Created by Axel Garcia

In [2]:
```
import pandas as pd
import numpy as np
import glob
pd.set_option('display.max_colwidth', -1)
```

Merge all .txt files in directory as data was kept seperate by year.

In [ ]:
```
path = r'/Users/axelgarcia/Documents/CSE 184/usafec.nosync/Data/' # use your p
ath
all_files = glob.glob(path + "/*.txt")

li = []

for filename in all_files:
    print(filename)
    df = pd.read_csv(filename, index_col=None, header=None, sep = "|",error_ba
d_lines=False)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)
```

Quick peek at data

In [ ]:
```
frame.head()
```

Drop unnecessary columns such as name, city, .etc

In [ ]:
```
cols = [1,2,3,4,5,6,7,8,10,12,15,16,17,18,19,20]
frame.drop(frame.columns[cols],axis=1,inplace=True)
```

In [ ]:
```
frame.head()
```

In [ ]:
```
df = frame.copy()
df.dropna(inplace=True)
df
```

Attempt to sort by date, this failed as 12212018.0 is bigger than 1212020.0

In [ ]:
```
df.sort_values(by=[13], inplace = True)
```

If date isn't of valid length, drop it.

```
In [ ]: df = df[df[13].astype(str).str.len() > 7]
        df
```

We only want year value, so keep 6th char to third to last char.

```
In [ ]: df[13] = df[13].astype(str).str[-6:-2]
```

Successfully sort by year.

```
In [ ]: df = df.sort_values(by=[13])
```

```
In [ ]: df[13] = df[13].astype(int)
```

Add names to columns.

```
In [ ]: df.columns = ['cmte_id', 'state', 'employer', 'year', 'amount']
```

Drop row if year is not within expected range.

```
In [ ]: df = df.drop(df[(df['year'] < 1979) | (df['year'] > 2020)].index)
```

Peek at data.

```
In [ ]: df
```

Read csv containing committees and their respective party affiliation.

```
In [3]: df1 = pd.read_csv('/Users/axelgarcia/Documents/CSE 184/usafec.nosync/Data/clea
        nCommittees.csv', index_col=None, sep = ",",error_bad_lines=False)
```

We only have use for data that has a possible party mapping, so we check if there is a corresponding mapping in committee data, if not we drop that data.

```
In [5]: df = df[df['cmte_id'].isin(df1['cmte_id'])]
```

```
In [ ]: Peek at data.
```

In [6]:
```
df
```

Out[6]:
```
                cmte_id  state                   employer  year  amount
       0      C00096941     KY  INSURANCE SALES AND ADM    1979    1000
       1      C00108407     MD                PHYSICIAN    1979    1000
       2      C00020040     CA                  RETIRED    1979     500
       3      C00107318     AZ                REQUESTED    1979    1000
       4      C00107318     AZ                REQUESTED    1979    1000
     ...            ...    ...                      ...   ...     ...
30489455      C00306704     AL                 ATTORNEY    2020     200
30489456      C00369140     OH                  RETIRED    2020     200
30489457      C00266932     GA   SHEPHERD CONSTRUCTION    2020     500
30489458      C00346544     VA                  RETIRED    2020     400
30489459      C00003418     CO            SELF-EMPLOYED    2020     250

30489460 rows × 5 columns
```

Now that we have only the data with possible mappings, merged both dataframes in order to get the party affiliation of the committee for the donation.

In [7]:
```
df = df.merge(df1,on='cmte_id',how='inner')
```

Sort by year again.

In [11]:
```
df = df.sort_values('year')
```

Store this cleaned data in csv.

In [12]:
```
df.to_csv('/Users/axelgarcia/Documents/CSE 184/usafec.nosync/Data/cleanDataWit
hParties.csv', index=False)
```

Read in inflation data.

In [6]:
```
dfInflation = pd.read_csv('/Users/axelgarcia/Documents/CSE 184/usafec.nosync/D
ata/CPIAUCNS-1.csv')
```

Merge dataframes to get respective inflation rate in each row.

In [8]:
```
df = df.merge(dfInflation,on='year',how='inner')
```

Peek data

```
In [13]: df
```

Out[13]:

	cmte_id	state	employer	year	amount	party	CPIAUCNS
0	C00096941	KY	INSURANCE SALES AND ADM	1979	1000	DEM	72.575000
1	C00078295	NY	WILKIE, FARR & GALLAGHER	1979	500	DEM	72.575000
2	C00078295	NY	WENDER, MURASE & WHITE	1979	1000	DEM	72.575000
3	C00078295	NY	SMILIN & SAFIER, INC	1979	300	DEM	72.575000
4	C00078295	NY	MUDGE ROSE ET AL	1979	500	DEM	72.575000
...	...	...	...	...	...	...	...
30489448	C00694455	DC	THE CALPRO GROUP	2019	200	DEM	251.106833
30489449	C00694455	WA	SELF-EMPLOYED	2019	100	DEM	251.106833
30489450	C00694455	CA	SELF-EMPLOYED	2019	2800	DEM	251.106833
30489451	C00694455	CA	IT'S A WRAPPER! FILMS	2019	2800	DEM	251.106833
30489452	C00702340	TX	SELF	2019	25	IND	251.106833

30489453 rows × 7 columns

Apply function in order to adjust donation amounts to 2019 dollars

```
In [16]: def func(a,b):
             a = a * (251.106833/b)
             return a
         df['adjusted'] = df.apply(lambda x: func(x['amount'],x['CPIAUCNS']), axis=1)
```

Drop unnecessary inflation rate

```
In [24]: df.drop(df2.columns[6],axis=1,inplace=True)
```

Save to csv

```
In [26]: df.to_csv('/Users/axelgarcia/Documents/CSE 184/usafec.nosync/Data/cleanDataWit
         hPartiesInflation.csv', index=False)
```