project2

March 17, 2023

[606]:
```
#Eric Feng

#Part1
import sqlite3
import pandas as pandaz
import numpy as np
import matplotlib.pyplot as plt
import statistics

%matplotlib inline

#Connects to an sqlite database file called lahman2014.sqlite
sqlite_file = 'lahman2014.sqlite'
conn = sqlite3.connect(sqlite_file)

#Define a SQL query that calcs total payroll
salary_query = "SELECT teamID, yearID, sum(salary)/1000 as␣
↪total_payroll_by_thousand, sum(salary)/count(salary) as payroll_mean FROM␣
↪Salaries GROUP BY teamID, yearID"
team_salaries = pandaz.read_sql(salary_query, conn)
#conn
#team_salaries.head()
```

[607]:
```
# Define two more SQL queries using the Teams and Salaries table

#Calculate winning percent
team_query = """
    SELECT
        ((W * 100.0) / G) as winning_percentage,
        *
    FROM
        Teams
    GROUP BY
        teamID,
        yearID
"""
```

```python
#Calculate mean salary
mean_query = """
    SELECT
        yearID,
        sum(salary) / count(salary) as salary_mean
    FROM
        Salaries
    GROUP BY
        yearID
"""

#Execute team_query and mean_query using panda's read_sql then stores the␣
 ↪results in dataframes
team_table = pandaz.read_sql(team_query, conn)
mean_table = pandaz.read_sql(mean_query,conn)

# Merge team salaries and team table on teamID and yearID, these two are JOIN␣
 ↪KEYS
result = pandaz.merge(
    team_salaries,   # left DataFrame to merge
    team_table,      # right DataFrame to merge
    how='outer',
    on=['teamID', 'yearID']  # columns
)

# The final merged DataFrame (for further analysis) as we include other colymns␣
 ↪that help
# when performing EDA later
result = pandaz.merge(
    result,          # left DataFrame to merge (result of previous merge)
    mean_table,      # right DataFrame to merge
    how='outer',
    on=['yearID']    # column
)

# Return the final merged DataFrame to be used for future analysius
result

#PROBLEM 1 ANALYSIS
#When dealing with missing data, an outer join was used to merge the␣
 ↪DataFrames,
#ensuring that all data was retained. If a team did not have payroll or winning
#percentage data for a particular year, the corresponding values in the␣
 ↪resulting
#DataFrame would be NaN.
```

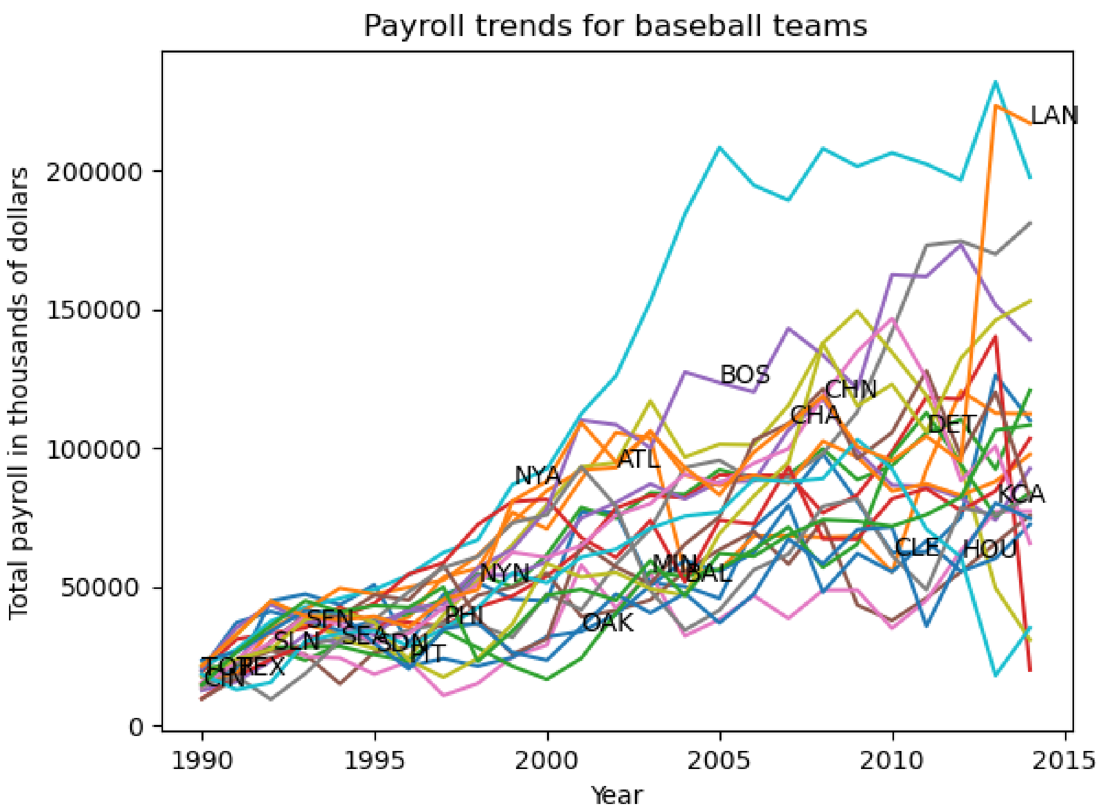
[607]:
```
      teamID  yearID  total_payroll_by_thousand   payroll_mean  \
0        ATL    1985                  14807.000  673045.454545
1        BAL    1985                  11560.712  525486.909091
2        BOS    1985                  10897.560  435902.400000
3        CAL    1985                  14427.894  515281.928571
4        CHA    1985                   9846.178  468865.619048
…        …     …                         …             …
2772     CHN    1878                        NaN            NaN
2773     CN1    1878                        NaN            NaN
2774     IN1    1878                        NaN            NaN
2775     ML2    1878                        NaN            NaN
2776     PRO    1878                        NaN            NaN

      winning_percentage lgID franchID divID  Rank      G  …    FP  \
0              40.740741   NL      ATL     W   5.0  162.0  …  0.97
1              51.552795   AL      BAL     E   4.0  161.0  …  0.98
2              49.693252   AL      BOS     E   5.0  163.0  …  0.97
3              55.555556   AL      ANA     W   2.0  162.0  …  0.98
4              52.147239   AL      CHW     W   3.0  163.0  …  0.98
…                    …  …        …   …    …    …   …   …
2772           49.180328   NL      CHC  None   4.0   61.0  …  0.89
2773           60.655738   NL      CNR  None   2.0   61.0  …  0.90
2774           38.095238   NL      IBL  None   5.0   63.0  …  0.89
2775           24.590164   NL      MLG  None   6.0   61.0  …  0.86
2776           53.225806   NL      PRO  None   3.0   62.0  …  0.89

                          name                          park attendance  \
0               Atlanta Braves  Atlanta-Fulton County Stadium  1350137.0
1            Baltimore Orioles               Memorial Stadium  2132387.0
2               Boston Red Sox                  Fenway Park II  1786633.0
3            California Angels               Anaheim Stadium  2567427.0
4            Chicago White Sox                  Comiskey Park  1669888.0
…                          …                            …          …
2772  Chicago White Stockings               Lake Front Park I        NaN
2773            Cincinnati Reds                Avenue Grounds        NaN
2774         Indianapolis Blues             South Street Park        NaN
2775            Milwaukee Grays               Eclipse Park II        NaN
2776          Providence Grays         Messer Street Grounds        NaN

        BPF    PPF teamIDBR  teamIDlahman45  teamIDretro    salary_mean
0     105.0  106.0      ATL             ATL          ATL  476299.447273
1      97.0   97.0      BAL             BAL          BAL  476299.447273
2     104.0  104.0      BOS             BOS          BOS  476299.447273
3     100.0  100.0      CAL             CAL          CAL  476299.447273
4     104.0  104.0      CHW             CHA          CHA  476299.447273
…       …     …      …             …            …            …
2772  106.0  105.0      CHC             CHN          CHN            NaN
```

```
2773   91.0   92.0      CIN              CN1          CN1             NaN
2774   87.0   89.0      IND              IN1          IN1             NaN
2775  106.0  113.0      MLG              ML2          ML2             NaN
2776  100.0   96.0      PRO              PRO          PRO             NaN

[2777 rows x 52 columns]
```

[608]:

```
## Part 2
## Problem 2

#Sorts the input by years
result = result.sort_values("yearID")

#Only contain rows between 1990 and 2014
df1 = result[(result['yearID'] > 1989) & (result['yearID'] < 2015)]

#Createm a new Series but only with unique vales
teams = df1['teamID'].unique()

#Update to only contain needed columns
df1 = df1[['yearID', 'teamID', 'total_payroll_by_thousand']]
df1 = df1.set_index('teamID')

#Set initial year
year = 1990

# Create the plot
fig, ax = plt.subplots()

# Loop over teams and plot their payroll data
for t in teams:
    # Filter the payroll data for the current year and team
    temp1 = df1[(df1.yearID == year) & (df1.index == t)]

    # Check if the team has payroll data for the current year
    if not temp1.empty:
        # Get the payroll value for the current year and team
        num = temp1['total_payroll_by_thousand'].values[0]

        # Add the team name as an annotation
        ax.annotate(t, xy=(year, num))

        # Plot the payroll value for the current year and team
        ax.plot(df1.loc[t, 'yearID'], df1.loc[t, 'total_payroll_by_thousand'])

    # Increment the year
    if year < 2014:
```

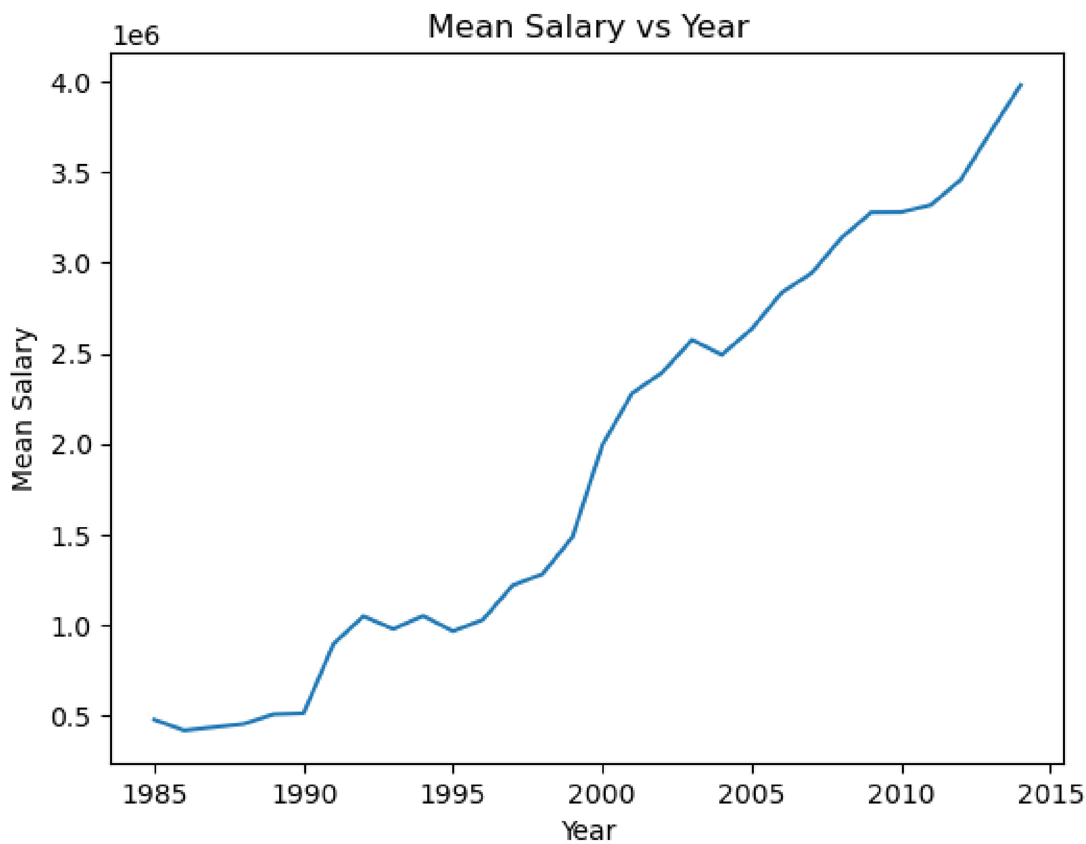
```
        year += 1
    else:
        year = 1990

# Set the axis labels and title
ax.set_xlabel('Year')
ax.set_ylabel('Total payroll in thousands of dollars')
ax.set_title('Payroll trends for baseball teams')

# Show the plot
plt.show()
```



[609]:
```
# Question 1
# From the graph, we can deduce that the payrolls of the teams are increasing␣
↪over time.
# The rates of the payrolls increasing is different however, as over time, the
# differences between the payrolls of the teams are also increasing
```

[610]:
```
#Problem 3
# In question 1, I stated that the payrolls of the teams increase over time
# Below is a plot of the average payroll over time
```

```
## Problem 3
result.sort_values("salary_mean")
plt.plot(result['yearID'],result['salary_mean'])
plt.xlabel('Year')
plt.ylabel('Mean Salary')
plt.title('Mean Salary vs Year')
plt.show()
```



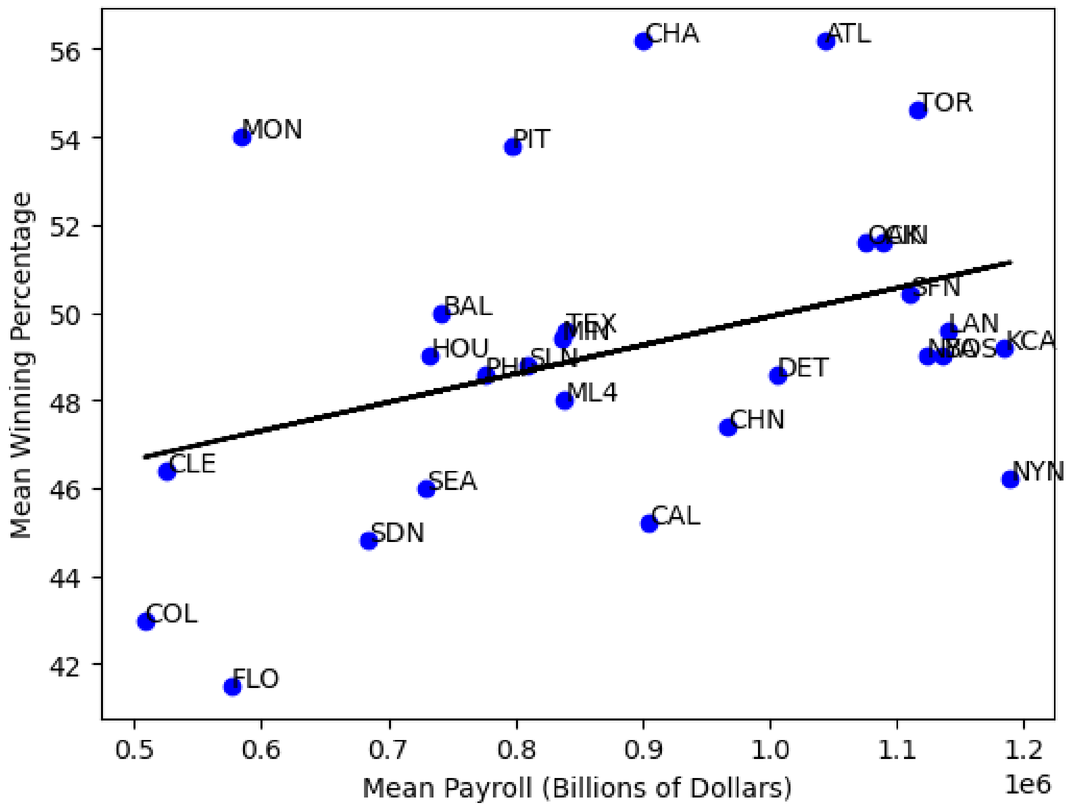
[611]:
```
#Problem 4

#Create database connections
q1 = pandaz.read_sql("SELECT  teamID,yearID, sum(salary)/count(salary) as␣
↪payroll_mean FROM Salaries GROUP BY teamID,yearID", conn)
q2 = pandaz.read_sql("SELECT  teamID,yearID, W*100/G as winning_percentage FROM␣
↪Teams GROUP BY teamID,yearID",conn)
```

[612]:
```
df1 = q1.merge(q2, how='outer', on=['yearID', 'teamID'])
```

[613]:
```
year_intervals = [1989,1994,1999,2004,2009,2015]
eras = ['1990-1994','1995-1999','2000-2004','2005-2009','2010-2015']
categories = pandaz.cut(df1['yearID'],year_intervals,labels = eras)
df1['categories'] = pandaz.cut(df1['yearID'], year_intervals, labels = eras)

df2 = df1.groupby('categories')
```

[614]:
```
# select unique teamIDs from the 'group1' DataFrame
group1 = df2.get_group('1990-1994')
teams = group1['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group1[['teamID', 'winning_percentage', 'payroll_mean']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['payroll_mean']
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Mean Payroll (Billions of Dollars)')
plt.ylabel('Mean Winning Percentage')
```

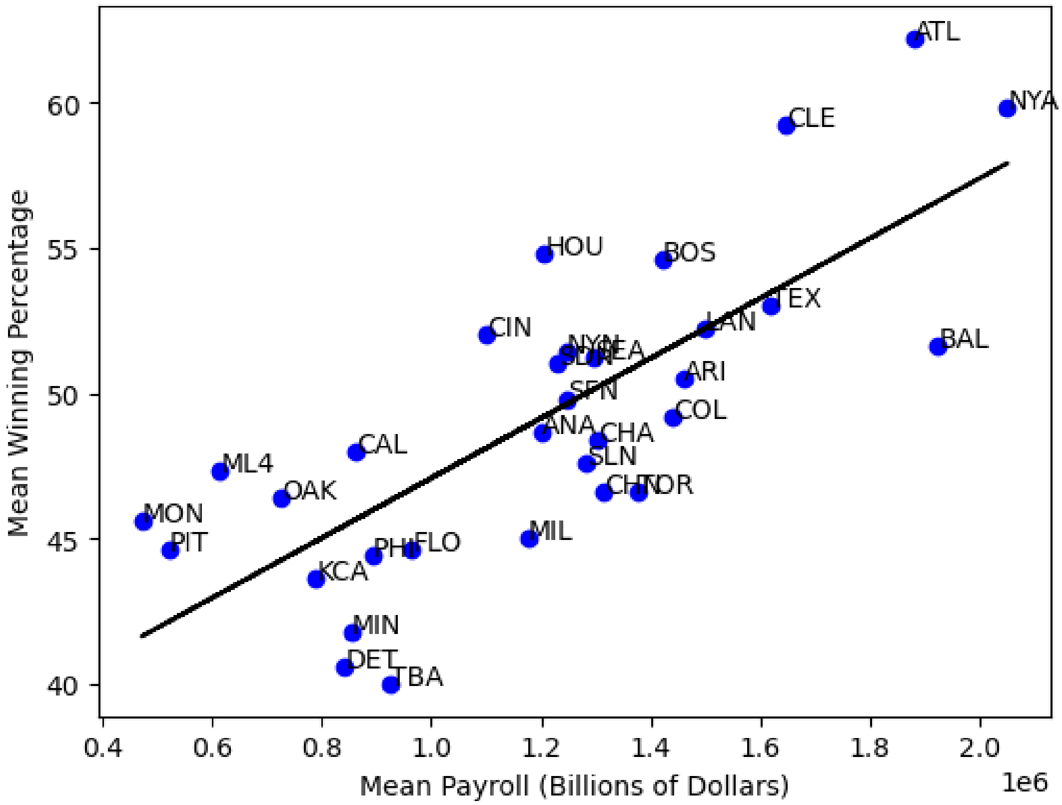
[614]:
```
Text(0, 0.5, 'Mean Winning Percentage')
```



[615]:
```
# select unique teamIDs from the 'group1' DataFrame
group2 = df2.get_group('1995-1999')
teams = group2['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group2[['teamID', 'winning_percentage', 'payroll_mean']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['payroll_mean']
```

```
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Mean Payroll (Billions of Dollars)')
plt.ylabel('Mean Winning Percentage')
```

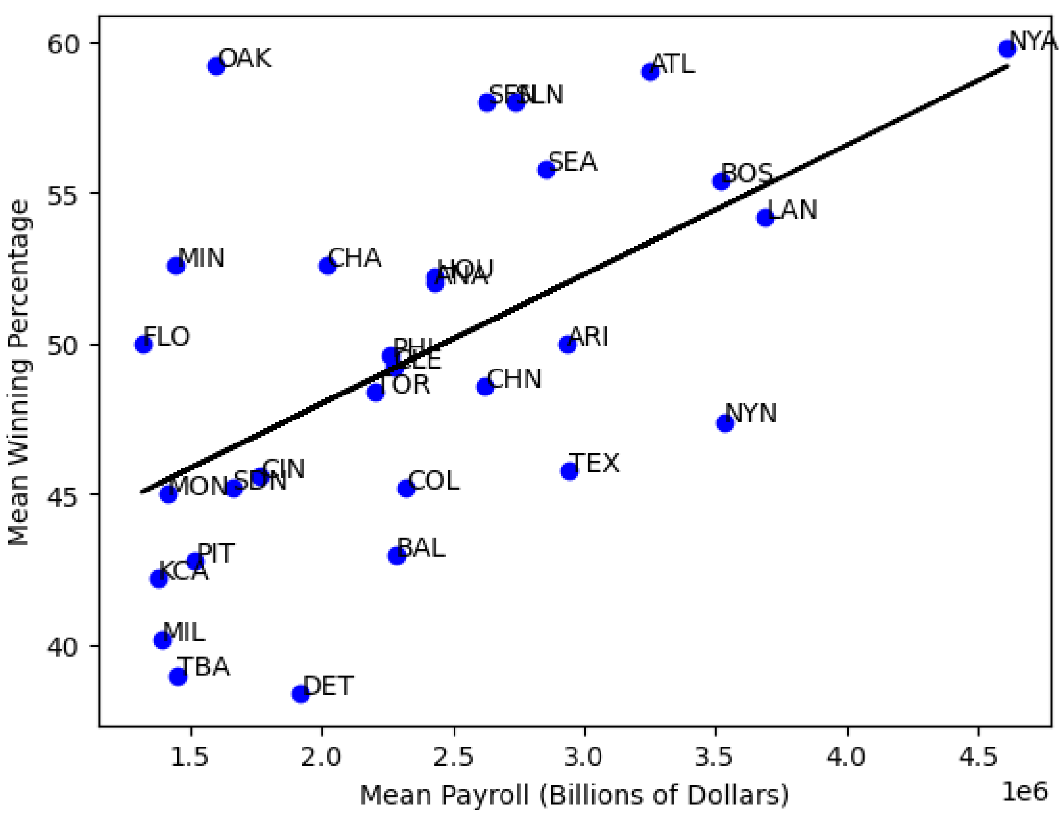
[615]: Text(0, 0.5, 'Mean Winning Percentage')



[616]:
```
# select unique teamIDs from the 'group1' DataFrame
group3 = df2.get_group('2000-2004')
teams = group3['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group3[['teamID', 'winning_percentage', 'payroll_mean']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
```

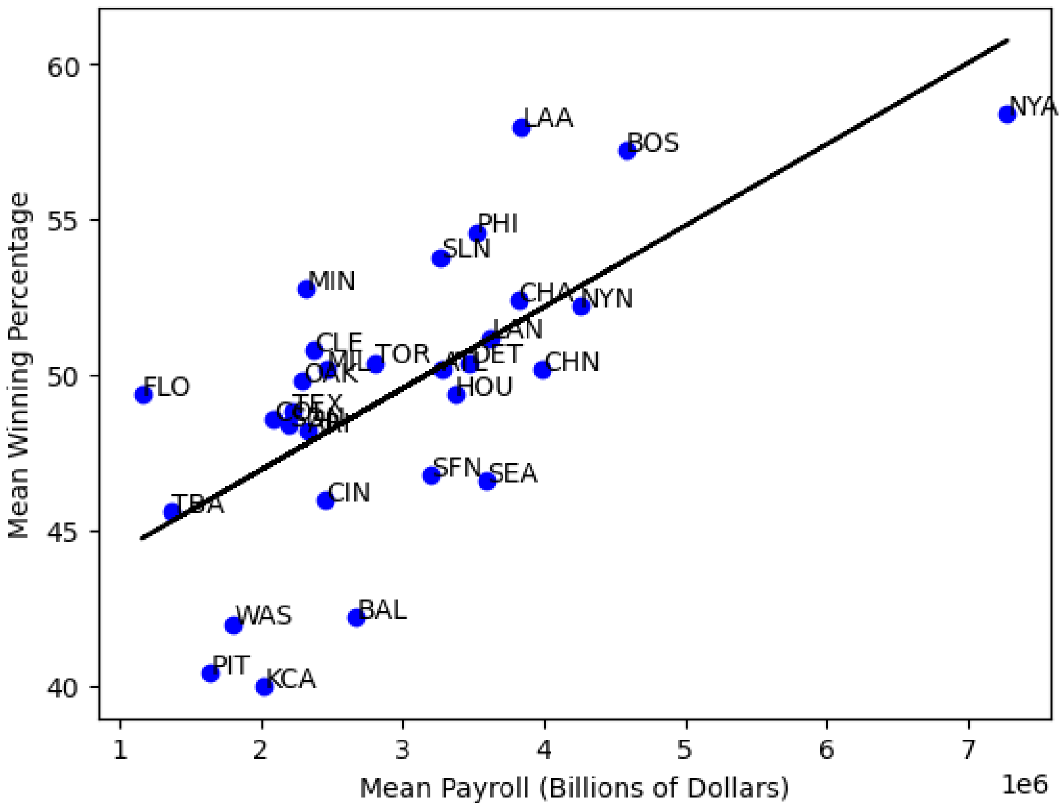
```
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['payroll_mean']
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Mean Payroll (Billions of Dollars)')
plt.ylabel('Mean Winning Percentage')
```

[616]: Text(0, 0.5, 'Mean Winning Percentage')



[617]:
```python
# select unique teamIDs from the 'group1' DataFrame
group4 = df2.get_group('2005-2009')
teams = group4['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group4[['teamID', 'winning_percentage', 'payroll_mean']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['payroll_mean']
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Mean Payroll (Billions of Dollars)')
plt.ylabel('Mean Winning Percentage')
```

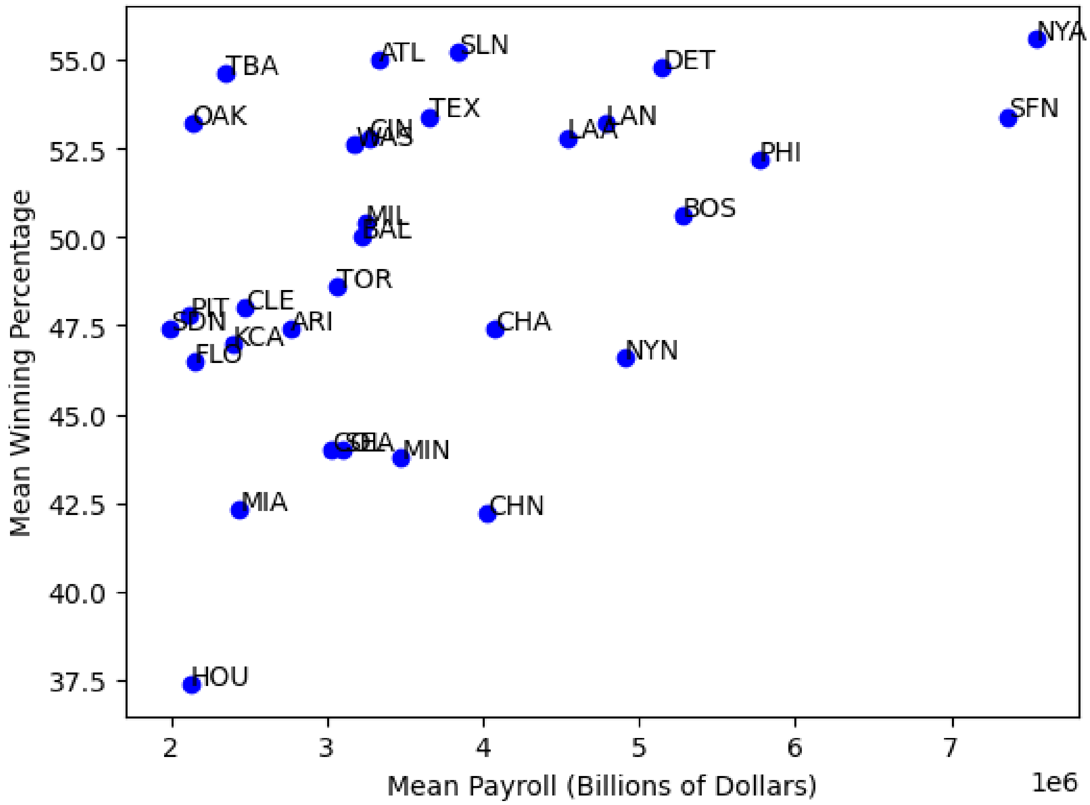
[617]:
```
Text(0, 0.5, 'Mean Winning Percentage')
```




```
[618]: # select unique teamIDs from the 'group1' DataFrame
group5 = df2.get_group('2010-2015')
teams = group5['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group5[['teamID', 'winning_percentage', 'payroll_mean']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'payroll_mean'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['payroll_mean']
```

```python
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Mean Payroll (Billions of Dollars)')
plt.ylabel('Mean Winning Percentage')
```

[618]: Text(0, 0.5, 'Mean Winning Percentage')



[619]:
```python
# Question 2
# The mean payroll is correlated with the win percentage.
# The more money a team spends, the more they win.  New York (NYA)
# seems to be a standout when it comes to teams that are good at
# paying for wins.  Oakland stays above the regression line - this means
# they are very efficient with their money as they spend less but are
# able to win more.
```

[620]:
```python
# Part 3
# Problem 5
```

```python
# Initialize variables to store payroll averages and standard deviations
starting_year = 1985
iterator = starting_year

payroll_avg = {}
payroll_std = {}

#Loop through each year in the DataFrame
for year in q1['yearID'].unique():
    # Get the rows corresponding to the current year
    year_rows = q1[q1['yearID'] == year]
    # Calculate the average and standard deviation of the payroll means for the␣
↪current year, store them in the dictionaries
    payroll_avg[year] = year_rows['payroll_mean'].mean()
    payroll_std[year] = year_rows['payroll_mean'].std()

#Loop through each row of the DataFrame again
stand_lst = []
for index, row in q1.iterrows():
    # Get the current year and payroll mean for the row
    curr_year = row['yearID']
    payroll = row['payroll_mean']
    # Calculate the standardized payroll for the row using the payroll mean,␣
↪average, and standard deviation for the year
    stand_lst.append((payroll - payroll_avg[curr_year]) /␣
↪payroll_std[curr_year])

#Add the standardized payroll list as a new column to the DataFrame
q1['standardized_payroll'] = stand_lst

q1
```

[620]:

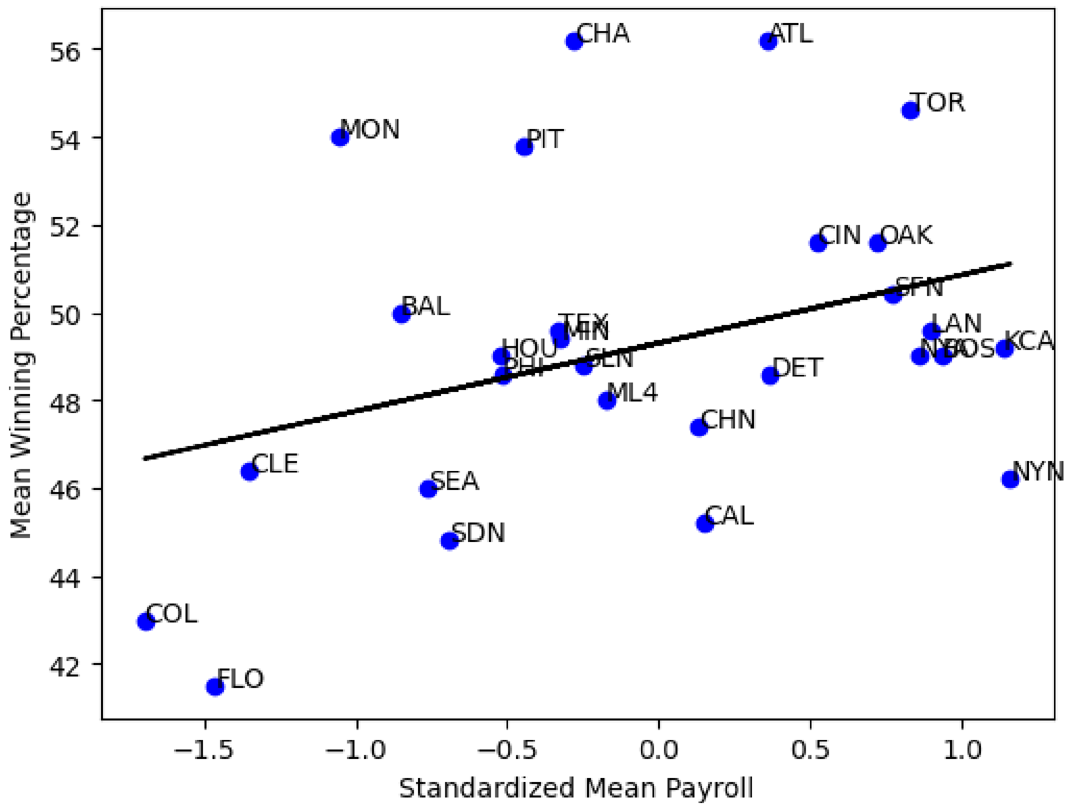
```
    teamID  yearID  payroll_mean  standardized_payroll
0      ATL    1985  6.730455e+05              1.946823
1      BAL    1985  5.254869e+05              0.500248
2      BOS    1985  4.359024e+05             -0.377984
3      CAL    1985  5.152819e+05              0.400205
4      CHA    1985  4.688656e+05             -0.054832
..     …       …          …                    …
855    SLN    2014  4.310464e+06             -0.089295
856    TBA    2014  2.907564e+06             -0.508067
857    TEX    2014  4.677294e+06              0.020206
858    TOR    2014  4.396804e+06             -0.063522
859    WAS    2014  4.399456e+06             -0.062730

[860 rows x 4 columns]
```

[621]:
```python
df1 = pandaz.merge(q1, q2, how='outer', on=['yearID','teamID'])
```

[622]:
```python
year_intervals = [1989,1994,1999,2004,2009,2015]
eras = ['1990-1994','1995-1999','2000-2004','2005-2009','2010-2015']
#Bin the values in the yearID col of df1 - segment and sort data into bins
categories = pandaz.cut(df1['yearID'],year_intervals,labels=eras)
df1['categories'] = pandaz.cut(df1['yearID'], year_intervals, labels=eras)
df2 = df1.groupby('categories')
```

[623]:
```python
#Problem 6
# select unique teamIDs from the 'group2' DataFrame
group1 = df2.get_group('1990-1994')
teams = group1['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group1[['teamID', 'winning_percentage', 'standardized_payroll']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'standardized_payroll'], temp_result.
↪loc[t, 'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'standardized_payroll'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['standardized_payroll']
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Standardized Mean Payroll')
plt.ylabel('Mean Winning Percentage')
```

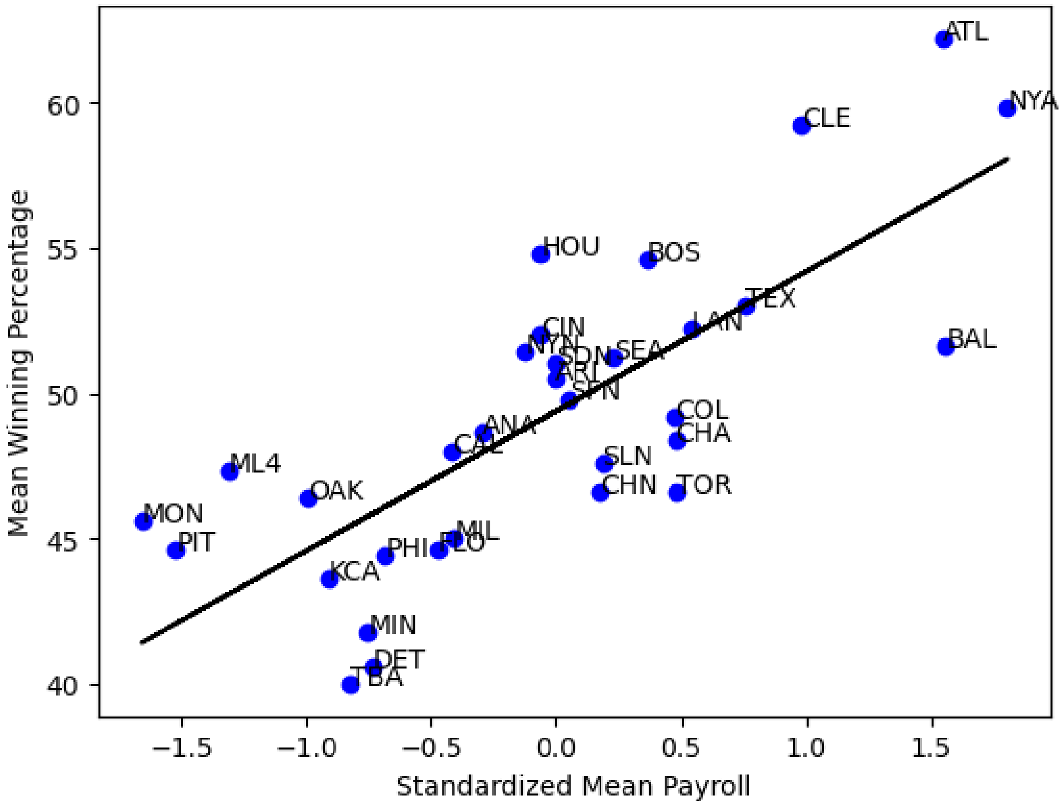
[623]:
```
Text(0, 0.5, 'Mean Winning Percentage')
```



[624]:
```
# select unique teamIDs from the 'group2' DataFrame
group2 = df2.get_group('1995-1999')
teams = group2['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group2[['teamID', 'winning_percentage', 'standardized_payroll']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'standardized_payroll'], temp_result.
↪loc[t, 'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'standardized_payroll'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['standardized_payroll']
```

```
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Standardized Mean Payroll')
plt.ylabel('Mean Winning Percentage')
```

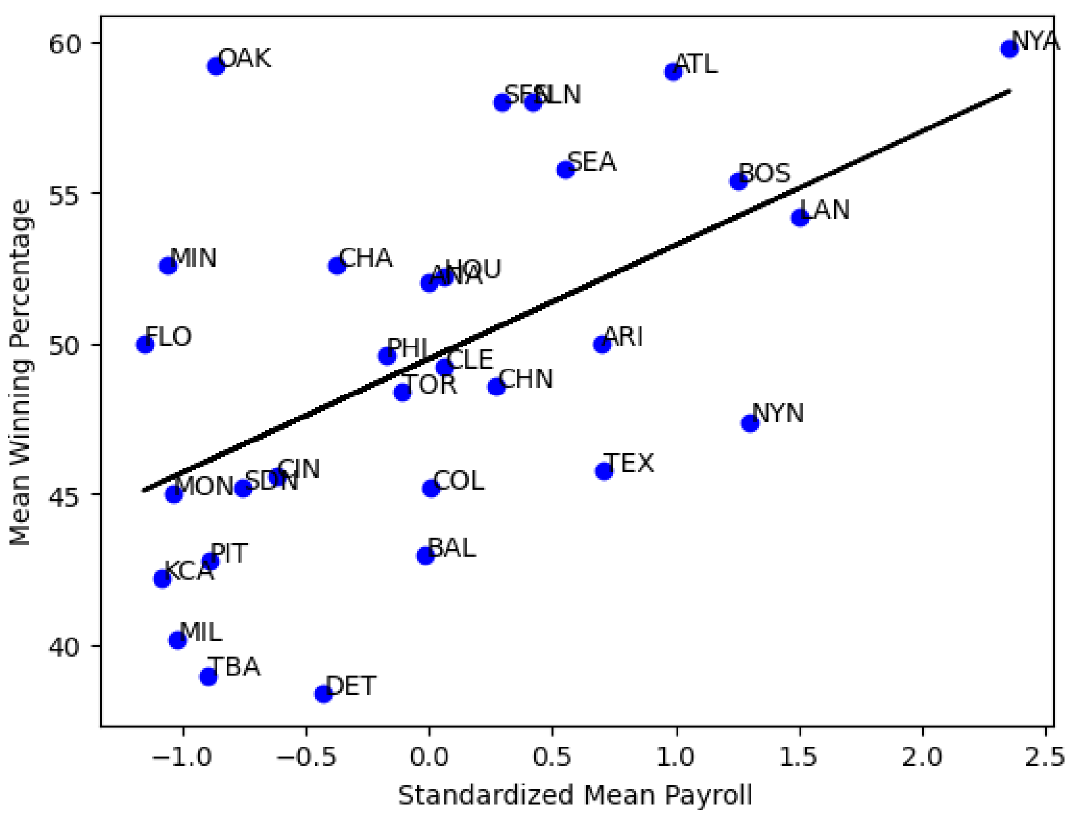
[624]: Text(0, 0.5, 'Mean Winning Percentage')



[625]:
```
# select unique teamIDs from the 'group2' DataFrame
group3 = df2.get_group('2000-2004')
teams = group3['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
 ↪percentage, and mean payroll
temp_result = group3[['teamID', 'winning_percentage', 'standardized_payroll']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
```

```
    # add a label to the plot for the current team at the coordinates of the mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'standardized_payroll'], temp_result.loc[t, 'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'standardized_payroll'], temp_result.loc[t, 'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['standardized_payroll']
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Standardized Mean Payroll')
plt.ylabel('Mean Winning Percentage')
```

[625]: Text(0, 0.5, 'Mean Winning Percentage')

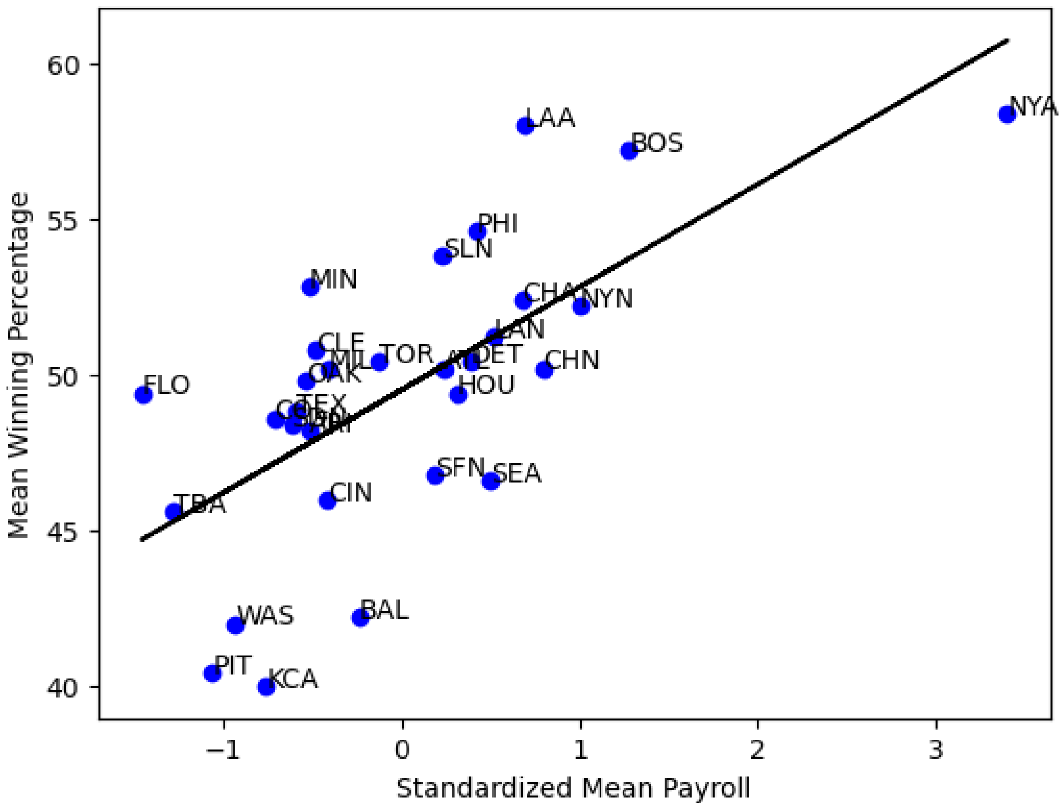


[626]:

```python
# select unique teamIDs from the 'group2' DataFrame
group4 = df2.get_group('2005-2009')
teams = group4['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group4[['teamID', 'winning_percentage', 'standardized_payroll']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'standardized_payroll'], temp_result.
↪loc[t, 'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'standardized_payroll'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['standardized_payroll']
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Standardized Mean Payroll')
plt.ylabel('Mean Winning Percentage')
```

[626]:

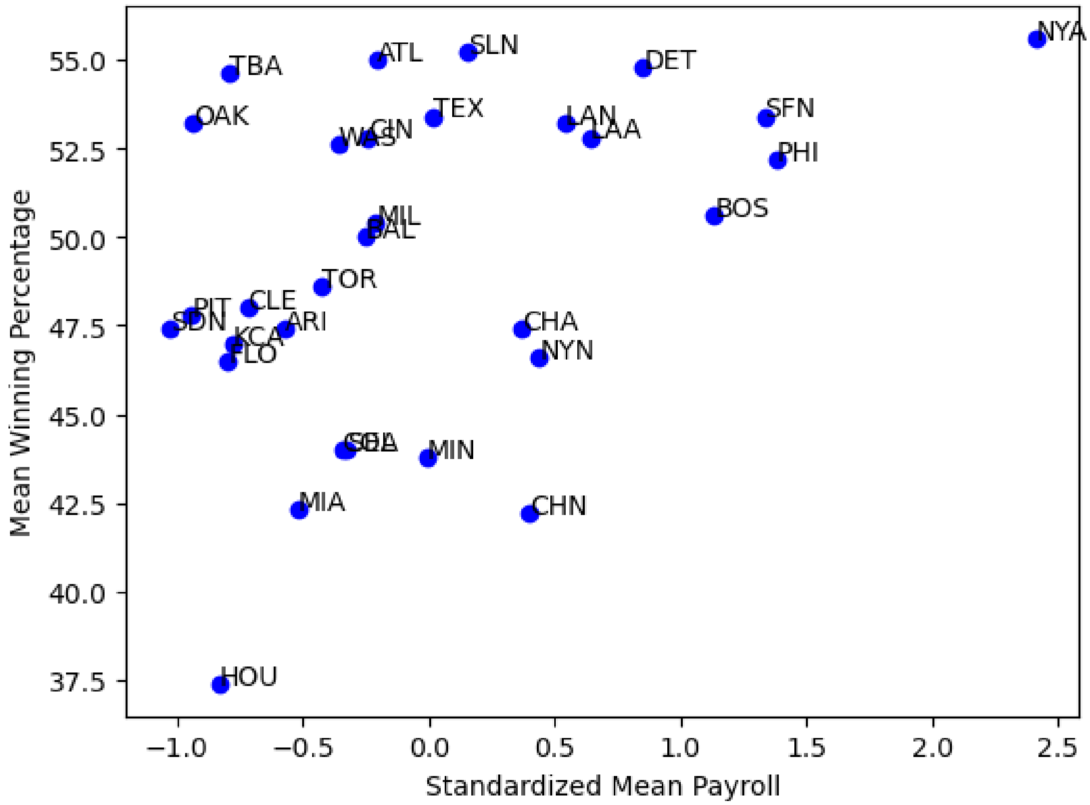
```
Text(0, 0.5, 'Mean Winning Percentage')
```



[627]:
```
# select unique teamIDs from the 'group2' DataFrame
group5 = df2.get_group('2010-2015')
teams = group5['teamID'].drop_duplicates()

# create a new DataFrame called 'temp_result' containing teamID, winning␣
↪percentage, and mean payroll
temp_result = group5[['teamID', 'winning_percentage', 'standardized_payroll']]
temp_result = temp_result.groupby('teamID').mean()

# loop through each team in the 'teams' list
for t in teams:
    # add a label to the plot for the current team at the coordinates of the␣
↪mean winning percentage and mean payroll
    plt.annotate(t, xy=(temp_result.loc[t, 'standardized_payroll'], temp_result.
↪loc[t, 'winning_percentage']))
    # plot a scatter plot for the current team
    plt.plot(temp_result.loc[t, 'standardized_payroll'], temp_result.loc[t,␣
↪'winning_percentage'], 'o', color="b")

# add a regression line to the scatter plot
x = temp_result['standardized_payroll']
```

```
y = temp_result['winning_percentage']
m, b = numpy.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='black')

# set the x and y labels to the appropriate column names from 'temp_result'
plt.xlabel('Standardized Mean Payroll')
plt.ylabel('Mean Winning Percentage')
```

[627]: Text(0, 0.5, 'Mean Winning Percentage')

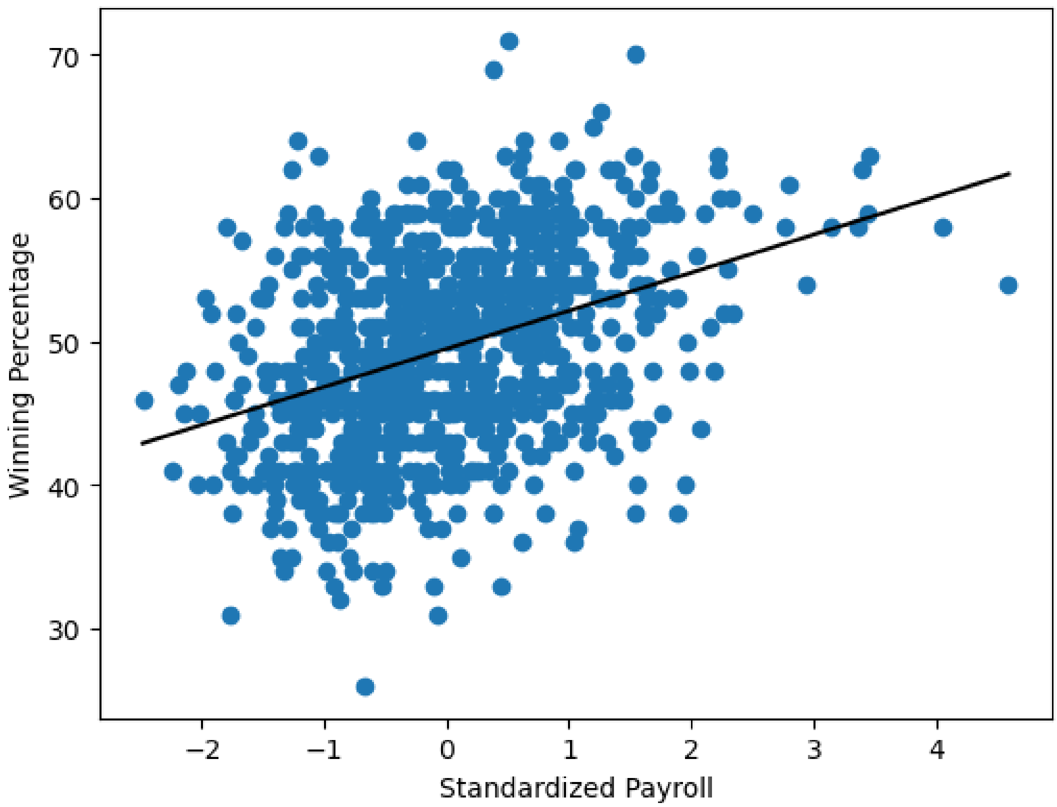


[605]:
```
# Question 3
# The plots from problem 4 and problem 6 are very similar.
# I would say that the spread is a little tigher with
# the new payroll variable.  Standardizing a par variable
# can be useful when the absolute values of pay are not directly
# comparable across different time periods or groups, however,
# the absolute values were valid therefore there is not too much
# change after we use our new variable.
```

[634]:
```
# Problem 7
```

```
#Drop rows with missing values
df1_clean = df1.dropna(subset=['winning_percentage', 'standardized_payroll'])

#Compute slope and intercept
x = df1_clean['standardized_payroll']
y = df1_clean['winning_percentage']
n = len(df1_clean)
x_mean = x.mean()
y_mean = y.mean()

slope = ((n * (x * y).sum()) - (x.sum() * y.sum())) / ((n * (x ** 2).sum()) -␣
↪(x.sum() ** 2))
intercept = y_mean - (slope * x_mean)
points = np.linspace(df1_clean['standardized_payroll'].min(),␣
↪df1_clean['standardized_payroll'].max(), 100)

#Plot the data
plt.plot(x, y, 'o')
plt.plot(points, slope * points + intercept, color='black')

#Add axis labels
plt.xlabel('Standardized Payroll')
plt.ylabel('Winning Percentage')
```

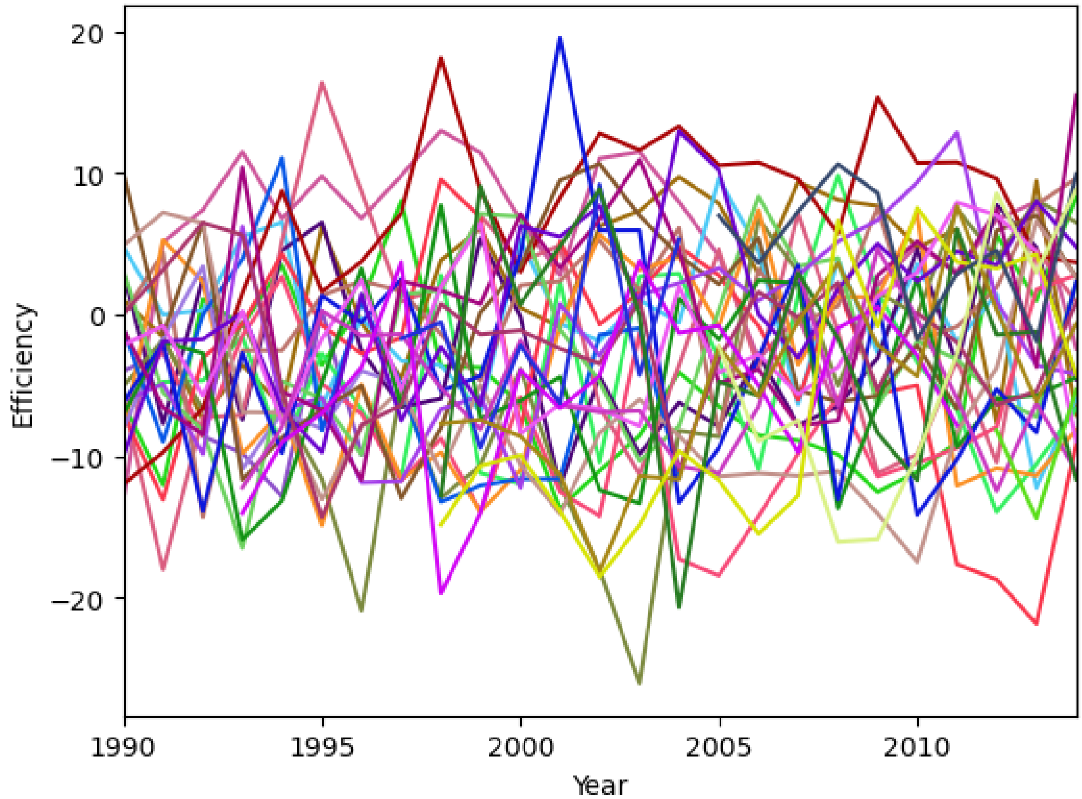
[634]:
```
Text(0, 0.5, 'Winning Percentage')
```




```
[640]: ## Problem 8
#Calculate efficiency scor
efficiency = group_table['winning_percentage'] - 50 + 2.5 *␣
 ↪group_table['Standardized_Payroll']
group_table['efficiency'] = efficiency

#Filter data for years between 1990 and 2014
temp_result = group_table[(group_table['yearID'] >= 1990) &␣
 ↪(group_table['yearID'] <= 2014)]
temp_result = temp_result[['yearID', 'teamID', 'efficiency']]

#Loop through each team and plot its efficiency score for each year from 1990␣
 ↪to 2014
for team in temp_result['teamID'].unique():
    team_data = temp_result[temp_result['teamID'] == team]
    plt.plot(team_data['yearID'], team_data['efficiency'], color=numpy.random.
 ↪rand(3,))
for year, score in zip(team_data['yearID'], team_data['efficiency']):
    plt.annotate(team, xy=(year, score))

#Set axis labels and limit the x-axis to the range 1990-2014
```

```python
plt.xlabel('Year')
plt.ylabel('Efficiency')
plt.xlim(1990, 2014)
```

[640]: (1990.0, 2014.0)



[641]:
```python
# Question 4
# We learn from this plot that Oaklands spending efficiency peaked around␣
 ↪2000-2002 but
# fluctuates other years similar to that of the other teams.  Teams that pay␣
 ↪more for
# their team are more RELIABLY efficient (for example, NYC), unlike one off␣
 ↪cases like
# Oakland.
```

[ ]: