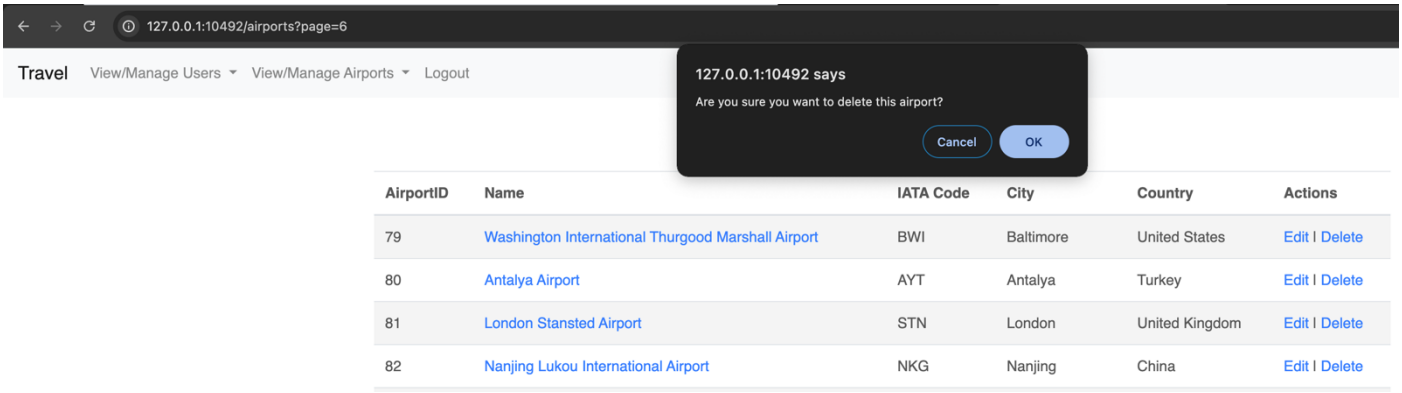
database.py:

```python
def delete_airport(airportid):
    """
    Delete an airport from the Airports table.
    """
    conn = database_connect()
    if(conn is None):
        return None
    cur = conn.cursor()
    try:
        sql = "DELETE FROM Airports WHERE AirportID = %s;"
        cur.execute(sql, (airportid,))
        conn.commit()
        cur.close()
        conn.close()
        return []
    except:
        print(f"Unexpected error deleting airport with AirportID {airportid}:", sys.exc_info()[0])
        cur.close()
        conn.close()
        raise
```

routes.py:

```python
def delete_airport_route(airportid):
    """
    Delete an airport.
    """
    try:
        database.delete_airport(airportid)
    except Exception as e:
        flash(f'Error deleting airport: {e}')

    return redirect(url_for('list_airports'))
```

Output:



Required functionality 6: display number of airports for each country

database.py:

```
def list_airport_stats(limit=None, offset=None):
    """
    Retrieve a summary of airports grouped by country with multiple pages.
    """
    conn = database_connect()
    if(conn is None):
        # If a connection cannot be established, send an Null object
        return None
        # Set up the rows as a dictionary
    cur = conn.cursor()
    returndict = None
    try:
        # Set-up our SQL query
        sql = """
            SELECT country, COUNT(*) as count
            FROM airports
            GROUP BY country
            ORDER BY count DESC
            LIMIT %s OFFSET %s;
        """


        # Retrieve all the information we need from the query
        returndict = dictfetchall(cur, sql, (limit, offset))

        # report to the console what we recieved
        print(returndict)
    except:
        # If there are any errors, we print something nice and return a null value
        print("Error Fetching Airport Summary", sys.exc_info()[0])

    # Close our connections to prevent saturation
    cur.close()
    conn.close()
```

routes.py:

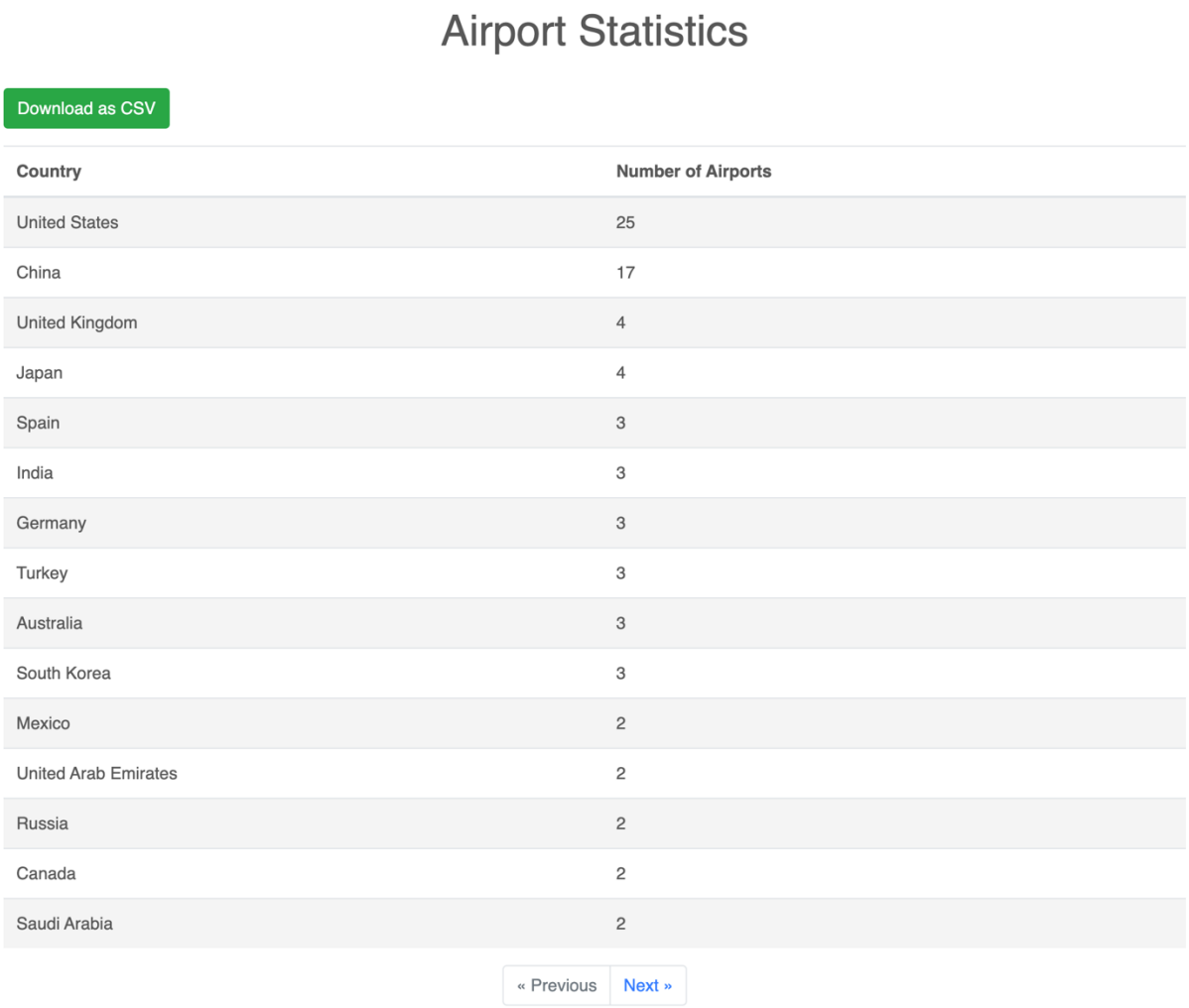
```python
def airport_stats():
    """
    Display a summary of airports grouped by country, with pagination.
    """
    # Get the current page from the request
    page = request.args.get('page', 1, type=int)
    items_per_page = 15
    offset = (page - 1) * items_per_page

    # Generate airport summaries with item per page limit and offset to paginate
    summary = database.list_airport_stats(limit=items_per_page, offset=offset)

    # Set the next URL if the page limit is reached
    next_url = url_for('airport_stats', page=page + 1) if len(summary) == items_per_page else None
    prev_url = url_for('airport_stats', page=page - 1) if page > 1 else None

    # create a page data dict
    page_data = {
        'title': 'Airport Summary by Country',
        'stats': summary,
        'next_url': next_url,
        'prev_url': prev_url
    }

    return render_template('airport_stats.html', page=page_data, session=session)
```

html:

```
{% include 'top.html' %}

<div id="content" class="container my-4">
    <h1 class="page-title">Airport Statistics</h1>
    <br/>

    <a href="{{ url_for('export_airport_stats_csv') }}" class="btn btn-success mb-3">Download as CSV</a>

    <table class="table table-striped">
        <thead>
            <tr>
                <th>Country</th>
                <th>Number of Airports</th>
            </tr>
        </thead>
        <tbody>
            {% for item in page.stats %}
            <tr>
                <td>{{ item.country }}</td>
                <td>{{ item.count }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
```

```
    <!-- Multiple Page Controls -->
    <nav aria-label="Page navigation">
        <ul class="pagination justify-content-center">
            {% if page.prev_url %}
            <li class="page-item">
                <a class="page-link" href="{{ page.prev_url }}" aria-label="Previous">
                    <span aria-hidden="true">&laquo; Previous</span>
                </a>
            </li>
            {% else %}
            <li class="page-item disabled">
                <span class="page-link">&laquo; Previous</span>
            </li>
            {% endif %}

            {% if page.next_url %}
            <li class="page-item">
                <a class="page-link" href="{{ page.next_url }}" aria-label="Next">
                    <span aria-hidden="true">Next &raquo;</span>
                </a>
            </li>
            {% else %}
            <li class="page-item disabled">
                <span class="page-link">Next &raquo;</span>
            </li>
            {% endif %}
        </ul>
    </nav>
</div>
{% include 'end.html' %}
```

Output:



Part B:

Extension 1: pagination:

The first extension was learning additional flask features to implement pagination. Pagination was implemented in both the page that lists airports and the page that provides a summary of airports per country. As seen im the 'database.py' output of required functionality 1, the SQL query was edited to include a LIMIT and an OFFSET.

Values for the limit and offset are passed in through routes.py, where the limit, or number of values to return from the query, is limited to 10 in the case of the listing airports page. And the offset, or number of rows to skip, is "(page - 1) * items per page". For example, on page 2, the number of rows to skip would be (2 – 1) * 10 = 10 rows: you only want to display the second 10 rows, and skip the first 10. In the list airports HTML file, buttons to swap between the previous and next page are implemented.

Extension 2: download as CSV

The second extension was learning the Response library from flask to allow users to download airport country statistics. In routes.py, the 'Country' and 'Number of Airports' headings are written into a csv, followed by each row in the respective columns, replicating the summary page in csv format. Then, a Response constructor is called to download the csv file onto the user's machine. A button was added to the summary HTML file, to call this function.