Source

https://drive.google.com/drive/folders/1yo7S9uXWpWcIvoEm8Q2TAw2KtjT6e4Bg?usp=sharing

Dataset 2015

Code Explaination:

```
In [34]: teams_db=pd.read_sql_query("""SELECT yearID,teamID,Rank,G,W,name FROM Teams """,con)
         print(teams_db)
```

```
      yearID teamID  Rank    G    W                    name
0       1871    BS1     3   31   20     Boston Red Stockings
1       1871    CH1     2   28   19  Chicago White Stockings
2       1871    CL1     8   29   10   Cleveland Forest Citys
3       1871    FW1     7   19    7     Fort Wayne Kekiongas
4       1871    NY2     5   33   16         New York Mutuals
...      ...    ...   ...  ...  ...                      ...
2800    2015    PIT     2  162   98       Pittsburgh Pirates
2801    2015    SDN     4  162   74         San Diego Padres
2802    2015    SFN     2  162   84     San Francisco Giants
2803    2015    SLN     1  162  100      St. Louis Cardinals
2804    2015    WAS     2  162   83     Washington Nationals

[2805 rows x 6 columns]
```

Read the Teams table from the Database

```
In [3]: salary_db=pd.read_sql_query("SELECT * FROM Salaries",con)
        print(salary_db)
```

```
       yearID teamID lgID   playerID      salary
0        1985    ATL   NL  barkele01    870000.0
1        1985    ATL   NL  bedrost01    550000.0
2        1985    ATL   NL  benedbr01    545000.0
3        1985    ATL   NL   campri01    633333.0
4        1985    ATL   NL  ceronri01    625000.0
...       ...    ...  ...        ...         ...
25570    2015    WAS   NL  treinbl01    512800.0
25571    2015    WAS   NL  ugglada01    507500.0
25572    2015    WAS   NL  werthja01  21000000.0
25573    2015    WAS   NL  zimmejo02  16500000.0
25574    2015    WAS   NL  zimmery01  14000000.0

[25575 rows x 5 columns]
```

Read the Salaries table from the Database

```
In [4]: """
        Make the earliest year = 1985 to ignore the missing data issue
        """
        TotalSalary = pd.read_sql_query("""
        SELECT Teams.teamID AS TeamName,sum(salary)/1000000 AS TotalSalaryinMillions,sum(Teams.W) AS TotalWins,sum(Teams.L) AS
        TotalLoses,sum(Teams.G) AS TotalGames, CAST(sum(Teams.W) AS FLOAT)/CAST(sum(Teams.G) AS FLOAT) * 100 AS 'AvgWinning'
        FROM Teams join Salaries
        ON Teams.teamID = Salaries.teamID And Teams.yearID = Salaries.yearID
        WHERE teams.yearid>=1985
        GROUP BY Teams.teamID
        """,con)

In [5]: TotalSalary.to_csv("Data.csv")
```

Run a query to merge the two tables Teams and Salaries and save it into a csv file from year 1985 to skip the missing data in the Salaries table

```
TotalSalary1990= pd.read_sql_query("""
SELECT Teams.teamID AS TeamName,sum(salary)/1000000 AS TotalSalaryinMillions,sum(Teams.W) AS TotalWins,
sum(Teams.L) AS TotalLoses,sum(Teams.G) AS TotalGames, CAST(sum(Teams.W) AS FLOAT)/CAST(sum(Teams.G) AS FLOAT) * 100
AS 'AvgWinning',Teams.yearID AS Year
FROM Teams,Salaries
WHERE Teams.yearID >= 1990 and Teams.yearID<=2015 and Teams.yearid=Salaries.yearID and teams.teamID =Salaries.teamID
GROUP BY Teams.yearID, Teams.teamID
""", con)
TotalSalary1990.to_csv("Data90.csv")
D90 = pd.read_csv("Data90.csv")
```

Run a query to merge the two tables Teams and SaRun a query to merge the two tables Teams and Salaries and save it into a csv file from year 1990 till 2015 as ordered

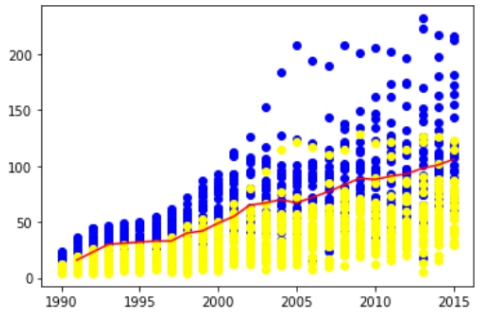
In [7]:
```
SalaryPerYear=[]
j=0
y=[]
year=D90.Year[j]
x=0
a=0
for i in D90.TotalSalaryinMillions:
    if year == D90.Year[j]:
        x=int(i)+x
        a=a+1
    if year != D90.Year[j]:
        z=x/a
        z=int(z)
        y.append(D90.Year[j])
        SalaryPerYear.append(z)

        for i in range(a):
            y.append(D90.Year[j])
            SalaryPerYear.append(z)
        x=0
        a=0
        year=D90.Year[j]
    j=j+1
    if j>=len(D90.TotalSalaryinMillions):
        for i in range(a):
            y.append(D90.Year[j-1])
            SalaryPerYear.append(z)
        break
```

A function to save the total salary and the year in an array for better usage

In [8]:
```
print (len(SalaryPerYear),len(D90.Year))
plt.scatter(D90.Year, D90.TotalSalaryinMillions, color='blue')
plt.plot(y,SalaryPerYear, color='red')
plt.scatter(D90.Year, (D90.TotalSalaryinMillions*(D90.AvgWinning/100)), color='yellow')
```

```
758 758
```

Out[8]: <matplotlib.collections.PathCollection at 0x1abf55bfeb0>



As we can see in the above graph we see can some yellow dot at the bottom and others flying up the reason of that is that the yellow dots represents each team spending efficiency by the formula of (Spending*(avgwinning/100)) the more the team wins the more effective of spending

This graph represents the data from 1985 till 2015 we can see that it's a growing

business

```
"""
The correlation between the average winning and the total salary for the year 2015
"""
fig = px.scatter(D90, x=D90.TotalSalaryinMillions[728:757], y=D90.AvgWinning[728:757], color=D90.TeamName[728:757],
title='The correlation between the average winning and the total salary in the year 2015')
fig.show()
```

A function to display a graph showing the corelation between Total Salary and the average winning for the year 2015

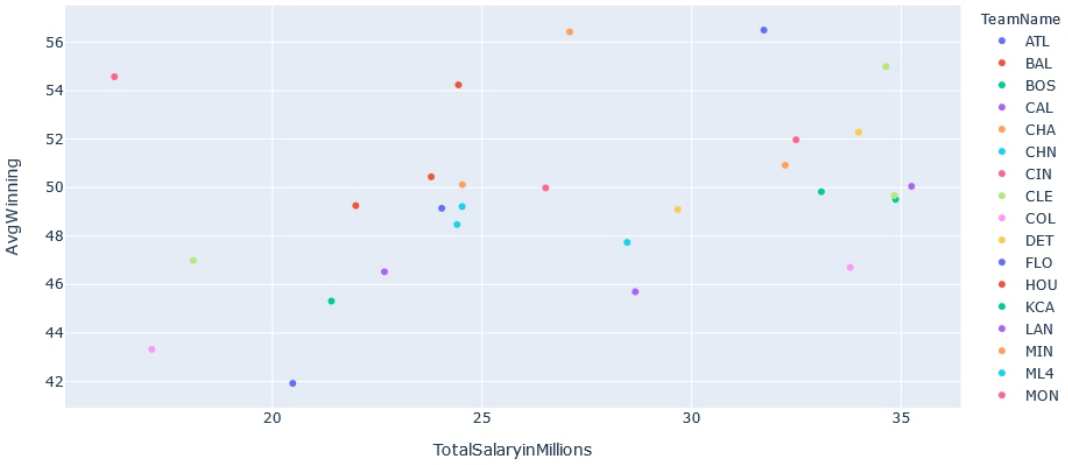
As we can see the more the club spends the more he increases his chances of increasing the average winning

This graph represent the data from the year of 2015

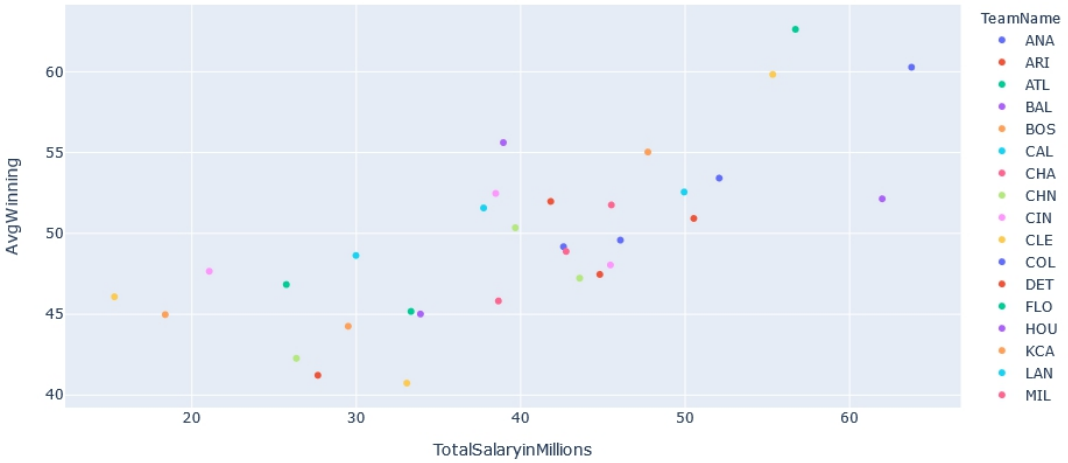
```
years = list(set(D90.Year))

for i in range(0, 25, 5):
    D5Y = D90[D90['Year'].isin(years[i:i+5])]
    D5Y = D5Y.groupby(['TeamName']).mean()
    D5Y['TeamName'] = D5Y.index
    fig = px.scatter(D5Y, x="TotalSalaryinMillions", y="AvgWinning", color='TeamName',
                     title=f'Period from {years[i:i+5][0]} to {years[i:i+6][-1]}')
    fig.show()
```

A function to divide the data into 5 years period

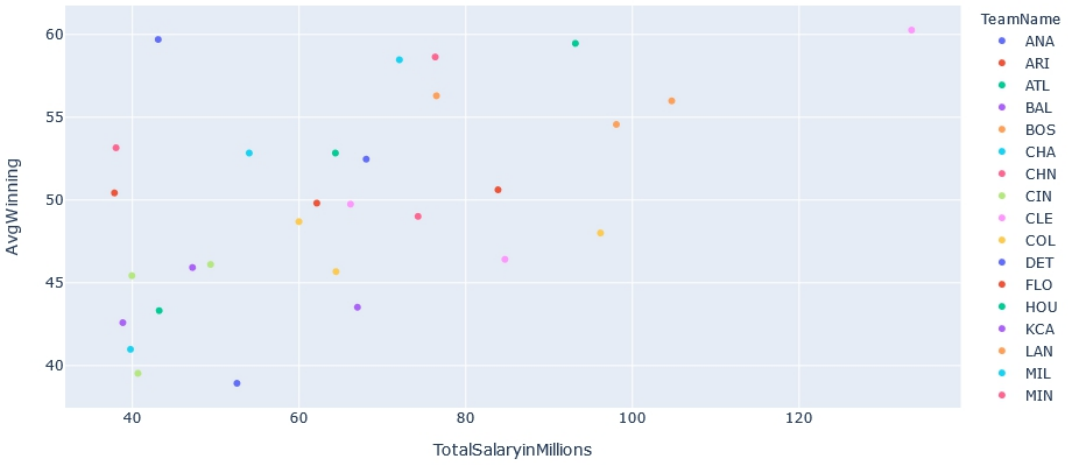
Period from 1990 to 1995



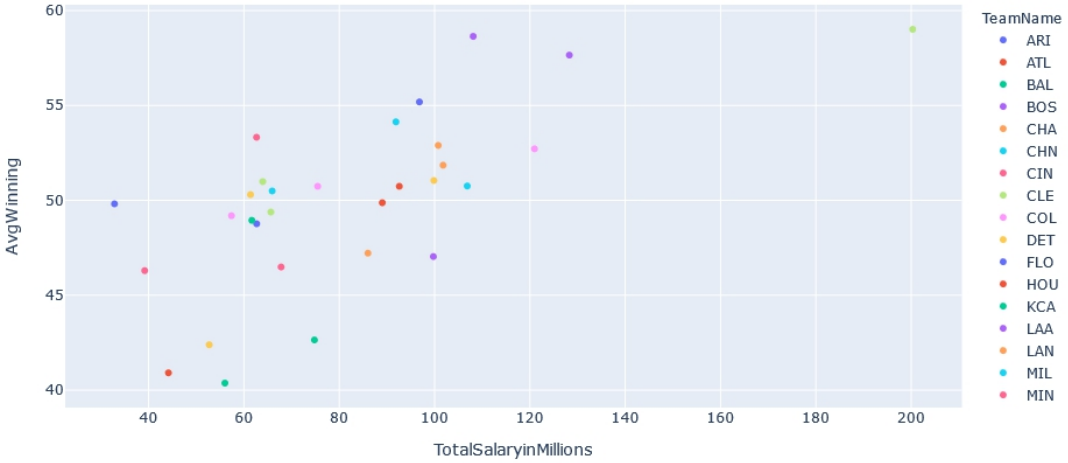
Period from 1995 to 2000

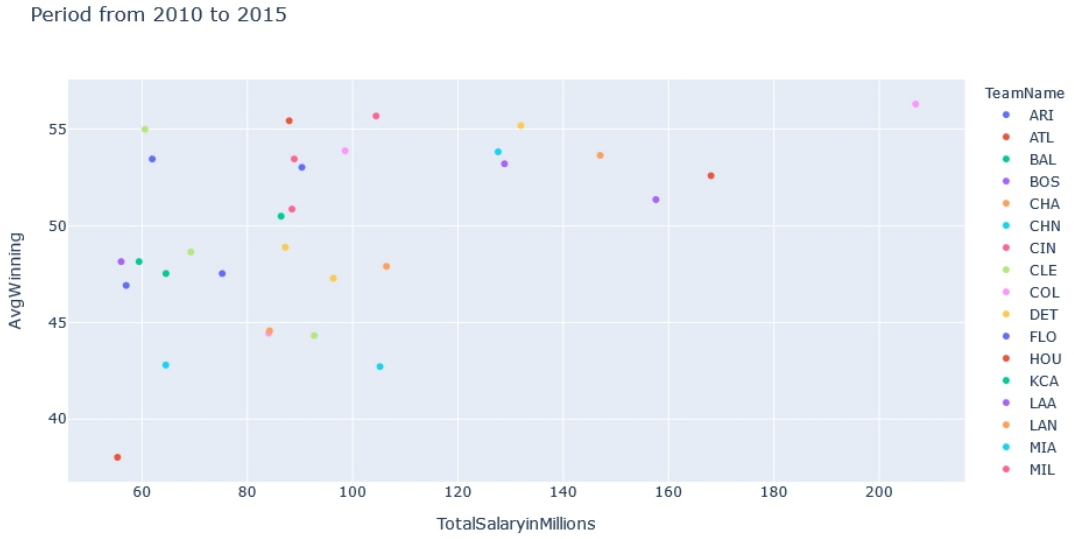


Period from 2000 to 2005



Period from 2005 to 2010






```
avg_std = {}
for year in set(D90.Year):
    avg = D90[D90.Year==year].TotalSalaryinMillions.mean()
    std = D90[D90.Year==year].TotalSalaryinMillions.std()
    avg_std[year] = [avg, std]
```

```
print('Mean and STD for each year:')
print(avg_std, end=',')
```

```
Mean and STD for each year:
{1990: [17.072353576923074, 3.771834264904344], 1991: [23.57878530769231, 6.894668740349858], 1992: [30.98243550000
0005, 9.150607115667599], 1993: [32.20500478571428, 9.23248473470758], 1994: [33.13701025000001, 8.52874924857643
5], 1995: [33.98104882142857, 9.447998182386566], 1996: [34.17798392857142, 10.688534632030327], 1997: [40.26021017
857143, 13.060728290788592], 1998: [42.60942903333333, 15.380810603487713], 1999: [49.807625000000016, 20.561328327
32466], 2000: [55.53783673333333, 21.416220275035727], 2001: [65.3554437666667, 24.7077063016089371, 2002: [67.4692
5073333334, 24.6921926673284561, 2003: [70.94207093333333, 28.011963452594934], 2004: [69.0221981, 32.8241142601368
24], 2005: [72.95711326666667, 34.17478068120626], 2006: [77.38242056666667, 32.26494805430753], 2007: [82.55629956
666665, 33.907053769054891, 2008: [89.49528899999999, 37.802000749774281, 2009: [88.82423313333332, 33.857092606273
454], 2010: [90.71199550000001, 38.11503119990699], 2011: [92.81684303333331, 40.8119744485815661, 2012: [97.758039
73333332, 36.81753825280752], 2013: [101.15085493333335, 48.83028738236983], 2014: [106.41058743333333, 42.50538127
628833], 2015: [117.13808563333333, 40.37166651825746]},
```

A function to get the average standard deviation

```
for index, row in D90.iterrows():
    D90.loc[index, 'std_pay'] = (row.TotalSalaryinMillions - avg_std[row.Year][0])/avg_std[row.Year][1]
```

```
D90.std_pay.mean(),D90.std_pay.std()
```

```
(-1.113152372975091e-17, 0.9833488195572834)
```

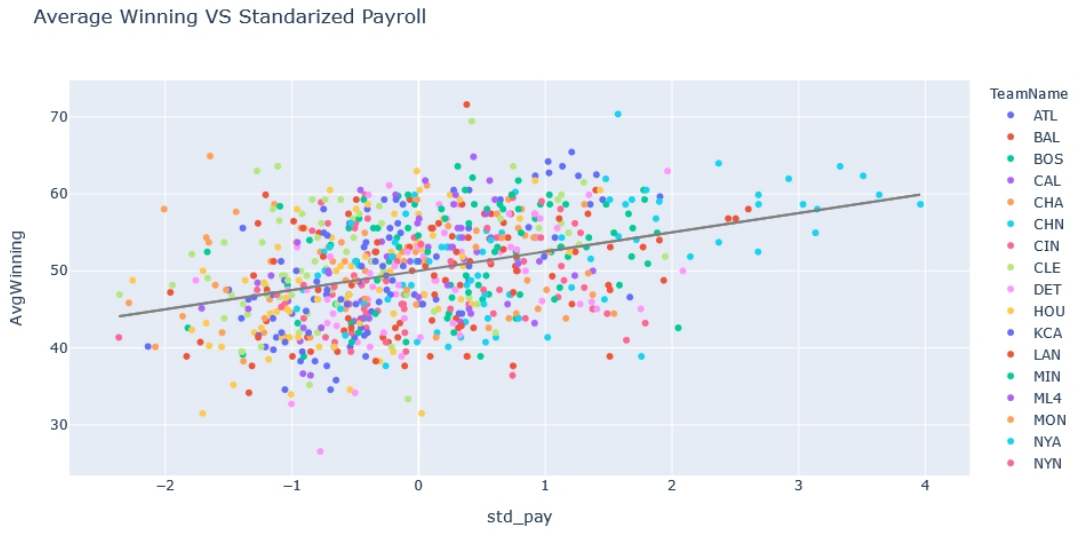
```
D90.to_csv("Data90.csv")
D90['expected_win_pct'] = 50 + 2.5 *D90['std_pay']
D90.head()
```

	Unnamed: 0	TeamName	TotalSalaryinMillions	TotalWins	TotalLoses	TotalGames	AvgWinning	Year	std_pay	expected_win_pct
0	0	ATL	14.555501	2080	3104	5184	40.123457	1990	-0.667275	48.331811
1	1	BAL	9.680084	2812	3145	5957	47.204969	1990	-1.959861	45.100348
2	2	BOS	20.558333	2816	2368	5184	54.320988	1990	0.924213	52.310533
3	3	CAL	21.720000	2800	2870	5670	49.382716	1990	1.232198	53.080495
4	4	CHA	9.491500	2914	2108	5022	58.024691	1990	-2.009859	44.975353

Add the standard deviation and expected winning to the csv file

In [19]:
```
# Standarized vs Average Winning
fig = px.scatter(D90, x="std_pay", y="AvgWinning", color="TeamName",
                 title='Average Winning VS Standarized Payroll')
#  the regression line
fig.add_trace(
    go.Scatter(
        x=D90['std_pay'],
        y=D90['expected_win_pct'],
        mode="lines",
        line=go.scatter.Line(color="gray"),
        showlegend=False)
)

fig.show()
```

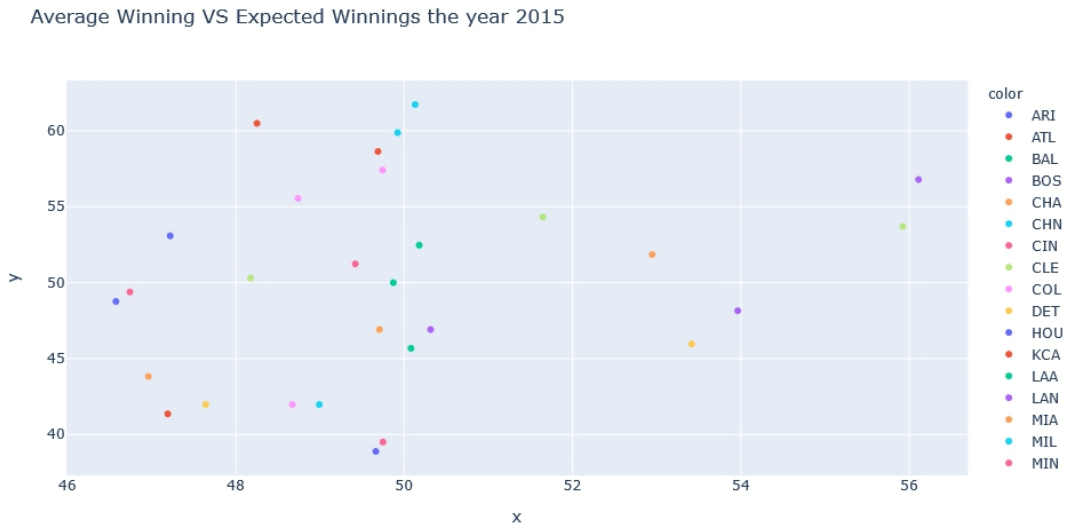
Plot a figure showing the relation between average winning and the standard deviation



As we can see the more the standarized total salary the more the team win games

In [21]:
```
fig = px.scatter(D90, x=D90.expected_win_pct[728:757], y=D90.AvgWinning[728:757], color=D90.TeamName[728:757],
title='Average Winning VS Expected Winnings the year 2015')
fig.show()
```

A figure to show the relation between the Average Winning and the Expected Winning



As we can see the LAN team was accurately predicted, the SLN team surpassed the exceptions while PHI has lowered the expectations of his followers

```
In [23]: D90['efficiency'] = D90['AvgWinning'] - D90['expected_win_pct']
         D90.head()
```

Out[23]:

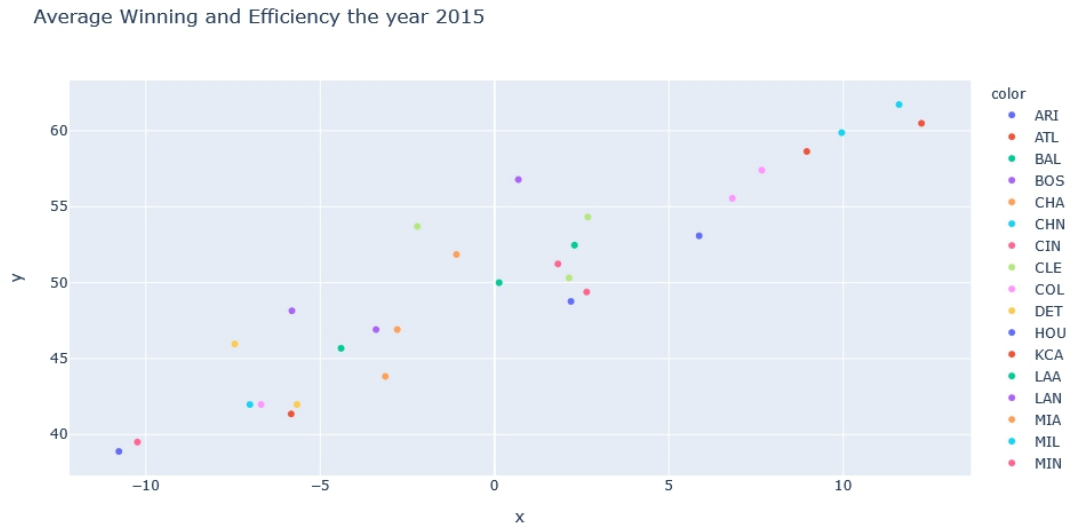
	Unnamed: 0	TeamName	TotalSalaryinMillions	TotalWins	TotalLoses	TotalGames	AvgWinning	Year	std_pay	expected_win_pct	efficiency
0	0	ATL	14.555501	2080	3104	5184	40.123457	1990	-0.667275	48.331811	-8.208354
1	1	BAL	9.680084	2812	3145	5957	47.204969	1990	-1.959861	45.100348	2.104621
2	2	BOS	20.558333	2816	2368	5184	54.320988	1990	0.924213	52.310533	2.010454
3	3	CAL	21.720000	2800	2870	5670	49.382716	1990	1.232198	53.080495	-3.697779
4	4	CHA	9.491500	2914	2108	5022	58.024691	1990	-2.009859	44.975353	13.049338

```
In [24]: D90.to_csv("Data90.csv")
```

A function to calculate then add efficiency to the table

```
fig = px.scatter(D90, x=D90.efficiency[728:757], y=D90.AvgWinning[728:757], color=D90.TeamName[728:757],
                 title='Average Winning and Efficiency the year 2015')
fig.show()
```

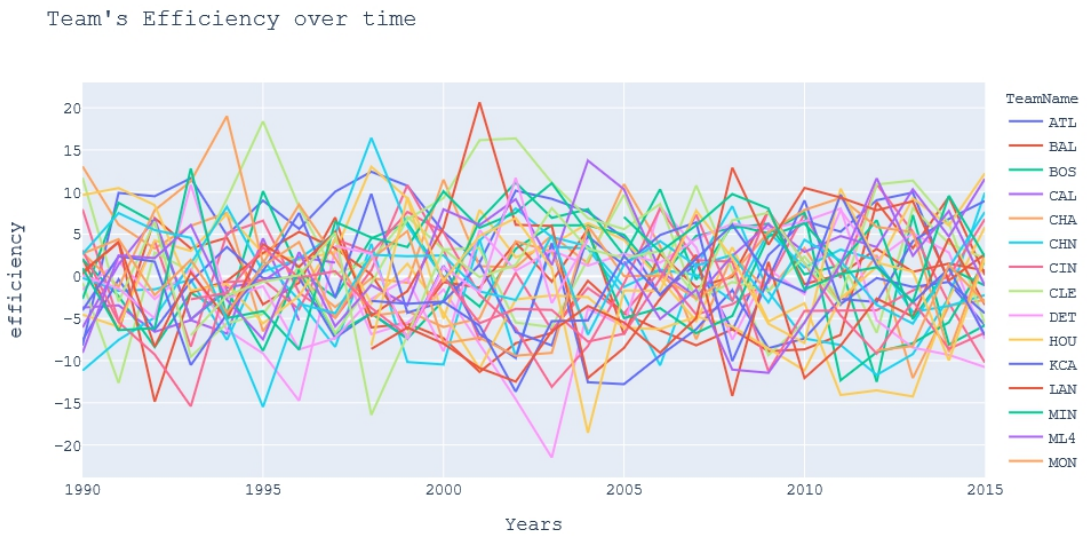
A figure to show the reltaion between average winning and the efficiency for the year in 2015



For Investments SLN would be the best to invest in it while PHI will be the worst to invest in it

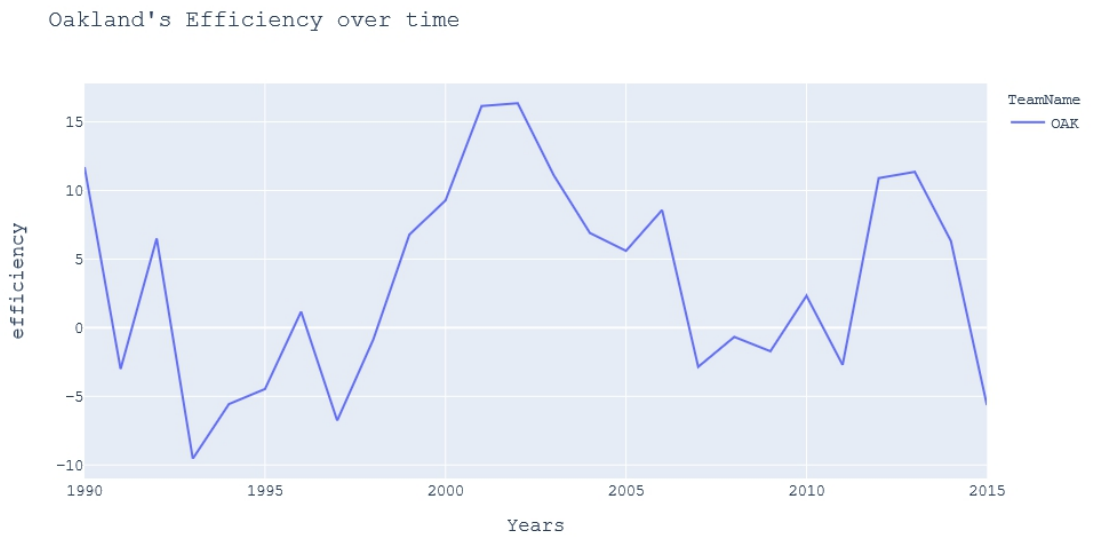
```
fig = px.line(D90, x="Year", y="efficiency", color="TeamName",
              title="Team's Efficiency over time")
fig.update_layout(
    xaxis_title='Years',
    yaxis_title='efficiency',
    font=dict(
        family="Courier New, monospace",
        size=14)
    )
fig.show()
```

A figue to show the Teams efficiency over the period between 1990 and 2015



```
fig = px.line(D90[D90.TeamName=='OAK'], x="Year", y="efficiency", color="TeamName",
              title="Oakland's Efficiency over time")
fig.update_layout(
    xaxis_title='Years',
    yaxis_title='efficiency',
    font=dict(
        family="Courier New, monospace",
        size=14)
    )
fig.show()
```

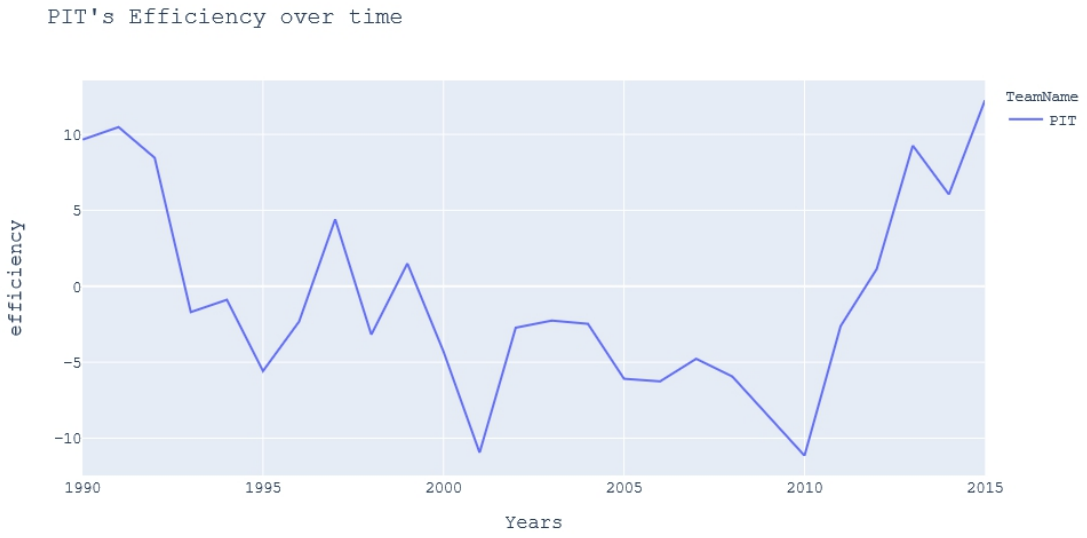
A figue to show the OAK team efficiency over the period between 1990 and 2015



The spending efficiency in OAK team isnot stable and is deteriorating from the year 2013 till 2015 and have decreased by more than 49%. OAK was in his best status during the year 2001 and 2002

```
fig = px.line(D90[D90.TeamName=='PIT'], x="Year", y="efficiency", color="TeamName",
              title="PIT's Efficiency over time")
fig.update_layout(
    xaxis_title='Years',
    yaxis_title='efficiency',
    font=dict(
        family="Courier New, monospace",
        size=14)
    )
fig.show()
```

A figue to show the PIT team efficiency over the period between 1990 and 2015



The best to invest in it during the year 2015 is the PIT team which had an in increase in efficiency of more than 200% since the 2010