In [1]:
```
import pyspark
from pyspark.sql.functions import lower, col
```

In [2]:
```
input_path = 'whitehouse_waves-2016_12.csv'
df = spark.read.format('csv').option('header','true').option('interSchema','true').load(input_path)
```

In [3]:
```
df.printSchema()
```

```
root
 |-- NAMELAST: string (nullable = true)
 |-- NAMEFIRST: string (nullable = true)
 |-- NAMEMID: string (nullable = true)
 |-- UIN: string (nullable = true)
 |-- BDGNBR: string (nullable = true)
 |-- ACCESS_TYPE: string (nullable = true)
 |-- TOA: string (nullable = true)
 |-- POA: string (nullable = true)
 |-- TOD: string (nullable = true)
 |-- POD: string (nullable = true)
 |-- APPT_MADE_DATE: string (nullable = true)
 |-- APPT_START_DATE: string (nullable = true)
 |-- APPT_END_DATE: string (nullable = true)
 |-- APPT_CANCEL_DATE: string (nullable = true)
 |-- Total_People: string (nullable = true)
 |-- LAST_UPDATEDBY: string (nullable = true)
 |-- POST: string (nullable = true)
 |-- LASTENTRYDATE: string (nullable = true)
 |-- TERMINAL_SUFFIX: string (nullable = true)
 |-- visitee_namelast: string (nullable = true)
 |-- visitee_namefirst: string (nullable = true)
 |-- MEETING_LOC: string (nullable = true)
 |-- MEETING_ROOM: string (nullable = true)
 |-- CALLER_NAME_LAST: string (nullable = true)
 |-- CALLER_NAME_FIRST: string (nullable = true)
 |-- CALLER_ROOM: string (nullable = true)
 |-- DESCRIPTION: string (nullable = true)
 |-- Release_Date: string (nullable = true)
```

In [4]:
```
print(df.count())
print(df.columns)
```

```
970504
['NAMELAST', 'NAMEFIRST', 'NAMEMID', 'UIN', 'BDGNBR', 'ACCESS_TYPE', 'TOA', 'POA', 'TOD', 'POD',
'APPT_MADE_DATE', 'APPT_START_DATE', 'APPT_END_DATE', 'APPT_CANCEL_DATE', 'Total_People', 'LAST_
UPDATEDBY', 'POST', 'LASTENTRYDATE', 'TERMINAL_SUFFIX', 'visitee_namelast', 'visitee_namefirst',
'MEETING_LOC', 'MEETING_ROOM', 'CALLER_NAME_LAST', 'CALLER_NAME_FIRST', 'CALLER_ROOM', 'DESCRIPT
ION', 'Release_Date']
```

select the useful columns

In [5]:
```
df_select = df.select('NAMELAST','NAMEFIRST', 'NAMEMID', 'visitee_namelast', 'visitee_namefirst')
df_select.show(3)
```

```
+--------------+---------+-------+---------------+----------------+
|      NAMELAST|NAMEFIRST|NAMEMID|visitee_namelast|visitee_namefirst|
+--------------+---------+-------+---------------+----------------+
|TAJOURIBESSASSI|   HANENE|   null|        Pelofsky|            Eric|
|       bageant|    laura|      j|     Baskerville|          Steven|
|      Broemson|     Earl|      H|     Baskerville|          Steven|
+--------------+---------+-------+---------------+----------------+
only showing top 3 rows
```

drop invalid records and convert to lowercase

In [6]:
```
# 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
# c. convert all characters to lowercase letters
# d. If a given visitor's lastname or visitee's last name has non-English characters thenthat recor

df_filter = df_select.na.drop(subset=['NAMELAST','visitee_namelast'])
df_lower = df_filter.withColumn('NAMELAST', lower(col('NAMELAST'))) \
                    .withColumn('NAMEFIRST', lower(col('NAMEFIRST'))) \
                    .withColumn('NAMEMID', lower(col('NAMEMID'))) \
                    .withColumn('visitee_namelast', lower(col('visitee_namelast'))) \
                    .withColumn('visitee_namefirst', lower(col('visitee_namefirst')))
df_letter = df_lower.filter(df_lower.NAMELAST.rlike('^[a-z]+$')).filter(df_lower.visitee_namelast.r
print(df_letter.count())
df_letter.show()
```

```
897037
+--------------+---------+-------+---------------+----------------+
|      NAMELAST|NAMEFIRST|NAMEMID|visitee_namelast|visitee_namefirst|
+--------------+---------+-------+---------------+----------------+
|tajouribessassi|   hanene|   null|        pelofsky|            eric|
|       bageant|    laura|      j|     baskerville|          steven|
|      broemson|     earl|      h|     baskerville|          steven|
|       mccrary|  richard|      l|     baskerville|          steven|
|       mulcahy|   joshua|      e|     baskerville|          steven|
|          ryan|   oliver|      j|     baskerville|          steven|
|        keeler|  douglas|      e|       goldstein|            jeff|
|         davis|   justin|      a|            drew|            maj|
|        glover|   vinson|      n|         lengyel|          jason|
|        ambler|   andrew|      s|          office|        visitors|
|        ambler|     john|      s|          office|        visitors|
|      anderson|    cindy|      l|          office|        visitors|
|      anderson|    wayne|      s|          office|        visitors|
|       andrade|   andrea|      m|          office|        visitors|
|       arcelle|   jeanne|      l|          office|        visitors|
|       arcelle|     mark|   null|          office|        visitors|
|        arnold|   curtis|   null|          office|        visitors|
|         baade|    kraig|      d|          office|        visitors|
|        bailey|  trinity|      n|          office|        visitors|
|         baird|    scott|      d|          office|        visitors|
+--------------+---------+-------+---------------+----------------+
only showing top 20 rows
```

save df as table visitlog

In [7]:
```
df_letter.createOrReplaceTempView('visitlog')
```

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>
df_mf_visitors = spark.sql('select NAMELAST, NAMEFIRST, NAMEMID, count(*) as frequency from visitlo
df_mf_visitors.show()
```

```
+--------+---------+-------+---------+
|NAMELAST|NAMEFIRST|NAMEMID|frequency|
+--------+---------+-------+---------+
|  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|   null|      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>
df_mf_visitee = spark.sql('select visitee_namelast, visitee_namefirst, count(*) as frequency from v
df_mf_visitee.show()
```

```
+---------------+----------------+---------+
|visitee_namelast|visitee_namefirst|frequency|
+---------------+----------------+---------+
|          office|         visitors|   430721|
|           waves|   visitorsoffice|    44115|
|          bryant|            ruth|    13970|
|           oneil|           olivia|    13155|
|        thompson|           jared|    11605|
|          burton|           collin|     9672|
|           megan|          matthew|     7943|
|        mayerson|           asher|     6885|
|      dessources|          kalisha|     5285|
|           evans|           karen|     2908|
+---------------+----------------+---------+
```

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

In [10]:
```
## (iii) The 10 most frequent visitor-visitee combinations. <visitor-visitee> <frequency>
df_mf_vv = spark.sql('select NAMELAST, NAMEFIRST, NAMEMID, visitee_namelast, visitee_namefirst, cou
df_mf_vv.show()
```

```
+--------+---------+-------+----------------+-----------------+---------+
|NAMELAST|NAMEFIRST|NAMEMID|visitee_namelast|visitee_namefirst|frequency|
+--------+---------+-------+----------------+-----------------+---------+
|    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|   null|          lambrew|           jeanne|       56|
| kidwell|   lauren|      k|          abraham|         yohannes|       55|
|  berner|katherine|      k|          abraham|         yohannes|       54|
|    haas|   jordan|      m|          abraham|         yohannes|       54|
+--------+---------+-------+----------------+-----------------+---------+
```

Q4: The number of records dropped

In [11]:
```
## The number of records dropped.
print("Q4 The number of records dropped: ", df.count() - df_letter.count())
```

```
Q4 The number of records dropped:  73467
```

In [ ]:
```

```