SI 206 Final Project Report

a. What is your group's name?

CS Lads

b. Who are the people in the group (first name, last name, umich email)?

Megan Su meganysu@umich.edu

Gunhaar Panjwani gunhaar@umich.edu

Vasundhara Kulkarni kulkvasu@umich.edu

c. What APIs/websites will you be gathering data from? The base URLs for the APIs/websites must be different for them to count as different APIs.

Basketball API:
Keys: 78776eefec674c20ea9530f0e4d0b9cd or 41425335f242d64ef15bcafaf42eaf20
https://api-sports.io/documentation/basketball/v1#section/Introduction

Football API:
https://info.soccerfootball.info/

```
'X-RapidAPI-Key': '8083a405acmsh6a6520e04afa316p171d36jsn6bd22ce85d62',
 'X-RapidAPI-Host': 'soccer-football-info.p.rapidapi.com'
```

NFL APL: (key: 22dc5861296e4a77857b2909edcd7fa4)
https://api.sportsdata.io/v3/nfl/scores/json/Standings/2021?key=22dc5861296e4a77857b2909edcd7fa4

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

The goals for our project were to analyze the correlation between goal difference / net points scored and the rank at the end of the season for multiple sports, in order to determine the impact of these metrics on the teams' performance. We intended to work with Basketball, Football (Soccer), and NFL APIs to demonstrate the correlation between standings and net points. Since for-points and against-points are provided in the APIs for each team within each division, we planned to calculate the net points by subtracting points/goals for subtracted by points/goals against for each team in the sports. For football, we gathered data from the 20/21 season and spread it across 6 leagues, for Basketball we used 19/20 season and spread it across 6 leagues, and for NFL we used a single league and 4 seasons — allowing us to obtain at least 100 data points per sport.

Basketball API: (https://api-sports.io/documentation/basketball/v1#section/Introduction)

Football API: (https://info.soccerfootball.info/)

NFL API: (https://api.sportsdata.io/)

By gathering data from these APIs, we aimed to study the relationship between goal difference / net points scored and the final rank of teams in various leagues and seasons. We will use this to help us understand the importance of these metrics in predicting team performance and success.

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

The goal of our project is to show the correlation between the standings and total points. We utilized three sport APIs: Basketball, Football (Soccer), and the NFL (website linked above). We planned to extract the for-points (the number of points a team has scored during a game, match, or tournament), against-points (number of points that have been scored by the opposing team), final position, conference name, and team name in order to determine the correlation. For Basketball and Football API we exclusively focused on the year of 2019-2020 and tracked multiple leagues and teams from various countries. For the NFL, since there is only one league per season, we tracked the one league and teams across the seasons of 2019-2022. We found the point differential by calculating the difference between for-points and against-points for all our sport teams. The point differential is used as a metric to measure the overall performance of a team. Typically a positive point differential indicates that a team is scoring more points than it is giving up, while the negative is the opposite. Ultimately a high point differential is considered to be a sign of a strong team. For all our various leagues and conferences, we calculated the point differential to see the relationship of how a higher standing is associated with a higher point differential value. We were able to graph plots that displayed the positive correlation between higher standings and point differentials to visually represent our analysis.

3. The problems that you faced (10 points)

The majority of our problems came from finding free APIs that had complete information about each sport in a manner that could help us. Many APIs stored partial, incomplete, or sometimes not useful information, making it difficult for us to gather enough datapoints to analyze. However, in the end, after tremendous amounts of research and testing, we were able to find three APIs that provided us with enough information to analyze and draw conclusions from.

Another difficulty was team scheduling. With conflicting schedules in a group of three, it was difficult to coordinate timings to work on the project. So, we decided to split the majority of the project, with each of us taking a leading role on one sport and helping the others when needed. We communicated well and were able to complete our assignments on our own for the most part, however, when required, we were able to show compromise and meet, despite time conflicts, to help each other out.

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

Football (Soccer) Calculations:

```
Calculation: Goal Difference = Goals For - Goals Against      You, 26 minutes ago • basketball done …

Premier League Data:
Goal Difference: [51, 29, 26, 22, 18, 15, 23, 16, 8, -1, 9, -16, -16, -25, -21, -6, -22, -26, -41, -43]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Bundesliga Data:
Goal Difference: [55, 28, 29, 24, 16, 14, 7, 8, 1, 0, -2, -17, -18, -11, -26, -26, -21, -61]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

LaLiga Data:
Goal Difference: [42, 39, 47, 20, 21, 0, 16, -2, -18, 4, -11, -22, -3, -11, -15, -21, -21, -19, -23, -23]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Serie A Data:
Goal Difference: [54, 45, 31, 45, 36, 8, 10, 6, -2, -2, -11, -11, -12, -16, -20, -16, -19, -35, -47, -44]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Ligue 1 Data:
Goal Difference: [41, 58, 34, 38, 7, 12, 1, -2, -3, -4, -12, -14, -18, -8, -9, -18, -16, -8, -31, -48]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Eredivisie Data:
Goal Difference: [79, 39, 34, 14, 28, 11, 3, 1, -11, -2, -8, -6, -9, -28, -22, -28, -48, -47]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
```

Basketball Calculations:

```
Calculation: Point Difference = Points For - Points Against

Liga A:
Regular Season: [142, 148, 186, 124, 201, 78, 76, 53, 4, 46, -115, 9, -153, 28, -25, -65, -53, -161, -198, -325]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

NBL - Regular Season:
Regular Season: [170, 120, 40, 78, 50, 46, -114, -90, -300]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9]

WNBL Women:
Regular Season: [137, 200, 85, 79, -11, -139, -178, -173]
Position: [1, 2, 3, 4, 5, 6, 7, 8]

National League - Regular Season:
Regular Season: [624, 102, 46, -48, -293, -431]
Position: [1, 2, 3, 4, 5, 6]

Premier League:
Regular Season: [432, 392, 142, 66, -455, -577]
Position: [1, 2, 3, 4, 5, 6]

NBA:
Western Conference: [411, 464, 154, 213, 178, 143, 371, -85, -78, 16, -79, -147, -93, -275, -566]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Atlantic: [449, 454, 174, -41, -426]
Position: [1, 2, 3, 4, 5]

Southwest: [213, 371, -78, -79, -93]
Position: [1, 2, 3, 4, 5]

Pacific: [411, 464, 16, -147, -566]
Position: [1, 2, 3, 4, 5]

Northwest: [154, 143, 178, -85, -275]
Position: [1, 2, 3, 4, 5]

Central: [736, 143, -200, -235, -513]
Position: [1, 2, 3, 4, 5]
      You, 56 minutes ago • first …
Eastern Conference: [736, 449, 454, 143, 215, 174, -41, -74, -439, -336, -200, -426, -235, -534, -513]
Position: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Southeast: [215, -74, -439, -336, -534]
Position: [1, 2, 3, 4, 5]
```

NFL Calculations:

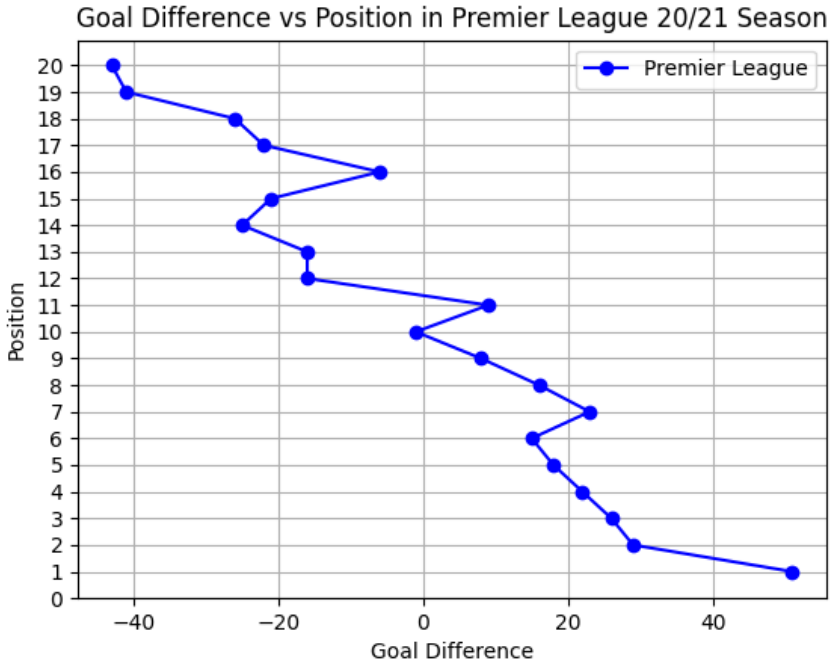
```
Calculation: Win Percentage = Wins / (Wins + Losses) and Net Points = For Points - Against Points

2019:
East
Win Percentage: [0.75, 0.625, 0.438, 0.312, 0.562, 0.5, 0.25, 0.188]
Net Points: [195, 55, -83, -188, 31, 113, -110, -169]

North
Win Percentage: [0.875, 0.5, 0.375, 0.125, 0.812, 0.625, 0.5, 0.2]
Net Points: [249, -14, -58, -141, 63, 104, -18, -82]

South
Win Percentage: [0.625, 0.562, 0.438, 0.375, 0.812, 0.438, 0.438, 0.312]
Net Points: [-7, 71, -12, -97, 117, 9, -18, -130]

West
Win Percentage: [0.75, 0.438, 0.438, 0.312, 0.812, 0.688, 0.562, 0.333]
Net Points: [143, -34, -106, -8, 169, 7, 30, -81]

2020:
East
Win Percentage: [0.812, 0.625, 0.438, 0.125, 0.438, 0.375, 0.375, 0.267]
Net Points: [126, 66, -27, -214, 6, -77, -78, -84]

North
Win Percentage: [0.75, 0.688, 0.688, 0.267, 0.812, 0.5, 0.438, 0.312]
Net Points: [104, 165, -11, -113, 140, 2, -45, -142]

South
Win Percentage: [0.688, 0.688, 0.25, 0.062, 0.75, 0.688, 0.312, 0.25]
Net Points: [89, 52, -80, -186, 145, 137, -52, -18]

West
Win Percentage: [0.875, 0.5, 0.438, 0.312, 0.75, 0.625, 0.5, 0.375]
Net Points: [111, -44, -42, -123, 88, 76, 43, -14]
```

```
2021:
East
Win Percentage: [0.647, 0.588, 0.529, 0.235, 0.706, 0.529, 0.412, 0.235]
Net Points: [194, 159, -32, -194, 172, 59, -99, -158]

North
Win Percentage: [0.588, 0.562, 0.471, 0.471, 0.765, 0.471, 0.353, 0.188]
Net Points: [84, -55, -22, -5, 79, -1, -96, -142]

South
Win Percentage: [0.706, 0.529, 0.235, 0.176, 0.765, 0.529, 0.412, 0.294]
Net Points: [65, 86, -172, -204, 158, 29, -146, -100]

West
Win Percentage: [0.706, 0.588, 0.529, 0.412, 0.706, 0.647, 0.588, 0.412]
Net Points: [116, -65, 15, 13, 88, 83, 62, 29]

2022:
East
Win Percentage: [0.812, 0.529, 0.471, 0.412, 0.824, 0.706, 0.562, 0.5]
Net Points: [169, -2, 17, -20, 133, 125, -6, -22]

North
Win Percentage: [0.75, 0.588, 0.529, 0.412, 0.765, 0.529, 0.471, 0.176]
Net Points: [96, 35, -38, -20, -3, 26, -1, -137]

South
Win Percentage: [0.529, 0.412, 0.25, 0.188, 0.471, 0.412, 0.412, 0.412]
Net Points: [54, -61, -138, -131, -45, -15, -27, -21]

West
Win Percentage: [0.824, 0.588, 0.353, 0.294, 0.765, 0.529, 0.294, 0.235]
Net Points: [127, 7, -23, -72, 173, 6, -77, -109]
```

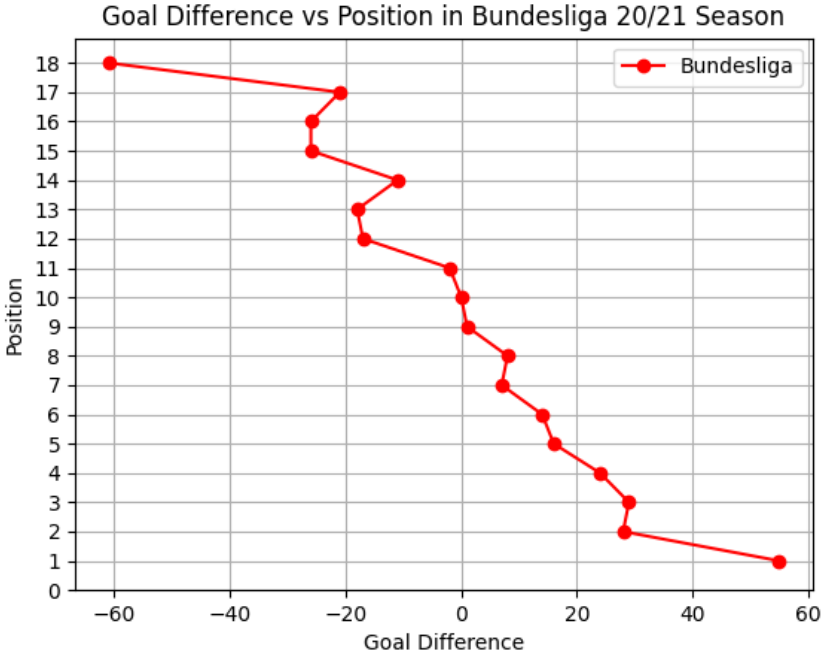
5. The visualization that you created (i.e. screen shot or image file) (10 points)

Football (Soccer) Visualizations:

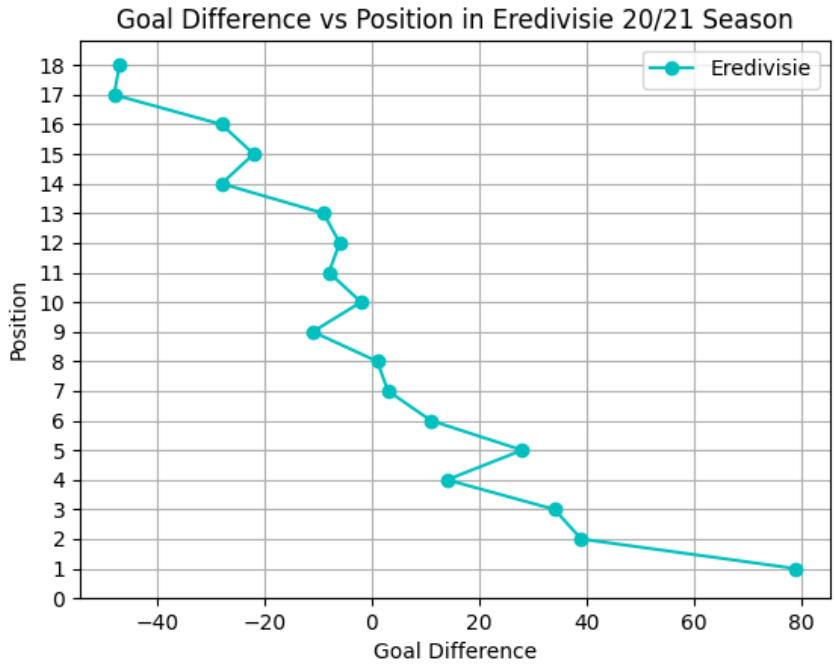
Premier League:



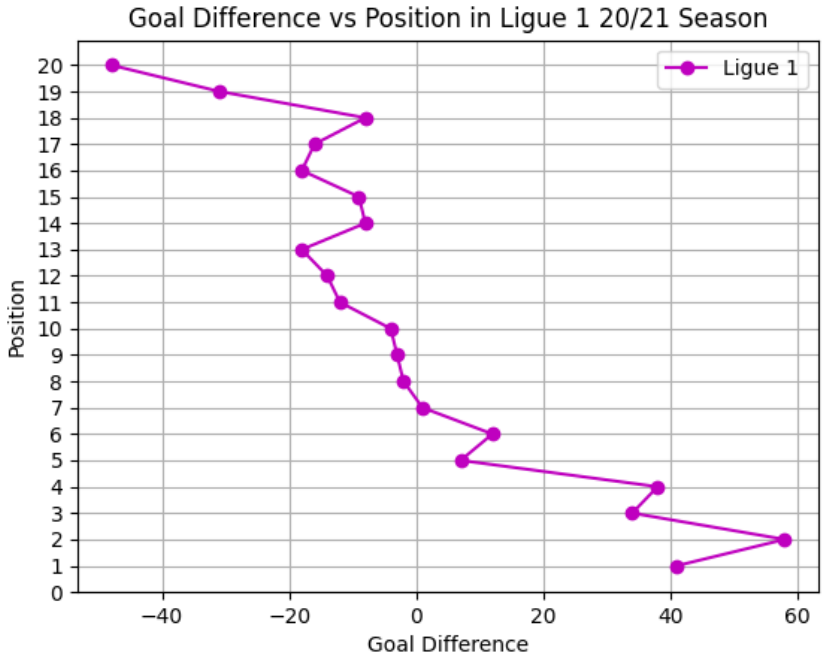
Bundesliga:



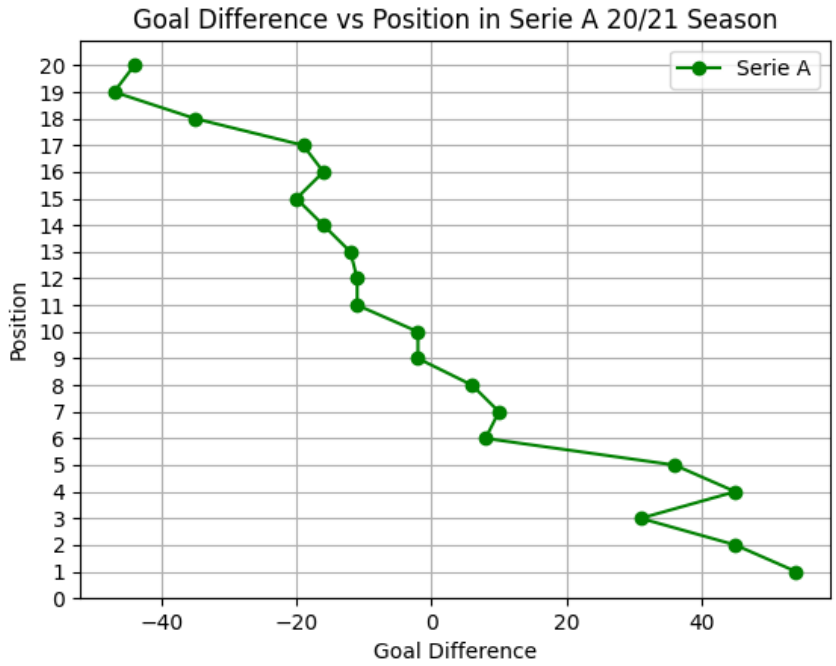
Eredivisie:



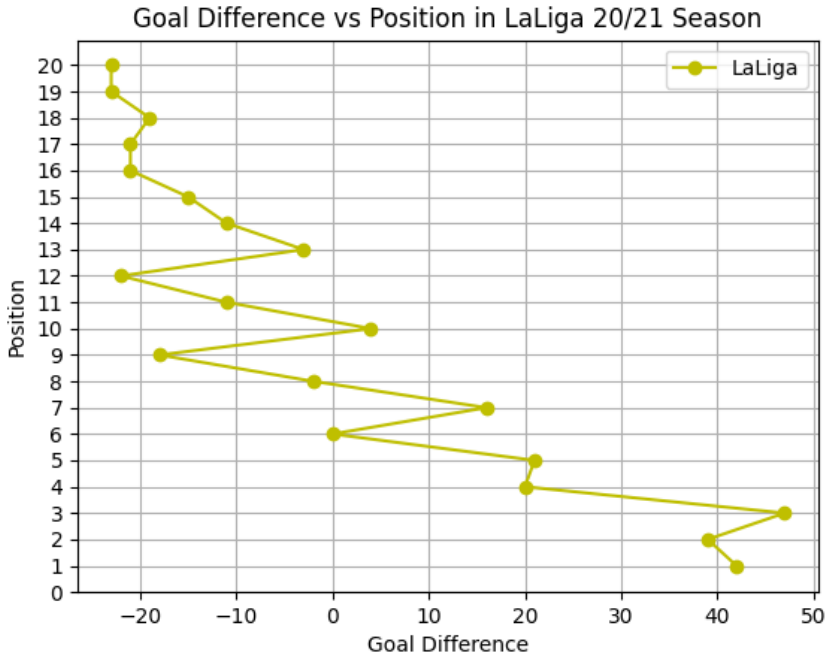
Ligue 1:



Serie A:

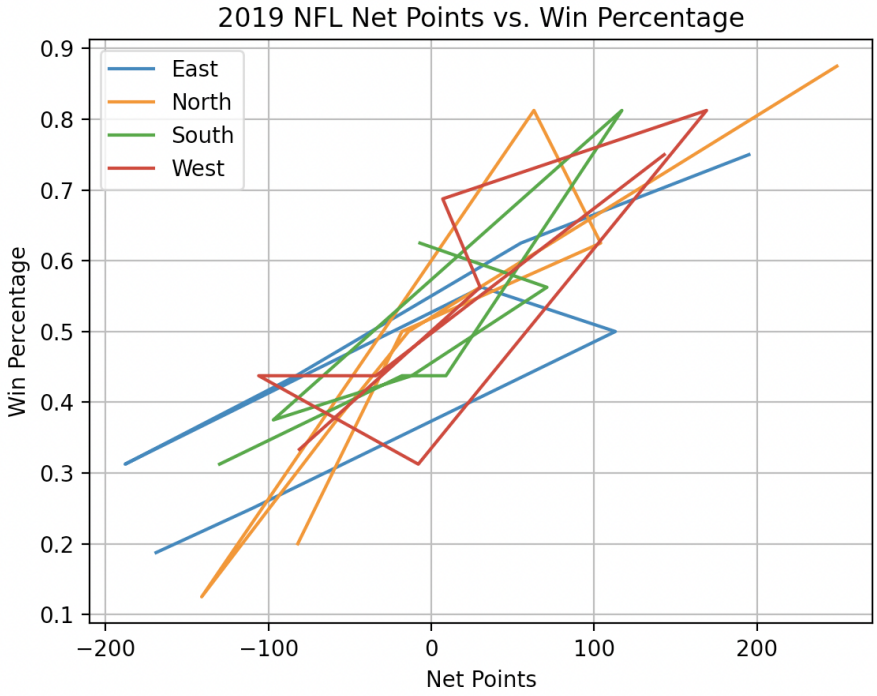


La Liga:

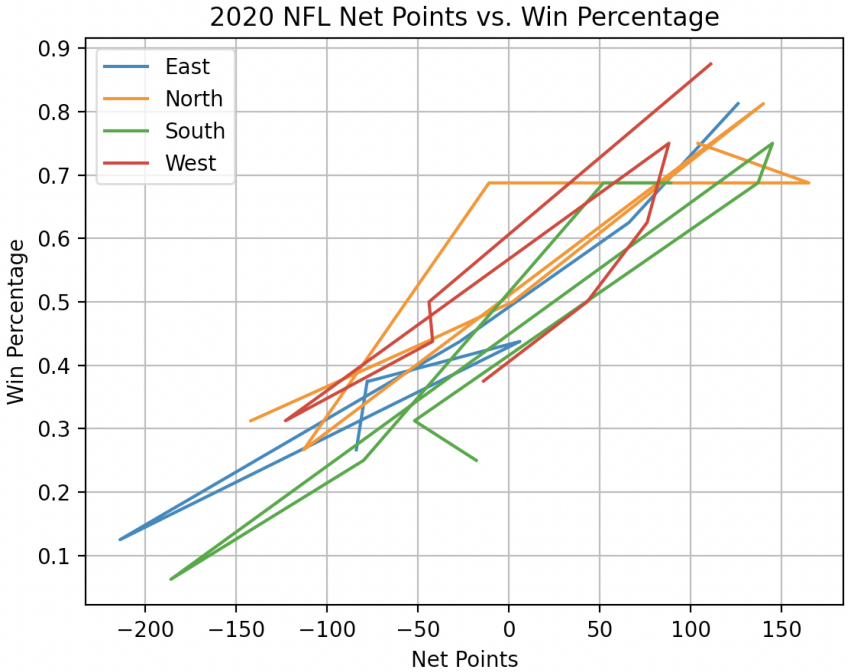


NFL Visualizations:

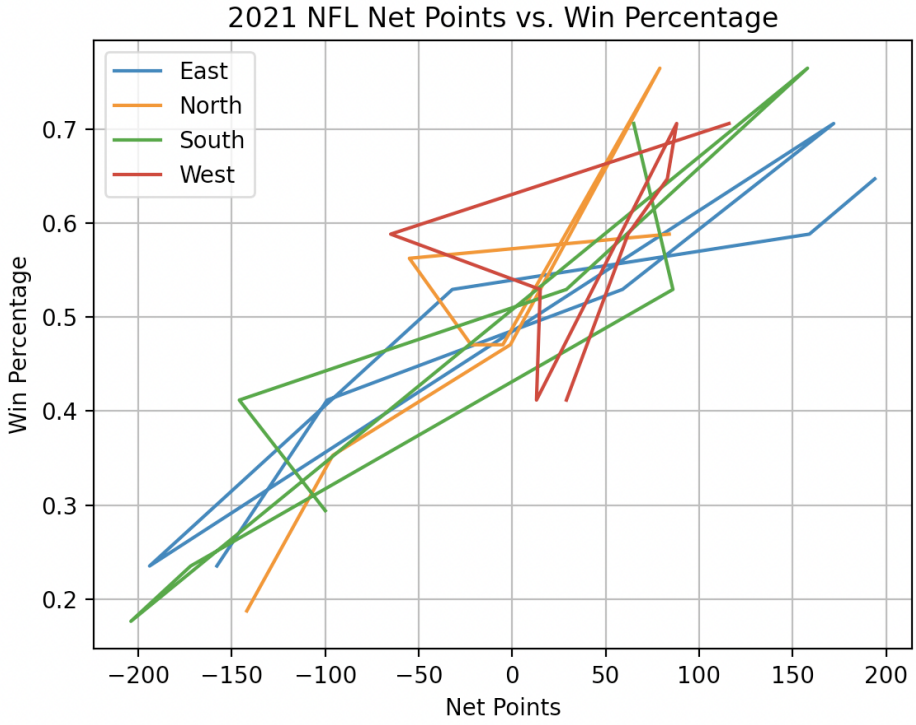
2019:



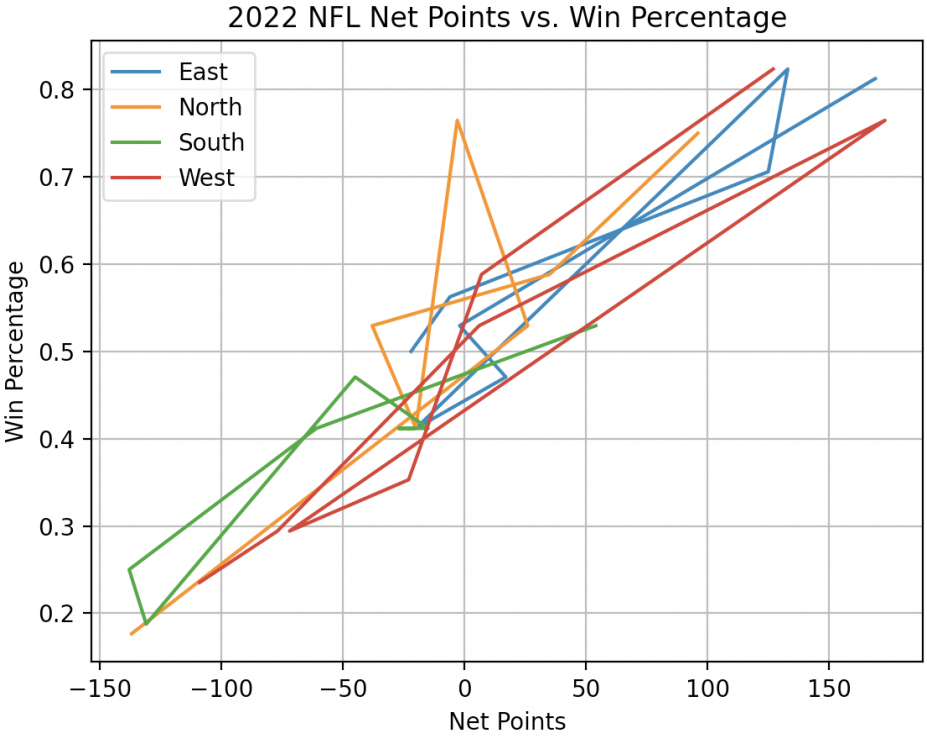
2020:



2021:

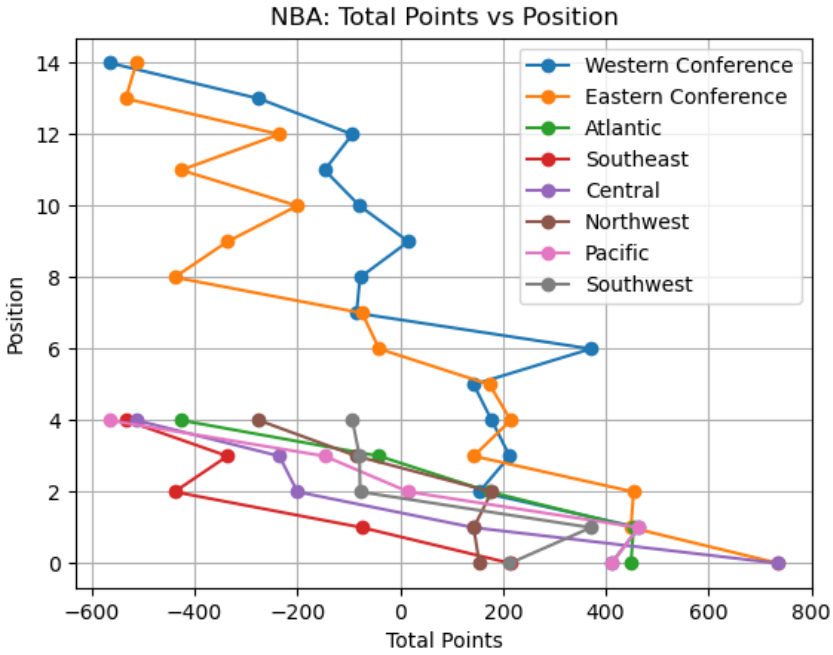

2021 NFL Net Points vs. Win Percentage
East
North
South
West
Win Percentage
0.2
0.3
0.4
0.5
0.6
0.7
−200
−150
−100
−50
0
50
100
150
200
Net Points

2022:

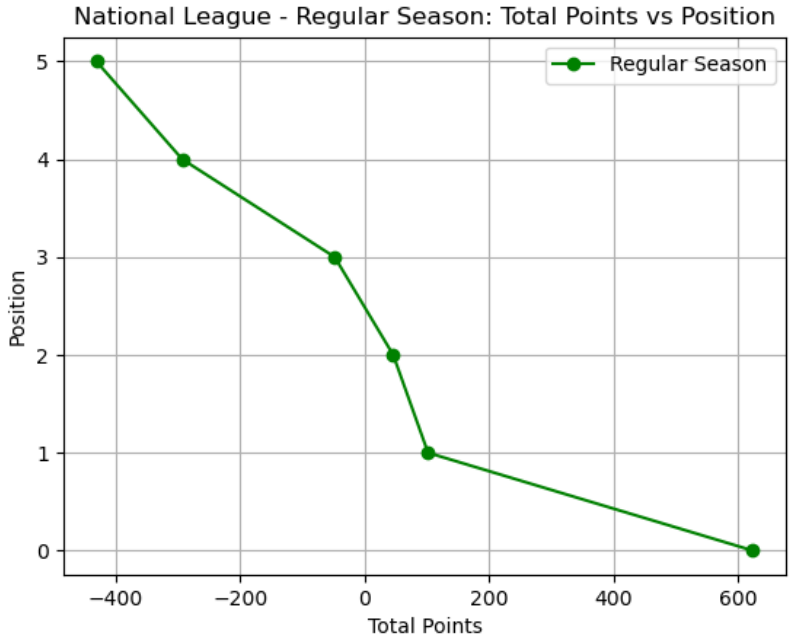

2022 NFL Net Points vs. Win Percentage
East
North
South
West
Win Percentage
0.2
0.3
0.4
0.5
0.6
0.7
0.8
−150
−100
−50
0
50
100
150
Net Points

Basketball Visualizations:

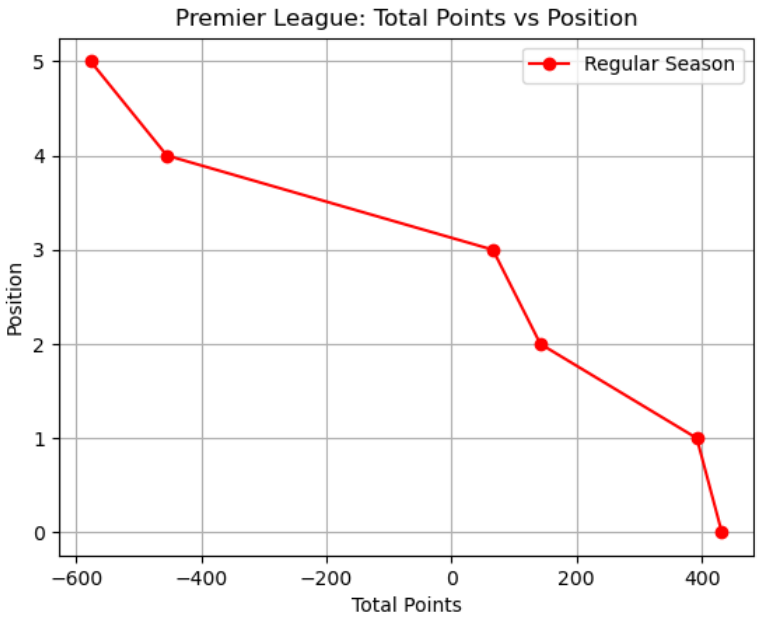
NBA:



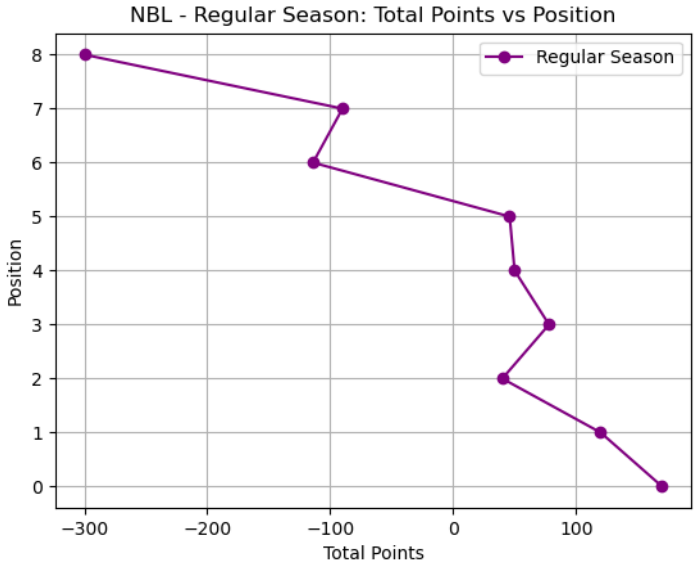
National League:



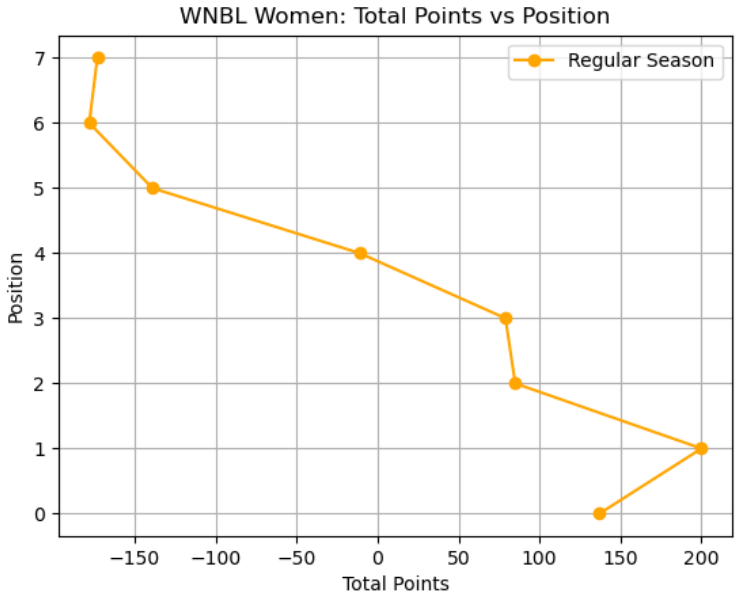
Premier League:



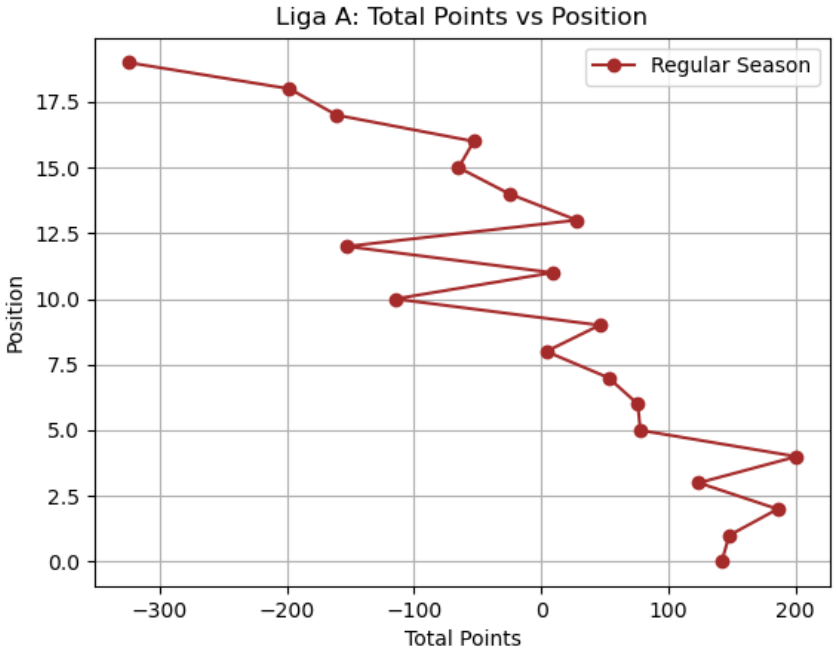
NBL League:



WNBL:


WNBL Women: Total Points vs Position
Regular Season
Position
Total Points
7
6
5
4
3
2
1
0
−150
−100
−50
0
50
100
150
200

Liga A:


Liga A: Total Points vs Position
Regular Season
Position
Total Points
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−300
−200
−100
0
100
200

6. Instructions for running your code (10 points)

WARNING: The API's we used can only take 100 requests per day

Football (Soccer):

1. Run football_tables.py: Creates League Database (Name: Football Leagues)
2. Run the football_<league>.py files in any order, where league abbreviation can be (PL, LL, L1, BL, ED, SA): Creates League Table Database (Name: Football)
3. Run football_data.py: Creates txt file of calculations and graphs

Basketball:

1. Run the basketball_<league>.py files in any order, where league can be (NBA_Liga, NBA_NBL_WNBL, NBA_Prem_Ntl, NBA_SE_CT_NW_PC_SW, NBAest&atlantic, NBAWestern) which respectively represent (Liga A, National Basketball League & Women's National Basketball League, Premier League, NBA: Southeast, Central, Northwest, Pacific, Southwest Conferences, NBA: Eastern & Atlantic Conferences, and NBA: Western Conference). This will create the database entries.
2. Run basketball_tables.py to produce text file of calculations and graphs

NFL:

1. Run nfl-2019.py
2. Run nfl-2020.py
3. Run nfl-2021.py
4. Run nfl-2022.py
5. Run nfl-data.py

7. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)

Football (Soccer):

football-tables.py:

Function 1: open_database

Description:

This function opens a SQLite3 database connection and returns a cursor and connection object.

Input:

db_name (str): Name of the SQLite database file.

Output:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Function 2-7: get_(prem, bundes, french, spanish, italian, dutch)_data

Description:

This function fetches the respective league's data from the API and inserts it into the Football_Leagues table in the database.

Input:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Output: None

Function 8: make_football_table

Description:

This function creates the Football_Leagues table in the database if it does not already exist.

Input:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Output: None

Function 9: main

Description:

This is the main function that calls all other functions. It connects to the "sports.db" database, creates the Football_Leagues table, and fetches data for each league to insert into the table.

Input: None

Output: None

football_(BL, PL, L1, LL, ED, SA).py:

Function 1: open_database

Description:

This function opens a SQLite3 database connection and returns a cursor and connection object.

Input:

db_name (str): Name of the SQLite database file.

Output:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Function 2: get_football_data

Description:

This function fetches the respective league's data from the API, specifically the 20/21 season standings and scores, and returns a dictionary containing the relevant information.

Input:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Output:

standings_dict (dict): A dictionary containing team names as keys and a nested dictionary containing their id, position, goals scored ('for'), goals conceded ('against'), and league id.

Function 3: make_football_table

Description:

This function creates the 'Football' table in the database if it does not already exist and inserts the Serie A data into it.

Input:

data (dict): The dictionary containing the respective league's team data.

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Output: None

Function 4: main

Description:

This is the main function that calls all other functions. It connects to the "sports.db" database, fetches the respective league's football data, and creates the 'Football' table with the fetched data.

Input: None

Output: None

football_data.py:

Function 1: open_database

Description:

This function opens a SQLite3 database connection and returns a cursor and connection object.

Input:

db_name (str): Name of the SQLite database file.

Output:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Function 2-7: process_(pl, bl, sa, l1, ed, ll)

Description:

These functions query the database to obtain the fields: position, goals scored, and goals conceded for teams in each league (Premier League, Bundesliga, Serie A, Ligue 1, Eredivisie, and LaLiga). The functions each return a tuple containing two lists - one for positions and the other for goal differences.

Input:

cur (sqlite3.Cursor): SQLite3 cursor object for executing commands.

conn (sqlite3.Connection): SQLite3 connection object for committing changes.

Output:

Tuple containing:

pos_list (list): A list of positions for each team in the respective league.

gd_list (list): A list of goal differences for each team in the respective league.

Function 8-13: draw(PL, Bundesliga, LaLiga, SerieA, Eredivisie, Ligue 1)Data

Description:

These functions create graphs showing the relationship between goal difference and position for each league using the data provided. The graphs are saved as PNG images.

Input:

data (tuple): A tuple containing two lists - one for positions and the other for goal differences.

Output: None

Function 14: print_to_file

Description:

This function writes the calculated data (goal differences and positions) for each league to a text file named "football.txt".

Input:

pl_data, bd_data, ll_data, sa_data, l1_data, ed_data (tuples): Tuples containing two lists - one for positions and the other for goal differences for each league.

Output: None

Function 15: main

Description:

This is the main function that calls all other functions. It connects to the "sports.db" database, processes the data for each league, writes the data to a text file, and creates the graphs.

Input: None

Output: None

NFL:

nfl-{season}.py

open_database("sports.db")

Description: This method takes in the name of the database, which we referred to as sports.db, as an input. It initializes a path, which becomes an input to create a conn, and later a cur. The function returns cur and conn, which are then passed to another function.

get_nfl_data()

Description: This method has no inputs, and it takes a url with the API subscription key as a header. Response is initialized by utilizing the requests library's get function, which takes in the url, as well as the header. Then, the json() method takes a Response stream and reads it to completion. Season is initialized, and an empty dictionary is created, which will later be returned. The method loops through standings, initializing name, wins, losses, pointsFor, pointsAgainst, and teamID based on the value within the json file. A dictionary for each unique season is created, and within each division, a dictionary for each team is created, which holds wins, losses, for, against, and id values. After looping through the entirety of the json file, the method returns the standings_dict dictionary.

make_nfl_table(data, cur, conn)

Description: The inputs of this method are data, which was the dictionary returned by get_nfl_data(), as well as cur and conn, which are both outputs of open_database. The method creates an NFL table, setting the id as the primary key, and adding in to columns season, division, name, wins, losses, for, and against. Then, it loops through each season within data, each division within data[season], and each team within data[season][div], inserting the row for the respective team if it has not already been inserted.

nfl-data.py

getData()

Description: The method initializes a path and creates conn and cursor before selecting all the columns from the NFL table in sports.db. It initializes two dictionaries, data and points, then iterates through each row, and if the particular entry is not in the table, it calculates and inserts the win percentage (which is rounded down to three decimal points using the round function) and net points into the data and points dictionaries, respectively. Once it has appended the values for each season, division, and team, the method returns data and points, which will be passed into another function later.

file(data, points)

Description: This method takes in data and points as inputs, initializing a base_path, which is utilized to initialize a full_path, writing into nfl-results.txt. The txt file has a header of the calculations, as well as headers for the particular season and division before writing the win percentages and net points into two distinct arrays.

plot(data, points)

Description: This method takes in data and points as inputs, which were initialized by the getData() method. It iterates through each season and division within data, creating a plot for each season, where it graphs the net points in the x-axis and the win percentage in the y-axis. A legend is added to the graph, as well as a title, grid, and axes labels. Base_path and full_path are both initialized, enabling each graph to be saved as a separate png within the folder.

Basketball:

basketball_{league}.py:

Function 1: open_database(db_name)

Input: db_name: the name of the SQLite database file to be opened

Output: *A tuple containing a cursor object and a connection object for the database*

Description: The input required for this function is an SQLite database file, which in this project is “sports.db”. The function creates a connection to the database utilizing the sqlite module in Python. It returns a tuple which contains a connection object and cursor object, which are then passed to another function. The cursor object can be used to execute SQL commands on the database, and the connection object can be used to commit changes to the database or to close the connection.

Function 2: get_basketball_data():

Input: None

Output: A nested dictionary called standings_dict containing the Basketball League Conference standings data for the 2019-2020 season.

Description: get_basketball_data fetches the standings data for the Basketball League Conference teams in the 2019-2020 season from an external API and returns the data in the form of a nested dictionary. The function defines an API endpoint url and makes a GET request to get a response in the form of a JSON object to contain information of the leagues such as team name, conference, league, points, etc.

Function 3: make_basketball_table(data, cur, conn):

Input: data: A nested dictionary containing Basketball League Conference standings data for the 2019-2020 season.cur: A cursor object that represents a database cursor. conn: A connection object that represents a database connection.

Output: None

Description: This function creates a table in a SQLite database and inserts the basketball data provided into it.

Function 4: main():

Input: None

Output: None

Description: The main() function is called automatically, which in turn calls the open_database(), get_basketball_data(), and make_basketball_table() functions.

basketball_tables.py:

Function 1: open_database(db_name):

Input: db_name: the name of the SQLite database file to be opened

Output: A tuple containing a cursor object and a connection object for the database

Description: The input required for this function is an SQLite database file, which in this project is “sports.db”. The function creates a connection to the database utilizing the sqlite module in Python. It returns a tuple which contains a connection object and cursor object, which are then passed to another function. The cursor object can be used to execute SQL commands on the database, and the connection object can be used to commit changes to the database or to close the connection.

Function 2: getData(cur, conn):

Input: cur: a cursor object used to execute SQL commands on a database, conn: a connection object used to connect to a database

Output: data: a dictionary containing the point differentials for each team in each conference of the NBA.

Description: This function retrieves data from the Basketball table of the sports.db database. It selects the columns conference, name, forPoints, againstPoints, and calculates the point differential by subtracting againstPoints from forPoints. It then loops through the retrieved rows and stores them in a dictionary:

Function 3: printToFile(data):

Input: A dictionary data containing basketball team and conference data.

Output: None (the function writes the data to a file).

Description: The printToFile function takes in the data dictionary, which contains the calculated point differences for each team in each conference. It writes a formatted output to a text file called "dataResults.txt" which includes the calculation method for goal difference, and then for each conference and each team in the conference, it writes the team name followed by its goal difference, and then a list of positions for the team.

Function 4: plot_data(data):

Input: A dictionary containing basketball data organized by conference and league. The dictionary has the total points and position for each team in each league.

Output: Plots each conference, showing the relationship between total points and position for each league. The plots are saved as PNG files in the working directory.

Description: The plot_data function takes dictionary data as input and creates a plot for each conference in the dictionary. For each conference, the function plots the total points versus position for each league within the conference. The position is determined by the rank or position of the team based on their point differential (calculated as for-points for minus against-points). The plot is saved as a PNG file with the conference name as the filename.

Function 5: main():

Input: None

Output: None

Description:

The main() function is the starting point of the program. It opens a connection to the database file "sports.db" using the open_database() function and gets the data from the database by calling the getData() function. It then passes the data to the printToFile() function and the plot_data() function to write the calculated results to a file and create plots respectively.

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 Result	Result
4/11	Finding APIs	All files taking data	We used: https://rapidapi.com/. To help us find all our APIs, directing use to appropriate websites and helping us test endpoints.
4/11	Football (Soccer) API	All Football files	We used this link for our Basketball API: (https://api-sports.io/documentation/basketball/v1#section/Introduction)
4/11	Basketball API	All Basketball files	We used this link for our Football API: (https://info.soccerfootball.info/)
4/11	NFL API	All NFL files	We used this linke for our NFL API: (https://api.sportsdata.io/)
4/11	Difficult to read JSON file while working on NFL portion	nfl-data.py	https://jsonformatter.org/json-pretty-print Solved the issue, formatted the file to make it extremely legible
4/13	Unable to figure out how to add gridlines on matplot graphs	nfl-data.py	Was able to add gridlines to the graphs (used lecture slides from canvas)
4/16	Unable to figure out how to add markers to graph	basketball_tables.py	Use: https://www.geeksforgeeks.org/how-to-add-markers-to-a-graph-plot-in-matplotlib-with-python/

			Was able to figure out how to add a circle marker to our line graphs