Guided Project - 1

IS5126 : Hands on with Business Analytics

Group - 11

9-Feb-2015

Anbarasan Thangapalam Rathnahar Gnanaselwyn (**A0119959J**)

Liu Wenru (**A0134523N**)

Ram Vibhakar Suthagar (**A0120054B**)

Vu Hai Van (**A0134472J**)

Our Approach:

Web Scrapping

- After identifying the starting URL's namely http://www.basketball-reference.com/players/ and http://www.basketball-reference.com/teams/, we verified the URL's with robot.txt file whether our crawler can crawl through these pages. This was done to respect the terms and conditions of basketball-reference.com
- We used a combination of BeautifulSoup with SoupStrainer and RegEx to scrap the data. This helped to get the data effectively.
- The **player_crawler.py** crawls through the player information and the **team_crawler.py** crawls through the team information.
- The scrapped data was stored as CSV files under data/ folder.

ETL

- The data from the CSV file is loaded into SQLITE3 into raw_* tables. The database filename is **nba_db.sqlite**
- **T**he data scrapped from the website is not clean, hence we performed the following cleaning operations:
    - Removed '$' and ',' from the salary column so that it can be converted to INTEGER for analysis
    - Removed unnecessary empty rows and 'Did not play' records from raw_player_statistics
    - Removed some special characters like '*' from season and player name records.
    - We used URL of franchises and player to bring relationship between the records. But some we had team URL which was redirected to franchise page. For example 'http://basketball-reference.com/teams/BRK/' will be redirected to 'http://www.basketball-reference.com/teams/NJN/index.html?redir'. So we wrote a python code (**find_redirect_urls()** from **team_crawler.py**) to extract the main team URL from redirect URL and used SQLITE to update the records
- Then we designed a database schema to load data from the raw tables. We applied normalization techniques to reduce the tables as much as possible.
- We defined proper data types and constraints like NULL, PRIMARY KEY and FOREIGN KEY to relevant columns
- We also defined **AUTOINCREMENT** keys to certain PRIMARY KEY so that the key could be automatically generated.
- Finally, we queried the database to answer the questions in Part 2

Part 1 - Web-scrapping

1. **Starting URL's to crawl are:**

    For player details : http://www.basketball-reference.com/players/
    For team details : http://www.basketball-reference.com/teams/

2. **Beautiful Soup vs Regular Expression.**

    A. **The regular expression code to retrieve links:**

```
request = urllib.urlopen(url)
all_player_links = re.findall('href="(/players/[a-z]/.*\.html)"',
request.read())
```

B. BeautifulSoup version of code to retrieve links:

```
soup = BeautifulSoup(request.read())
table = soup.find_all('table', {'class':'sortable stats_table'})
table_rows = table[0].tbody.find_all('tr')
all_player_link = []
for row in table_rows:
    td = row.find_all('td')
    if td[0].a:
        all_player_links.append(td[0].a['href'])
```

C. Comparision of Regular Expression and BeautifulSoup:

We chose Regular Expression for parsing the URL's as it is very easy and fast to scrape the required text from the HTML. This task was very difficult in BeautifulSoup as we had to navigate into different elements of the HTML DOM to retrieve the link. Also BeautifulSoup took a lot of time to parse the whole HTML document and convert into BeautifulSoup object.

But BeautifulSoup was helpful in scraping other data like tables as it is aware of the HTML DOM. This task was difficult in Regular expression.

So for the remaining part of scraping data we used a combination of BeautifulSoup with SoupStrainer and Regular Expression. The SoupStrainer class helps us to parse only specific section of HTML document so that the parsing is efficient.

Below is the code which uses SoupStrainer class with BeautifulSoup. Using this method we were able to enjoy the advantages of both BeautifulSoup and Regular Expression.

```
table_strainer = SoupStrainer('table', {'id': 'players'})
soup = BeautifulSoup(request.read(), parse_only=table_strainer)
urls = soup.find_all(href=re.compile('/players/[a-zA-Z]/[^
\.]*.html'))
```

We also made a benchmarking of different methods for getting the URL of all players from the starting URL using **timeit** package. The results are shown in the following table.

METHOD	TIME TAKEN (SECONDS)
REGULAR EXPRESSION	71.4506931305
BEAUTIFUL SOUP	147.86285305
BEAUTIFUL SOUP WITH SOUPSTRAINER & REGULAR EXPRESSION	84.560970068

3. Parse the following information using BeautifulSoup:

A. Basic player profile information:

The following basic profile Information of each player is scrapped and stored in **data/player_basic_profile_info.csv**.

- name
- from

- to
- position
- height
- weight
- date of birth
- college
- active or inactive

B. Player statistics table:

The details from player statistics table are scrapped and stored into **data/player_stats.csv.**

C. Player Salaries:

The player salaries season wise are extracted and stored into **data/player_salary.csv**

D. Using RegEx:

Regular Expressions can be used to extract the information but it is not aware of the HTML document object model (DOM). So it is difficult to write RegEx that can scrape tabular data. Also it becomes difficult to debug and extend the functionality. BeautifulSoup provides convenient methods to read different tag, attributes and text. Hence BeautifulSoup is better than RegEx in this case.

4. Scrap team page:

Each team page is obtained from http://www.basketball-reference.com/teams/ and the details are scrapped using BeautifulSoup

A. Basic Team Information

The following basic team information as scrapped and stored into **data/team_basic_profile_info.csv**

- name
- league
- from and to
- games
- wins and losses
- win percentage
- no of years the team qualified playoffs
- no of years the team won in its division
- years the team won conference championship
- years the team won league championship

B. Team statistics:

The team statistics of each Active team is scrapped and stored into **data/team_stats.csv** file.

5. Other Information:

The other information that can be useful in building analytics for the team owners are

- The points information of various awards from (http://www.basketball-reference.com/awards/awards_2014.html) for each season. This information would be greatly helpful for the owners to choose players, decide the salary, find winning odds of the game.
    - Most Valuable Player award
    - Rookie of the year award

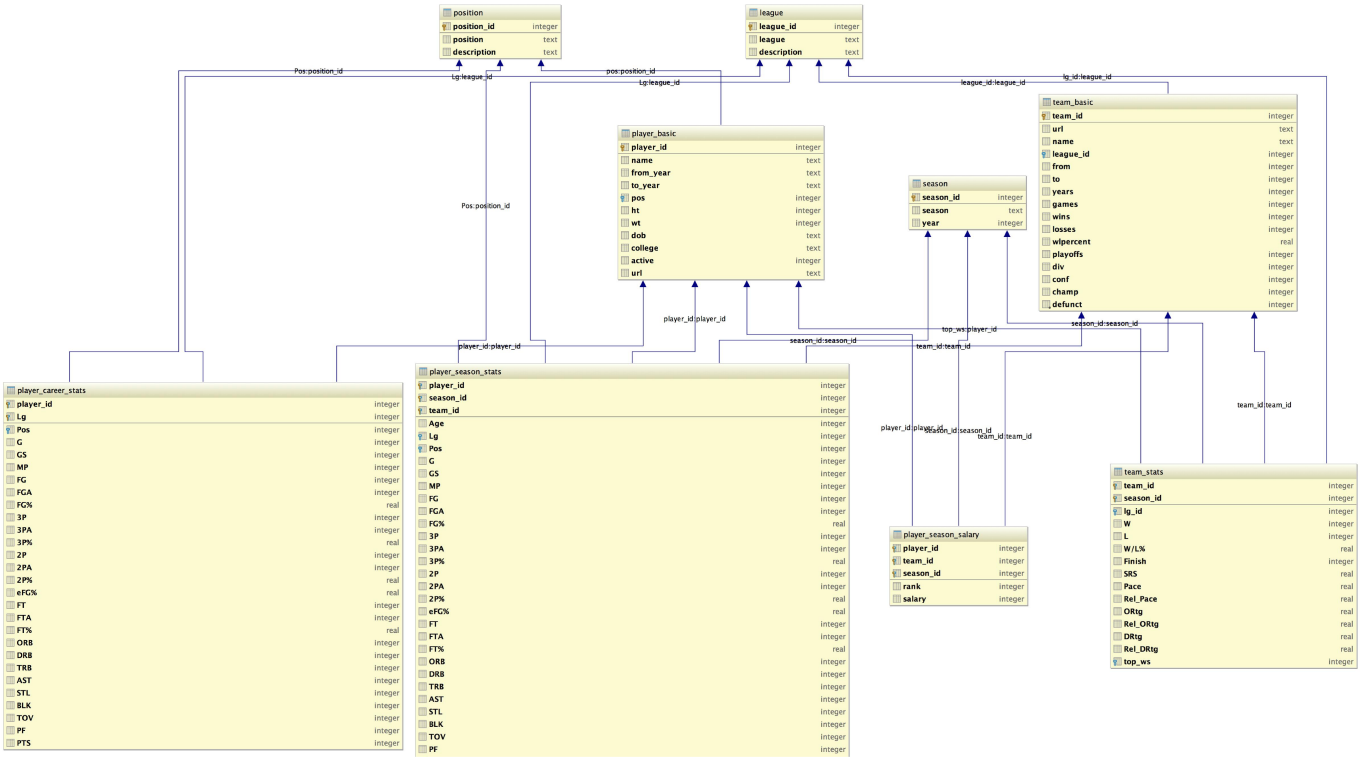
- Defensive player of the year award
- Sixth man of the year award
- Most improved player award
- Team mate of the year award
- Finals most valuable player award
- All-Star Game Most Valuable award
- Comeback Player of the Year award
- Sporting News Most Valuable Player awards
- Sporting News Rookie of the Year awards
- Coach of the Year awards

- Coach information (http://www.basketball-reference.com/coaches/) : This data would be helpful to choose coaches for the team and predict the effect of a coach on the team.

Part 2- ETL

1. Design of Database:

We used normalization techniques to reduce the database schema and cleaned data from raw_* tables are loaded into these tables. The tables and the relationships are shown in the figure.



Note: A high resolution DB diagram is available inside the folder

2. A. How many active player in 2011-12?

```
SELECT COUNT(DISTINCT player_id)
```

```
FROM player_season_stats AS pss, season AS s
WHERE pss.season_id = s.season_id
AND s.season = '2011-12';
```

Answer: **478**

2. B. How may play in each position?

```
SELECT COUNT(DISTINCT pss.player_id) AS player,p.position pos,
p.description descript
FROM player_season_stats AS pss, position AS p, season AS s
WHERE pss.Pos = p.position_id
 AND s.season_id = pss.season_id
 AND s.season = '2011-12'
 GROUP BY p.position;
```

Answer:

Count	position	description
101	C	Center
98	PF	Power Forward
94	PG	Point Guard
95	SF	Small Forward
93	SG	Shooting Guard

The total count is 281 which is greater than the active players in 2011-12. This is because few players have played for more than a team in different positions on the same season 2011-12.

2.C. Average age, weight, experience and salary of players active in '2011-12'

```
SELECT AVG(pss.Age) as Age, AVG(pb.wt) as weight ,
 ROUND(avg(2012 - pb.from_year),1) AS experience, AVG(pssl.salary)
AS salary
FROM player_season_stats AS pss, season AS s, player_basic AS pb,
player_season_salary as pssl
WHERE pb.player_id = pss.player_id
 AND s.season_id = pss.season_id
 AND pssl.player_id = pss.player_id
 AND pssl.season_id = s.season_id
 AND s.season = '2011-12';
```

Answer:

avg_age	avg_weight	avg_experience	avg_salary
26.62	219.62	5	4228351.46

2.D Career salary for 2011-12

```
SELECT pb.name, SUM(pss.salary) AS average
FROM player_season_salary AS pss, player_basic AS pb, season AS s
WHERE pss.player_id = pb.player_id
  AND pss.season_id = s.season_id
  AND s.season ='2011-12'
GROUP BY pb.player_id ORDER BY pb.name ;
```

Results are stored in **2D_results.csv**

3.A Top 10% of best paid player in 2011-12

```
SELECT pb.name,PSS.salary FROM player_basic AS pb,
player_season_salary AS pss, season s2
WHERE pb.player_id = pss.player_id
  AND pss.season_id =s2.season_id
  AND s2.season = '2011-12'
ORDER BY salary Desc
LIMIT ((SELECT count(DISTINCT player_id) FROM player_season_stats
p, season s WHERE p.season_id = s.season_id AND
s.season='2011-12')*10/100);
```

The results are stored in **3A_results.csv**

3.B Top 10% of least paid player in 2011-12

```
SELECT pb.name,PSS.salary FROM player_basic AS pb,
player_season_salary AS pss, season s2
WHERE pb.player_id = pss.player_id
  AND pss.season_id =s2.season_id
  AND s2.season = '2011-12'
ORDER BY salary ASC
LIMIT ((SELECT count(DISTINCT player_id) FROM player_season_stats
p, season s WHERE p.season_id = s.season_id AND
s.season='2011-12')*10/100);
```

The results are stored in **3B_results.csv**

3. C Middle 50% players in 2011-12

```
SELECT pb.name,PSS.salary FROM player_basic AS pb,
player_season_salary AS pss, season s2
WHERE pb.player_id = pss.player_id
  AND pss.season_id =s2.season_id
  AND s2.season = '2011-12'
ORDER BY salary DESC
  LIMIT (SELECT count(DISTINCT player_id) FROM player_season_stats
```

```
p, season s WHERE p.season_id = s.season_id AND
s.season='2011-12')*50/100 OFFSET (SELECT count(DISTINCT
player_id) FROM player_season_stats p, season s WHERE p.season_id
= s.season_id AND s.season='2011-12')*25/100;
```

The results are stored in **3C_results.csv**

3. D. Over all seasons of the active players in the 2011-2012 season, how much money was paid to all users by season? How many players were active in each season What is the average per player by season.

```
SELECT s2.season, total, cnt, total/cnt average
FROM
  (SELECT pss.season_id seas, SUM(pss.salary) total,
count(players.player) cnt FROM
  (SELECT DISTINCT player_id AS player
FROM player_season_stats pss, season s
WHERE pss.season_id=s.season_id AND s.season='2011-12') AS
players, player_season_salary pss
WHERE players.player = pss.player_id
AND pss.season_id IN
          (SELECT DISTINCT(season_id) as seasons FROM
player_season_stats WHERE player_id IN (SELECT DISTINCT
     (pss.player_id) FROM player_season_stats pss, season s
    WHERE s.season='2011-12' AND s.season_id = pss.season_id))
GROUP BY pss.season_id) result, season s2
WHERE s2.season_id=result.seas
GROUP BY s2.season_id ORDER BY s2.season_id DESC
```

Results of this query are stored **3D_results.csv**

4. A. Average salary and variance

```
SELECT s.season seas, ROUND(AVG(pss.Age),2) avg_age
FROM player_season_stats pss,season s
WHERE s.season_id = pss.season_id
AND s.season_id BETWEEN 56 AND 66
GROUP BY s.season_id;
```

Results are stored in **4A_avg_sal.csv**

```
SELECT avg(pss.salary*pss.salary) -
avg(pss.salary)*avg(pss.salary) FROM player_season_salary pss
WHERE pss.season_id BETWEEN 56 AND 66;
```

Answer : Variance of salaries of all players between 2002 to 2012 is: **18346526884893.68.** This high variance indicates that the salaries of players are spread away from the mean salary which is 3972654.61.

The variance of average salary by teams are stored in **4A_variance.csv** . From the results it is found that the team “Los Angeles Lakers” which means the team has not payed same salary to players across various seasons.

4.B. Average age , Average experience and variance of experience

```
SELECT tb.name team,  s.season seas, ROUND(AVG(pss.Age),2) avg_age
FROM team_basic tb, player_season_stats pss,season s
  WHERE tb.team_id = pss.team_id
  AND tb.defunct = 0
  AND s.season_id = pss.season_id
  AND s.season_id BETWEEN 56 AND 66
  GROUP BY tb.team_id,s.season_id;
```

The results are stored in **4B_avg_age_results.csv**

```
SELECT tb.name, s.season,
ROUND(AVG(s.year - pb.from_year),2) as experience,
ROUND(AVG((s.year - pb.from_year)*(s.year -
pb.from_year))*AVG((s.year - pb.from_year)*(s.year -
pb.from_year)) - AVG(s.year - pb.from_year)*AVG(s.year -
pb.from_year),2) AS variance
FROM player_season_stats pss,team_basic tb, season s,player_basic
pb
WHERE pss.season_id = s.season_id
AND s.season_id BETWEEN 56 AND 66
AND tb.team_id = pss.team_id
AND pb.player_id = pss.player_id
GROUP BY tb.team_id, s.season_id
```

Note: The experience in the above query is the experience of the player in that season and not the overall career experience. The results are stored in **4B_exp_var.csv**

4.C Cross tabulation

```
SELECT DISTINCT t.team as TeamName,
   SUM(CASE WHEN t.seas='2011-12' THEN t.avg_sal END) '2011-12',
   SUM(CASE WHEN t.seas='2010-11' THEN t.avg_sal END) '2010-11',
   SUM(CASE WHEN t.seas='2009-10' THEN t.avg_sal END) '2009-10',
   SUM(CASE WHEN t.seas='2008-09' THEN t.avg_sal END) '2008-09',
   SUM(CASE WHEN t.seas='2007-08' THEN t.avg_sal END) '2007-08',
   SUM(CASE WHEN t.seas='2006-07' THEN t.avg_sal END) '2006-07',
   SUM(CASE WHEN t.seas='2005-06' THEN t.avg_sal END) '2005-06',
   SUM(CASE WHEN t.seas='2004-05' THEN t.avg_sal END) '2004-05',
   SUM(CASE WHEN t.seas='2003-04' THEN t.avg_sal END) '2003-04',
   SUM(CASE WHEN t.seas='2002-03' THEN t.avg_sal END) '2002-03',
   SUM(CASE WHEN t.seas='2001-02' THEN t.avg_sal END) '2001-02'
FROM
(SELECT tb.name team,  s.season seas, ROUND(AVG(pss.salary),2)
```

```
avg_sal FROM team_basic tb, player_season_salary pss,season s
WHERE tb.team_id = pss.team_id  AND tb.defunct = 0  AND
s.season_id = pss.season_id  AND s.season_id BETWEEN 56 AND 66
GROUP BY tb.team_id,s.season_id) AS t GROUP BY t.team;
```

The result of above query is stored in **4C_avg_sal.csv**

```
SELECT DISTINCT t.team as TeamName,
   SUM(CASE WHEN t.seas='2011-12' THEN t.experience END)
'2011-12',
   SUM(CASE WHEN t.seas='2010-11' THEN t.experience END)
'2010-11',
   SUM(CASE WHEN t.seas='2009-10' THEN t.experience END)
'2009-10',
   SUM(CASE WHEN t.seas='2008-09' THEN t.experience END)
'2008-09',
   SUM(CASE WHEN t.seas='2007-08' THEN t.experience END)
'2007-08',
   SUM(CASE WHEN t.seas='2006-07' THEN t.experience END)
'2006-07',
   SUM(CASE WHEN t.seas='2005-06' THEN t.experience END)
'2005-06',
   SUM(CASE WHEN t.seas='2004-05' THEN t.experience END)
'2004-05',
   SUM(CASE WHEN t.seas='2003-04' THEN t.experience END)
'2003-04',
   SUM(CASE WHEN t.seas='2002-03' THEN t.experience END)
'2002-03',
   SUM(CASE WHEN t.seas='2001-02' THEN t.experience END) '2001-02'
FROM
(SELECT tb.name team, s.season seas,
ROUND(AVG(s.year - pb.from_year),2) as experience,
ROUND(AVG((s.year - pb.from_year)*(s.year -
pb.from_year))*AVG((s.year - pb.from_year)*(s.year -
pb.from_year)) - AVG(s.year - pb.from_year)*AVG(s.year -
pb.from_year),2) AS variance
FROM player_season_stats pss,team_basic tb, season s,player_basic
pb
WHERE pss.season_id = s.season_id AND s.season_id BETWEEN 56 AND
66 AND tb.team_id = pss.team_id AND pb.player_id = pss.player_id
GROUP BY tb.team_id, s.season_id) AS t GROUP BY t.team;
```

The results of the above average experience cross tabulation is saved in **4C_avg_experience.csv**

```
SELECT DISTINCT t.team as TeamName,
   SUM(CASE WHEN t.seas='2011-12' THEN t.avg_age END) '2011-12',
   SUM(CASE WHEN t.seas='2010-11' THEN t.avg_age END) '2010-11',
   SUM(CASE WHEN t.seas='2009-10' THEN t.avg_age END) '2009-10',
   SUM(CASE WHEN t.seas='2008-09' THEN t.avg_age END) '2008-09',
```

```
   SUM(CASE WHEN t.seas='2007-08' THEN t.avg_age END) '2007-08',
   SUM(CASE WHEN t.seas='2006-07' THEN t.avg_age END) '2006-07',
   SUM(CASE WHEN t.seas='2005-06' THEN t.avg_age END) '2005-06',
   SUM(CASE WHEN t.seas='2004-05' THEN t.avg_age END) '2004-05',
   SUM(CASE WHEN t.seas='2003-04' THEN t.avg_age END) '2003-04',
   SUM(CASE WHEN t.seas='2002-03' THEN t.avg_age END) '2002-03',
   SUM(CASE WHEN t.seas='2001-02' THEN t.avg_age END) '2001-02'
FROM
(SELECT tb.name team,  s.season seas, ROUND(AVG(pss.Age),2)
avg_age FROM team_basic tb, player_season_stats pss,season s
  WHERE tb.team_id = pss.team_id
  AND tb.defunct = 0
  AND s.season_id = pss.season_id
  AND s.season_id BETWEEN 56 AND 66
  GROUP BY tb.team_id,s.season_id) AS t GROUP BY t.team;
```

The results are stored in **4C_avg_age_results.csv**

5. A) What other data explains salary?

- The player awards can be used to explain the high salary. A person with more awards are likely to get higher salary in the next season.
- The leaderboard data can be used to explain salary. Higher the rank of a player, higher is his salary.
- Also the fan follower of the page in social networking sites can be used to explain the salary. The twitter and Facebook profiles of each player is available in the website. This data can be used to compute the fan following and also do social media analytics to predict the salary

5. B) Recommendation for team owners:

- Use team statistics to analyze the strengths and weakness of team. This could be used in hiring the best players in future.
- Coach performance data can be used in hiring the coaches. Also the statistics of coaches when they were players in NBA could be used to analyze them and find their salary.
- Also after finding the weakness of the team with team and player statistics, the coach with relevant strength can be chosen so that the team performance can be improved and win share can be increased

5. C) High prices of players:

For example from the salary table it is found that Michael Jordan has got the highest salary. On analyzing his statistics he has got most field goals amongst the players played in last 20 seasons. Also he has got 6 NBA champion, 5 Most valuable player awards and 14 times in NBA all star. Hence the factors like performance (player statistics) and awards justify the high prices of the player.