SI 206- Final Project

Nadav Aaronson

Ondrej Styler

APIs, SQL, and Visualization

https://github.com/nadavaa/final_project_sports_data

Project Goals

Our goal for this project was to gather information about sports we are interested in. In the NBA we wanted to compare the difference between the number of points a team scores at home vs away, and the number of points they receive at home vs away. In the NHL, we wanted to look at the height of players who were drafted and look at the differences by nationality. Moreover, we were both excited in coding a real project completely from scratch and honing our API and SQL skills.

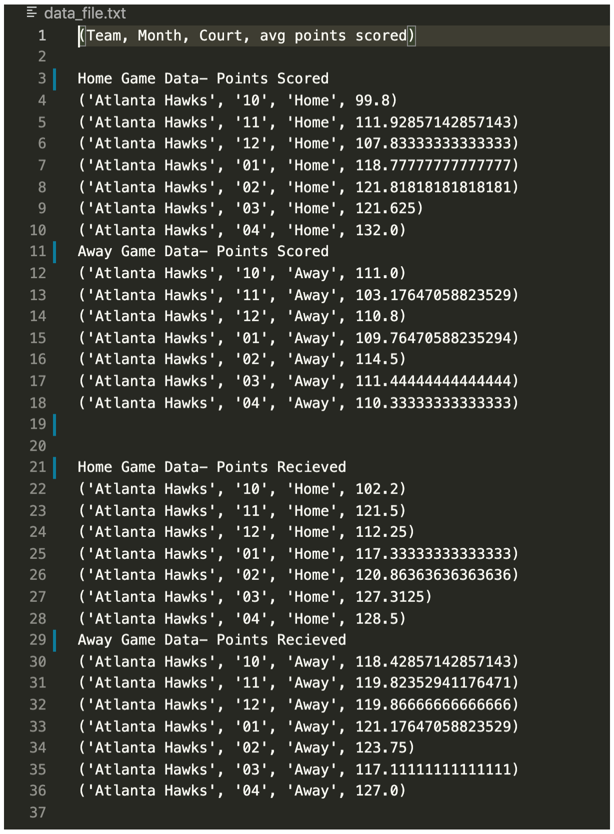
Goals Achieved

As a result of some research prior to starting the project, we were able to achieve most of the goals we set for ourselves. The API “balldontlie” was a great one to use for gathering NBA data. The instructions on how the data is organized and how to fetch it were clear, and the information itself was very useful to get the data we wanted. The API from GitLab about NHL records was easy to understand. The documentation was clear to follow and the data gathered were exactly what we were looking for.

Problems

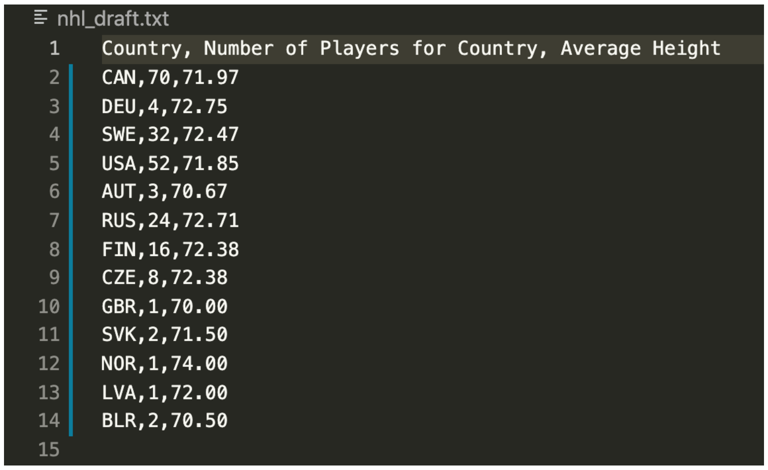
We faced a little challenge at the beginning with switching one of the API from tennis topic to NHL due to insufficient information. Ultimately, everything in the process of this project was relatively smooth. However, we were struggling with writing the proper SQL queries to gather the data. At first, we weren’t able to fetch any data from the database, and it would return empty lists. The way we solved this problem was by writing the queries in the DB Browser, where it was easier to understand where our mistakes were. Another struggle that we faced was creating the visualizations. Although we learned that in class, we were struggling with creating two visualizations in the same figure. We were able to show only one graph at a time, or all the graphs information but in one graph. We solved this problem by referring back to the slides and learning online, especially W3Schools. Once we figured out the SQL queries and the graphs we took a second to think about when it made sense to display the visualizations which was more of a logistic problem.

Calculations



data_file.txt

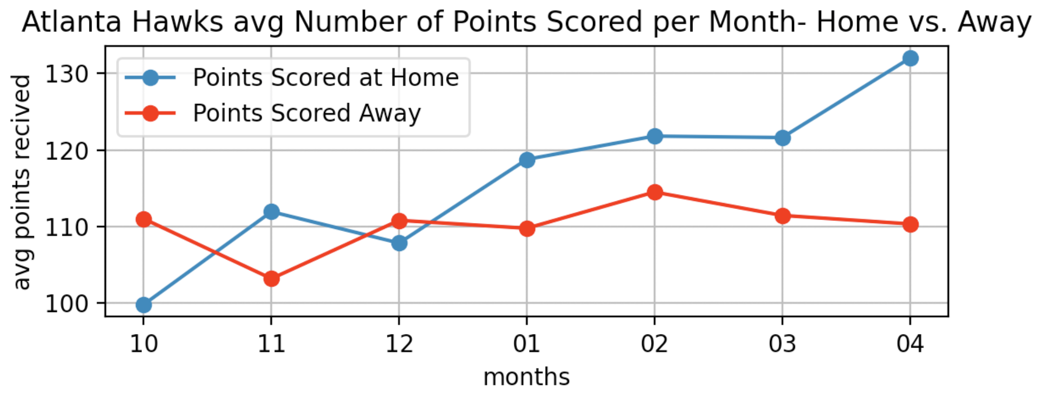
```
(Team, Month, Court, avg points scored)

Home Game Data- Points Scored
('Atlanta Hawks', '10', 'Home', 99.8)
('Atlanta Hawks', '11', 'Home', 111.92857142857143)
('Atlanta Hawks', '12', 'Home', 107.83333333333333)
('Atlanta Hawks', '01', 'Home', 118.77777777777777)
('Atlanta Hawks', '02', 'Home', 121.81818181818181)
('Atlanta Hawks', '03', 'Home', 121.625)
('Atlanta Hawks', '04', 'Home', 132.0)
Away Game Data- Points Scored
('Atlanta Hawks', '10', 'Away', 111.0)
('Atlanta Hawks', '11', 'Away', 103.17647058823529)
('Atlanta Hawks', '12', 'Away', 110.8)
('Atlanta Hawks', '01', 'Away', 109.76470588235294)
('Atlanta Hawks', '02', 'Away', 114.5)
('Atlanta Hawks', '03', 'Away', 111.44444444444444)
('Atlanta Hawks', '04', 'Away', 110.33333333333333)


Home Game Data- Points Recieved
('Atlanta Hawks', '10', 'Home', 102.2)
('Atlanta Hawks', '11', 'Home', 121.5)
('Atlanta Hawks', '12', 'Home', 112.25)
('Atlanta Hawks', '01', 'Home', 117.33333333333333)
('Atlanta Hawks', '02', 'Home', 120.86363636363636)
('Atlanta Hawks', '03', 'Home', 127.3125)
('Atlanta Hawks', '04', 'Home', 128.5)
Away Game Data- Points Recieved
('Atlanta Hawks', '10', 'Away', 118.42857142857143)
('Atlanta Hawks', '11', 'Away', 119.82352941176471)
('Atlanta Hawks', '12', 'Away', 119.86666666666666)
('Atlanta Hawks', '01', 'Away', 121.17647058823529)
('Atlanta Hawks', '02', 'Away', 123.75)
('Atlanta Hawks', '03', 'Away', 117.11111111111111)
('Atlanta Hawks', '04', 'Away', 127.0)
```

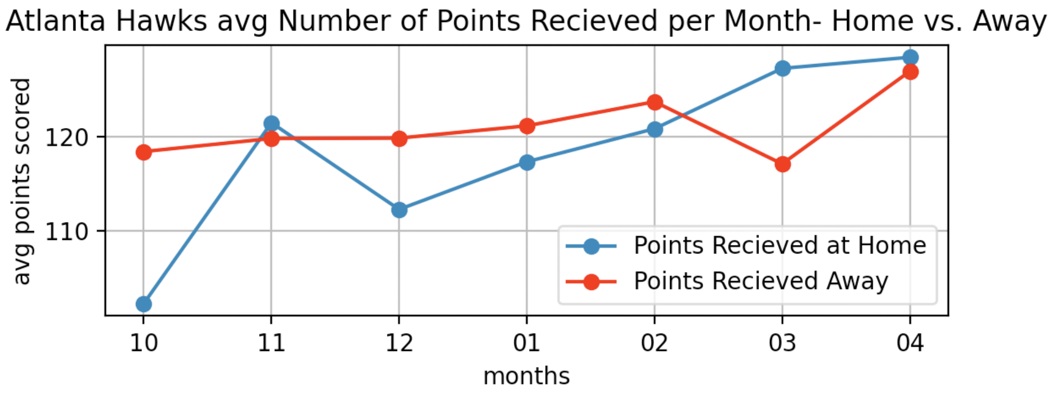


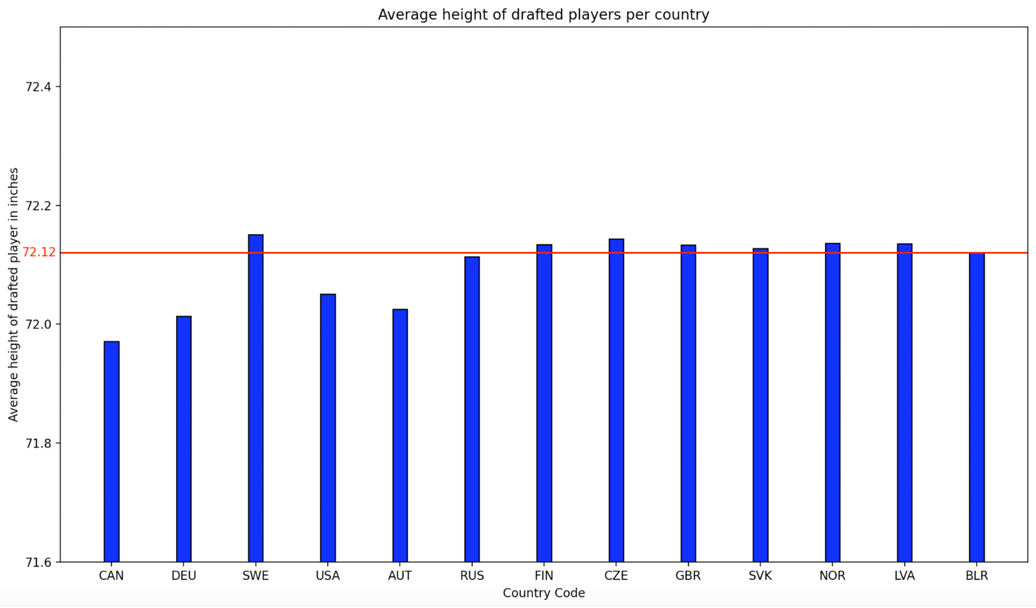
nhl_draft.txt

```
Country, Number of Players for Country, Average Height
CAN,70,71.97
DEU,4,72.75
SWE,32,72.47
USA,52,71.85
AUT,3,70.67
RUS,24,72.71
FIN,16,72.38
CZE,8,72.38
GBR,1,70.00
SVK,2,71.50
NOR,1,74.00
LVA,1,72.00
BLR,2,70.50
```

Visualizations





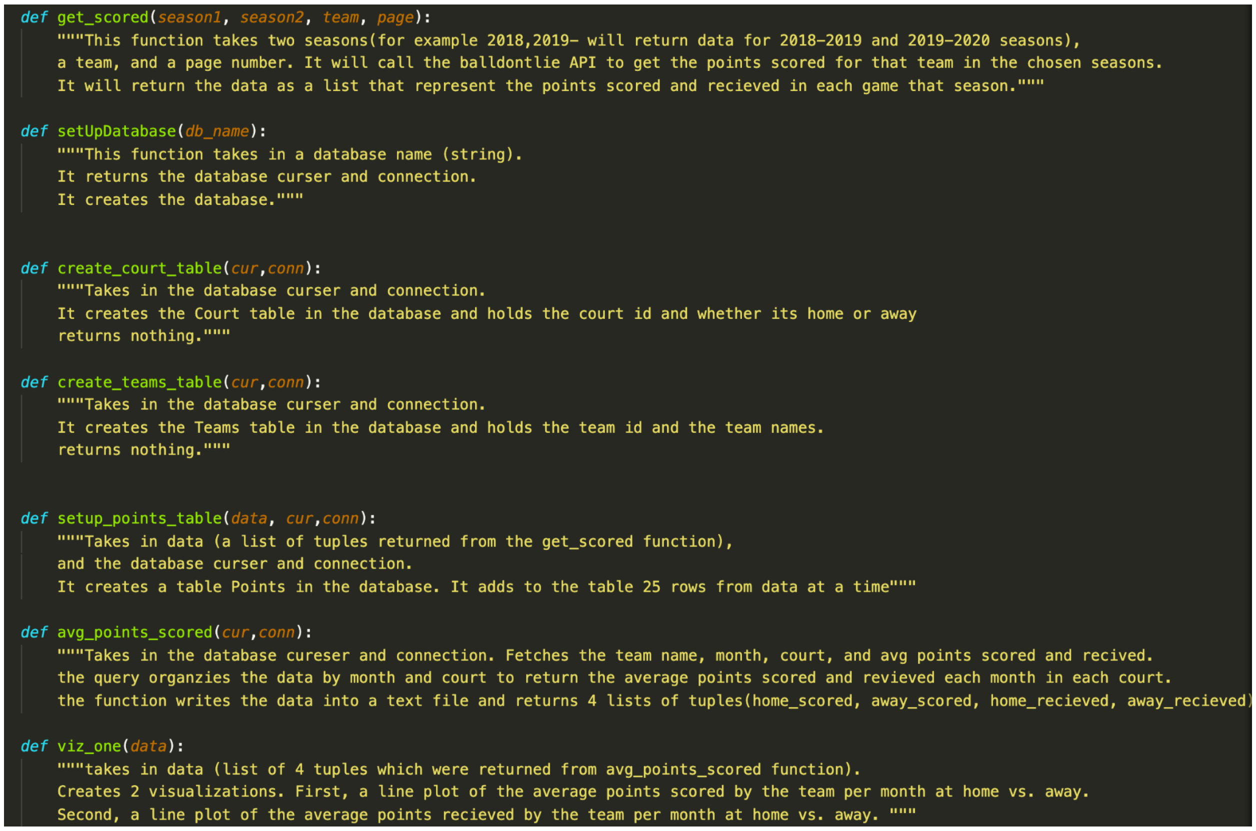


Instructions for running code

- Before running any py files, make sure that sports_data.db does not already exist in your files. If it does, delete it so it can be recreated.
- *Nba_project_cody.py-*
    - Scroll down to the bottom of the code to the main function
    - Choose what 2 seasons you are interested in viewing (note: the year chosen will show the data from the season that started on that year and went on until the next year. ex. choosing 2014, 2015- would provide data from the 2014-2015, and 2015-2016 seasons)
    - Choose what team index you are interested in viewing- at the beginning of the main function attached a table indicating the teams and their indexes.
    - Run code until visualization appears (visualization will appear once no more data is added to the database).
- *Nhl.py-*
    - Scroll through the file to read each function's descriptions if curious.
    - Run the code 9 times to gather all the data from the NHL draft in 2020 and to see the visualization that shows the average height of a player drafted from each country.
    - Every time you run the code, it downloads 25 items from the API - all the way up to 216.

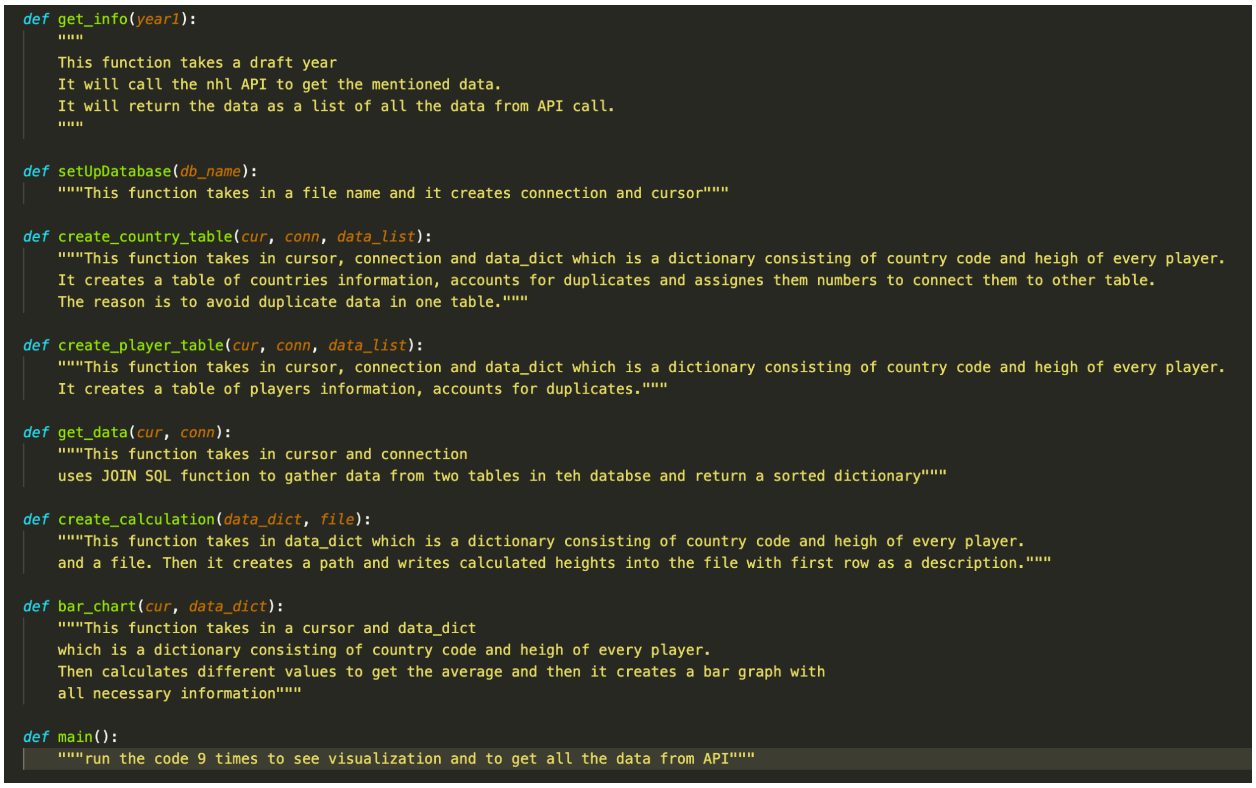
Documentation

nba_project_code.py



```python
def get_scored(season1, season2, team, page):
    """This function takes two seasons(for example 2018,2019- will return data for 2018-2019 and 2019-2020 seasons),
    a team, and a page number. It will call the balldontlie API to get the points scored for that team in the chosen seasons.
    It will return the data as a list that represent the points scored and recieved in each game that season."""

def setUpDatabase(db_name):
    """This function takes in a database name (string).
    It returns the database curser and connection.
    It creates the database."""


def create_court_table(cur,conn):
    """Takes in the database curser and connection.
    It creates the Court table in the database and holds the court id and whether its home or away
    returns nothing."""

def create_teams_table(cur,conn):
    """Takes in the database curser and connection.
    It creates the Teams table in the database and holds the team id and the team names.
    returns nothing."""


def setup_points_table(data, cur,conn):
    """Takes in data (a list of tuples returned from the get_scored function),
    and the database curser and connection.
    It creates a table Points in the database. It adds to the table 25 rows from data at a time"""

def avg_points_scored(cur,conn):
    """Takes in the database cureser and connection. Fetches the team name, month, court, and avg points scored and recived.
    the query organzies the data by month and court to return the average points scored and revieved each month in each court.
    the function writes the data into a text file and returns 4 lists of tuples(home_scored, away_scored, home_recieved, away_recieved)

def viz_one(data):
    """takes in data (list of 4 tuples which were returned from avg_points_scored function).
    Creates 2 visualizations. First, a line plot of the average points scored by the team per month at home vs. away.
    Second, a line plot of the average points recieved by the team per month at home vs. away. """
```

nhl.py



```python
def get_info(year1):
    """
    This function takes a draft year
    It will call the nhl API to get the mentioned data.
    It will return the data as a list of all the data from API call.
    """

def setUpDatabase(db_name):
    """This function takes in a file name and it creates connection and cursor"""

def create_country_table(cur, conn, data_list):
    """This function takes in cursor, connection and data_dict which is a dictionary consisting of country code and heigh of every player.
    It creates a table of countries information, accounts for duplicates and assignes them numbers to connect them to other table.
    The reason is to avoid duplicate data in one table."""

def create_player_table(cur, conn, data_list):
    """This function takes in cursor, connection and data_dict which is a dictionary consisting of country code and heigh of every player.
    It creates a table of players information, accounts for duplicates."""

def get_data(cur, conn):
    """This function takes in cursor and connection
    uses JOIN SQL function to gather data from two tables in teh databse and return a sorted dictionary"""

def create_calculation(data_dict, file):
    """This function takes in data_dict which is a dictionary consisting of country code and heigh of every player.
    and a file. Then it creates a path and writes calculated heights into the file with first row as a description."""

def bar_chart(cur, data_dict):
    """This function takes in a cursor and data_dict
    which is a dictionary consisting of country code and heigh of every player.
    Then calculates different values to get the average and then it creates a bar graph with
    all necessary information"""

def main():
    """run the code 9 times to see visualization and to get all the data from API"""
```

Resources

Date	**Issue description**	**Location of resource**	**result**
12/1	Needed to extract data from a long json file	https://jsonformatter.curiousconcept.com/	Was able to see the data in an organized way
12/1	Creating a database	Used the code we wrote for the discussion 11 assignment	Well written function to create a database
12/4	Pulling 25 rows at a time from the API	https://piazza.com/class/ksgl012i3yw3lz?cid=331	Was able to get 25 rows at a time
12/5	Extracting the month out of a date in the database	https://www.w3schools.com/sql/func_sqlserver_month.asp	Extracted the month by using this line: `strftime('%m', Points.date)`
12/7	Creating 2 visualizations in 1 figure	https://matplotlib.org/devdocs/gallery/subplots_axes_and_figures/subplots_demo.html	`plt.subplot(2, 1, 1)`
12/3	Creating tables and make sure there are no duplicates	https://towardsdatascience.com/how-to-deal-with-duplicate-entries-using-sql-4b017465d6dc	Interesting article that put me on the right path
12/5	How to read only one row and all rows	https://pynative.com/python-cursor-fetchall-fetchmany-fetchone-to-read-rows-from-table/	Explained how to work with fetchone, fetchall
12/6	How to write into file	https://docs.python.org/3/library/csv.html	Reminded me how csv, reader,writer work and how I can utilize it in my code
12/9	Creating average line in graph visualization	https://pretagteam.com/question/average-line-for-bar-chart-in-matplotlib	I was able to figure out how to do it.