Spring 2019

Basketball Charts

Kevin Lewis
kjl48@zips.uakron.edu

Introduction

Across the internet, there are numerous web sites and applications that provide access to basketball statistics. As statistics and analytics continue to play an increasingly significant role in sports' strategy, these sites provide a very convenient resource for allowing the common fan to do his or her own research into some of the data that is available. However, the primary drawback of a lot of these sites is that they only provide the raw data with no convenient means of interpreting it. If a user is interested in drawing visual relations and conclusions from the data, the user must then extract the data manually and use some third-party software to develop their representation. This shortcoming that is present in so many sports sites across the web served as the central inspiration for this project and ultimately the creation of the website called *Basketball Charts*.

The main obstacle in developing *Basketball Charts* was finding a reliable data source that was easy to access as well as developing a method of automatically updating this data on a daily basis. The NBA and other professional sports leagues are notorious for not exposing their more advanced statistics to the general public. The best source of data is available directly from a site called *NBA Stats* where a variety of undocumented endpoints exist that meet the needs of this project. How this data was eventually accessed and the difficulties in doing so will be further described in the development section of this report. Aside from the collection of data, the creation of dynamic SQL queries that respond to user filters was another challenging aspect of this project and will also be described in detail later on.

The core functionality of the *Basketball Charts* is the chart builder tool that allows users to make use of the acquired data to generate unique visuals to their specifications. At the moment, scatter plots, pie charts, and column charts are the available chart types with more being planned for the future. Once the user selects a chart type, that user is then able to filter the pool of included players by team, experience, and position among other options. All standard box score stats are available as parameters for the chart with the goal being to add even more in the future. The scatter plot and bar charts allow the user to choose two statistical categories to search for relationships between them or just to compare certain players. The pie chart takes just one statistical category and is a great way to see which players excel in certain areas.

The project itself was built with JavaScript and PHP while also utilizing Bootstrap as a front-end framework. The data used in this project was stored in a MySQL database and is self-updated through a Python script that is scheduled to run on a daily basis. Highcharts, a tool for developing interactive JavaScript charts, was also used. *Basketball Charts* can be found at http://www.bball-charts.com.

Development

The development of this project can be broken down into four phases that will be described subsequently. These four phases are:

- Data Collection
- Database Design
- Server-Side Communication
- Front-End Construction

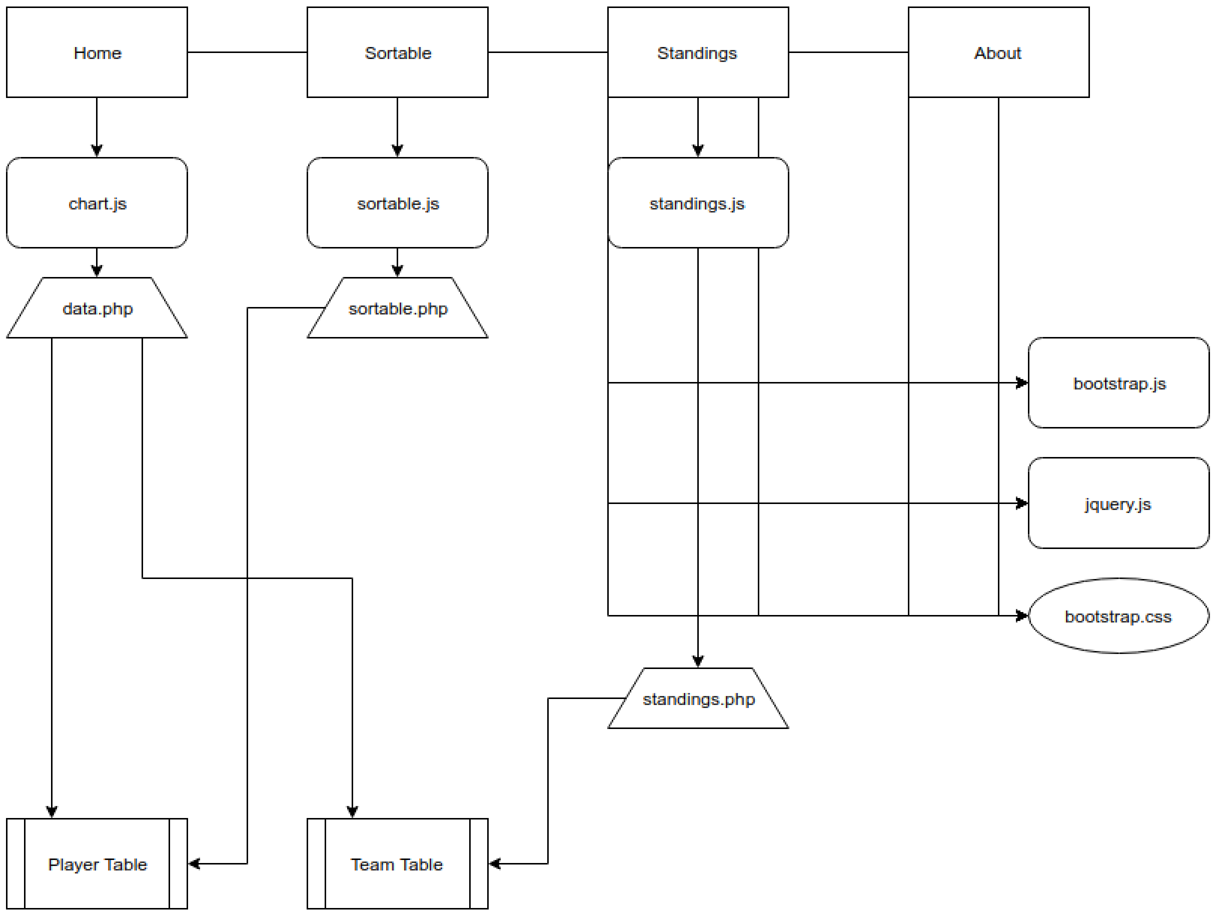
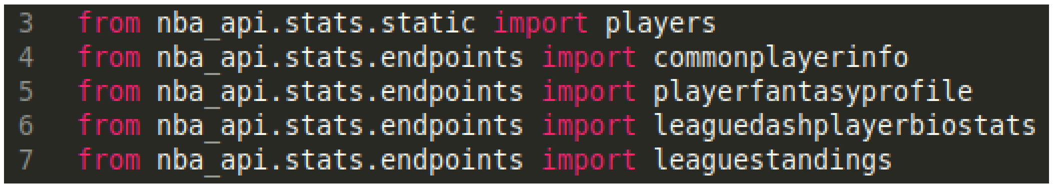


Figure 1: A simple diagram describing the structure of the site

Data Collection

While there are many places to acquire statistical data from the NBA, the official *NBA Stats* website is the most complete source and provides plenty of data that is not publicly available anywhere else. Although endpoints to access this data exist, they have unfortunately never been officially documented. For this reason, a few different community driven APIs are being developed on GitHub that attempt to remedy this problem. This project is using an API called *nba_api* that has documented many of the available endpoints from the *NBA Stats* site.

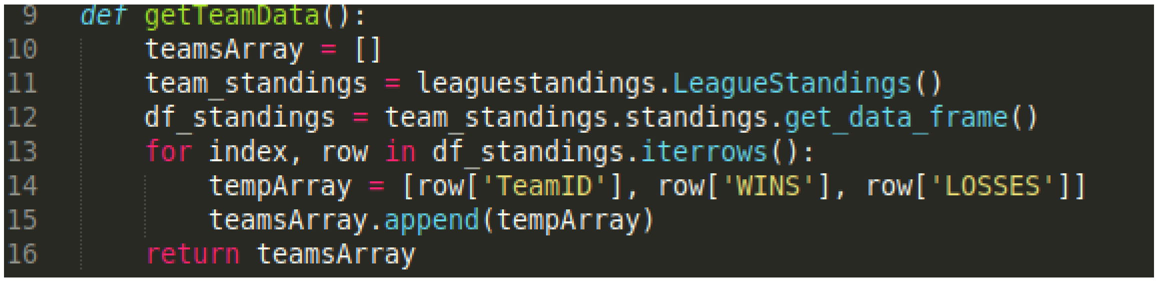
Since the API is written in Python, the script used by this project to collect data is also written in Python. The project in its current state accesses four endpoints to obtain the majority of its data.



```python
from nba_api.stats.static import players
from nba_api.stats.endpoints import commonplayerinfo
from nba_api.stats.endpoints import playerfantasyprofile
from nba_api.stats.endpoints import leaguedashplayerbiostats
from nba_api.stats.endpoints import leaguestandings
```

Figure 2: Endpoint imports

The league standings endpoint is used first and simply collects the win/loss totals of all 30 teams in the NBA as follows:

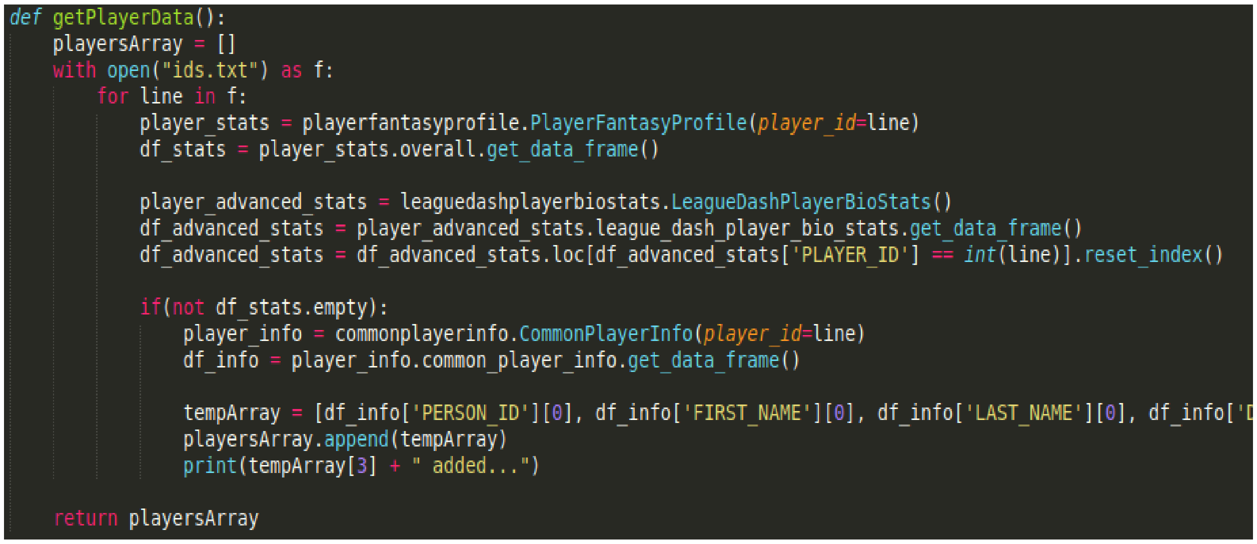


```python
def getTeamData():
    teamsArray = []
    team_standings = leaguestandings.LeagueStandings()
    df_standings = team_standings.standings.get_data_frame()
    for index, row in df_standings.iterrows():
        tempArray = [row['TeamID'], row['WINS'], row['LOSSES']]
        teamsArray.append(tempArray)
    return teamsArray
```

Figure 3: Function to access team info and standings

Essentially, the endpoint is accessed and then reformatted using a software library called *Pandas*. From here, the newly created data frame is iterated over and information for each team is stored in an array for future use.

Once the team data is acquired, the script then collects the player data:

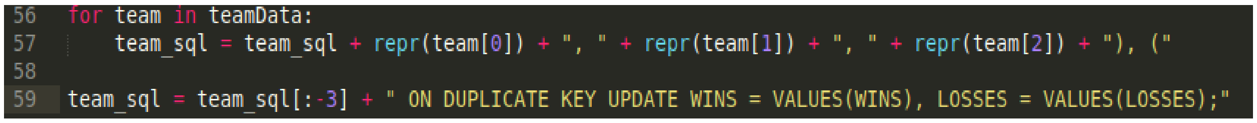


```python
def getPlayerData():
    playersArray = []
    with open("ids.txt") as f:
        for line in f:
            player_stats = playerfantasyprofile.PlayerFantasyProfile(player_id=line)
            df_stats = player_stats.overall.get_data_frame()

            player_advanced_stats = leaguedashplayerbiostats.LeagueDashPlayerBioStats()
            df_advanced_stats = player_advanced_stats.league_dash_player_bio_stats.get_data_frame()
            df_advanced_stats = df_advanced_stats.loc[df_advanced_stats['PLAYER_ID'] == int(line)].reset_index()

            if(not df_stats.empty):
                player_info = commonplayerinfo.CommonPlayerInfo(player_id=line)
                df_info = player_info.common_player_info.get_data_frame()

                tempArray = [df_info['PERSON_ID'][0], df_info['FIRST_NAME'][0], df_info['LAST_NAME'][0], df_info['D
                playersArray.append(tempArray)
                print(tempArray[3] + " added...")

    return playersArray
```

Figure 4: Function to access player info and statistics

A similar process is followed here in Figure 4 as when the team data was collected. The endpoints are accessed and then reformatted into easier to manipulate data frames. The common player info endpoint collects basic info like the players name, position, and jersey number. The player fantasy profile endpoint gathers basic box score data. The league dash player bio stats endpoint obtains some more advanced stats such as true shooting percentage and player plus/minus. This data is all stored into an array like the team data and saved for future use.

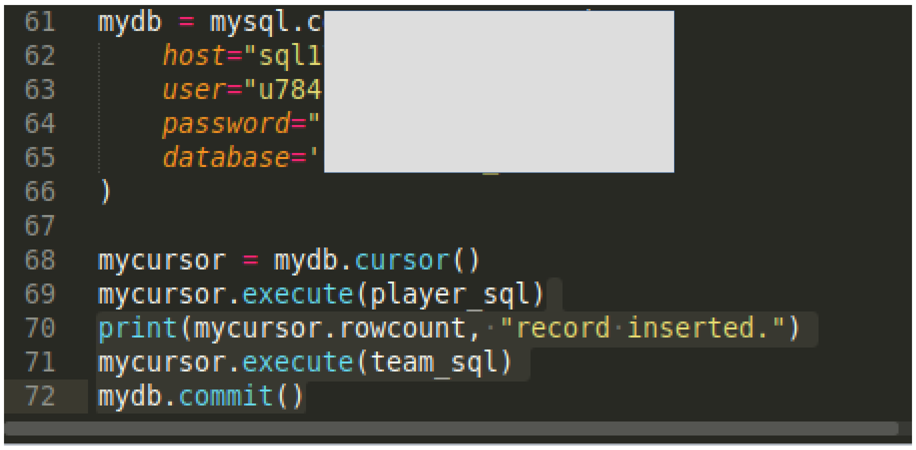
With both the player and team data stored in separate arrays, the data in these arrays is used to create two SQL statements.



```
for team in teamData:
    team_sql = team_sql + repr(team[0]) + ", " + repr(team[1]) + ", " + repr(team[2]) + "), ("

team_sql = team_sql[:-3] + " ON DUPLICATE KEY UPDATE WINS = VALUES(WINS), LOSSES = VALUES(LOSSES);"
```

Figure 5: A for loop to construct a query that updates the team data

Using another python library that allows for connections to MySQL databases to be established, the SQL statements that were just created are executed to update the database used by *Basketball Charts*.



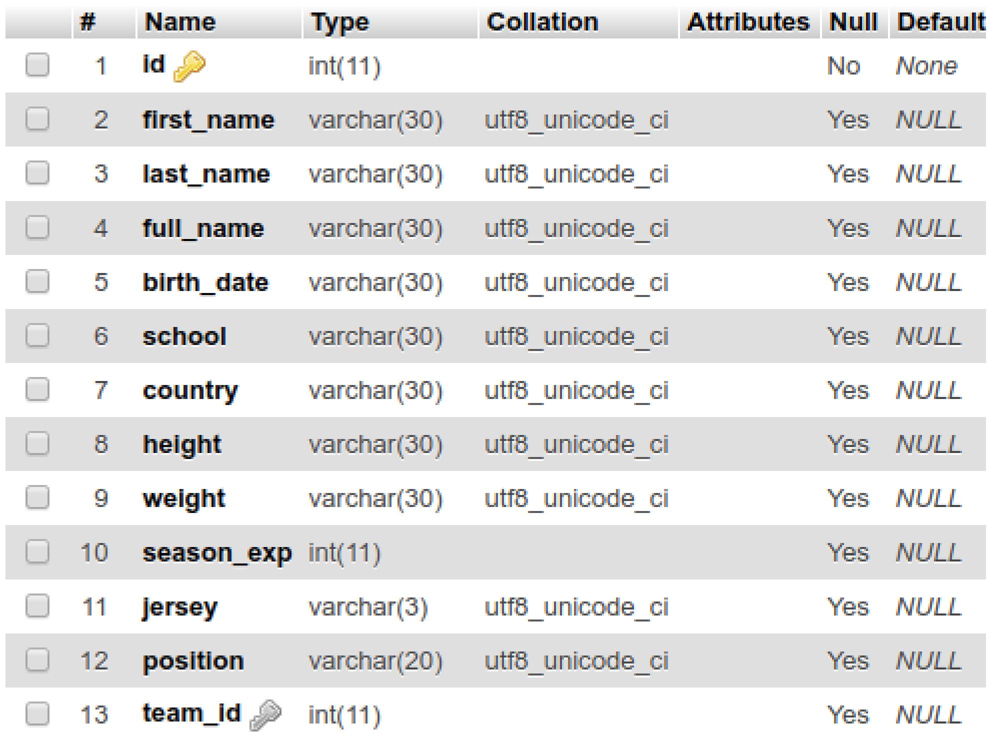
```
mydb = mysql.c[redacted]
    host="sql1
    user="u784
    password="
    database='
)

mycursor = mydb.cursor()
mycursor.execute(player_sql)
print(mycursor.rowcount, "record inserted.")
mycursor.execute(team_sql)
mydb.commit()
```

Figure 6: A connection to a MySQL database is established and the queries are executed

This entire process takes approximately 20 minutes. In order to make sure the project data is always up to date, the script is scheduled to run everyday at 1AM.

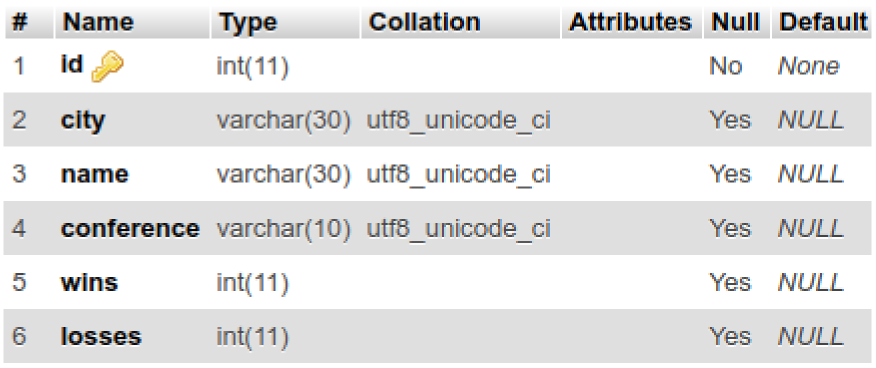
Database Design

The data used by *Basketball Charts* is stored in a MySQL database. At the moment, there is currently a player table and a team table that contains all the various data mentioned in the previous section. The team table contains 30 records and the player table consists of all 483 players that were active in the NBA this year.



	#	Name	Type	Collation	Attributes	Null	Default
☐	1	**id**	int(11)			No	*None*
☐	2	**first_name**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	3	**last_name**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	4	**full_name**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	5	**birth_date**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	6	**school**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	7	**country**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	8	**height**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	9	**weight**	varchar(30)	utf8_unicode_ci		Yes	*NULL*
☐	10	**season_exp**	int(11)			Yes	*NULL*
☐	11	**jersey**	varchar(3)	utf8_unicode_ci		Yes	*NULL*
☐	12	**position**	varchar(20)	utf8_unicode_ci		Yes	*NULL*
☐	13	**team_id**	int(11)			Yes	*NULL*

Figure 7: Player Table Structure



#	Name	Type	Collation	Attributes	Null	Default
1	id	int(11)			No	None
2	city	varchar(30)	utf8_unicode_ci		Yes	NULL
3	name	varchar(30)	utf8_unicode_ci		Yes	NULL
4	conference	varchar(10)	utf8_unicode_ci		Yes	NULL
5	wins	int(11)			Yes	NULL
6	losses	int(11)			Yes	NULL

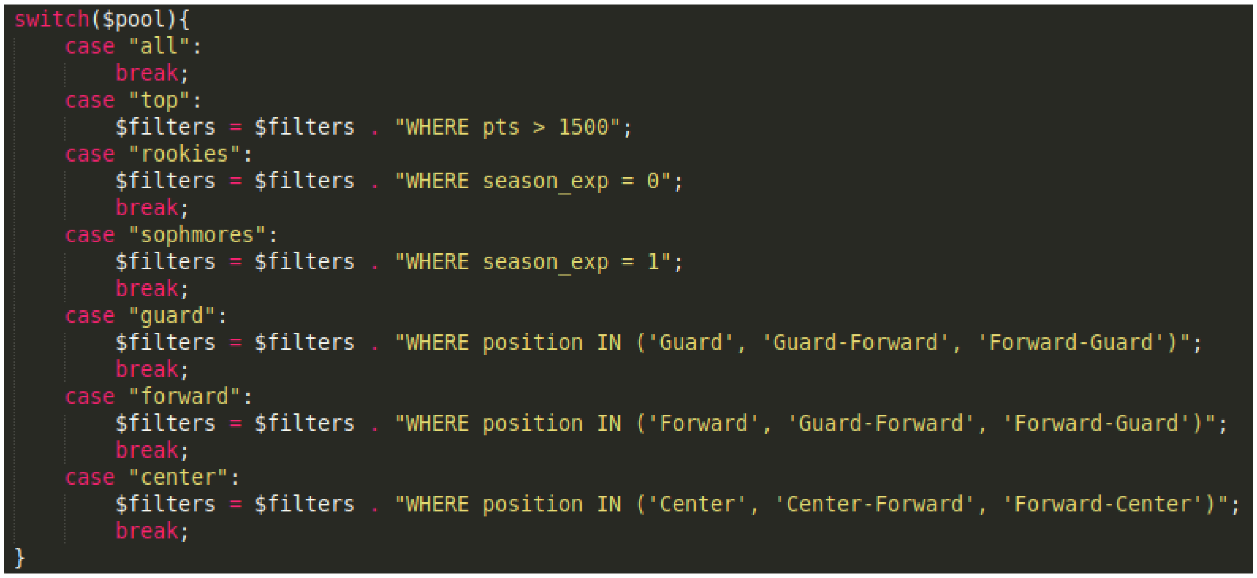
Figure 8: Team Table Structure

Looking at Figure 7, the player table consists of a primary key called *id* and a foreign key called *team_id*. The primary key utilizes the same values as the IDs assigned to each player by *NBA Stats*. This allows for the player data to be updated quickly with no translation process when accessing individual players. The foreign key associates the player with a team from the team table (Figure 8). The team table also contains a unique ID as its primary key which is once again pulled directly from *NBA Stats*. All other values in both tables are either string values or numeric values accessed from one of the many endpoints made available by *NBA Stats*.

Server-Side Communication

As previously mentioned, the server-side portion of this site was written in PHP. All of the data obtained from the server is requested from separate AJAX calls whenever a page is loaded or the user applies a new filter. The standings and sortable stats pages simply request a complete list of JSON data from the team table and player table respectively and then use DataTables to display it. The home page of *Basketball Charts* which contains the chart builder tool is a bit more complicated.

The first factor the chart builder tool takes into consideration is the player pool that the user can define. By default, it is set to all players but the user has the ability to limit it by position among other things. When the user specifies a filter on the player pool, a new AJAX request is made with the filter also being sent to the server. The server code formats this filter into a SQL condition and appends it to the final query that is ultimately executed.

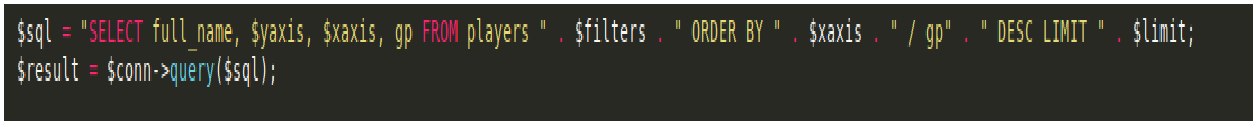


```
switch($pool){
    case "all":
        break;
    case "top":
        $filters = $filters . "WHERE pts > 1500";
    case "rookies":
        $filters = $filters . "WHERE season_exp = 0";
        break;
    case "sophmores":
        $filters = $filters . "WHERE season_exp = 1";
        break;
    case "guard":
        $filters = $filters . "WHERE position IN ('Guard', 'Guard-Forward', 'Forward-Guard')";
        break;
    case "forward":
        $filters = $filters . "WHERE position IN ('Forward', 'Guard-Forward', 'Forward-Guard')";
        break;
    case "center":
        $filters = $filters . "WHERE position IN ('Center', 'Center-Forward', 'Forward-Center')";
        break;
}
```

Figure 9: A switch statement to determine the player pool

Because the chart builder tool allows for three separate chart types to be generated, a variable specifying what chart type is being requested must also be passed to the server. This variable along with variables for the charts value parameters and an optional team filter are all sent to the server. The server file checks the chart type and generates the correct SQL statement based on this information. If a team filter was used, it is appended to the SQL statement much like the player pool filter described earlier.

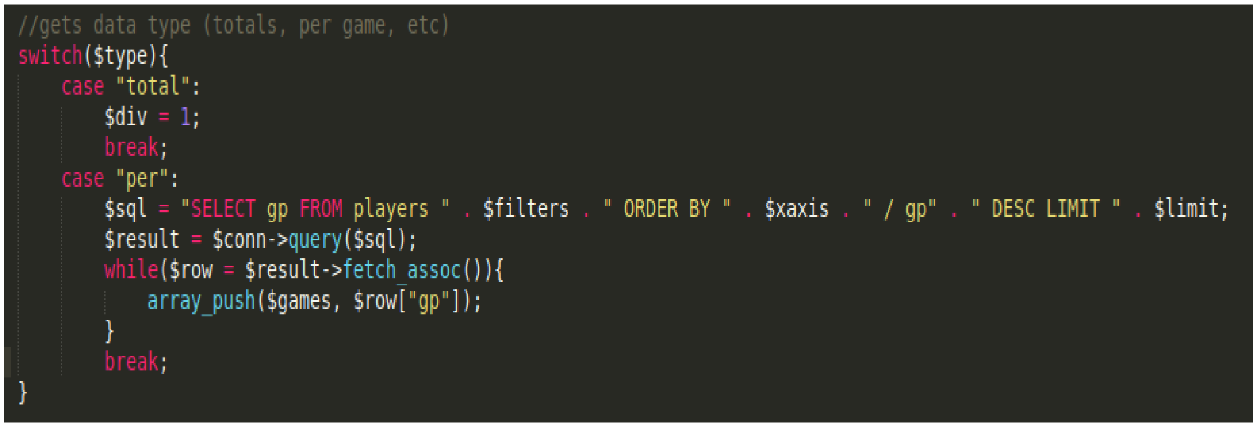
Finally, the user also has the ability to select a *type* for the data. This allows the user to



```
$sql = "SELECT full_name, $yaxis, $xaxis, gp FROM players " . $filters . " ORDER BY " . $xaxis . " / gp" . " DESC LIMIT " . $limit;
$result = $conn->query($sql);
```

Figure 10: An example of a SQL query being generated based on filters applied by a user

request either the raw totals for the specified values or view this data on a per game basis. If the user requests the per game version of the data, a query is run requesting the players' total games played and then these values are used in the final query to convert the data to this format.



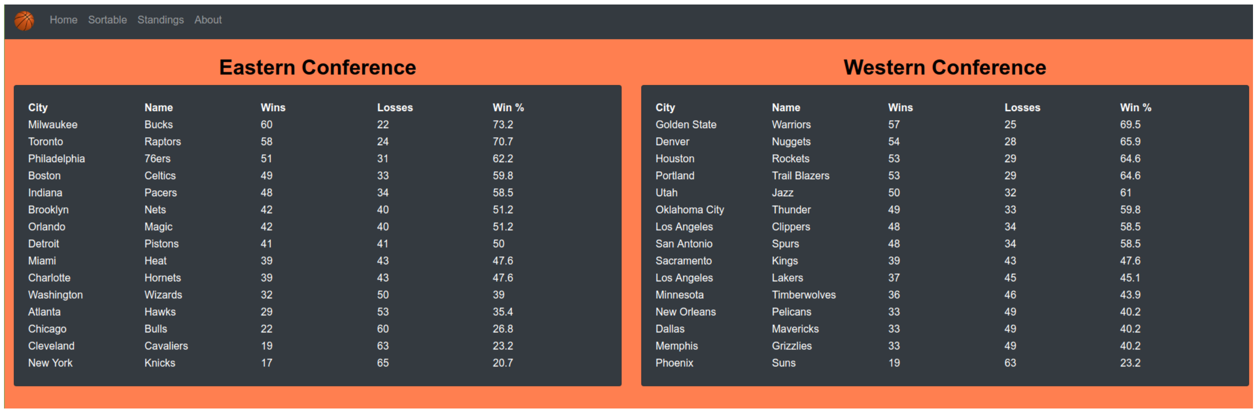
```
//gets data type (totals, per game, etc)
switch($type){
    case "total":
        $div = 1;
        break;
    case "per":
        $sql = "SELECT gp FROM players " . $filters . " ORDER BY " . $xaxis . " / gp" . " DESC LIMIT " . $limit;
        $result = $conn->query($sql);
        while($row = $result->fetch_assoc()){
            array_push($games, $row["gp"]);
        }
        break;
}
```

Figure 11: A switch statement that determines whether or not per game values are used

<u>Front-End Construction</u>

The front-end of *Basketball Charts* was written using HTML, CSS, and JavaScript/JQuery. The CSS framework, Bootstrap, was also used to improve the overall look and fluidity of the site.

Simplicity and responsiveness were the two primary principles that were kept in mind when developing the front end of the site. As can be seen in Figure 12, a simple black and orange color scheme was used throughout *Basketball Charts* with either a white or black font depending on the background color. The Bootstrap card class was also used to hold sections of data and dynamically resizes based on the dimensions of the device being used to access the page.



Eastern Conference

City	Name	Wins	Losses	Win %
Milwaukee	Bucks	60	22	73.2
Toronto	Raptors	58	24	70.7
Philadelphia	76ers	51	31	62.2
Boston	Celtics	49	33	59.8
Indiana	Pacers	48	34	58.5
Brooklyn	Nets	42	40	51.2
Orlando	Magic	42	40	51.2
Detroit	Pistons	41	41	50
Miami	Heat	39	43	47.6
Charlotte	Hornets	39	43	47.6
Washington	Wizards	32	50	39
Atlanta	Hawks	29	53	35.4
Chicago	Bulls	22	60	26.8
Cleveland	Cavaliers	19	63	23.2
New York	Knicks	17	65	20.7

Western Conference

City	Name	Wins	Losses	Win %
Golden State	Warriors	57	25	69.5
Denver	Nuggets	54	28	65.9
Houston	Rockets	53	29	64.6
Portland	Trail Blazers	53	29	64.6
Utah	Jazz	50	32	61
Oklahoma City	Thunder	49	33	59.8
Los Angeles	Clippers	48	34	58.5
San Antonio	Spurs	48	34	58.5
Sacramento	Kings	39	43	47.6
Los Angeles	Lakers	37	45	45.1
Minnesota	Timberwolves	36	46	43.9
New Orleans	Pelicans	33	49	40.2
Dallas	Mavericks	33	49	40.2
Memphis	Grizzlies	33	49	40.2
Phoenix	Suns	19	63	23.2

Figure 12: The contrasting black and orange color scheme as well as the Bootstrap 'Card' elements that were used throughout the site are visible here

Functionality

Chart Builder Tool – Sortable Stats – Team Standings

The primary functionality of the project is available right from the home page of the web application where the chart builder tool is located. As has been described previously, the chart builder tool provides the user with a variety of options for customizing the chart type, parameters, and filters.

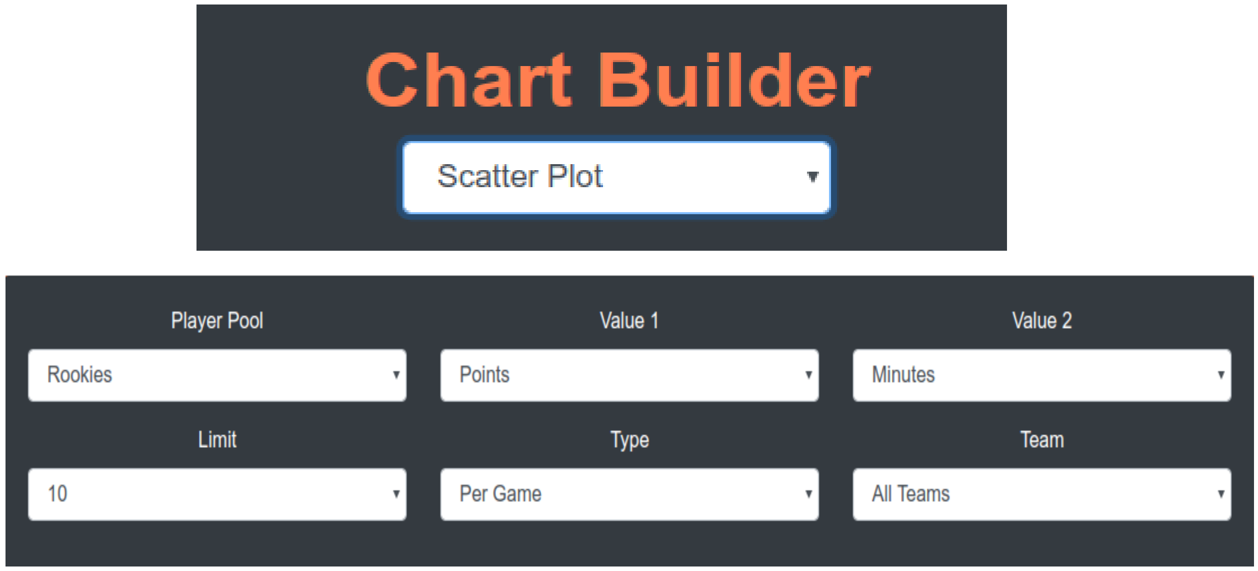


Chart Builder

Scatter Plot

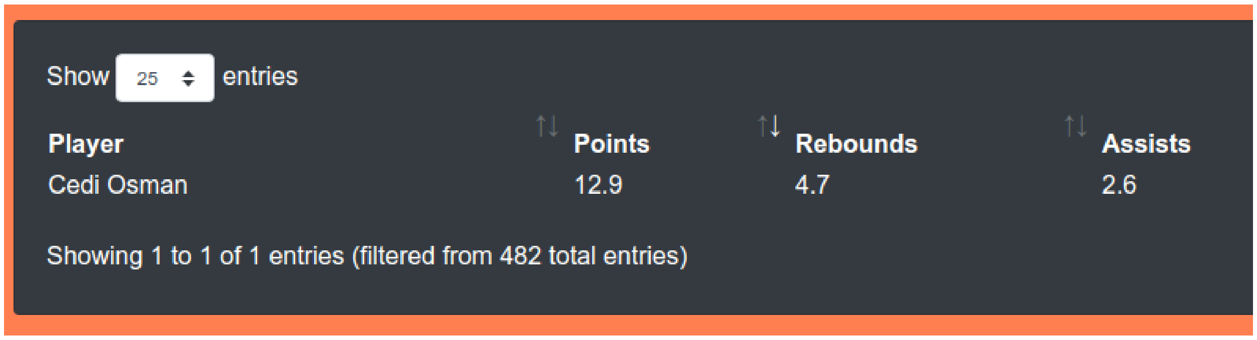
Player Pool	Value 1	Value 2
Rookies	Points	Minutes

Limit	Type	Team
10	Per Game	All Teams

Figure 13: A screenshot of the filtering options available to the user

The tool allows the user to choose between a scatter plot, column chart, or pie chart with the goal being for the user to be able to come with appealing and meaningful visuals that can be saved and shared elsewhere. To streamline this process, HighCharts' export options are enabled allowing users to easily download the images of the charts they create in a variety of common formats. Some examples of the types of charts that can be generated using this tool will be described at the end of this section.

Basketball Charts also includes a standings and sortable stats page that utilizes DataTables to display a static list of the data stored within the MySQL database. The sortable stats page allows for an easy view of the league leaders in a particular category and also allows for the user to search for the stat line of a particular player.



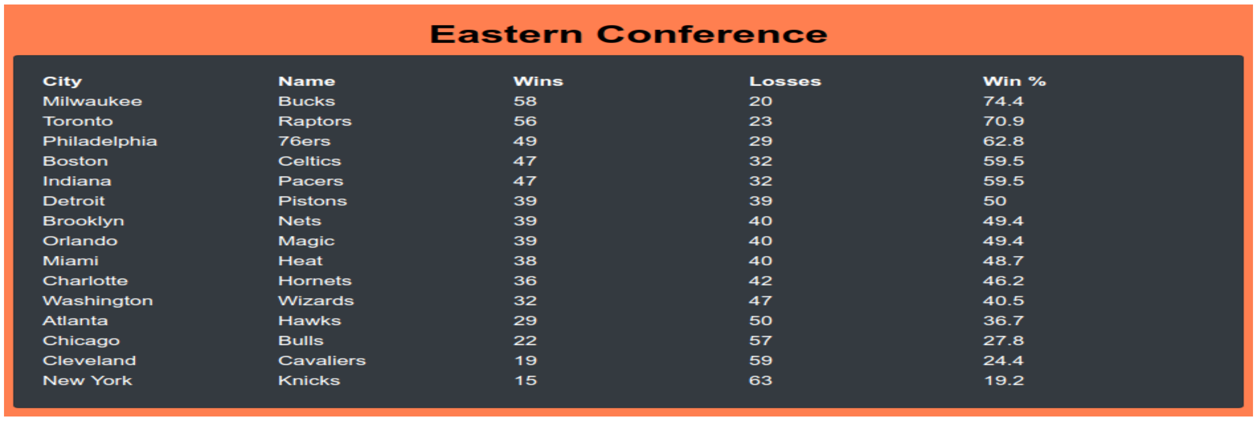
Show 25 entries

Player	Points	Rebounds	Assists
Cedi Osman	12.9	4.7	2.6

Showing 1 to 1 of 1 entries (filtered from 482 total entries)

Figure 14: A demonstration of the sortable stats search functionality

The standings page shows the league standings separated by conference and is sorted by overall win percentage. Both of these pages are included to give the site a more complete feel and allow users to access the data in an alternate manner in comparison to the way data is presented through the chart builder tool.

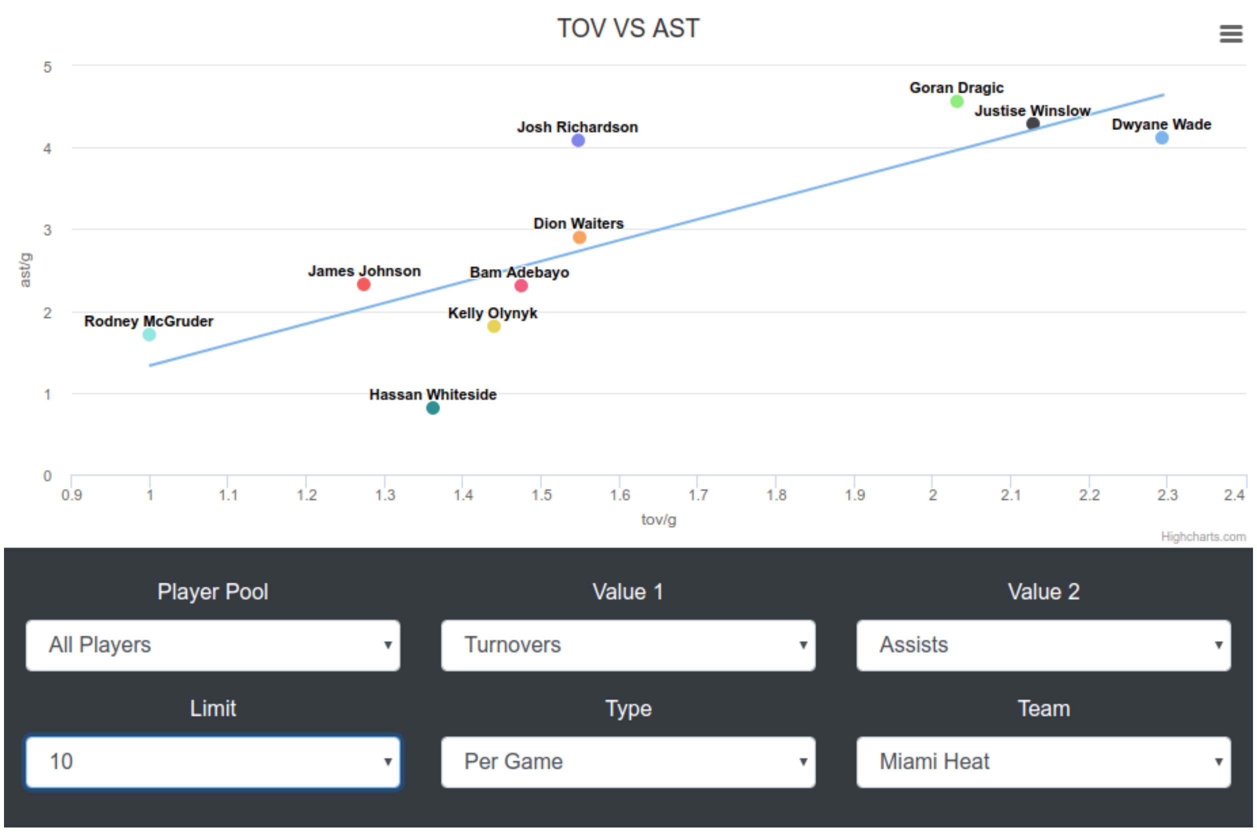


Eastern Conference

City	Name	Wins	Losses	Win %
Milwaukee	Bucks	58	20	74.4
Toronto	Raptors	56	23	70.9
Philadelphia	76ers	49	29	62.8
Boston	Celtics	47	32	59.5
Indiana	Pacers	47	32	59.5
Detroit	Pistons	39	39	50
Brooklyn	Nets	39	40	49.4
Orlando	Magic	39	40	49.4
Miami	Heat	38	40	48.7
Charlotte	Hornets	36	42	46.2
Washington	Wizards	32	47	40.5
Atlanta	Hawks	29	50	36.7
Chicago	Bulls	22	57	27.8
Cleveland	Cavaliers	19	59	24.4
New York	Knicks	15	63	19.2

Figure 15: A screenshot of the Eastern Conference standings

Chart Builder Tool Examples



Player Pool: All Players
Value 1: Turnovers
Value 2: Assists
Limit: 10
Type: Per Game
Team: Miami Heat

Figure 16: A scatter plot comparing the ast/to ratio among members of the Miami Heat

Figure 16 shows a scatter plot looking at the assist to turnover ratio of players for the

Miami Heat. As you can see, a line of best fit was calculated using the least squared method. Looking at this chart, users are able to draw conclusions about the play-making ability of Heat players by searching for players who manage to get a lot of assists which remaining about the best fit line. Goran Dragic and Josh Richardson have very positive assist to turnover ratios while someone like Hassan Whiteside is not a very strong passer.

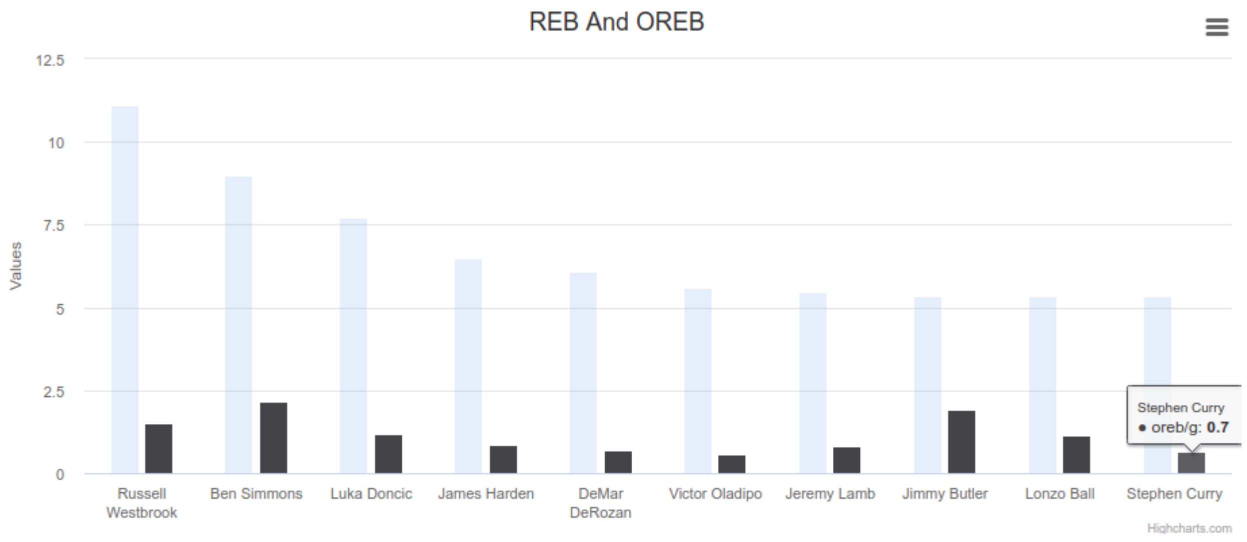


Figure 17: A column chart that shows the total rebounds in comparison to offensive rebounds among NBA guards

Figure 17 compares the total rebounds averaged per game among guards to the number of offensive rebounds they grab each game. Through this chart you can see that although Russell Westbrook is the best overall rebounder among guards, Ben Simmons and Jimmy Butler do a better job of creating extra possessions on offense.

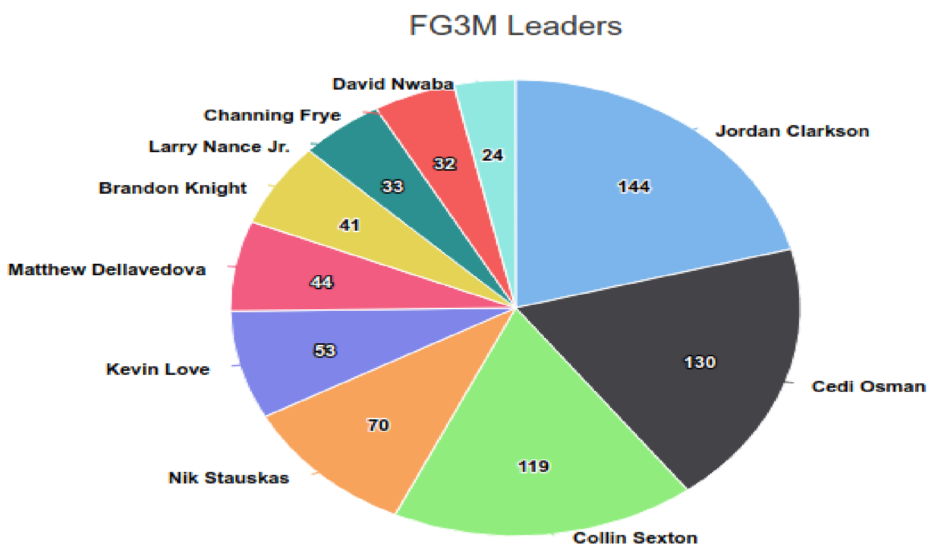


Figure 18: A pie chart showing the leaders in 3 point field goals made among Cavalier players

The pie chart option only uses one value and provides a nice breakdown of who gets the majority of stats in a particular category. In the case of Figure 18, the leading three point shooters in terms of field goals made on the Cleveland Cavaliers is shown.

<u>**Future Work**</u>

The primary goal of *Basketball Charts* was to develop the ability for users to create useful visual representations of preexisting data. In that respect, this project was a success. However, there are many other features that could potentially be added in the future to create an even more complete source of professional basketball statistical information.

The addition of player and team profiles is one feature in particular that will likely be the next goal in the development process. The idea will be to allow users to select a team or player name as a hyperlink wherever they appear on the site and then display more detailed information relating to that team or player through a popover. The player profiles would simply show basic bio information that is already stored in the MySQL database as well as displaying basic box score data to go with it. The team profiles would display basic team info like location and record as well as a complete roster list with links to player profiles.

Another feature that is not present in the *Basketball Charts'* current state is information relating to the NBA draft. The draft is a very important part of the NBA's off-season so adding the ability to view data on the current year's biggest prospects would greatly enhance the overall usefulness of the website. Prospect data in addition to an updated list of draft lottery odds for each team are both items that will hopefully be added in the future.

Finally, expanding the database to include historical data is one more major feature that could really increase the overall site quality. At the moment, data from the current year is all that is available but including data from previous years and allowing users to filter through that data should be something that is very achievable going forward.

Conclusion

When this project first started, the plan was to create a place where users could take advantage of all of the interesting statistical data available on the web and construct compelling visuals out of them to be easily saved and shared with others. The chart builder tool developed as a part of this project achieved this goal and therefore *Basketball Charts* should be considered a success. Working on this project was also beneficial in terms of gaining more experience with charting tools such as *HighCharts* as well as gaining valuable experience working with a community driven API in *NBA Stats*. As mentioned in the previous section, there is also still plenty of room for further development of both the chart builder tool and other aspects of the site. The plan is to continue to expand the web application over time while continuing to support the functionality that already exists. Once the site reaches a point where the majority of features described in the previous section have been developed, the next step will be to share it with others in order to begin the process of establishing a user base. In its current state, the site can already be accessed at www.bball-charts.com.

References

DataTables. (n.d.). Retrieved from https://datatables.net/

Highcharts. (n.d.). Retrieved from https://www.highcharts.com/

NBA Stats. (n.d.). Retrieved from https://stats.nba.com/