SI206 Final Project Report
Laura Li, Ariel Huang
Github repository: https://github.com/lilaura/206fin

1. **The goals for your project.**
   Our original goal was to explore flight delays, weather and their relationships. We planned to collect and visualize the percentage of flight delay at different airports each day, the amount of precipitation at the airport location each day, and the correlation between the two.

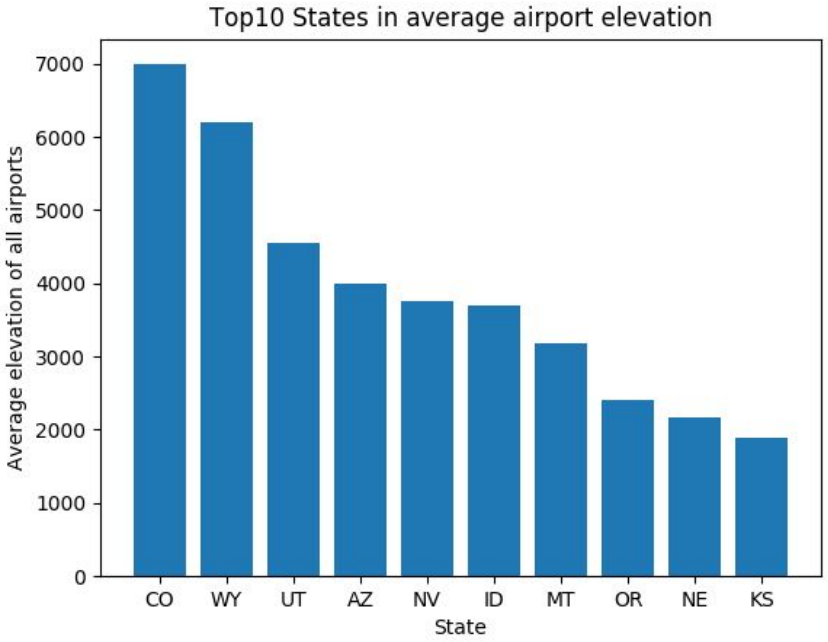
2. **The goals that were achieved.**
   We created a US map with data points of airports mapped onto it, when the user hover over the data points, then can see different weather attributes on the day the files are executed. This supports the visualization of average temperature, precipitation intensity, wind speed, and visibility of different airports in the US, and allows the users to see where the data points/airports are on a map based on longitude and latitude. The weather attributes are chosen based on their importance towards influencing flight delays.
   We calculated and visualized the top 10 states whose airports have the highest elevation on average. This allows users to get a sense of the elevations of airports in each state and most elevated state in general.

3. **The problems that you faced.**
   a. It was difficult to find a flight API that provides sufficient historical delay data at one request. Some APIs have a limit of 80 items max; some limit the window of available data to 6 hours.
   b. It was even more difficult to find a weather API that provides sufficient weather data within the limits in a). Highest precision of weather data is hourly, which does not provide us with sufficient data.
   c. The scatterplot map was more difficult to implement than expected. The documentation provided was only for certain kinds of data, extra research and modifications had to be done in order to avoid error messages。

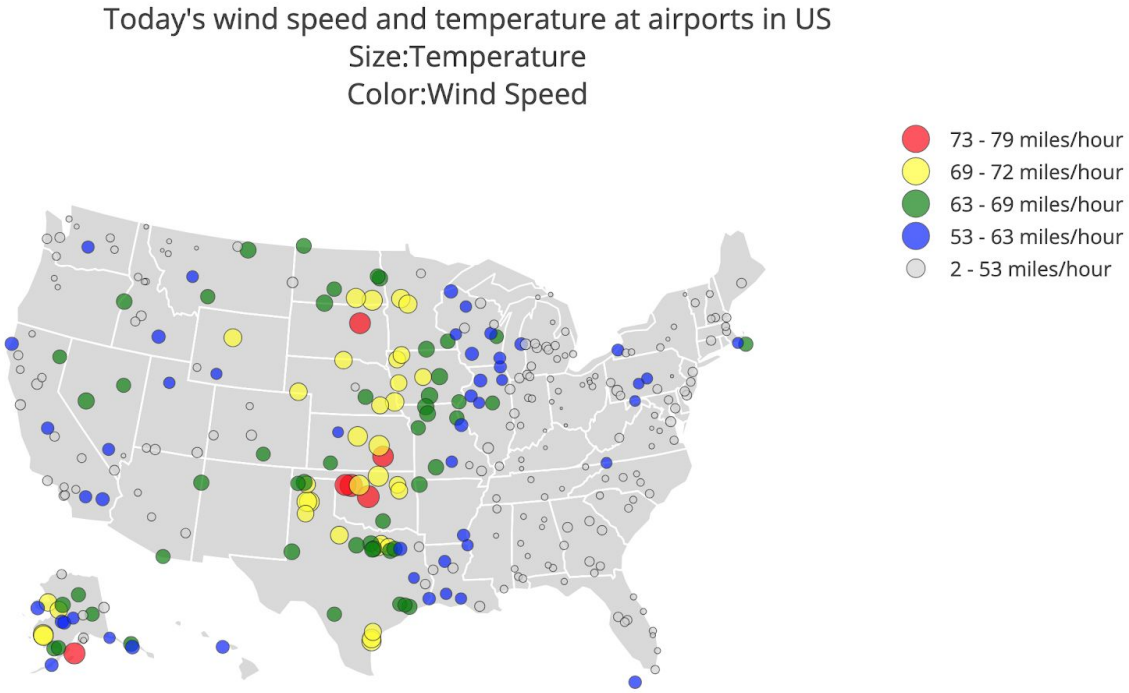
4. **Your file that contains the calculations from the data in the database.**
   See calc.json
   https://drive.google.com/file/d/1E9u0wFNA6UlUR8uJPKCEsPigddk4sInk/view?usp=sharing

5. **The visualization that you created (i.e. screenshot or image file).**

See: 1. Top10 States in average airport elevation.png



2. Airports-bubblemap.png



https://plot.ly/~ilaurali/2/todays-wind-speed-and-temperature-at-airports-in-us-size-temperature-color-wind-/#/

6. **Instructions for running your code.**
    a. Start with running vis1.py
    b. Database will be populated 20 items at a time, run vis1.py multiple times to get sufficient amount of data
    c. Then run vis2.py to create calculations and visualizations

7. **Documentation for each function that you wrote. This includes the input and output for each function.**
    a. get_airport_lst(APIid, APIkey)
    This function takes the id and key for the Airport API as input. It gets all airports by country code (US) from the API using the Requests module as a response object. It then parses the response object with json. A list of dictionaries of airport information is returned as the output.

    b. add_airport_to_db(airport_lst)
    This function takes the airport list returned by get_airport_lst as an input. It keeps a count of number of items added to the database and quits after the count reaches 20. It has a for loop that loops through the dictionaries in airport_lst. For each airport (dictionary) it checks if the airport is already in the database. If yes, it prints "Found in database". If not, it inserts a row into the Airports table, commit the changes to the database and increment the count by 1. For each airport added to the database, it also adds a tuple of latitude and longitude to a list, and return this list as the output of the function. This list is used as an input in add_weather to_db to make sure that lat and lng are the keys of both tables, and there are no duplicate data in the table weather.

    c. Add_weather_to_db(conn, cur, longlatlist)
    This function take in the connection to sqlite database, the cursor, and the latitude, longitude list returned by add_airport_to_db. With each time the function is called, it checks for duplication in the table, and avoids saving them into the table. It uses the latitude and longitude in longlatlist to request precipitation intensity, visibility, and windspeed from darksky api. For each location requested, the function insert the result data into the matching columns in the database.

    d. write_csv(calc_file, calc_lst, elev_dict)

This function takes a file name and two dictionaries as input, and return calculated data in the dictionary as a csv file.

e. map_calc(conn,cur)

   This function takes the conn (connection to sqlite database) and cur as inputs. It selects the `temp_high, temp_low,latitude, longitude, precip_int, wind_speed, visibility` *columns from the Weather table. For each airport/row, the average temperature of the day at each airport from the first two columns would be calculated, then the calculated result and the rest of the attributes will be saved in a list to be returned as output.*

f. bubble_map(csvfile)

   This function takes the filename of the output file as input. It creates a geo scatterplot with latitude and longitude of the airports. The function enables the following features:

   - *Each datapoint has its size determined by temperature at the location of the airport, the higher the temperature, the larger the circle.*
   - *The colour of the data points are determined by the wind speed at the location of the airport, with legends on the side.*
   - *The text displayed when hovering over the data points indicates the precipitation and visibility at the location of the airport.*

g. calc_avg_elev(conn,cur)

   This function takes the conn (connection to sqlite database) and cur as inputs. It selects the state and elevation columns from the Airports table. A new dictionary is created. For each row, the state is added as key and the elevation of each airport is added to a list as the dictionary value. After looping through all the rows, the average elevation of each state's airports is calculated. A dictionary with states as keys and average elevation as values is returned as the output.

h. sort_elev_top_ten(elev_dict)

   This function takes the dictionary returned by calc_avg_elev as the input. It sorts the dictionary items by the average elevation values in descending order. The top 10 items are returned as a list as the output.

i. bar_chart_avg_elev_by_state(top_elev_dict)
This function takes the dictionary returned by sort_elev_top_ten as the input. It creates a bar chart with the state names as the x-axis and the average elevation values as the y-axis.

8. **You must also clearly document all resources you used. The documentation should be of the following form:**

Date	**Issue Description**	**Location of Resource**	**Result (Resolved Y/N)**
04/21	Check Airport API Documentation; check out response using interactive documentation	https://developer.flightstats.com/api-docs/airports/v1	Y
04/21	Check Weather API documentation for necessary parameters; check example responses for json structure	https://darksky.net/dev/docs	Y
04/21	Lookup how to use plotly to create a bubble US map	https://plot.ly/python/bubble-maps/	Y
04/21	Lookup top influential weather attributes for flight delays	https://www.weatherworksinc.com/weather-air-travel	Y