Simon Miller, Kate Leib, Evan Rasmussen

SI 206 FINAL PROJECT REPORT

1. Original Goals
    a. Our original goals were to use three API's to:
        i. Graph of height vs weight (NBA)
        ii. Chart of most league titles for each team (soccer)
        iii. Distribution of ages of active players (all)
        iv. Average ERA for righties/lefties (MLB)
2. Goals Achieved
    a. Our goals changed a bit as we began to work with the API's as we ran into a few problems. We decided to change the data collected/visualization for the NBA API to be a given team's average total game score for one season. For the soccer API, we decided to instead collect the average heights of players in the Premier League by position. We did collect average ERA for righties and lefties. In addition, we also collected the average WHIP for righty and lefty pitchers in the MLB.
3. Problems Faced
    a. For the soccer API, we were originally planning on using a BeautifulSoup to create a table of countries and then create a visualization of the distribution of what countries players are from in a pie chart. However, this table of countries had only 60 countries in it, so we had to drop it since it was less than 100. We decided to instead collect heights of players in the Premier League and organize it in a bar chart by position.
    b. For the baseball API, we noticed that for some reason many of the pitchers were not getting complete data returned from the API. To fix this problem, we added some if/else statements and try statements to check if the data was present or not.

The pitchers that did not have complete data were not added to the table in the database. The ones who did have complete data were added to the table.

c. For all three APIs we faced issues of getting our API connection reset or ratelimited by the API. To fix this we had to read through the documentation on the rate limits for each specific API. We also used the 'Time' package to slow down the execution of some functions as well as printing out messages to let the user know to wait some time before running again.

4. File with calculations → Same as 'data.json' in zip (sorry for long list, needed for scatterplot):


```
{
   "Basketball": {
       "Knicks": 215.42073170731706,
       "Lakers": 221.47560975609755
   },
   "Baseball": {
       "ERA": {
           "Left": 5.366333333333334,
           "Right": 5.175769230769231
       },
       "WHIP": {
           "Left": 1.6116666666666666,
           "Right": 1.5128205128205128
       },
       "Total": {
           "ERA": [
               5.14,
               4.39,
               3.19,
               3.64,
               2.01,
               3.1,
               3.15,
               4.76,
               4.74,
               3.21,
```

```
            2.44,
            4.15,
            3,
            2,
            15.75,
            10.69,
            3.93,
            4.5,
            4.91,
            1.65,
            7.2,
            3.46,
            12.38,
            3.11,
            2.86,
            2.85,
            4.29,
            2.94,
            5.95,
            4.43,
            4.92,
            18,
            3.23,
            4.45,
            3.9,
            7.2,
            17.05,
            6,
            2.83,
            1.88,
            10.8,
            3.51,
            3.94,
            4.03,
            2.11,
            18,
            6.43,
            3.43,
            4.5,
            7.71,
            1.93,
            5.45,
```

```
            9,
            5.29,
            3.96,
            4.9,
            7.71,
            3.35,
            3,
            3.99,
            5.59,
            4.88,
            5.63,
            13.5,
            6.43,
            3.6,
            15.43,
            5.92,
            8.49,
            5.4,
            10.46,
            5.55,
            5.4,
            3.65,
            1.6,
            7.41,
            4.7,
            4.2,
            4.17,
            4.39,
            2.74,
            6.08,
            4.28,
            3.22,
            3.58,
            3.82,
            2.11,
            8.1,
            3.77,
            5.63,
            3.18,
            3.68,
            3.27,
            2.68,
```

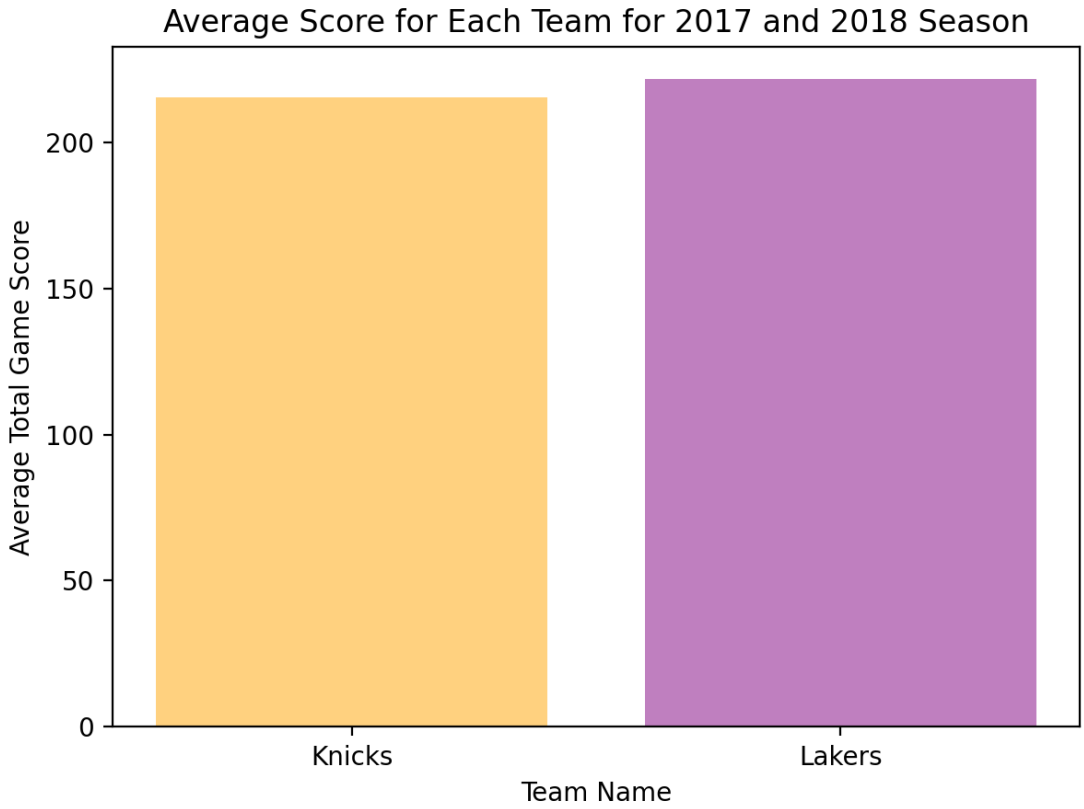
```
            2.93,
            5.3,
            2.18,
            4.95,
            7.65,
            2.6,
            5.17,
            1.46,
            3.44,
            3.32,
            11.81,
            1.04,
            4,
            3.99
        ],
        "WHIP": [
            1.43,
            1.43,
            1.19,
            1.14,
            1.04,
            1.34,
            1.05,
            1.5,
            1.45,
            1.08,
            1.09,
            1.27,
            0.89,
            1.19,
            3.25,
            2,
            1.35,
            1.33,
            2.73,
            1.1,
            1.2,
            1.54,
            2.88,
            1.27,
            1.22,
            1.08,
```

```
            1.45,
            1.37,
            1.22,
            1.52,
            1.46,
            4,
            1.29,
            1.76,
            1.33,
            1.8,
            2.68,
            1.33,
            1.08,
            0.98,
            2,
            1.44,
            1.17,
            1.34,
            1.17,
            3,
            2,
            1.19,
            1.67,
            1.68,
            1.22,
            1.41,
            2,
            1.58,
            1.45,
            1.41,
            3,
            1.27,
            1.26,
            1.19,
            2.22,
            1.38,
            2.25,
            1.8,
            1.33,
            1.1,
            4.29,
            1.53,
```

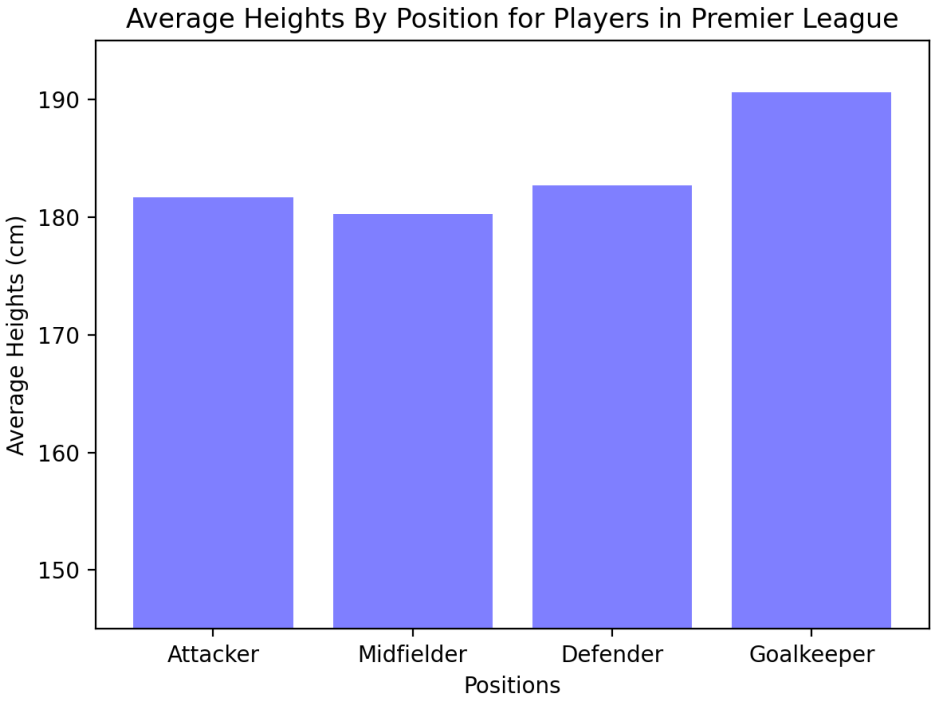
```
            1.89,
            1.56,
            2.21,
            1.62,
            1.45,
            1.26,
            0.77,
            1.82,
            1.7,
            1.33,
            1.43,
            1.36,
            0.99,
            1.77,
            1.18,
            1.12,
            1.14,
            1.26,
            0.86,
            2.25,
            1.4,
            1.58,
            1.45,
            1.16,
            1.21,
            1.25,
            1.37,
            1.8,
            1.04,
            1.43,
            1.88,
            1.31,
            1.47,
            1.14,
            1.15,
            1.31,
            2.25,
            1.15,
            1,
            1.37
        ]
    }
```

```
    },
    "Soccer": {
        "Attacker": 181.7058823529412,
        "Midfielder": 180.26315789473685,
        "Defender": 182.675,
        "Goalkeeper": 190.6
    }
}
```


5. Visualizations:

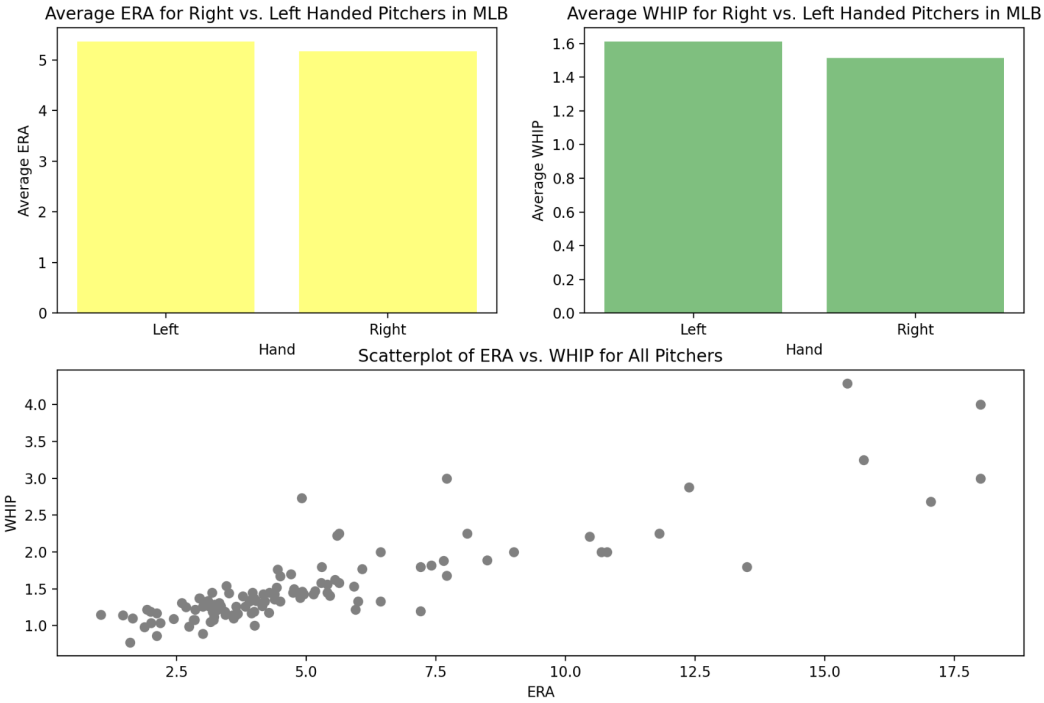


a.


Average Heights By Position for Players in Premier League
Average Heights (cm)
190
180
170
160
150
Attacker
Midfielder
Defender
Goalkeeper
Positions

b.

c.


Average ERA for Right vs. Left Handed Pitchers in MLB
Average ERA
Left
Right
Hand
Average WHIP for Right vs. Left Handed Pitchers in MLB
Average WHIP
Left
Right
Hand
Scatterplot of ERA vs. WHIP for All Pitchers
WHIP
ERA

6. Instructions for running code:
    a. For baseball data:
        i. Navigate to ‘Baseball-GetData.py’
        ii. Run the code
        iii. Wait 15 seconds before running again
        iv. Repeat steps 2-3 until total players is greater than 100 or equal to 100
            1. 

(ex)
        v. Now navigate to ‘Baseball-ReadData.py’
        vi. Run the code and see the visualizations!
    b. For basketball data:
        i. Navigate to ‘Basketball-GetData.py’
        ii. Run the code
        iii. Enter any NBA team name
        iv. Enter a page number 1-7
        v. Repeat steps 2-4 until all 7 pages have been added
        vi. Do this again for at least one other different team
        vii. Now navigate to ‘Basketball-ReadData.py’
        viii. On line 78, enter the list of teams that you added to the database into the second parameter
        ix. Run the code and see the visualization!
    c. For Soccer data:
        i. Navigate to ‘Soccer-GetData.py’
        ii. Run the code

iii. Wait 30 seconds before running again

iv. Repeat steps 2-3 until total players is greater than or equal to 100

1. 

(ex)

v. Now navigate to 'Soccer-ReadData.py'

vi. Run the code and see the visualization

7. Documentation for Each function:

a. All files have the function setUpDatabase(db_name)

```
def setUpDatabase(db_name):
    '''
    Create the database and return the cursor and connection
objects.
    Used in function to update databases
    '''
```

b. In 'Baseball-GetData'

i.

```
def create_id_list():
    '''
    Returns a list of all of the team ids from the MLB in the API
--> 30 teams
    Used in create_pitchers_list()
    '''
```

ii.

```
def create_id_table():
    '''
```

```
   Creates a table of player IDs
   Players only added to table if their position is pitcher
   Pitcher hand (0 or 1) is also a column in the table --> 0 is L,
1 is R
   10 new players added on each run by using a count
   '''
```

iii.

```
def create_table():
   '''
   Creates a table of Pitchers with there ERA and WHIP
   Pitchers only added to database if stats are available
(cleaning)
   Max of 10 added at a time (but could be less) by using
'max(id)'
   '''
```

iv.

```
def main():
   '''
   Main driver of file
   Adds 10 pitchers to IDs table
   Then takes 15 second break before cleaning
   Then cleans and adds the pitchers to Pitchers Table
   '''
```

c. In ‘Baseball-ReadData.py’

i.

```
def doCalc(filename):
   '''
   Calculates average ERA and WHIP based on hand
   JOIN used in SELECT statement to get hand of pitcher
   Input: json file that holds data
   Data then added and outputted to the as a new key
   '''
```

ii.

```
def showViz(filename):
   '''
```

```
   Two bar charts created
   One shows average ERA for Right vs. Left Hand
   Other shows average WHIP for Right vs. Left Hand
   Input is json file that holds the data needed for the bar
charts
   Output is the two bar charts
   '''
```

d. In ‘Soccer-GetData.py’

i.

```
def createPlayers():
   '''
   Create a players table with players from the EPL in the 2019
season
   - player_id is the id of that player given by the API
   - height is the height of the player in cm
   - position is a number that correlates to the players position
   - - 0(attacker), 1(midfielder), 2(defender), 3(goalkeeper)
   '''
```

e. In ‘Soccer-ReadData.py’

i.

```
def doCalc(filename):
   '''
   Do the Calculations:
   - Get the average height of each player by position (0-3)
   Input is json file that hold data
   New calculations then outputted to json file as new key
'Soccer'
   '''
```

ii.

```
def showViz(filename):
   '''
   Create the visual
   Bar chart of heights by different position
   Input is json file with the data
   Output is the visual
   '''
```

f. In ‘Basketball-GetData.py’

i.

```
def createGamesTable():
    '''
    Creates a table of all of the games from the 2017 and 2018
seasons for a team
    User Input is team name, must be spelled correctly or function
will inform you that the team does not exist
    User also inputs a page number 1-7
    Output is the table
    Max of 25 games added per time as we limit the amount of games
per API request to 25
    '''
```

g. In ‘Basketball-ReadData.py’

i.

```
def doCalc(filename, teams):
    '''
    Get average total game score for the team
    Input: json file that holds data
    Input: List of team names that have data for.. if data does not
exist or spelled wrong, function will inform you
    Data then added and outputted to the json file as new key
‘Basketball’
    '''
```

ii.

```
def showViz(filename):
    '''
    Create a bar chart that shows average total game score for the
teams
    team name on x axis
    average total game score on y axis
    Input is json file that holds the data needed for the bar
charts
    Output is the bar chart
    '''
```

8. Documentation of Resources

Date	Issue Description	Location of Resource	Result
4/21	When trying to convert to a float, some data not returned as a number and instead as '**.*'	https://docs.python.org/3/tutorial/errors.html	Solved
4/19	Understanding request params and data returned from different types of requests (for basketball and soccer API). Postman made it easy to mess around with different requests to get the data that we were looking for. Also on how to properly format requests	https://balldontliesi.postman.co/home	Solved
4/25	Creating three plots on one figure and changing their respective titles and position	https://stackoverflow.com/questions/37360568/python-organisation-of-3-subplots-with-matplotlib	Solved
4/24	All three APIs sometime returning faulty requests	https://appac.github.io/mlb-data-api-docs/ https://www.balldontlie.io/#introduction https://www.api-football.com/documentation-v3	Solved (but still sometimes get connection reset or ratelimited randomly)
4/24	Same issue as before	https://www.programiz.com/python-programming/time/sleep	Helped Solve issue
4/21	Problem with pushing and pulling	https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/overview	Solved