Chris Acciaioli & Erik Martin

SI 206 Final Project Report

Link to github repo: https://github.com/rickmart2/206finalproj

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)

Originally we had planned to use a transportation API (*Uber* API) and a weather API (*AccuWeather* API) to see how different types of weather affects various transportation statistics. Moreover, we planned to gather weather data about precipitation, temperature, and wind speed as well as transportation data regarding delays, frequency/availability, cancellations, and pricing in order to eventually see if there was a relationship between the two.

2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)

Once we realized that the API's we had originally planned to use were not as ideal for our project as we had hoped, we decided to continue searching for better API's to use and ended up changing the theme our project and the goals we hoped to achieve from analyzing the relationship between weather and transportation to the effect that weather has on hitting success in major league baseball. To do this, we worked with two API's (*Visual Crossing* API to collect weather related data and the *MLB.com* API to get season statistics for Major League Baseball). From the weather API, we gathered information regarding temperature, wind chill, precipitation, windspeed, and visibility in Chicago and from the MLB API, we collected data about the amount of runs scored by both home and away teams at two different ballparks in Chicago.

3. The problems that you faced (10 points)

Overall, this project ran relatively smoothly! However, there were three minor issues that caused us some trouble while completing our data collection, calculations, and visualizations. To start, when first using the weather API (*Visual Crossing* API) to collect weather data, our api keys kept expiring and we did not know why since we thought that we could make 1000 requests each day. However, when looking deeper into the documentation, we realized that we could only collect 1000 records each day from the API (since each day counts as a single record of weather data and we were originally collecting information for hundreds of days, our keys were expiring very quickly). In addition, we had trouble using our created database for calculations/visualizations until we realized that we had duplicate string data. Finally, we were having issues displaying our subplots correctly (i.e. they were overlapping or axis labels were displayed incorrectly), but after reviewing lecture slides for matplotlib we were able to solve our issues on our own.

4. The calculations from the data in the database (i.e. a screenshot) (10 points)

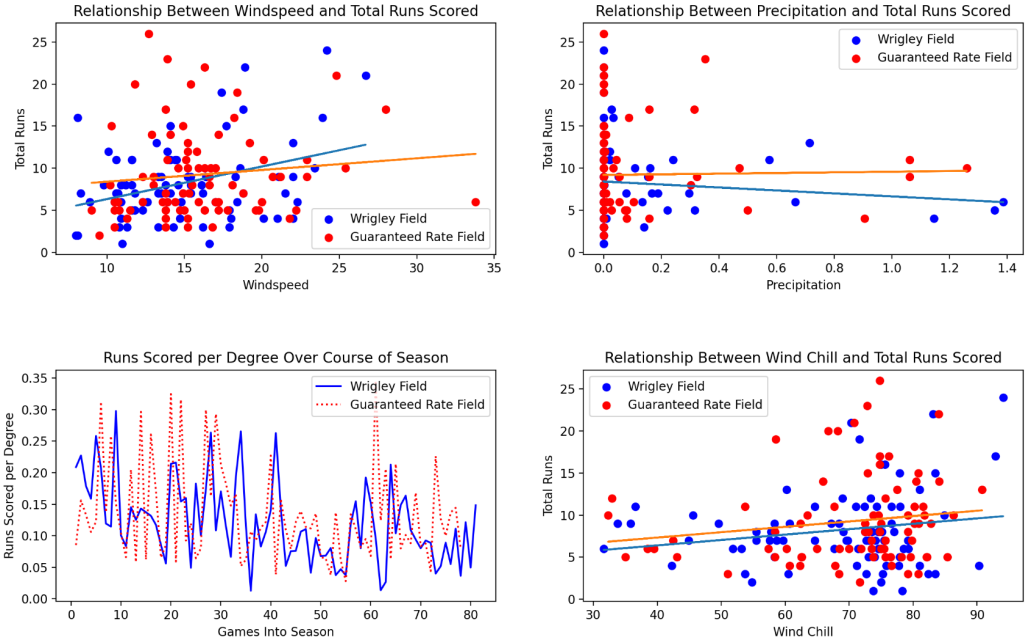
```
def main():
    path = os.path.dirname(os.path.abspath(__file__))
    conn = sqlite3.connect(path+'/proj.db')
    cur = conn.cursor()

    path = os.path.dirname(os.path.abspath(__file__))
    cur.execute("SELECT WeatherData.temp, WeatherData.precip, WeatherData.windspeed, Mlb.home_score, Mlb.away_score, Mlb.stadium from M
    data = cur.fetchall()

    with open (path+'/'+'calc.txt', 'w') as f:
        f.write('Temperature, Total Score, Stadium, Run Scored per Degree')
        f.write('\n')
        for item in data:
            if item[5] == 1:
                stadium = "Guaranteed Rate Field"
            else:
                stadium = "Wrigley Field"
            total = item[3] + item[4]
            temp = item[0]
            tot_p_temp = total/temp
            f.write(str(temp)+","+str(total)+","+str(stadium)+","+str(tot_p_temp))
            f.write("\n")
```

```
Temperature, Total Score, Stadium, Run Scored per Degree
43.1,9,Wrigley Field,0.2088167053364269
39.6,9,Wrigley Field,0.22727272727272727
50.3,9,Wrigley Field,0.17892644135188868
58.7,5,Guaranteed Rate Field,0.08517887563884156
64.1,10,Guaranteed Rate Field,0.15600624024961
45.1,6,Guaranteed Rate Field,0.13303769401330376
47.2,5,Guaranteed Rate Field,0.1059322033898305
41.0,5,Guaranteed Rate Field,0.12195121951219512
38.4,12,Guaranteed Rate Field,0.3125
37.8,6,Wrigley Field,0.15873015873015875
42.6,11,Wrigley Field,0.25821596244131456
48.2,10,Wrigley Field,0.20746887966804978
58.7,7,Wrigley Field,0.11925042589437819
52.4,6,Wrigley Field,0.11450381679389313
70.5,21,Wrigley Field,0.2978723404255319
69.6,7,Wrigley Field,0.10057471264367816
43.7,6,Guaranteed Rate Field,0.13729977116704806
38.7,10,Guaranteed Rate Field,0.25839793281653745
46.4,7,Guaranteed Rate Field,0.15086206896551724
57.4,6,Guaranteed Rate Field,0.10452961672473868
60.7,4,Guaranteed Rate Field,0.06589785831960461
53.7,11,Guaranteed Rate Field,0.20484171322160147
51.6,3,Guaranteed Rate Field,0.05813953488372093
47.9,4,Wrigley Field,0.08350730688935282
48.2,7,Wrigley Field,0.14522821576763484
55.9,7,Wrigley Field,0.12522361359570663
55.9,8,Wrigley Field,0.14311270125223613
58.3,8,Wrigley Field,0.137221269296741
70.8,21,Guaranteed Rate Field,0.2966101694915254
80.3,5,Guaranteed Rate Field,0.062266500622665005
```

5. The visualization that you created (i.e. screenshot or image file) (10 points)



6. Instructions for running your code (10 points)
    a. Run the API1.py file until it prints: “All rows have been added to the database”
    b. Then run the weather_api.py file until it prints: “All rows have been added to the database”
    c. Proj.db now has all 3 tables.
    d. Run Join_Calc.py, you can now see the written file, calc.txt, and the visualizations should also appear now.

7. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)
    - API1.py:
        - getMLBdata():
            - There is no input for this function. This function requests the data from the MLB.com API, then filters it down to just the games played in Chicago. Then it creates a list of dictionaries containing the information we need and outputs that list of dictionaries.
        - setUpDb():
            - The input for this function is the name of the database you want to set up/create. This function just creates the database in SQLite.
        - createMLBtab():
            - This function takes 4 inputs, the database connection and cursor, a list (the list of dictionaries from getMLBdata(), and a start index.

This function then starts from the starting index and adds 25 rows to the database under the Mlb table and commits it.
    - main():
        - Similar to the homeworks, the main function just calls to all of the functions described above to execute the process of getting the data from the API, formatting it, then adding it to the database.
- Weather_api.py:
    - collect_weather_data():
        - This function takes no inputs. It requests data from the *Visual Crossing* API, selects only desired data for each of the days in the 2022 MLB regular season, and returns desired data for each of those days in the form of a list of dictionaries.
    - construct_data_base():
        - The input for this function is the name of the database you want to set up/create. This function creates your database in SQLite.
    - create_weather_table():
        - This function takes 4 inputs (the list of dictionaries returned from the collect_weather_data() function, the database connection and cursor, and a starting index. This function then starts from the starting index and adds 25 rows to the database under the WeatherData table and commits it.
    - main():
        - The main function calls all of the previously described functions and executes the process of retrieving the data from the API, properly formatting it as a list of dictionaries, and adding that data to the database.

- Join_calc.py:
    - main():
        - All of the calculations and visualizations are encapsulated in the main() function. Running the main functions creates the visualizations as well as calc.txt.

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	Issue Description	Location of Resource	Result (Did you solve the issue?)
4/20	Trouble adding a line of best fit to our matplotlib scatter plots	https://www.statology.org/line-of-best-fit-python/	Yes
4/18	Trouble wrangling the right data from the MLB.com API. (Getting	https://github.com/brianhaferkamp/mlba	Yes

	game result info for the whole season)	pidata/blob/master/README.md#new-api-data	