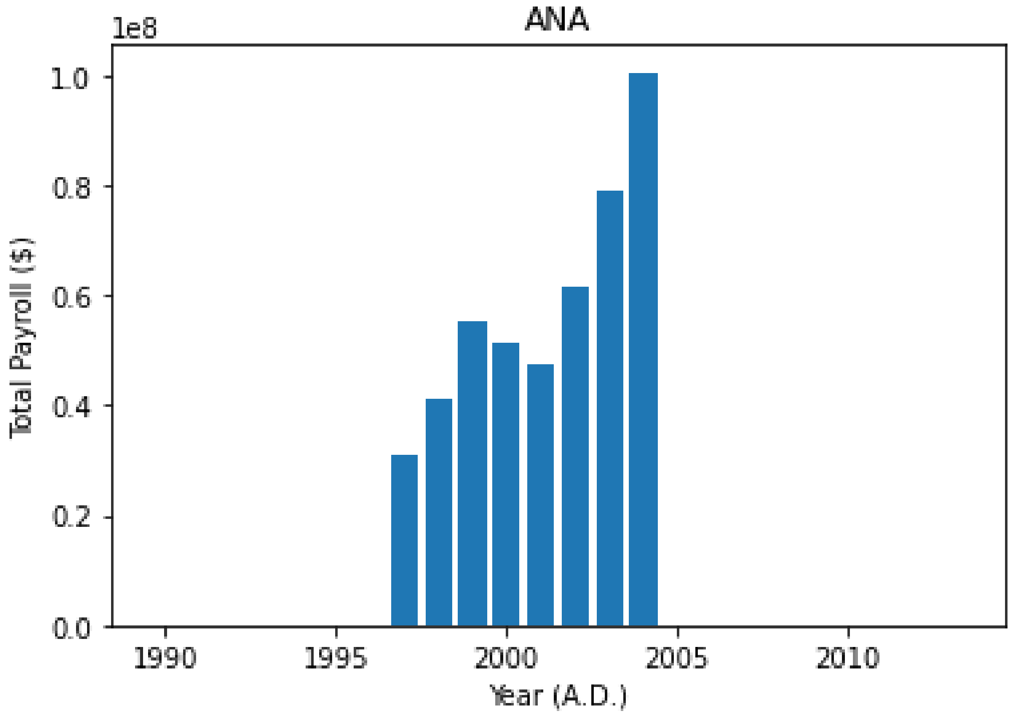
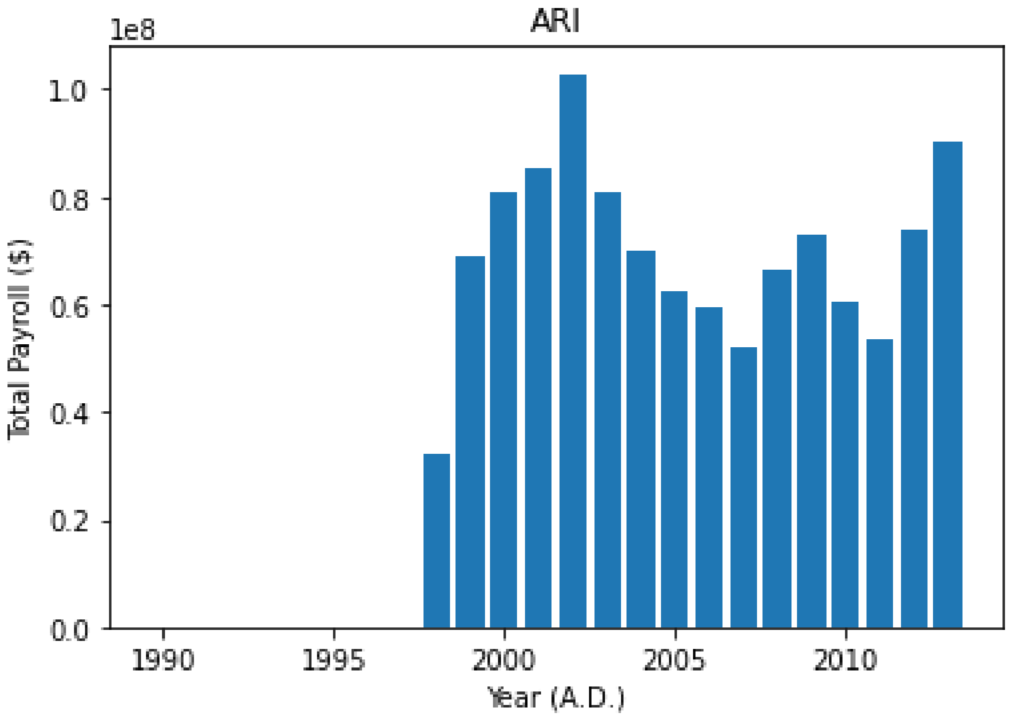
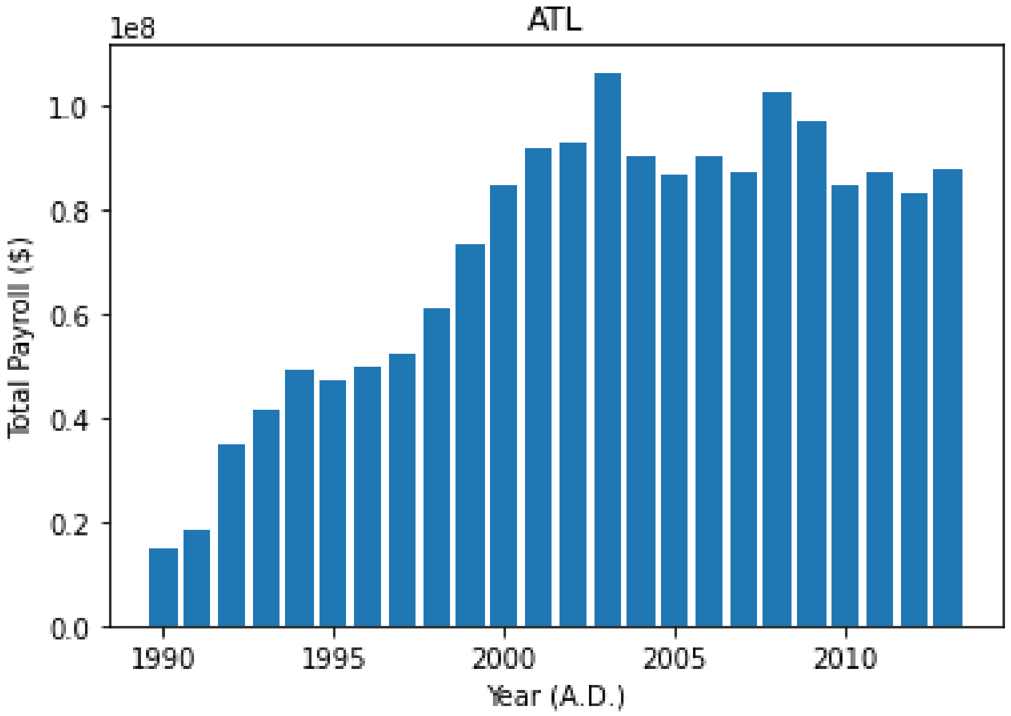
```
import pandas as pd
import sqlite3
import matplotlib as mpl
#---------------------------------PART 1:
WRANGLING------------------------------------------------

#Problem 1

sqlite_file = 'lahman2014.sqlite'
conn = sqlite3.connect(sqlite_file)
teamSalaryQuery = "SELECT teamID, yearID, sum(salary) as total_payroll
FROM Salaries GROUP BY teamID,yearID ORDER BY teamID"
MLBsalaries = pd.read_sql(teamSalaryQuery, conn)
extraInfoQuery = "SELECT teamID, yearID,sum(W) as wins, sum(W)+sum(L)
as total_games, 100*CAST(sum(W) AS float)/(sum(W)+sum(L)) as 'win_rate
%', franchID FROM teams \
            GROUP BY teamID,yearID ORDER BY teamID"
winfo = pd.read_sql(extraInfoQuery, conn)
# merging these two queries together removes any inputs with missing
data
custom_query = MLBsalaries.merge(winfo, how = "inner", left_on =
["teamID","yearID"], right_on = ["teamID","yearID"])
print(custom_query)
```

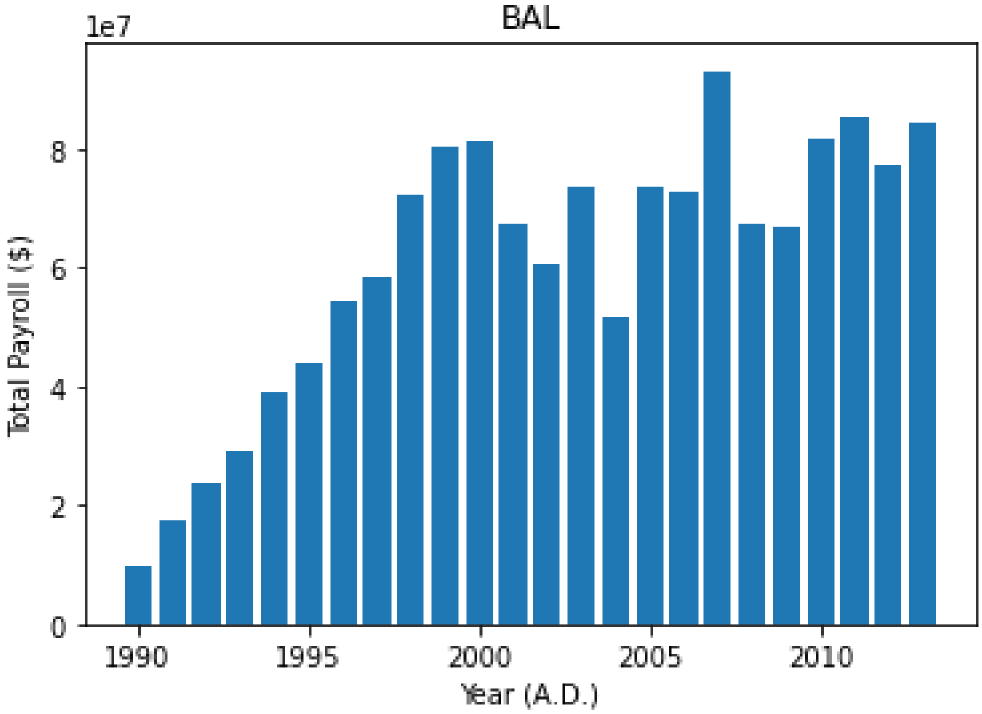
```
    teamID  yearID  total_payroll  wins  total_games  win_rate%
franchID
0      ANA    1997     31135472.0    84           162  51.851852
ANA
1      ANA    1998     41281000.0    85           162  52.469136
ANA
2      ANA    1999     55388166.0    70           162  43.209877
ANA
3      ANA    2000     51464167.0    82           162  50.617284
ANA
4      ANA    2001     47535167.0    75           162  46.296296
ANA
..     ...     ...            ...   ...          ...        ...      .
..
853    WAS    2010     61400000.0    69           162  42.592593
WSN
854    WAS    2011     63856928.0    80           161  49.689441
WSN
855    WAS    2012     80855143.0    98           162  60.493827
WSN
856    WAS    2013    113703270.0    86           162  53.086420
WSN
857    WAS    2014    131983680.0    96           162  59.259259
WSN

[858 rows x 7 columns]
```

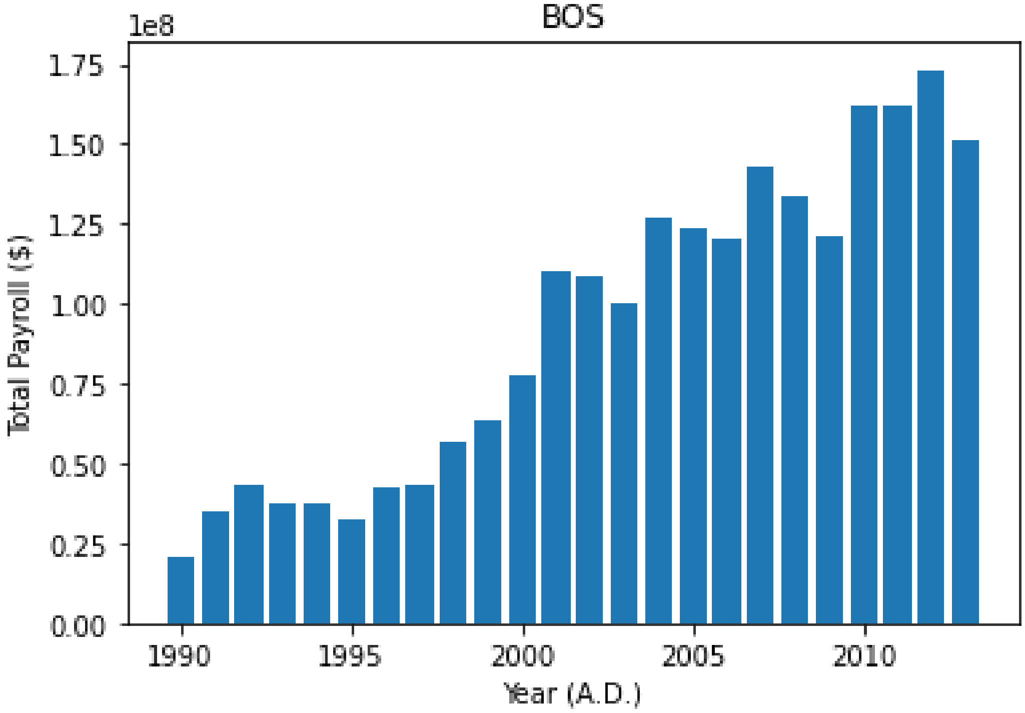
```
#------------------------PART 2: EXPLORATORY DATA
ANALYSIS----------------------
import matplotlib.pyplot as mplpp
import numpy as np
# Problem 2
MLBteams = np.unique(custom_query.iloc[:,0].values)
for corruptOwner in MLBteams:
    payThem_df = pd.DataFrame({'yearID':range(1990, 2014)})
    payThem_vars = custom_query[['yearID', 'teamID', 'total_payroll']]
    payThem_grouping =
payThem_vars.groupby(['teamID']).get_group(corruptOwner)
    payThem_table = payThem_df.merge(payThem_grouping, how = "left",
left_on=['yearID'], right_on=['yearID'])
    mplpp.bar(payThem_table['yearID'],
payThem_table['total_payroll'].fillna(value=0))
    mplpp.xlabel("Year (A.D.)")
    mplpp.ylabel("Total Payroll ($)")
    mplpp.title(corruptOwner)
    mplpp.show()
```

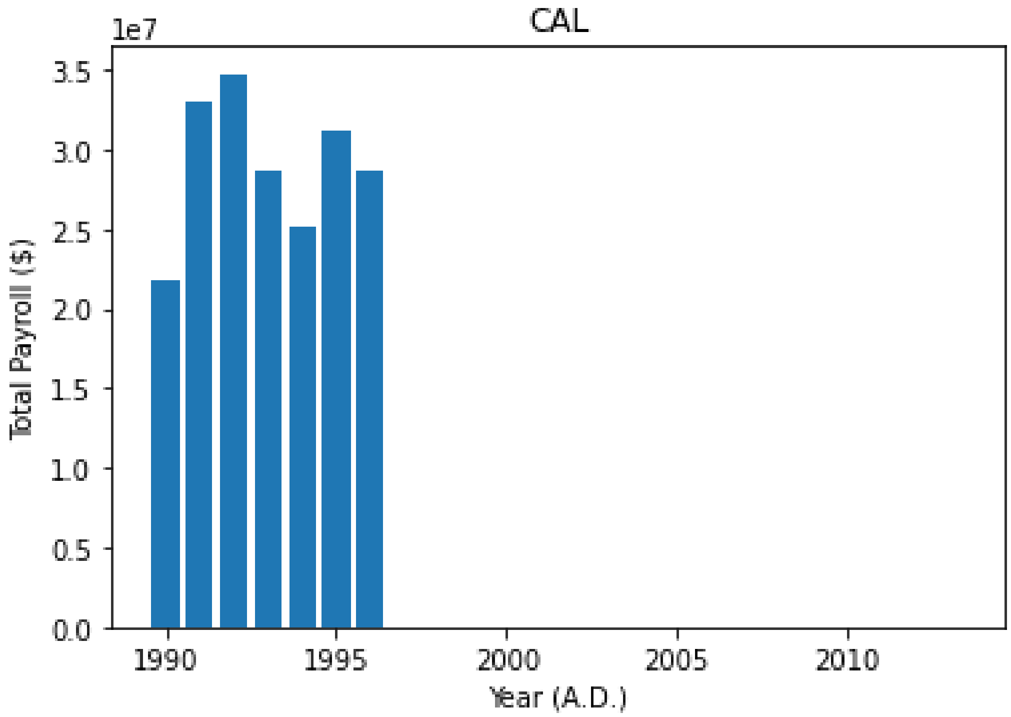


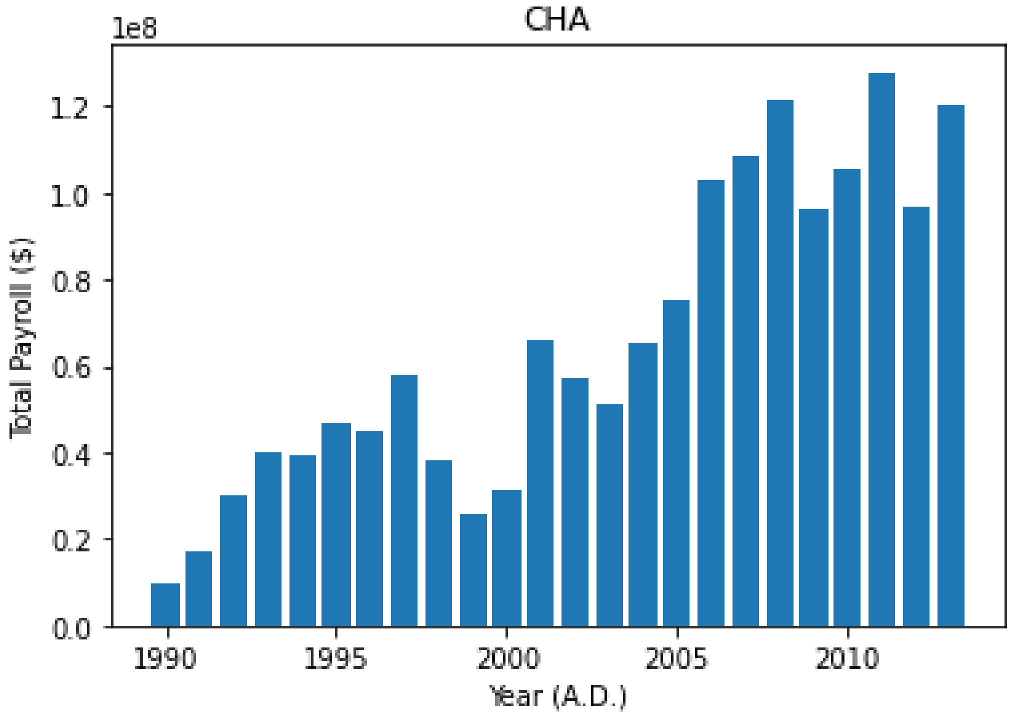

ARI
1e8
Total Payroll ($)
1.0
0.8
0.6
0.4
0.2
0.0
1990
1995
2000
2005
2010
Year (A.D.)

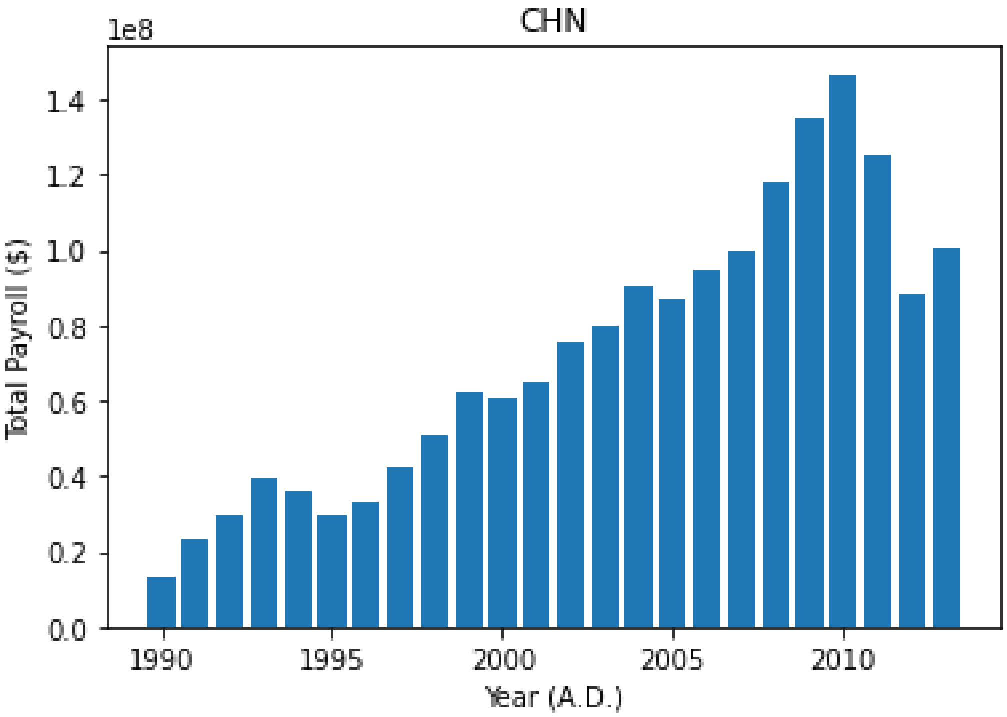

ATL
1e8
Total Payroll ($)
1.0
0.8
0.6
0.4
0.2
0.0
1990
1995
2000
2005
2010
Year (A.D.)

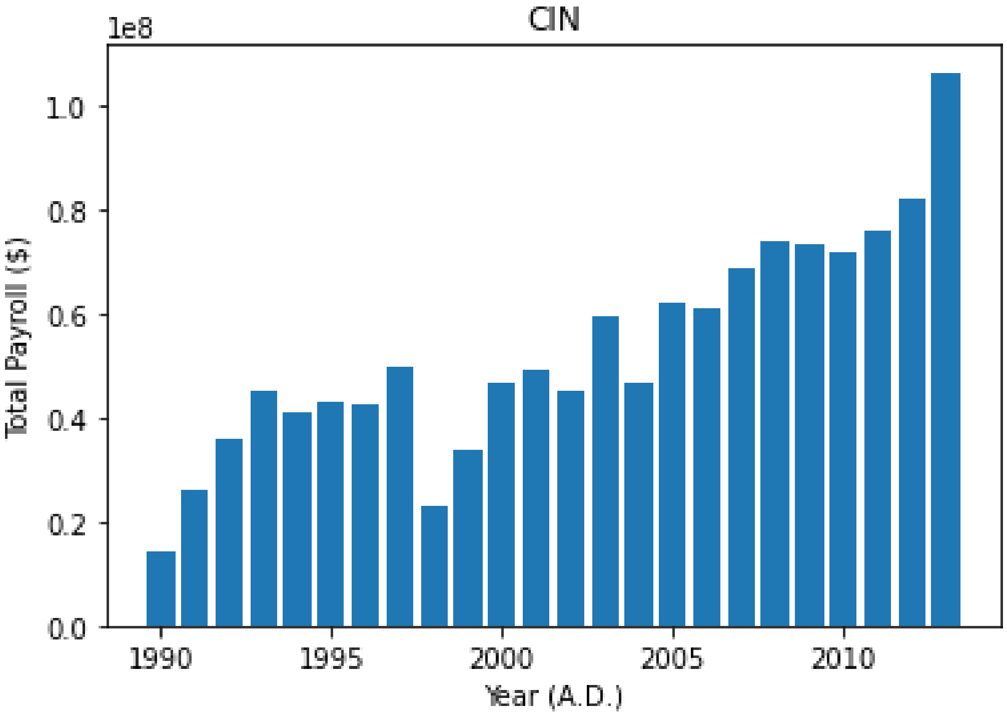

BAL
1e7
Total Payroll ($)
0
2
4
6
8
1990
1995
2000
2005
2010
Year (A.D.)

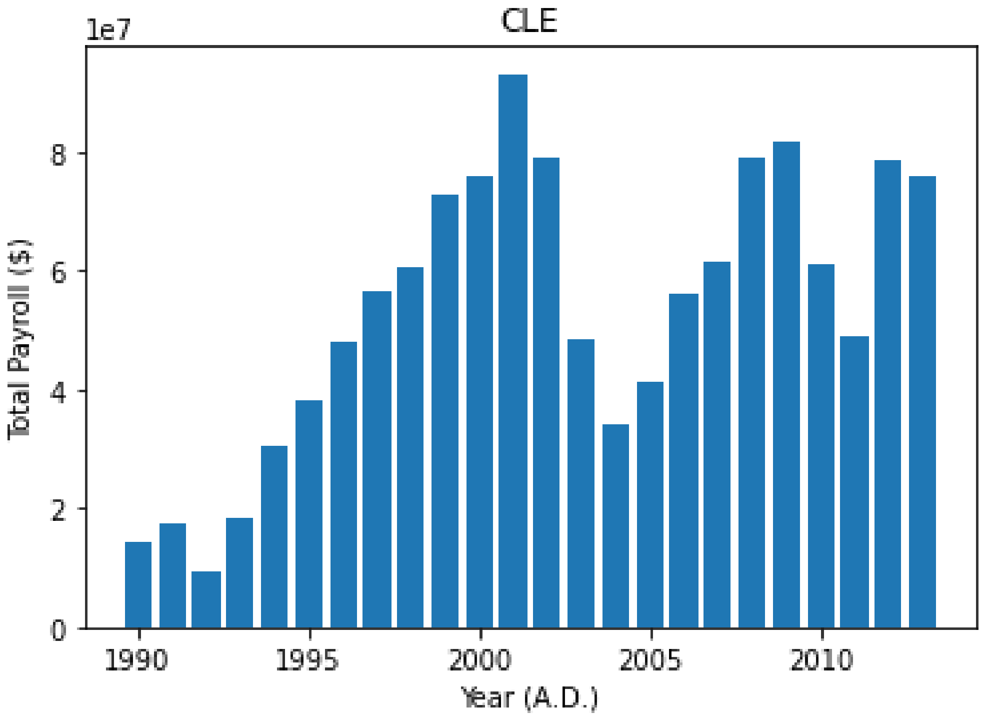

BOS
1e8
Total Payroll ($)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1990
1995
2000
2005
2010
Year (A.D.)

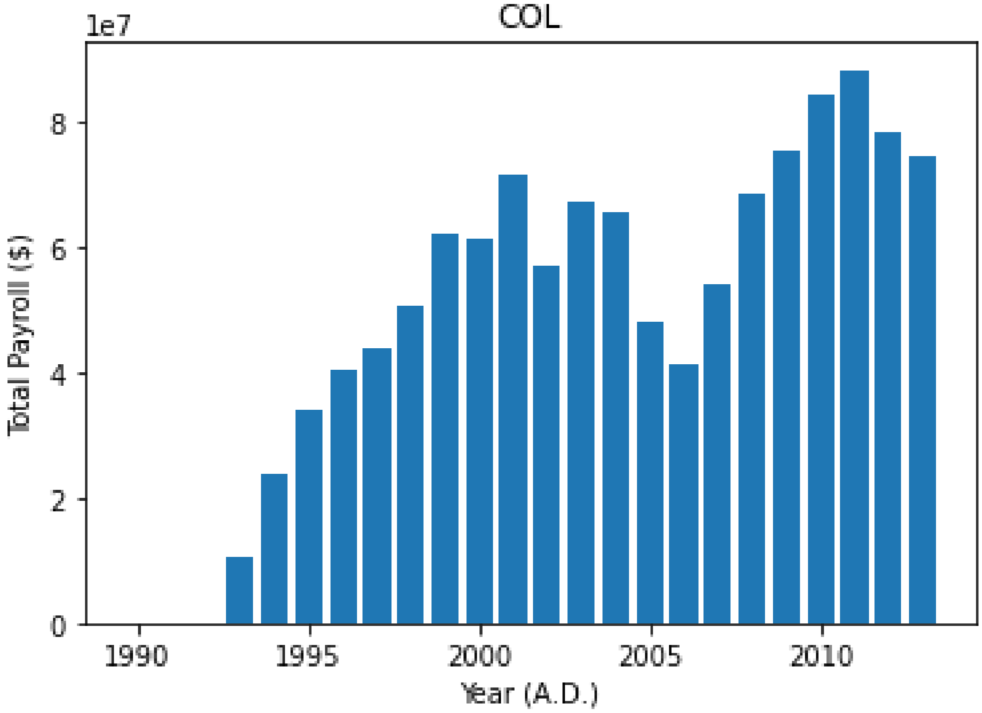

CAL
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

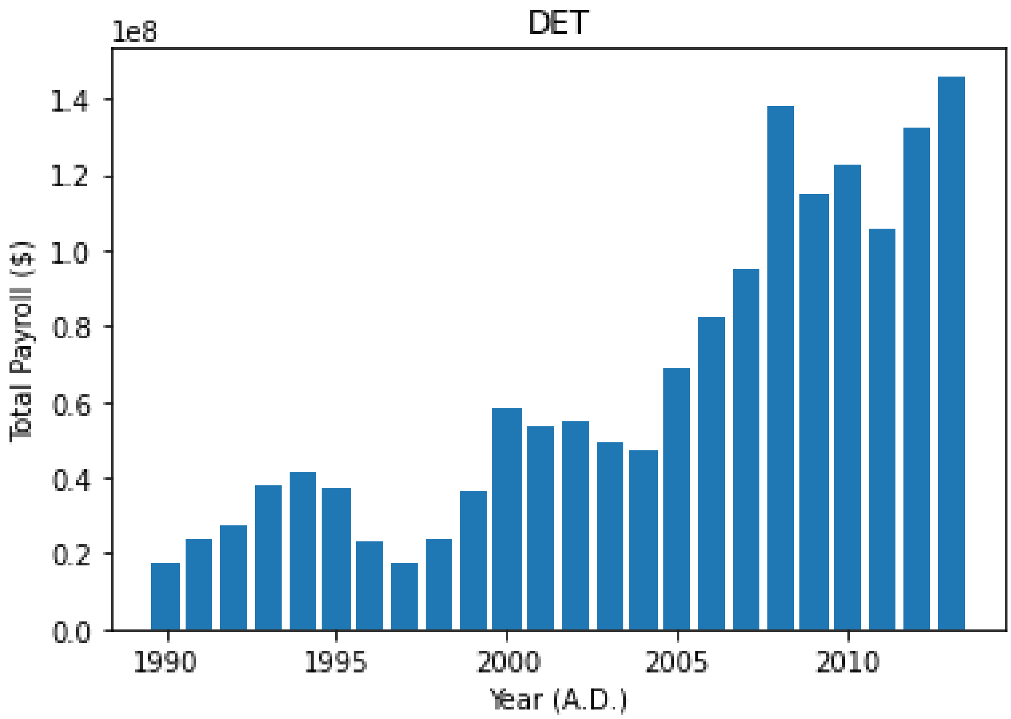

CHA
1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

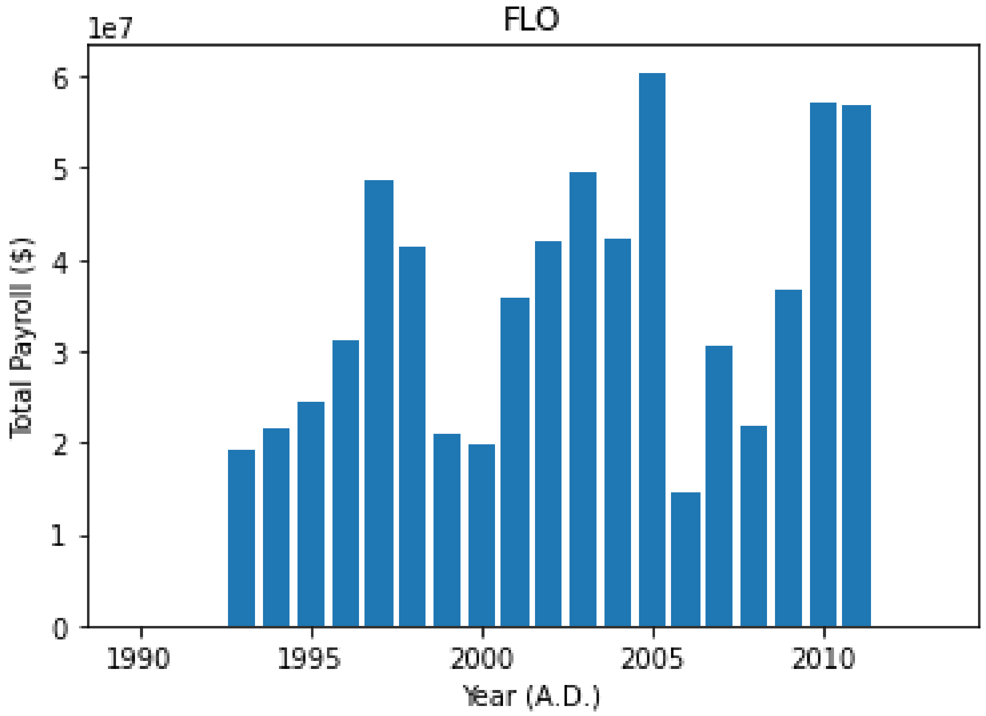

CHN
1e8
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

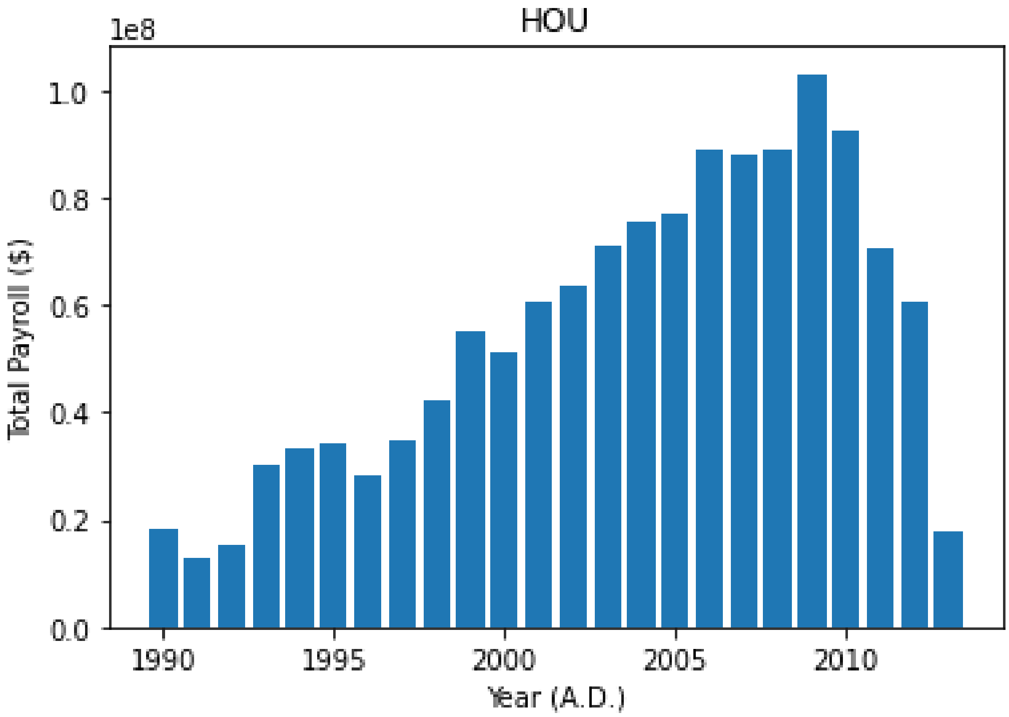

CIN
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

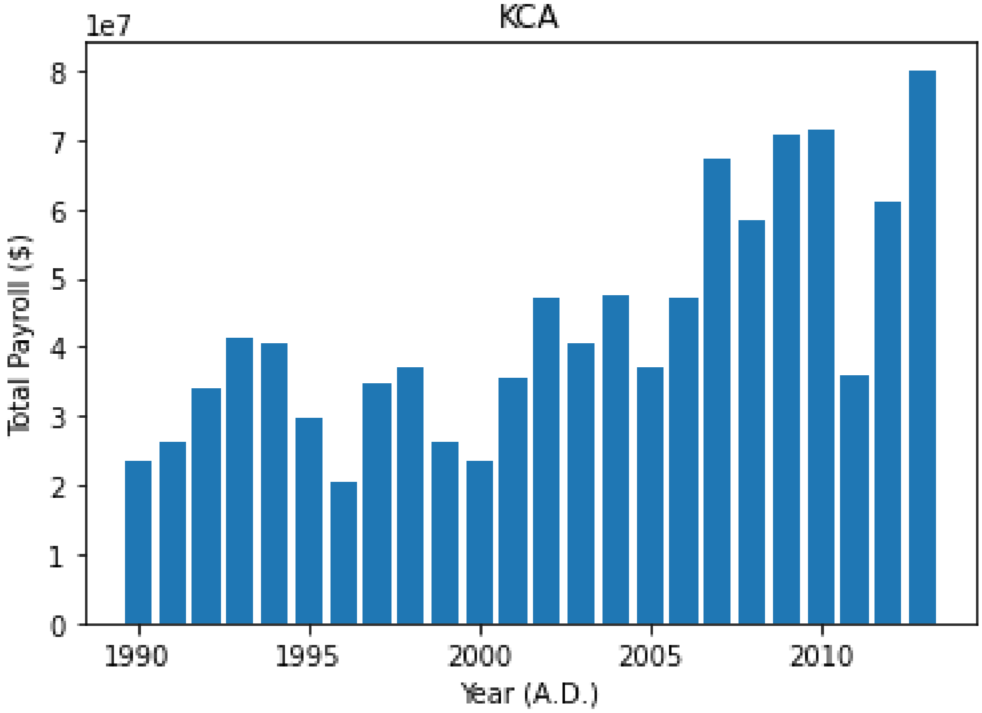

CLE
1e7
Total Payroll ($)
0
2
4
6
8
1990
1995
2000
2005
2010
Year (A.D.)

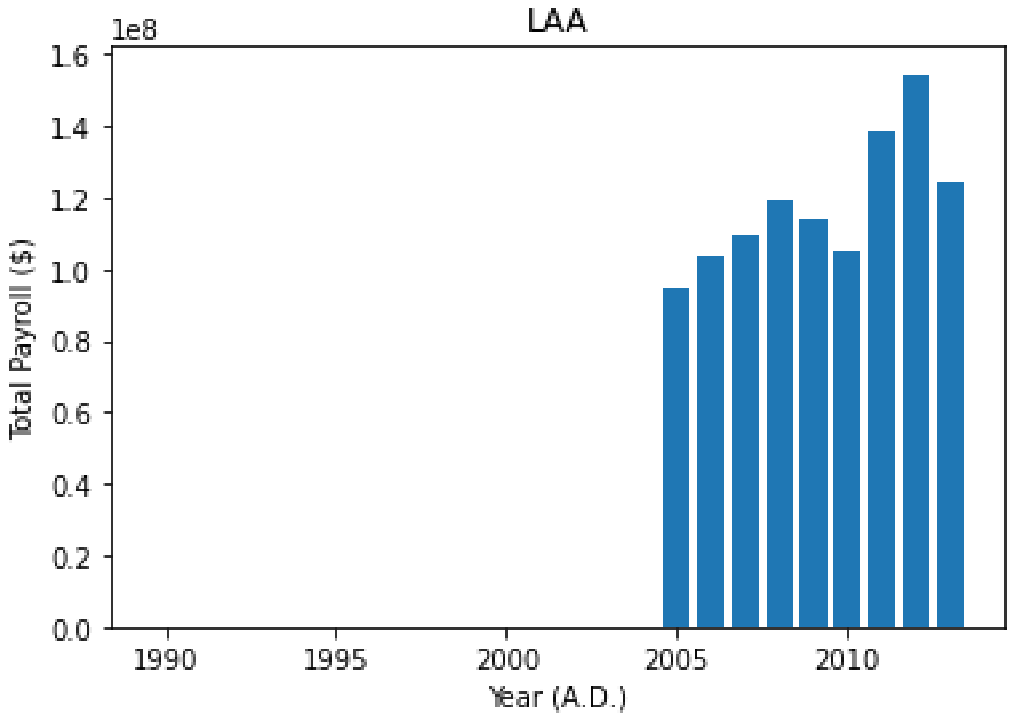

COL
1e7
Total Payroll ($)
0
2
4
6
8
1990
1995
2000
2005
2010
Year (A.D.)


DET
1e8
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

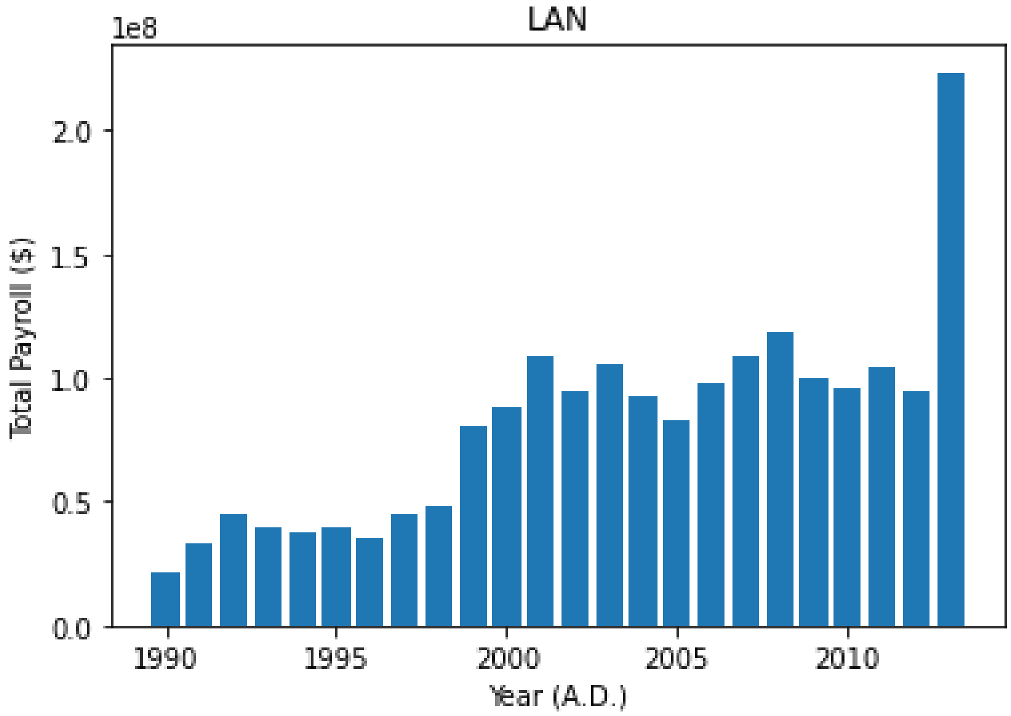

FLO
1e7
6
5
4
3
2
1
0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)


HOU
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

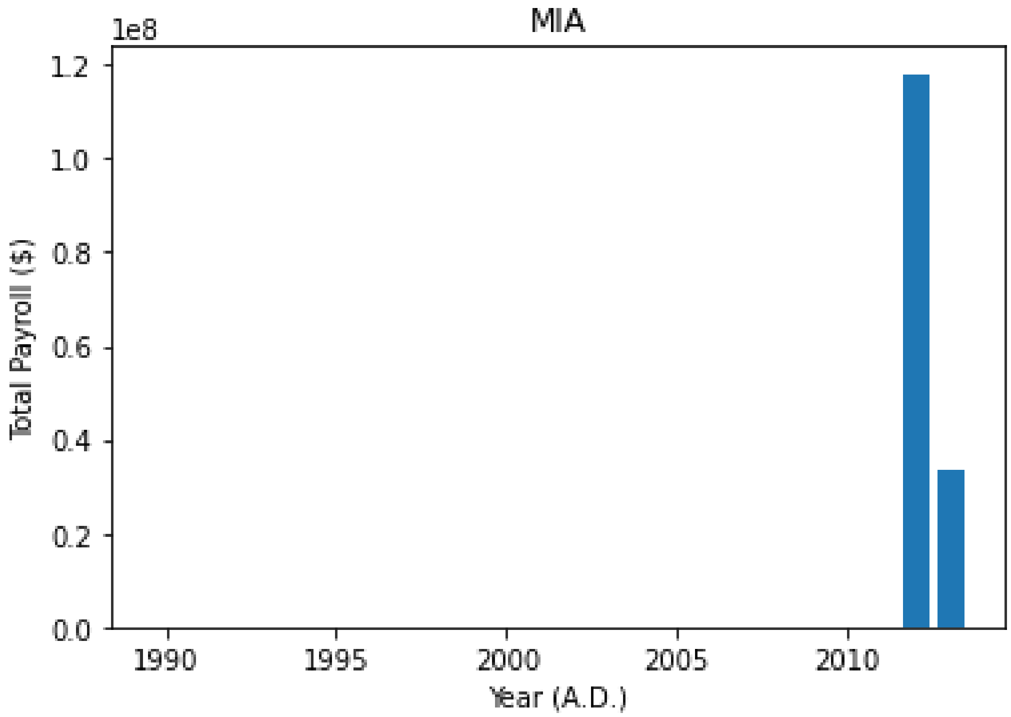

KCA
1e7
8
7
6
5
4
3
2
1
0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

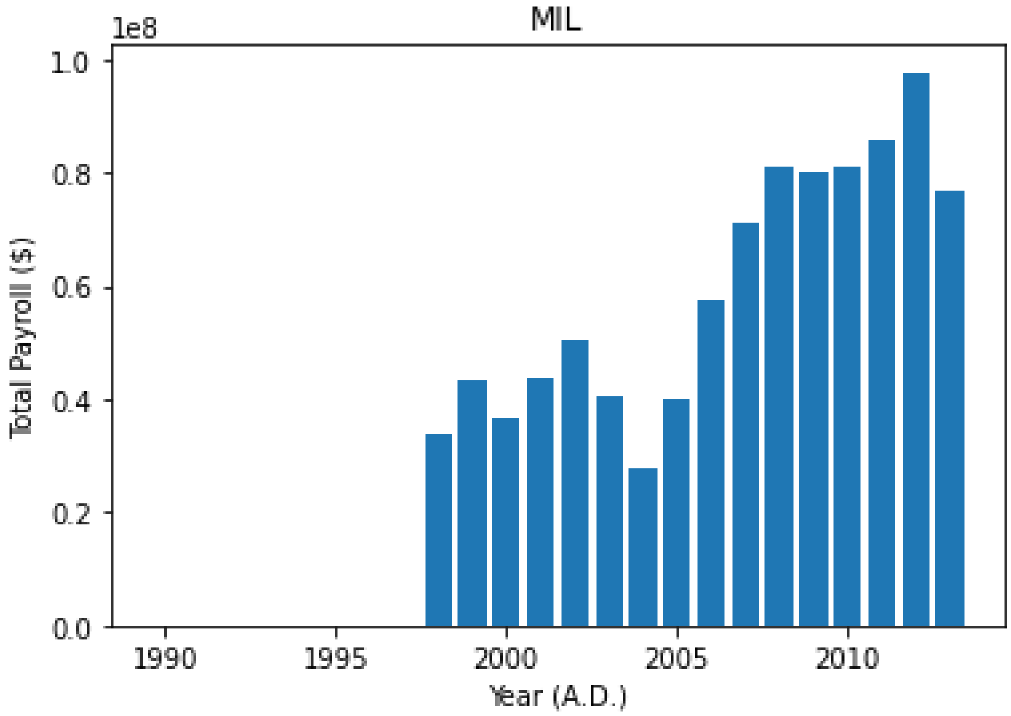

LAA

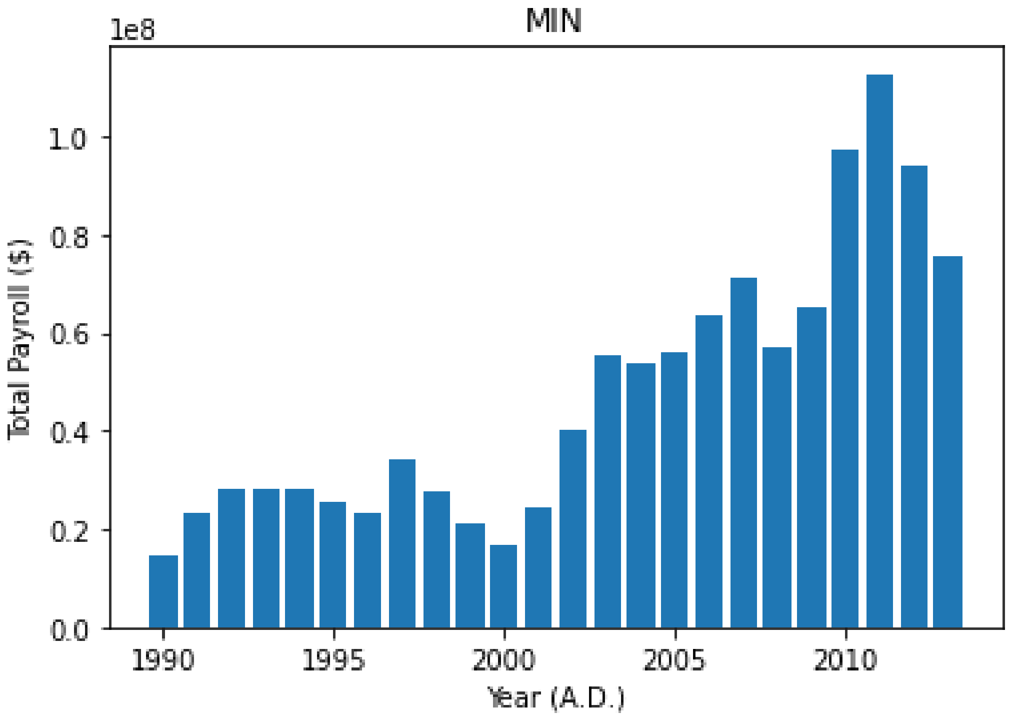
1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

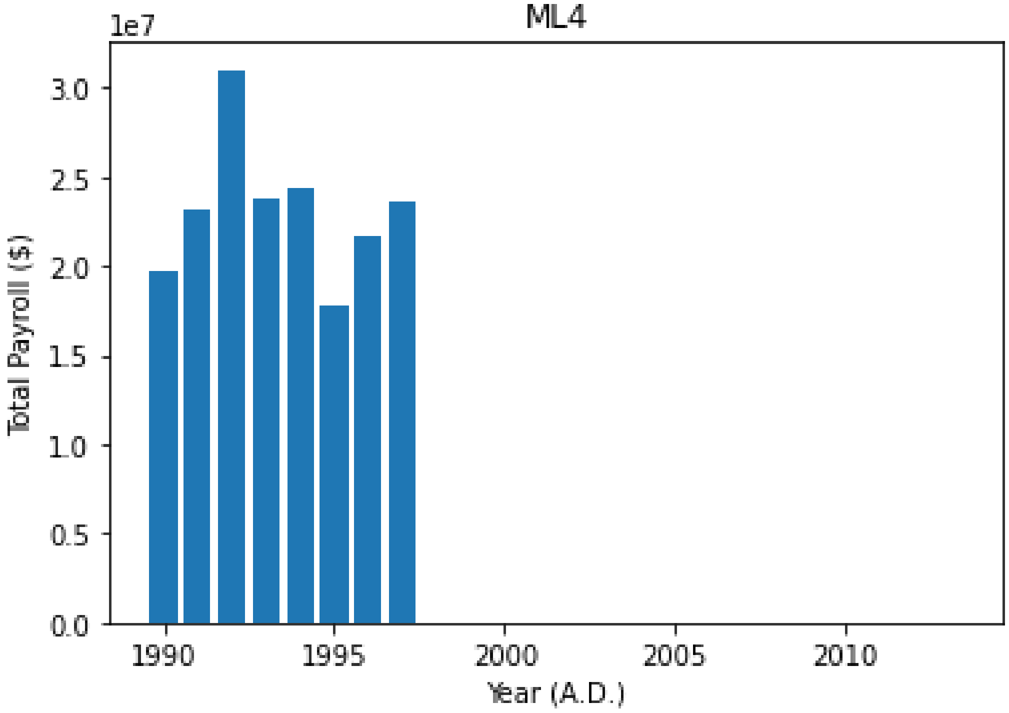

LAN

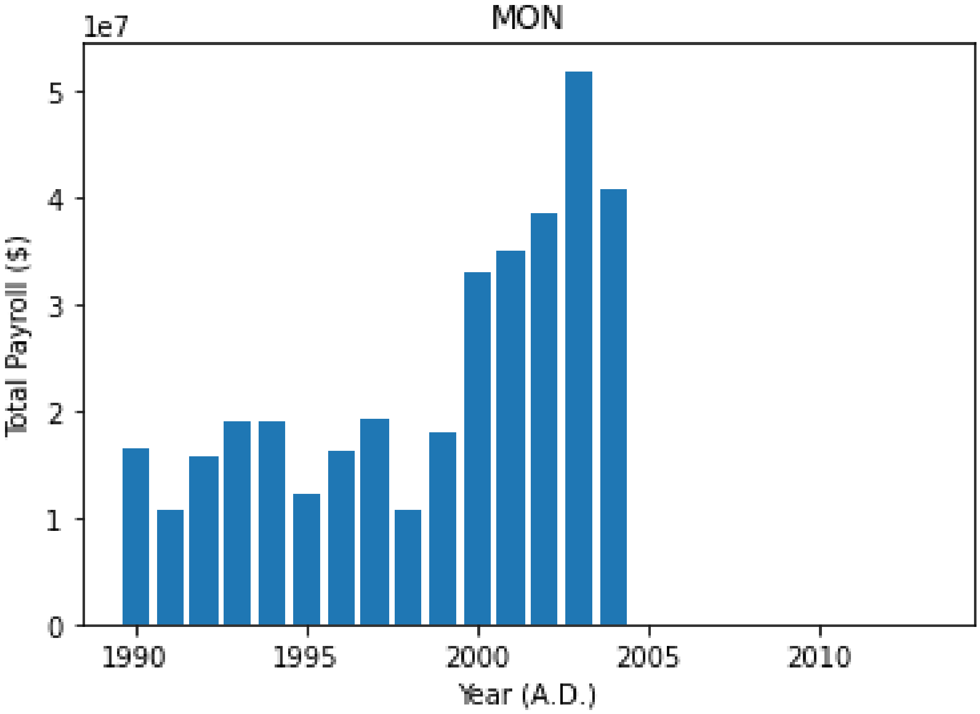
1e8
2.0
1.5
1.0
0.5
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

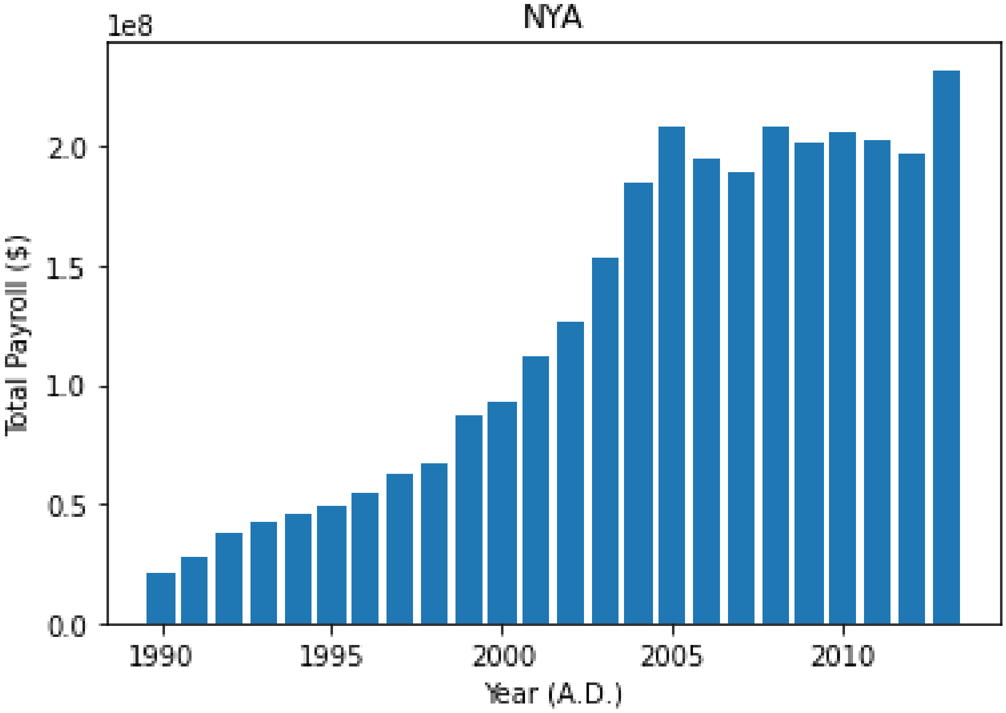

MIA
1e8
Total Payroll ($)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1990
1995
2000
2005
2010
Year (A.D.)

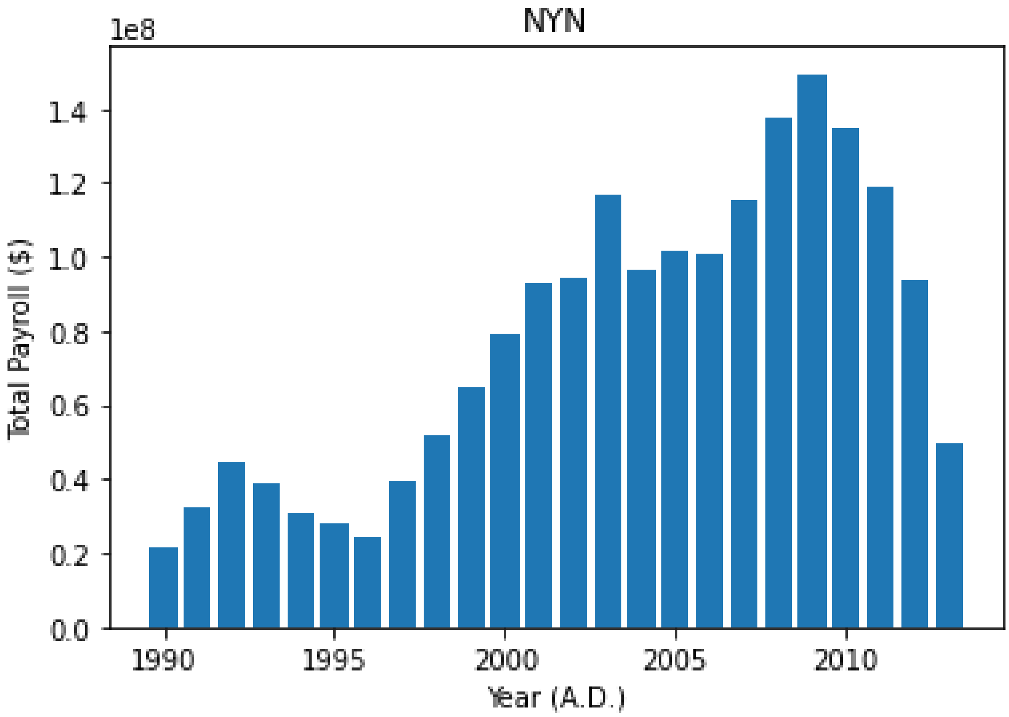

MIL
1e8
Total Payroll ($)
1.0
0.8
0.6
0.4
0.2
0.0
1990
1995
2000
2005
2010
Year (A.D.)

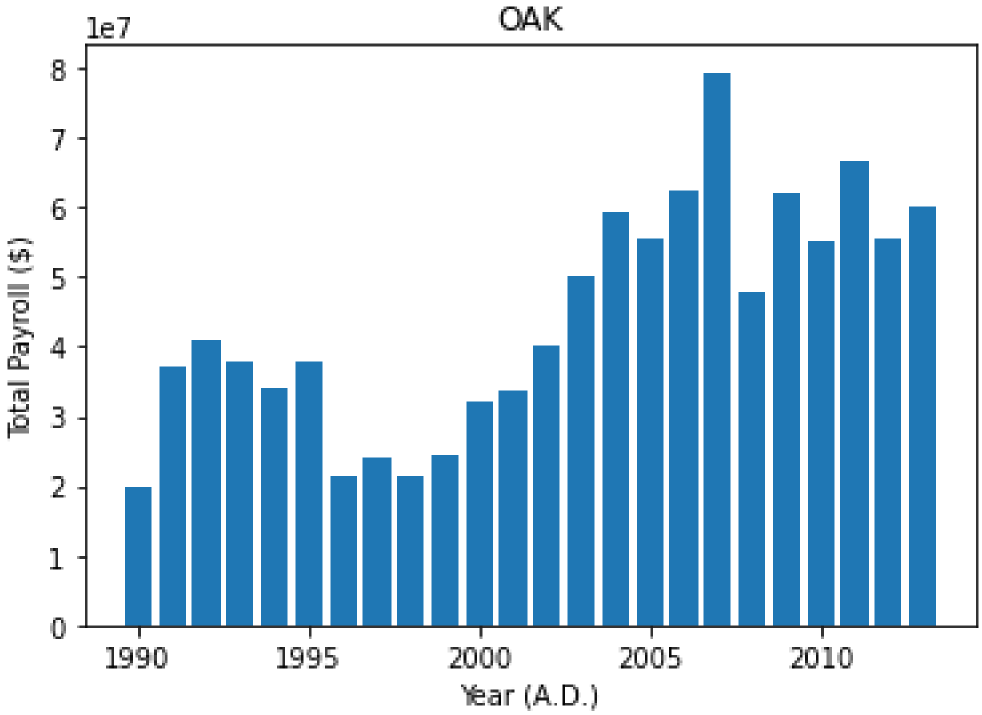

MIN
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

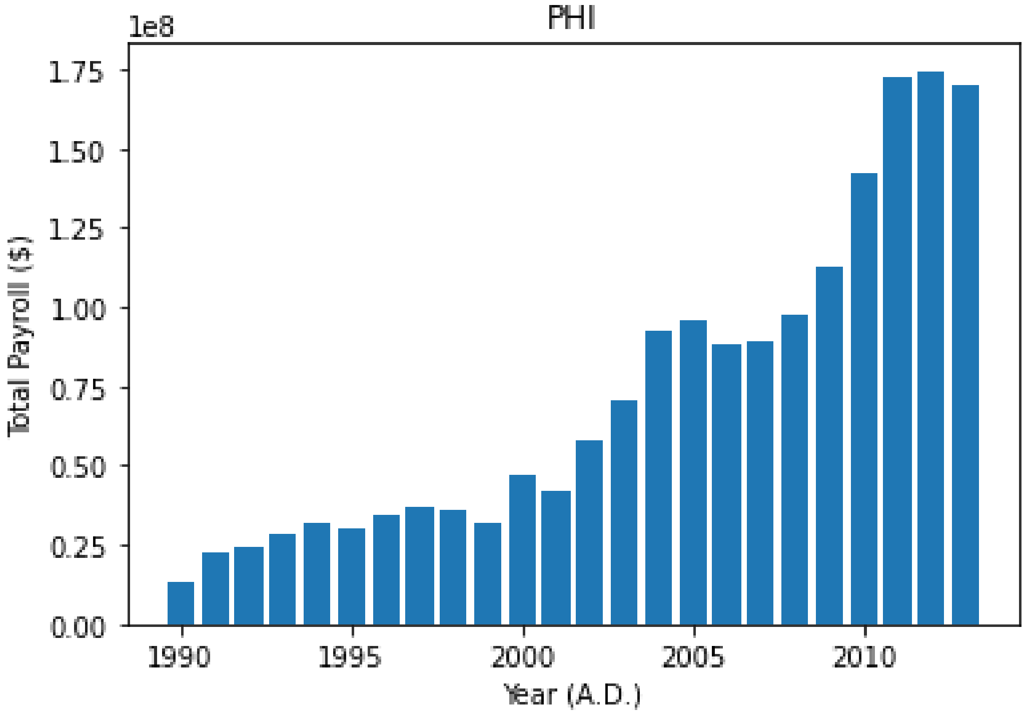

ML4
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

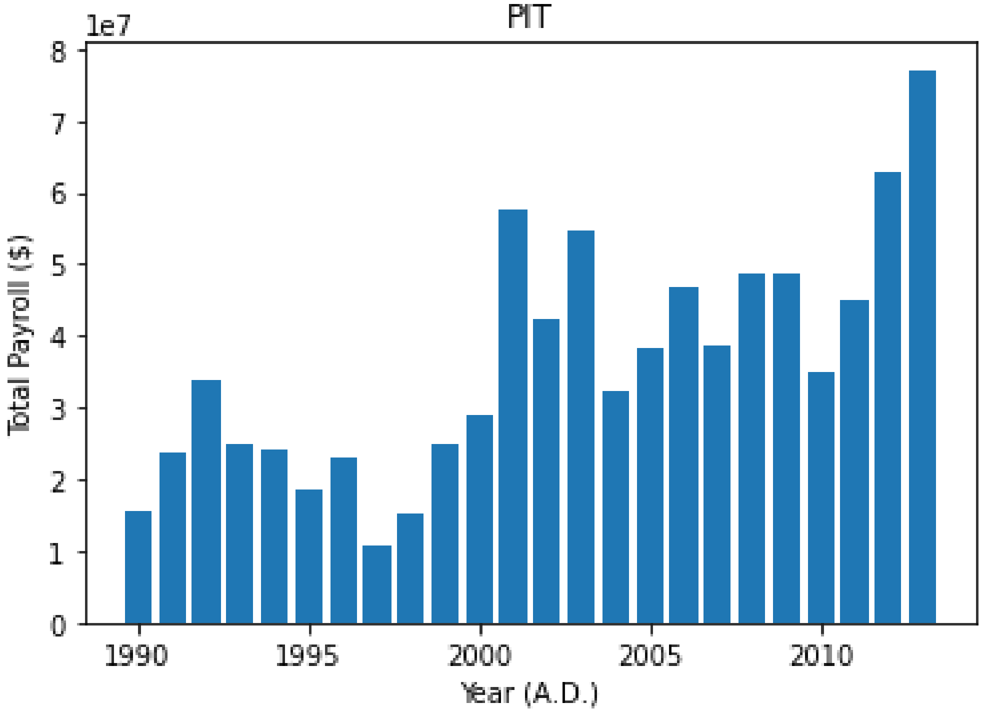

MON
1e7
Total Payroll ($)
0
1
2
3
4
5
1990
1995
2000
2005
2010
Year (A.D.)

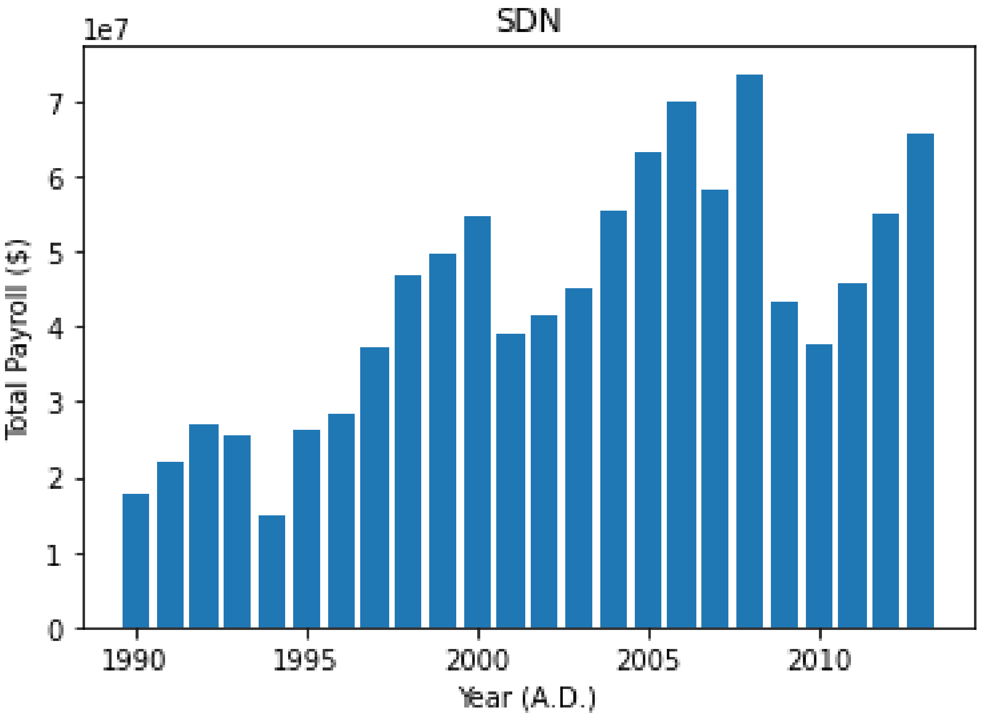

NYA
1e8
Total Payroll ($)
0.0
0.5
1.0
1.5
2.0
1990
1995
2000
2005
2010
Year (A.D.)

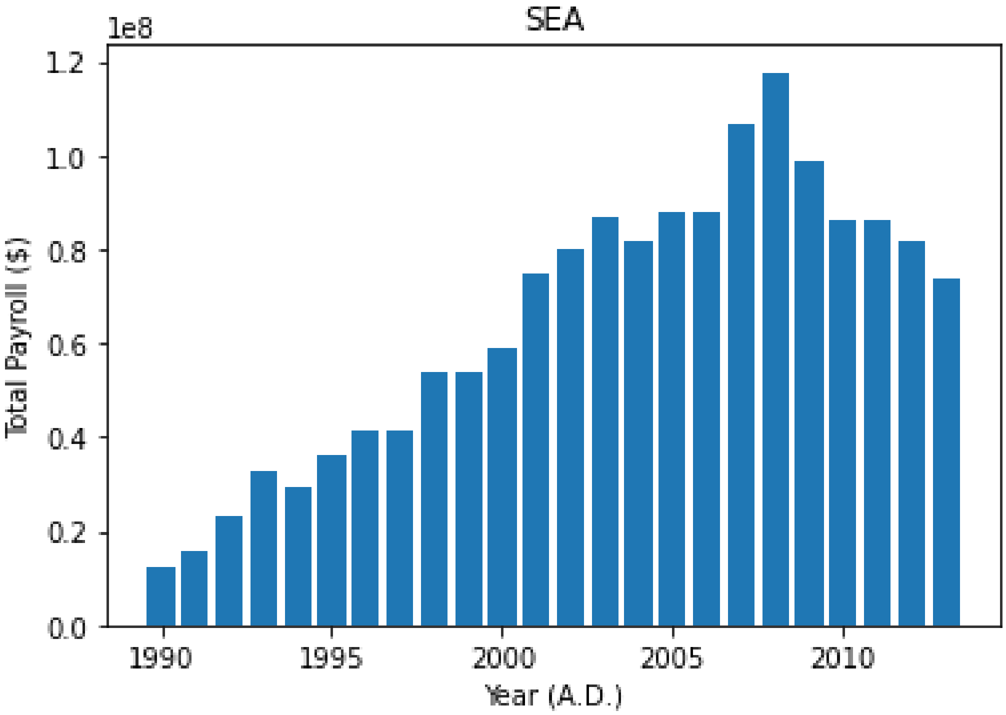

NYN
1e8
Total Payroll ($)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1990
1995
2000
2005
2010
Year (A.D.)

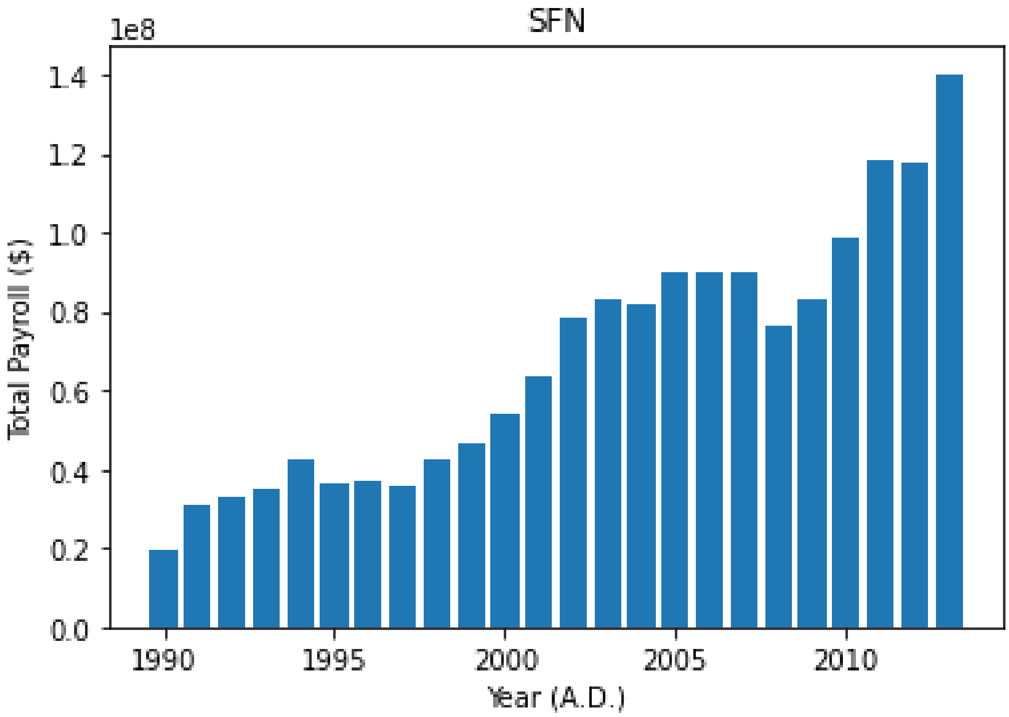

OAK
1e7
Total Payroll ($)
0
1
2
3
4
5
6
7
8
1990
1995
2000
2005
2010
Year (A.D.)

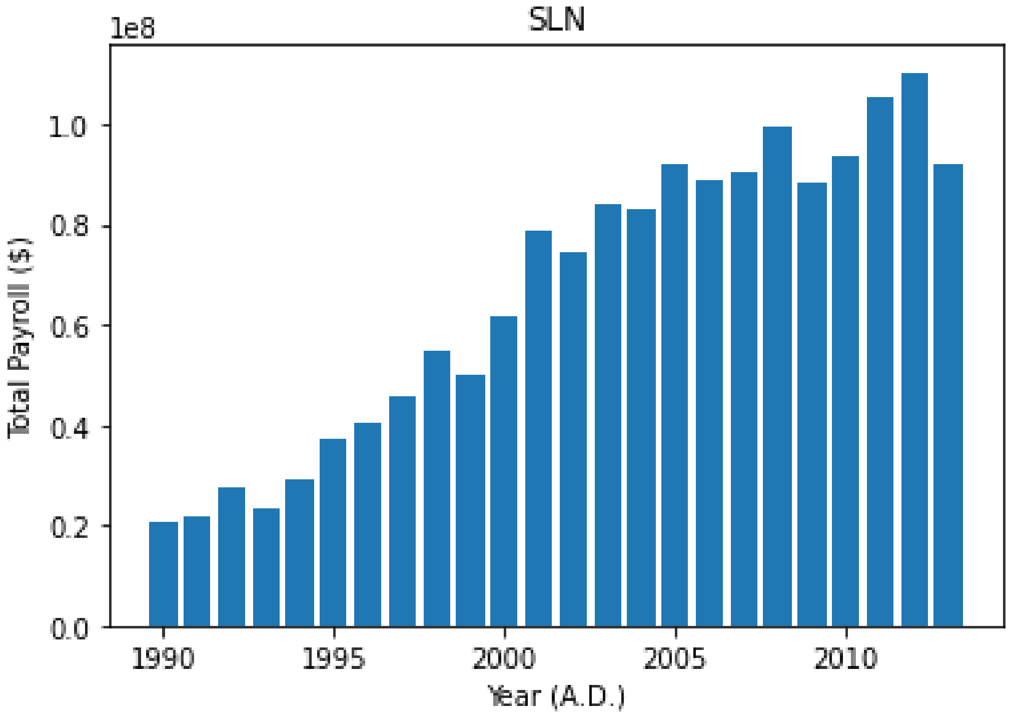

PHI
1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

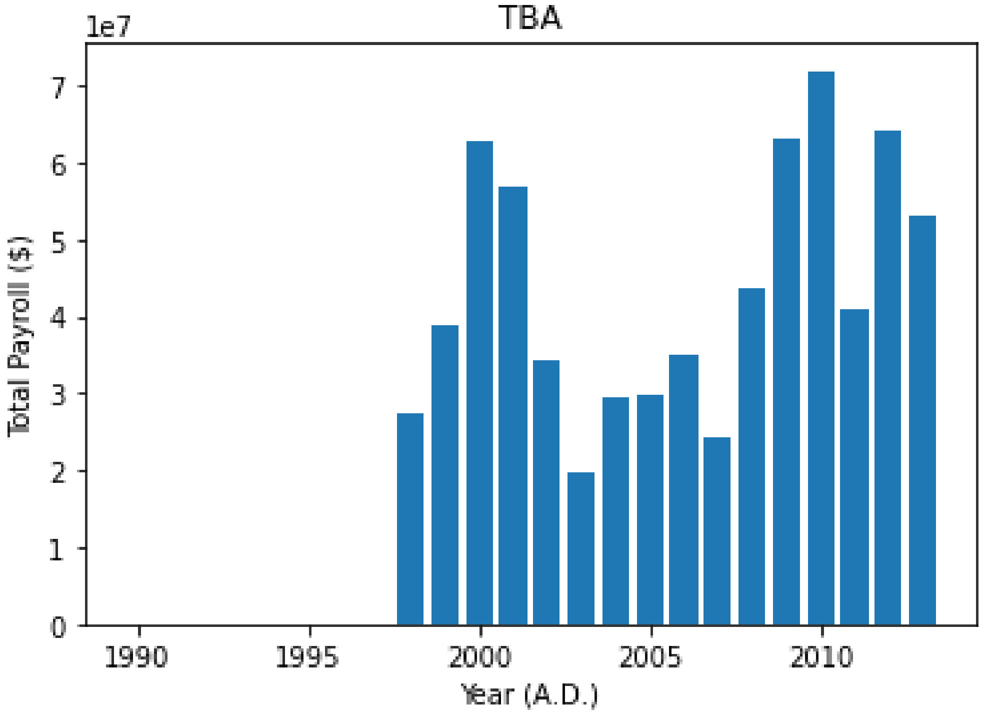

PIT
1e7
8
7
6
5
4
3
2
1
0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)

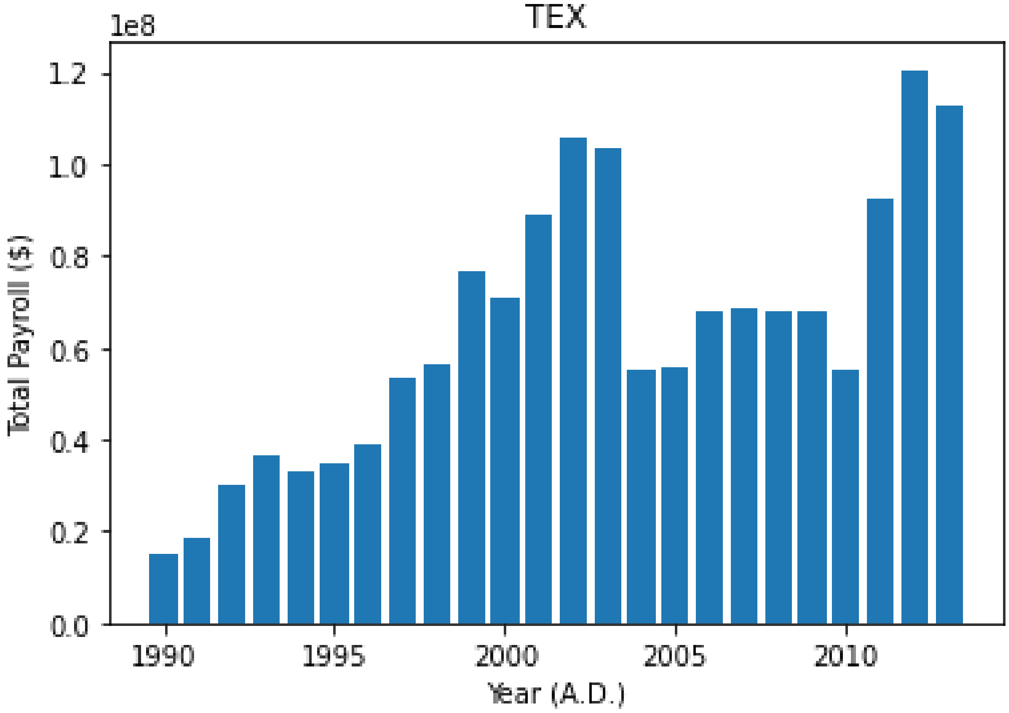

SDN
1e7
Total Payroll ($)
0
1
2
3
4
5
6
7
1990
1995
2000
2005
2010
Year (A.D.)

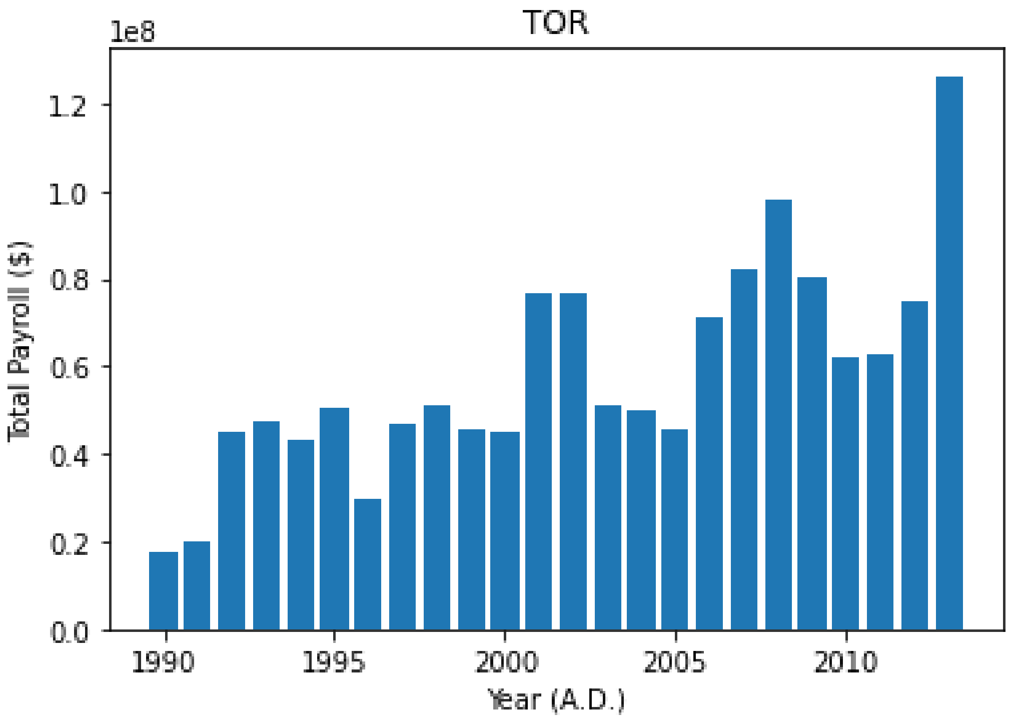

SEA
1e8
Total Payroll ($)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1990
1995
2000
2005
2010
Year (A.D.)

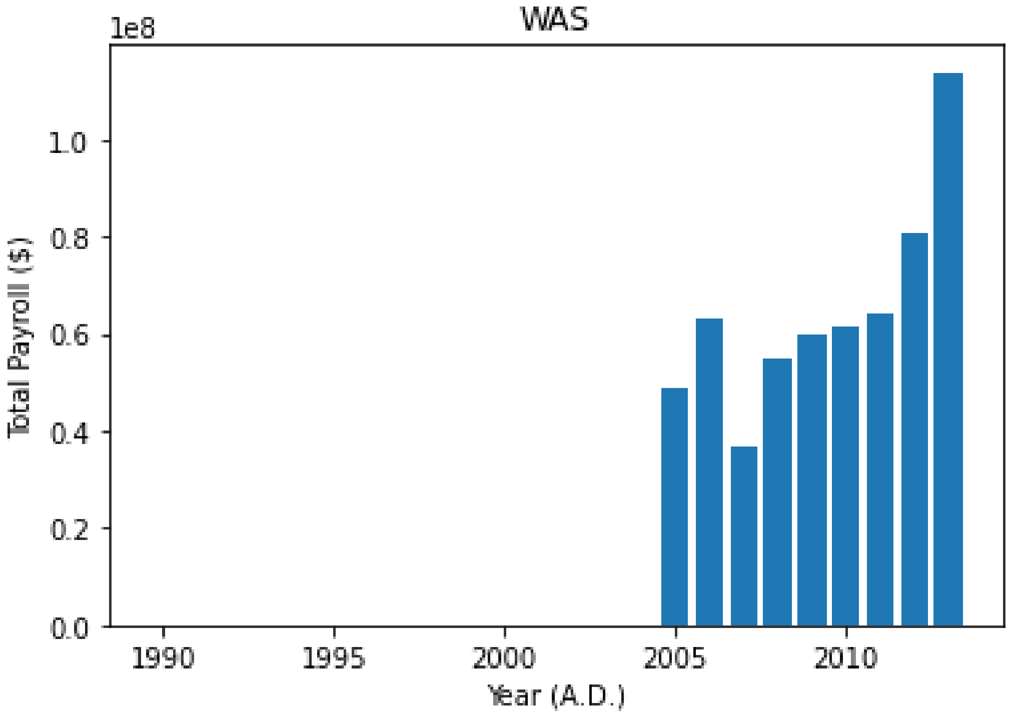

SFN
1e8
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)


SLN
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Total Payroll ($)
1990
1995
2000
2005
2010
Year (A.D.)


TBA
1e7
Total Payroll ($)
0
1
2
3
4
5
6
7
1990
1995
2000
2005
2010
Year (A.D.)

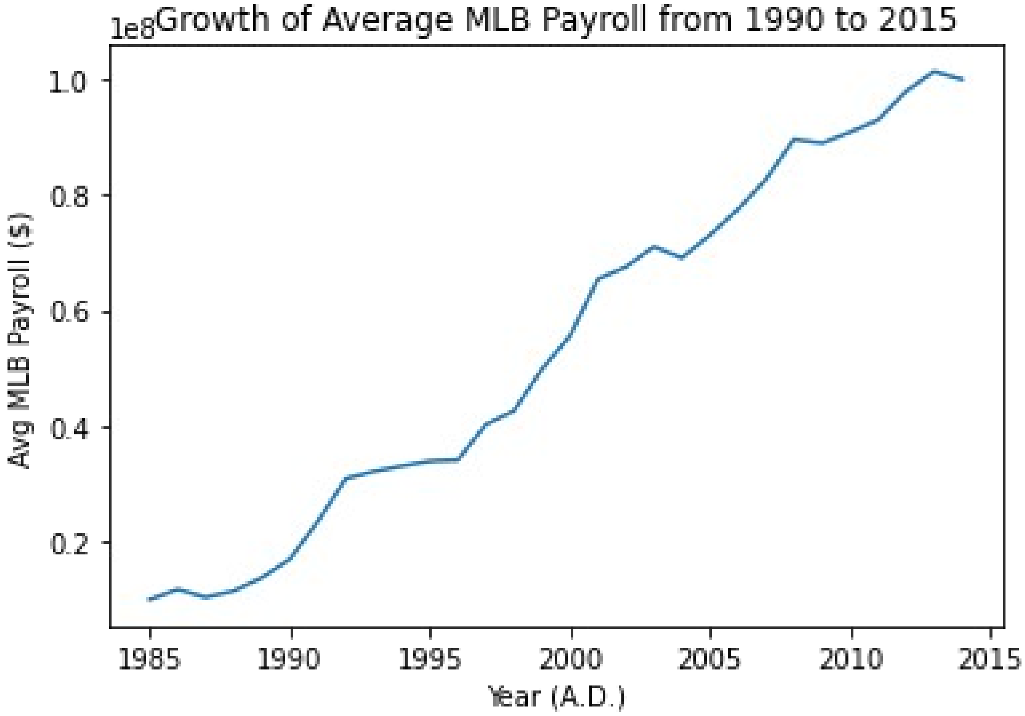

TEX
1e8
Total Payroll ($)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1990
1995
2000
2005
2010
Year (A.D.)





```
# Question 1
# The most common trend we see amonst payrolls as time goes on, is
that
# the payrolls increase from 1990-2014, regardless of the team that is
being
```

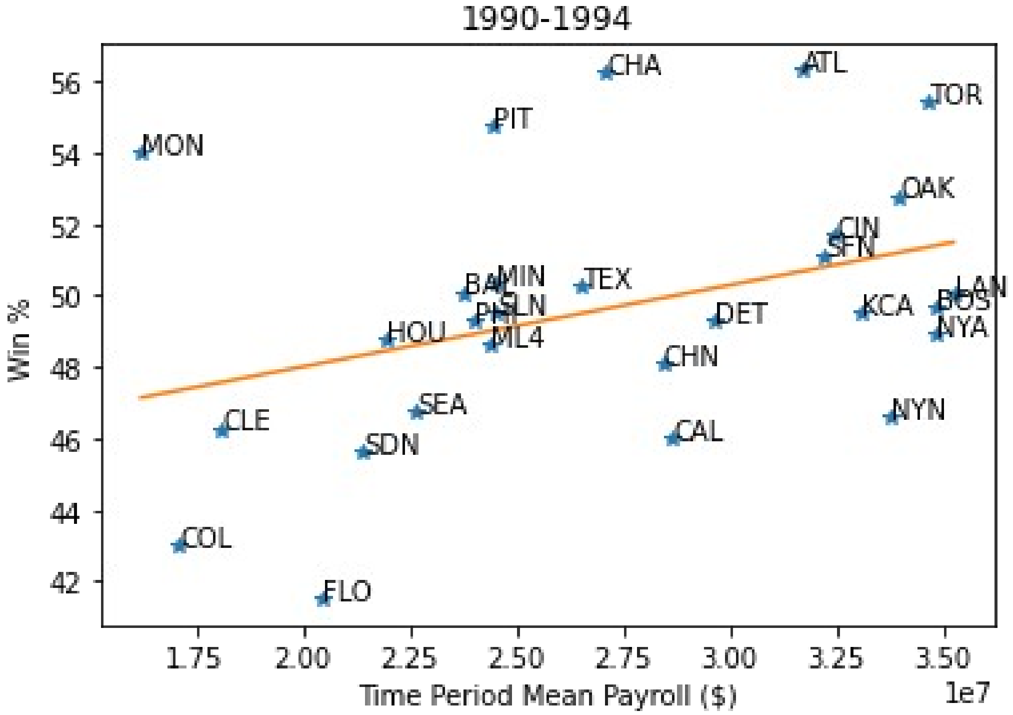
```
# observed. Therefore, the central tendnecy of mean payroll should
increase
# over time as well.

# Problem 3
ct_meanPayroll = custom_query[['yearID',
'total_payroll']].groupby('yearID').mean()
mplpp.plot(ct_meanPayroll.index,ct_meanPayroll['total_payroll'])
mplpp.xlabel("Year (A.D.)")
mplpp.ylabel("Avg MLB Payroll ($)")
mplpp.title("Growth of Average MLB Payroll from 1990 to 2015")
mplpp.show()
```



```
# Problem 4
df = pd.DataFrame({'yearID':range(1990, 2015)})
vars = custom_query[['yearID', 'teamID', 'total_payroll', 'wins',
'total_games']]
table = df.merge(vars, how="left", left_on=['yearID'],
right_on=['yearID'])
cutoffYears = [1990, 1995, 2000, 2005, 2010, 2015]
timePeriods = ['1990-1994', '1995-1999', '2000-2004', '2005-2009',
'2010-2014']
table['time'] = pd.cut(table['yearID'], cutoffYears, right=False,
labels=timePeriods)

#teamWAR = pd.DataFrame(columns = ["teamID", "WAR"])
for tp in timePeriods:
    winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
    winPrcts['tp_win%'] =
((winPrcts['wins'])/(winPrcts['total_games']))*100
    winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
    avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
    avgPayrolls.columns = ['teamID', 'tp_mean_payroll']
    scatPlotTeamPayrolls = avgPayrolls.merge(winPrcts)
    xVals = scatPlotTeamPayrolls['tp_mean_payroll'].values
    yVals = scatPlotTeamPayrolls['tp_win%'].values
    ds = np.polyfit(x = xVals, y = yVals, deg = 1)
    graph = np.poly1d(ds)
    xRange = np.linspace(xVals.min(), xVals.max(), 100)
    yRange = graph(xRange)
    mplpp.plot(xVals, yVals, '*', xRange, yRange)
    for i,data in enumerate(scatPlotTeamPayrolls['teamID']):
        mplpp.annotate(data, (xVals[i], yVals[i]), size=10)
    mplpp.xlabel("Time Period Mean Payroll ($)")
    mplpp.ylabel("Win %")
    mplpp.title(tp)
    mplpp.show()
#Vals
```

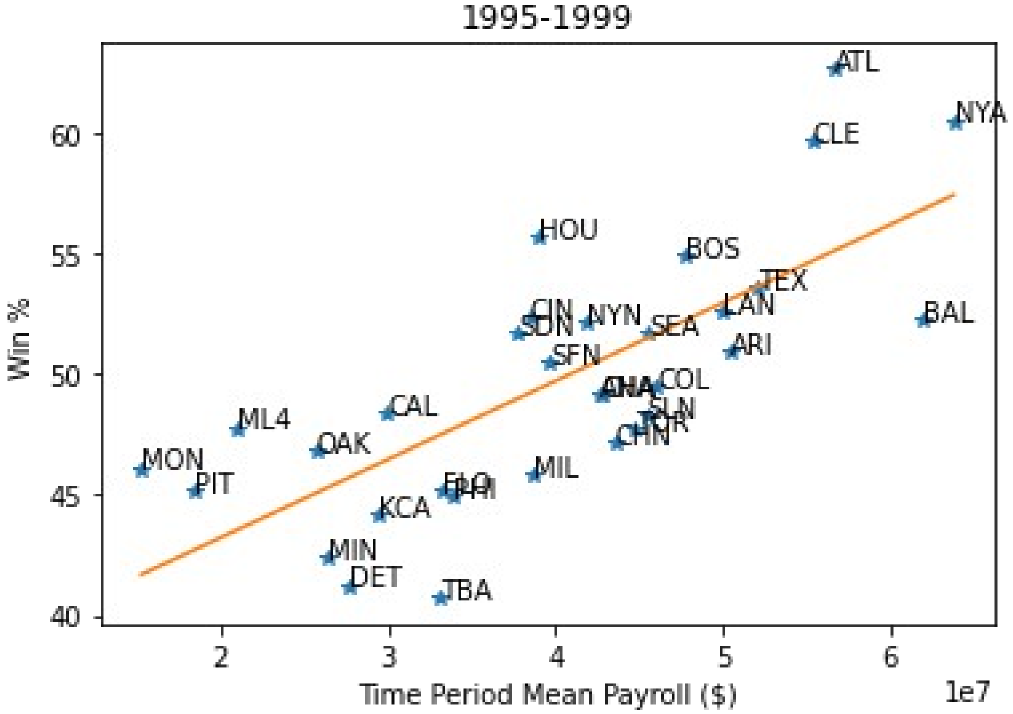
```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/2417793144.py:3: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:3: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
```

```
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
```



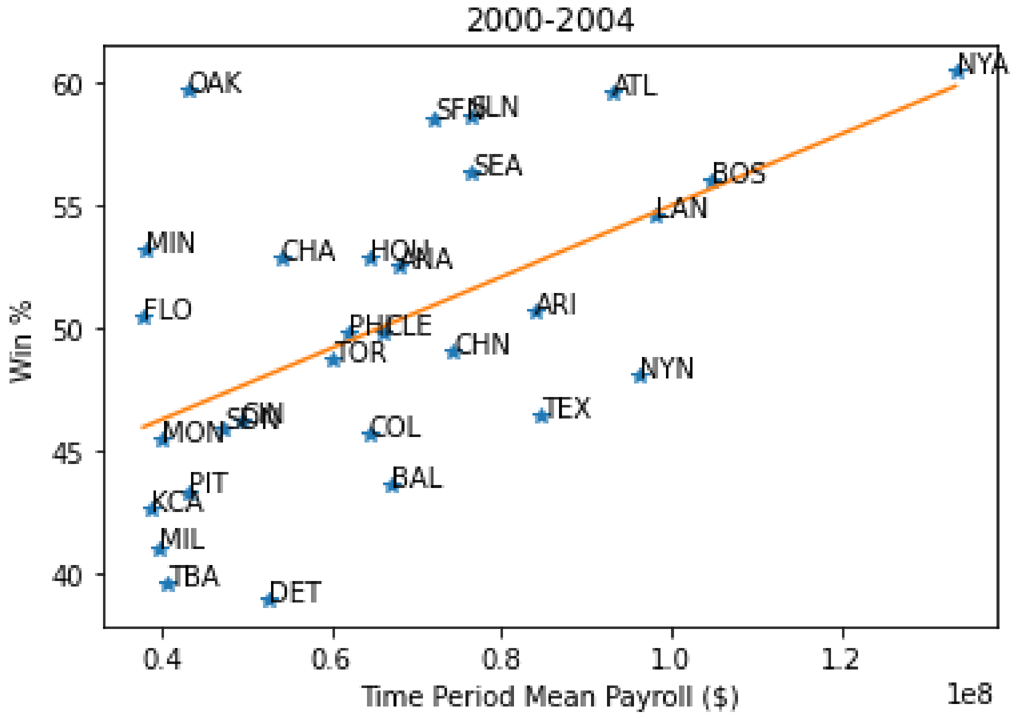
```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/2417793144.py:3: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
```

```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:3: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
```



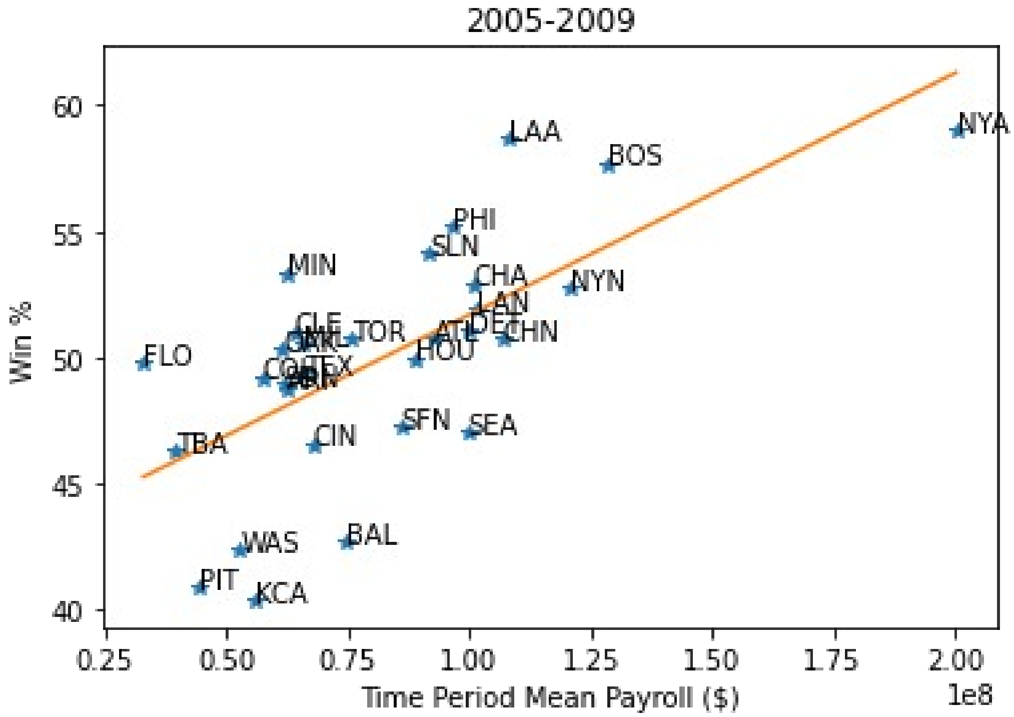
```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/2417793144.py:3: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:3: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
```

```
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
```



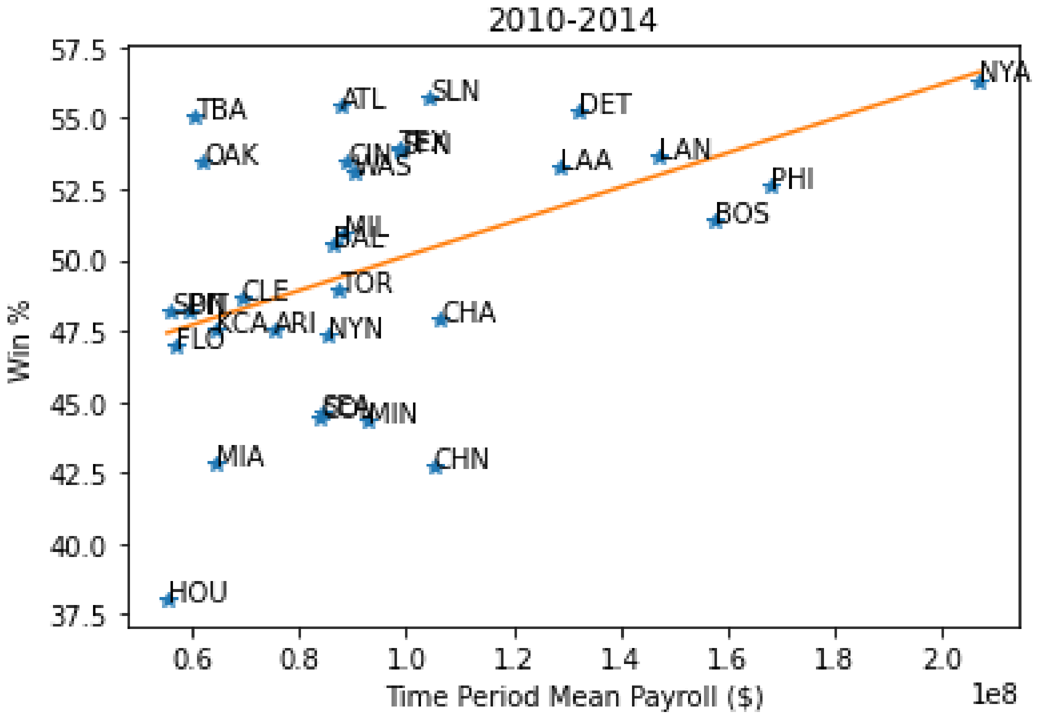
```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/2417793144.py:3: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:3: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
```

```
only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
```



```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/2417793144.py:3: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:3: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPrcts = winPrcts.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
```

```
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/24177
93144.py:6: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  avgPayrolls =
table.groupby(['time']).get_group(tp).drop('wins',1).drop('yearID',1).
drop('total_games',1).groupby(['teamID']).mean().reset_index()
```



```
# Question 2
# To look at the teams who paid for wins most effectively, we would
want to
# look for teams who have a lower payroll (more left) while having a
higher
# win percentage (more up). Alternatively, to look for teams that
least
# effectively paid for wins, we would want to look for teams with
higher
# payrolls and lower win percentages. Thankfully, this can be simply
```

```
done by
# looking at the teams that are above and below the regression line
most often.
# The team that stood out the most among the most efficient at
spending for wins
# was the Atlanta Braves (ATL), who were always above the expected
wins by a
# considerable marigin. The team that stood out the most among the
least
# efficient at spending for wins was the Chicago White Sox (CHN), who
surprisingly
# were the only team I observed that never had a win percentage above
the expected
# amount for any time period, which was surprising for a sport like
baseball
# with fluctuations of win percentage happening constantly regardless
of payroll.
# They also had a mid-tier payroll compared to the rest of the league,
which made
# it especially surprising.

# We can see, however, a general trend from the line of regression
that teams
# who have higher payrolls more typically often have higher win
percentages.
# We can also see that although the mean payroll increased over these
time
# period continuously increased, the trends did not change, and teams'
efficiency
# of buying wins would fluctuate from time period to time period.

# Looking at the Oakland A's, in the 90s, they were a team that
typically
# hovered slightly above the expected wins for the payroll they had.
Therefore,
# for the time before Moneyball, they were a team that was good at
paying
# for wins compared to the whole league. However, we could see a pay
decrease
# from the early 90s to the late 90s. When that payroll continued to
decrease
# in the early 2000s, the Oakland A's win percentages did not go down
with it.
# It in fact rose to among the highest win percentages over that time.
This outlier
# state would fade away in the late 2000s, only to return in the early
2010s,
# although not as extreme as the Moneyball era. Therefore, while
Oakland has
# always been a good team at buying wins, in the Moneyball era they
```

```
were especially
# great at this, despite their payroll decreasing at the time.
# In both the early 2000s and 2010s the Oakland A's had one of the
highest win
# percentages despite having one of the lowest payrolls in the league.
That's Moneyball.

#----------------------PART 3: DATA
TRANSFORMATIONS-------------------------------

# Problem 5
avgPayroll = (custom_query[['yearID',
'total_payroll']].groupby('yearID')).mean()
sdPayroll = (custom_query[['yearID',
'total_payroll']].groupby('yearID')).std()
avgPayroll.columns = ['average_payroll']
sdPayroll.columns = ['standardized_payroll']
p5table = table.copy()
p5table = p5table.drop('wins', 1)
p5table.columns = ['yearID', 'teamID', 'total_payroll',
'standardized_payroll', 'time']
for index,row in p5table.iterrows():
    iatVal = (row["total_payroll"] - avgPayroll["average_payroll"]
[row["yearID"]]) / (sdPayroll["standardized_payroll"][row["yearID"]])
    p5table.iat[index, 3] = iatVal
# merge final table
p5table = df.merge(p5table, how="left", left_on=["yearID"],
right_on=["yearID"])
p5table
```

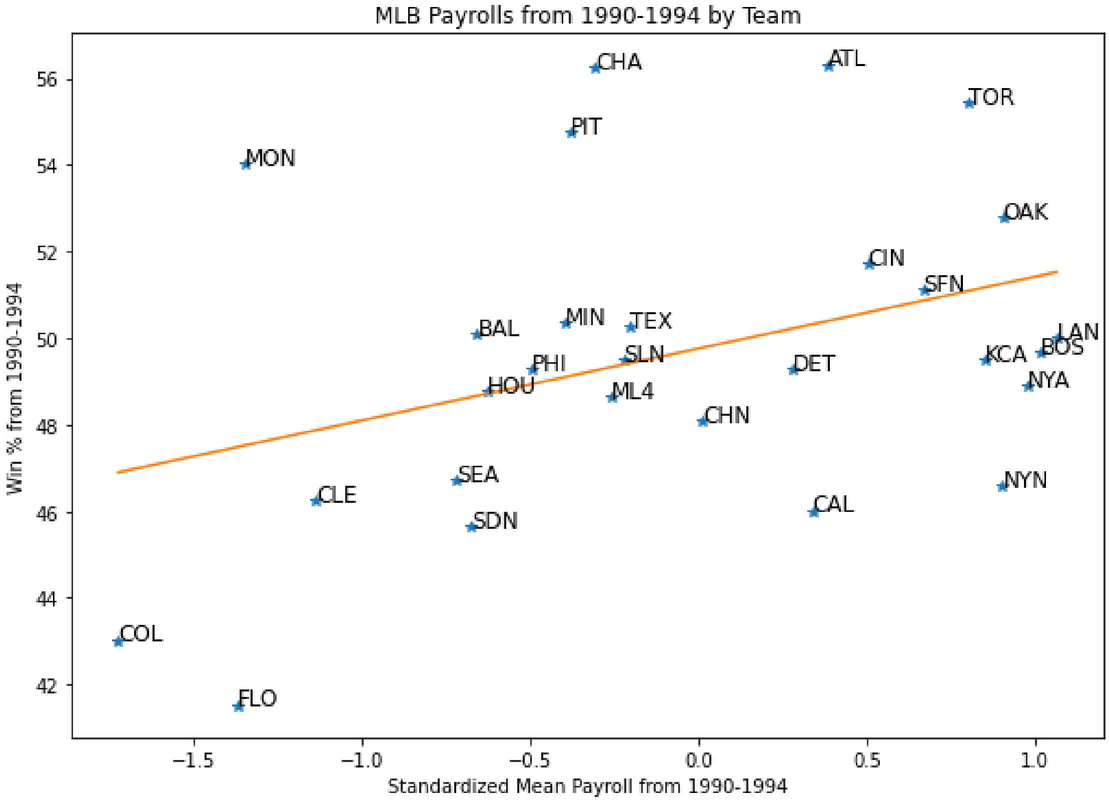
```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/1043833952.py:9: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  p5table = p5table.drop('wins', 1)

     yearID teamID  total_payroll  standardized_payroll       time
0      1990    ATL     14555501.0             -0.667275  1990-1994
1      1990    BAL      9680084.0             -1.959861  1990-1994
2      1990    BOS     20558333.0              0.924213  1990-1994
3      1990    CAL     21720000.0              1.232198  1990-1994
4      1990    CHA      9491500.0             -2.009859  1990-1994
..      ...    ...            ...                   ...        ...
723    2014    SLN    120693000.0              0.457126  2010-2014
724    2014    TBA     72689100.0             -0.593171  2010-2014
725    2014    TEX    112255059.0              0.272509  2010-2014
726    2014    TOR    109920100.0              0.221422  2010-2014
727    2014    WAS    131983680.0              0.704160  2010-2014

[728 rows x 5 columns]
```

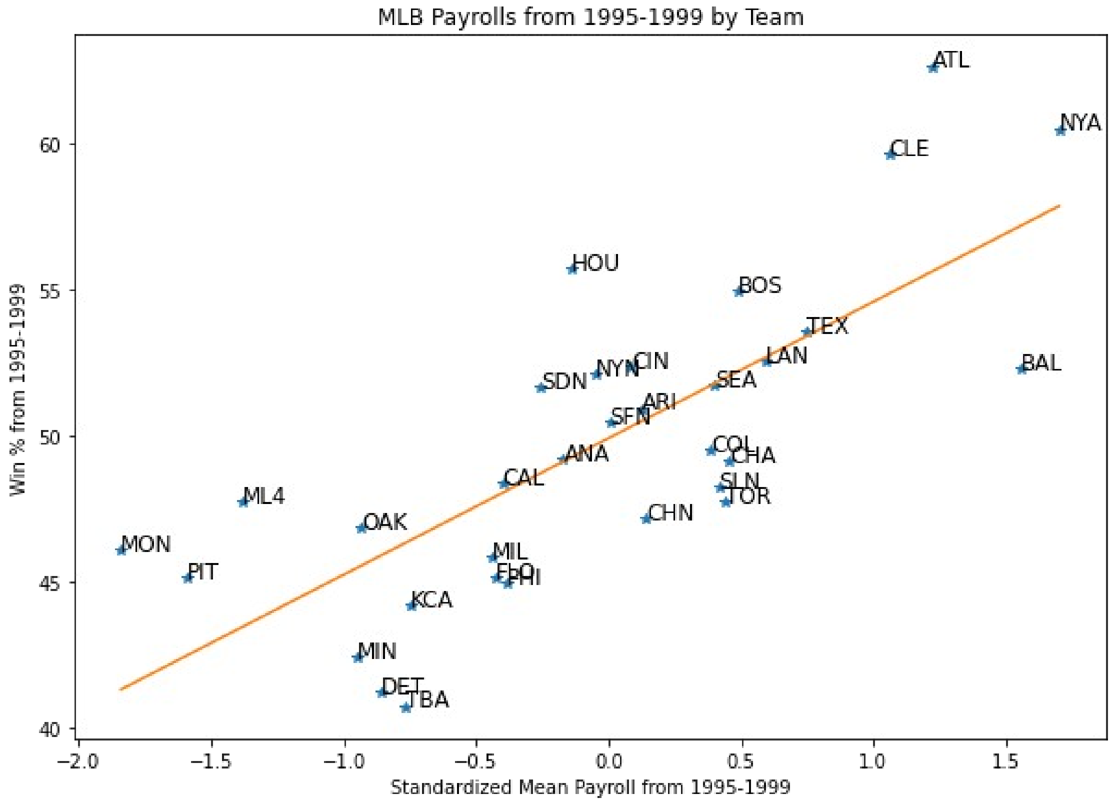
```
# Problem 6
import matplotlib.pyplot as mplpp
import matplotlib_inline as mplil
for tp in timePeriods:
    sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
    sdPayrollTable.columns = ['teamID','tp_standardized_payroll']
    winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
    winPctTable['tp_win%'] = 100 * winPctTable['wins'] /
winPctTable['total_games']
    winPctable = winPctTable.drop('wins',1).drop('total_games',1)
    p6table = winPctTable.merge(sdPayrollTable)
    xVals = p6table['tp_standardized_payroll'].values
    yVals = p6table['tp_win%'].values
    ds = np.polyfit(x=xVals,y=yVals,deg=1)
    graph = np.poly1d(ds)
    x = np.linspace(xVals.min(), xVals.max(), 100)
    y = graph(x)
    mplpp.figure(figsize=(10,7))
    mplpp.plot(xVals, yVals, '*', x, y)
    for index, row in enumerate(p6table['teamID']):
        mplpp.annotate(row, (xVals[index] ,yVals[index]), size=12)
    mplpp.xlabel("Standardized Mean Payroll from "+tp)
    mplpp.ylabel("Win % from "+tp)
    mplpp.title("MLB Payrolls from "+tp+" by Team")
    mplpp
```

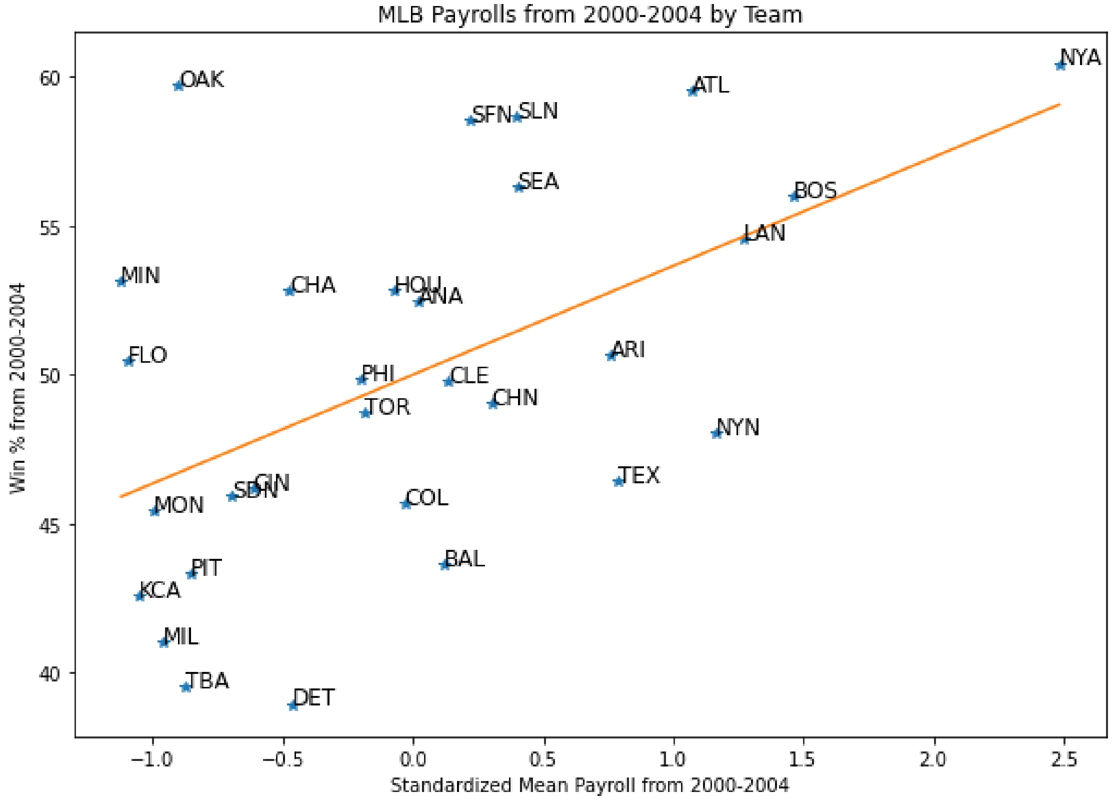
```
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/
ipykernel_13315/4079096892.py:5: FutureWarning: In a future version of
pandas all arguments of DataFrame.drop except for the argument
'labels' will be keyword-only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
```

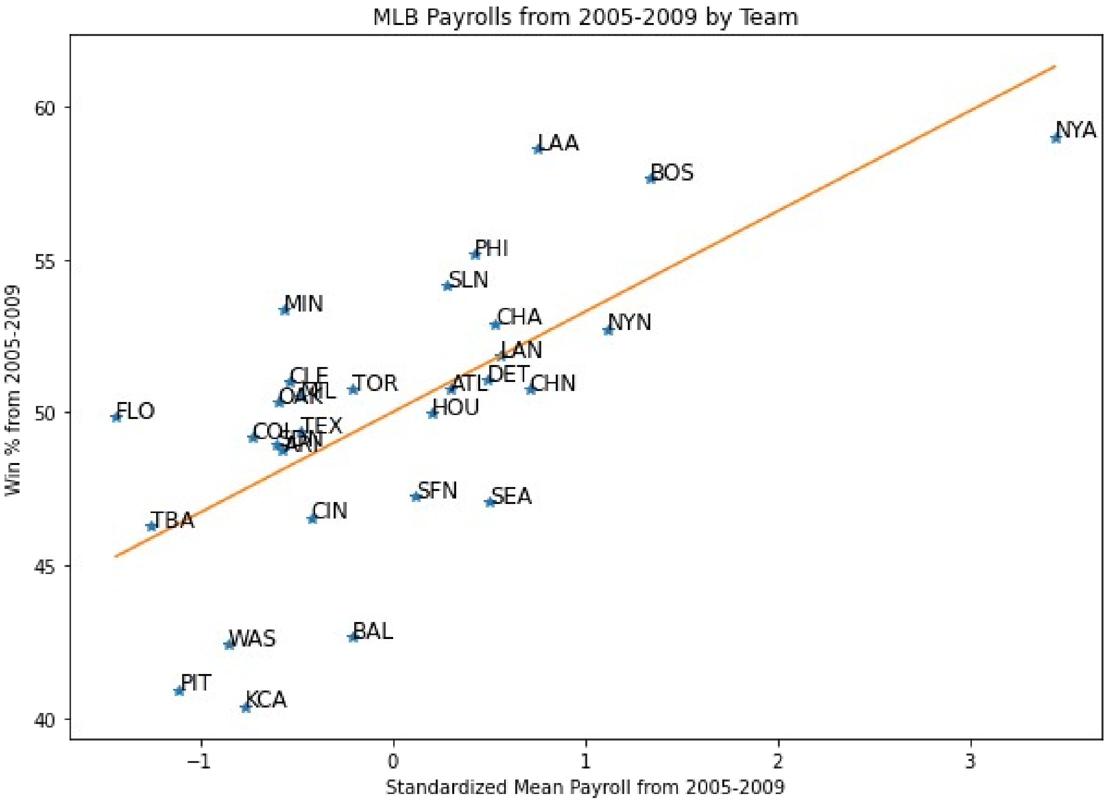
```
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:5: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  sdPayrollTable =
p5table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('
yearID',1).groupby(['teamID']).mean().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
```

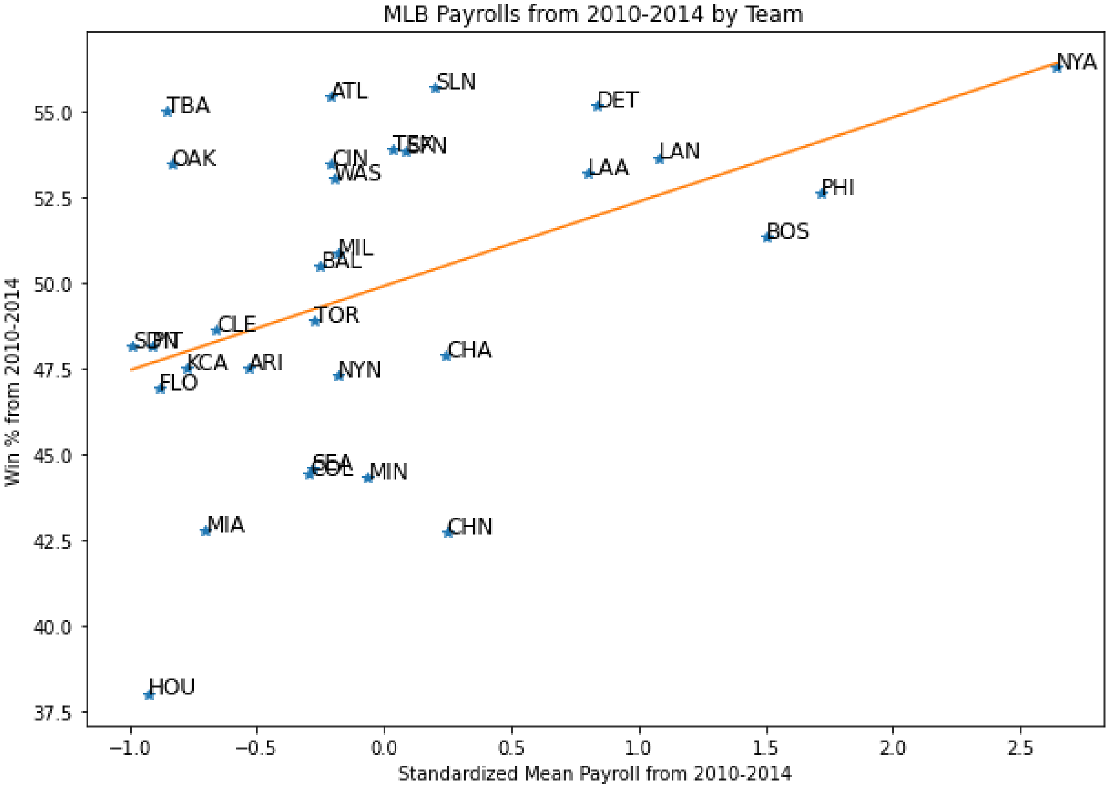
```
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:7: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctTable =
table.groupby(['time']).get_group(tp).drop('total_payroll',1).drop('ye
arID',1).groupby(['teamID']).sum().reset_index()
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
/var/folders/yl/0_18jst15nb9gbl2n_j_z5tw0000gn/T/ipykernel_13315/40790
96892.py:9: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-
only.
  winPctable = winPctTable.drop('wins',1).drop('total_games',1)
```



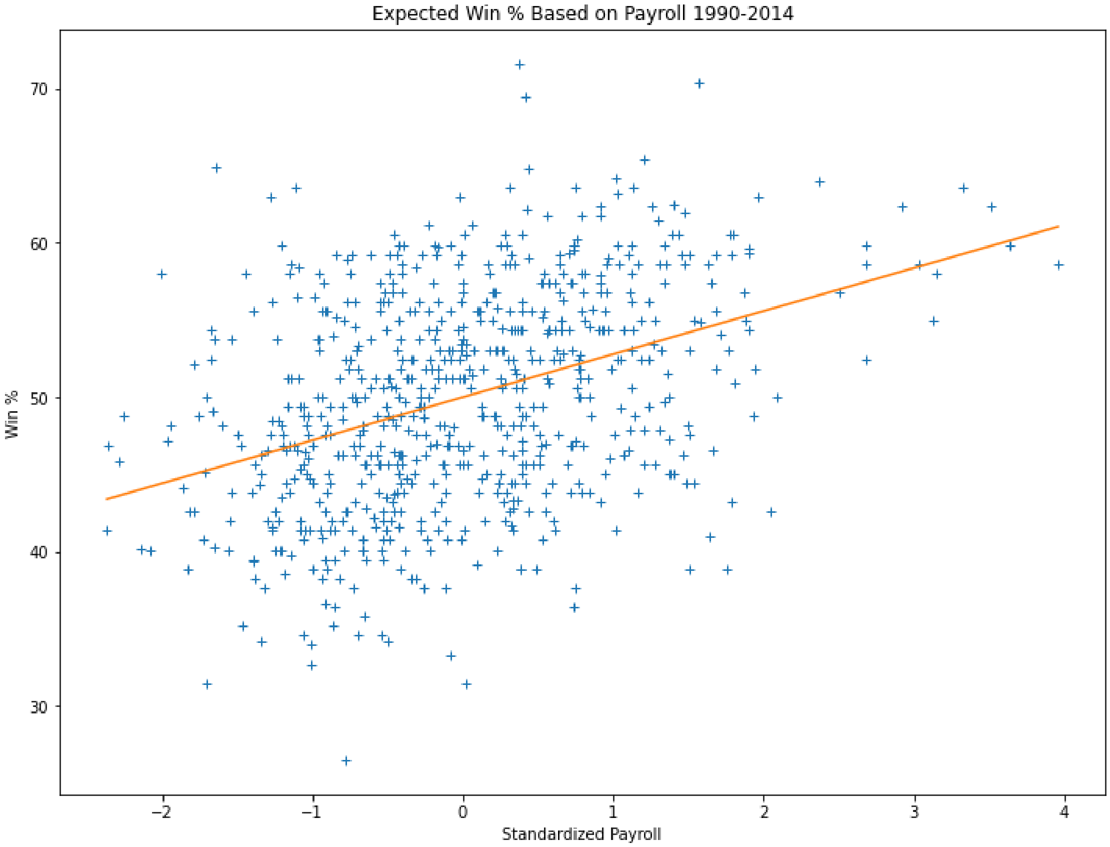

MLB Payrolls from 1995-1999 by Team
Win % from 1995-1999
Standardized Mean Payroll from 1995-1999
40
45
50
55
60
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
ATL
NYA
CLE
HOU
BOS
TEX
NYN
CIN
LAN
BAL
SDN
SEA
ARI
SFN
COL
ANA
CHA
CAL
SLN
ML4
TOR
CHN
OAK
MON
MIL
PIT
FLO
PHI
KCA
MIN
DET
TBA


MLB Payrolls from 2000-2004 by Team
Win % from 2000-2004
Standardized Mean Payroll from 2000-2004
40
45
50
55
60
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
NYA
OAK
ATL
SFN
SLN
SEA
BOS
LAN
MIN
CHA
HOU
ANA
FLO
ARI
PHI
CLE
CHN
TOR
NYN
TEX
CIN
SDN
COL
MON
BAL
PIT
KCA
MIL
TBA
DET


MLB Payrolls from 2005-2009 by Team
Win % from 2005-2009
Standardized Mean Payroll from 2005-2009
40
45
50
55
60
−1
0
1
2
3
NYA
LAA
BOS
PHI
SLN
MIN
CHA
NYN
LAN
CLE
MIL
OAK
TOR
DET
ATL
CHN
FLO
HOU
COL
TEX
SDN
ARI
SFN
SEA
CIN
TBA
WAS
BAL
PIT
KCA

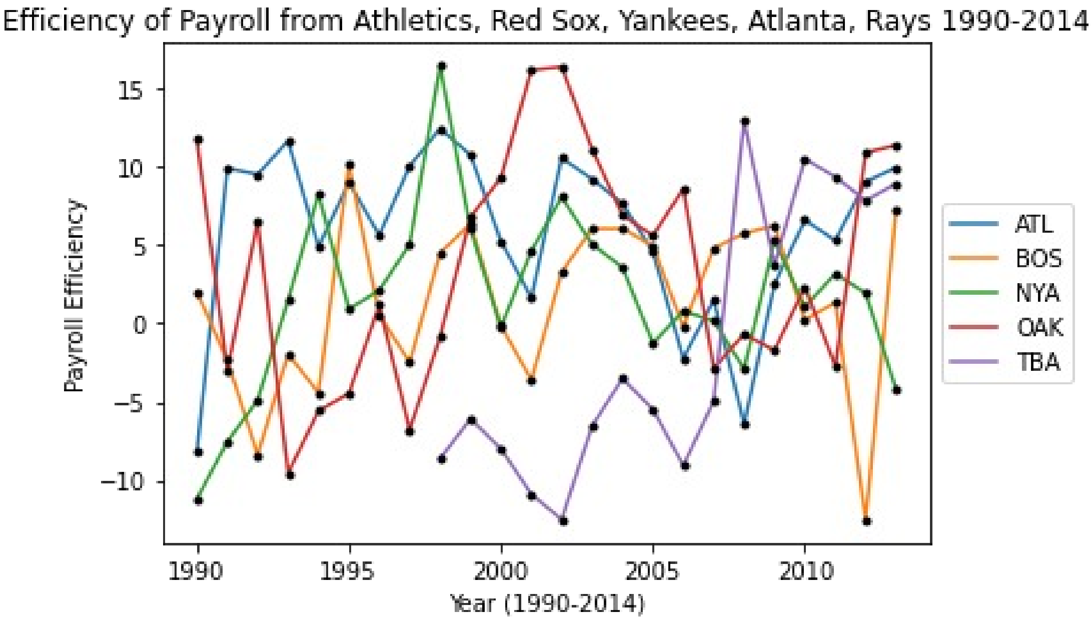

MLB Payrolls from 2010-2014 by Team
Win % from 2010-2014
Standardized Mean Payroll from 2010-2014
37.5
40.0
42.5
45.0
47.5
50.0
52.5
55.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
NYA
SLN
ATL
DET
TBA
TEX
SFN
LAN
OAK
CIN
WAS
LAA
PHI
BOS
MIL
BAL
TOR
SDN
PIT
CLE
CHA
KCA
ARI
NYN
FLO
SEA
COL
MIN
MIA
CHN
HOU

```
# Question 3

# The plots above display the mean payrolls for the MLB teams in 5
year spans from 1990 to 2014. The graphs
# in both problem 6 and 4 look nearly identical. Therefore, it was not
so much the points that were different,
# but the values of the x axis. Instead of it being the dollar value
of the mean payroll, it was the
# standardized value of the payroll, basically switching the payroll
value from dollars to the standard
# deviation.

p7data = custom_query[['yearID','teamID','total_payroll','win_rate%']]
p7table = payThem_df.merge(p7data, how = "left", left_on = ['yearID'],
right_on = ['yearID'])
projWinPct = p7table[['teamID','yearID','win_rate%']].copy()
projWinPct['sd_payroll'] = 0
for index,row in p7table.iterrows():
    iatVal = (row["total_payroll"] - avgPayroll["average_payroll"]
[row["yearID"]]) / (sdPayroll["standardized_payroll"][row["yearID"]])
    projWinPct.iat[index, 3] = iatVal
xVals = projWinPct['sd_payroll'].values
yVals = projWinPct['win_rate%'].values
ds = np.polyfit(x = xVals, y = yVals, deg=1)
graph = np.poly1d(ds)
x = np.linspace(xVals.min(), xVals.max(), 100)
y = graph(x)
mplpp.figure(figsize=(12, 9))
mplpp.plot(xVals, yVals, '+', x, y)
mplpp.xlabel("Standardized Payroll")
mplpp.ylabel("Win %")
mplpp.title("Expected Win % Based on Payroll 1990-2014")
mplpp.show()
projWinPct['expected_win%'] = projWinPct['sd_payroll']*2.5+50
#projWinPct
```



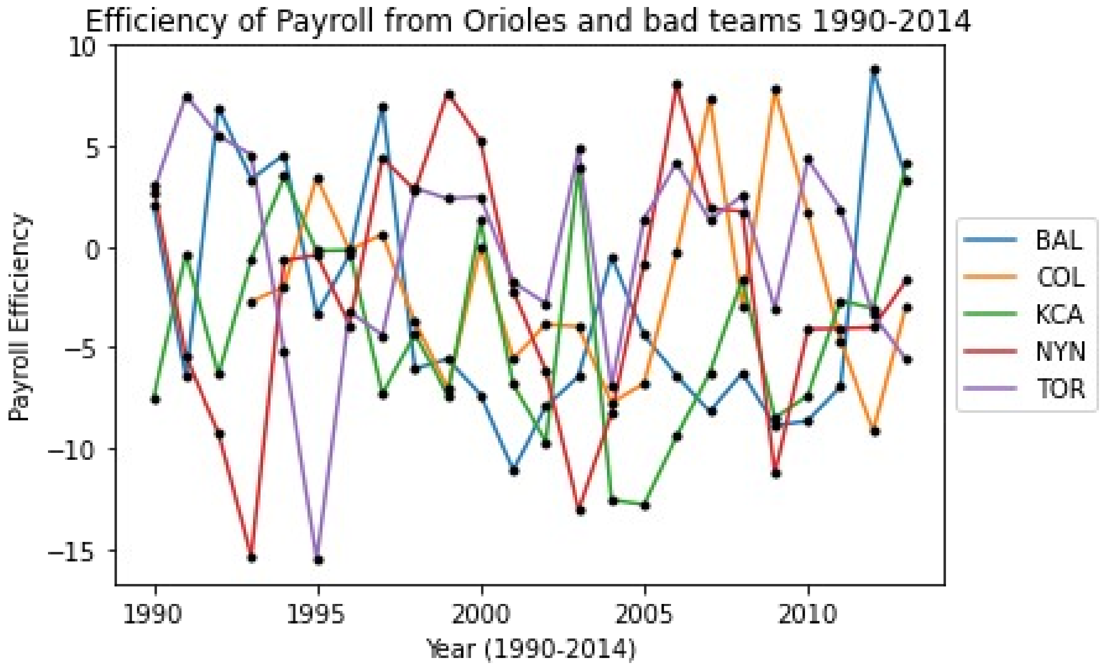
```
# Problem 8

p8table = p7table[['yearID', 'teamID', 'win_rate%']].copy()
p8table['expected_win%'] = projWinPct['expected_win%']
p8table['efficiency'] = p8table['win_rate%'] - p8table['expected_win
%']
spending_efficiency = p8table.loc[p8table['teamID'].isin(['OAK',
'BOS', 'NYA', 'ATL', 'TBA'])].sort_values(['teamID', 'yearID'],
ascending = [True, True])
graph,ds = mplpp.subplots()
teams = []
for key,eff in spending_efficiency.groupby(['teamID']):
    ds = eff.plot(ax=ds, kind='line', x='yearID', y='efficiency')
    teams.append(key)
lines,_ = ds.get_legend_handles_labels()
ds.legend(lines, teams, loc='center left', bbox_to_anchor=(1.0, 0.5))
mplpp.plot(spending_efficiency['yearID'].values,
spending_efficiency['efficiency'].values ,'.', c = 'black')
mplpp.xlabel("Year (1990-2014)")
mplpp.ylabel("Payroll Efficiency")
mplpp.title("Efficiency of Payroll from Athletics, Red Sox, Yankees,
Atlanta, Rays 1990-2014")
mplpp
```

```
<module 'matplotlib.pyplot' from
'/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-
packages/matplotlib/pyplot.py'>
```



```
# Question 4

# We see from this chart that on every team measured, that there were
large jumps up and down in terms
# of efficiency. Such is the case with sports. However, while the
graph in Question 2 showed the payroll's
# correlation with win percentages from 5 year periods, and the graph
in Question 3 did the same but with
# the standardized payroll, which was pretty much the same 2 graphs.
However, the graph above shows that
# correlation between payroll and win percentage as one value (payroll
efficiency). We also see these
# values from every year, as opposed to the mean payroll over 5 years,
which provides more accurate values.
# We can see then, from the graph, that the Oakland Athletics from
2000-2004 were far more efficient than
# their peers. Therefore, the "Moneyball" A's were legit. This can be
seen from the charts in questions
# 2 and 3 as well. However, this is the best proof of their
legitimacy.

#for fun
fun_table = p7table[['yearID', 'teamID', 'win_rate%']].copy()
fun_table['expected_win%'] = projWinPct['expected_win%']
fun_table['efficiency'] = fun_table['win_rate%'] -
p8table['expected_win%']
fur_fun = fun_table.loc[fun_table['teamID'].isin(['NYN', 'BAL', 'KCA',
'TOR', 'COL'])].sort_values(['teamID', 'yearID'], ascending = [True,
True])
graph,ds = mplpp.subplots()
teams = []
for key,eff in fur_fun.groupby(['teamID']):
    ds = eff.plot(ax=ds, kind='line', x='yearID', y='efficiency')
    teams.append(key)
lines,_ = ds.get_legend_handles_labels()
ds.legend(lines, teams, loc='center left', bbox_to_anchor=(1.0, 0.5))
mplpp.plot(fur_fun['yearID'].values,
fur_fun['efficiency'].values ,'.', c = 'black')
mplpp.xlabel("Year (1990-2014)")
mplpp.ylabel("Payroll Efficiency")
mplpp.title("Efficiency of Payroll from Orioles and bad teams 1990-
2014")
mplpp
```

```
<module 'matplotlib.pyplot' from
'/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-
packages/matplotlib/pyplot.py'>
```


Efficiency of Payroll from Orioles and bad teams 1990-2014
Payroll Efficiency
Year (1990-2014)
BAL
COL
KCA
NYN
TOR