# Final Project Documentation

Qingyuan Liu

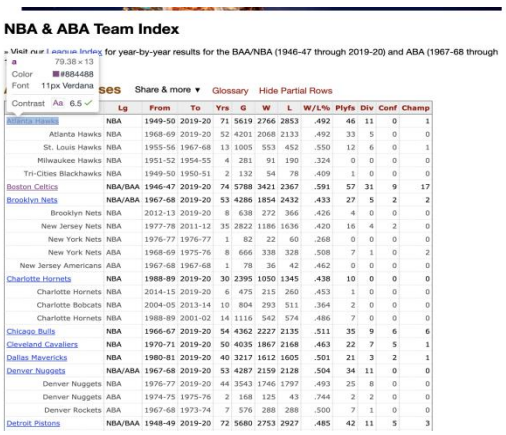
***Challenge Score：** 8 points (*Crawling [and scraping] multiple pages in a site you haven't used before ❀)

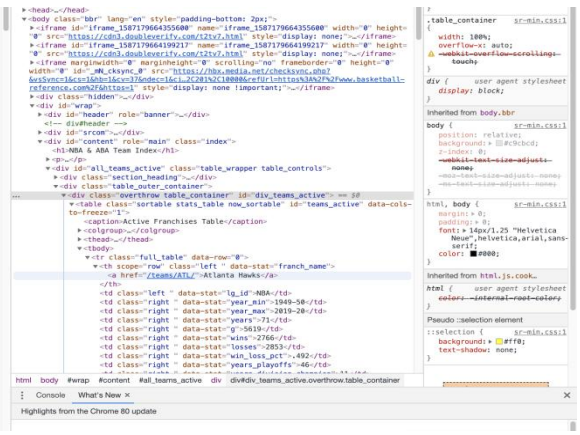
***My Final Project Demo link on youtube:**https://youtu.be/a8eYoYPt05U

1. **My github link:** https://github.com/Qingyuan-Liu/507Final_project

2. **About how to run my code:**
- First, you could enter the command python3 app.py, get the url and paste it to the browser. Since, it will be a little bit slow to load the huge amount of data to the database, so I use the database that I loaded before which is this nba.sqlite. And I will use it for the demo. If you want to check the function to load the data to the database, you could uncomment these 398 lines in the app.py file for testing. (I have already uncomment this line for the version I uploaded to the github, since I need to put the .sqlite file and .cache file to the .gitignore. So you may need to load it again, and it might take you some time to load the page.)
- I put the website's content which I got by using BeautifulSoup with their uniquekey to the cache.json file which is in the .gitignore file, you could run the program then it will show up. So, if I use cache, it will show 'Using cache', and if I delete the cache file and it will show 'Fetching'. I demoed that in another video (You could see it later in this document).

3. **Data Source and Database:**
- First, I get all the ***30 active franchises***' links from the <a> tag in this page: https://www.basketball-reference.com/teams/.
  And also, I get all the **teams'** information like team name, league, games, etc. in this page and load them to my database nba.sqlite. (Here is my first table: **Teams**)

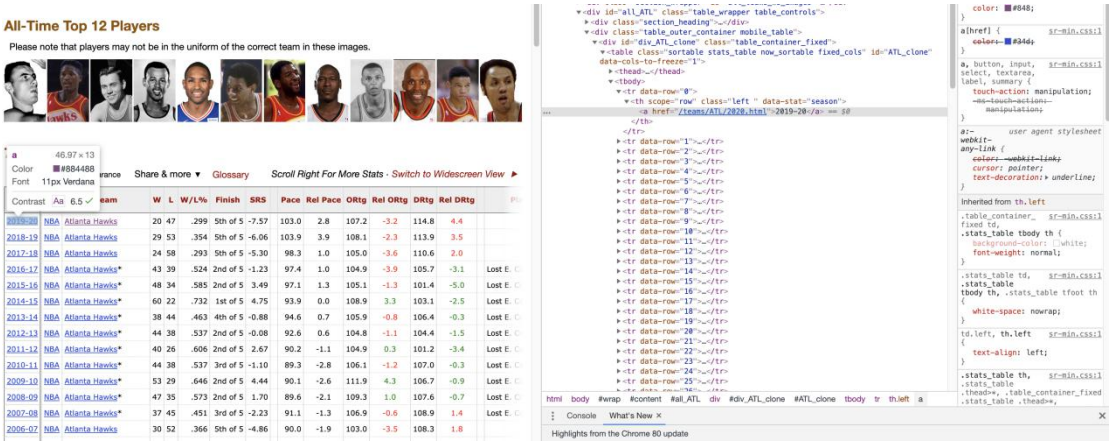


```
create_teams='''
    CREATE TABLE IF NOT EXISTS "Teams" (
        "Id" INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
        "Team_name" TEXT NOT NULL,
        "Intro_url" TEXT,
        "League" TEXT NOT NULL,
        "Since"  TEXT NOT NULL,
        "Until" TEXT NOT NULL,
        "Year" INTEGER NOT NULL,
        "Games" INTEGER NOT NULL,
        "Wins" INTEGER NOT NULL,
        "Losts" INTEGER NOT NULL,
        "Win_lose_percentage" REAL NOT NULL,
        "Playoffs" INTEGER NOT NULL,
        "Divisions" INTEGER NOT NULL,
        "Conference_champ" INTEGER NOT NULL,
        "Championship" INTEGER NOT NULL
    );
'''
```

| Name | Type | Schema |
|---|---|---|
| Teams | | CREATE TABLE "Teams" ( "Id" INTE |
| Id | INTEGER | "Id" INTEGER PRIMARY KEY AUTOI |
| Team_name | TEXT | "Team_name" TEXT NOT NULL |
| Intro_url | TEXT | "Intro_url" TEXT |
| Leage | TEXT | "Leage" TEXT NOT NULL |
| Since | TEXT | "Since" TEXT NOT NULL |
| Until | TEXT | "Until" TEXT NOT NULL |
| Year | INTEGER | "Year" INTEGER NOT NULL |
| Games | INTEGER | "Games" INTEGER NOT NULL |
| Wins | INTEGER | "Wins" INTEGER NOT NULL |
| Losts | INTEGER | "Losts" INTEGER NOT NULL |
| Win_lose_p... | REAL | "Win_lose_percentage" REAL NOT |
| Playoffs | INTEGER | "Playoffs" INTEGER NOT NULL |
| Divisions | INTEGER | "Divisions" INTEGER NOT NULL |
| Conference... | INTEGER | "Conference_champ" INTEGER NO |
| Champions... | INTEGER | "Championship" INTEGER NOT NU |

- Second, I followed the link to the teams' detail page (eg: https://www.basketball-reference.com/teams/ATL/) and it include the teams' information in all seasons. So, I got the latest season's link of this team through the <a> tag, and scrape that following page.
-



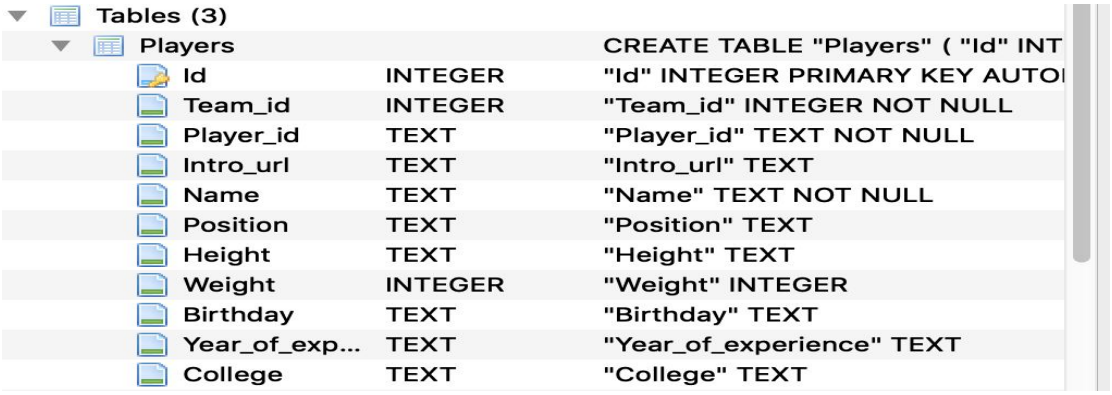

- Then, through the link that I got from the last step, and now I'm in the web page (eg: https://www.basketball-reference.com/teams/ATL/2020.html) which include all the players in that team during the latest season. And I got the players' information like player number, name, position, height, etc. and stored them into

my database (Here is my second table: Players). ***There will be 472 records in the Players table.***

Roster (TW) - Signed to two-way contract w/ G-League affiliate Share & more ▾ Glossary

| No. | Player | Pos | Ht | Wt | Birth Date | | Exp | College |
|---|---|---|---|---|---|---|---|---|
| 12 | De'Andre Hunter | SF | 6-7 | 225 | December 2, 1997 | | R | Virginia |
| 11 | Trae Young | PG | 6-1 | 180 | September 19, 1998 | | 1 | Oklahoma |
| 15 | Vince Carter | SF | 6-6 | 220 | January 26, 1977 | | 21 | UNC |
| 22 | Cam Reddish | SF | 6-8 | 208 | September 1, 1999 | | R | Duke |
| 3 | Kevin Huerter | SG | 6-7 | 190 | August 27, 1998 | | 1 | Maryland |
| 24 | Bruno Fernando | C | 6-9 | 233 | August 15, 1998 | | R | Maryland |
| 30 | Damian Jones | C | 6-11 | 245 | June 30, 1995 | | 3 | Vanderbilt |
| 95 | DeAndre' Bembry | SG | 6-5 | 210 | July 4, 1994 | | 3 | Saint Joseph's |
| 20 | John Collins | PF | 6-9 | 235 | September 23, 1997 | | 2 | Wake Forest |
| 0 | Brandon Goodwin | PG | 6-0 | 180 | October 2, 1995 | | 1 | Central Florida, Florida Gulf Coast |
| 00 | Jeff Teague | PG | 6-3 | 195 | June 10, 1988 | | 10 | Wake Forest |
| 2 | Treveon Graham | SG | 6-5 | 223 | October 28, 1993 | | 3 | Virginia Commonwealth |
| 14 | Dewayne Dedmon | C | 7-0 | 245 | August 12, 1989 | | 6 | USC |
| 4 | Charlie Brown (TW) | SG | 6-6 | 199 | February 2, 1997 | | R | Saint Joseph's |
| | Clint Capela | C | 6-10 | 240 | May 18, 1994 | | 5 | |
| | Skal Labissière | F | 6-10 | 216 | March 18, 1996 | | 3 | Kentucky |

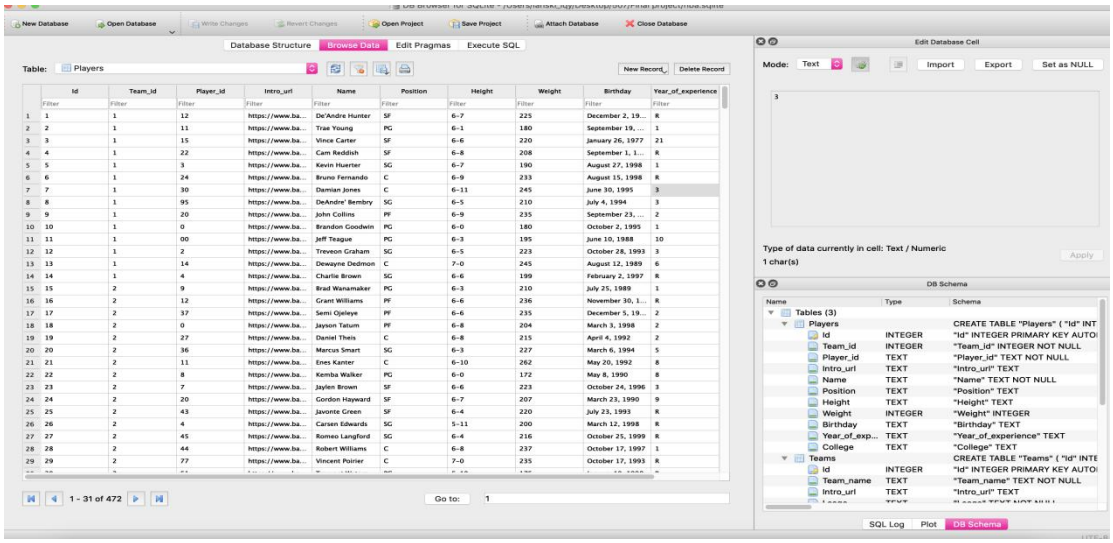
```
create_players = '''
    CREATE TABLE IF NOT EXISTS "Players" (
        "Id"         INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
        "Team_id" INTEGER NOT NULL,
        "Player_id"  TEXT NOT NULL,
        "Intro_url" TEXT,
        "Name"       TEXT NOT NULL,
        "Position" TEXT,
        "Height" TEXT,
        "Weight" INTEGER,
        "Birthday" TEXT,
        "Year_of_experience" TEXT,
        "College" TEXT,
        FOREIGN KEY(Team_id) REFERENCES Team(Id)
    );
'''
```

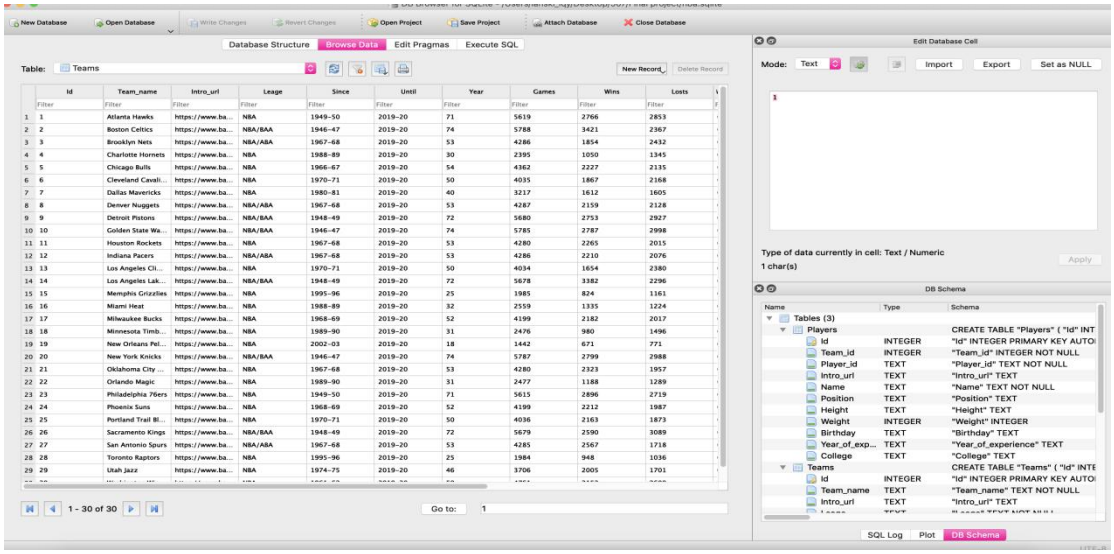


Tables (3)

| | | |
|---|---|---|
| Players | | CREATE TABLE "Players" ( "Id" INT |
| Id | INTEGER | "Id" INTEGER PRIMARY KEY AUTOI |
| Team_id | INTEGER | "Team_id" INTEGER NOT NULL |
| Player_id | TEXT | "Player_id" TEXT NOT NULL |
| Intro_url | TEXT | "Intro_url" TEXT |
| Name | TEXT | "Name" TEXT NOT NULL |
| Position | TEXT | "Position" TEXT |
| Height | TEXT | "Height" TEXT |
| Weight | INTEGER | "Weight" INTEGER |
| Birthday | TEXT | "Birthday" TEXT |
| Year_of_exp... | TEXT | "Year_of_experience" TEXT |
| College | TEXT | "College" TEXT |

- About the two tables in my database: The Team_id in my Players table is a ***foreign key*** which reference to the Id in my Teams table. We could query like:
  'SELECT avg(Players.Height)
  FROM Players,Teams
  WHERE Teams.Id=Players.Team_id
  GROUP by Players.Team_id'
  Here are the screen shots of my two tables: Players and Teams





- About caching: I stored the URL and the html content into the cache.json file and when I use cache, it will print ‘Using cache’ and when the unique_key is not in cache, I will fetch it and store it in cache and then it will print ‘Fetching’.

cache.json


```
{"https://www.basketball-reference.com/teams/_": "\n<!DOCTYPE html>\n<html data-
version=\"klecko-\" data-root=\"/home/bbr/build\" itemscope itemtype=\"https://schema.org/
WebSite\" lang=\"en\" class=\"no-js\" >\n<head>\n    <meta charset=\"utf-8\">\n    <meta
http-equiv=\"x-ua-compatible\" content=\"ie=edge\">\n    <meta name=\"viewport\"
content=\"width=device-width, initial-scale=1.0, maximum-scale=2.0\" />\n    <link
rel=\"dns-prefetch\" href=\"https://d2p3bygnnzw9w3.cloudfront.net/req/202004011\" /
>\n\n<!-- no:cookie fast load the css.              -->\n<script>function gup(n) {n =
n.replace(/[\\[]/, '\\\\[').replace(/[\\]]/, '\\\\]'); var r = new RegExp('[\\\\?
&]'+n+'=([^&#]*)'); var re = r.exec(location.search);    return re ===
null?'':decodeURIComponent(re[1].replace(/\\+/g,' '));}; document.srdev = gup('srdev')</
script>\n<link rel=\"preconnect\" href=\"https://d2p3bygnnzw9w3.cloudfront.net\"
crossorigin>\n<link rel=\"preconnect\" href=\"https://d2cwpp38twqe55.cloudfront.net\"
crossorigin>\n <style>html,body{margin:0;padding:0;font:14px/1.25 \"Helvetica
Neue\",helvetica,arial,sans-serif;color:#000}body{position:relative;background:#c9cbcd;z-
index:0;-webkit-text-size-adjust:none;-moz-text-size-adjust:none;-ms-text-size-
adjust:none}a img{border:0}ul,li,ol{margin:0;padding:0;list-style-type:none}table th,table
td{border:0}iframe{max-width:100%}code{background:#eee}
a,button,input,select,textarea,label,summary{touch-action:manipulation;-ms-touch-
action:manipulation}::selection{background:#ff0;text-shadow:none}html.no-
js .hasmore,html.no-js .js,html.js .no-js,html.no-js
button,html.is_build .hide_build,html.is_dev .hide_dev,html.is_live .hide_live{display:non
e!important}.js-select{visibility:hidden}a[href]{color:#34d}a[href]:visited{color:#848}
a[href]:hover{color:#5e44b3}a[href]:active{color:#b12}a[href^=\"tel:\"]::before{content:
\"\\260E\";display:inline-block;margin-right:0}a[href^=\"mailto:\"]::before{content:\"\
\2709\";display:inline-block;margin-right:0}.icon_group a[href^=\"tel:
\"]:before,.icon_group a[href^=\"mailto:\"]:before{content:
\"\"}.screen_only{display:initial}.user_logged_in .not_logged_in,.logged_in,.sr_cpi_contro
l,.print_only{display:none}#perflog,#modernizr,.user_logged_in .logged_in{display:block}
h2{color:#000;margin-bottom:5px;font-size:1.5em}img.right{float:right;margin:0 0 5px 5px}
img.left{float:left;margin:0 15px 15px 0}sup{font-size:.
```


- Here is the first picture is when I delete the cache.json file and run the program.

```
507Final_project — Python • Python app.py — 80×24
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 280-056-610
 * Detected change in '/Users/lanski_lqy/Desktop/507/507Final_project/app.py', r
eloading
 * Restarting with stat
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
```

- The second picture is when the second time I run the program after having the cache.json file.

```
[^CliuqingyuandeMacBook-Pro:507Final_project lanski_lqy$ python3 app.py
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
```

***Here is the link of the video to show the cache operation:***
***https://www.youtube.com/watch?v=6mV0yTUyVUI&feature=youtu.be***

***Here is my code to show that I use cache:***

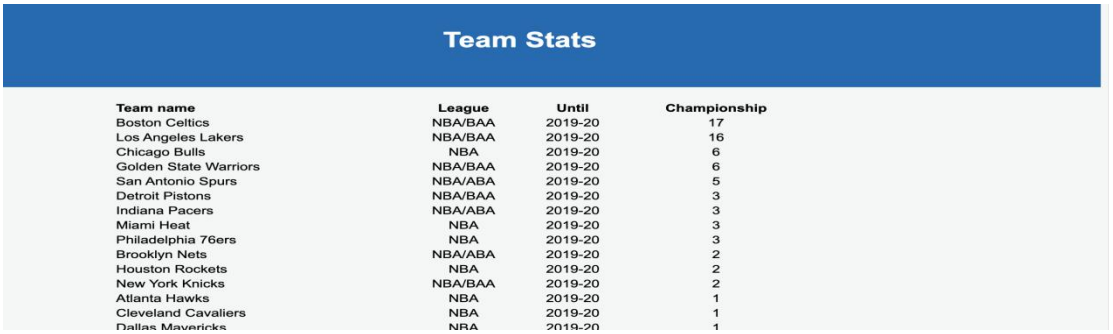
app.py

```python
DBNAME = 'nba.sqlite'
CACHE_FILENAME = "cache.json"


def open_cache():
    try:
        cache_file = open(CACHE_FILENAME, 'r')
        cache_contents = cache_file.read()
        cache_dict = json.loads(cache_contents)
        cache_file.close()
    except:
        cache_dict = {}
    return cache_dict

def save_cache(cache_dict):
    dumped_json_cache = json.dumps(cache_dict)
    fw = open(CACHE_FILENAME,"w")
    fw.write(dumped_json_cache)
    fw.close()


def construct_unique_key(baseurl, params):
    param_strings = []
    connector = '_'
    for k in params.keys():
        param_strings.append(f'{k}_{params[k]}')
    unique_key = baseurl + connector + connector.join(param_strings)
    return unique_key


def make_url_request_using_cache(url, params, c_type="text"):
    unique_key = construct_unique_key(url, params)
    cache = open_cache()
    if (unique_key in cache.keys()):
        print("Using cache")
    else:
        print("Fetching")
        response = requests.get(url, params=params)
        if c_type == "json":
            cache[unique_key] = response.json()
        else:
            cache[unique_key] = response.text
        save_cache(cache)

    return cache[unique_key]


def player_stats(year):
    url = "https://www.basketball-reference.com/leagues/NBA_{}_per_game.html".format(year)
    response = make_url_request_using_cache(url, {})
    soup=BeautifulSoup(response,'html.parser')
```

## 4. Interaction and Presentation plan:

I use Plotly and Flask to built this website, in the home page like I attached above. The user could choose the sort way, sort direction and what they want to see in the first part.

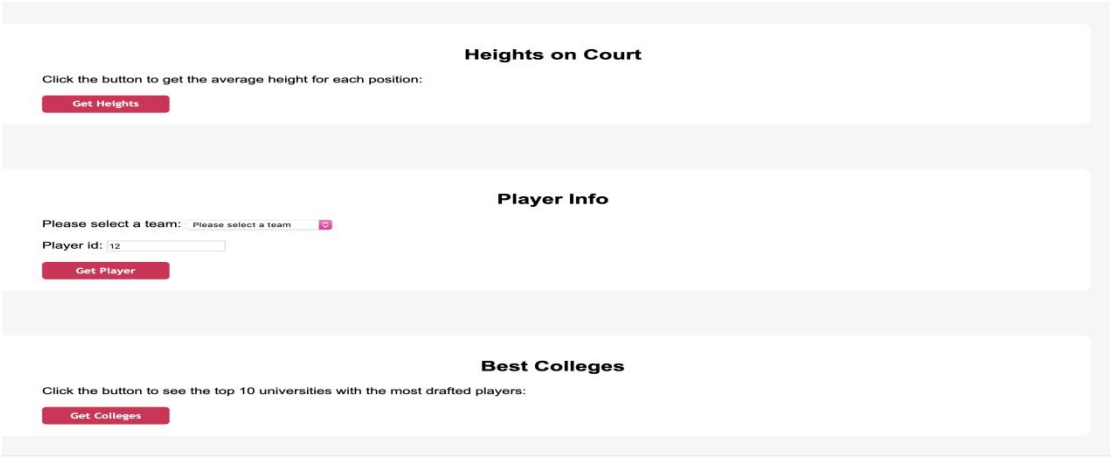
And the user could get the information like the second picture(I only showed the part of the results).

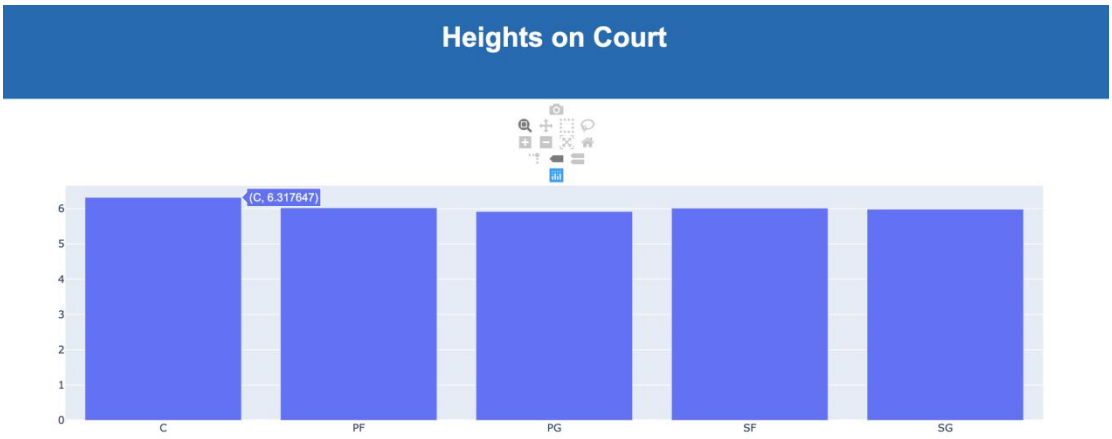



**Team Stats**

| Team name | League | Until | Championship |
|---|---|---|---|
| Boston Celtics | NBA/BAA | 2019-20 | 17 |
| Los Angeles Lakers | NBA/BAA | 2019-20 | 16 |
| Chicago Bulls | NBA | 2019-20 | 6 |
| Golden State Warriors | NBA/BAA | 2019-20 | 6 |
| San Antonio Spurs | NBA/ABA | 2019-20 | 5 |
| Detroit Pistons | NBA/BAA | 2019-20 | 3 |
| Indiana Pacers | NBA/ABA | 2019-20 | 3 |
| Miami Heat | NBA | 2019-20 | 3 |
| Philadelphia 76ers | NBA | 2019-20 | 3 |
| Brooklyn Nets | NBA/ABA | 2019-20 | 2 |
| Houston Rockets | NBA | 2019-20 | 2 |
| New York Knicks | NBA/BAA | 2019-20 | 2 |
| Atlanta Hawks | NBA | 2019-20 | 1 |
| Cleveland Cavaliers | NBA | 2019-20 | 1 |
| Dallas Mavericks | NBA | 2019-20 | 1 |

Then, they could click the ‘Get Heights’ button to get the average heights for the five positions in each team, and this information would be shown in Plotly.






There will be 6 different webpage to show the different results, users could access that by clicking the button and enter the required input. If there is no query result that the user want to see, I will show them the error message page (err_msg.html) and let them go back to the home page.