Data Collection

```
players = pd.DataFrame(columns = ['Year', 'Player', 'Team', 'G', 'College', 'BirthDate'])
holder = pd.DataFrame()

for year in range(1999, 2024):
    # 49ers
    s = 'https://www.pro-football-reference.com/teams/sfo/%s_roster.htm' % year
    url = requests.get(s)
    parse = BeautifulSoup(url.text, "lxml").prettify()
    start = "<table"
    end = "</table"
    n, begin = 2, 0
    while begin >= 0 and n >= 1:
        begin = parse.find(start, begin+len(start), len(parse))
        index1 = begin if begin != -1 else index1
        n -= 1
    n, finnish = 2, 0
    while finnish >= 0 and n >= 1:
        finnish = parse.find(end, finnish+len(end))
        index2 = finnish if finnish != -1 else index2
        n -= 1

    holder = pd.read_html(parse[index1:index2+8])[0]
    holder['Year'] = year
    holder['Team'] = '49ers'
    holder = holder[['Year', 'Player', 'Team', 'G', 'BirthDate', 'College/Univ']]
    holder = holder.rename(columns = {'College/Univ' : 'College'})
    players = pd.concat([players, holder])
```

Data can be collected via PFR, as they run the immaculate gride website. Due to scraping prevention techniques, one-minute pauses are needed every 16 teams. The process above is repeated for every team. Teams created after 2000 need a special check. Data is then saved into a file to avoid having to repeat this long process often.

```
df = pd.read_csv('players.txt').iloc[: , 1:]
df['Player'] = df['Player'].apply(lambda x: x[:-5] if x[-5:] == ' (IR)' else x)
df['Player'] = df['Player'].apply(lambda x: x[:-6] if x[-6:] == ' (IRD)' else x)
df['Player'] = df['Player'].apply(lambda x: x[:-6] if x[-6:] == ' (SUS)' else x)
df['Player'] = df['Player'].apply(lambda x: x[:-6] if x[-6:] == ' (NON)' else x)
df = df[df['Player'] != 'Team Total']

players = df.groupby(['Player', 'BirthDate'])
players.get_group((list(players.groups)[0]))

transfers = pd.DataFrame(columns = ['Year', 'G', 'Player', 'BirthDate', 'College', 'From', 'To', 'Active'])
transfers['Active'] = transfers['Active'].astype(bool)

for k, v in players:
    v['Value'] = (v['Team'] != v['Team'].shift()).cumsum()
    v.drop_duplicates(subset = ['Value'], inplace = True)
    v.reset_index(inplace = True)
    v.drop(columns = ['index'], inplace = True)

    if v['Year'].iloc[-1] == CURRENT_YEAR:
        active = True
    else:
        active = False
    for i in range(0, len(v.index)):
        if i == 0:
            transfers = pd.concat([transfers, pd.DataFrame(data = {'Year' : [v.iloc[i]['Year']], 'Player' :[
        else:
            transfers = pd.concat([transfers, pd.DataFrame(data = {'Year' : [v.iloc[i]['Year']], 'Player' :[

transfers = transfers.sort_values(['Player', 'Year', 'G'], ascending = True)
```

Unnecessary suffixes are removed from player names. Players are then grouped by their keys, name and birthdate. Columns are reduced by necessity and an Active column is added for active players. Each player now has all their transactions listed. Duplicates are dropped, the active column is updated, and each official transaction is added to the transfers data frame. If this is the player's first transaction, 'From' is set to None. The data frame is then sorted.

```
# #--- Player Corrections ---
transfers.reset_index(inplace = True)
#James Harrison
indexDrop = transfers[(transfers['Player'] == 'James Harrison') & (transfers['From'] == 'Patriots')].index
transfers.drop(indexDrop, inplace = True)
#Kahzin Daniels
transfers.loc[transfers.Player == 'Kahzin Daniels', 'College'] = 'Charleston'
#Miles Brown
transfers.loc[transfers.Player == 'Miles Brown', 'College'] = 'Wofford'

transfers.reset_index(inplace = True)
transfers.drop(columns = ['index', 'level_0'], inplace = True)
transfers['Transactions'] = 1
transfers.to_csv('transfers.txt')
```

Individual corrections are made where needed. A static column of 1 is added for Plotly.

Application

Since Github can only render static HTML's, I cannot upload the app here. Click the project link to download the project, the data, and run it on your own.