206 Final Project Report

By: Niko Bogoevich & Stephen Liu

Date: 12/16/24

Git:
https://github.com/sliliu/si206-final-project-seniors.git

Goals Planned:

The goal of our project was to see if the weather during the game had an effect on how the University of Michigan football team was affected by the weather as well as seeing if there was correlation between passing completion percentage and total points they scored. We planned on using the college football API, https://collegefootballdata.com/, to get total rush attempts, total passing attempts, total points scored, passing completions over attempts, rushing yards, and passing yards for each game from the 2024 to 2015. We also planned on using the weather API, https://weatherstack.com/?utm_source=Github& in order to the temperature, precipitation, snow, wind speed, and wind gust from the location and the days the University of Michigan team played their games.

Goals Achieved:

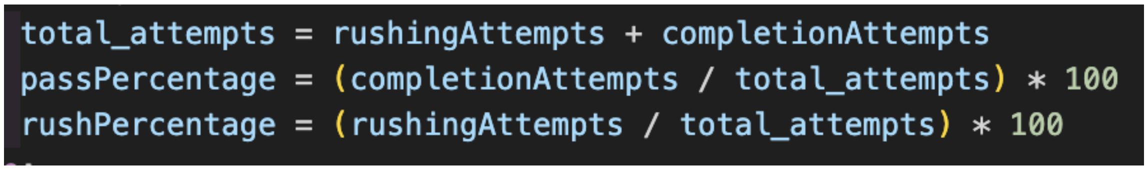
After finishing the project, we looked back at what we were able to accomplish and we gathered the said data from the college football API. We had to change our plan for the weather data. We figured out that if we wanted historical weather data from the API we were planning on using we would have to pay for it. We searched online and found a free weather API that had historical data, https://archive-api.open-meteo.com. We still gathered the same data that we were planning on gathering, but just from this new API. We were successfully able to calculate the average total points based on four different categories of passing completion percentages, 50% and below, 51% - 60%, 61% - 70%, and 71% and above. We were able to calculate the average total points based on three categories of temperature, 32 and below, 33 - 50 degrees, and 51 degrees and above. We also calculated the rushing percentage and passing percentage and compared the two percentages based on wind speed. Lastly, we calculated the passing completion percentage based on the wind speed.

Problems:

Some of the problems we faced were trying to find a new weather API we could use and we fixed that by finding another API on Google. We also ran into a problem where we were not gathering all the data from the weather API that we needed since we had 117 rows of data for football and we had 106 rows for the weather API. We quickly found that we were not looping the weather API enough times so we incremented the API call loop.

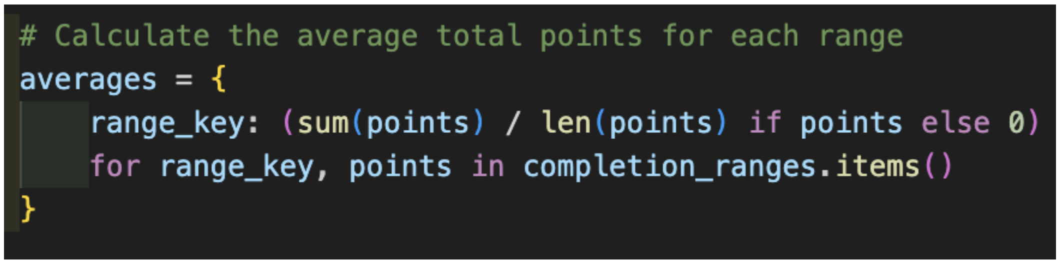
Calculation:

- Calculation for percentage of rushes and passes per game



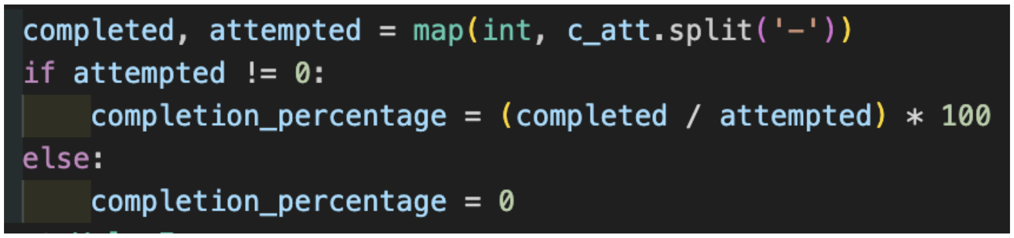
```
total_attempts = rushingAttempts + completionAttempts
passPercentage = (completionAttempts / total_attempts) * 100
rushPercentage = (rushingAttempts / total_attempts) * 100
```

- Calculation for Average total points based on temperature

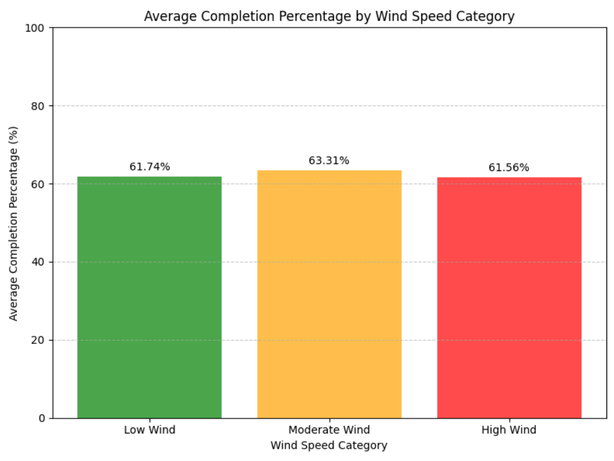


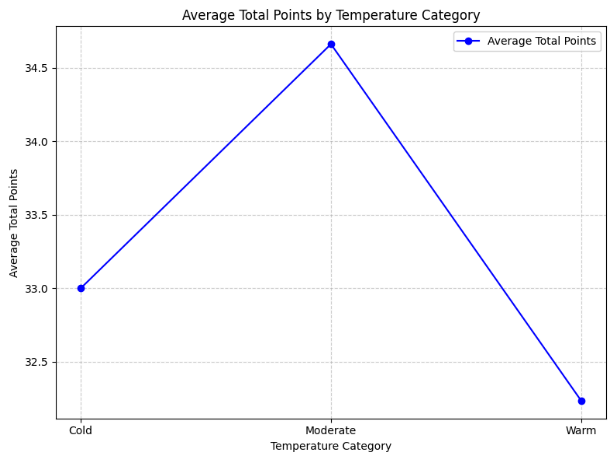
```
# Calculate the average total points for each range
averages = {
    range_key: (sum(points) / len(points) if points else 0)
    for range_key, points in completion_ranges.items()
}
```

- Calculation for pass completion percentage

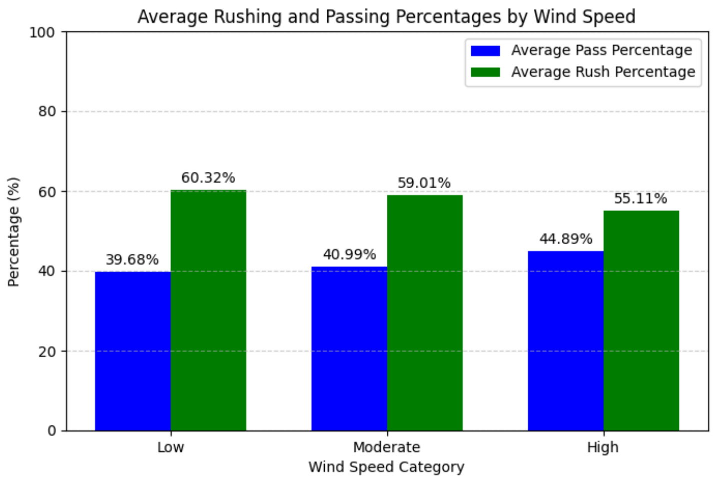


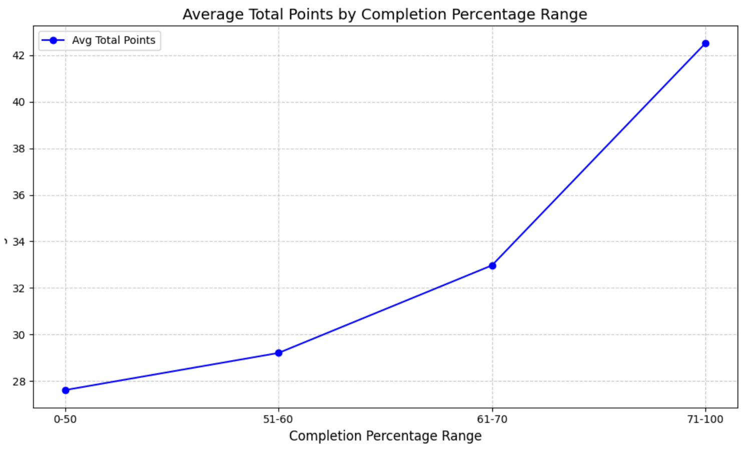
```
completed, attempted = map(int, c_att.split('-'))
if attempted != 0:
    completion_percentage = (completed / attempted) * 100
else:
    completion_percentage = 0
```





Visuals:



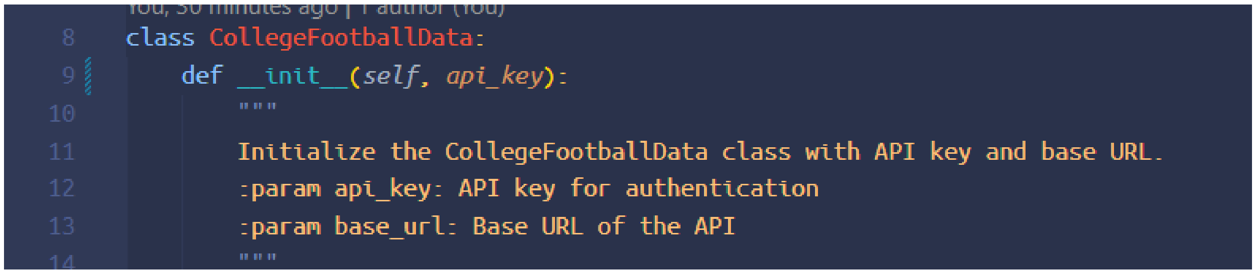


Running Code Instructions:

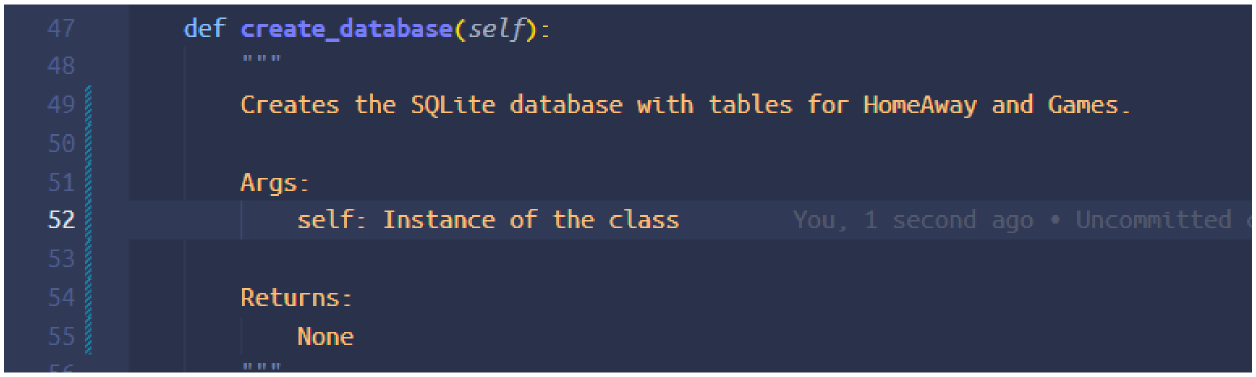
To run the code all you have to do is click the play button that runs that code and it will create the database file and the data text file, if in VSCode. If you are using the terminal you use -python3 final.py. You only run the code five times to get all 17 rows in the database. Each time you run it takes about one minute to run. After you run the api.py five times you need to run the calculations.py file once to get all the visuals.

Documentation for functions

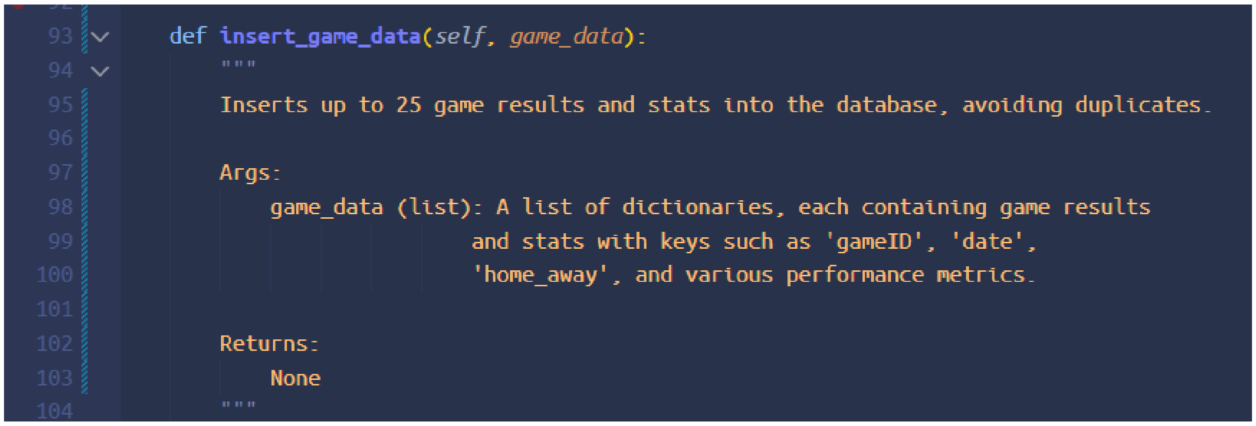
apis.py



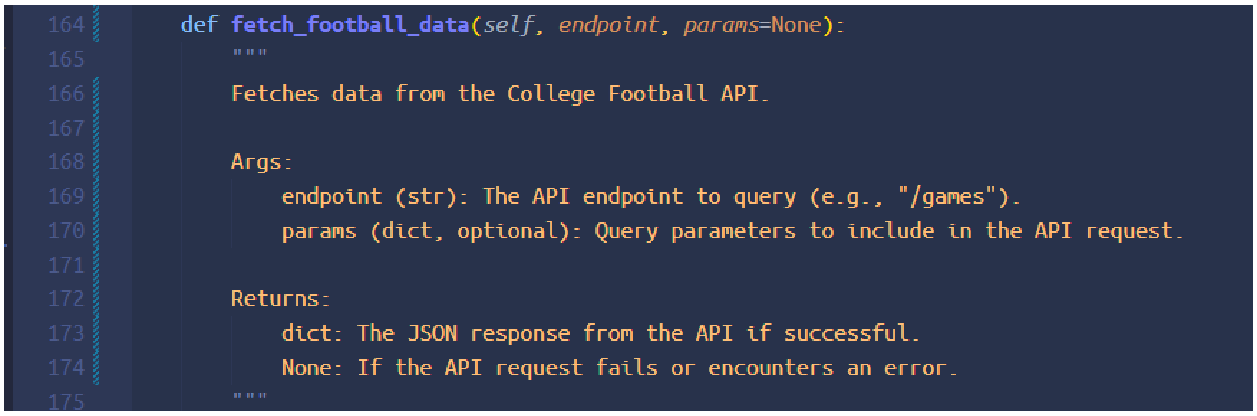
```
class CollegeFootballData:
    def __init__(self, api_key):
        """
        Initialize the CollegeFootballData class with API key and base URL.
        :param api_key: API key for authentication
        :param base_url: Base URL of the API
        """
```



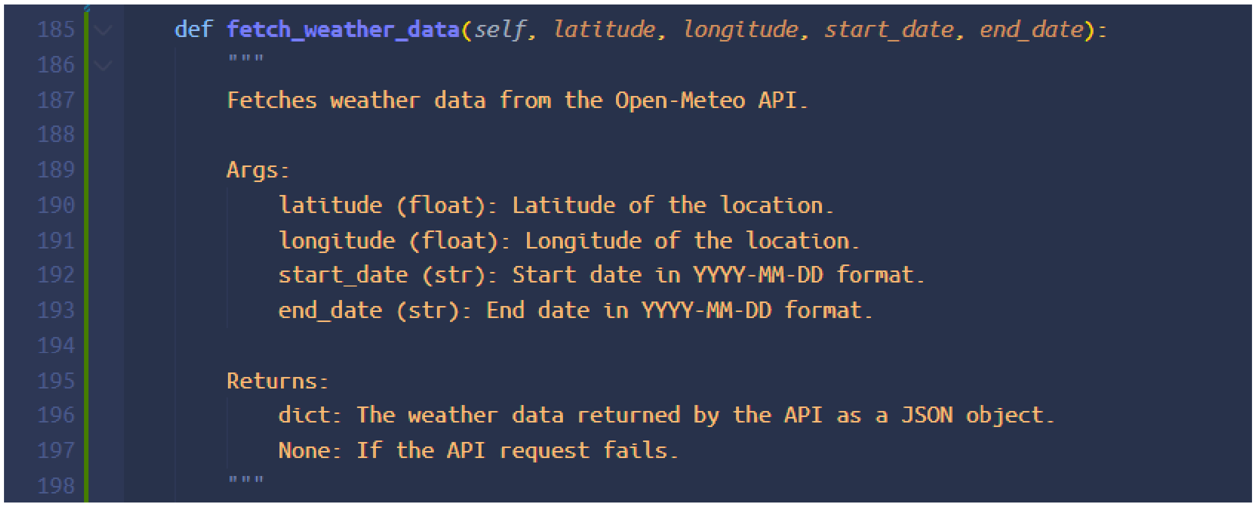
```
    def create_database(self):
        """
        Creates the SQLite database with tables for HomeAway and Games.

        Args:
            self: Instance of the class          You, 1 second ago • Uncommitted

        Returns:
            None
        """
```



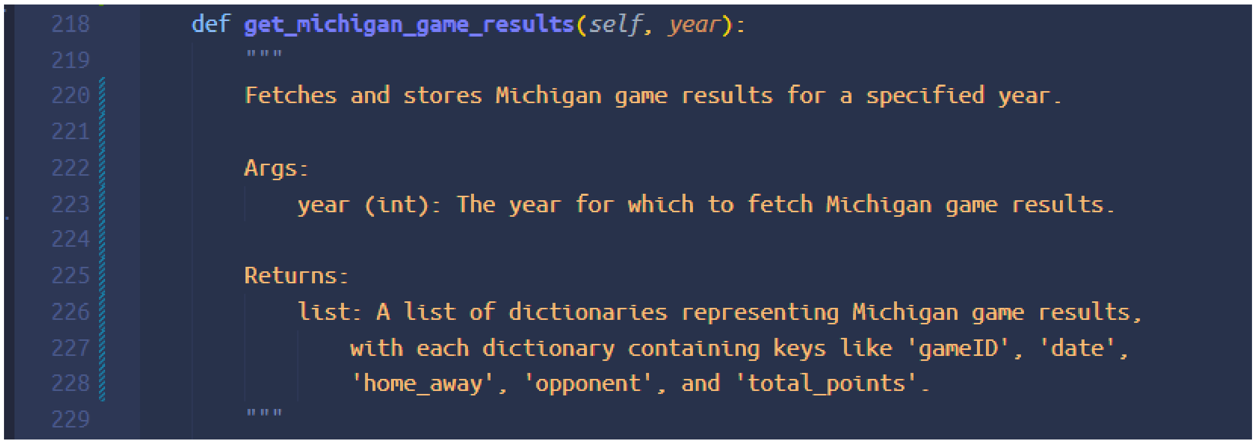
```
    def insert_game_data(self, game_data):
        """
        Inserts up to 25 game results and stats into the database, avoiding duplicates.

        Args:
            game_data (list): A list of dictionaries, each containing game results
                                and stats with keys such as 'gameID', 'date',
                                'home_away', and various performance metrics.

        Returns:
            None
        """
```



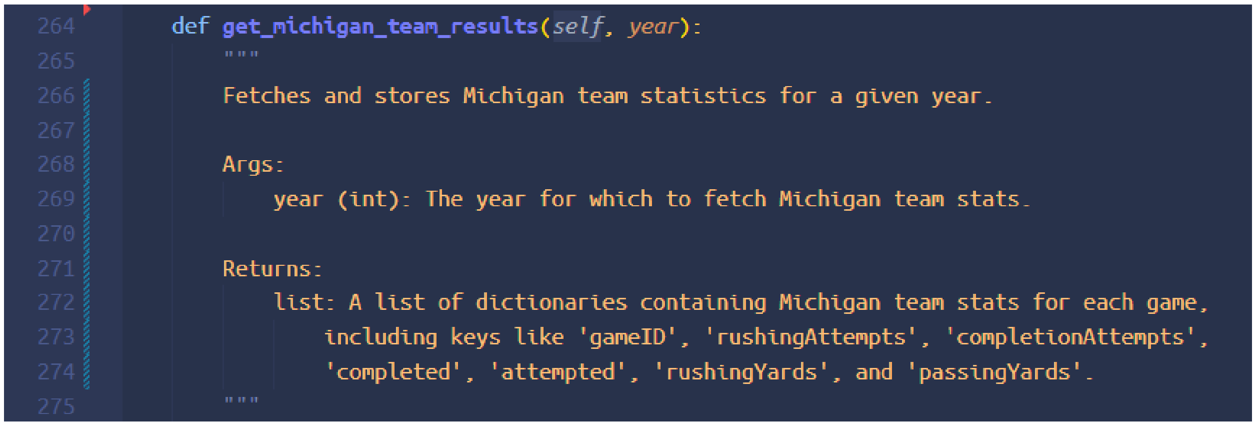
```
    def fetch_football_data(self, endpoint, params=None):
        """
        Fetches data from the College Football API.

        Args:
            endpoint (str): The API endpoint to query (e.g., "/games").
            params (dict, optional): Query parameters to include in the API request.

        Returns:
            dict: The JSON response from the API if successful.
            None: If the API request fails or encounters an error.
        """
```



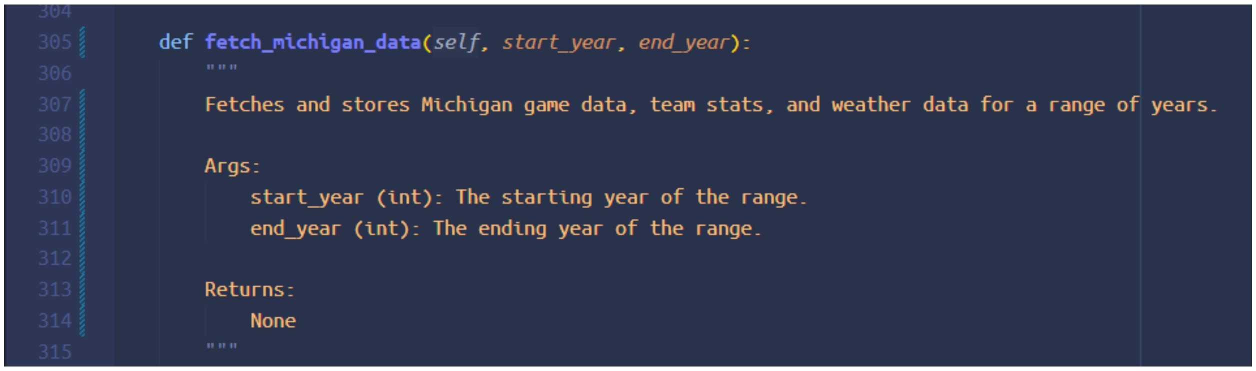
```python
    def fetch_weather_data(self, latitude, longitude, start_date, end_date):
        """
        Fetches weather data from the Open-Meteo API.

        Args:
            latitude (float): Latitude of the location.
            longitude (float): Longitude of the location.
            start_date (str): Start date in YYYY-MM-DD format.
            end_date (str): End date in YYYY-MM-DD format.

        Returns:
            dict: The weather data returned by the API as a JSON object.
            None: If the API request fails.
        """
```



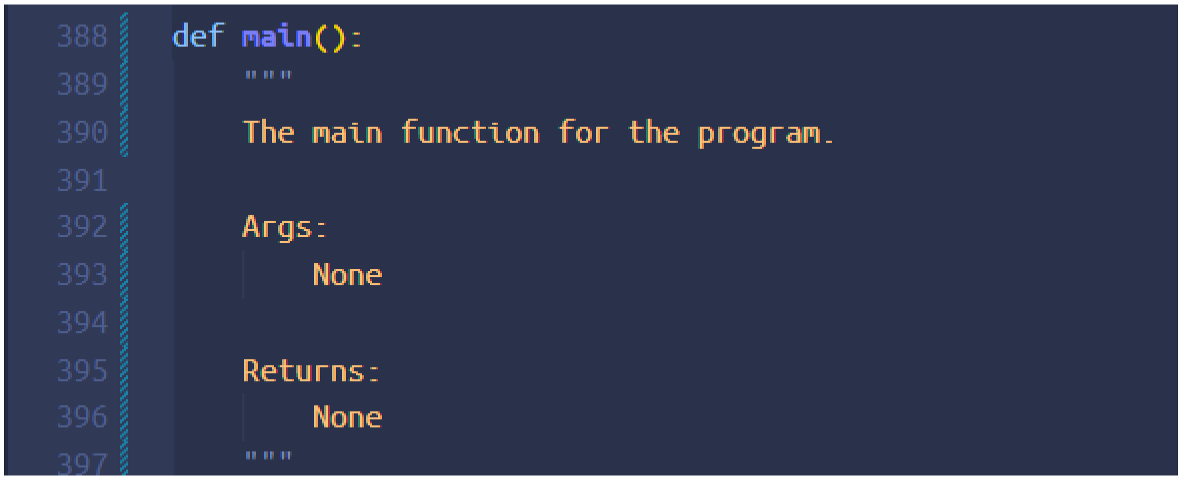
```python
    def get_michigan_game_results(self, year):
        """
        Fetches and stores Michigan game results for a specified year.

        Args:
            year (int): The year for which to fetch Michigan game results.

        Returns:
            list: A list of dictionaries representing Michigan game results,
                with each dictionary containing keys like 'gameID', 'date',
                'home_away', 'opponent', and 'total_points'.
        """
```



```python
    def get_michigan_team_results(self, year):
        """
        Fetches and stores Michigan team statistics for a given year.

        Args:
            year (int): The year for which to fetch Michigan team stats.

        Returns:
            list: A list of dictionaries containing Michigan team stats for each game,
                including keys like 'gameID', 'rushingAttempts', 'completionAttempts',
                'completed', 'attempted', 'rushingYards', and 'passingYards'.
        """
```

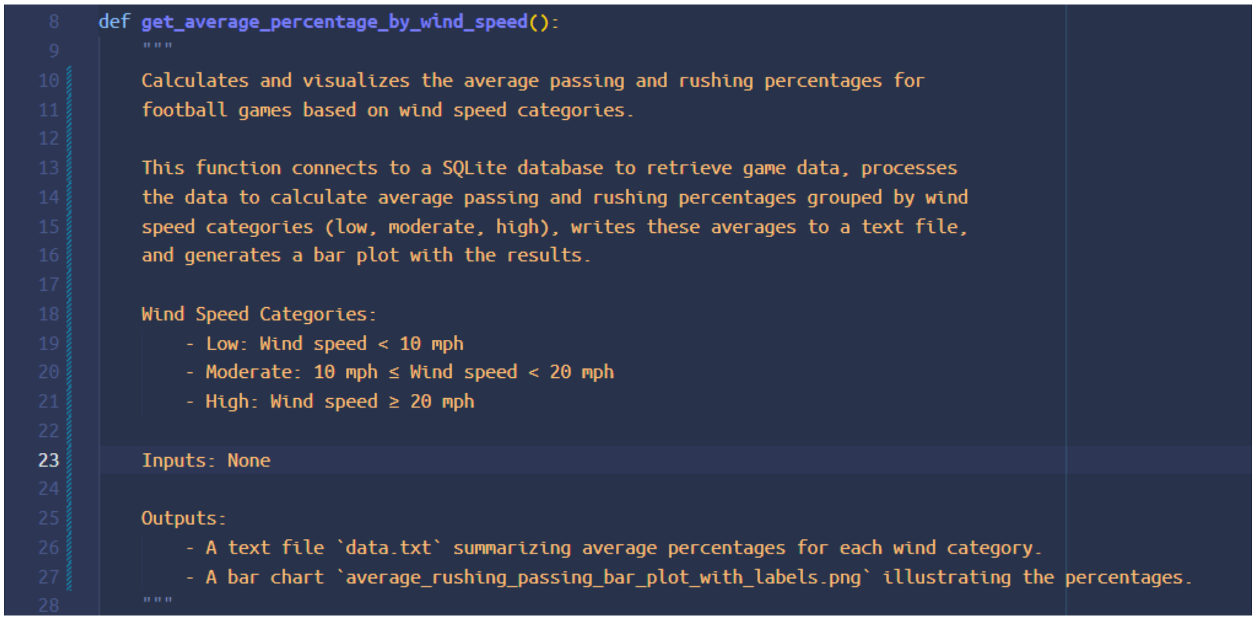


```python
    def fetch_michigan_data(self, start_year, end_year):
        """
        Fetches and stores Michigan game data, team stats, and weather data for a range of years.

        Args:
            start_year (int): The starting year of the range.
            end_year (int): The ending year of the range.

        Returns:
            None
        """
```

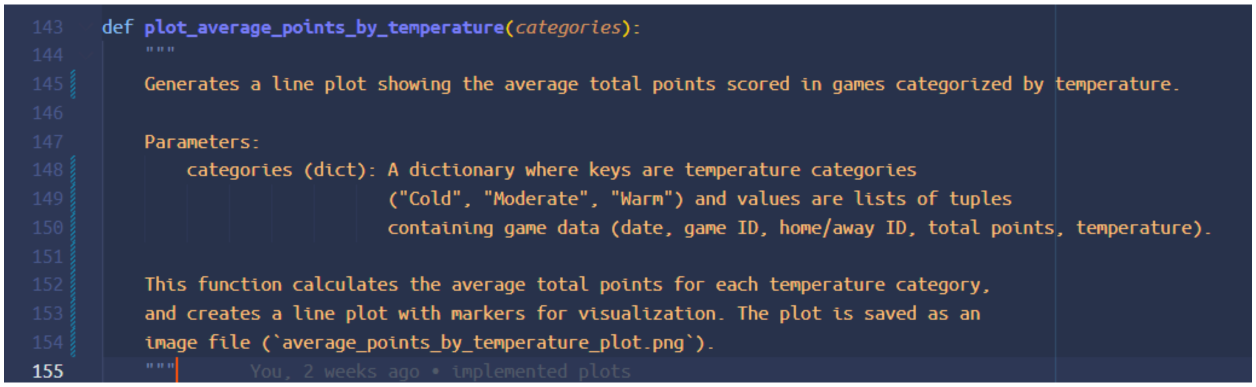


```python
def main():
    """
    The main function for the program.

    Args:
        None

    Returns:
        None
    """
```

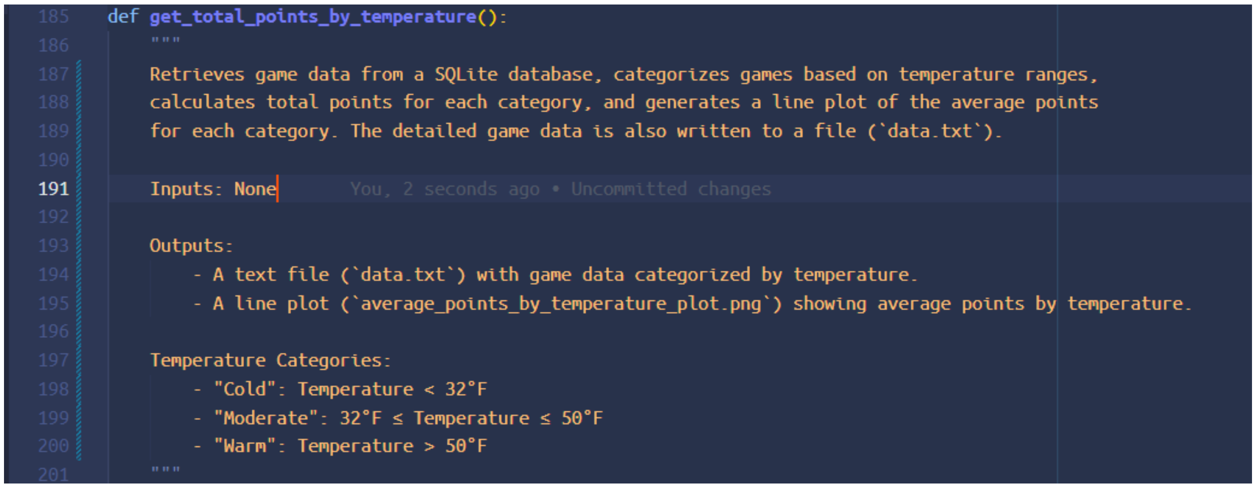
calculations.py



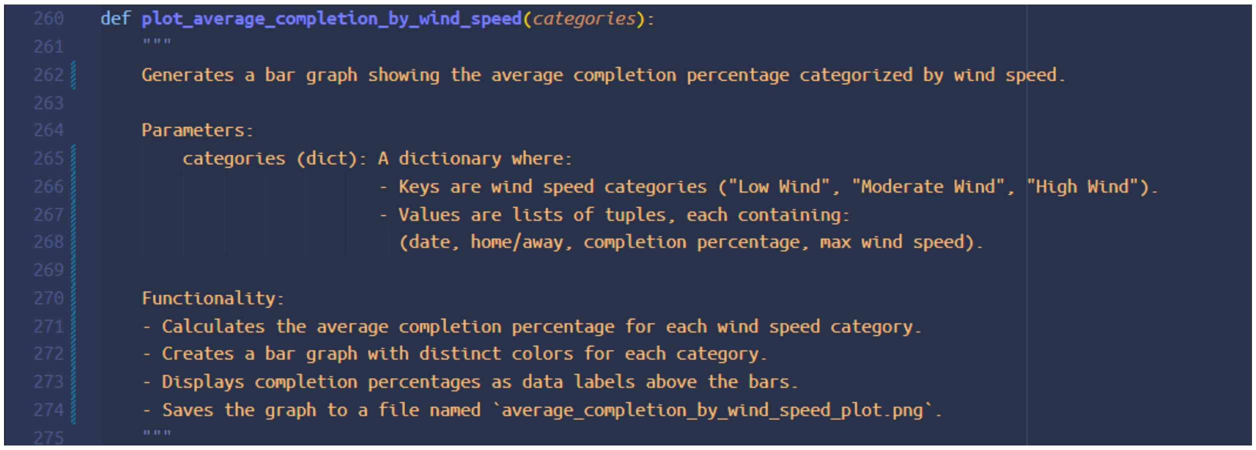
```python
def get_average_percentage_by_wind_speed():
    """
    Calculates and visualizes the average passing and rushing percentages for
    football games based on wind speed categories.

    This function connects to a SQLite database to retrieve game data, processes
    the data to calculate average passing and rushing percentages grouped by wind
    speed categories (low, moderate, high), writes these averages to a text file,
    and generates a bar plot with the results.

    Wind Speed Categories:
        - Low: Wind speed < 10 mph
        - Moderate: 10 mph ≤ Wind speed < 20 mph
        - High: Wind speed ≥ 20 mph

    Inputs: None

    Outputs:
        - A text file `data.txt` summarizing average percentages for each wind category.
        - A bar chart `average_rushing_passing_bar_plot_with_labels.png` illustrating the percentages.
    """
```



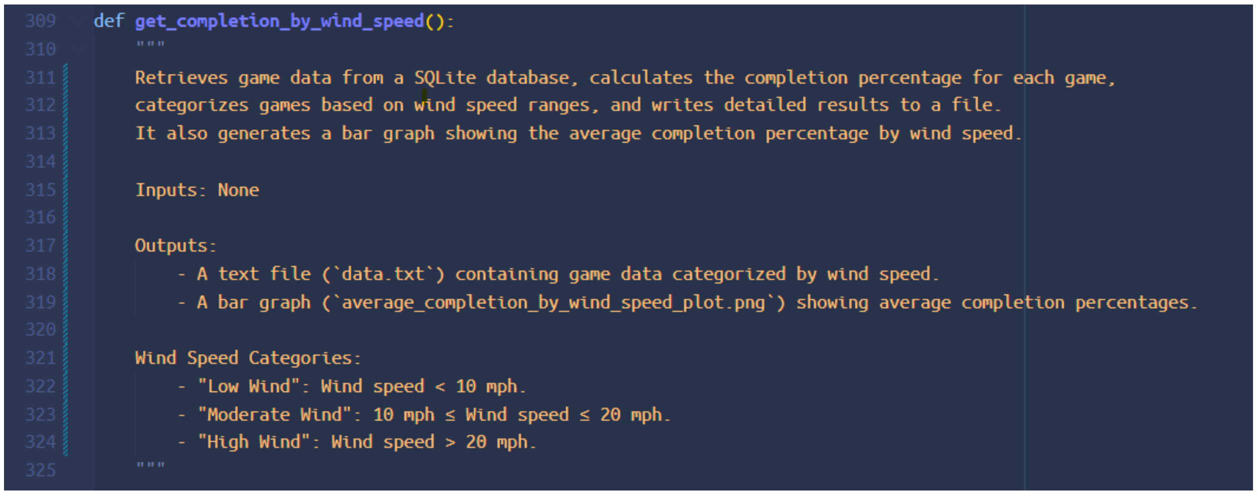
```python
def plot_average_points_by_temperature(categories):
    """
    Generates a line plot showing the average total points scored in games categorized by temperature.

    Parameters:
        categories (dict): A dictionary where keys are temperature categories
                           ("Cold", "Moderate", "Warm") and values are lists of tuples
                           containing game data (date, game ID, home/away ID, total points, temperature).

    This function calculates the average total points for each temperature category,
    and creates a line plot with markers for visualization. The plot is saved as an
    image file (`average_points_by_temperature_plot.png`).
    """
```



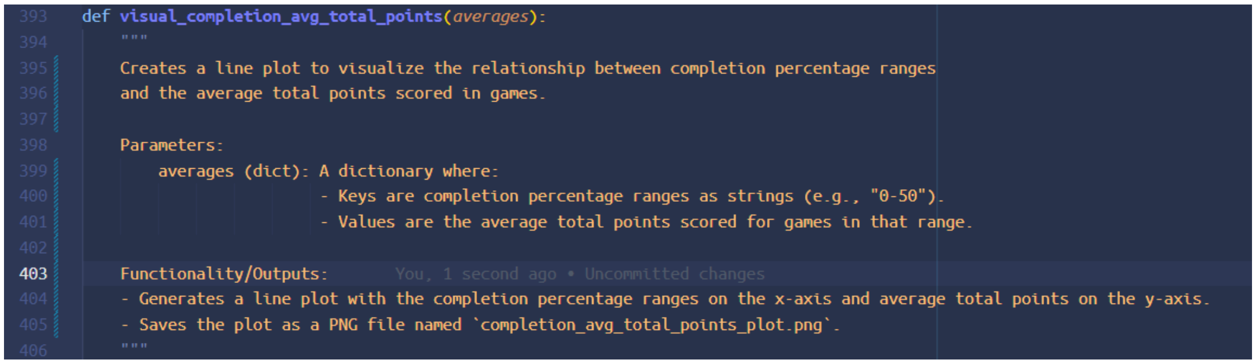
```python
def get_total_points_by_temperature():
    """
    Retrieves game data from a SQLite database, categorizes games based on temperature ranges,
    calculates total points for each category, and generates a line plot of the average points
    for each category. The detailed game data is also written to a file (`data.txt`).

    Inputs: None

    Outputs:
        - A text file (`data.txt`) with game data categorized by temperature.
        - A line plot (`average_points_by_temperature_plot.png`) showing average points by temperature.

    Temperature Categories:
        - "Cold": Temperature < 32°F
        - "Moderate": 32°F ≤ Temperature ≤ 50°F
        - "Warm": Temperature > 50°F
    """
```



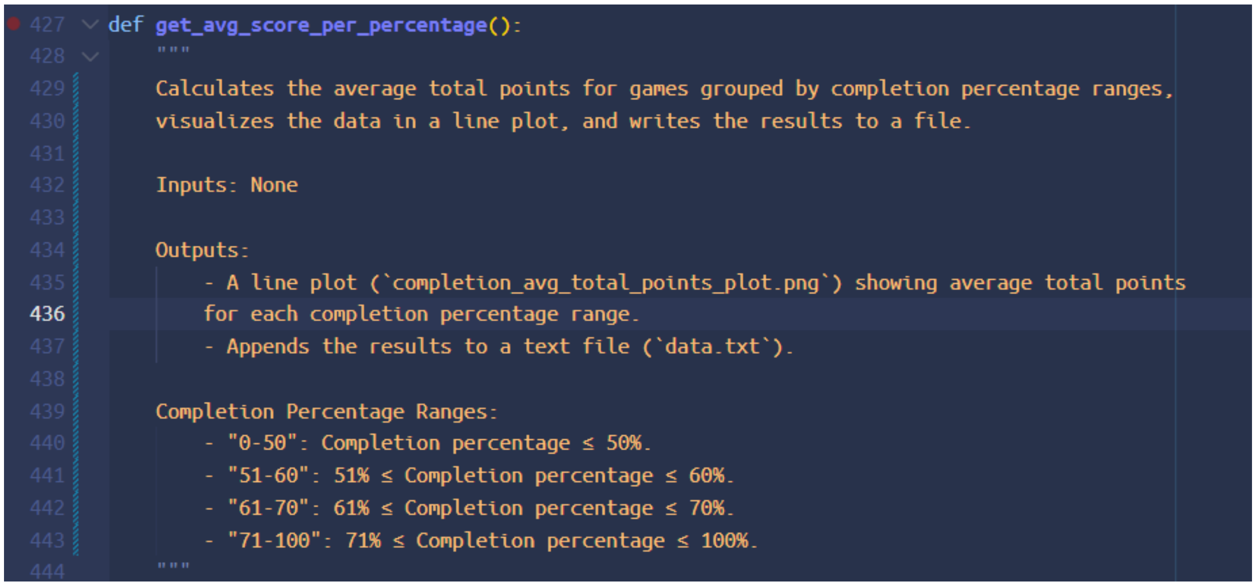
```python
def plot_average_completion_by_wind_speed(categories):
    """
    Generates a bar graph showing the average completion percentage categorized by wind speed.

    Parameters:
        categories (dict): A dictionary where:
                           - Keys are wind speed categories ("Low Wind", "Moderate Wind", "High Wind").
                           - Values are lists of tuples, each containing:
                             (date, home/away, completion percentage, max wind speed).

    Functionality:
    - Calculates the average completion percentage for each wind speed category.
    - Creates a bar graph with distinct colors for each category.
    - Displays completion percentages as data labels above the bars.
    - Saves the graph to a file named `average_completion_by_wind_speed_plot.png`.
    """
```



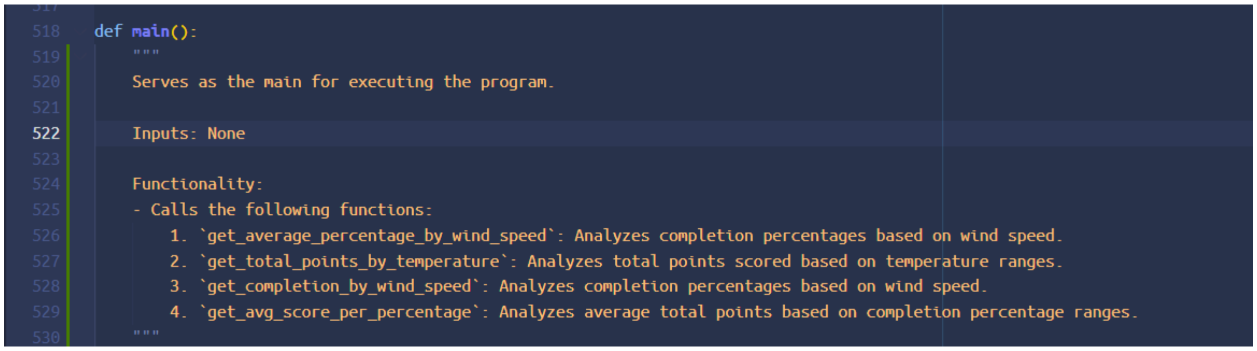
```python
def get_completion_by_wind_speed():
    """
    Retrieves game data from a SQLite database, calculates the completion percentage for each game,
    categorizes games based on wind speed ranges, and writes detailed results to a file.
    It also generates a bar graph showing the average completion percentage by wind speed.

    Inputs: None

    Outputs:
        - A text file (`data.txt`) containing game data categorized by wind speed.
        - A bar graph (`average_completion_by_wind_speed_plot.png`) showing average completion percentages.

    Wind Speed Categories:
        - "Low Wind": Wind speed < 10 mph.
        - "Moderate Wind": 10 mph ≤ Wind speed ≤ 20 mph.
        - "High Wind": Wind speed > 20 mph.
    """
```



```python
def visual_completion_avg_total_points(averages):
    """
    Creates a line plot to visualize the relationship between completion percentage ranges
    and the average total points scored in games.

    Parameters:
        averages (dict): A dictionary where:
                            - Keys are completion percentage ranges as strings (e.g., "0-50").
                            - Values are the average total points scored for games in that range.

    Functionality/Outputs:       You, 1 second ago • Uncommitted changes
    - Generates a line plot with the completion percentage ranges on the x-axis and average total points on the y-axis.
    - Saves the plot as a PNG file named `completion_avg_total_points_plot.png`.
    """
```



```python
def get_avg_score_per_percentage():
    """
    Calculates the average total points for games grouped by completion percentage ranges,
    visualizes the data in a line plot, and writes the results to a file.

    Inputs: None

    Outputs:
        - A line plot (`completion_avg_total_points_plot.png`) showing average total points
        for each completion percentage range.
        - Appends the results to a text file (`data.txt`).

    Completion Percentage Ranges:
        - "0-50": Completion percentage ≤ 50%.
        - "51-60": 51% ≤ Completion percentage ≤ 60%.
        - "61-70": 61% ≤ Completion percentage ≤ 70%.
        - "71-100": 71% ≤ Completion percentage ≤ 100%.
    """
```



```python
def main():
    """
    Serves as the main for executing the program.

    Inputs: None

    Functionality:
    - Calls the following functions:
        1. `get_average_percentage_by_wind_speed`: Analyzes completion percentages based on wind speed.
        2. `get_total_points_by_temperature`: Analyzes total points scored based on temperature ranges.
        3. `get_completion_by_wind_speed`: Analyzes completion percentages based on wind speed.
        4. `get_avg_score_per_percentage`: Analyzes average total points based on completion percentage ranges.
    """
```

Resources

- Resources 1
    - 12/3
    - We didn’t know how to use the API for the football data
    - API website - https://collegefootballdata.com/
    - We were able to learn how to make the correct calls to the API because of the Documentation
- Resources 2
    - 12/4
    - We didn’t know how to use the API for the weather data
    - API website - https://weatherstack.com/?utm_source=Github&
    - We were able to learn how to make the correct calls to the API because of the Documentation
- Resources 3
    - 12/4
    - We didn't know specifics on matplotlib, like how to get the percentages on top of the bars
    - Documentation - https://matplotlib.org/stable/api/index
    - We were able to look through the documentation and find the information we needed.
    -

Data:

```
-------------Average Rushing and Passing Percentage by Wind Speed-------------
Category	Games	Average Pass Percentage	Average Rush Percentage
Low	49	39.95%	60.05%
Moderate	61	40.74%	59.26%
High	7	44.89%	55.11%

-------------Total Points Based on Temperature-------------

--- Cold Games ---
Date	Game ID	Home/Away	Total Points	Temperature
1446872400	400763559	1	49	16.5°F
```

1448082000	400763574	2	28	28.4°F
1448686800	400763578	1	13	21.0°F
1475899200	400869636	2	78	21.6°F
1480136400	400869650	2	27	25.7°F
1477108800	400869676	1	41	25.1°F
1477713600	400869681	2	32	14.5°F
1478318400	400869685	1	59	11.9°F
1479013200	400869690	2	13	13.7°F
1479531600	400869695	1	20	9.8°F
1507348800	400935372	1	10	20.2°F
1509163200	400935393	1	35	22.6°F
1509854400	400935398	1	33	28.1°F
1511586000	400935420	1	20	19.1°F
1541826000	401012848	2	42	11.8°F
1542430800	401012860	1	31	4.3°F
1540008000	401012884	2	21	9.5°F
1539403200	401012893	1	38	31.6°F
1541217600	401012894	1	42	14.0°F
1543035600	401012895	2	39	29.0°F
1574485200	401112167	2	39	30.9°F
1573880400	401112219	1	44	14.4°F
1572062400	401112227	1	45	25.2°F
1575090000	401112228	1	27	5.2°F
1604116800	401247305	1	24	24.1°F
1605416400	401247319	1	11	29.4°F
1606021200	401247324	2	48	-6.0°F
1606539600	401247333	1	17	22.8°F
1636171200	401282727	1	29	5.9°F
1637384400	401282770	2	59	31.6°F
1635566400	401282777	2	33	30.9°F
1633752000	401282778	2	32	31.9°F
1634961600	401282779	1	33	23.3°F
1636779600	401282780	2	21	30.5°F
1637989200	401282781	1	42	6.3°F
1638680400	401331447	2	42	24.5°F
1667016000	401405125	1	29	24.9°F
1667620800	401405133	2	52	19.5°F
1668229200	401405137	1	34	9.7°F
1668834000	401405145	1	19	18.6°F
1669438800	401405153	2	45	31.2°F

1670130000	401437031	1	43	10.3°F
1699678800	401520394	2	24	25.8°F
1700888400	401520434	1	30	25.2°F
1701579600	401539480	2	26	24.9°F
1727496000	401628497	1	27	26.3°F
1729915200	401628529	1	24	22.0°F
1730520000	401628536	1	17	22.3°F
1732338000	401628557	1	50	9.0°F
1732942800	401628566	2	13	23.9°F

--- Moderate Games ---

Date	Game ID	Home/Away	Total Points	Temperature
1442030400	400757019	1	35	43.3°F
1442635200	400763511	1	28	46.5°F
1443240000	400763519	1	31	34.8°F
1444449600	400763535	1	38	44.2°F
1445054400	400763542	1	23	38.9°F
1446264000	400763553	2	29	32.1°F
1447477200	400763565	2	48	38.7°F
1473480000	400869510	1	51	49.4°F
1474689600	400869658	1	49	45.8°F
1475294400	400869664	1	14	48.2°F
1504324800	400933830	2	33	40.9°F
1504929600	400935243	1	36	46.3°F
1505534400	400935352	1	29	44.8°F
1508558400	400935386	2	13	39.2°F
1510376400	400935404	2	35	46.9°F
1510981200	400935416	2	10	38.9°F
1538798400	401012876	1	42	35.3°F
1536379200	401012889	1	49	47.1°F
1536984000	401012890	1	45	36.9°F
1537588800	401012891	1	56	43.6°F
1570248000	401112194	1	10	36.0°F
1572667200	401112208	2	38	48.6°F
1567828800	401112223	1	24	43.6°F
1569643200	401112225	1	52	48.3°F
1571457600	401112226	2	21	41.3°F
1603512000	401247294	2	49	41.5°F
1604725200	401247309	2	21	42.6°F
1630728000	401282772	1	47	45.1°F

```
1631419200    401282773    1       31      43.4°F
1631937600    401282774    1       63      46.4°F
1632542400    401282775    1       20      38.3°F
1663387200    401404149    1       59      46.7°F
1662177600    401405067    1       51      46.1°F
1663992000    401405097    1       34      43.0°F
1664596800    401405100    2       27      34.2°F
1665806400    401405115    1       41      34.7°F
1694232000    401520202    1       35      48.1°F
1694836800    401520232    1       31      41.4°F
1695441600    401520260    1       31      32.8°F
1696651200    401520303    2       52      49.1°F
1697256000    401520321    1       52      35.1°F
1697860800    401520340    2       49      35.0°F
1699070400    401520368    1       41      32.6°F
1700283600    401520410    2       31      37.6°F
1725681600    401628347    1       12      42.1°F
1725076800    401628452    1       30      47.4°F
1726286400    401628479    1       28      41.3°F
1726891200    401628489    1       27      47.1°F
1728100800    401628505    2       17      49.3°F
1731128400    401628541    2       15      45.1°F

--- Warm Games ---
Date     Game ID        Home/Away    Total Points    Temperature
1441339200    400756883    2       17      61.5°F
1443844800    400763530    2       28      64.2°F
1474084800    400869111    1       45      50.1°F
1472875200    400869180    1       63      51.9°F
1506139200    400935364    2       28      72.3°F
1507953600    400935379    2       27      53.8°F
1535774400    401012888    2       17      69.0°F
1538193600    401012892    2       20      60.9°F
1570852800    401112150    2       42      62.0°F
1567224000    401112222    1       40      53.6°F
1569038400    401112224    2       14      80.8°F
1633147200    401282776    2       38      60.6°F
1662868800    401405077    1       56      52.4°F
1665201600    401405108    2       31      58.1°F
1693627200    401520162    1       30      50.8°F
```

```
1696046400    401520286     2          45        57.3°F
1729310400    401628518     2          7         53.6°F

-------------Completion Percentage Based on Wind Speed-------------

--- Low Wind ---
Date     Home/Away    Completion Percentage          Max Wind Speed
1444449600    1        73.91%           8.6 mph
1445054400    1        60.00%           7.6 mph
1446872400    1        69.23%           7.2 mph
1448686800    1        53.19%           7.6 mph
1474084800    1        53.33%           7.2 mph
1472875200    1        85.00%           8.2 mph
1473480000    1        65.85%           9.0 mph
1480136400    2        63.89%           9.1 mph
1477713600    2        64.00%           5.2 mph
1479013200    2        42.31%           5.2 mph
1479531600    1        43.75%           8.7 mph
1504324800    2        46.15%           6.8 mph
1505534400    1        60.87%           9.1 mph
1508558400    2        57.14%           9.5 mph
1509163200    1        65.00%           6.1 mph
1511586000    1        53.12%           4.8 mph
1541826000    2        66.67%           5.6 mph
1542430800    1        57.14%           9.4 mph
1538798400    1        71.43%           8.1 mph
1540008000    2        56.00%           8.5 mph
1536984000    1        77.78%           9.4 mph
1537588800    1        58.06%           9.5 mph
1541217600    1        64.71%           8.4 mph
1543035600    2        60.53%           9.8 mph
1570852800    2        50.00%           9.4 mph
1570248000    1        53.85%           9.7 mph
1567224000    1        57.58%           9.1 mph
1571457600    2        58.54%           6.9 mph
1572062400    1        57.14%           6.9 mph
1575090000    1        41.86%           5.6 mph
1604116800    1        61.54%           7.0 mph
1605416400    1        50.00%           9.2 mph
1606021200    2        66.67%           5.2 mph
```

1636171200	1	53.57%	7.7 mph
1630728000	1	76.47%	8.0 mph
1635566400	2	64.58%	5.4 mph
1636779600	2	65.52%	7.3 mph
1638680400	2	64.29%	6.5 mph
1663387200	1	80.77%	7.5 mph
1665806400	1	70.83%	7.4 mph
1667620800	2	48.15%	3.4 mph
1668229200	1	50.00%	9.4 mph
1670130000	1	64.71%	5.0 mph
1695441600	1	71.43%	9.8 mph
1697256000	1	86.36%	9.5 mph
1700283600	2	52.17%	9.0 mph
1700888400	1	80.95%	7.9 mph
1701579600	2	73.33%	8.7 mph
1729915200	1	70.00%	7.9 mph

--- Moderate Wind ---

Date	Home/Away	Completion Percentage	Max Wind Speed
1441339200	2	62.79%	16.2 mph
1442030400	1	69.23%	10.2 mph
1442635200	1	56.00%	14.8 mph
1443240000	1	56.00%	11.9 mph
1443844800	2	48.48%	14.0 mph
1446264000	2	59.26%	11.7 mph
1447477200	2	71.74%	12.8 mph
1448082000	2	65.79%	12.4 mph
1475899200	2	50.00%	11.3 mph
1474689600	1	60.00%	10.8 mph
1475294400	1	62.50%	12.1 mph
1477108800	1	71.43%	10.7 mph
1478318400	1	77.78%	16.0 mph
1504929600	1	58.62%	12.7 mph
1506139200	2	67.74%	12.3 mph
1507348800	1	45.71%	15.5 mph
1507953600	2	50.00%	16.3 mph
1509854400	1	61.54%	17.6 mph
1510376400	2	50.00%	15.3 mph
1510981200	2	42.31%	10.9 mph
1536379200	1	72.22%	11.7 mph

1538193600	2	62.50%	17.0 mph
1539403200	1	70.00%	12.6 mph
1572667200	2	55.17%	10.8 mph
1573880400	1	72.73%	18.2 mph
1567828800	1	64.52%	10.5 mph
1569038400	2	40.48%	16.1 mph
1569643200	1	74.07%	12.4 mph
1603512000	2	60.00%	15.3 mph
1604725200	2	52.94%	10.5 mph
1606539600	1	46.43%	14.9 mph
1637384400	2	72.50%	17.8 mph
1631419200	1	46.67%	11.1 mph
1631937600	1	70.59%	10.6 mph
1632542400	1	56.25%	16.9 mph
1633752000	2	56.41%	19.1 mph
1634961600	1	71.88%	11.6 mph
1637989200	1	70.00%	19.1 mph
1662177600	1	64.29%	11.5 mph
1662868800	1	77.27%	10.6 mph
1663992000	1	69.23%	11.6 mph
1664596800	2	75.00%	14.1 mph
1665201600	2	77.78%	12.0 mph
1667016000	1	57.69%	11.4 mph
1668834000	1	52.94%	16.0 mph
1669438800	2	52.00%	15.5 mph
1693627200	1	83.87%	15.6 mph
1694232000	1	82.14%	12.2 mph
1694836800	1	61.54%	10.0 mph
1696651200	2	69.57%	12.2 mph
1697860800	2	80.00%	11.2 mph
1699070400	1	64.86%	10.5 mph
1699678800	2	87.50%	11.5 mph
1725076800	1	59.26%	10.3 mph
1726286400	1	72.22%	10.6 mph
1726891200	1	58.33%	13.0 mph
1728100800	2	52.00%	16.9 mph
1729310400	2	62.50%	17.2 mph
1730520000	1	52.00%	13.0 mph
1732338000	1	74.29%	12.2 mph
1732942800	2	56.25%	12.2 mph

```
--- High Wind ---
Date      Home/Away   Completion Percentage        Max Wind Speed
1535774400    2         66.67%          23.1 mph
1574485200    2         62.50%          29.9 mph
1633147200    2         60.00%          31.9 mph
1696046400    2         69.57%          41.1 mph
1725681600    1         66.67%          22.5 mph
1727496000    1         55.56%          22.5 mph
1731128400    2         50.00%          23.8 mph

-------------Average Score Per Completion Percentage Range-------------
Range  Average Total Points
0-50    27.61
51-60   29.21
61-70   32.97
71-100 42.52
```