Clean Common Errors from Names Using NLP

In [1]:
```
import pandas as pd
import numpy as np
pd.set_option('display.max_colwidth', None)
pd.set_option("display.max_row",None)
import nltk
import spacy
import re
```

In [5]:
```
df = pd.read_csv("catalog\main_file_all_text.csv")
df.drop(columns='Unnamed: 0',inplace=True)
df.rename(columns={"0":"Text"},inplace=True)
# Replace the common unwanted words appearing in the beginning of rows
df.Text = df.Text.str.replace(r"MICROFILM|MICROFILM MANUSCRIPTS|Treasure room|MANUSCRIPTS Restricted|MANUSCRIPT
"",case=False)
```

```
<ipython-input-5-8b8d5ed3cfb4>:5: FutureWarning: The default value of regex will change from True to False in a
future version.
  df.Text = df.Text.str.replace(r"MICROFILM|MICROFILM MANUSCRIPTS|Treasure room|MANUSCRIPTS Restricted|MANUSCRI
PTS|^Chapel|MSS|NUCMC|^FILM|^RESTRICTED",
```

In [3]:
```
# CHUNKING WITH NO START LETTER PARAMETER
def chunk(df):
    word_tok = nltk.word_tokenize(df[:50])
    taged_sent= nltk.pos_tag(word_tok)
    # Checks for Proper Noun,  coordinating conjunction(and,&), Proper Nouns
    #TODO Improve this pattern to better extract names
    grammar = "Name: {((<NNP><,>)?<NNP><.>?<CC>?<NNP>?<CC>?<NNP>*)}"
    cp = nltk.RegexpParser(grammar,loop=1)
    chunked = cp.parse(taged_sent)
    for subtree in chunked.subtrees(filter =  lambda x : x.label()=="Name"):
         # Generate all subtrees
        return " ".join([i[0] for i in subtree.leaves()])


df["Name"] = df.Text.apply(chunk)
# Modify comma and space, replace common unwanted word endings/titles

endings = ["Papers", "Letters", "Diary", "Notebook", "Book", "Scrapbook", "Screenplay", "Memoir", "Card", "Dayb
           "Account", "Sketch", "Journal", "Letter", "Record", "Notes", "Ledger", "Rent", "Letterpress", "Addre
for i in endings:
    df.Name = df.Name.str.replace(i,"",case=False)
df.Name = df.Name.str.replace(" , ",", ")
df.Name = df.Name.str.replace("For Information.*","")
df.Name = df.Name.str.strip()
df.Name = df.Name.str.title()
```

In [2]:
```
# CHUNKING WITH START LETTER PARAMETER
def chunk(df,start_letter):
    start_letter = start_letter.upper()
    df =  re.sub(r'[0-9]+', '', df)
    first_ind = df.find(start_letter)
    # If word starting with start letter does not exist, return None
    if(first_ind ==-1):
        return None
    word_tok = nltk.word_tokenize(df[first_ind:50+first_ind])
    taged_sent= nltk.pos_tag(word_tok)
    # Checks for Proper Noun,  coordinating conjunction(and,&), Proper Nouns
    grammar = "Name: {((<NN.><,>)?<NNP><.>?<CC>?<NNP>?<CC>?<NNP>*)}"
    cp = nltk.RegexpParser(grammar,loop=1)
    chunked = cp.parse(taged_sent)
    for subtree in chunked.subtrees(filter =  lambda x : x.label()=="Name"):
         # Generate all subtrees
        li = [i[0] for i in subtree.leaves()]
#         print(li)

        if li[0].startswith(start_letter):
            return " ".join([i[0] for i in subtree.leaves()])
        # if no chunk start with the letter
        # Get the first word starting with the letter
        for i in chunked.leaves():
            if i[0].startswith(start_letter):
                first = i[0]
        return first + ", "+ " ".join(li)


# Apply chunking / put in start_letter
df["Name"] = df.Text.apply(chunk, start_letter)
# Modify comma and space, replace common unwanted word endings/titles

endings = ["Papers", "Letters", "Diary", "Notebook", "Book", "Scrapbook", "Screenplay", "Memoir", "Card", "Dayb
           "Account", "Sketch", "Journal", "Letter", "Record", "Notes", "Ledger", "Rent", "Letterpress", "Addre
for i in endings:
    df.Name = df.Name.str.replace(i,"",case=False)
df.Name = df.Name.str.replace(" , ",", ")
df.Name = df.Name.str.replace("For Information.*","")
df.Name = df.Name.str.strip()
df.Name = df.Name.str.title()
```

In [7]:
```
df.to_csv("all_text_chunked_name.csv",index=False)
```