In [7]:

```
import numpy as np
import pandas as pd
```

Read all the *detail.csv.*
Renamed "2015Q2-house-disburse-detail.csv" to "2015Q2-house-disburse-detail-old.csv" Then renamed "2015Q2-house-disburse-detail-updated.csv" to "2015Q2-house-disburse-detail.csv". Then redirected all the filenames to "filename.txt" using the command: ls detail.csv > filename.txt

In [8]:

```
# Create a list of filename called file_list
# Strip '\n' at the end of the filename
#Ref: https://stackoverflow.com/questions/42488579/
#remove-n-from-each-string-stored-in-a-python-list

file_list = []
with open('filename.txt', 'r', encoding='utf-8') as myfile:
    for line in myfile:
        st_line = line.rstrip()
        file_list.append(st_line)
file_list=file_list[26:30]  #Slicing 2016 files
print(file_list)
```

```
['2016Q1-house-disburse-detail.csv', '2016Q2-house-disburse-detail.c
sv', '2016Q3-house-disburse-detail.csv', '2016Q4-house-disburse-deta
il.csv']
```

In [9]:

```
#Create a dataframe for each of 2016 quarter files and concatenate the
4 dataframes
df1 = pd.read_csv('2016Q1-house-disburse-detail.csv', low_memory = Fal
se)
df2 = pd.read_csv('2016Q2-house-disburse-detail.csv', low_memory = Fal
se)
df3 = pd.read_csv('2016Q3-house-disburse-detail.csv', low_memory = Fal
se)
df4 = pd.read_csv('2016Q4-house-disburse-detail.csv', low_memory = Fal
se)
```

In [10]:

```
df = pd.concat([df1, df2, df3, df4])
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 385613 entries, 0 to 90674
Data columns (total 16 columns):
AMOUNT           385613 non-null object
BIOGUIDE_ID      306557 non-null object
CATEGORY         385613 non-null object
DATE             328689 non-null object
END DATE         385612 non-null object
OFFICE           385613 non-null object
PAYEE            334724 non-null object
PROGRAM          90675 non-null object
PURPOSE          385611 non-null object
QUARTER          385613 non-null object
RECIP (orig.)    334724 non-null object
RECORDID         328690 non-null object
START DATE       385612 non-null object
TRANSCODE        328692 non-null object
TRANSCODELONG    250648 non-null object
YEAR             385613 non-null object
dtypes: object(16)
memory usage: 50.0+ MB
```

In [12]:

```
df.head()
```

Out[12]:

	AMOUNT	BIOGUIDE_ID	CATEGORY	DATE	END DATE	OFFICE	I
0	380.00	NaN	SUPPLIES AND MATERIALS	03-18	02/28/16	OFFICE OF THE SPEAKER	CITI PCARD-GALLERIA FL(
1	6,666.67	NaN	PERSONNEL COMPENSATION	NaN	03/31/16	OFFICE OF THE SPEAKER	ALTHOUSE,JC S
2	25,666.67	NaN	PERSONNEL COMPENSATION	NaN	03/31/16	OFFICE OF THE SPEAKER	ANDRES,DOU R
3	18,333.33	NaN	PERSONNEL COMPENSATION	NaN	03/31/16	OFFICE OF THE SPEAKER	ANDREWS,TH S
4	26,250.00	NaN	PERSONNEL COMPENSATION	NaN	03/31/16	OFFICE OF THE SPEAKER	ANTELL,GEOF

In [13]:

```
#Check if any column has null values

df.columns[df.isnull().any()].tolist()
```

Out[13]:

```
['BIOGUIDE_ID',
 'DATE',
 'END DATE',
 'PAYEE',
 'PROGRAM',
 'PURPOSE',
 'RECIP (orig.)',
 'RECORDID',
 'START DATE',
 'TRANSCODE',
 'TRANSCODELONG']
```

In [21]:

```
#Look only at 'PERSONNEL COMPENSATION' value in 'CATEGORY' column.
df = df[df['CATEGORY'] == 'PERSONNEL COMPENSATION']
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 56863 entries, 1 to 90165
Data columns (total 16 columns):
AMOUNT            56863 non-null object
BIOGUIDE_ID       42306 non-null object
CATEGORY          56863 non-null object
DATE              36 non-null object
END DATE          56863 non-null object
OFFICE            56863 non-null object
PAYEE             56858 non-null object
PROGRAM           12616 non-null object
PURPOSE           56861 non-null object
QUARTER           56863 non-null object
RECIP (orig.)     56858 non-null object
RECORDID          36 non-null object
START DATE        56863 non-null object
TRANSCODE         38 non-null object
TRANSCODELONG     31 non-null object
YEAR              56863 non-null object
dtypes: object(16)
memory usage: 7.4+ MB
```

In [22]:

```
df['BIOGUIDE_ID'].nunique()
```

Out[22]:

```
444
```

In [33]:

```
#Convert AMOUNT from a string to a float and check if the AMOUNT is positive.
df['AMOUNT'] = pd.to_numeric(df['AMOUNT'], errors='coerce')
df = df[df['AMOUNT'] > 0]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22710 entries, 16 to 90165
Data columns (total 16 columns):
AMOUNT            22710 non-null float64
BIOGUIDE_ID       17353 non-null object
CATEGORY          22710 non-null object
DATE              15 non-null object
END DATE          22710 non-null object
OFFICE            22710 non-null object
PAYEE             22707 non-null object
PROGRAM           12555 non-null object
PURPOSE           22710 non-null object
QUARTER           22710 non-null object
RECIP (orig.)     22707 non-null object
RECORDID          15 non-null object
START DATE        22710 non-null object
TRANSCODE         15 non-null object
TRANSCODELONG     13 non-null object
YEAR              22710 non-null object
dtypes: float64(1), object(15)
memory usage: 2.9+ MB
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/ipykernel_launc
her.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
```

In [36]:

```
#Columns we want are  BIOGUIDE_ID, PAYEE and AMOUNT
rep_df = df[['BIOGUIDE_ID', 'PAYEE', 'AMOUNT']]
rep_df.head()
```

Out[36]:

	BIOGUIDE_ID	PAYEE	AMOUNT
16	NaN	CASTINE,PETER L	416.46
30	NaN	GILLESPIE,JAMES M	25.24
40	NaN	JORDON,BENJAMIN D	194.44
41	NaN	KITTLE,ALLIE M	408.33
46	NaN	MARROLETTI,CHRISTOPHER V	832.92

In [38]:

```
#Remove all rows with NaN entries for BIOGUIDE_ID
rep_df = rep_df[rep_df['BIOGUIDE_ID'].notnull()]
rep_df.head()
```

Out[38]:

	BIOGUIDE_ID	PAYEE	AMOUNT
5393	A000374	ARNOLD,EMILY M	200.00
5405	A000374	PIERCE,ANN S	666.67
5565	A000374	ARNOLD,EMILY M	204.17
5566	A000374	AVERY,ROBERT C	379.17
5567	A000374	BOIES,LILIA C	233.33

In [46]:

```
groupby_rep = rep_df.groupby(['BIOGUIDE_ID']).sum()
groupby_rep.head()
```

Out[46]:

	AMOUNT
BIOGUIDE_ID	
A000055	278080.62
A000367	283904.27
A000369	287807.81
A000370	246273.94
A000371	257840.80

In [52]:

```
groupby_rep_payee = rep_df.groupby(['BIOGUIDE_ID', 'PAYEE']).sum()
groupby_rep_payee.head(3)
```

Out[52]:

		AMOUNT
BIOGUIDE_ID	**PAYEE**	
A000055	**ABERNATHY PAMELA M.**	17354.40
	CLARK CARSON G	17874.99
	DAWSON MARK E.	6000.00

In [55]:

```
groupby_rep_sum = rep_df.groupby(['BIOGUIDE_ID']).sum()
groupby_rep_sum.head()
```

Out[55]:

	AMOUNT
BIOGUIDE_ID	
A000055	278080.62
A000367	283904.27
A000369	287807.81
A000370	246273.94
A000371	257840.80

In [67]:

```
groupby_rep_sum['PAYEE_COUNT'] = rep_df.groupby('BIOGUIDE_ID')['PAYEE'
].nunique()
groupby_rep_sum.head()
```

Out[67]:

	AMOUNT	PAYEE_COUNT
BIOGUIDE_ID		
A000055	278080.62	22
A000367	283904.27	33
A000369	287807.81	36
A000370	246273.94	23
A000371	257840.80	33

In [77]:

```
groupby_rep_sum['AVG SALARY'] = groupby_rep_sum['AMOUNT']/groupby_rep_sum['PAYEE_COUNT']
groupby_rep_sum.sort_values(by='AVG SALARY', ascending = False).head()
```

Out[77]:

	AMOUNT	PAYEE_COUNT	AVG SALARY
BIOGUIDE_ID			
B001278	367105.55	21	17481.216667
D000626	231240.00	15	15416.000000
E000215	224578.49	15	14971.899333
H001070	267979.13	19	14104.164737
H001059	224223.08	16	14013.942500

In [ ]: