Detroit's Biggest Fans
Final Project Report

1. For our project, we wanted to find out which sport (between basketball, baseball and hockey) had the players with the highest weights on average. Additionally, as visual learners, we also wanted to be able to visualize the results in a clear way with colors to differentiate teams and correlate to the sport.
2. We achieved all of the goals we set out for ourselves and more. We found the weights of players in the NBA, MLB, and NHL and their averages and we were able to compare them in a database and via histogram using matplotlib. We went beyond our goals by also being able to visualize the number of players on each team in the NHL data we gathered using a shared key between two tables.
3.

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
4/15	Originally, we planned on comparing weights using only Detroit's sports teams (hence our team name). However, after gathering data, we realized we did not have enough numbers to complete the assignment correctly using only Detroit's teams.	No resource, we decided to expand within our APIs and evaluate players from other teams besides Detroit.	Our solution was to open up our data to more teams in the leagues. By adding more players to the database for each sport, we were able to analyze over 100 players for each sport.
4/19	Another issue we faced was inserting our data into the database and figuring out how to only add 25 items to the database at a time for the MLB API specifically. We gathered data from three separate team ids and added	https://pynative.com/python-mysql-insert-data-into-database-table/	This solved our issue perfectly, because when we ran the code after this then 24 pieces of data were added to the database at once and then we kept running it more times until it reached 120 players and their weights in

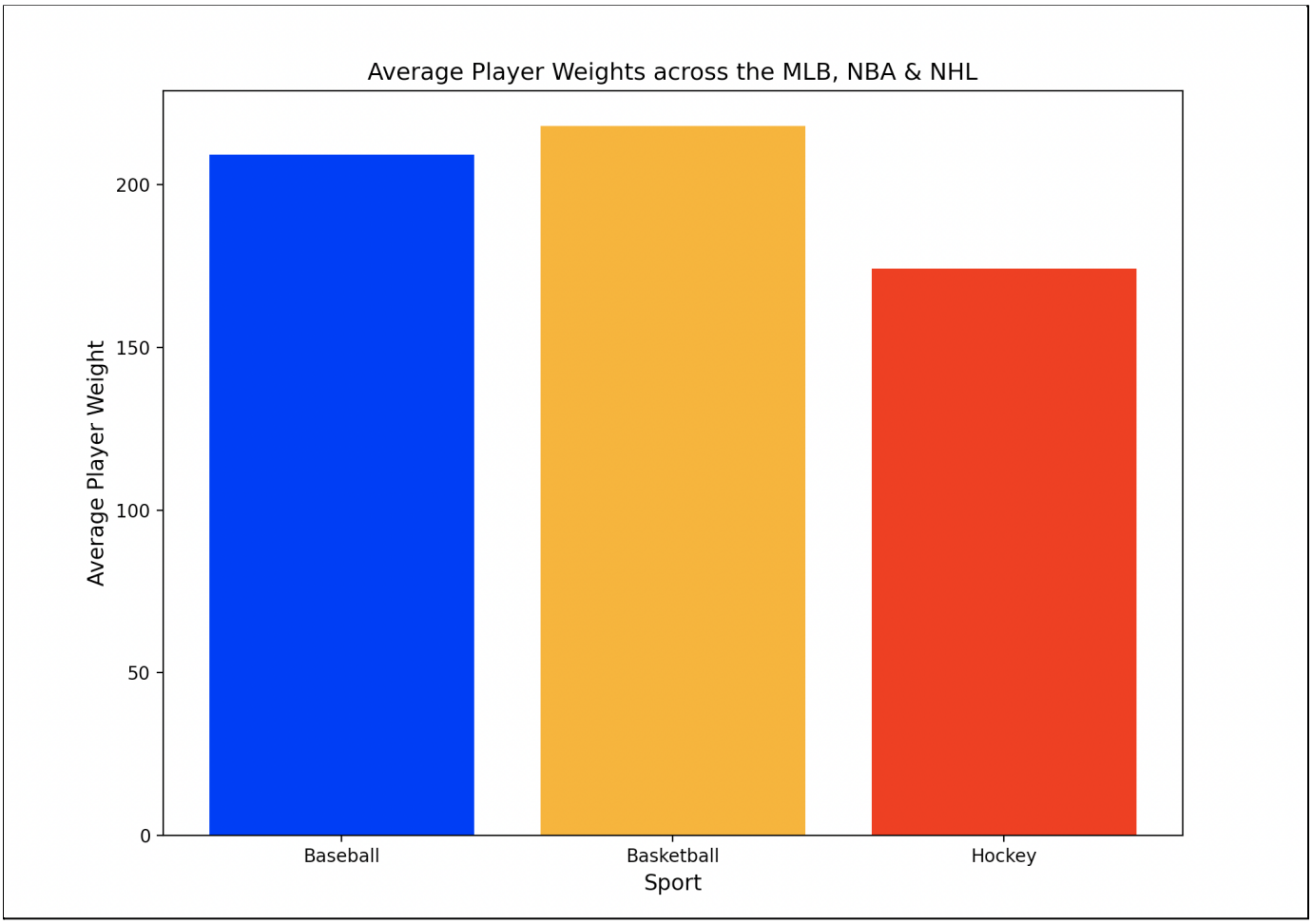
	them at the same time to the database so in order to add less than 25 at once, we had to limit each team id to 8 at a time, totaling 24 pieces of data added to the database at a time.		the MLB table.
4/20	When creating the histograms, we ran into trouble outlining the bins for a clearer visualization. At first, we manually added vertical and horizontal lines to the NBA histogram but the same method proved harder and more tedious for the next to histograms when it was difficult to locate exact x and y axis coordinates for each line.	https://www.tutorialkart.com/matplotlib-tutorial/matplotlib-pyplot-bar-plot-edge-color/	We knew there must be a better way to add in outlines to the bins so we researched online and found the 'edgecolor' parameter which allowed us to make the histograms much more user friendly and visible in an easy way.
4/24	When writing out our calculations to a csv file, we ran into the issue of the row overwriting and replacing the previous writerow() statement.	https://www.geeksforgeeks.org/how-to-append-a-new-row-to-an-existing-csv-file/	We found that instead of using with open (filename, 'w') as file, we should use with open (filename, 'a') as file.

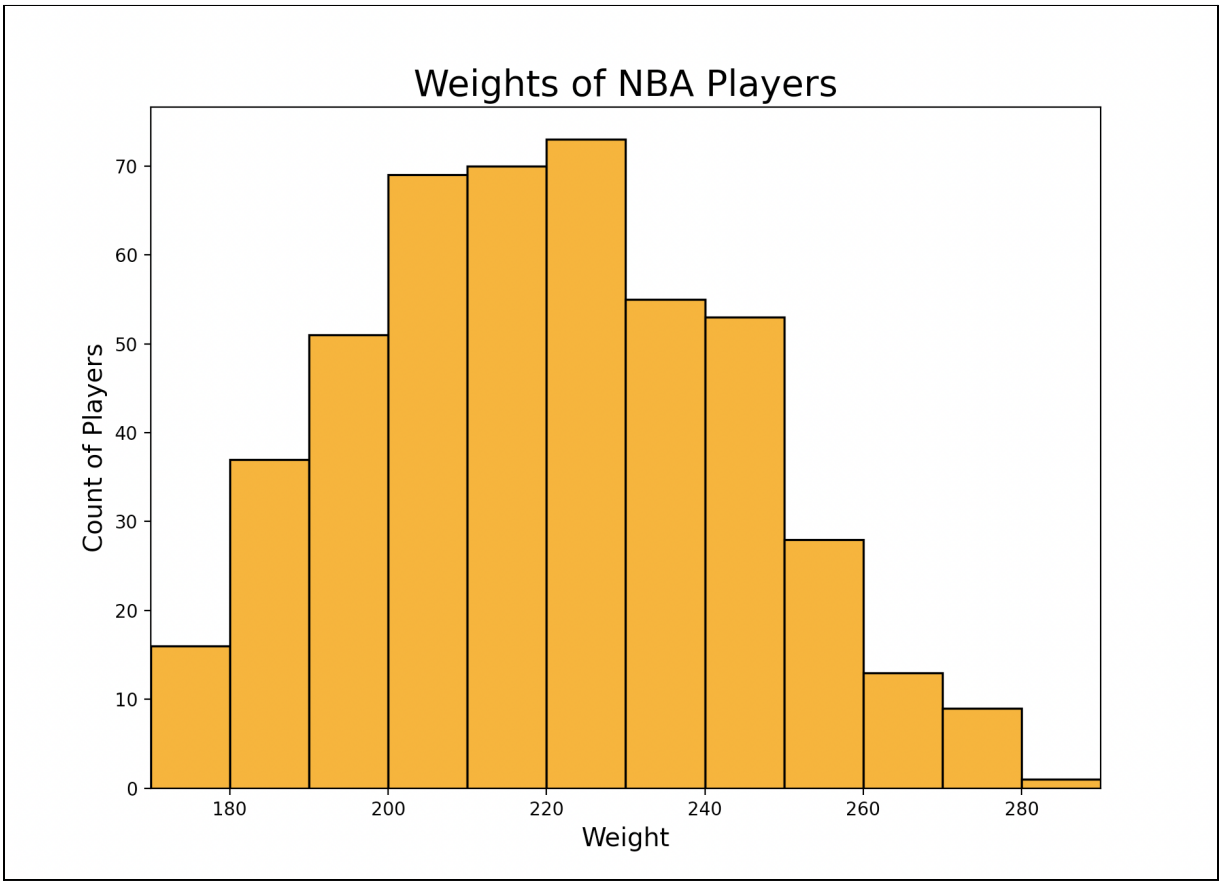
4. Our file that contains the calculations from the data in the database:

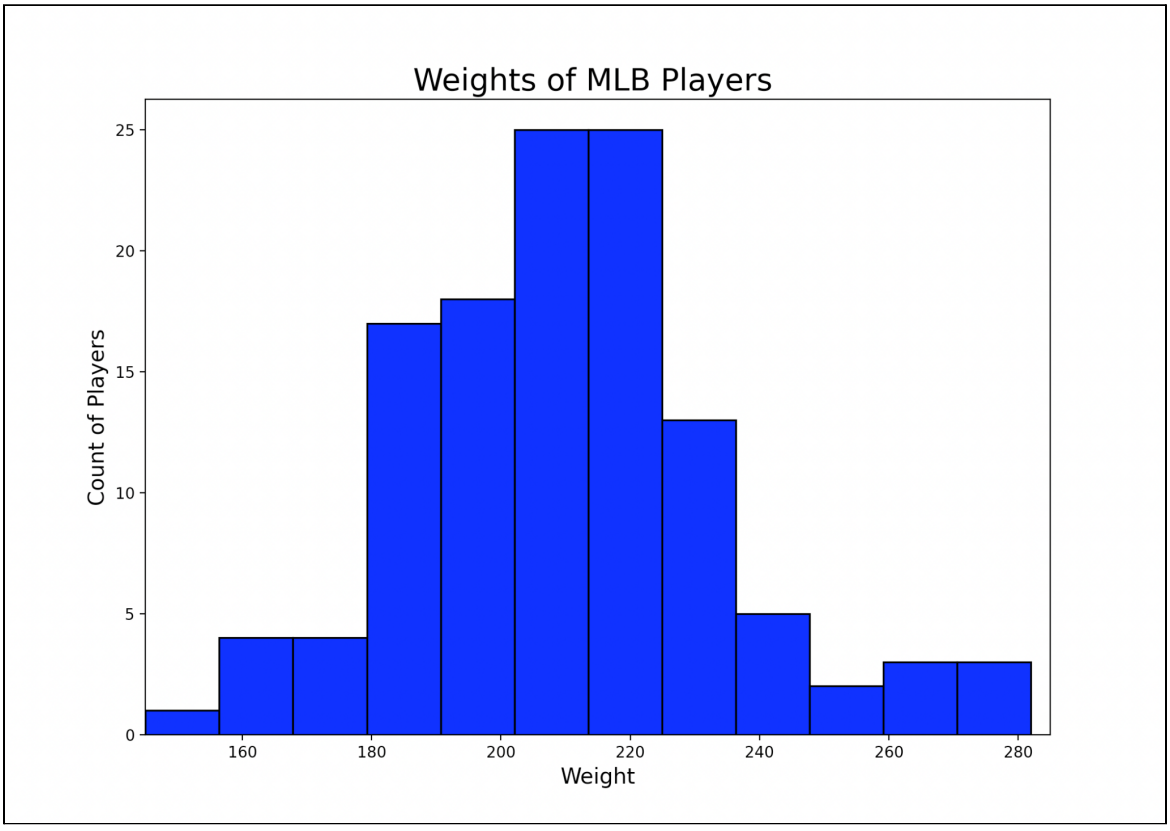
calculations.txt

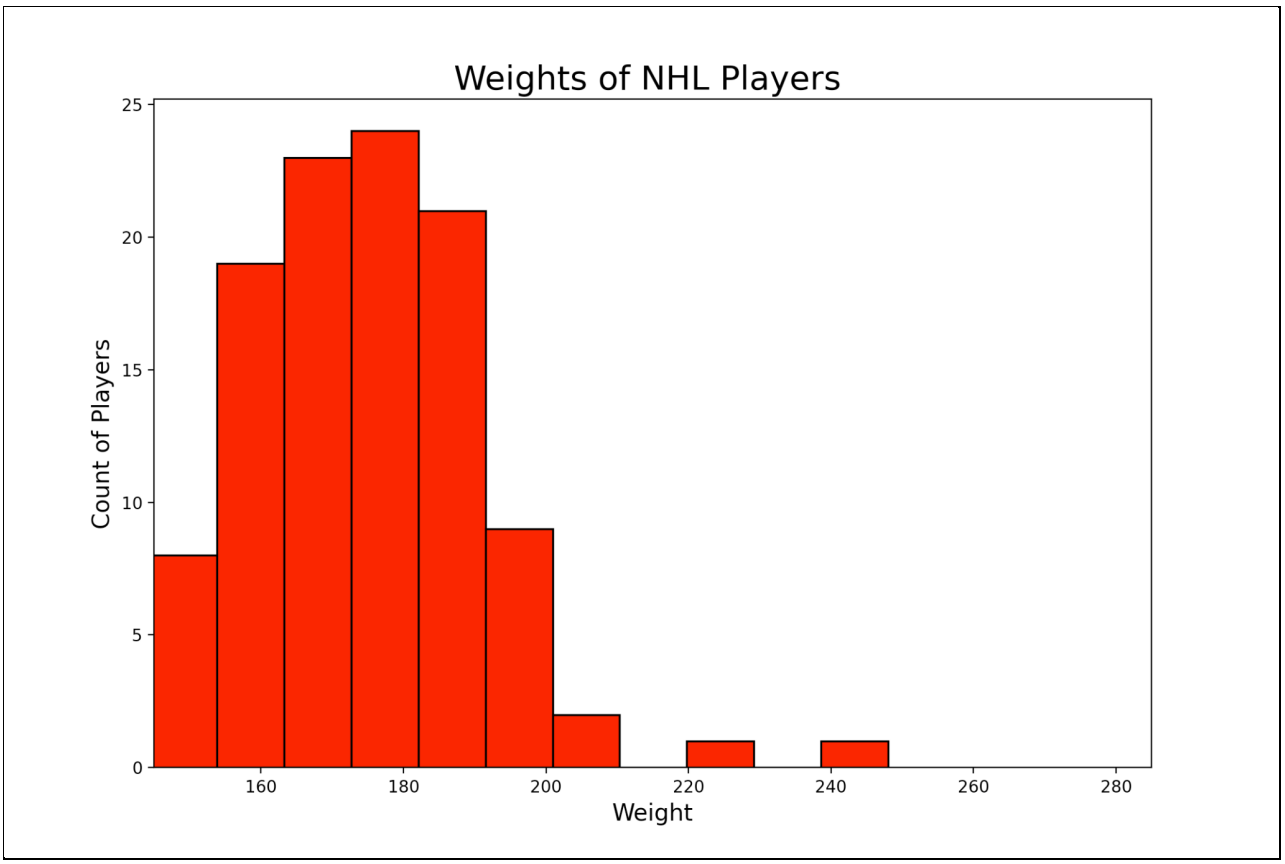
```
The minimum and maximum weights of NBA players in the Detroit_NBA table,"(170, 290)"
The average weight of the NBA players in the Detroit_NBA table,217.97684210526316
The minimum and maximum weights of NHL players in the Detroit_NHL table ,"(135, 248)"
The average weight of the NHL players in the Detroit_NHL table ,174.11818181818182
The minimum and maximum weights of MLB players in the MLB_Players table,"(145, 282)"
The average weight of the MLB players in the MLB_Players table,209.35
NHL teams and players per team in 'final.db',{'New York Rangers': 16, 'Philadelphia Flyers': 2,
'Pittsburgh Penguins': 3, 'Boston Bruins': 17, 'Montréal Canadiens': 15, 'Toronto Maple Leafs': 20,
'Washington Capitals': 3, 'Chicago Blackhawks': 13, 'Detroit Red Wings': 15, 'St. Louis Blues': 1,
'Edmonton Oilers': 1, 'Vancouver Canucks': 2, 'Los Angeles Kings': 2}
```

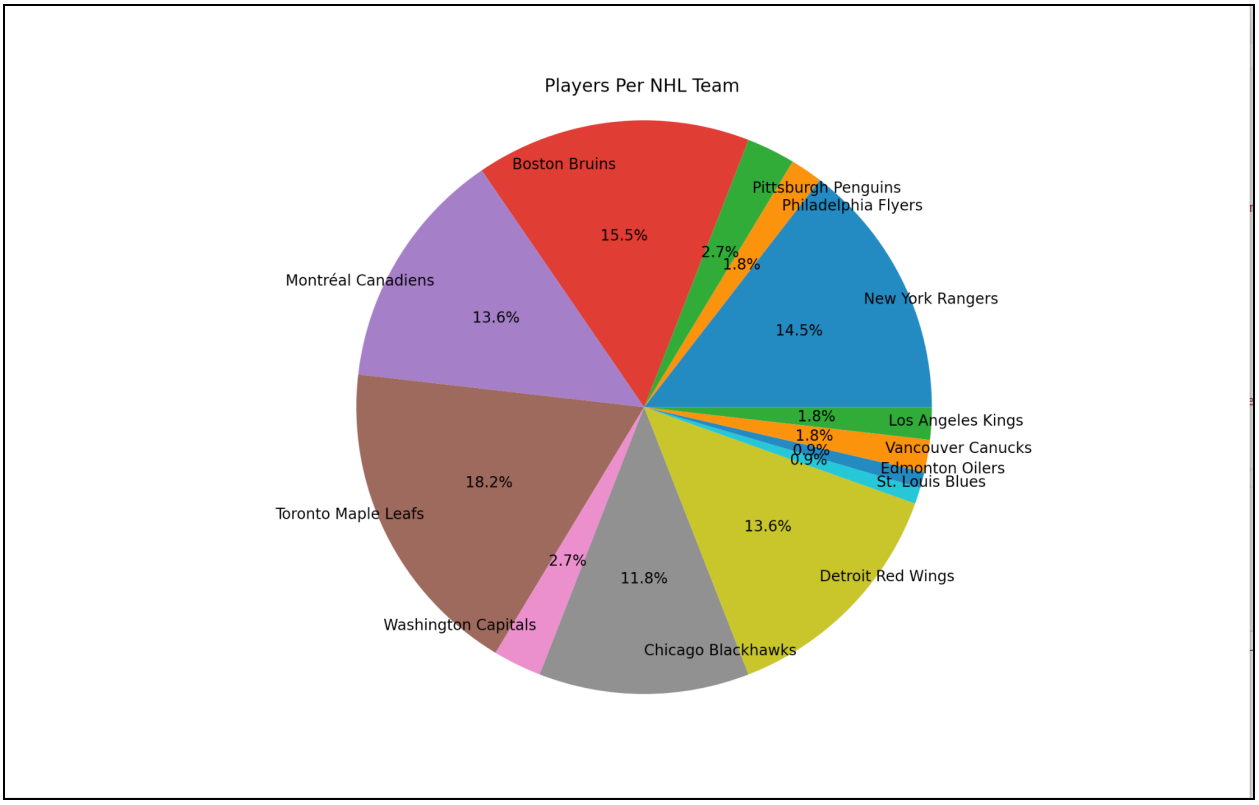
5. The visualizations that we created:




Weights of NBA Players
Count of Players
Weight
0
10
20
30
40
50
60
70
180
200
220
240
260
280


Weights of MLB Players
Count of Players
Weight
0
5
10
15
20
25
160
180
200
220
240
260
280


Weights of NHL Players
Count of Players
Weight
0
5
10
15
20
25
160
180
200
220
240
260
280


Players Per NHL Team
Boston Bruins
15.5%
Montréal Canadiens
13.6%
Toronto Maple Leafs
18.2%
Washington Capitals
2.7%
Chicago Blackhawks
11.8%
Detroit Red Wings
13.6%
St. Louis Blues
0.9%
Edmonton Oilers
0.9%
Vancouver Canucks
1.8%
Los Angeles Kings
1.8%
New York Rangers
14.5%
Philadelphia Flyers
1.8%
Pittsburgh Penguins
2.7%

6. Instructions for running our code:
    a. First, you have to run nba2.py four times and each time you run this it will add 25 new items into the database 'final.db'. In this database, there will be a table called "Detroit_NBA" and it will show you 100 NBA players with an id, their full name, and corresponding weight. Running this code will also append a csv file called 'calculations.txt' by adding the minimum NBA weight, maximum NBA weight, and average NBA weight.
    b. Next, you have to run mlb.py five times and each time you run this it will add 24 new items into the database 'final.db'. 8 MLB players from the Detroit Tigers, New York Yankees, and Los Angeles Dodgers will be added at a time to a table called "MLB_Players". This table includes an id, full name, and corresponding weight. Running this code will also append a csv file called 'calculations.txt' by adding the minimum MLB weight, maximum MLB weight, and average MLB weight.
    c. Third, you have to run nhl.py five times and each time you run this it will add 25 new items into the database 'final.db'. In this database, there will be 2 tables, one called TeamNames and the other called Detroit_NHL. After running the code the first two times TeamNames will be filled with 32 teams, their id, and the data_id. After five times of running the code, Detorit_NHL will be filled with 110 players and their associated data: id, weight, and team_id. Running will also append a csv file called 'calculations.txt' by adding the minimum NHL weight, maximum NHL weight, average NHL weight, and a dictionary with the team names as keys and their associated value of how many players in the table are on that team.
7. Documentation for each function that you wrote. This includes the input and output for each function (20 points)
    a. setUpDatabase(db_name)
        i. This is in nba2.py, nhl.py, and mlb.py
        ii. Takes in the name of the database as a parameter ('final.db') and sets up the database in
        iii. db sqlite3
    b. createNbaTable(cur, conn)
        i. This function is in nba2.py
        ii. It creates the table Detroit_NBA in the 'final.db' database
    c. getPlayerData(lst)
        i. This function is in nba2.py
        ii. It takes in a list as a parameter (player_weight) and it retrieves data from the https://www.balldontlie.io/api/v1/players/ API and outputs a list of tuples of the player names and their integer weights
    d. addPlayerWeightsToTable(cur, conn, lst)
        i. This function is in nba2.py

ii. It takes in a list as a parameter (player_weight) and inserts this data into the ‘final.db’ database to Detroit_NBA table 25 items at a time.

e. sorting_weights(lst, filename)
   i. This function is in nba2.py
   ii. It takes in a list as a parameter (weight_lst) and it finds the minimum weight and the maximum weight of the NBA players retrieved from the API and returns these in a tuple. It then appends the csv file ‘calculations.txt’ and adds the tuple of min and max weights.

f. avg_weight(cur, conn, filename)
   i. This function is in nba2.py
   ii. It selects the data from the weight column in the Detroit_NBA table in ‘final.db’ and returns the average of the weights . It then appends the csv file ‘calculation.txt’ and adds the average weight.

g. main()
   i. This specific main function is in nba2.py
   ii. It calls all of the functions in this file
   iii.

```
def main():
    cur, conn = setUpDatabase('final.db')
    createNbaTable(cur, conn)
    getPlayerData(player_weight)
    addPlayerWeightsToTable(cur, conn, player_weight)
    print(sorting_weights(weight_lst, 'calculations.txt'))
    print(avg_weight(cur, conn, 'calculations.txt'))
main()
```

h. create_nhl_table(cur, conn)
   i. This function is in nhl.py
   ii. It creates the table Detroit_NHL in the ‘final.db’ database

i. create_team_table(cur, conn)
   i. This function is in nhl.py
   ii. It creates the table TeamNames in ‘final.db’ that shares a key with Detroit_NHL on TeamNames.id and Detorit_NHL.team_id

j. team_data(cur, conn)
   i. This function is in nhl.py
   ii. It gets the team data from this link 'https://statsapi.web.nhl.com/api/v1/teams' of the nhl api and adds the team id and team name to a list of tuples called teams
   iii. Next, that data in the teams list is added to the TeamNames table, 25 items at a time.
   iv. Finally, teams tuples list is returned

k. get_player_data(cur, conn)
   i. This function is in nhl.py
   ii. It gets the player data from the link: 'https://records.nhl.com/site/api/player' of the nhl api

iii. It parses through the first 275 players and adds their full name, weight, and corresponding team_id selected from the TeamNames table and adds that information in a tuple to a list called new_player_weight
iv. New_player_weight is returned

l. addPlayerWeightsToTable(cur, conn, tup_lst)
    i. This function is in nhl.py
    ii. It uses the new_player_weight list returned from the function get_player_data and adds that information for 110 players to the Detroit_NHL table, 25 items at a time
m. sorted_weights(players_data)
    i. This function is in nhl.py
    ii. It takes in a list, players_data, that is returned by the function get_player_data
    iii. It appends the weights of every players form the Detroit_NHL table to a new list called weight_lst, that is then sorted
    iv. The max and min weights for the nhl players is then known from the first and last numbers in weight_lst
    v. That min_max tuple is written out to the csv file 'calculations.txt'
    vi. A tuple is returned of the min and max weights
n. avg_weight(cur, conn)
    i. This function is in nhl.py
    ii. It finds the average weight from the weights column of the Detroit_NHL table and returns that value
    iii. It also writes out that avg weight to a csv file 'calculations.txt'
o. players_per_team(dict, num, cur, conn)
    i. This function is in nhl.py
    ii. It takes in an empty dict that is created outside of the function
    iii. It also takes in a num, that stands for an id number in the TeamNames table
    iv. It joins the tables Detroit_NHL and TeamNames on Detroit_NHL.team_id and TeamNames.id in order to return a list of the same team name for the amount of times a player from Detroit_NHL is on that team
    v. The number of times that team name is selected is the value for the team name key added to the empty dictionary, dict
    vi. None is returned
p. main()
    i. This function is in nhl.py

ii. It calls all of the functions in this file

```
def main():
    cur,conn = setUpDatabase('final.db')
    create_nhl_table(cur, conn)
    create_team_table(cur,conn)
    teams = team_data(cur,conn)
    players_data = get_player_data(cur, conn)
    addPlayerWeightsToTable(cur, conn, players_data)
    sorted_weights(players_data, weight_lst, 'calculations.txt')
    print(avg_weight(cur, conn, 'calculations.txt'))
    for num in range(len(teams)):
        players_per_team(team_count, num+1, cur, conn)
    print(team_count)

main()
```

q. createMlbTable(cur, conn)
   i. This function is in mlb.py
   ii. It creates the table Detroit_MLB in the 'final.db' database
r. get_tigers_players(lst)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (tigers_player_weight) and it retrieves data from the http://lookup-service-prod.mlb.com/json/named.roster_40.bam?team_id='116' API and outputs a list of tuples of the player names and their integer weights
s. get_yankees_player(lst)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (yankees_player_weight) and it retrieves data from the http://lookup-service-prod.mlb.com/json/named.roster_40.bam?team_id='147' API and outputs a list of tuples of the player names and their integer weights
t. get_dodgers_player(lst)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (dodgers_player_weight) and it retrieves data from the http://lookup-service-prod.mlb.com/json/named.roster_40.bam?team_id='119' API and outputs a list of tuples of the player names and their integer weights
u. add_to_table(cur, conn, lst)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (tigers_player_weight) and inserts this data into the 'final.db' database to Detroit_MLB table 8 items at a time.
v. add_to_table2(cur, conn, lst)
   i. This function is in mlb.py

ii. It takes in a list as a parameter (yankees_player_weight) and inserts this data into the 'final.db' database to Detroit_MLB table 8 items at a time.

w. add_to_table3(cur, conn, lst)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (dodgers_player_weight) and inserts this data into the 'final.db' database to Detroit_MLB table 8 items at a time.

x. sorting_weights(lst, filename)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (weight_lst) and it finds the minimum weight and the maximum weight of the MLB players retrieved from the API and returns these in a tuple. It then appends the csv file 'calculations.txt' and adds the tuple of min and max weights.

y. avg_weight(cur, conn, filename)
   i. This function is in mlb.py
   ii. It takes in a list as a parameter (weight_lst) and it finds the average weight of the MLB players from all three teams retrieved from the API and returns this average. It then appends the csv file 'calculations.txt' and adds the average weight.

z. main()
   i. This specific main function is in mlb.py
   ii. It calls all of the functions in this file
   iii.

```
def main():
    cur, conn = setUpDatabase('final.db')
    createMlbTable(cur, conn)
    get_tigers_players(tigers_player_weight)
    get_yankees_player(yankees_player_weight)
    get_dodgers_player(dodgers_player_weight)
    add_to_table(cur,conn,tigers_player_weight)
    add_to_table2(cur,conn,yankees_player_weight)
    add_to_table3(cur,conn,dodgers_player_weight)
    sorting_weights(weight_lst, 'calculations.txt')
    avg_weight(cur, conn, 'calculations.txt')
    print(weight_lst)
main()
```

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)
   a. Included in issues table above