In [3]:
```
import pyspark
```

In [4]:
```
input_path = 'whitehouse_waves-2016_12.csv'
records = spark.sparkContext.textFile(input_path)
records.count()
```

Out[4]: 970505

remove the header / column names

In [14]:
```
## remove the header/first line/column names
header = records.first()
input_data = records.filter(lambda x: x != header)
input_data.count()
```

Out[14]: 970504

drop invalid records and convert to lowercase

In [7]:
```
# a. if a visitor's last name (i.e., NAMELAST) is null/empty, then drop that record
# b. if visitee_namelast is null/empty, then drop that record
rdd_filter = input_data.map(lambda x: x.lower().split(',')).filter(lambda x: x[0] and x[19])
rdd_filter.count()
```

Out[7]: 911249

Q1: The 10 most frequent visitors, (visitor, frequency)

In [8]:
```
## (i) The 10 most frequent visitors (NAMELAST, NAMEFIRST, NAMEMID) to the White House.
## <visitor> <frequency>
rdd_mf_visitors = rdd_filter.map(lambda x: ((x[0],x[1],x[2]), 1)).reduceByKey(lambda a,b: a+b)
rdd_mf_visitors.takeOrdered(10, key=lambda x: -x[1])
```

Out[8]:
```
[(('thomas', 'benjamin', 'l'), 185),
 (('berner', 'katherine', 'k'), 176),
 (('haas', 'jordan', 'm'), 152),
 (('grant', 'patrick', 'c'), 151),
 (('kidwell', 'lauren', 'k'), 145),
 (('haro', 'steven', 'm'), 140),
 (('garza', 'steven', 'a'), 127),
 (('strait', 'elan', ''), 107),
 (('lew', 'shoshana', 'm'), 102),
 (('zeitlin', 'daniel', 'l'), 98)]
```

Q2: The 10 most frequently visited people, (visitee, frequency)

In [9]:
```
## (ii) The 10 most frequently visited people (visitee_namelast, visitee_namefirst) in the White Ho
## <visitee> <frequency>
rdd_mf_visitee = rdd_filter.map(lambda x: ((x[19],x[20]), 1)).reduceByKey(lambda a,b: a+b)
rdd_mf_visitee.takeOrdered(10, key=lambda x: -x[1])
```

Out[9]:
```
[(('office', 'visitors'), 430881),
 (('waves', 'visitorsoffice'), 44129),
 (('bryant', 'ruth'), 13970),
 (('oneil', 'olivia'), 13155),
 (('thompson', 'jared'), 11618),
 (('/', 'potus'), 10900),
 (('burton', 'collin'), 9672),
 (('megan', 'matthew'), 7944),
 (('mayerson', 'asher'), 6886),
 (('dessources', 'kalisha'), 5289)]
```

Q3: The 10 most frequently visitor-visitee combinations, (visitor-visitee, frequency)

In [10]:
```
## (iii) The 10 most frequent visitor-visitee combinations. <visitor-visitee> <frequency>
rdd_mf_vv = rdd_filter.map(lambda x: ((x[0],x[1],x[2],x[19],x[20]), 1)).reduceByKey(lambda a,b: a+
rdd_mf_vv.takeOrdered(10, key=lambda x: -x[1])
```

Out[10]:
```
[(('haas', 'jordan', 'm', 'yudelson', 'alex'), 90),
 (('thomas', 'benjamin', 'l', 'yudelson', 'alex'), 89),
 (('grant', 'patrick', 'c', 'yudelson', 'alex'), 88),
 (('berner', 'katherine', 'k', 'yudelson', 'alex'), 82),
 (('roche', 'shannon', 'e', 'yudelson', 'alex'), 70),
 (('urizar', 'jennifer', 'a', 'johnson', 'katie'), 68),
 (('martin', 'kathryn', '', 'lambrew', 'jeanne'), 56),
 (('kidwell', 'lauren', 'k', 'abraham', 'yohannes'), 55),
 (('haas', 'jordan', 'm', 'abraham', 'yohannes'), 54),
 (('angerman', 'elizabeth', '', 'mader', 'david'), 54)]
```

Q4: The number of records dropped

In [13]:
```
## The number of records dropped.
print("Q4 The number of records dropped: ", input_data.count() - rdd_filter.count())
```

```
Q4 The number of records dropped:  59255
```

In [ ]:
```

```