UNIVERSITY OF TEXAS AT AUSTIN

CS 327E Elements of Databases

Project Final Report

Young Min Cho, Simon Yu, Kyoung Won Oe

1. MOTIVATION

The overall goal of this project is to use the data in soccer, the most popular sport in the world, to address informations and statistics. Out of many leagues in the world, we chose English Premier League Soccer because it is the world's most watched league with a TV audience of 4.7 billion people. Similar to sabermetrics in baseball, it has been growing over years. (Every EPL teams are equipped with 8-10 digital cameras to track every player on the field. Where Prozone generate 1-4 million data points per game and they are commercial use only).

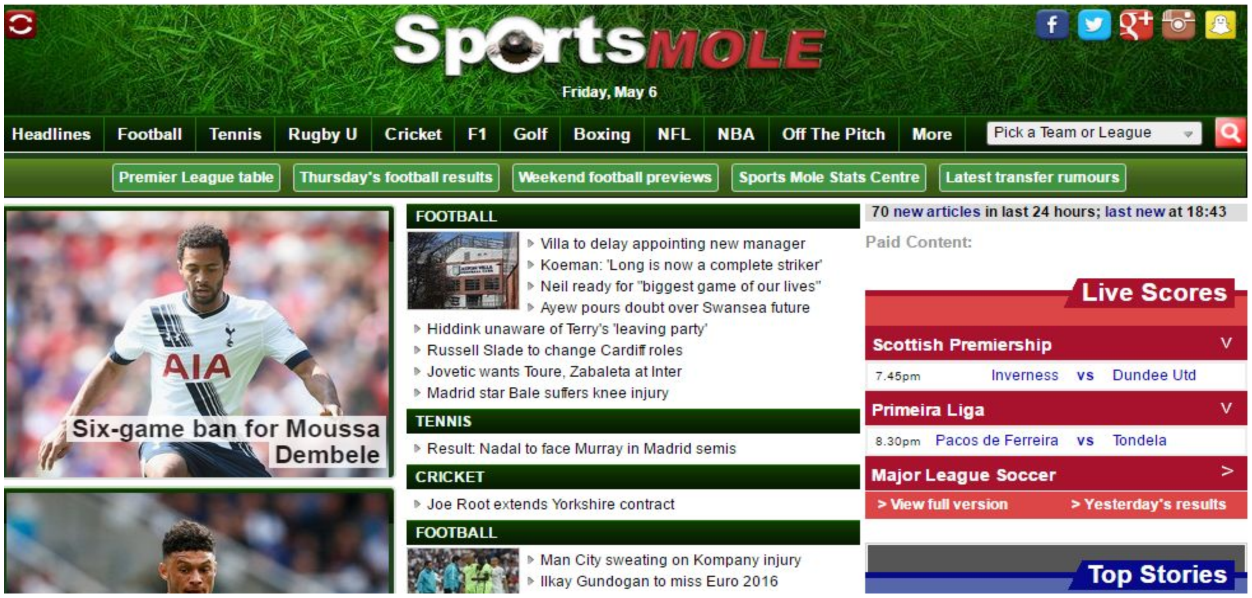
Currently, there is no well-known place that consolidates and displays the current status of metrics and key performance indicators for English Premier League soccer. By choosing which metrics to track, we wanted to identify the core key values in soccer that are actually affecting the game. Also, we wanted to find metrics that has relations in which influences other metrics. (ex. Whether soccer referees are affecting the tackle rates of Home or Away team?)

Since this is automatically brought into our database real-time that should help proactively find stats of players and teams. Which may help others make statistical performance analysis or make insights from a volume of data collected in the future.

2. Websites and APIs

Websites Used:

1. http://www.sportsmole.co.uk/


SportsMOLE

Friday, May 6

Headlines | Football | Tennis | Rugby U | Cricket | F1 | Golf | Boxing | NFL | NBA | Off The Pitch | More | Pick a Team or League

Premier League table | Thursday's football results | Weekend football previews | Sports Mole Stats Centre | Latest transfer rumours

Six-game ban for Moussa Dembele

FOOTBALL
- Villa to delay appointing new manager
- Koeman: 'Long is now a complete striker'
- Neil ready for "biggest game of our lives"
- Ayew pours doubt over Swansea future
- Hiddink unaware of Terry's 'leaving party'
- Russell Slade to change Cardiff roles
- Jovetic wants Toure, Zabaleta at Inter
- Madrid star Bale suffers knee injury

TENNIS
- Result: Nadal to face Murray in Madrid semis

CRICKET
- Joe Root extends Yorkshire contract

FOOTBALL
- Man City sweating on Kompany injury
- Ilkay Gundogan to miss Euro 2016

70 new articles in last 24 hours; last new at 18:43

Paid Content:

Live Scores

Scottish Premiership

7.45pm Inverness vs Dundee Utd

Primeira Liga

8.30pm Pacos de Ferreira vs Tondela

Major League Soccer

> View full version > Yesterday's results

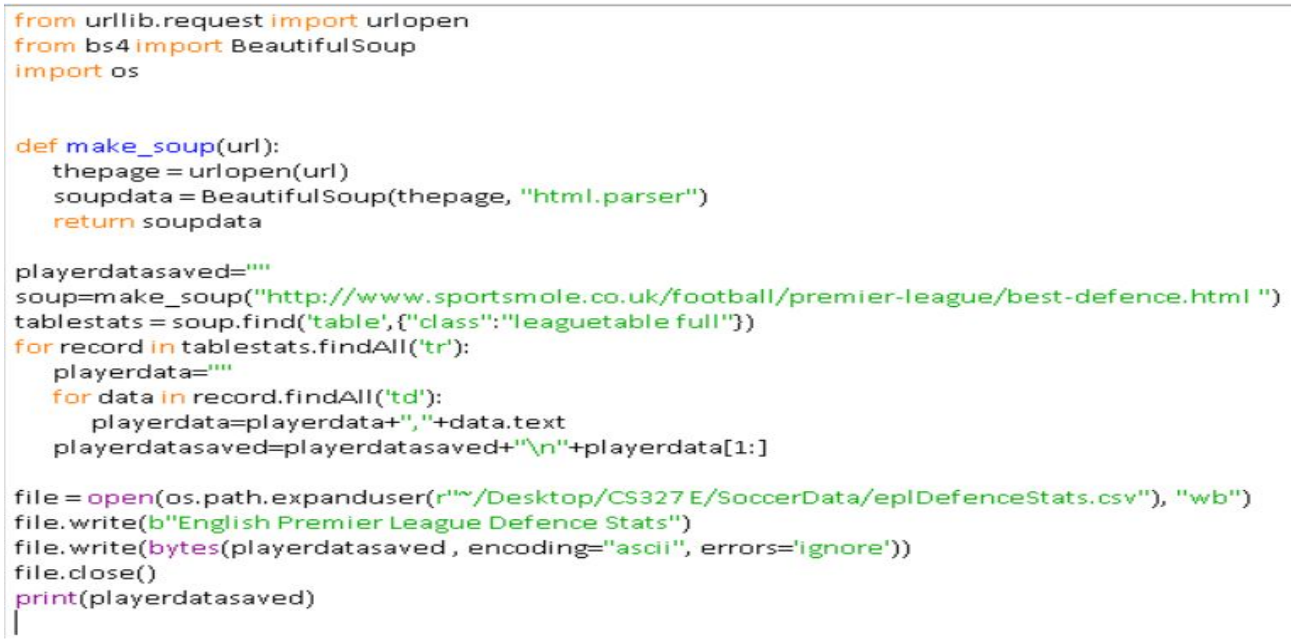
Top Stories

We chose sportsmole.com and soccerstats.com because their website was simply built which enabled us to read and scrape with no problem.

The python code below scrapes the best-defence team in the website sportsmole then puts into CSV file in which we imported os function. Since html tags are different for every and each websites, we first read through everyone of them.

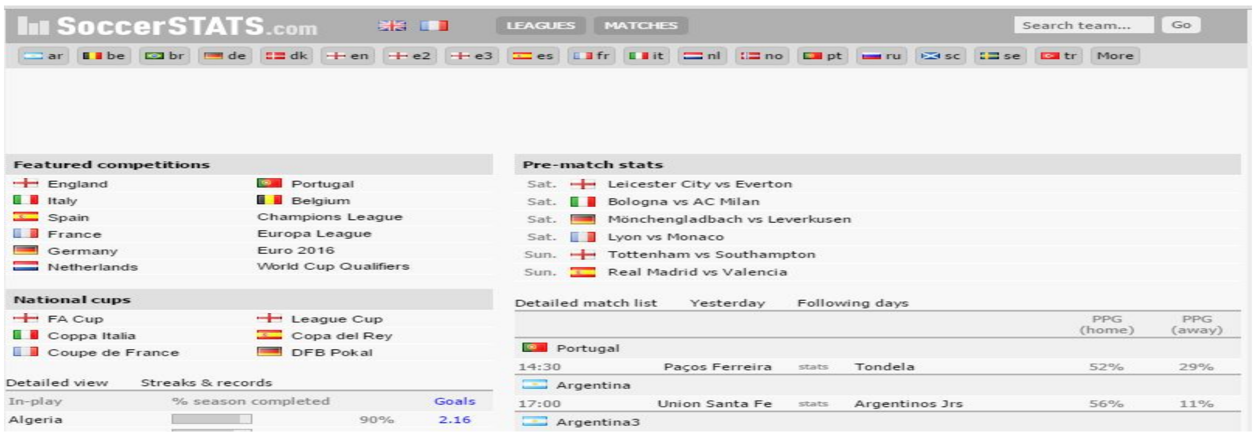
First we made a function make_soup to easily open up url as beautifulsoup. Then we brought up the table we wanted by using .find function in beautifulsoup. Then we started looping through the rows by calling in the <tr> data. Since each rows contain columns with data we called in <td> and looped through them all including commas between to later transfer into the CSV.

This website is one of the website that we can actually scrape off from because, many other website had some sort of visualization effects or in-scrapable locks which gave us blank even when we called up basic simple beautifulsoup html.



```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import os


def make_soup(url):
    thepage = urlopen(url)
    soupdata = BeautifulSoup(thepage, "html.parser")
    return soupdata

playerdatasaved=""
soup=make_soup("http://www.sportsmole.co.uk/football/premier-league/best-defence.html ")
tablestats = soup.find('table',{"class":"leaguetable full"})
for record in tablestats.findAll('tr'):
    playerdata=""
    for data in record.findAll('td'):
        playerdata=playerdata+","+data.text
    playerdatasaved=playerdatasaved+"\n"+playerdata[1:]

file = open(os.path.expanduser(r"~/Desktop/CS327 E/SoccerData/eplDefenceStats.csv"), "wb")
file.write(b"English Premier League Defence Stats")
file.write(bytes(playerdatasaved , encoding="ascii", errors='ignore'))
file.close()
print(playerdatasaved)
```

2. http://www.soccerstats.com/



SoccerSTATS.com

LEAGUES MATCHES Search team... Go

ar be br de dk en e2 e3 es fr it nl no pt ru sc se tr More

Featured competitions

England	Portugal
Italy	Belgium
Spain	Champions League
France	Europa League
Germany	Euro 2016
Netherlands	World Cup Qualifiers

National cups

FA Cup	League Cup
Coppa Italia	Copa del Rey
Coupe de France	DFB Pokal

Detailed view   Streaks & records

In-play	% season completed		Goals
Algeria		90%	2.16

Pre-match stats

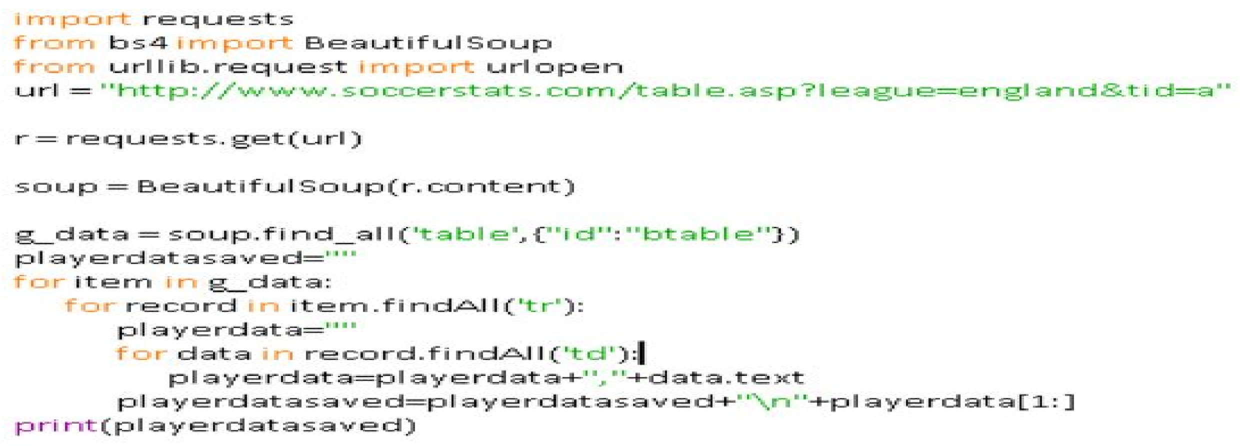
Sat.	Leicester City vs Everton
Sat.	Bologna vs AC Milan
Sat.	Mönchengladbach vs Leverkusen
Sat.	Lyon vs Monaco
Sun.	Tottenham vs Southampton
Sun.	Real Madrid vs Valencia

Detailed match list   Yesterday   Following days

				PPG (home)	PPG (away)
Portugal					
14:30	Paços Ferreira	stats	Tondela	52%	29%
Argentina					
17:00	Union Santa Fe	stats	Argentinos Jrs	56%	11%
Argentina3					

This is similar logic as above, but instead of putting it into the CSV we just added after each data by doing [1:]

This website and one above had exactly same trouble. Because the table names are named exactly the same, as an example “btable”. There was no way for us to differentiate any of the tables. Since one page included so much data, it either brought unnecessary data or just the first table on a certain page.



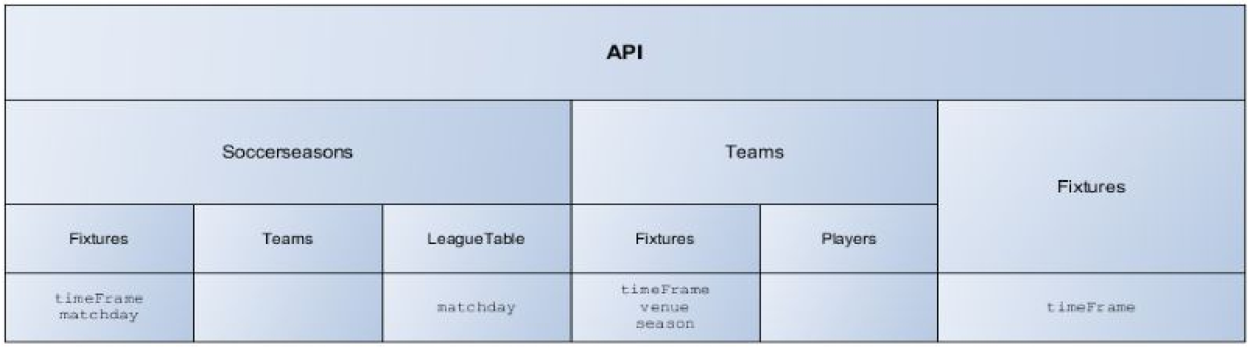
```
import requests
from bs4 import BeautifulSoup
from urllib.request import urlopen
url = "http://www.soccerstats.com/table.asp?league=england&tid=a"

r = requests.get(url)

soup = BeautifulSoup(r.content)

g_data = soup.find_all('table',{"id":"btable"})
playerdatasaved=""
for item in g_data:
    for record in item.findAll('tr'):
        playerdata=""
        for data in record.findAll('td'):
            playerdata=playerdata+","+data.text
        playerdatasaved=playerdatasaved+"\n"+playerdata[1:]
print(playerdatasaved)
```

3. api.football-data.org

Scraping data was difficult so we decided to use APIs.

Problem: There aren’t so many places where they provide free APIs for the English Premier League. However, we happened to find one called api.football-data.org.

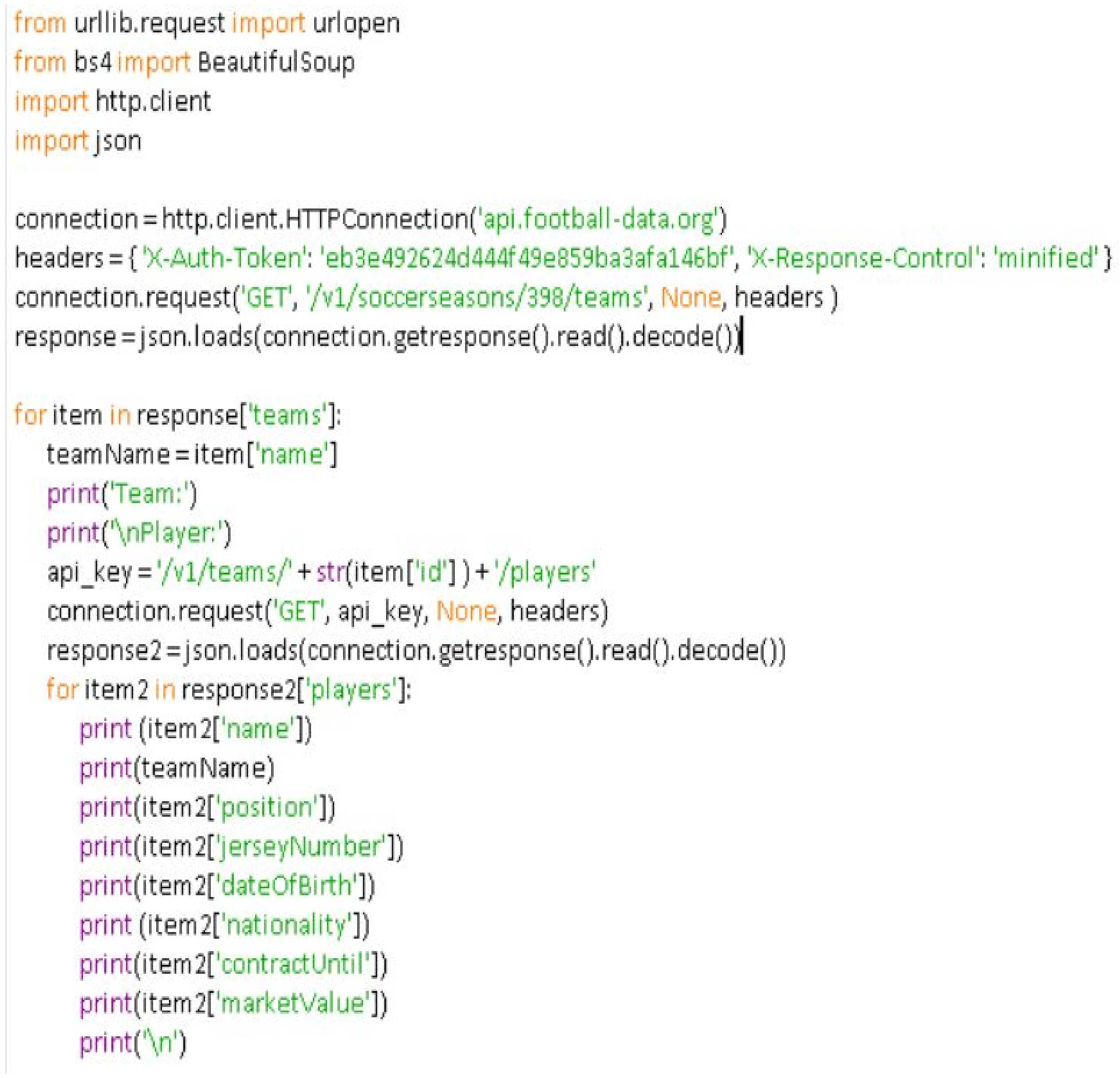
We e-mailed the owner of the website Daniel, to get the API key. After few minutes he gave me API authentication token which will act as a key. The drawing below is the basic visualized API resources and filters of the following website.



The screenshot below depicts the part saying import http.client to use the function in order to open up the url in a different way as the owner wanted us to do. This url contained numerous brackets meaning it’s dictionary. Which also means we need to use json.

The header X-Auth-Token is basically the API Key to request. X-Response-Control has three different choices of data; ‘Full’, ‘Minified’, ‘Compressed’. There are numerous other request we can call. To call up particular leagues in the season 2015. As an example do,, -X GET http://api.football-data.org/v1/soccerseasons /?seasons=2015. Then it will bring in Bundesliga 1,2 (German Ligue), Ligue 1,2(French), Premier League (English), Serie A (Italy), ….etc each with specific ID number for the season 2015/2016. EPL had the “id” = 398, Then, b calling -X GET ~/v1/soccerseasons/398/teams we can bring in every teams in EPL, each with specific IDs. In that same logic, I ran a for-loop to bring in every team in EPL. To bring in Every player in EPL, I ran a for-loop inside once more with ‘~/v1/teams/' + str(item['id']) + ‘/players’ to get into players on each team with different ‘id’s.

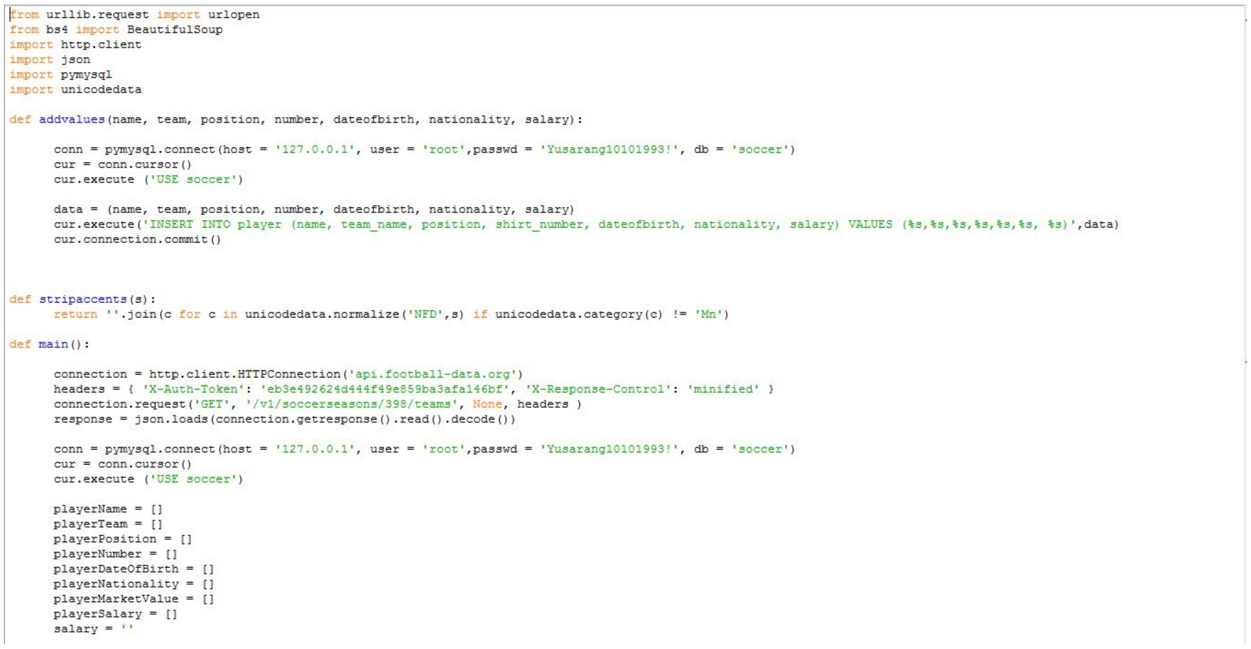
For the players, the API contained 'id','name','position','jersey number', 'DOB', 'nationality', 'contract information', and 'market value' (in euros). For the teams, it contained 'name', 'short name', 'squad market value', 'url of logo in wikipedia'.



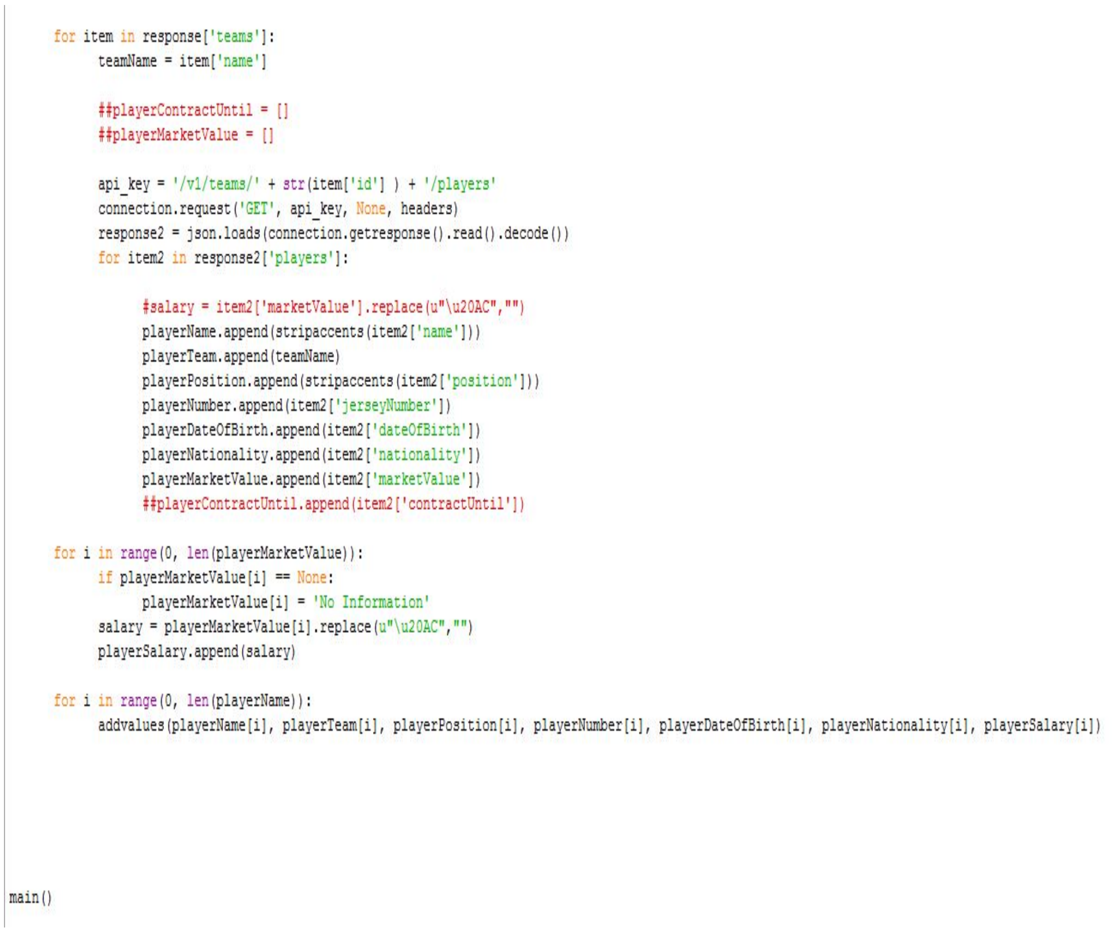
```python
from urllib.request import urlopen
from bs4 import BeautifulSoup
import http.client
import json

connection = http.client.HTTPConnection('api.football-data.org')
headers = { 'X-Auth-Token': 'eb3e492624d444f49e859ba3afa146bf', 'X-Response-Control': 'minified' }
connection.request('GET', '/v1/soccerseasons/398/teams', None, headers )
response = json.loads(connection.getresponse().read().decode())

for item in response['teams']:
    teamName = item['name']
    print('Team:')
    print('\nPlayer:')
    api_key = '/v1/teams/' + str(item['id'] ) + '/players'
    connection.request('GET', api_key, None, headers)
    response2 = json.loads(connection.getresponse().read().decode())
    for item2 in response2['players']:
        print (item2['name'])
        print(teamName)
        print(item2['position'])
        print(item2['jerseyNumber'])
        print(item2['dateOfBirth'])
        print (item2['nationality'])
        print(item2['contractUntil'])
        print(item2['marketValue'])
        print('\n')
```

3. **PyMySQL**

The screenshot below shows the python code we wrote for implementing pymysql to transfer scraped data from the website into our mysql database. Addvalues function takes each element of data from the lists of scraped data and adds them into mysql database. It first connects to mysql to log into the database then "cur.execute" inserts the data into "player" table. "cur.connect.commit()" sends a commit statement to the MYSQL server to commit the current transaction of data, Stripaccents function basically replaces any alphabets with accents with normal alphabets. One of the problem we reached during transferring data was that some of the player's name had special characters and cmd could not recognize those characters. In order to fix this, we called unicodedata that gave us access to the Unicode Character Database. The function then normalizes the special character to normal character if it finds one. For storing data from the website, we first made empty lists of player name, team, position, etc… Then we appended each elements in the data into each lists, using for loops. There was similar problem for player's market value where the data had euro symbol, which could not be recognized by cmd. Since euro symbol could not be replaced with stripaccent function, we had to use a different function or write a different code to get rid of euro symbol. In order to do this, we first stored all of the player's market value data into a list, then we looped through each element in the list to replace the euro symbol with empty character. The fixed market value in string format was added to a new list for simplification. Some of the player's did not have market value information, which got stored as NULL in MYSQL Database. To fix this, we used if function to replace None value to "No Information", so that the database can recognize that there is no market value data for certain players.



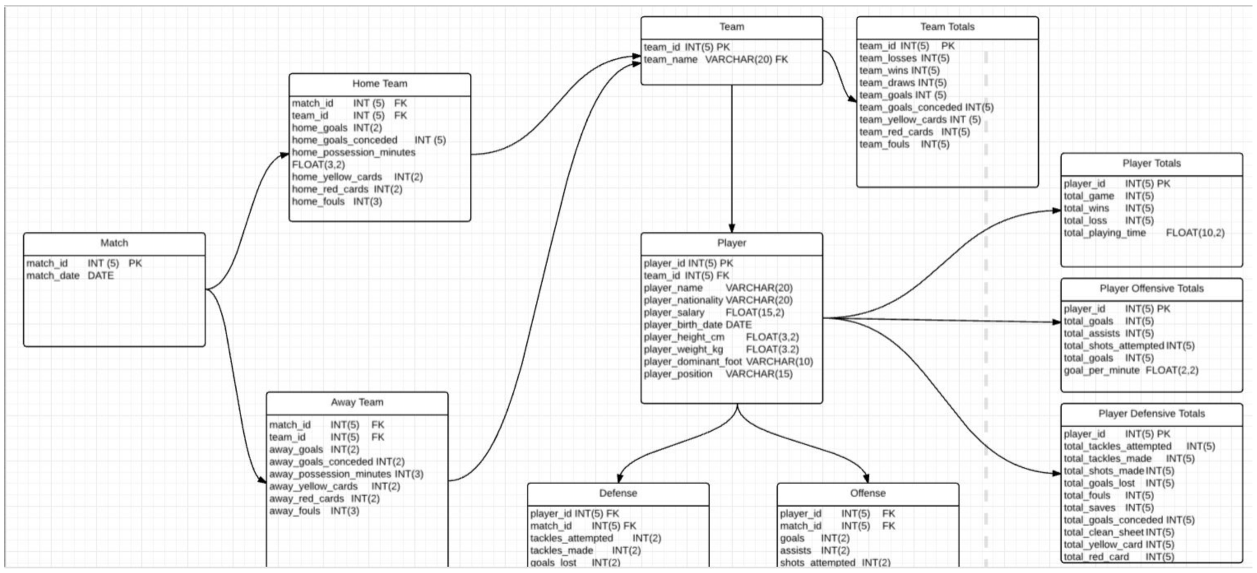
```python
from urllib.request import urlopen
from bs4 import BeautifulSoup
import http.client
import json
import pymysql
import unicodedata

def addvalues(name, team, position, number, dateofbirth, nationality, salary):

      conn = pymysql.connect(host = '127.0.0.1', user = 'root',passwd = 'Yusarang10101993!', db = 'soccer')
      cur = conn.cursor()
      cur.execute ('USE soccer')

      data = (name, team, position, number, dateofbirth, nationality, salary)
      cur.execute('INSERT INTO player (name, team_name, position, shirt_number, dateofbirth, nationality, salary) VALUES (%s,%s,%s,%s,%s,%s, %s)',data)
      cur.connection.commit()



def stripaccents(s):
      return ''.join(c for c in unicodedata.normalize('NFD',s) if unicodedata.category(c) != 'Mn')

def main():

      connection = http.client.HTTPConnection('api.football-data.org')
      headers = { 'X-Auth-Token': 'eb3e492624d444f49e859ba3afa146bf', 'X-Response-Control': 'minified' }
      connection.request('GET', '/v1/soccerseasons/398/teams', None, headers )
      response = json.loads(connection.getresponse().read().decode())

      conn = pymysql.connect(host = '127.0.0.1', user = 'root',passwd = 'Yusarang10101993!', db = 'soccer')
      cur = conn.cursor()
      cur.execute ('USE soccer')

      playerName = []
      playerTeam = []
      playerPosition = []
      playerNumber = []
      playerDateOfBirth = []
      playerNationality = []
      playerMarketValue = []
      playerSalary = []
      salary = ''
```



```python
      for item in response['teams']:
            teamName = item['name']

            ##playerContractUntil = []
            ##playerMarketValue = []

            api_key = '/v1/teams/' + str(item['id'] ) + '/players'
            connection.request('GET', api_key, None, headers)
            response2 = json.loads(connection.getresponse().read().decode())
            for item2 in response2['players']:

                  #salary = item2['marketValue'].replace(u"\u20AC","")
                  playerName.append(stripaccents(item2['name']))
                  playerTeam.append(teamName)
                  playerPosition.append(stripaccents(item2['position']))
                  playerNumber.append(item2['jerseyNumber'])
                  playerDateOfBirth.append(item2['dateOfBirth'])
                  playerNationality.append(item2['nationality'])
                  playerMarketValue.append(item2['marketValue'])
                  ##playerContractUntil.append(item2['contractUntil'])

      for i in range(0, len(playerMarketValue)):
            if playerMarketValue[i] == None:
                  playerMarketValue[i] = 'No Information'
            salary = playerMarketValue[i].replace(u"\u20AC","")
            playerSalary.append(salary)

      for i in range(0, len(playerName)):
            addvalues(playerName[i], playerTeam[i], playerPosition[i], playerNumber[i], playerDateOfBirth[i], playerNationality[i], playerSalary[i])






main()
```

We combined all the scraping scripts into one executable python script so that data could be imported into the database by running one file. This file is labeled ‘scraping.py’.

4. Design & Methodology



Initially, the structure of our database was designed to be a lot more complex than it is now. As you can see in the figure above, the database design included numerous data entries that we were unable to retrieve. Due to the inaccessibility of soccer data in general, we had to simplify the database to account for the data that we were able to scrape. Offensive and Defensive stats were completely removed along with the home team and away team tables. The home team and away team data was consolidated into the match table which was renamed to game due to a pre-existing function in MySQL with the same name. The new data dictionary was made as follows…

Game

game_id SMALLINT UNSIGNED AUTO_INCREMENT (PK)

game_date VARCHAR(10)

home_team VARCHAR(20)

away_team VARCHAR(20)

home_goals INT(2)

away_goals INT(2)

Player

player_id SMALLINT UNSIGNED AUTO_INCREMENT (PK)

name VARCHAR(50)

shirt_number INT(2)

position VARCHAR(20)

nationality VARCHAR(20)

team_name VARCHAR(40)

date_of_birth VARCHAR(10)

salary VARCHAR(20)

<u>Team</u>

Team_id SMALLINT UNSIGNED AUTO_INCREMENT (PK)

Team_name VARCHAR(30)

Rank INT(2)

<u>Team_Totals</u>

Team_id SMALLINT UNSIGNED AUTO_INCREMENT (PK)

Team_wins INT(5)

Team_losses INT(5)

Team_draws INT(5)

Team_goals INT(5)

Team_goals_conceded INT(5)

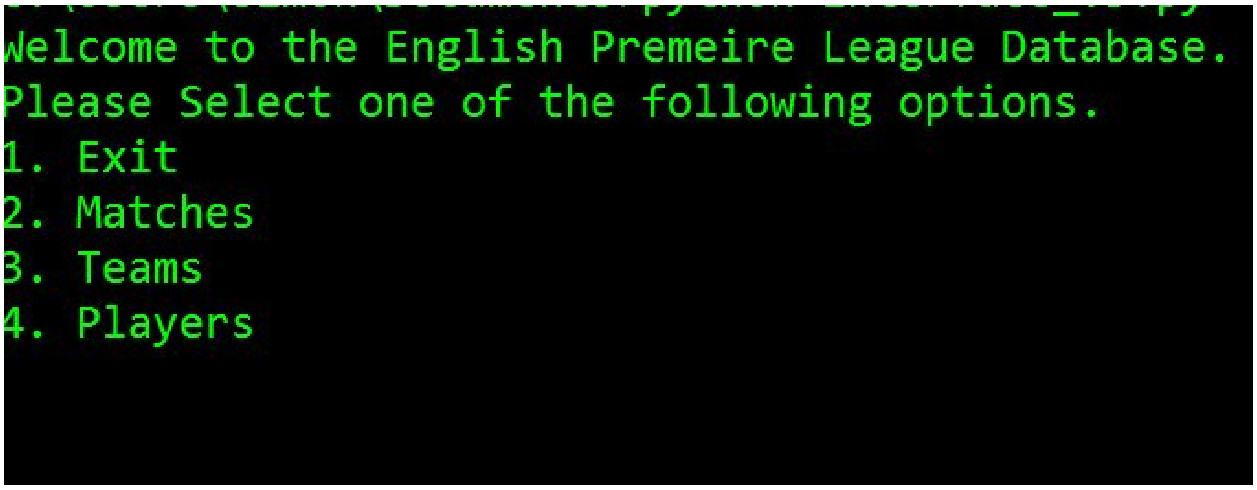
Team_games INT(5)

Team_points INT(5)

Team_difference INT(5)

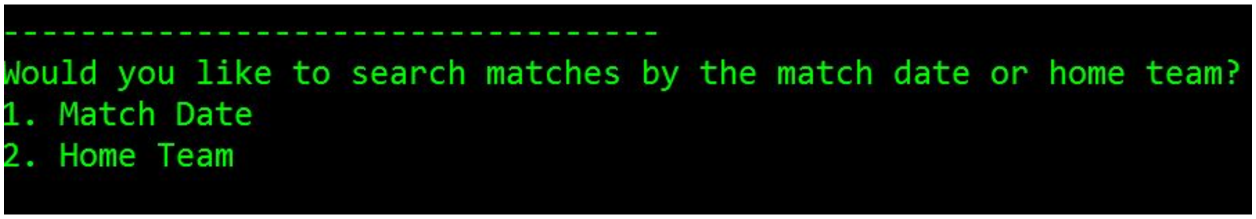
As visible in the data dictionary, the table was simplified into 4 tables that are connected through team_id and player_id. This allows for joins between tables to produce various types of data.

The user interface to traverse through the database was a simple python script that creates a command line interface. On the opening screen the user is welcomed to a menu options list. Here the user can select whether to exit, look at team data, look at player data, or look at match data.



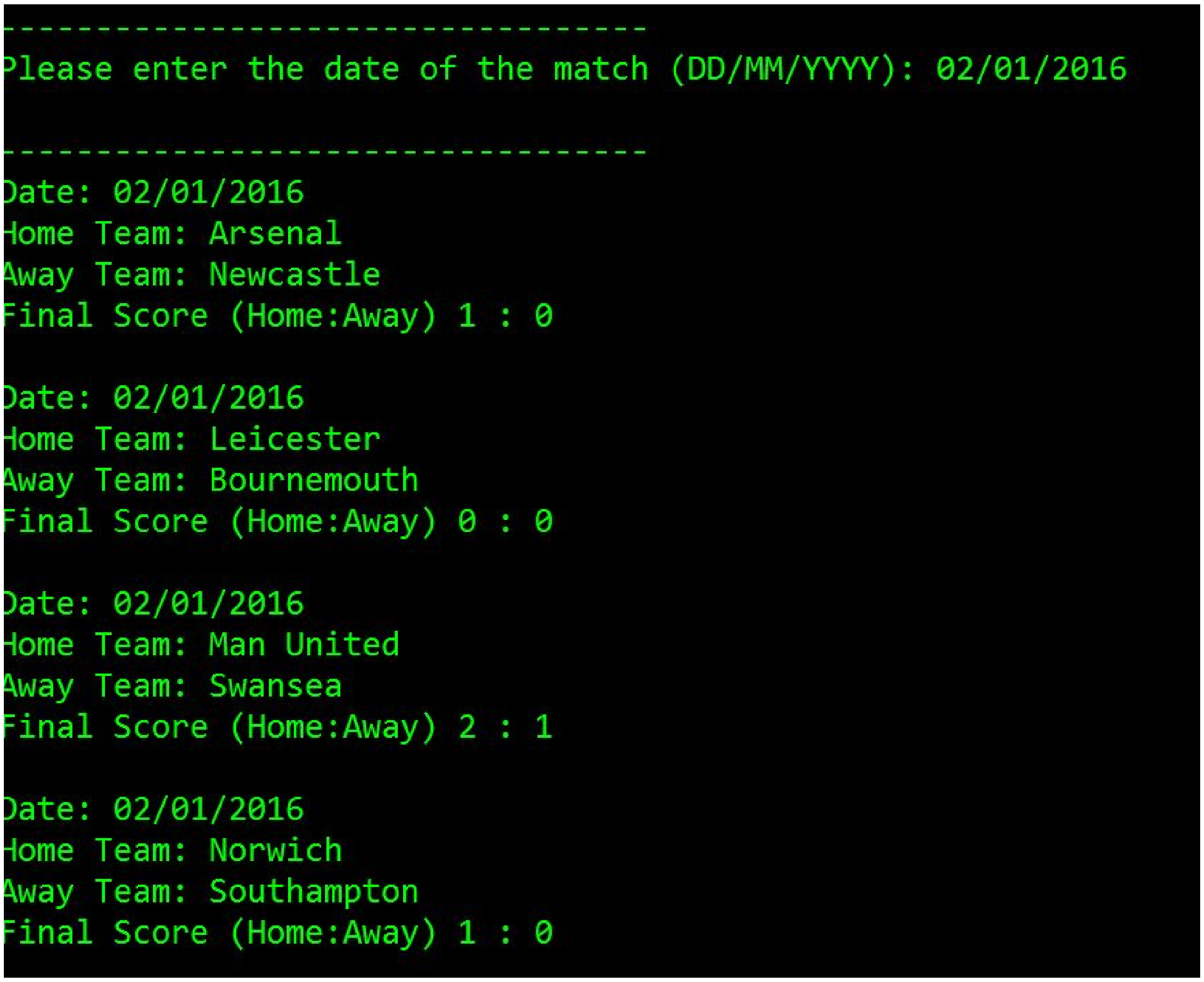
```
Welcome to the English Premeire League Database.
Please Select one of the following options.
1. Exit
2. Matches
3. Teams
4. Players
```

The user then inputs the number of the corresponding menu option and hits enter. For match data the user is allowed to choose between searching by match date or home team.



```
----------------------------------
Would you like to search matches by the match date or home team?
1. Match Date
2. Home Team
```

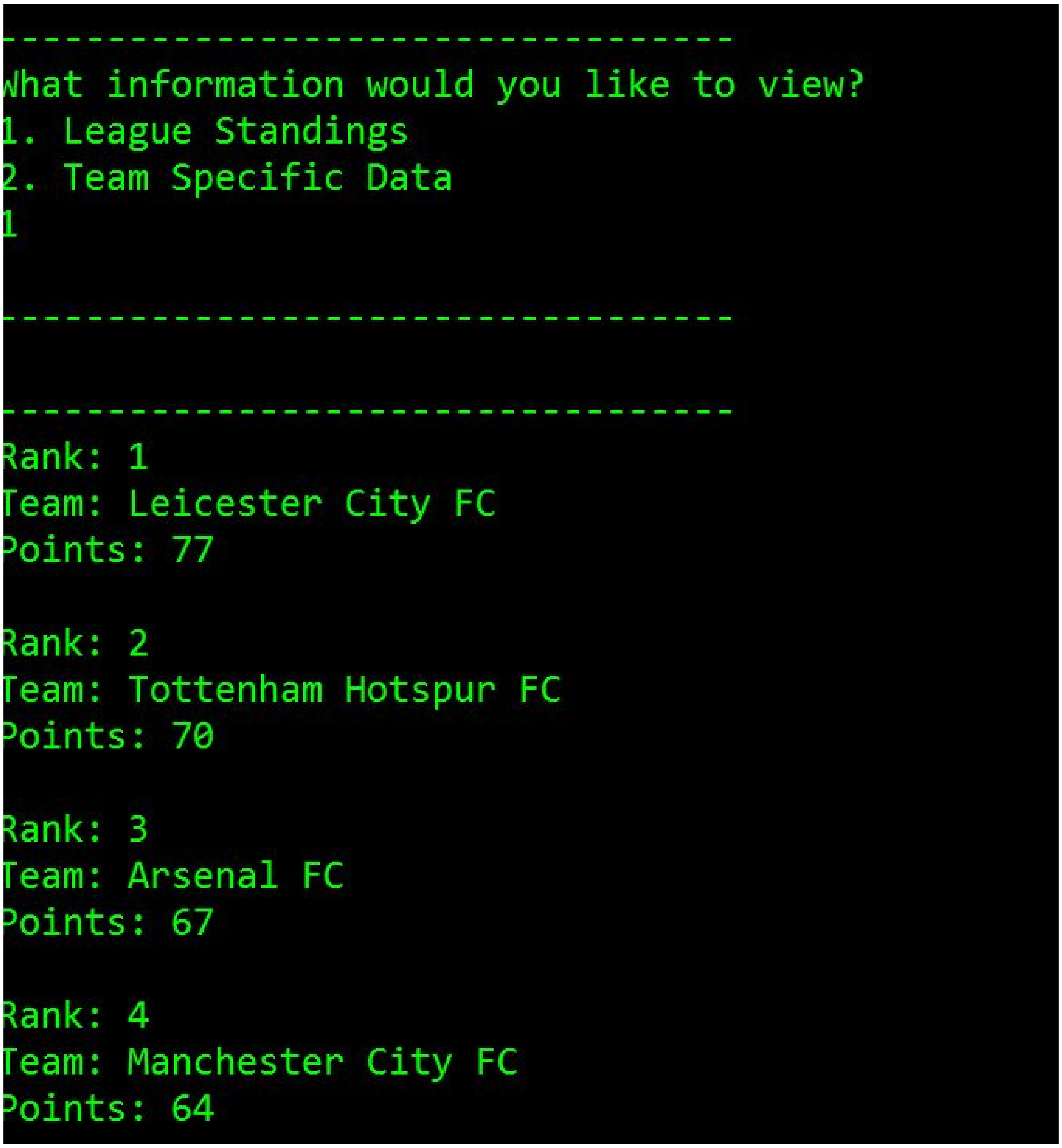
If searching by match date, the user will be prompted to enter the date of the match in the format (DD/MM/YYYY) and the interface will display all matches played on that date. For example, here are the match listings from 02/01/2016.



```
----------------------------------
Please enter the date of the match (DD/MM/YYYY): 02/01/2016

----------------------------------
Date: 02/01/2016
Home Team: Arsenal
Away Team: Newcastle
Final Score (Home:Away) 1 : 0

Date: 02/01/2016
Home Team: Leicester
Away Team: Bournemouth
Final Score (Home:Away) 0 : 0

Date: 02/01/2016
Home Team: Man United
Away Team: Swansea
Final Score (Home:Away) 2 : 1

Date: 02/01/2016
Home Team: Norwich
Away Team: Southampton
Final Score (Home:Away) 1 : 0
```

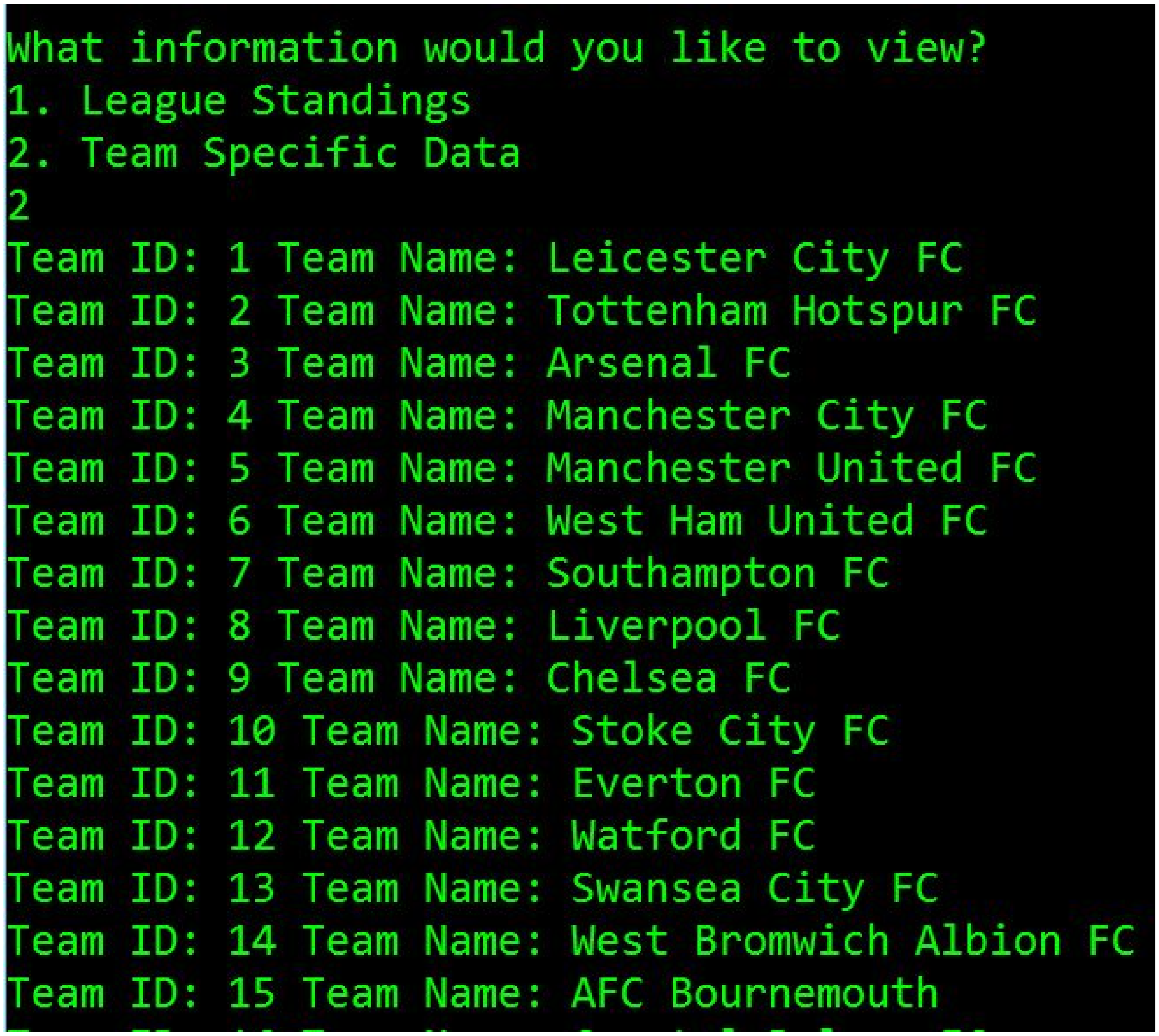
If searching by home team, the user will be prompted to enter the name of the home team and all the matches played by that home team will be displayed.

If the user chooses to look at team data, the user will be prompted if he or she would like to look at league standings or team specific data. If the user selects league standings, teams along with their corresponding ranks and amount of points they accrued will be displayed in the console

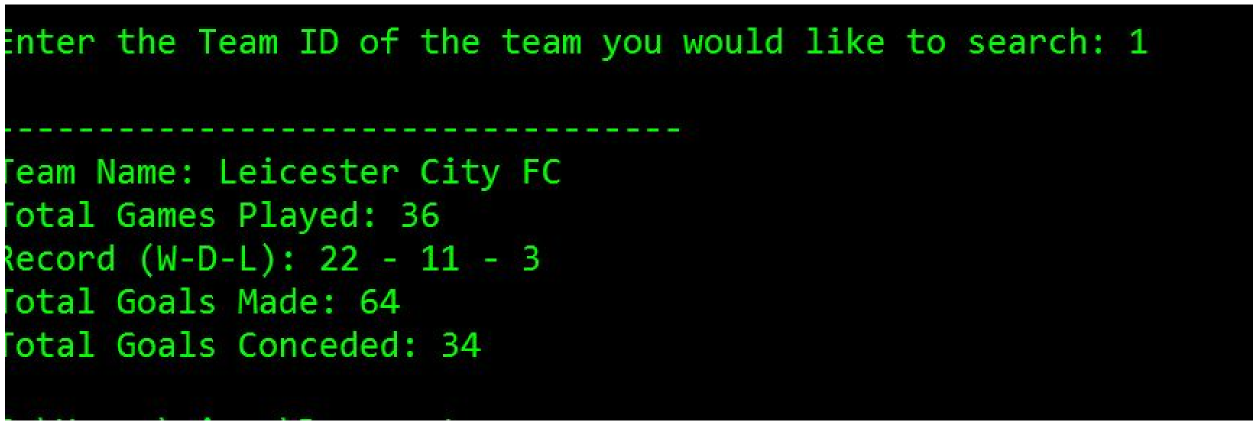


```
----------------------------------
What information would you like to view?
1. League Standings
2. Team Specific Data
1


----------------------------------


----------------------------------
Rank: 1
Team: Leicester City FC
Points: 77

Rank: 2
Team: Tottenham Hotspur FC
Points: 70

Rank: 3
Team: Arsenal FC
Points: 67

Rank: 4
Team: Manchester City FC
Points: 64
```

If the user decides to look at team specific data, the user will be given a list of the teams along with their corresponding team ids. The user will then select the team id of the team he or she wishes to view and then the corresponding information will display.

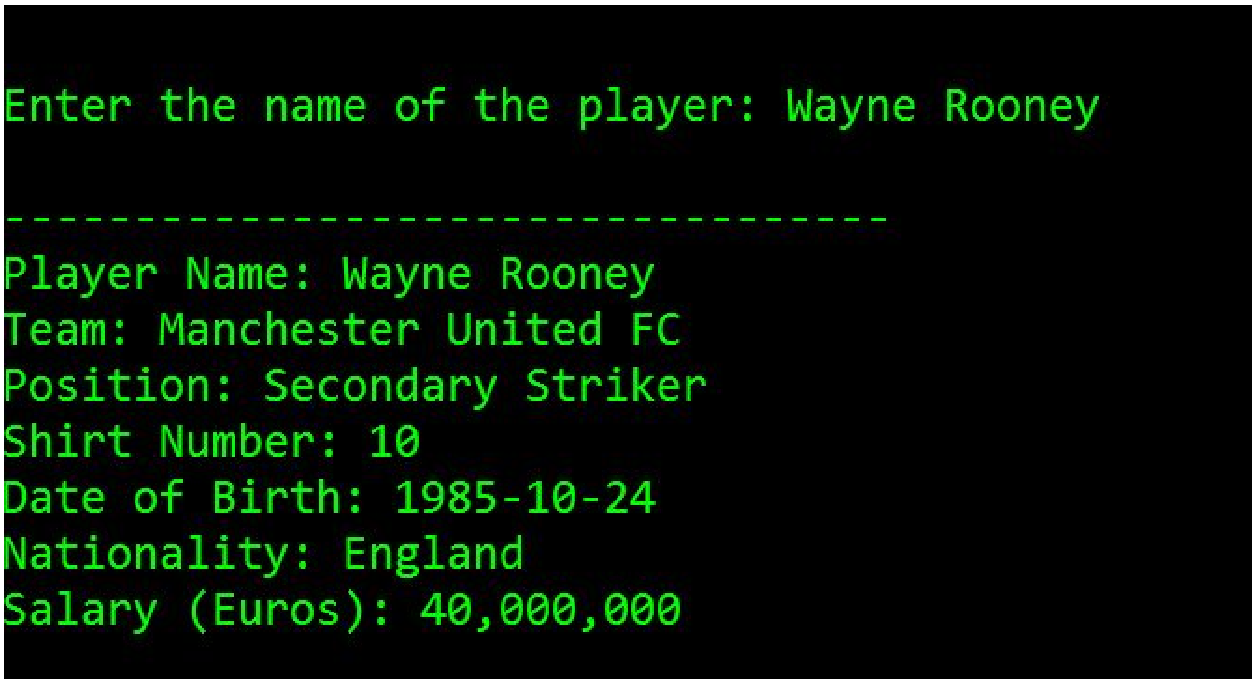


```
What information would you like to view?
1. League Standings
2. Team Specific Data
2
Team ID: 1 Team Name: Leicester City FC
Team ID: 2 Team Name: Tottenham Hotspur FC
Team ID: 3 Team Name: Arsenal FC
Team ID: 4 Team Name: Manchester City FC
Team ID: 5 Team Name: Manchester United FC
Team ID: 6 Team Name: West Ham United FC
Team ID: 7 Team Name: Southampton FC
Team ID: 8 Team Name: Liverpool FC
Team ID: 9 Team Name: Chelsea FC
Team ID: 10 Team Name: Stoke City FC
Team ID: 11 Team Name: Everton FC
Team ID: 12 Team Name: Watford FC
Team ID: 13 Team Name: Swansea City FC
Team ID: 14 Team Name: West Bromwich Albion FC
Team ID: 15 Team Name: AFC Bournemouth
```



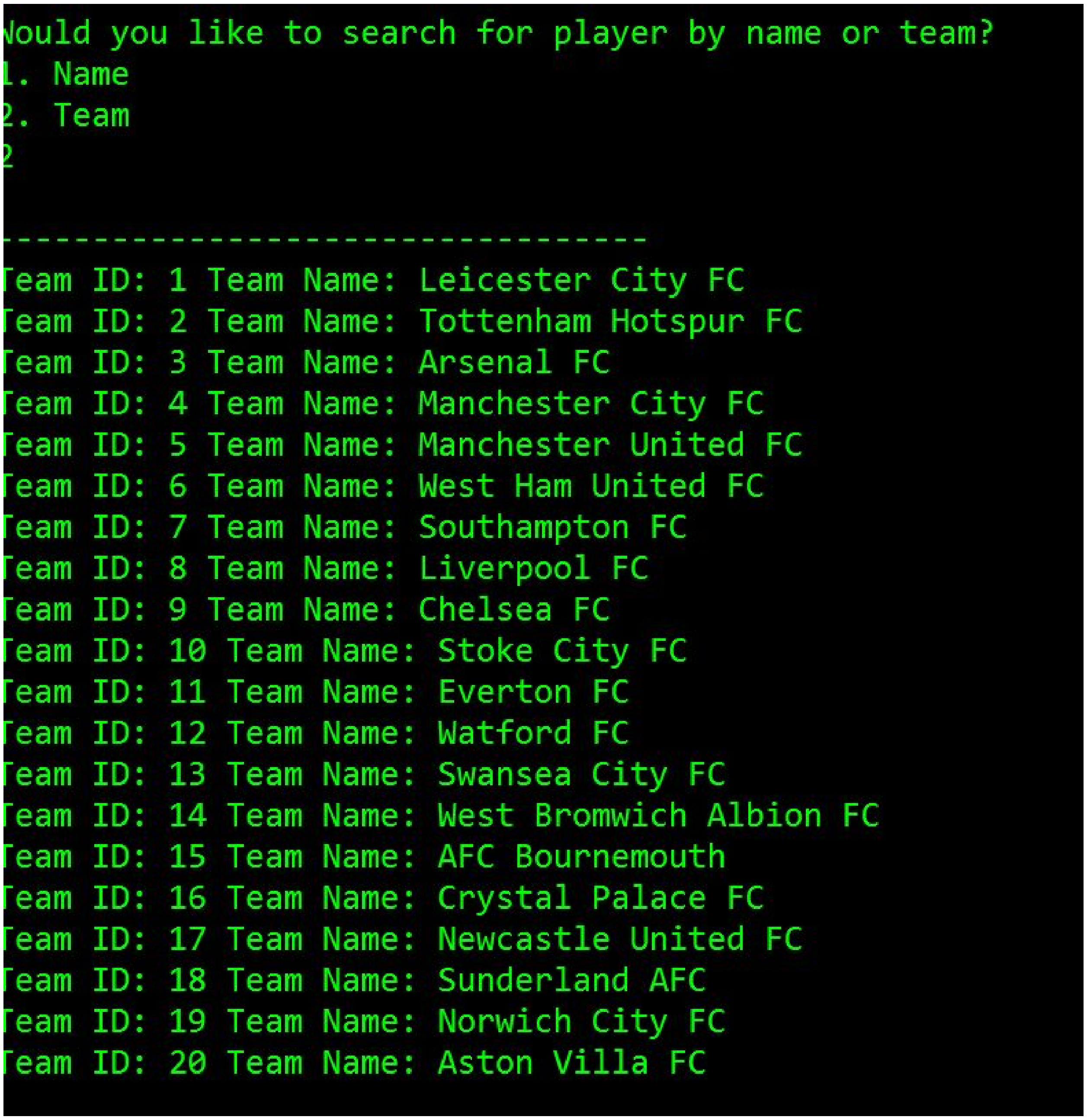
```
nter the Team ID of the team you would like to search: 1


----------------------------------
eam Name: Leicester City FC
otal Games Played: 36
ecord (W-D-L): 22 - 11 - 3
otal Goals Made: 64
otal Goals Conceded: 34
```

Finally , the user can choose to view players. In this option, the user will be allowed to search for the player by their full name or by the team. If the user selects name, the user will be asked to input the whole name of the player. After inputting this information, the user will be displayed the player's corresponding information.

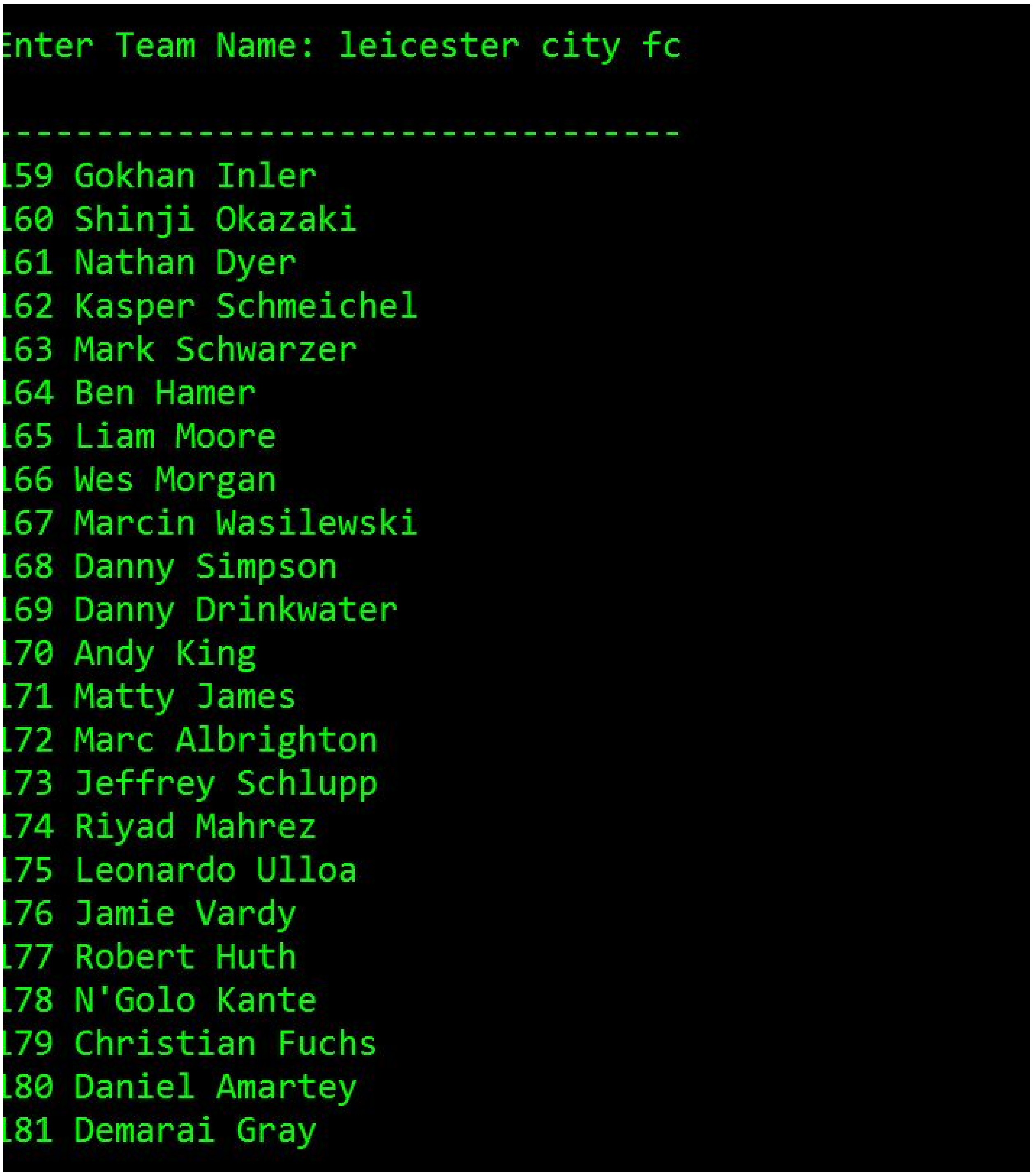


```
Enter the name of the player: Wayne Rooney

----------------------------------
Player Name: Wayne Rooney
Team: Manchester United FC
Position: Secondary Striker
Shirt Number: 10
Date of Birth: 1985-10-24
Nationality: England
Salary (Euros): 40,000,000
```

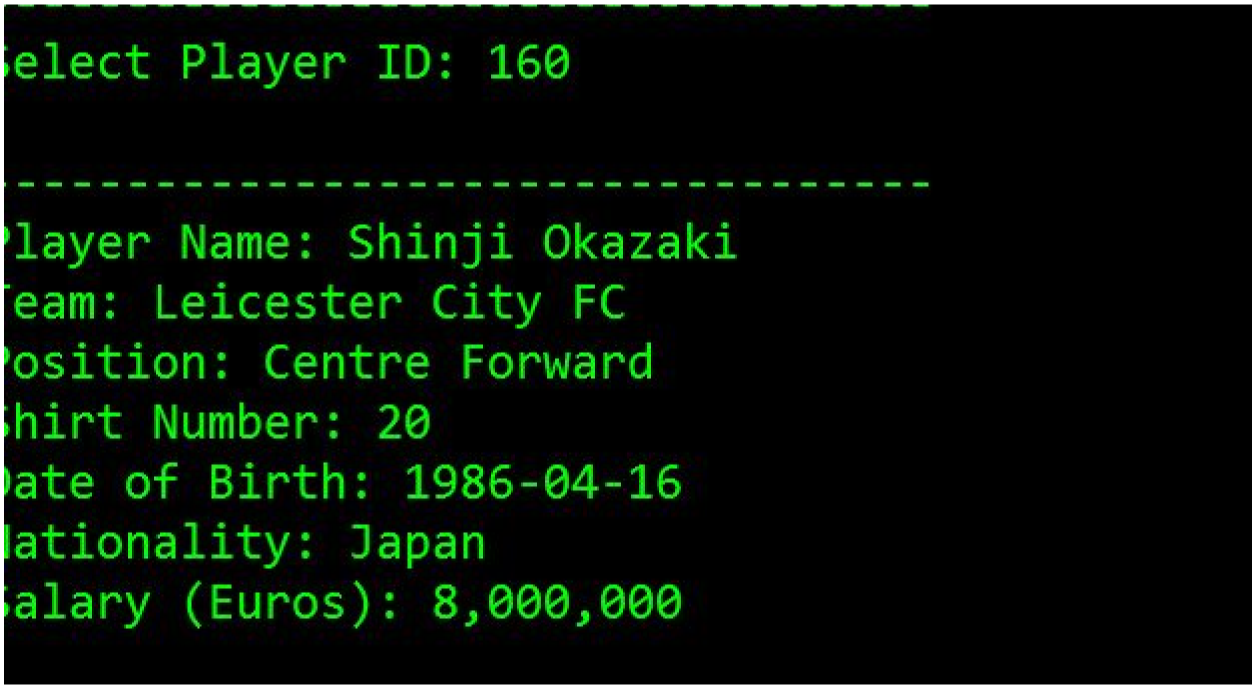
If the user chooses to search for the player by team, the user will be asked to enter the name of the team they wish to search for. After inputting the team name, the user will see a list of players and their corresponding player ids. The user will then input the player id of the player he or she wishes to view information on and the console will display this desired information



```
Would you like to search for player by name or team?
1. Name
2. Team
2


---------------------------------
Team ID: 1 Team Name: Leicester City FC
Team ID: 2 Team Name: Tottenham Hotspur FC
Team ID: 3 Team Name: Arsenal FC
Team ID: 4 Team Name: Manchester City FC
Team ID: 5 Team Name: Manchester United FC
Team ID: 6 Team Name: West Ham United FC
Team ID: 7 Team Name: Southampton FC
Team ID: 8 Team Name: Liverpool FC
Team ID: 9 Team Name: Chelsea FC
Team ID: 10 Team Name: Stoke City FC
Team ID: 11 Team Name: Everton FC
Team ID: 12 Team Name: Watford FC
Team ID: 13 Team Name: Swansea City FC
Team ID: 14 Team Name: West Bromwich Albion FC
Team ID: 15 Team Name: AFC Bournemouth
Team ID: 16 Team Name: Crystal Palace FC
Team ID: 17 Team Name: Newcastle United FC
Team ID: 18 Team Name: Sunderland AFC
Team ID: 19 Team Name: Norwich City FC
Team ID: 20 Team Name: Aston Villa FC
```



```
nter Team Name: leicester city fc

----------------------------------
59 Gokhan Inler
60 Shinji Okazaki
61 Nathan Dyer
62 Kasper Schmeichel
63 Mark Schwarzer
64 Ben Hamer
65 Liam Moore
66 Wes Morgan
67 Marcin Wasilewski
68 Danny Simpson
69 Danny Drinkwater
70 Andy King
71 Matty James
72 Marc Albrighton
73 Jeffrey Schlupp
74 Riyad Mahrez
75 Leonardo Ulloa
76 Jamie Vardy
77 Robert Huth
78 N'Golo Kante
79 Christian Fuchs
80 Daniel Amartey
81 Demarai Gray
```



```
elect Player ID: 160

-------------------------------
layer Name: Shinji Okazaki
eam: Leicester City FC
osition: Centre Forward
hirt Number: 20
ate of Birth: 1986-04-16
ationality: Japan
alary (Euros): 8,000,000
```

5. Future improvements

For the future improvement, being able to find additional data is the key. If more data is provided we can add numerous other queries to find out what key value is leading the teams to victory. As mentioned above, finding the data you want/need is very difficult in this area. I have contacted API owners and tried various other ways; however since this data can be used in personal business or services to make money, many of websites were in-scrapable or difficult to scrape due to poorly structured html tags and large volume of data with countless <tr>s and <td>s.

6. If Done Again

The most of the problem in this project comes from scraping the data. Although we have made a great relational database, it was extremely difficult for us fill them in. While I was scraping and Simon has already almost finished our database. Instead of the data we needed for our database, I was only able to find some other portion of data such as, Score leaders, Home Defence Stats, Scoring Time, Goal Types, EPL Team Stats, ..etc. and all of these data wasn’t able to goin to our database because we would have to start everything over.

If we can start from the beginning, we must check what data sources are actually available then make relational database and move on.