Adam Dinhofer
Ben Dinhofer
John Redfern
GitHub Link: https://github.com/bdinhofer/206-Final-Project/tree/main

Final Project Report

Project Goals:

The goal of this project is to obtain information from the nba-api, data-usa api and various web pages to make insights about trends in the National Basketball Association and the market size for sports franchises in the United States. Overall we are looking to specify if a team's value and market size are a good indicator for success. This will be accomplished by completing three separate sub-goals. First, we will analyze the newly adopted strategy of relying on the 3 point field goal as opposed 2 point field goals in the NBA. We will determine if this strategy is crucial for team success by comparing a team's total 3 point field goals made with their winning percentage for the 2019-2020 regular season. Our next goal is to verify if there is a correlation between a city's population and the value of its franchises. This will be accomplished by comparing city population with the average net-worth of its franchises in the NBA, MLB, NFL, and NHL. Lastly, we will create a graph visualizing each team's winning percentage with the location they play to see if their value determines their success.

Goals Achieved:

We were successful in accomplishing all 3 of our sub-goals. From, figure 1 we were able to see that NBA teams that were better at 3 point shooting tended to be better than teams that were worse at 3 point shooting. This gave us one way to measure a team's success. Figure 2 shows that there is a positive correlation between a city's population and their sports franchises' net worths. However, by cross-referencing the data points from figures 2 and 3 it is apparent that a team's value and market size do not correspond to success. Looking up a cities average net worth for their sports franchises from "cities_vis_data.csv" and comparing that to their win percentage displayed in figures 1 and 2 allows us to see that there is no correlation. Teams with larger market sizes and values are scattered across the league standings meaning these factors are not a good indicator of team success.

Problems:

There were multiple challenges we faced as a group throughout this project. First, the nba-api was complicated to use due to a limit on the amount of data you can pull in a certain amount of time. This was solved using the time module to stagger each individual call to the api. Furthermore, we had problems with the json response object returned from calls to the nba-api. Since a few players have apostrophes in their names, the string returned was unable to be processed by the json.loads() function. To solve this, we had to use a json formatter to easily search through the response and remove the apostrophes in these players names. Another problem we faced was finding a web page with information about the net-worth for sports teams from all 4 major sports. Hence, to gather this information we needed to use 4 separate web pages. Additionally, the information from these web pages was difficult to clean in the processing stage

of the project because the data was written in the style: “$4.5 billion”. We had to write code to convert strings of this style to their corresponding integer values in order to make calculations. This data was also difficult to match with their corresponding populations from the data-usa api because some teams have a different location in their name than the city that they play in (For example, the Arizona Cardinals play in Phoenix Arizona). We had to sort through these teams and match them with the city that they play in.

Data:

nba_vis_data.csv (text copied)

team,total3s,win%
Warriors,651,0.231
Cavaliers,727,0.292
Timberwolves,853,0.297
Hawks,805,0.299
Pistons,791,0.303
Knicks,631,0.318
Bulls,793,0.338
Wizards,859,0.347
Hornets,785,0.354
Pelicans,980,0.417
Kings,914,0.431
Spurs,760,0.451
Magic,807,0.452
Grizzlies,796,0.466
Suns,816,0.466
Trail Blazers,952,0.473
Nets,936,0.486
Mavericks,1136,0.573
76ers,848,0.589
Heat,979,0.603
Thunder,770,0.611
Rockets,1126,0.611
Jazz,963,0.611
Pacers,743,0.616
Nuggets,801,0.63
Celtics,905,0.667
Clippers,895,0.681
Lakers,781,0.732
Raptors,995,0.736
Bucks,1007,0.767

cities_vis_data.csv (text copied)

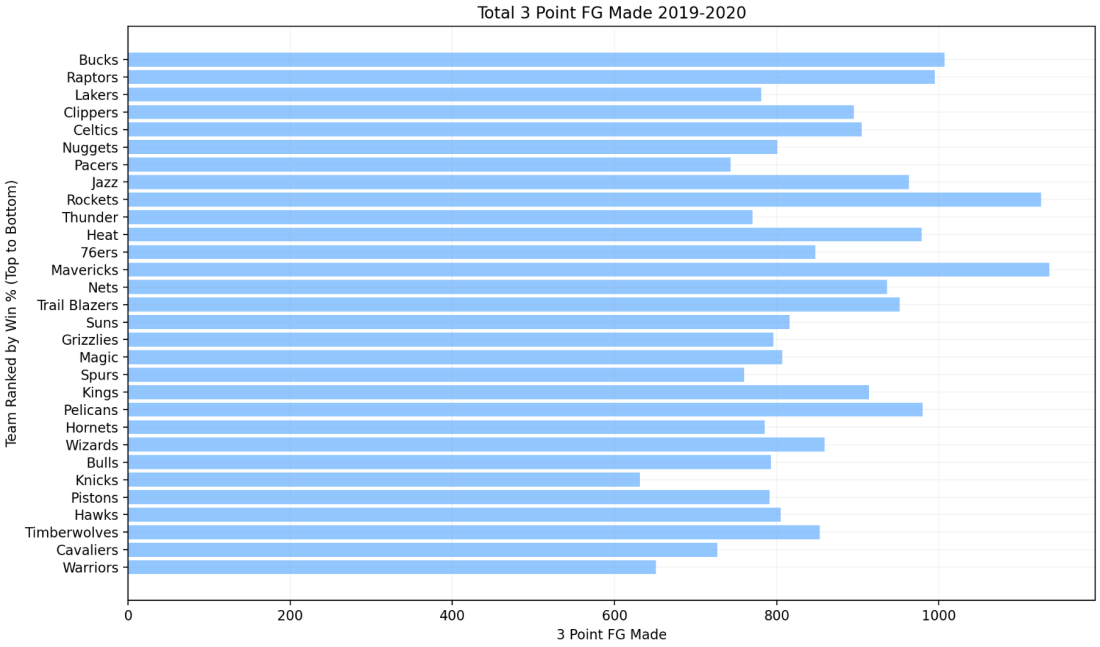
```
city,population,avgNetWorth
New York,8398748,3171250000.0
Los Angeles,3990469,2913571428.571429
Chicago,2705988,2591000000.0
Toronto,2503281,1775000000.0
Houston,2326090,2556666666.6666665
Phoenix,1660272,1411250000.0
Philadelphia,1584138,2087500000.0
San Antonio,1532212,1850000000.0
San Diego,1425999,1500000000.0
Dallas,1345076,2633750000.0
San Jose,1030119,540000000.0
Jacksonville,903896,2450000000.0
Columbus,895877,326000000.0
San Francisco,883305,3487500000.0
Charlotte,872506,1500000000.0
Seattle,744949,2352500000.0
Denver,716492,1656250000.0
Washington,702455,2000000000.0
Boston,695926,3016250000.0
Detroit,672681,1402500000.0
Portland,652573,1900000000.0
Memphis,650632,1300000000.0
Oklahoma City,649410,1575000000.0
Las Vegas,644664,2260000000.0
Baltimore,602495,2202500000.0
Milwaukee,592002,1422500000.0
Sacramento,508517,1825000000.0
Atlanta,498073,2090000000.0
Kansas City,491809,1780000000.0
Miami,470911,1963333333.3333333
Oakland,429114,2912500000.0
Tampa,392905,1268333333.3333333
New Orleans,391006,1912500000.0
Cleveland,383781,1690000000.0
Anaheim,352018,480000000.0
St. Louis,302838,1387500000.0
Cincinnati,302615,1542500000.0
```

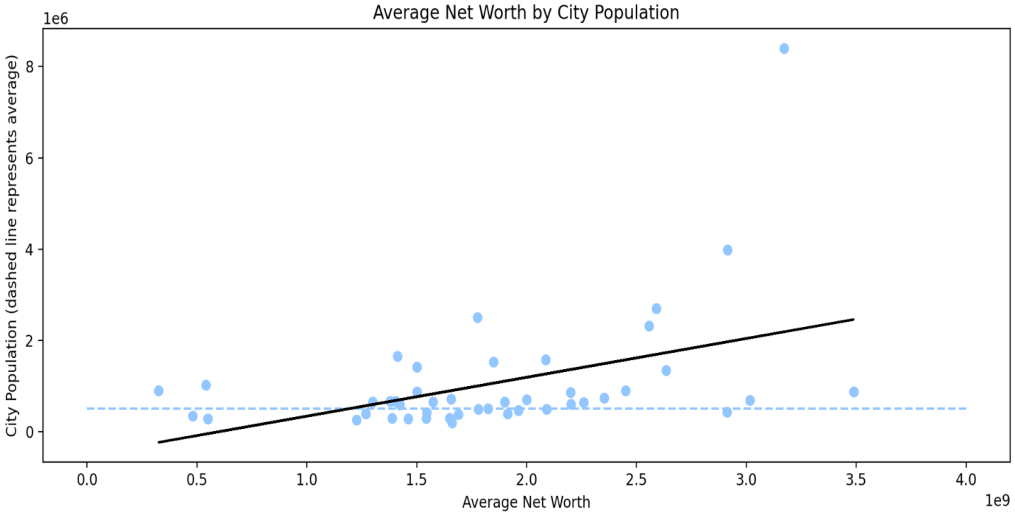
Pittsburgh,301038,1650000000.0
Orlando,285705,1460000000.0
Buffalo,256322,1225000000.0
Minneapolis,425395,1546250000.0
Indianapolis,864131,2200000000.0
Salt Lake City,200576,1660000000.0
Nashville,665498,1380000000.0
Newark,282102,550000000.0
Average City Population: 513088.76

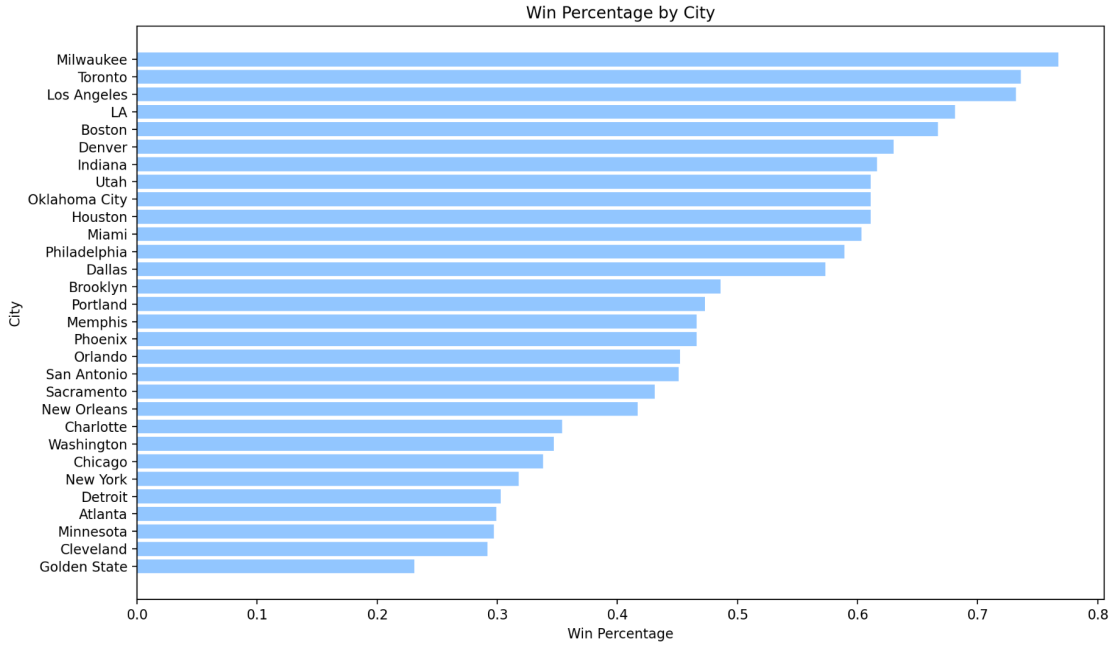
city,winpct
Golden State,0.231
Cleveland,0.292
Minnesota,0.297
Atlanta,0.299
Detroit,0.303
New York,0.318
Chicago,0.338
Washington,0.347
Charlotte,0.354
New Orleans,0.417
Sacramento,0.431
San Antonio,0.451
Orlando,0.452
Phoenix,0.466
Memphis,0.466
Portland,0.473
Brooklyn,0.486
Dallas,0.573
Philadelphia,0.589
Miami,0.603
Houston,0.611
Oklahoma City,0.611
Utah,0.611
Indiana,0.616
Denver,0.63
Boston,0.667
LA,0.681
Los Angeles,0.732
Toronto,0.736

Milwaukee,0.767

Visualizations:







Instructions:

1. Obtaining data from APIs and writing them to files
    a. Run the code in Data-USA-API.py to create the file “Cities.csv”
    b. Run the code in Web-Scrape.py to create the file “NetWorths.csv”
    c. Run the code in NBA-API.py to create the files “PLAYER_STATS.txt” and “TEAM_STATS.txt”.
        i. The main function takes approximately 20-25 minutes to run due to the limit on the rate of data extraction from the NBA-API
        ii. To ensure this code runs correctly do not let your computer shut off while this file runs
        iii. Once the program is finished copy and paste the contents from “PLAYER_STATS.txt” to the json formatter/validator at this link:
            1. https://jsonformatter.curiousconcept.com/#
        iv. Delete the original contents of “PLAYER_STATS.txt” Copy and paste the correctly formatted json to the file
        v. Then do a spotlight search for apostrophes in the file and remove any apostrophe within a player's name. There are only a few players with this problem.
        vi. This is done to correct errors in the json formatted string that is written to the file which allows the json.loads() function to work properly.
2. Add the data from the files to a Database
    a. Comment out the three function calls below create_player_table(d, cur, conn, 0)

i. Make sure the i parameter = 0
ii. Run the code and each time increase the i parameter by 25 until all 567 players are in the table

b. Repeat this for the three function calls below.
i. Make sure to comment out the function calls that are not in use.
ii. The nba_teams table should have 30 rows
iii. The cities table should have 150 rows
iv. The Net_Worths table should have 123 rows

3. Process data and create visualizations
   a. Run the code in NBA_vis.py to create figure 1 and the file "nba_vis_data.csv"
   b. Run the code in Cities.vis.py to create figures 2, 3 and the "file cities_vis_data.csv"

Documentation:

Data-USA-API.py

- get_city_populations():
  - Takes in no input. This Function makes calls to the Datausa API to grab a list of all of the cities in the United States and then returns a list of tuples. The first element in the tuple is the city name and the second element is the corresponding population
- get_pop_toronto():
  - Takes in no input. This Function uses the BeautifulSoup Library and web scraping to find Toronto's population from the following url:
    - https://en.wikipedia.org/wiki/List_of_the_100_largest_municipalities_in_Canada_by_population
  - This function is needed because the Toronto Raptors are an NBA team.
  - It returns a tuple where the first element is Toronto and the second is its population. This format is identical to the tuples in the list returned by get_city_populations().
- sort_pop_lst(lst, tup):
  - This function takes in a list of tuples, and another tuple with additional information. It appends the tuple to the list of tuples and then sorts the list by its tuples second element in descending order(in this case population). It returns the first 150 tuples in the list, therefore returning the 150 most populated cities in the United States and Toronto.
- write_to_file():
  - Takes in no input. This function uses the previous three functions to create the final list of tuples containing the city name and population. It then writes the

information in the list to a csv file titled “Cities.csv” for later use. It doesn’t return anything

Web-Scrape.py

- most_valuable_NBA(url):
  - This function takes in the url.
    - https://www.forbes.com/sites/kurtbadenhausen/2021/02/10/nba-team-values-2021-knicks-keep-top-spot-at-5-billion-warriors-bump-lakers-for-second-place/?sh=37469da645b7
  - It then uses the BeautifulSoup library to scrape each nba team and their corresponding net worth. It puts this information into a list of tuples where the first element is the team name and the second element is the team’s net worth.
- most_valuable_NHL(url):
  - This function takes in the url:
    - https://www.forbes.com/sites/mikeozanian/2020/09/10/the-nfls-most-valuable-teams-2020-how-much-is-your-favorite-team-worth/?sh=57f46abe2ba4
  - This function should do the same thing as most_valuable_NBA but with NFL teams
- most_valuable_NHL(url):
  - This function takes in the url:
    - https://dailyhive.com/vancouver/nhl-team-values-forbes-2019
  - This function should do the same thing as most_valuable_NBA but with NHL teams
- most_valuable_NBA(url):
  - This function takes in the url:
    - https://brobible.com/sports/article/most-valuable-mlb-teams-2021-yankees-redsox/
  - This function should do the same thing as most_valuable_NBA but with MLB teams.
- create_list(lst1, lst2, lst3, lst4,):
  - This function takes in 4 separate lists and uses .extend to combine them into one list. It returns the combined list.
- write_to_file():
  - Takes in no input. This function utilizes all the previous functions from the file to create a list of tuples of the four major sports leagues in the United States where the first element is the team name and the second element is the teams net worth. It then writes this data to a csv file titled “NetWorths.csv for later use. It Doesn’t return anything.

NBA-API.py

- get_player_stats():

    - Takes no input. This function makes calls to the nba-api to retrieve the statistics for all the active players in the NBA for the 2019-2020 season. It returns the response in a json format and writes the response to the file "PLAYER_STATS.txt".
- get_team_stats():
    - Takes no input. This function makes calls to the nba-api to retrieve the statistics for every team in the NBA for the 2019-2020 season. It returns the response in a json format and writes the response to the file "TEAM_STATS.txt"
- main():
    - Takes no input. Calls the functions get_player_stats() and get_team_stats(). Returns None.

create-db.py

- set_up_db(db_name):
    - Takes a string that acts as the filename for a new database as input. This function creates a new sqlite3 database file and returns an sqlite3 database cursor object and a connection object in the variables cur and conn, respectively.
- read_Data_From_File(filename):
    - Takes .txt filename as input. The file should contain a string that is json formatted and returns this exact string but as a json object.
- read_list_from_file(filename):
    - This function takes in a csv filename and returns a list of all of the lines of data in the file.
- create_team_table(data, cur, conn, i):
    - The first input, data, must be a json object. Cur and conn must be sqlite3 cursor and connection objects. The 'i' parameter is a user entered number that tracks iterations through data to limit the number of database entries added to 25. Every time the file "Create-db.py" is run the user should increase the value of i by 25 (starting with 0). This function adds all 30 NBA teams to the database table "nba_teams" when it is run twice. Each row consists of the following columns: id, team, city, winpct. Returns None.
- create_player_table(data, cur, conn, i):
    - This function works the same as create_team_table() but adds all 567 NBA teams to the database table "nba_teams" when it is run 23 times. Each row consists of the following columns: team_id, name, threes
- create_city_table(lines, cur, conn, i)
    - This function takes in the lines from a file as a list, an sqlite3 database cursor object, a connection to that database, and the i parameter. This function works the same as create_team_table(data, cur, conn, i), however, it takes data from the

"Cities.csv" file instead of a json object. Each row should include "city_name" and "population". The table should be titled "cities". It returns None.

- create_Net_worth_table(lines, cur, conn i):
    - This function works the same as create_city_table. It should use data from the "NetWorths.csv" file and each row in the table should consist of "team_name" and "net_worth". The table should be titled "Net_Worths". It returns None.
- main():
    - Takes no input. This function calls the functions in the file "create-db.py" to create the database "Final-Data.db" and close the connection object. Returns None.

NBA_vis.py

- grab_data(cur, conn):
    - Takes in an sqlite3 cursor and connection object to connect to the "Final-Data.db" database. This function returns a list of tuples with desired data from the tables nba_players and nba_teams.
- process_data(stats):
    - Takes in stats, the list returned from grab_data(). This function returns 3 lists. A list of team names, a list of the total number of 3 point field goals made for each team in the 2019-2020 season and a list of each team's win percentage in the 2019-2020 season.
- bar_graph(x, y):
    - Takes in two lists (x and y) that contain the data it will plot on the appropriate axis. This function returns None but plots a horizontal bar graph with total 3 point field goals made on the x axis for each team on the y axis. The teams are graphed in order by decreasing win percentage from top to bottom
- write_data_to_csv(filename, x, y, wins):
    - Takes a .csv filename and 3 lists as input. This function returns None but writes the contents of the lists to the csv file.
- main():
    - Takes no input. This function establishes a connection to the database "Final-Data.db" then calls the functions in "NBA_vis.py" to create the graph described in bar_graph() and the file "nba_vis_data.csv". Then it closes its connection to the database. Returns None.

Cities_vis.py

- convert_str(conn, cur):
    - This function takes in an sqlite3 database cursor object and a connection to the database. Then it converts all of the net worth value strings formatted "$4 billion" or "$990 million" into the correct integer value. It returns a list of tuples with the team name as the first element and the net worth as an integer in the second element.

- get_in_city_dict(lst, conn, cur):
    - This function takes in a list of tuples, an sqlite3 database cursor object and a connection to the database. It returns a dictionary where the keys are cities that have sports teams and values that are a list of all the teams that play in that city.
- find_not_city(clean_lst, teams_in_city):
    - This function takes in the list created from convert_str(conn, cur) and the dictionary created from get_in_city_dict(lst, conn, cur). It returns a list of all the sports teams that do not use their city in their name. For example, Arizona Coyotes or Golden State Warriors.
- Change_team_name(not_in_cities, teams_in_city):
    - This function takes in the list of teams returned from find_not_city and the dictionary returned from get_in_city_dict. It appends each team to the list of teams that play in a city in the dictionary. It returns the dictionary with these teams now accounted for.
- get_population(conn, cur)
    - This function takes in an sqlite3 database cursor object and a connection to the database. It returns a list of tuples where the first element is a city and the second is the city's population.
- get_average_population(lst):
    - This function takes in a list of tuples where the first element is a city and the second element is the city's population. It returns the average population of all the cities in the list
- get_average_net(teams_in_city):
    - This function takes in the dictionary returned from Change_team_name (not_in_cities, teams_in_city). It returns a dictionary where the keys are the same as the dictionary it takes as input and the values lists where the only element is the average net worth for the teams that play in that city.
- add_pop(pop_lst, d):
    - This function takes in the list returned from get_population(conn, cur) and the dictionary returned from get_average_net(teams_in_city). It returns the same dictionary with populations added to the list values of the corresponding city.
- get_viz_lists(d):
    - This function takes in the dictionary returned from add_pop(pop_lst, d). It returns three separate lists. The first is a list of populations from the dictionary, the second is a list of average net worths from the dictionary and the third is a list of each city in the dictionary.
- Scatter plot(x, y, z, lst)
    - This function takes in the three lists returned from get_viz_lists(d) and the list returned from get_population(conn, cur). It creates a scatter plot with a cities average team net worth on the x-axis and the cities population on the y-axis. It

should include a line of best fit to show the correlation and a dashed line to represent the average population. It returns nothing.

- get_win(conn, cur):
    - This function takes in an sqlite3 database cursor object and a connection to the database. It returns two separate lists. The first is a list of cities, and the second is a list of that city's NBA team winning percentage. Each list should be sorted by winning percentage in descending order.
- bar_graph(x, y):
    - This function takes in the lists returned from get_win(conn, cur). It creates a horizontal bar graph with cities on the y-axis and their winning percentage on the x-axis.
- write_data_to_csv(filename, x, y, z, x2, y2, num):
    - Takes a .csv filename and 5 lists as input. This function returns None but writes the contents of the lists to the csv file. This function also writes the average population of the top 150 U.S. cities (stored in num) to the csv file.
- main():
    - Takes in no input. This function establishes a connection to the database "Final-Data.db" and runs all of the functions in the file to clean and organize the data from the database. It also creates the scatter plot and bar graph. It then writes the processed data from the database to the csv file titled "cities_vis_data.csv". Finally, it closes the connection to the database.

Resources:

Date	Issue Description	Location of Resource	Result
4/23/21	Nba-api returned incorrectly formatted json	https://jsonformatter.curiousconcept.com/#	Nicely formatted json string allowing us to remove apostrophes
4/20/21	Struggling with nba-api documentation	Youtube: https://www.youtube.com/watch?v=NCyPY-jfb3I	Much better understanding of nba-api
4/25/21	Did not know how to plot line of best fit	https://www.kite.com/python/answers/how-to-plot-a-line-of-best	Learned how to plot a line of best fit using the numpy library

		-fit-in-python	
4/20/21	Forbes had a paywall to access a fourth article	https://brobible.com/sports/article/most-valuable-mlb-teams-2021-yankees-redsox/	Was able to retrieve information about the networth of MLB teams
4/20/21	No Issue -Link to API	https://github.com/swar/nba_api	Able to retrieve statistics about NBA teams and players
4/20/21	No Issue -Link to API	https://datausa.io/about/api/	Able to retrieve data about city population
4/20/21	No Issue -Link to webpage	https://dailyhive.com/vancouver/nhl-team-values-forbes-2019	Able to retrieve data about net worth of NHLteams
4/20/21	No Issue -Link to webpage	https://www.forbes.com/sites/mikeozanian/2020/09/10/the-nfls-most-valuable-teams-2020-how-much-is-your-favorite-team-worth/?sh=57f46abe2ba4	Able to retrieve data about networth of NFL teams
4/20/21	No Issue -Link to webpage	https://www.forbes.com/sites/kurtbadenhausen/2021/02/10/nba-team-values-2021-knicks-keep-top-spot-at-5-billion-warriors-bump-lakers-for-second-place/?sh=37469da645b7	Able to retrieve data data net worth of NBA teams
4/19/21	The datausa API did not provide information on Toronto	https://en.wikipedia.org/wiki/List_of_the_100_largest_municipalities_in_Canada_by_population	Retrieved Toronto's population by web-scraping wikipedia