Ericson's #1 Fan Club Report

Matthew Palazzolo
Affaan Waheed
Sharath Jegannathan

Initial Project Goals

Our initial goal for this project was to analyze the potential correlation between the *heights of sports team players and team performance* for 5 different sports. We planned to use the following APIs:

- For Baseball: *MLB Records and Stats*
  https://appac.github.io/mlb-data-api-docs/
- For Basketball: *NBA Stats*
  https://any-api.com/nba_com/nba_com/docs
- For Football: *College Football Data*
  https://api.collegefootballdata.com/api/docs/?url=/api-docs.json
- For Soccer: *Football-Data*
  https://www.football-data.org/documentation/quickstart
- For Hockey: *NHL Records and Stats*
  https://gitlab.com/dword4/nhlapi

Final Project Goals

By the end of the project, there were many changes to the initial goals. We decided to analyze *the difference in win percentages at home versus away* for 4 different sports. Additionally, we decided to use these 3 APIs due to better documentation:

- For Baseball: *MLB Stats*
  https://github.com/toddrob99/MLB-StatsAPI/wiki
- For Basketball: *Ball Don't Lie*
  https://www.balldontlie.io/home.html#introduction
- For Soccer: *OpenLigaDB*
  https://api.openligadb.de/index.html

and BeautifulSoup on this website:

- For Football: *TheSportsDB*
  https://www.thesportsdb.com/season/4391-NFL/2022&all=1

Data Collection

For each sport (baseball, basketball, football, soccer), we populated two tables - a *games table* to store information about games played in 2022, and a *names table* to map the id of teams to their name (and thus avoid duplicate string data). Pictured below is a portion of the games table (left) and names table (right) for baseball.

game_id	home_id	away_id	home_score	away_score
1	1	2	14	1
2	3	4	5	2
3	4	3	3	4
4	5	6	5	3
5	7	8	4	2
6	1	9	7	6
7	10	11	5	9
8	12	13	6	6
9	14	2	3	1
10	15	16	4	3

id	name
1	Boston Red Sox
2	Minnesota Twins
3	Chicago White Sox
4	Chicago Cubs
5	Colorado Rockies
6	Arizona Diamondbacks
7	St. Louis Cardinals
8	Houston Astros
9	Tampa Bay Rays
10	Baltimore Orioles

Limiting Data Storage

An important part of data collection is managing the growth of your database. To limit the max number of new rows added to the database each run to 25, we used the following algorithm:

```
limit = 25
old size = get_size(database)

loop:
    add new row to the database
    new size = get_size(database)
    if new_size - old_size >= limit:
        break
```

Calculations

For each sport, we calculated the top 5 teams with the highest home win percentage and away win percentage, and wrote the results to text files. Pictured below are the 4 text files.

baseball.txt

```
--- Top 5 Home Win Percentages ---
Atlanta Braves 100
Boston Red Sox 85
Baltimore Orioles 83
Philadelphia Phillies 83
Pittsburgh Pirates 80
--- Top 5 Away Win Percentages ---
Kansas City Royals 75
Chicago White Sox 66
San Francisco Giants 66
Colorado Rockies 60
St. Louis Cardinals 60
```

basketball.txt

```
--- Top 5 Home Win Percentages ---
Denver Nuggets 100
Miami Heat 100
Detroit Pistons 100
Utah Jazz 100
San Antonio Spurs 100
--- Top 5 Away Win Percentages ---
LA Clippers 100
Indiana Pacers 100
Milwaukee Bucks 100
Boston Celtics 66
Philadelphia 76ers 66
```

football.txt

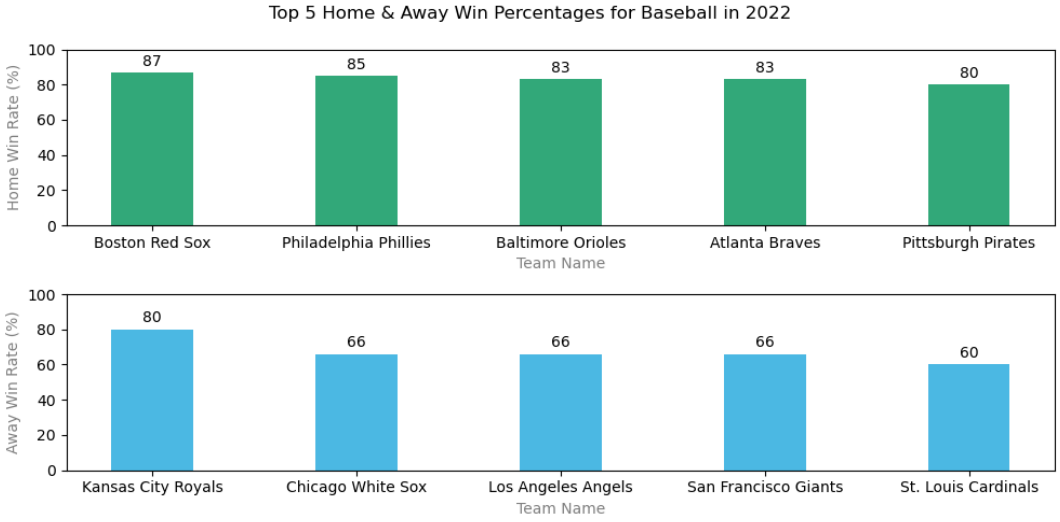
```
--- Top 5 Home Win Percentages ---
Buffalo Bills 100
Las Vegas Raiders 83
Dallas Cowboys 83
New York Giants 80
Tennessee Titans 80
--- Top 5 Away Win Percentages ---
New York Jets 100
Philadelphia Eagles 83
Baltimore Ravens 80
New York Giants 66
Miami Dolphins 66
```

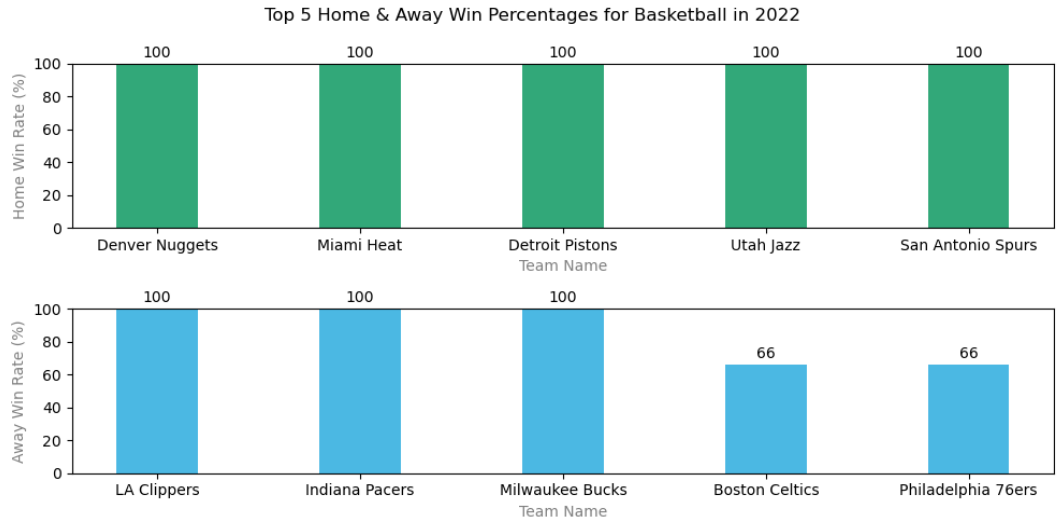
soccer.txt

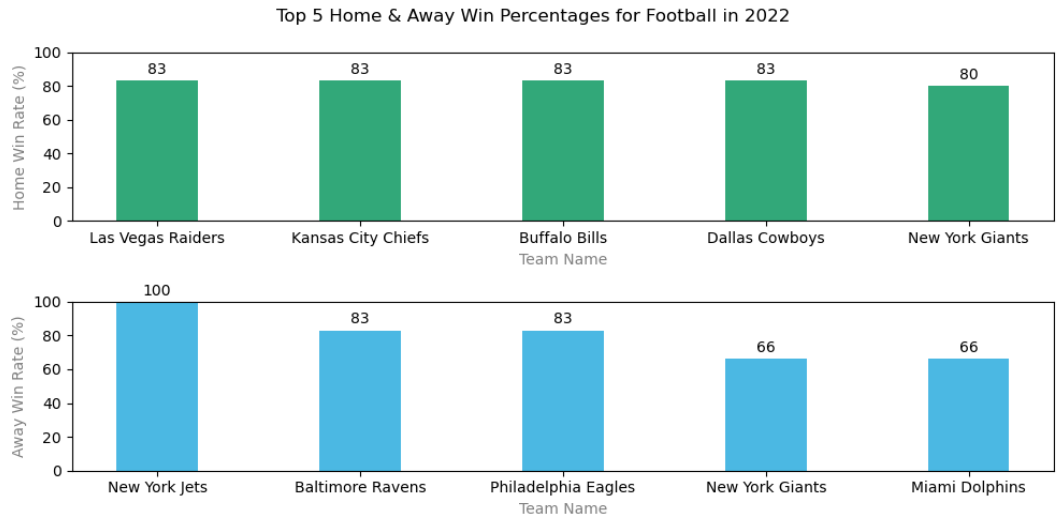
```
--- Top 5 Home Win Percentages ---
FC Bayern M□nchen 100
1. FC Union Berlin 100
SC Freiburg 87
Borussia Dortmund 87
RB Leipzig 87
--- Top 5 Away Win Percentages ---
FC Bayern M□nchen 85
Eintracht Frankfurt 66
1. FC Union Berlin 60
Werder Bremen 57
Borussia Dortmund 54
```

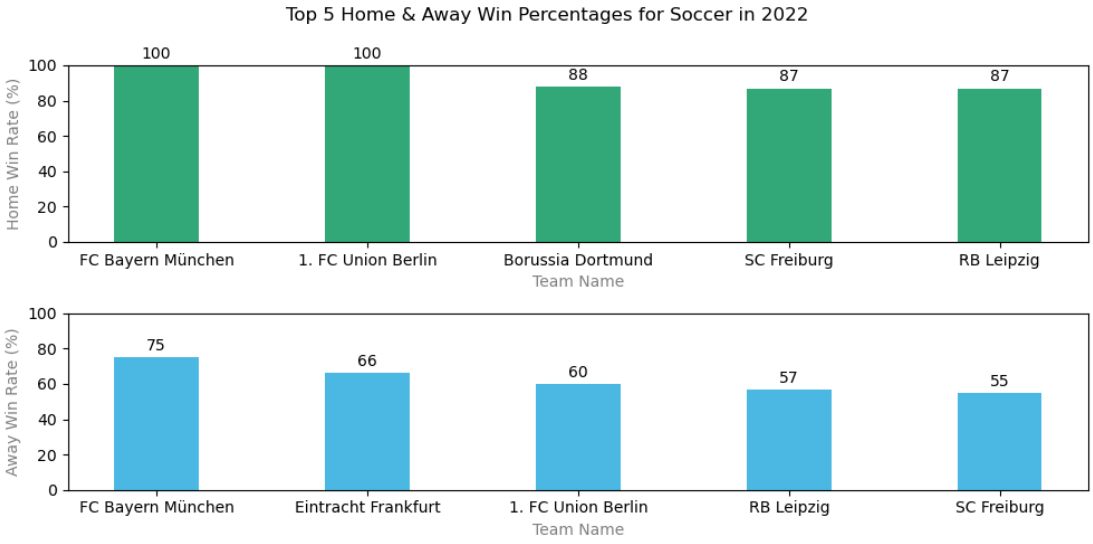
Visualizations

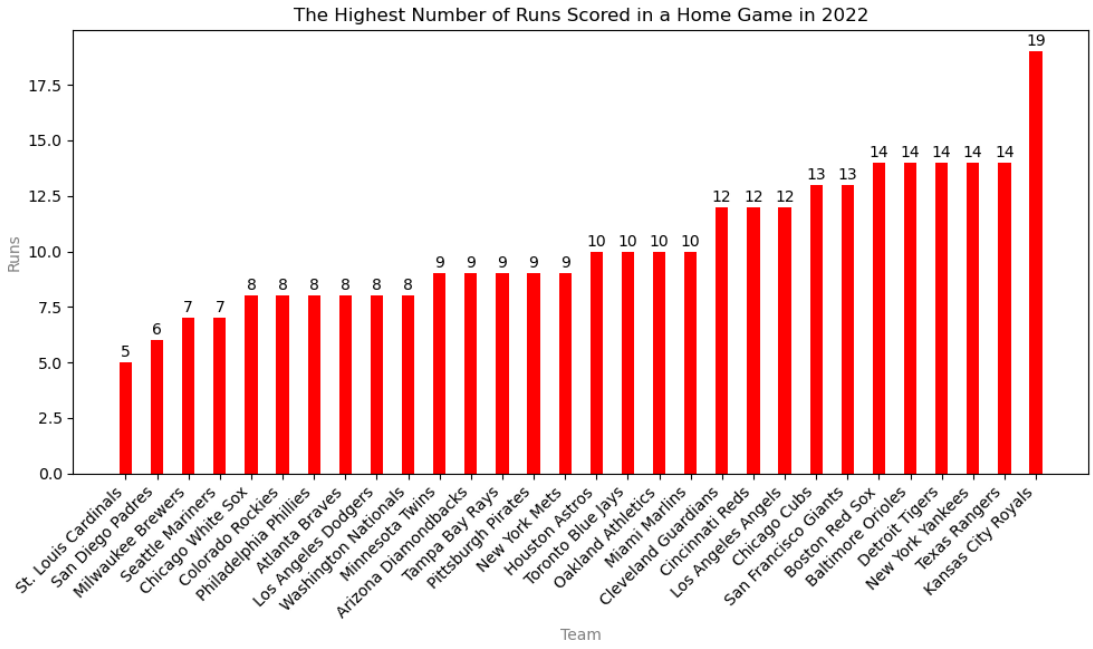
In total, we created 5 visualizations. First, from the above text files, we created a bar chart visualization for each sport using matplotlib. Then, using a JOIN statement, we created an extra bar chart for baseball, showcasing the maximum number of runs scored in a home game per team. Pictured below are the final visualizations.


Top 5 Home & Away Win Percentages for Baseball in 2022
Home Win Rate (%)
Boston Red Sox
Philadelphia Phillies
Baltimore Orioles
Atlanta Braves
Pittsburgh Pirates
Team Name
Away Win Rate (%)
Kansas City Royals
Chicago White Sox
Los Angeles Angels
San Francisco Giants
St. Louis Cardinals
Team Name


Top 5 Home & Away Win Percentages for Basketball in 2022
Home Win Rate (%)
Denver Nuggets
Miami Heat
Detroit Pistons
Utah Jazz
San Antonio Spurs
Team Name
Away Win Rate (%)
LA Clippers
Indiana Pacers
Milwaukee Bucks
Boston Celtics
Philadelphia 76ers
Team Name


Top 5 Home & Away Win Percentages for Football in 2022
Home Win Rate (%)
Las Vegas Raiders
Kansas City Chiefs
Buffalo Bills
Dallas Cowboys
New York Giants
Team Name
Away Win Rate (%)
New York Jets
Baltimore Ravens
Philadelphia Eagles
New York Giants
Miami Dolphins
Team Name


Top 5 Home & Away Win Percentages for Soccer in 2022
Home Win Rate (%)
100
100
88
87
87
FC Bayern München
1. FC Union Berlin
Borussia Dortmund
SC Freiburg
RB Leipzig
Team Name
Away Win Rate (%)
75
66
60
57
55
FC Bayern München
Eintracht Frankfurt
1. FC Union Berlin
RB Leipzig
SC Freiburg
Team Name


The Highest Number of Runs Scored in a Home Game in 2022
Runs
Team
St. Louis Cardinals
San Diego Padres
Milwaukee Brewers
Seattle Mariners
Chicago White Sox
Colorado Rockies
Philadelphia Phillies
Atlanta Braves
Los Angeles Dodgers
Washington Nationals
Minnesota Twins
Arizona Diamondbacks
Tampa Bay Rays
Pittsburgh Pirates
New York Mets
Houston Astros
Toronto Blue Jays
Oakland Athletics
Miami Marlins
Cleveland Guardians
Cincinnati Reds
Los Angeles Angels
Chicago Cubs
San Francisco Giants
Boston Red Sox
Baltimore Orioles
Detroit Tigers
New York Yankees
Texas Rangers
Kansas City Royals

Problems Faced

We hit many problems along the way. As discussed earlier, many of the originally planned APIs had very limited and/or incorrect documentation. This made gathering data difficult and frustrating. We also ran into a problem with BeautifulSoup - we learned that finding tags based on class name is not reliable due to their non deterministic nature. Finally, we frequently had issues with the visualizations. Specifically, the x and y labels kept getting cut off. We eventually learned that plt.tight_layout() solves this.

Instructions to Run the Code

- Clone the repo: https://github.com/mattbatw/SI206-Final-Project

- Ensure that you have the following files:
  - main.py
  - gather_and_store.py
  - process_and_visualize.py

- Run main.py at least 50 times. By default, the visualizations will pop up on every run, and you should notice them getting more accurate every time you run the code (because more data is added to the database). If you receive an error within the first few runs, they can be safely ignored.

  While not required, we strongly recommend ONLY RUNNING PART 2 IN MAIN.PY (and commenting out everything else) until the final run. That way the visualizations don't pop up every time. On the final run, uncomment everything, and you will see the final visualizations. Alternatively, you can just put Part 2 into a loop, and run main.py once (but this breaks the 25 at a time rule):

```
# Part 2 - Gather data, store to database
for _ in range(50):
    gather_and_store.populate_baseball_tables()
    gather_and_store.populate_basketball_tables()
    gather_and_store.populate_football_tables()
    gather_and_store.populate_soccer_tables()
```

Documentation

All functions in the repo include detailed docstrings and comments. However, we will include a summary of those docstrings here for your convenience. *No functions have a return value.*

main.py:

- main():
  - *Top level program function.*
    Executes Part 2 using gather_and_store, then Parts 3 and 4 using process_and_visualize. Also creates an extra visualization.

gather_and_store.py:

- store_to_db(table_name, headers, rows)
  - *Stores rows into the database to a specific table.*
- populate_baseball_tables()
  - *Calls Baseball API and extracts useful data.*
    Uses store_to_db() to store to the games table and names table.
- populate_basketball_tables()
  - *Calls Basketball API and extracts useful data.*
    Uses store_to_db() to store to the games table and names table.
- populate_football_tables()
  - *Creates Football BeautifulSoup and extracts useful data.*
    Uses store_to_db() to store to the games table and names table.
- populate_soccer_tables()
  - *Calls Soccer API and extracts useful data.*
    Uses store_to_db() to store to the games table and names table.

process_and_visualize.py:

- calculate_ratios(sport)
  - *Selects data for a specific sport from the database, calculates the home/away win percentages for each team, and writes the top 5 results to a text file.*
- visualize(sport)
  - *Reads the text file for a specific sport, and creates a bar graph visualization for the top 5 home/away win percentages.*
- extra_visualization()
  - *For baseball only, selects data from the database and creates a bar chart visualization for the maximum runs scored in a home game.*

Notes to the Instructors
To make the grading process as easy as possible, we like to demonstrate where in our project we have met each requirement. This will hopefully clear up any questions or confusion.

Access at least 3 APIs and 1 website - This is done in gather_and_store.py.

- populate_baseball_tables() calls a baseball API
- populate_basketball_tables() calls a basketball API
- populate_soccer_tables() calls a soccer API
- populate_football_tables() uses BeautifulSoup on a football website

Store at least 100 rows of data from each API/website - This is done in gather_and_store.py. As long as Part 2 is run many times, the games tables for each sport will reach at least 100 items (some more!)

At least 1 API/website must have 2 tables that share an integer key - This is done all over the database. Every games table has a corresponding names table which maps a name for home_id and away_id.

Limited the amount of data to a max of 25 items stored in the database each time a file is run to gather the data - This one is a little bit interesting to explain. In main.py, we call 4 populate functions, which each call store_to_db() two times. Notice that store_to_db() is limited to inserting 3 rows per call. Therefore, the max rows added to the database per run = 4 * 2 * 3 = 24.

Select items from the tables and calculate something from the data - This is done in process_and_visualize.py. In calculate_ratios(), we calculate the top 5 home/away win percentages for each team.

At least one database join used when selecting the data - This is done in process_and_visualize.py. In extra_visualization(), we use a JOIN.

Write a well-formatted, self explanatory file from the calculations - This is done in process_and_visualize.py. After a full run, calculate_ratios() will have written to baseball.txt, basketball.txt, football.txt, and soccer.txt.

Create visualizations from the data and/or calculations - This is done in process_and_visualize.py. We created a total of 5 visualizations. 4 are done by visualize(), and 1 is done by extra_visualization()

Resources Used

Date	Issue Description	Location of Resource	Result
11/28	Need to figure out how to count the number of rows in the databse in order to limit data stored per run	https://learnsql.com/cookbook/how-to-count-the-number-of-rows-in-a-table-in-sql/	Solved. We can use the statement "SELECT COUNT(*) FROM table_name"
11/30	Need to figure out how to sort a list based on a specific element of each tuple	https://chat.openai.com/	Solved. We can use a lambda key
11/30	Need to figure out how to combine 2 bar charts into one in matplotlib	https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html	Solved. We can use the plt.subplots module
12/6	Need to figure out how to select a row with a maximum value	https://stackoverflow.com/questions/7745609/sql-select-only-rows-with-max-value-on-a-column	Solved. We can use the MAX keyword