Introduction:

Our team is a big fan of the Houston Rockets for National Basketball Association. We watched their games almost 10 seasons, Rockets still cannot bring back one championship trophy for Houston in the 21st century. We asked ourselves what wrong with this pro team, what else they could improve their line-ups. Daryl Morey, General Manager of the Houston Rockets, is renowned for analytical methods, having created the “true shooting percentage”. One idea comes to our mind.

Mission Statement:

Our goal is to apply detailed stats both from players and teams to analyze how the players’ performance related to the team’s ranking by making the grading system. We are trying to build grading system to rate every single NBA player and how every player’s performance related to team’s ranking. Moreover, we want to investigate on how the team payroll affects the team's ranking.

Dataset

There are no existing datasets for us to use, so in order to achieve our objective, we plan to find data on the website of basketball-reference.com and scrape the details we need from the website. There are four datasets including:

• The first part is based on players’ performance on the court, such as points, rebounds, assists, steals, blocks, turnovers, field goals, field goal attempt, field goal percentage, three-point shooting percentage, free throw percentage, games started, minutes played and etc.

• The second part is going to focus on the players' physical stats including their positions, height, weight, ages and career years.

• The third part contained each team's name and how many games they win/lose and their winning rate.

• The fourth part focuses on team payroll.

In [1]:
```
import pandas as pd
import numpy as np
pd.set_option('display.max_rows',200)
```

1. Players’ Performance Dataset:

1. Players' Performance Dataset:

We download the NBA players' performance dataset from the https://www.basketball-reference.com/leagues/NBA_2019_per_game.html (https://www.basketball-reference.com/leagues/NBA_2019_per_game.html). We clean the data first. We find there are lots of columns that might not help our analysis in the future, therefore, we ONLY keep these columns in the below:

• Player: player name

• Pos: position

• Age: age

• Tm: team

• G: game played

• GS: game started

• MP: minutes played

• FG%: field goal percentage

• 3P%: 3-point percentage

• 2P%: 2-point percentage

• FT%: free throw percentage

• TRB: total rebounds

• AST: assist

• STL: steal

• BLK: block

• TOV: turnover

• PF: personal fouls

• PTS: points

Next, we find in the dataset there are lots of meaningless rows, they just repeated tell the readers what all these attributes' names. We delete these unnecessary columns.

Moreover, we find some players appeared repeatedly in the dataset because these players might be traded during the season. For example, Trevor Ariza is a Small Forward in the Washington Wizards right now. But, in the first half season, he was played in the Phoenix Suns. The dataset separated his stats into two teams he

played, and also created the new row called ‘TOT’(total) in the team columns, which is the average stats for him in both teams. We decide to delete ‘TOT’ columns for these traded players since we are going to put every player into his team. There is no team called ‘TOT’.

Also, there are lots of missing data in the dataset. For instance, maybe the player hasn’t attempted free throws in this season, we decide to fill all NA to 0.

Finally, we create the new column named 'TMP' which means total minutes this player played in this season. We use the game he played multiply the average minutes he played per game.

In [2]:
```
# read tables from website
url='https://www.basketball-reference.com/leagues/NBA_2019_per_game.html'
df=pd.read_html(url, index_col=0)[0]
```

In [3]:
```
df.head(10)
```

Out[3]:

Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	FT%	ORB	DRB	TRB	A
1	Alex Abrines	SG	25	OKC	31	2	19.0	1.8	5.1	.357	...	.923	0.2	1.4	1.5	
2	Quincy Acy	PF	28	PHO	10	0	12.3	0.4	1.8	.222	...	.700	0.3	2.2	2.5	
3	Jaylen Adams	PG	22	ATL	34	1	12.6	1.1	3.2	.345	...	.778	0.3	1.4	1.8	
4	Steven Adams	C	25	OKC	80	80	33.4	6.0	10.1	.595	...	.500	4.9	4.6	9.5	
5	Bam Adebayo	C	21	MIA	82	28	23.3	3.4	5.9	.576	...	.735	2.0	5.3	7.3	
6	Deng Adel	SF	21	CLE	19	3	10.2	0.6	1.9	.306	...	1.000	0.2	0.8	1.0	
7	DeVaughn Akoon-Purcell	SG	25	DEN	7	0	3.1	0.4	1.4	.300	...	.500	0.1	0.4	0.6	
8	LaMarcus Aldridge	C	33	SAS	81	81	33.2	8.4	16.3	.519	...	.847	3.1	6.1	9.2	
9	Rawle Alkins	SG	21	CHI	10	1	12.0	1.3	3.9	.333	...	.667	1.1	1.5	2.6	
10	Grayson Allen	SG	23	UTA	38	2	10.9	1.8	4.7	.376	...	.750	0.1	0.5	0.6	

10 rows × 29 columns

In [4]:
```
#drop meaningless columns
df=df.drop(['GS','FG','FGA','3P','3PA','2P','2PA','eFG%','FT','FTA','ORB','DRB'],axis=1)
#drop abnormal statistics
df=df.drop('Rk',axis=0)
#drop some total statistics
df=df[df['Tm']!='TOT']
#fill empty cell with 0
df=df.fillna(0)
```

In [5]:
```
#create new column called TMP(TOTAL MINUTES PLAYED)
df['TMP']=df['G'].astype(float)*df['MP'].astype(float)
```

2. Players' Physical Stats Dataset:

We fail to find the dataset that includes all the players' physical stats in the league in a whole dataset. Instead, we could find players' physical stats grouped by each team, and create the for loop to gather every team's stats in one big dataset. We keep the columns named:

• Player: player name

• Ht: height

• Wt: weight

• Exp: # of career years

Then, we apply the regular expression to clean the data for some tedious players' names. for instance, if a player's name appeared with (TW), it means he is signed to a two-way contract with G-league affiliate. We want to remove (TW) behind the player's name since we need to make sure that names in dataset 1 have to be the same in dataset 2, which would help us to merge in the further step. Also, we clean some suffix in order to make these names coherent.

Also, in the 'EXP' columns, there are some career years named 'R' which means Rookie year. We change it to 0 in order to keep all the values in numerical types.

Next, we merge the dataset 1 and dataset 2 together with the key columns 'player'.

We filter out those players who played less than 5 mins per game since we think those fringe players cannot significantly influence the team's performance.

In [6]:
```
#read each teams' tables
urlleft='https://www.basketball-reference.com/teams/'
urlright='/2019.html'
teamname=['ATL','BRK','BOS','CHO','CHI','CLE','DAL','DEN','DET','GSW',
'HOU','IND','LAC','LAL','MEM','MIA','MIL','MIN','NOP','NYK','OKC','ORL
','PHI','PHO','POR','SAC','SAS','TOR','UTA','WAS']
df_bios=pd.DataFrame()
for name in teamname:
    urlfull=urlleft+name+urlright
    df_team=pd.read_html(urlfull)[0]
    df_bios=df_bios.append(df_team)
```

In [7]:
```
#set new index
df_bios['Index']=range(1,df_bios['Player'].count()+1)
df_bios=df_bios.set_index('Index')
#drop meaningless columns
df_bios=df_bios[['Player','Ht','Wt','Exp']]
# normalize names
df_bios['Player']=df_bios['Player'].str.replace('( \DTW\D)','')
df_bios['Player']=df_bios['Player'].str.replace('( III)','')
df_bios['Player']=df_bios['Player'].str.replace('( Jr\.)','')
df_bios['Player']=df_bios['Player'].str.replace('(Nene)','Nene Hilario
')
df_bios.head()
```

Out[7]:

Index	Player	Ht	Wt	Exp
1	Justin Anderson	6-6	230	3
2	Kent Bazemore	6-5	201	6
3	Taurean Prince	6-8	220	2
4	DeAndre' Bembry	6-6	210	2
5	Miles Plumlee	6-11	249	6

In [8]:
```
# normalize names
df_bios['Player']=df_bios['Player'].str.replace('( \DTW\D)','')
df_bios['Player']=df_bios['Player'].str.replace('( III)','')
df_bios['Player']=df_bios['Player'].str.replace('( Jr\.)','')
df_bios['Player']=df_bios['Player'].str.replace('(Nene)','Nene Hilario
')
df_bios.head()
```

Out[8]:

Index	Player	Ht	Wt	Exp
1	Justin Anderson	6-6	230	3
2	Kent Bazemore	6-5	201	6
3	Taurean Prince	6-8	220	2
4	DeAndre' Bembry	6-6	210	2
5	Miles Plumlee	6-11	249	6

In [9]:
```
#merge to dataset
df_full=pd.merge(df, df_bios, on='Player',how='inner')
#change 'R' in Exp to 0
df_full['Exp']=df_full['Exp'].str.replace('R','0')
df_full.head()
```

Out[9]:

	Player	Pos	Age	Tm	G	MP	FG%	3P%	2P%	FT%	...	AST	STL	BLK	TOV	F
0	Jaylen Adams	PG	22	ATL	34	12.6	.345	.338	.361	.778	...	1.9	0.4	0.1	0.8	1
1	Steven Adams	C	25	OKC	80	33.4	.595	.000	.596	.500	...	1.6	1.5	1.0	1.7	2
2	Bam Adebayo	C	21	MIA	82	23.3	.576	.200	.588	.735	...	2.2	0.9	0.8	1.5	2
3	Deng Adel	SF	21	CLE	19	10.2	.306	.261	.385	1.000	...	0.3	0.1	0.2	0.3	0
4	LaMarcus Aldridge	C	33	SAS	81	33.2	.519	.238	.528	.847	...	2.4	0.5	1.3	1.8	2

5 rows × 21 columns

New Variable 'BMI'

What are the perfect height and weight for those athletes? We keep asking ourselves, since on the court, we need 5 different positions, SG,PG,SF,PF and C. each position requires different heights and weights. We search it online and find there is a term called BMI(Body Mass Index), which is the value to estimate your physical fitness.

The formula of BMI = weight(kg) / height(m)**2.

We change the unit from lb to kg, inch/feet to m.

According to Jonathan Kinnick, who is a CF-L3 Trainer and lecturer in Economics at Cal Poly Pomona, he analyzed how the relationship between Avg. Fitness Performance and BMI for male. He concluded that

• for people with 18-20 BMI scored 34.9 fitness point;(out of 65)

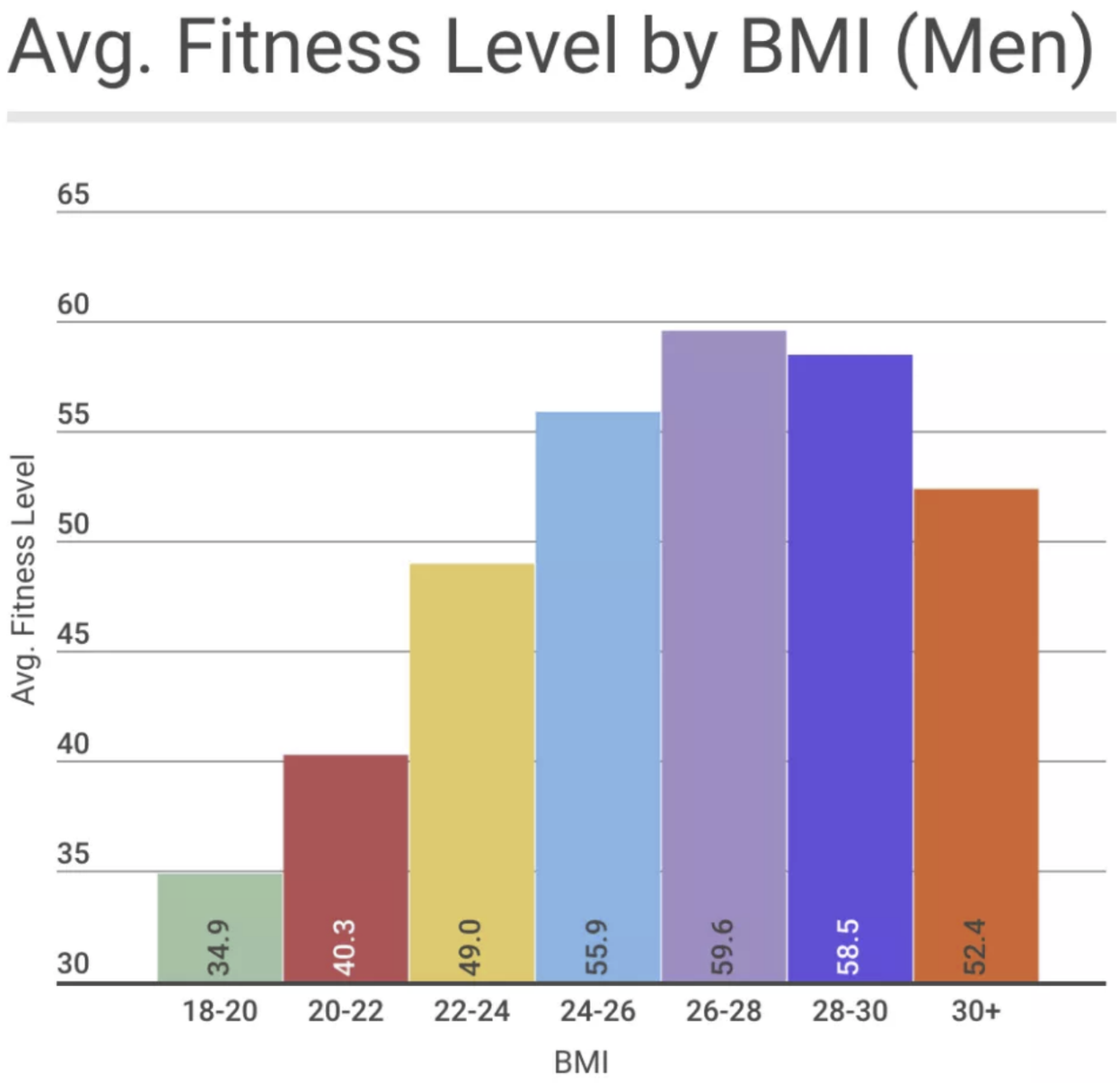
• for people with 20-22 BMI scored 40.3 fitness point;

• for people with 22-24 BMI scored 49.0 fitness point.

• for people with 24-26 BMI scored 55.9 fitness point.

• for people with 26-28 BMI scored 59.6 fitness point.

• for people with 28-30 BMI scored 58.5 fitness point.



We put these players into different BMI groups based on Mr.Kinnick's investigation, and create the new column called 'BMI Score Percentage'. For instance, for those players with 18-20 BMI, their BMI Score Percentage is 34.9/65 =53.69%.

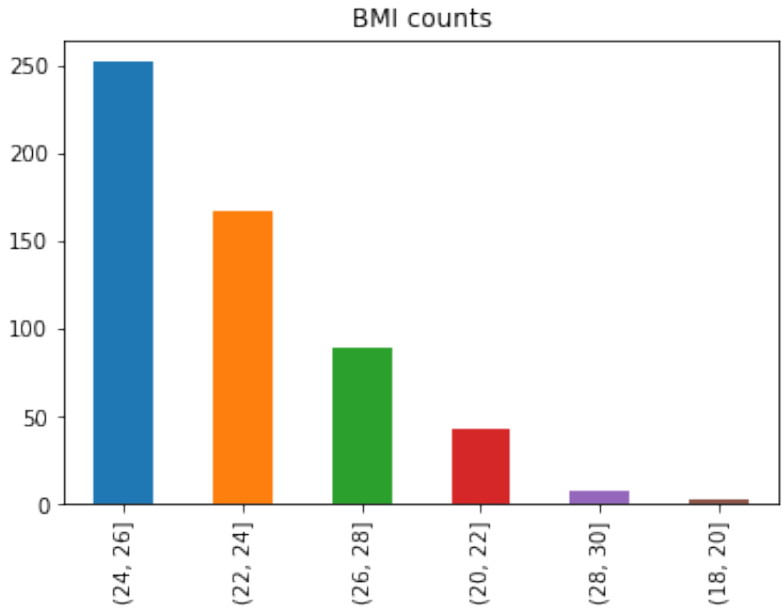
In [10]:
```
# transform lbs to kg, feet to m
ser1=df_full['Ht'].str.split('-').str[0].astype(float)
ser2=df_full['Ht'].str.split('-').str[1].astype(float)
ser3=df_full['Wt'].astype(float)
df_bmi=pd.DataFrame({'feet': ser1*30.48, 'inch': ser2*2.54,'kg':ser3*0
.45359237})
df_bmi['m']=(df_bmi['feet']+df_bmi['inch'])/100
df_bmi=df_bmi.drop(['feet','inch'],axis=1)
df_bmi.head()
```

Out[10]:

	kg	m
0	86.182550	1.8796
1	120.201978	2.1336
2	115.666054	2.0828
3	90.718474	2.0066
4	117.934016	2.1082

In [11]:
```
#change the abbreviation term to full-name
df_full.loc[df_full['Tm']=='ATL','Tm']='Atlanta Hawks'
df_full.loc[df_full['Tm']=='OKC','Tm']='Oklahoma City Thunder'
df_full.loc[df_full['Tm']=='MIA','Tm']='Miami Heat'
df_full.loc[df_full['Tm']=='CLE','Tm']='Cleveland Cavaliers'
df_full.loc[df_full['Tm']=='SAS','Tm']='San Antonio Spurs'
df_full.loc[df_full['Tm']=='HOU','Tm']='Houston Rockets'
df_full.loc[df_full['Tm']=='LAL','Tm']='Los Angeles Lakers'
df_full.loc[df_full['Tm']=='MIL','Tm']='Milwaukee Bucks'
df_full.loc[df_full['Tm']=='UTA','Tm']='Utah Jazz'
df_full.loc[df_full['Tm']=='PHO','Tm']='Phoenix Suns'
df_full.loc[df_full['Tm']=='DET','Tm']='Detroit Pistons'
df_full.loc[df_full['Tm']=='GSW','Tm']='Golden State Warriors'
df_full.loc[df_full['Tm']=='POR','Tm']='Portland Trail Blazers'
df_full.loc[df_full['Tm']=='NOP','Tm']='New Orleans Pelicans'
df_full.loc[df_full['Tm']=='BOS','Tm']='Boston Celtics'
df_full.loc[df_full['Tm']=='WAS','Tm']='Washington Wizards'
df_full.loc[df_full['Tm']=='CHO','Tm']='Charlotte Hornets'
df_full.loc[df_full['Tm']=='MEM','Tm']='Memphis Grizzlies'
df_full.loc[df_full['Tm']=='DAL','Tm']='Dallas Mavericks'
df_full.loc[df_full['Tm']=='IND','Tm']='Indiana Pacers'
df_full.loc[df_full['Tm']=='BRK','Tm']='Brooklyn Nets'
df_full.loc[df_full['Tm']=='CHI','Tm']='Chicago Bulls'
df_full.loc[df_full['Tm']=='NYK','Tm']='New York Knicks'
df_full.loc[df_full['Tm']=='SAC','Tm']='Sacramento Kings'
df_full.loc[df_full['Tm']=='ORL','Tm']='Orlando Magic'
df_full.loc[df_full['Tm']=='DEN','Tm']='Denver Nuggets'
df_full.loc[df_full['Tm']=='MIN','Tm']='Minnesota Timberwolves'
df_full.loc[df_full['Tm']=='PHI','Tm']='Philadelphia 76ers'
df_full.loc[df_full['Tm']=='TOR','Tm']='Toronto Raptors'
df_full.loc[df_full['Tm']=='LAC','Tm']='Los Angeles Clippers'
#merge two tables
df_full['kg']=df_bmi['kg']
df_full['m']=df_bmi['m']
#calculate BMI
df_full['BMI']=df_full['kg']/df_full['m']**2
#drop meaningless columns
df_full=df_full.drop(['Ht','Wt','kg','m'],axis=1)
# cut into bins and plot
bins=[18,20,22,24,26,28,30]
df_full['Binned']=pd.cut(df_full['BMI'],bins)
bin_number=df_full['Binned'].value_counts()
```

In [29]:
```
pd.Series(bin_number).plot(kind='bar',title='BMI counts');
```



In [13]:
```
#add column
df_full['BMI score percentage']=0
df_full['BMI'].astype(float)
#calculate BMI score
df_full.loc[(df_full['BMI']>26) & (df_full['BMI']<28),'BMI score perce
ntage']=59.6/65
df_full.loc[(df_full['BMI']>22) & (df_full['BMI']<24),'BMI score perce
ntage']=49.0/65
df_full.loc[(df_full['BMI']>20) & (df_full['BMI']<22),'BMI score perce
ntage']=40.3/65
df_full.loc[(df_full['BMI']>24) & (df_full['BMI']<26),'BMI score perce
ntage']=55.9/65
df_full.loc[(df_full['BMI']>28) & (df_full['BMI']<30),'BMI score perce
ntage']=58.5/65
df_full.loc[(df_full['BMI']>18) & (df_full['BMI']<20),'BMI score perce
ntage']=34.9/65
df_full.head()
```

Out[13]:

	Player	Pos	Age	Tm	G	MP	FG%	3P%	2P%	FT%	...	STL	BLK	TOV	PI
0	Jaylen Adams	PG	22	Atlanta Hawks	34	12.6	.345	.338	.361	.778	...	0.4	0.1	0.8	1.3
1	Steven Adams	C	25	Oklahoma City Thunder	80	33.4	.595	.000	.596	.500	...	1.5	1.0	1.7	2.(
2	Bam Adebayo	C	21	Miami Heat	82	23.3	.576	.200	.588	.735	...	0.9	0.8	1.5	2.5
3	Deng Adel	SF	21	Cleveland Cavaliers	19	10.2	.306	.261	.385	1.000	...	0.1	0.2	0.3	0.7
4	LaMarcus Aldridge	C	33	San Antonio Spurs	81	33.2	.519	.238	.528	.847	...	0.5	1.3	1.8	2.2

5 rows × 22 columns

Model (Player)

We apply the stats in the above to predict individual scores through a linear regression model. We do not use all the variables in the dataset. For instance, we think the FG% and 2-PT&3-PT FG% are multicollinearities, therefore, we only include the FG% in our regression model.

Based on our over 10 years watching games experience, we designed this linear regression model. We modified it several times by changing different parameters and including different variables. Next, we compared it to 2K19 to see if it make sense. Finally, we decide to include the variable and coefficients in the below:

Score= 5FG% + 10FT% + 3TRB + 2AST + 3STL + 2BLK - 3TOV + 2PTS + 2EXP + 10BMI SCORE PERCENTAGE

After that, by observation we found that the range for our score system is so large, therefore, we decide to apply the normalization on it. The method we use is min-max normalization and limit it in the range of [60,100].

In [14]:
```
#predict final score
df_full['Final score']=df_full['FG%'].astype(float)*5+df_full['FT%'].a
stype(float)*10+df_full['TRB'].astype(float)*3+df_full['AST'].astype(f
loat)*2+df_full['STL'].astype(float)*3+df_full['BLK'].astype(float)*2+
df_full['TOV'].astype(float)*(-3)+df_full['PTS'].astype(float)*2+df_fu
ll['Exp'].astype(float)*2+df_full['BMI score percentage'].astype(float
)*10
df_full.head()
```

Out[14]:

	Player	Pos	Age	Tm	G	MP	FG%	3P%	2P%	FT%	...	BLK	TOV	PF	PT
0	Jaylen Adams	PG	22	Atlanta Hawks	34	12.6	.345	.338	.361	.778	...	0.1	0.8	1.3	3.
1	Steven Adams	C	25	Oklahoma City Thunder	80	33.4	.595	.000	.596	.500	...	1.0	1.7	2.6	13.
2	Bam Adebayo	C	21	Miami Heat	82	23.3	.576	.200	.588	.735	...	0.8	1.5	2.5	8.
3	Deng Adel	SF	21	Cleveland Cavaliers	19	10.2	.306	.261	.385	1.000	...	0.2	0.3	0.7	1.
4	LaMarcus Aldridge	C	33	San Antonio Spurs	81	33.2	.519	.238	.528	.847	...	1.3	1.8	2.2	21.

5 rows × 23 columns

In [15]:
```
df_full['Final score']=((df_full['Final score']-df_full['Final score']
.min())*(100-60))/(df_full['Final score'].max()-df_full['Final score']
.min())+60
df_full['Final score']=df_full['Final score'].astype(int)
#sort from highest score to lowest
df_full.sort_values(by='Final score',ascending=False).head()
```

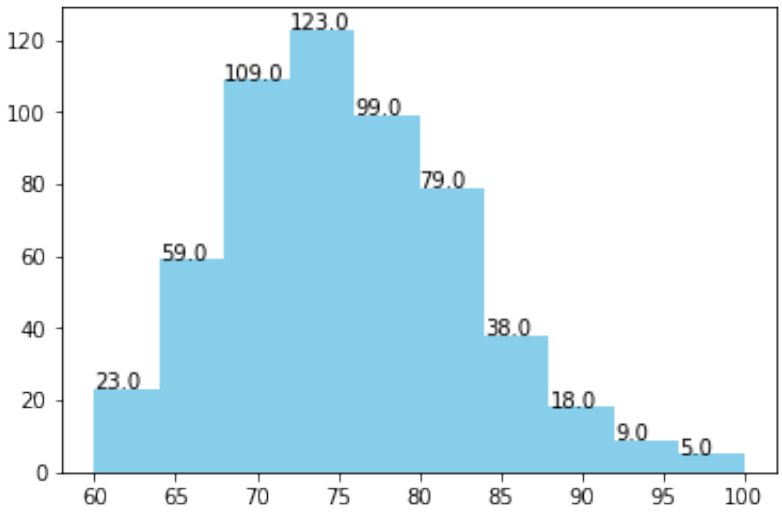
Out[15]:

	Player	Pos	Age	Tm	G	MP	FG%	3P%	2P%	FT%	...	BLK	TOV
269	LeBron James	SF	34	Los Angeles Lakers	55	35.2	.510	.339	.582	.665	...	0.6	3.6
217	James Harden	PG	29	Houston Rockets	78	36.8	.442	.368	.528	.879	...	0.7	5.0
535	Russell Westbrook	PG	30	Oklahoma City Thunder	73	36.0	.428	.290	.481	.656	...	0.5	4.5
128	Anthony Davis	C	25	New Orleans Pelicans	56	33.0	.517	.331	.547	.794	...	2.4	2.0
14	Giannis Antetokounmpo	PF	24	Milwaukee Bucks	72	32.8	.578	.256	.641	.729	...	1.5	3.7

5 rows × 23 columns

In [16]:
```
%pylab inline
import seaborn as sns
fig, ax = plt.subplots()
ax.hist(df_full['Final score'], color='skyblue')
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() , p.get_height()))
```

```
Populating the interactive namespace from numpy and matplotlib
```



This is the histogram that shows us the distribution of overall each player's score. We can see the top set includes only 5 players in the score 95-100. Most of the players are centered in the score between 75-80.

New Variable 'Contribution'!

In this step, our goal is to find the team score based on our player grading system.

The bottleneck we face is in each team dataset, there are much more numbers of players than the offical 15 players' list. The reason is the team dataset includes all the players even if they only played one game for that team. For instance, some players were traded in the middle of the season, they are still in the previous team dataset and also their current team dataset. We cannot just use the average player score who belonged to that team to represent the team performance.

Instead, we come up with a better way. Each player should have different weights in that team, for instance, a 10-year all-star player and one fringe player should not weight in the same contribution for that team. Therefore, we create the new variable called contribution(include those who contributed only in one game in this season for this team). And the contribution is based on the individual grading system we designed in the last step multiply the total minutes played in that team.

In [17]:
```
df_full['Contribution']=df_full["Final score"]*df_full['TMP']
df_full.head()
```

Out[17]:

	Player	Pos	Age	Tm	G	MP	FG%	3P%	2P%	FT%	...	TOV	PF	PTS	T
0	Jaylen Adams	PG	22	Atlanta Hawks	34	12.6	.345	.338	.361	.778	...	0.8	1.3	3.2	42
1	Steven Adams	C	25	Oklahoma City Thunder	80	33.4	.595	.000	.596	.500	...	1.7	2.6	13.9	267
2	Bam Adebayo	C	21	Miami Heat	82	23.3	.576	.200	.588	.735	...	1.5	2.5	8.9	191
3	Deng Adel	SF	21	Cleveland Cavaliers	19	10.2	.306	.261	.385	1.000	...	0.3	0.7	1.7	19
4	LaMarcus Aldridge	C	33	San Antonio Spurs	81	33.2	.519	.238	.528	.847	...	1.8	2.2	21.3	268

5 rows × 24 columns

3. Team Winning Rate Dataset.

In this step, what we want is the table that contained each team's name and how many games they win/lose and their winning rate. We also retrieve the data from the basketball-reference.com website. We dropped some unnecessary columns and apply the regular expression to clean the data. Since the original data behind the team name they marked the star signs for team that qualified for the playoffs and also named the rank in their conference.

In [18]:
```
#clean the data
url_team='https://www.basketball-reference.com/leagues/NBA_2019.html'
df_team_E=pd.read_html(url_team)[0]
df_team_E=df_team_E.drop(['GB','PS/G','PA/G','SRS'],axis=1)
df_team_E.columns=['Conference','W','L','W/L%']
df_team_W=pd.read_html(url_team)[1]
df_team_W=df_team_W.drop(['GB','PS/G','PA/G','SRS'],axis=1)
df_team_W.columns=['Conference','W','L','W/L%']
df_team=df_team_E.append(df_team_W)
#use regular expression to clean the name of each team
df_team['Team']=df_team['Conference'].str.extract('(.*?)[*]?\s[(]\d+[)
]')
df_team=df_team.reset_index()[['Conference','W','L','W/L%','Team']]
df_team.head()
```

Out[18]:

	Conference	W	L	W/L%	Team
0	Milwaukee Bucks* (1)	60	22	0.732	Milwaukee Bucks
1	Toronto Raptors* (2)	58	24	0.707	Toronto Raptors
2	Philadelphia 76ers* (3)	51	31	0.622	Philadelphia 76ers
3	Boston Celtics* (4)	49	33	0.598	Boston Celtics
4	Indiana Pacers* (5)	48	34	0.585	Indiana Pacers

Convert the Individual Contribution to the Team Contribution

We want to use the 'contribution' column we create before to help us get the 'Total Team Contribution' which is the team score. We want to sum up the individuals' contribution which belonged to the same team together.

Then we find the values in 'Total Team Contribution' are too large. Therefore, we used the min-max normalization again and limit it in the range of [60,100].

Last, we add the team score it to the winning rate dataset to give us the better views what the relationship between the team score we designed and the actual team winning rate this season.

In [19]:
```
#add new column in the winning rate dataset.
df_team['Total Team Contribution']=0
df_team=df_team[['Team','W','L','W/L%','Total Team Contribution']]
for index, row in df_team.iterrows():
    team_value=df_full[df_full['Tm']==row['Team']]['Contribution'].sum
()
    df_team.at[index,'Total Team Contribution'] = team_value
df_team.sort_values(by='Total Team Contribution',ascending=False).head
()
```

Out[19]:

	Team	W	L	W/L%	Total Team Contribution
15	Golden State Warriors	57	25	0.695	1608421
21	San Antonio Spurs	48	34	0.585	1597744
2	Philadelphia 76ers	51	31	0.622	1587231
0	Milwaukee Bucks	60	22	0.732	1586734
20	Oklahoma City Thunder	49	33	0.598	1574322

In [20]:
```
df_team=df_team.set_index('Team')
df_team['Total Team Contribution']=((df_team['Total Team Contribution'
]-df_team['Total Team Contribution'].min())*(100-60))/(df_team['Total
Team Contribution'].max()-df_team['Total Team Contribution'].min())+60
df_team['Total Team Contribution']=df_team['Total Team Contribution'].
astype(int)
df_team.head()
```

Out[20]:

Team	W	L	W/L%	Total Team Contribution
Milwaukee Bucks	60	22	0.732	95
Toronto Raptors	58	24	0.707	93
Philadelphia 76ers	51	31	0.622	96
Boston Celtics	49	33	0.598	92
Indiana Pacers	48	34	0.585	90

4. Team Payroll Dataset

We also retrieve the data from the basketball-reference.com website. We dropped some unnecessary columns and rows such as individual player's salaries, future's payroll, types of contracted players signed and the amount of guaranteed salaries.

Next, we merge it to the previous winning rate dataset which also contained the team score. And this is our final table. It contained all the data we need. We will do some visualizations based on this table in the next few steps.

In [21]:
```
#retreive the payroll dataset and drop unnecessary columns
dict_salaries={}
for i in teamname:
    urlbbb='https://www.basketball-reference.com/contracts/'+i+'.html'
    df_bbb=pd.read_html(urlbbb)[0]
    bbb=df_bbb.iloc[len(df_bbb)-1,2]
    dict_salaries.update({i:bbb})
df_salaries=pd.DataFrame.from_dict(dict_salaries,orient='index')
df_salaries.head()
```

Out[21]:

	0
ATL	$107,748,156
BRK	$118,267,134
BOS	$125,524,849
CHO	$122,327,044
CHI	$112,799,337

In [22]:
```
#change the abbrevation to the full name in order to help us merge in
next step
df_salaries.rename(index={'ATL':'Atlanta Hawks','OKC':'Oklahoma City T
hunder','MIA':'Miami Heat','CLE':'Cleveland Cavaliers','SAS':'San Anto
nio Spurs','HOU':'Houston Rockets','LAL':'Los Angeles Lakers','MIL':'M
ilwaukee Bucks','UTA':'Utah Jazz','PHO':'Phoenix Suns','DET':'Detroit
Pistons','GSW':'Golden State Warriors','POR':'Portland Trail Blazers',
'NOP':'New Orleans Pelicans','BOS':'Boston Celtics','WAS':'Washington
Wizards','CHO':'Charlotte Hornets','MEM':'Memphis Grizzlies','DAL':'Da
llas Mavericks','IND':'Indiana Pacers','BRK':'Brooklyn Nets','CHI':'Ch
icago Bulls','NYK':'New York Knicks','SAC':'Sacramento Kings','ORL':'O
rlando Magic','DEN':'Denver Nuggets','MIN':'Minnesota Timberwolves','P
HI':'Philadelphia 76ers','TOR':'Toronto Raptors','LAC':'Los Angeles Cl
ippers'},inplace=True)
df_salaries.columns=['Salaries']
df_salaries.head()
```

Out[22]:

	Salaries
Atlanta Hawks	$107,748,156
Brooklyn Nets	$118,267,134
Boston Celtics	$125,524,849
Charlotte Hornets	$122,327,044
Chicago Bulls	$112,799,337

In [23]:
```
#merge to dataset. This is the final dataset we want.
df_team=df_team.join(df_salaries)
df_team.head()
```

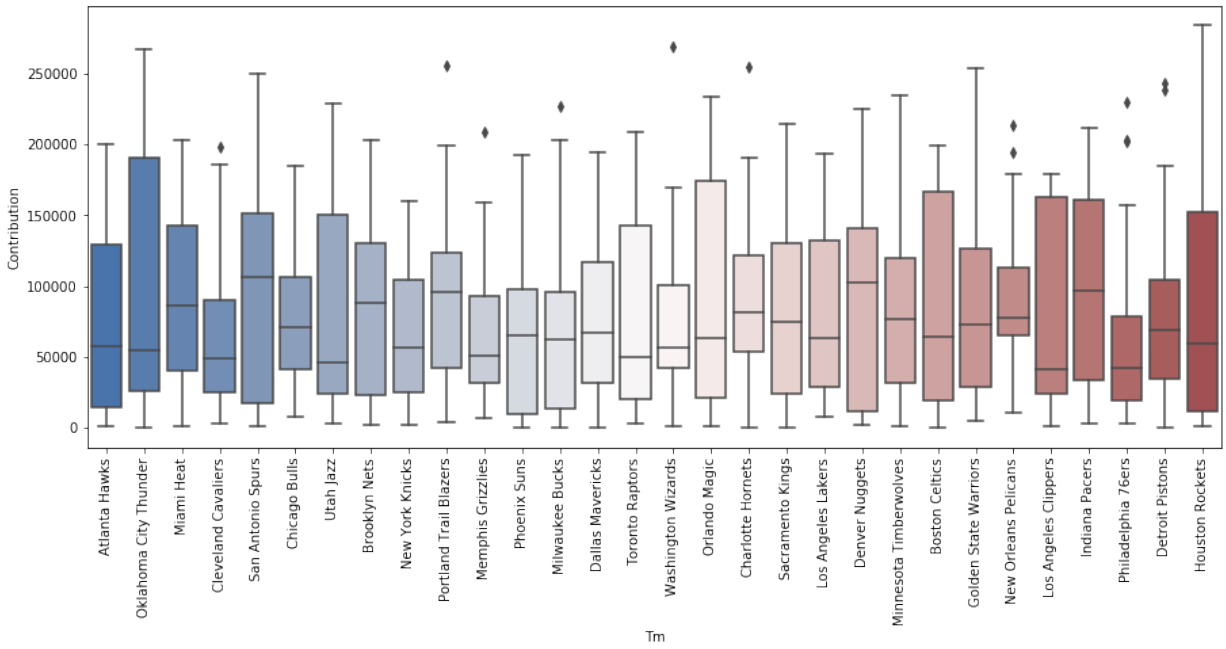
Out[23]:

Team	W	L	W/L%	Total Team Contribution	Salaries
Milwaukee Bucks	60	22	0.732	95	$125,884,880
Toronto Raptors	58	24	0.707	93	$136,224,561
Philadelphia 76ers	51	31	0.622	96	$112,883,127
Boston Celtics	49	33	0.598	92	$125,524,849
Indiana Pacers	48	34	0.585	90	$110,804,431

More Visualizations and Analysis

In [24]:
```
plt.figure(figsize=(15,6))
ax=sns.boxplot(x='Tm', y='Contribution', data=df_full, palette='vlag')
ax.set_xticklabels(df_full['Tm'].unique(),rotation=90)
ax
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x107608080>



The boxplot in the above that shows us each player's contributions in each team. San Antonio Spurs and Denver Nuggets have the highest average player contribution. the contribution gap in the Houston Rockets is the largest.

In [25]:
```
fig, ax = plt.subplots(5, 6, sharex=True, sharey=True, figsize=(20,20)
)
ax = ax.flatten()
for i, team in zip(range(len(df_full['Tm'].unique())),df_full['Tm'].un
ique()):
    ax[i].hist(df_full[df_full['Tm']==team]['Contribution'])
    ax[i].set_xticklabels(df_full[df_full['Tm']==team]['Contribution']
,rotation=90,fontsize=5)
    ax[i].set_title(team)
fig.tight_layout()
```


Atlanta Hawks
Oklahoma City Thunder
Miami Heat
Cleveland Cavaliers
San Antonio Spurs
Chicago Bulls
Utah Jazz
Brooklyn Nets
New York Knicks
Portland Trail Blazers
Memphis Grizzlies
Phoenix Suns
Milwaukee Bucks
Dallas Mavericks
Toronto Raptors
Washington Wizards
Orlando Magic
Charlotte Hornets
Sacramento Kings
Los Angeles Lakers
Denver Nuggets
Minnesota Timberwolves
Boston Celtics
Golden State Warriors
New Orleans Pelicans
Los Angeles Clippers
Indiana Pacers
Philadelphia 76ers
Detroit Pistons
Houston Rockets

These are 30 graphs that shows us how each player's distribution in each team. The y-axis represents numbers of players in the range of contribution(We cannot label y-axis in the table, it will be too tiny to read). Let us first analyze the Golden State Warriors, which lead the total team contribution in our grading system. Their distribution tells us the reason it is high because they have 3 players that contributed too much to the team. The remaining part of the team is nearly the same as the other teams. But these top 3 players helped bump up the average team contributions.

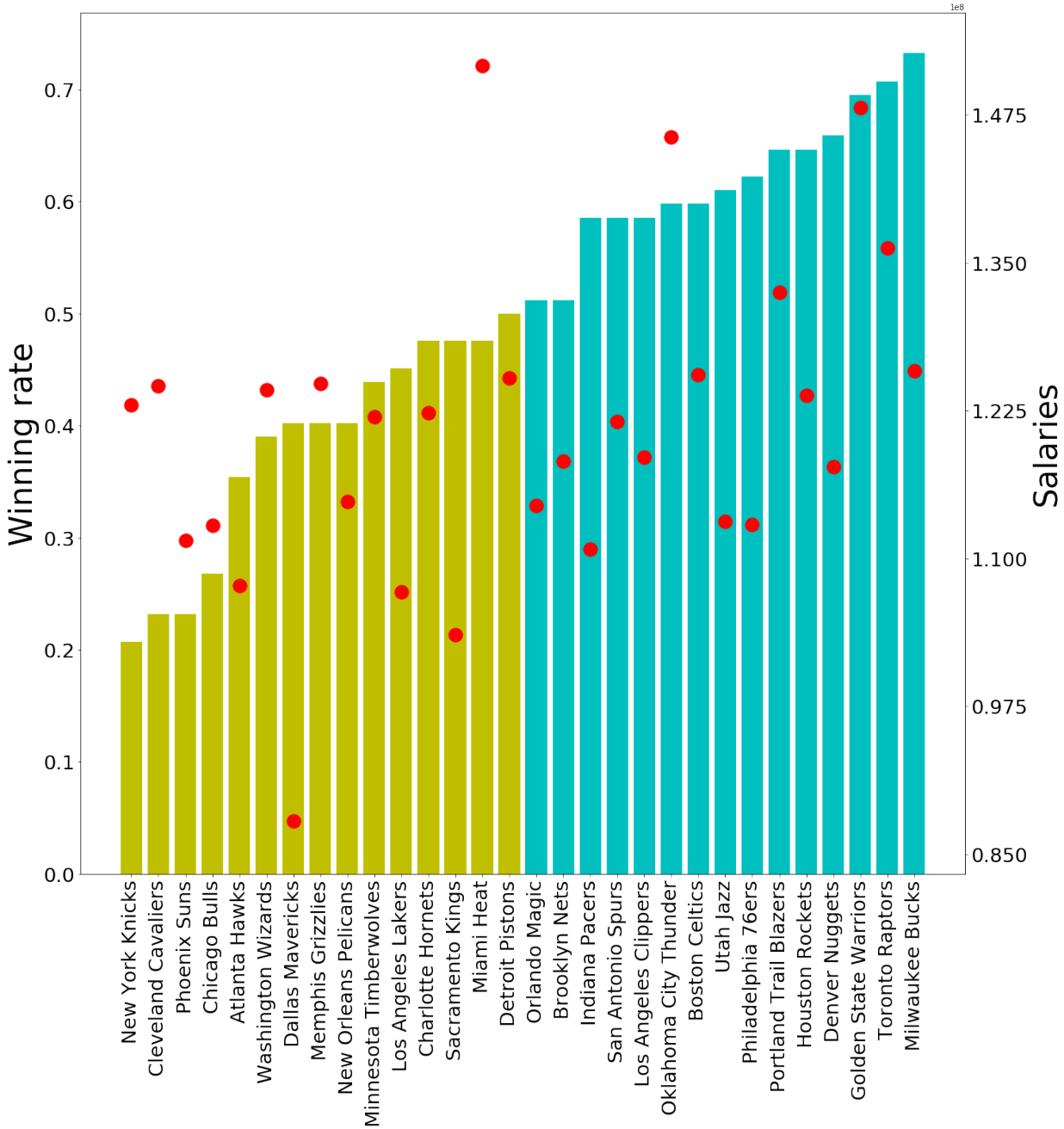
Next, we observe the contribution distributions in the team New York Knicks, Phoneix Suns ,and Cleveland Cavaliers, the reason they were in the bottom in the league because they do not have those players whose contribution could boost the whole team. That does not mean they do not have any star players. Remember, our contribution formula is based on the minutes played and the performance. Some of them may have the good performance on the court, but they may have some injuries during the season or have been traded, which could limit the time they played for that team, that is the main reason why some star players do not have as high contribution as we thought.

In [26]:
```
df_team['Salaries']=df_team['Salaries'].str.extract('[$](.*)')
df_team['Salaries']=df_team['Salaries'].str.replace(',','')
df_team['Salaries']=df_team['Salaries'].astype(int)
```

In [27]:
```
%pylab inline
df_team=df_team.sort_values(by='W/L%')
fig,axes=plt.subplots(figsize=(20,20))
my_colors = 'yyyyyyyyyyyyyyyycccccccccccccccc'
axes.bar(df_team.index,df_team['W/L%'],color=my_colors)
axes.set_xticklabels(df_team.index,rotation=90,fontsize=25)
axes.tick_params(axis='y', which='major', labelsize=25)
axes.set_ylabel('Winning rate', fontsize=40)
axes2=axes.twinx()
axes2.set_yticks(np.arange(85000000,155000000,12500000))
axes2.scatter(df_team.index,df_team['Salaries'],color='red',s=300)
axes2.tick_params(axis='y', which='major', labelsize=25)
axes2.set_ylabel('Salaries', fontsize=40)
```

```
Populating the interactive namespace from numpy and matplotlib
```

Out[27]: Text(0, 0.5, 'Salaries')

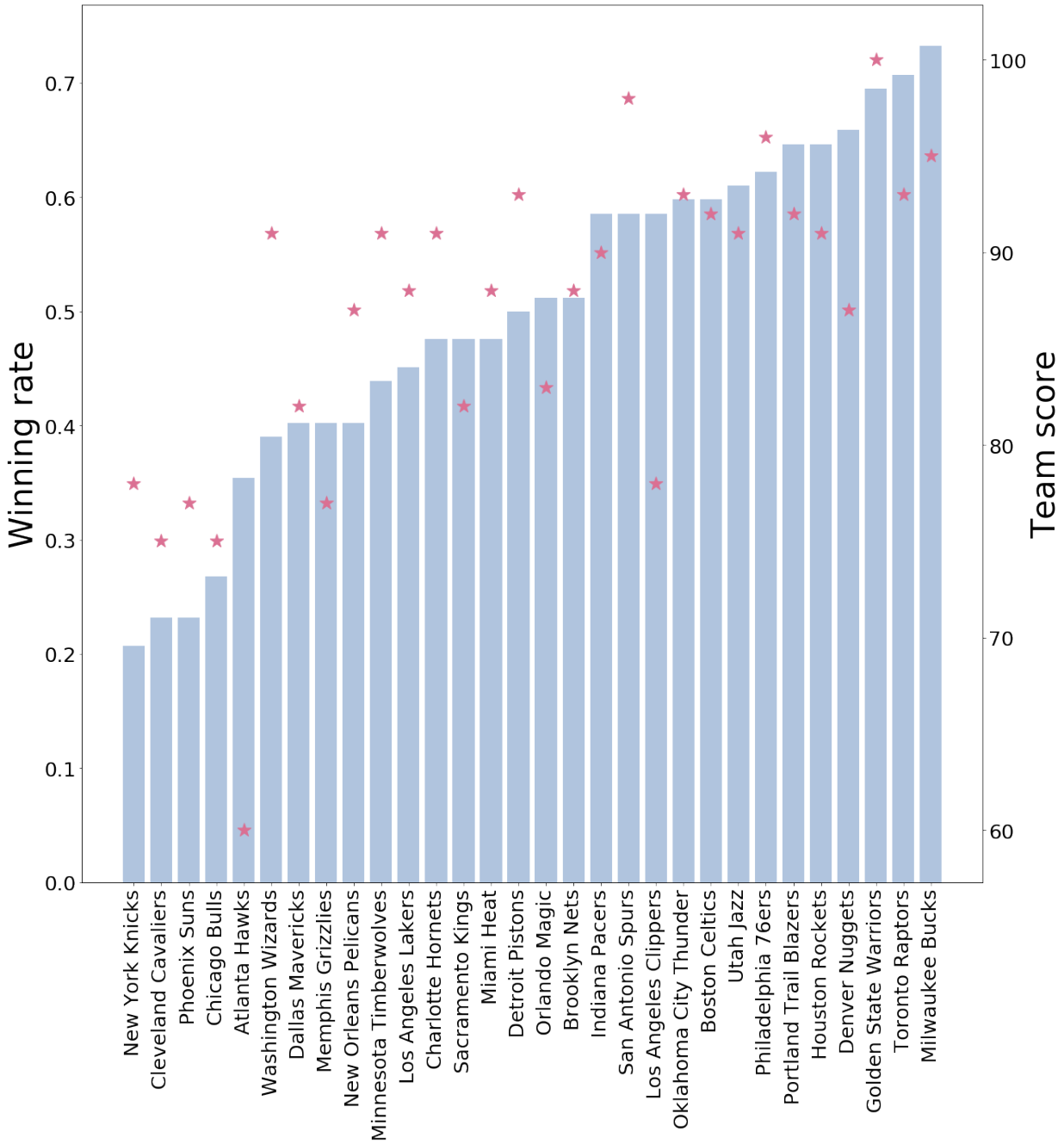


First, we plot the graph based on the team's winning rate this season and each team's payroll. we want to see whether there are any relationships between these two factors. The bar chart is each team's winning rate from high to low. The scatter plot is each team's payroll. We observe that the dot randomly assigns on the bar chart, which means there is no relationship between these two. However, we can separate all the teams into two groups, who played in the play-off, and who did not. The dots distribution for playoff teams is higher than the rest of the teams.

We conclude that if you are the manager or the boss of the team, probably heavy investing is not always the best idea. Although it could somehow help the team break into the playoffs, the highest investing/payroll could not guarantee you the first place in the league. This season the highest payroll is Miami Heat, but they even did not have the chance to play in the playoffs since they need to pay Chris Bosh $26 million who did not play any games in this season due to Blood Clots illness. Also, history told us the same conclusion. For instance, in 2004 Los Angles Lakers, had Kobe Bryant and Shaq O'Neal, which are both one of the best players in the league. But they were still not satisfied and then signed "The Glove" Gary Payton and Moses Malone. Every fan was shocked by their luxury line-ups. But this luxury line-ups still cannot guarantee their championships and they lost in the finals.

In [28]:
```
df_team=df_team.sort_values(by='W/L%')
fig,axes=plt.subplots(figsize=(20,20))
axes.bar(df_team.index,df_team['W/L%'],color='lightsteelblue')
axes.set_xticklabels(df_team.index,rotation=90,fontsize=25)
axes.tick_params(axis='y', which='major', labelsize=25)
axes.set_ylabel('Winning rate', fontsize=40)
axes2=axes.twinx()
axes2.set_yticks(np.arange(60,110,10))
axes2.scatter(df_team.index,df_team['Total Team Contribution'],color='
palevioletred',s=300,marker="*")
axes2.tick_params(axis='y', which='major', labelsize=25)
axes2.set_ylabel('Team score', fontsize=40)
```

Out[28]:
```
Text(0, 0.5, 'Team score')
```



The second graph we plot is we want to testify whether our grading system of total team contribution is relevant to the actual winning rate. Our assumption is when the total team contribution increases, the winning rate should increase too. In the graph, the star sign has a clear downward trend which implies that when the total team contribution decreases, the winning rate could highly have the chance to drop. Therefore, our grading system is somehow related to the winning rate.

Conclusion

In conclusion, I think our grading system performs well. We compare the individual score to the 2K19 system, the error is within 5. And our grading system is daily-updated when they have the game on that day since the system is based on the player's stats. For instance, the score for Jamal Murray boosted from 81 to 83 in these two-weeks in our system since he performed very well in the playoffs. When we convert the individual score to team contributions, it makes more sense to analyze since a player only has one score which represents overall his performance. But the player may have two or more contributions toward to different teams he belonged if he has been traded during the season.

Then we find out the relationship between team payroll and overall winning rate this season. We conclude that the highest spending on the player's salaries cannot guarantee the team to have the best winning rate. However, if we divide 30 teams into two groups, playoff teams and not. The higher team payroll may somehow help the team break into the playoffs.

In addition, we want to testify if our contribution grading system is following the actual winning rate. We find these two are somehow relevant, in other words, when the team contribution is higher, the winning rate for that team will be higher as well. Therefore, in the next season, when we have the new line-ups and some basic stats for each team, we can continue to utilize our grading system to predict the team's ranking in next season.