

Table des matières

Table des figures	2
Introduction	4
1 Part 1 - webscraping	5
1.1 Question 1	5
1.2 Question 2	5
1.2.1 Question 2.a	5
1.2.2 Question 2.b	5
1.2.3 Question 2.c	5
1.3 Question 3	6
1.3.1 Question 3.a	6
1.3.2 Question 3.b	6
1.3.3 Question 3.c	6
1.3.4 Question 3.d	7
1.4 Question 4	7
1.4.1 Question 4.a	7
1.4.2 Question 4.b	7
1.5 Question 5	7
2 Part 2	8
2.1 Question 1	8
2.2 Question 2	8
2.3 Question 3	10
2.3.1 Question 3.a	10
2.3.2 Question 3.b	10
2.3.3 Question 3.c	10
2.3.4 Question 3.d	10
2.4 Question 4	10
2.4.1 Question 4.a	10
2.4.2 Question 4.b	10
2.4.3 Question 4.c	10
2.5 Question 5	10
Introduction	11

Table des figures

1	Profile table schema. <i>Source : SQLite Browser</i>	8
2	Player's Totals table schema <i>Source : SQLite Browser</i>	8
3	Player's Per Game table schema <i>Source : SQLite Browser</i>	9
4	Salary table schema <i>Source : SQLite Browser</i>	9
5	Active Players table schema <i>Source : SQLite Browser</i>	9
6	Team Information table schema <i>Source : SQLite Browser</i>	9
7	Team Statistics table schema <i>Source : SQLite Browser</i>	10

Introduction

1 Part 1 - webscraping

1.1 Question 1

We want to retrieve data for the players and the teams. The starting URL to do this are <http://www.basketball-reference.com/players/> and <http://www.basketball-reference.com/teams/> respectively.

1.2 Question 2

We decided to scrap the pages of all the players and not only the active ones, as the data could prove useful to assess the salaries of the players. However we also created a function `active_players_BS()` which return which players are active and which are not.

The players are classified according to the first letter of their last name. We first retrieve all the letters which have players whose name starts with it, from <http://www.basketball-reference.com/players>. For example, there is no player whose name starts with X. For a letter *c*, we then download the page <http://www.basketball-reference.com/players/c> and extract the links to the players' page.

1.2.1 Question 2.a

The regular expression we use to retrieve the letters is :

```
<a href="/players/([a-z]+)/">[A-Z]+</a></td>
```

The regular expression we use to identify the links to the players' page is the following :

```
[^p]><a href="(./players/./.+)"
```

1.2.2 Question 2.b

The code we use to retrieve the letters with BeautifulSoup is :

```
1 for row in soup('td', {'class': 'align_center bold_text valign_bottom xx_large_text'}):
2     letter = str(row.a.get('href')).split('/')[2])
```

The code we use to identify the links with BeautifulSoup is the following :

```
1 for player in soup.tbody.find_all('tr'):
2     link = str(player.td.a.get('href'))
```

1.2.3 Question 2.c

We choosed to use the regex code because it is faster and use less memory. In fact, we ran 100 instances of both versions and obtained the following results :

Method	Time	Memory size
regex	1.167 s	2.6 MB
BS	3.911 s	3.7 MB

Those results were obtained using pre-downloaded pages to avoid measuring the download times.

Furthermore, the regex code is more readable since it contains less functions that need to called.

1.3 Question 3

1.3.1 Question 3.a

Using the links parser from part 2, the entire player database was parsed using BeautifulSoup and basic profile information was scraped from the website. A standard template was designed to hold variables such as PlayerID (derived from the player's HTML link was used as an unique identifier), name, positions, shooting hand, height, weight, birth date, city, state/country of birth, experience and death date. Where there are no values listed in the profile page, null values are assigned to the variables.

1.3.2 Question 3.b

The method used to scrap the basic statistics and salaries data, is as follows :

- **Scraping phase** by analyzing the structure of the tables (HTML tags) into .csv files
- Adding an unique **Player ID** at each table constructed as follows : `/players/b/bryanko01.html` becomes `bbryanko01`. Each player then can be accessed with its unique key
- **Cleaning phase** where some column's formats are modified so that they can be loaded and manipulated in the SQLite database

Among all statistics of the basketball players, scraping some tables are sufficient to do the analysis requested in Part 2 (see ?? the tables selected as player's statistics are below :

- Totals
- Per Game

These tables deals with statistics by season (Totals) and by game (Per Game) which is enough to conduct the requested analysis. However, since tables are not exactly identical (HTML tags), scraping them wasn't as easy as expected. Numerous specialties have had to be taken in consideration.

As an example, here is referenced a special case encountered in Totals and Per Game tables when the player qualified to All-Star games. Next to the Season value, a star appears. Below the HTML code corresponding to the Totals table of Kobe Bryant.

```
1 <td align="left" >
2     <a href="/players/b/bryanko01/gamelog/1998/">1997-98</a>
3     <span class="bold_text" style="color:#c0c0c0">&nbsp;&#x2605;</span>
4 </td>
```

Scraping these web pages had been tricky in which the code should catch all these specialties. For further information about the code providing the player's statistics scraper, see the `player_statistics_BS.py` file.

The output tables could be find in `Data/Part2/3b` folder.

1.3.3 Question 3.c

Scraping the player's salaries tables require to scrap two main tables :

- Salaries (previous salaries)
- Contract (current and futures contracts if any)

Thus, the method is to merge these two tables. Here again, some specialties on tables structure render the scraping tricky. Indeed, Contract table doesn't have a thead, tbody and tfoot tag. So its scraping is done differently. Furthermore, salary's format is \$15,000,000 for instance. Unchanged, salary cannot be loaded to the SQLite database. Dollar symbol and commas have been removed so that the new salary's format is 15000000 (Cleaning phase).

1.3.4 Question 3.d

1.3.1 can be repeated using Regex. I have written a sample skeleton code (*profile_parser_regex.py*) to parse the play

1.4 Question 4

1.4.1 Question 4.a

1.4.2 Question 4.b

1.5 Question 5

2 Part 2

2.1 Question 1

Below is the database structure of the different tables. In total, there are 7 tables. The principal key for tables in figure 1, 2, 3 and 4 is `playerid`. For others tables, figure 5, 6 and 7, the principal key is `teamid`.














 <code>playerid</code>	<code>varchar(12)</code>	<code>'playerid' varchar(12) NOT NULL</code>
 <code>name</code>	<code>varchar(56)</code>	<code>'name' varchar(56) NOT NULL</code>
 <code>position1</code>	<code>varchar(2)</code>	<code>'position1' varchar(2)</code>
 <code>position2</code>	<code>varchar(2)</code>	<code>'position2' varchar(2)</code>
 <code>position3</code>	<code>varchar(2)</code>	<code>'position3' varchar(2)</code>
 <code>shoots</code>	<code>char(1)</code>	<code>'shoots' char(1)</code>
 <code>height</code>	<code>integer</code>	<code>'height' integer</code>
 <code>weight</code>	<code>integer</code>	<code>'weight' integer</code>
 <code>dob</code>	<code>date</code>	<code>'dob' date</code>
 <code>city</code>	<code>varchar(56)</code>	<code>'city' varchar(56)</code>
 <code>state</code>	<code>varchar(56)</code>	<code>'state' varchar(56)</code>
 <code>experience</code>	<code>integer</code>	<code>'experience' integer</code>
 <code>dod</code>	<code>date</code>	<code>'dod' date</code>

FIGURE 1 – Profile table schema. *Source : SQLite Browser*
































 <code>playerid</code>	<code>varchar(12)</code>	<code>'playerid' varchar(12) NOT NULL</code>
 <code>season</code>	<code>integer</code>	<code>'season' integer</code>
 <code>age</code>	<code>integer</code>	<code>'age' integer</code>
 <code>team</code>	<code>char(3)</code>	<code>'team' char(3)</code>
 <code>league</code>	<code>char(3)</code>	<code>'league' char(3)</code>
 <code>position</code>	<code>varchar(6)</code>	<code>'position' varchar(6)</code>
 <code>g</code>	<code>integer</code>	<code>'g' integer</code>
 <code>gs</code>	<code>integer</code>	<code>'gs' integer</code>
 <code>mp</code>	<code>integer</code>	<code>'mp' integer</code>
 <code>fg</code>	<code>integer</code>	<code>'fg' integer</code>
 <code>fga</code>	<code>integer</code>	<code>'fga' integer</code>
 <code>fg_percent</code>	<code>double</code>	<code>'fg_percent' double</code>
 <code>three_point</code>	<code>integer</code>	<code>'three_point' integer</code>
 <code>three_point_a</code>	<code>integer</code>	<code>'three_point_a' integer</code>
 <code>three_point_percent</code>	<code>double</code>	<code>'three_point_percent' double</code>
 <code>two_point</code>	<code>integer</code>	<code>'two_point' integer</code>
 <code>two_point_a</code>	<code>integer</code>	<code>'two_point_a' integer</code>
 <code>two_point_percent</code>	<code>double</code>	<code>'two_point_percent' double</code>
 <code>efg_percent</code>	<code>double</code>	<code>'efg_percent' double</code>
 <code>ft</code>	<code>integer</code>	<code>'ft' integer</code>
 <code>fta</code>	<code>integer</code>	<code>'fta' integer</code>
 <code>ft_percent</code>	<code>double</code>	<code>'ft_percent' double</code>
 <code>orb</code>	<code>integer</code>	<code>'orb' integer</code>
 <code>drb</code>	<code>integer</code>	<code>'drb' integer</code>
 <code>trb</code>	<code>integer</code>	<code>'trb' integer</code>
 <code>ast</code>	<code>integer</code>	<code>'ast' integer</code>
 <code>stl</code>	<code>integer</code>	<code>'stl' integer</code>
 <code>blk</code>	<code>integer</code>	<code>'blk' integer</code>
 <code>tov</code>	<code>integer</code>	<code>'tov' integer</code>
 <code>pf</code>	<code>integer</code>	<code>'pf' integer</code>
 <code>pts</code>	<code>integer</code>	<code>'pts' integer</code>

FIGURE 2 – Player's Totals table schema *Source : SQLite Browser*

2.2 Question 2

Active players during 2011-2012 season According to the SQLite script `question2.sql` in `SQLdirectory`, the 2012 season among the 4288 players in total.

Distribution in each position ---- 20 C 91 C-PF 1 PF 90 PF-SF 1 PG 78 SF 86 SF-PF 1 SG 84

playerid	varchar(12)	`playerid` varchar(12) NOT NULL
season	integer	`season` integer
age	integer	`age` integer
team	char(3)	`team` char(3)
league	char(3)	`league` char(3)
position	varchar(6)	`position` varchar(6)
g	integer	`g` integer
gs	integer	`gs` integer
mp	double	`mp` double
fg	double	`fg` double
fga	double	`fga` double
fg_percent	double	`fg_percent` double
three_point	integer	`three_point` integer
three_point_a	double	`three_point_a` double
three_point_percent	double	`three_point_percent` double
two_point	integer	`two_point` integer
two_point_a	double	`two_point_a` double
two_point_percent	double	`two_point_percent` double
ft	double	`ft` double
fta	double	`fta` double
ft_percent	double	`ft_percent` double
orb	double	`orb` double
drb	double	`drb` double
trb	double	`trb` double
ast	double	`ast` double
stl	double	`stl` double
blk	double	`blk` double
tov	double	`tov` double
pf	double	`pf` double
pts	double	`pts` double

FIGURE 3 – Player's Per Game table schema *Source : SQLite Browser*

playerid	varchar(12)	`playerid` varchar(12) NOT NULL
season	integer	`season` integer
team	varchar(24)	`team` varchar(24)
league	char(3)	`league` char(3)
salary	integer	`salary` integer

FIGURE 4 – Salary table schema *Source : SQLite Browser*

playerID	TEXT	`playerID` TEXT
active	TEXT	`active` TEXT

FIGURE 5 – Active Players table schema *Source : SQLite Browser*

teamid	varchar(12)	`teamid` varchar(12) NOT NULL
city	varchar(24)	`city` varchar(24)
state	varchar(24)	`state` varchar(24)
season	integer	`season` integer
first_season	integer	`first_season` integer
last_season	integer	`last_season` integer
wins	integer	`wins` integer
losses	integer	`losses` integer
winlose_percent	double	`winlose_percent` double
playoff_app	integer	`playoff_app` integer
championships	integer	`championships` integer

FIGURE 6 – Team Information table schema *Source : SQLite Browser*

Average age avg age = 30.01

Average weight avg weight = 219.947

Average experience avg experience = 7.35

teamid	varchar(12)	`teamid` varchar(12) NOT NULL
season	integer	`season` integer
league	char(3)	`league` char(3)
team	varchar(24)	`team` varchar(24) NOT NULL
win	integer	`win` integer
loss	integer	`loss` integer
wl_percent	double	`wl_percent` double
finish	integer	`finish` integer
srs	double	`srs` double
pace	double	`pace` double
rel_pace	double	`rel_pace` double
ortg	double	`ortg` double
rel_ortg	double	`rel_ortg` double
drtg	double	`drtg` double
rel_drtg	double	`rel_drtg` double
playoffs	varchar(128)	`playoffs` varchar(128)

FIGURE 7 – Team Statistics table schema *Source : SQLite Browser*

Average salary in the season `avg salary = 4699756.0`

Average salary in the career `avg career sal = 36147056.02`

2.3 Question 3

2.3.1 Question 3.a

2.3.2 Question 3.b

2.3.3 Question 3.c

2.3.4 Question 3.d

2.4 Question 4

2.4.1 Question 4.a

2.4.2 Question 4.b

2.4.3 Question 4.c

2.5 Question 5

Conclusion