In [1]:
```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
```

In [2]:
```
def pull_stats1(year):

    # Create url with year that was provided
    url = "https://www.basketball-reference.com/leagues/NBA_{}_totals.html#totals
    soup = BeautifulSoup(urlopen(url))

    # Create list with headers of the table
    headers = [th.getText() for th in soup.findAll('tr')[0].findAll('th')]

    # Remove the first header which is the index
    headers = headers[1:]

    # Identify the rows of the dataframe
    rows = soup.findAll('tr')[1:]

    # Create a list of stats for each player
    player_stats = [[td.getText() for td in rows[i].findAll('td')]
                for i in range(len(rows))]

    # Create a dataframe from the stats and headers
    stats = pd.DataFrame(player_stats, columns = headers)

    return stats
```

In [3]:
```
def pull_stats2(year):

    # Create url with year that was provided
    url = "https://www.basketball-reference.com/leagues/NBA_{}_advanced.html#adva
    soup = BeautifulSoup(urlopen(url))

    # Create list with headers of the table
    headers = [th.getText() for th in soup.findAll('tr')[0].findAll('th')]

    # Remove the first header which is the index
    headers = headers[1:]

    # Identify the rows of the dataframe
    rows = soup.findAll('tr')[1:]

    # Create a list of stats for each player
    player_stats = [[td.getText() for td in rows[i].findAll('td')]
                for i in range(len(rows))]

    # Create a dataframe from the stats and headers
    stats = pd.DataFrame(player_stats, columns = headers)

    # Add a column for the year
    stats['Year'] = year
    # Remove blank columns
    stats = stats.drop([stats.columns[18], stats.columns[23]], axis='columns')

    return stats
```

In [4]:
```
def pull_salary(year, pages=50):
    # Create empty list
    full = []

    # Loop through each page on the website
    for i in range(1,pages+1):

        # Create url
        url = "http://www.espn.com/nba/salaries/_/year/{}/page/{}".format(year, i

        soup = BeautifulSoup(urlopen(url))

        headers = [td.getText() for td in soup.findAll('tr')[0].findAll('td')]

        rows = soup.findAll('tr')[1:]

        player_salary = [[td.getText() for td in rows[i].findAll('td')]
                    for i in range(len(rows))]

        salary = pd.DataFrame(player_salary, columns = headers)
        salary['Year'] = year

        # Remove players position from the 'NAME' column
        for i in range(len(salary)):
            salary['NAME'][i] = salary['NAME'][i].split(',')[0]

        full.append(salary)

    # Turn the llist into a dataframe
    full = pd.concat(full)

    # Remove the repeated headers
    full.drop(full[full['RK']=='RK'].index, inplace=True)

    # Reset index
    full = full.reset_index().drop(columns='index')

    return full
```

In [5]:
```
def create_df(years):

    # Create empty dataframe
    final = pd.DataFrame()

    # Loop through list of years
    for year in years:
        temp_sta1 = pull_stats1(year)
        temp_sta2 = pull_stats2(year)
        temp_sal = pull_salary(year)

        # Create temp dataframe
        cols = list(temp_sta1.columns)[:-1]
        cols2 = list(list(temp_sta2.columns)[6:])
        cols.extend(cols2)
        cols.append('Salary')
        new = pd.DataFrame(columns=cols)

        # Loop through each player and record their index
        for i in range(len(temp_sta1)):
            player = temp_sta1['Player'][i]
            index_stat2 = temp_sta2[temp_sta2['Player']==player].index.values
            index_sal = temp_sal[temp_sal['NAME']==player].index.values

            # Check of the player is in all three data sets
            if index_stat2.size == 0 or index_sal.size == 0:
                continue
            else:

                # Combine the player data into one dataframe
                array = temp_sta1.iloc[i][:-1].append(temp_sta2.iloc[index_stat2
                df_temp = pd.DataFrame(array).T
                df_temp['Salary'] = temp_sal['SALARY'][index_sal].values[0]
                df_temp['Salary'] = df_temp['Salary'].replace('[\$,]', '', regex=
                new = new.append(df_temp)

        # Remove dupicate players that played for multiple teams
        new = new.loc[(new['Tm'] == 'TOT')|~new['Player'].duplicated()]

        # Append dataframe to the final dataframe
        final = final.append(new)

    # Remove rows with blanks
    final.replace('', np.nan, inplace=True)
    final.dropna(inplace=True)


    return final
```

In [6]:
```
df = create_df(range(2010,2020))
```

```
C:\Users\ande5\Anaconda3\lib\site-packages\ipykernel_launcher.py:25: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [263]:
```
df.to_csv('Full_scrape.csv', index = False)
```

```
In [10]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3638 entries, 0 to 727
Data columns (total 50 columns):
Player    3638 non-null object
Pos       3638 non-null object
Age       3638 non-null object
Tm        3638 non-null object
G         3638 non-null object
GS        3638 non-null object
MP        3638 non-null object
FG        3638 non-null object
FGA       3638 non-null object
FG%       3638 non-null object
3P        3638 non-null object
3PA       3638 non-null object
3P%       3638 non-null object
2P        3638 non-null object
2PA       3638 non-null object
2P%       3638 non-null object
eFG%      3638 non-null object
FT        3638 non-null object
FTA       3638 non-null object
FT%       3638 non-null object
ORB       3638 non-null object
DRB       3638 non-null object
TRB       3638 non-null object
AST       3638 non-null object
STL       3638 non-null object
BLK       3638 non-null object
TOV       3638 non-null object
PF        3638 non-null object
PER       3638 non-null object
TS%       3638 non-null object
3PAr      3638 non-null object
FTr       3638 non-null object
ORB%      3638 non-null object
DRB%      3638 non-null object
TRB%      3638 non-null object
AST%      3638 non-null object
STL%      3638 non-null object
BLK%      3638 non-null object
TOV%      3638 non-null object
USG%      3638 non-null object
OWS       3638 non-null object
DWS       3638 non-null object
WS        3638 non-null object
WS/48     3638 non-null object
OBPM      3638 non-null object
DBPM      3638 non-null object
BPM       3638 non-null object
VORP      3638 non-null object
Year      3638 non-null int64
Salary    3638 non-null float64
dtypes: float64(1), int64(1), object(48)
memory usage: 1.4+ MB
```

In [ ]:

```

```