Card Catalog Spatial Frequencies

While the maps and charts in this notebook may not be entirely accurate to the true contents of the catalog, due to OCR errors compounding through the data pipeline, they serve to give a general idea of the geographic demographics of the manuscript collections cataloged in the Rubenstein Library Card Catalog.

Many of the collections cataloged in the library contain a metadata field of the location of where the items in that collection come from. Using the Python SpaCy package, we attempted to extract these locations. Using those which we could glean from the data, we created spatial heat maps of cards from the United States and North Carolina counties and computed the international card counts using the GeoPandas and matplotlib packages.

In [1]:
```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="darkgrid")
%matplotlib inline
```

Spatial Frequency USA Map

In [3]:
```
# Read shapefile and load in card catalog dataset
usa = gpd.read_file('spatial/tl_2020_us_state/tl_2020_us_state.shp')

df = pd.read_csv('~/main_file_dataset.csv')

# Sort alphabetically
usa = usa.sort_values(by=['STATEFP'])
```

In [4]:
```
# Store names and counts
# State Name    USPS Abbreviation       Traditional Abbreviation

states = {
"Alabama-AL-Ala.": 0, "Alaska-AK-Alaska": 0, "Arizona-AZ-Ariz.": 0, "Arkansas-AR-Ark.": 0, "California-CA-Calif
"Colorado-CO-Colo.": 0, "Connecticut-CT-Conn.": 0, "Delaware-DE-Del.": 0, "District of Columbia-DC-D.C.": 0,
"Florida-FL-Fla.": 0, "Georgia-GA-Ga.": 0, "Hawaii-HI-Hawaii": 0, "Idaho-ID-Idaho": 0, "Illinois-IL-Ill.": 0,
"Indiana-IN-Ind.": 0, "Iowa-IA-Iowa": 0, "Kansas-KS-Kans.": 0, "Kentucky-KY-Ky.": 0, "Louisiana-LA-La.": 0,
"Maine-ME-Maine": 0, "Maryland-MD-Md.": 0, "Massachusetts-MA-Mass.": 0, "Michigan-MI-Mich.": 0, "Minnesota-MN-M
"Mississippi-MS-Miss.": 0, "Missouri-MO-Mo.": 0, "Montana-MT-Mont.": 0, "Nebraska-NE-Neb. or Nebr.": 0, "Nevada
"New Hampshire-NH-N.H.": 0, "New Jersey-NJ-N.J.": 0, "New Mexico-NM-N.Mex.": 0, "New York-NY-N.Y.": 0,
"North Carolina-NC-N.C.": 0, "North Dakota-ND-N.Dak.": 0, "Ohio-OH-Ohio": 0, "Oklahoma-OK-Okla.": 0, "Oregon-OF
"Pennsylvania-PA-Pa.": 0, "Rhode Island-RI-R.I.": 0, "South Carolina-SC-S.C.": 0, "South Dakota-SD-S.Dak.": 0,
"Tennessee-TN-Tenn.": 0, "Texas-TX-Tex. or Texas": 0, "Utah-UT-Utah": 0, "Vermont-VT-Vt.": 0, "Virginia-VA-Va.'
"Washington-WA-Wash.": 5, "West Virginia-WV-W.Va.": 0, "Wisconsin-WI-Wis. or Wisc.": 0, "Wyoming-WY-Wyo.": 0,
"American Samoa-AS-Amer. Samoa": 0, "Guam-GU-Guam": 0, "Northern Mariana Islands-MP-M.P.": 0, "Puerto Rico-PR-F
"Virgin Islands-VI-V.I.": 0}
```

In [5]:
```
# Add state counts from catalog dataset
# Function to check if location is a state and add to counts
def find_state(s):
    # Check in list of states
    s = s.strip()
    if len(s) > 1:
        for key in states:
            if key != "Washington-WA-Wash." and ((s + "-") in key or ("-" + s) in key):
                states[key] = states.get(key) + 1
                break

# Loop through each main entry and try to gather location
for index, row in df.iterrows():
    if row['Coll_head'] == 1:
        # Try to get state name or abbreviation
        locs = str(row['Loc']).split(",")
        if len(locs) < 1:
            continue
        elif len(locs) == 1:
            find_state(locs[0])
        elif len(locs) == 2:
            find_state(locs[1])
            for i in range(2, len(locs)):
                find_state(locs[i])

print(states)
```

```
{'Alabama-AL-Ala.': 153, 'Alaska-AK-Alaska': 0, 'Arizona-AZ-Ariz.': 1, 'Arkansas-AR-Ark.': 16, 'California-CA-C
alif.': 11, 'Colorado-CO-Colo.': 3, 'Connecticut-CT-Conn.': 76, 'Delaware-DE-Del.': 5, 'District of Columbia-DC
-D.C.': 43, 'Florida-FL-Fla.': 17, 'Georgia-GA-Ga.': 371, 'Hawaii-HI-Hawaii': 1, 'Idaho-ID-Idaho': 0, 'Illinois
-IL-Ill.': 25, 'Indiana-IN-Ind.': 21, 'Iowa-IA-Iowa': 13, 'Kansas-KS-Kans.': 3, 'Kentucky-KY-Ky.': 53, 'Louisia
na-LA-La.': 71, 'Maine-ME-Maine': 11, 'Maryland-MD-Md.': 218, 'Massachusetts-MA-Mass.': 169, 'Michigan-MI-Mic
h.': 12, 'Minnesota-MN-Minn.': 3, 'Mississippi-MS-Miss.': 63, 'Missouri-MO-Mo.': 33, 'Montana-MT-Mont.': 0, 'Ne
braska-NE-Neb. or Nebr.': 3, 'Nevada-NV-Nev.': 0, 'New Hampshire-NH-N.H.': 22, 'New Jersey-NJ-N.J.': 17, 'New M
exico-NM-N.Mex.': 6, 'New York-NY-N.Y.': 302, 'North Carolina-NC-N.C.': 1188, 'North Dakota-ND-N.Dak.': 1, 'Ohi
o-OH-Ohio': 63, 'Oklahoma-OK-Okla.': 3, 'Oregon-OR-Ore.-Oreg.': 2, 'Pennsylvania-PA-Pa.': 193, 'Rhode Island-RI
-R.I.': 19, 'South Carolina-SC-S.C.': 299, 'South Dakota-SD-S.Dak.': 0, 'Tennessee-TN-Tenn.': 81, 'Texas-TX-Te
x. or Texas': 22, 'Utah-UT-Utah': 4, 'Vermont-VT-Vt.': 13, 'Virginia-VA-Va.': 1631, 'Washington-WA-Wash.': 5,
'West Virginia-WV-W.Va.': 48, 'Wisconsin-WI-Wis. or Wisc.': 3, 'Wyoming-WY-Wyo.': 1, 'American Samoa-AS-Amer. S
amoa': 0, 'Guam-GU-Guam': 0, 'Northern Mariana Islands-MP-M.P.': 0, 'Puerto Rico-PR-P.R.': 1, 'Virgin Islands-V
I-V.I.': 0}
```

In [8]:
```
# State counts after running above code
states = {'Alabama-AL-Ala.': 153, 'Alaska-AK-Alaska': 0, 'Arizona-AZ-Ariz.': 1, 'Arkansas-AR-Ark.': 16, 'Califo
```

In [6]:
```
# Add state counts to geo dataframe
vals = states.values()
usa['Count'] = vals
```

In [7]:
```
# Print initial heatmap---code adapted from https://medium.com/@m_vemuri/create-a-geographic-heat-map-of-the-ci
fig, ax = plt.subplots(1, figsize=(40, 20))
ax.axis('off')
ax.set_title("USA Spatial Frequency of Card Catalog Manuscripts", fontdict={'fontsize': '50', 'fontweight' : '3

color = 'YlGnBu'
sm = plt.cm.ScalarMappable(cmap=color, norm=plt.Normalize(vmin=0, vmax=1700))
sm._A = []
cbar = fig.colorbar(sm)
cbar.ax.tick_params(labelsize=20)
plt.xlim([-130, -60])
plt.ylim([20, 50])

usa.plot('Count', cmap=color, linewidth=0.6, ax=ax, edgecolor="black", figsize=(40, 20))

# Add quantity labels, code adapted from https://stackoverflow.com/questions/38899190/geopandas-label-polygons
usa_nums = usa.copy()
usa_nums['coords'] = usa_nums['geometry'].apply(lambda x: x.centroid.coords[:])
usa_nums['coords'] = [coords[0] for coords in usa_nums['coords']]
# Remove DC to avoid label clashes
usa_nums.drop(usa_nums.index[8], inplace = True)

for index, row in usa_nums.iterrows():
    plt.annotate(text=row['Count'], xy=row['coords'], horizontalalignment='center', color='#C84E00',
                fontsize=20, fontweight='bold')
```

Based on the heat map, we can see that the states with the most hits are Virginia and North Carolina. New York, South Carolina, and Georgia are also pretty common. With this algorithm, Washington had around 200 hits; however, it counted the name "Washington," Washington DC, and several counties called Washington incorrectly, so the value of 5 was manually added. Alaska does not have any cards, so it was omitted from the map. Outside of the continental US, Hawaii and Puerto Rico have one hit each. Washington DC has a lot of hits, but was not always picked up from the location tagging code, due to the recurring OCR error of "DeCe," so we do not know how many cards from DC there actually are. There are some odd outliers here e.g., why is North Carolina not the most represented state? Perhaps NC originating cards were labeled with county instead of state, let's check it out.

Spatial Frequency Map by N.C. County

In [9]:
```
# Read in NC county shapefile
nc = gpd.read_file('nc_counties/nc_covid.shp')
nc.drop(['cases', 'deaths', 'case100', 'death100'], axis=1, inplace=True)
```

In [10]:
```
# Dictionary of NC county names

counties = {
"Alamance County": 0, "Alexander County": 0, "Alexander County": 0, "Alleghany County": 0, "Anson County": 0, '
"Avery County": 0, "Beaufort County": 0, "Bertie County": 0, "Bladen County": 0, "Brunswick County": 0, "Buncon
"Burke County": 0, "Cabarrus County": 0, "Caldwell County": 0, "Camden County": 0, "Carteret County": 0, "Caswe
"Catawba County": 0, "Chatham County": 0, "Cherokee County": 0, "Chowan County": 0, "Clay County": 0, "Clevelar
"Columbus County": 0, "Craven County": 0, "Cumberland County": 0, "Currituck County": 0, "Dare County": 0, "Dav
"Davie County": 0, "Duplin County": 0, "Durham County": 0, "Edgecombe County": 0, "Forsyth County": 0, "Frankli
"Gaston County": 0, "Gates County": 0, "Graham County": 0, "Granville County": 0, "Greene County": 0, "Guilforc
"Halifax County": 0, "Harnett County": 0, "Haywood County": 0, "Henderson County": 0, "Hertford County": 0, "Hc
"Hyde County": 0, "Iredell County": 0, "Jackson County": 0, "Johnston County": 0, "Jones County": 0, "Lee Count
"Lenoir County": 0, "Lincoln County": 0, "Macon County": 0, "Madison County": 0, "Martin County": 0, "McDowell
"Mecklenburg County": 0, "Mitchell County": 0, "Montgomery County": 0, "Moore County": 0, "Nash County": 0,
"New Hanover County": 0, "Northampton County": 0, "Onslow County": 0, "Orange County": 0, "Pamlico County": 0,
"Pasquotank County": 0, "Pender County": 0, "Perquimans County": 0, "Person County": 0, "Pitt County": 0, "Poll
"Randolph County": 0, "Richmond County": 0, "Robeson County": 0, "Rockingham County": 0, "Rowan County": 0,
"Rutherford County": 0, "Sampson County": 0, "Scotland County": 0, "Stanly County": 0, "Stokes County": 0, "Sur
"Swain County": 0, "Transylvania County": 0, "Tyrrell County": 0, "Union County": 0, "Vance County": 0, "Wake C
"Warren County": 0, "Washington County": 0, "Watauga County": 0, "Wayne County": 0, "Wilkes County": 0, "Wilsor
"Yadkin County": 0, "Yancey County": 0}
```

In [11]:
```
# Collect county card counts
# Function to check if location is a county and add to counts
def find_county(c):
    # Check in list of counties
    for key in counties:
        if c == key or c.strip(".") == key.rstrip("unty"):
            counties[key] = counties.get(key) + 1
            break

# Loop through each main entry and try to gather location
for index, row in df.iterrows():
    if row['Coll_head'] == 1:
        # Try to get county name
        locs = str(row['Loc']).split(",")
        if len(locs) < 1:
            continue
        else:
            for i in range(0, len(locs)):
                find_county(locs[i])

print(counties)
```

```
{'Alamance County': 6, 'Alexander County': 2, 'Alleghany County': 0, 'Anson County': 7, 'Ashe County': 0, 'Aver
y County': 0, 'Beaufort County': 11, 'Bertie County': 9, 'Bladen County': 3, 'Brunswick County': 13, 'Buncombe
County': 10, 'Burke County': 7, 'Cabarrus County': 5, 'Caldwell County': 2, 'Camden County': 0, 'Carteret Count
y': 0, 'Caswell County': 10, 'Catawba County': 3, 'Chatham County': 11, 'Cherokee County': 5, 'Chowan County':
5, 'Clay County': 1, 'Cleveland County': 2, 'Columbus County': 0, 'Craven County': 6, 'Cumberland County': 4,
'Currituck County': 0, 'Dare County': 1, 'Davidson County': 10, 'Davie County': 3, 'Duplin County': 1, 'Durham
County': 31, 'Edgecombe County': 5, 'Forsyth County': 4, 'Franklin County': 16, 'Gaston County': 2, 'Gates Coun
ty': 0, 'Graham County': 0, 'Granville County': 18, 'Greene County': 5, 'Guilford County': 9, 'Halifax County':
25, 'Harnett County': 5, 'Haywood County': 2, 'Henderson County': 1, 'Hertford County': 2, 'Hoke County': 0, 'H
yde County': 5, 'Iredell County': 21, 'Jackson County': 3, 'Johnston County': 6, 'Jones County': 2, 'Lee Count
y': 1, 'Lenoir County': 4, 'Lincoln County': 5, 'Macon County': 3, 'Madison County': 3, 'Martin County': 8, 'Mc
Dowell County': 0, 'Mecklenburg County': 9, 'Mitchell County': 0, 'Montgomery County': 14, 'Moore County': 1,
'Nash County': 2, 'New Hanover County': 4, 'Northampton County': 5, 'Onslow County': 1, 'Orange County': 23, 'P
amlico County': 0, 'Pasquotank County': 0, 'Pender County': 0, 'Perquimans County': 4, 'Person County': 3, 'Pit
t County': 1, 'Polk County': 0, 'Randolph County': 26, 'Richmond County': 5, 'Robeson County': 6, 'Rockingham C
ounty': 7, 'Rowan County': 13, 'Rutherford County': 0, 'Sampson County': 3, 'Scotland County': 0, 'Stanly Count
y': 1, 'Stokes County': 7, 'Surry County': 10, 'Swain County': 0, 'Transylvania County': 0, 'Tyrrell County':
0, 'Union County': 5, 'Vance County': 0, 'Wake County': 26, 'Warren County': 8, 'Washington County': 20, 'Watau
ga County': 0, 'Wayne County': 9, 'Wilkes County': 10, 'Wilson County': 0, 'Yadkin County': 5, 'Yancey County':
0}
```

In [1]:
```
# County counts stored from running above code
counties = {'Alamance County': 6, 'Alexander County': 2, 'Alleghany County': 0, 'Anson County': 7, 'Ashe County
```

In [12]:
```
# Add county counts to geo dataframe
vals = counties.values()
nc['Count'] = vals
```

In [14]:
```
# Print nc county heatmap---code adapted from article linked above
fig, ax = plt.subplots(1, figsize=(40, 20))
ax.axis('off')
ax.set_title("NC County Card Catalog Frequency", fontdict={'fontsize': '50', 'fontweight' : '3'})

color = 'YlGnBu'
sm = plt.cm.ScalarMappable(cmap=color, norm=plt.Normalize(vmin=0, vmax=31))
sm._A = []
cbar = fig.colorbar(sm)
cbar.ax.tick_params(labelsize=20)

nc.plot('Count', cmap=color, linewidth=0.5, ax=ax, edgecolor="black", figsize=(40, 20))

# Add quantity labels, code adapted from https://stackoverflow.com/questions/38899190/geopandas-label-polygons
nc_nums = nc.copy()
nc_nums['coords'] = nc_nums['geometry'].apply(lambda x: x.centroid.coords[:])
nc_nums['coords'] = [coords[0] for coords in nc_nums['coords']]

for index, row in nc_nums.iterrows():
    plt.annotate(text=row['Count'], xy=row['coords'], horizontalalignment='center', color='#C84E00',
                fontsize=18, fontweight='bold')
```

When we add up all the county collection counts up, we get 551 cards cataloging collections that are specifically from North Carolina counties. It looks like there are a lot of cards from Durham County--which makes sense because the Rubenstein Library is located in it--as well as Washington County, along with the counties that border Durham. After checking for overlap, most of the cards with a county also have North Carolina or an abbreviation of the state. So, the question still remains around why there are so many more cards from Virginia. According to our word cloud, the civil war is a very common topic in the catalog, and as much of this was fought in Virginia, perhaps that could explain why there are so many cards from the non-NC state.

International Country Counts

We've seen where the cards in the United States hail from, but what about the rest of the world? Let's see how many cards we have from other countries.

In [2]:
```
# Dictionary of non-US countries pre-1990

countries = {
"Afghanistan": 0, "Albania": 0, "Algeria": 0, "Andorra": 0, "Angola": 0, "Antigua and Barbuda": 0, "Argentina":
"Australia": 0, "Austria": 0, "Bahamas": 0, "Bahrain": 0, "Bangladesh": 0, "Barbados": 0, "Belgium": 0, "Belize
"Bhutan": 0, "Bolivia": 0, "Bosnia and Herzegovina": 0, "Botswana": 0, "Brazil": 0, "Brunei": 0, "Bulgaria": 0,
"Burkina Faso": 0, "Burma": 0, "Burundi": 0, "Cabo Verde": 0, "Cambodia": 0, "Cameroon": 0, "Canada": 0,
"Central African Republic": 0, "Chad": 0, "Chile": 0, "China": 0, "Colombia": 0, "Comoros": 0, "Democratic Repu
"Republic of the Congo": 0, "Costa Rica": 0, "Côte d'Ivoire": 0, "Croatia": 0, "Cuba": 0, "Cyprus": 0, "Czech F
"Denmark": 0, "Djibouti": 0, "Dominica": 0, "Dominican Republic": 0, "Ecuador": 0, "Egypt": 0, "El Salvador": (
"Equatorial Guinea": 0, "Eritrea": 0, "Eswatini": 0, "Ethiopia": 0, "Fiji": 0, "Finland": 0, "France": 0, "Gabc
"Germany": 0, "Ghana": 0, "Greece": 0, "Grenada": 0, "Guatemala": 0, "Guinea": 0, "Guinea-Bissau": 0, "Guyana":
"Honduras": 0, "Hungary": 0, "Iceland": 0, "India": 0, "Indonesia": 0, "Iran": 0, "Iraq": 0, "Ireland": 0, "Isı
"Jamaica": 0, "Japan": 0, "Jordan": 0, "Kenya": 0, "Kiribati": 0, "North Korea": 0, "South Korea": 0, "Kosovo":
"Laos": 0, "Lebanon": 0, "Lesotho": 0, "Liberia": 0, "Libya": 0, "Liechtenstein": 0, "Luxembourg": 0, "Madagasc
"Malaysia": 0, "Maldives": 0, "Mali": 0, "Malta": 0, "Marshall Islands": 0, "Mauritania": 0, "Mauritius": 0, "M
"Micronesia": 0, "Monaco": 0, "Mongolia": 0, "Montenegro": 0, "Morocco": 0, "Mozambique": 0, "Namibia": 0, "Nau
"Netherlands": 0, "New Zealand": 0, "Nicaragua": 0, "Niger": 0, "Nigeria": 0, "North Macedonia": 0, "Norway": (
"Pakistan": 0, "Palau": 0, "Panama": 0, "Papua New Guinea": 0, "Paraguay": 0, "Peru": 0, "Philippines": 0, "Pol
"Qatar": 0, "Romania": 0, "Rwanda": 0, "Saint Kitts and Nevis": 0, "Saint Lucia": 0, "Saint Vincent and the Gre
"Samoa": 0, "San Marino": 0, "Sao Tome and Principe": 0, "Saudi Arabia": 0, "Scotland": 0, "Senegal": 0, "Serbi
"Sierra Leone": 0, "Singapore": 0, "Slovakia": 0, "Slovenia": 0, "Solomon Islands": 0, "Somalia": 0, "South Afı
"Sri Lanka": 0, "Sudan": 0, "Sudan, South": 0, "Suriname": 0, "Sweden": 0, "Switzerland": 0, "Syria": 0, "Taiwa
"Thailand": 0, "Timor-Leste": 0, "Togo": 0, "Tonga": 0, "Trinidad and Tobago": 0, "Tunisia": 0, "Turkey": 0, "T
"United Arab Emirates": 0, "United Kingdom": 0, "Uruguay": 0, "Vanuatu": 0, "Vatican City": 0, "Venezuela": 0,
"Wales": 0, "Yemen": 0, "Zambia": 0, "Zimbabwe": 0}
```

In [17]:
```
# Collect country card counts
# Function to check if location is a country and add to counts
def find_country(c):
    # Check in list of countries
    for key in countries:
        if c.strip(".") == key:
            countries[key] = countries.get(key) + 1
            continue

# Loop through each main entry and try to gather location
for index, row in df.iterrows():
    if row['Coll_head'] == 1:
        # Try to get county name
        locs = str(row['Loc']).split(",")
        if len(locs) < 1:
            continue
        else:
            for i in range(0, len(locs)):
                find_country(locs[i])
```

In [18]:
```
temp = {}
for key, value in countries.items():
    if value > 0:
        temp[key] = value

print(temp)
```

```
{'Australia': 1, 'Austria': 3, 'Belgium': 1, 'Brazil': 8, 'Canada': 1, 'Chile': 1, 'China': 5, 'Colombia': 1,
'Cuba': 4, 'Denmark': 4, 'Egypt': 3, 'England': 63, 'Fiji': 1, 'France': 47, 'Germany': 30, 'Greece': 3, 'Grena
da': 2, 'Guatemala': 1, 'India': 25, 'Indonesia': 1, 'Iran': 1, 'Ireland': 5, 'Israel': 3, 'Italy': 3, 'Jamaic
a': 4, 'Japan': 9, 'Jordan': 13, 'Lebanon': 2, 'Liberia': 1, 'Madagascar': 1, 'Malta': 2, 'Mexico': 8, 'Morocc
o': 1, 'New Zealand': 1, 'Peru': 5, 'Philippines': 3, 'Poland': 4, 'San Marino': 1, 'South Africa': 5, 'Spain':
22, 'Suriname': 1, 'Sweden': 2, 'Switzerland': 5, 'Syria': 1, 'Thailand': 1, 'Tunisia': 1, 'Turkey': 3, 'Vietna
m': 3, 'Wales': 2}
```

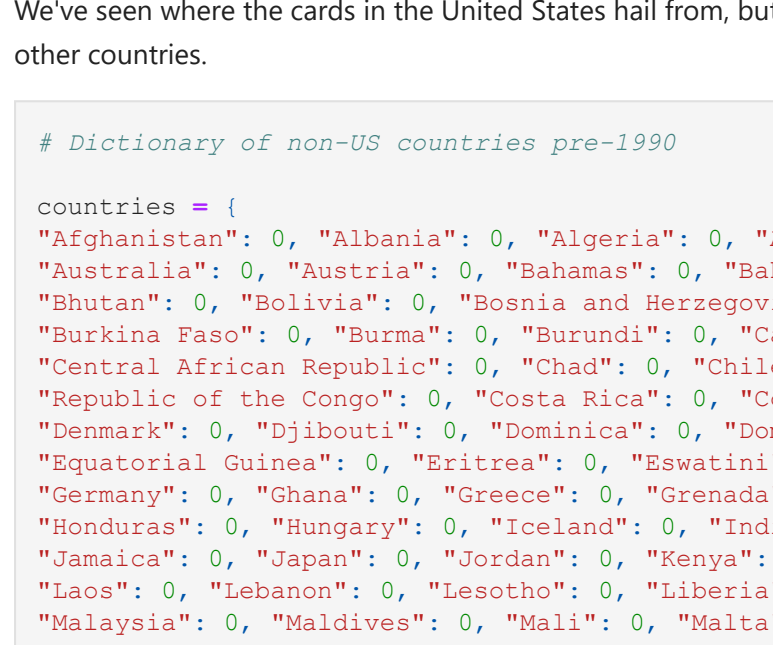
In [12]:
```
# Country counts after running above code
countries = {'Australia': 1, 'Austria': 3, 'Belgium': 1, 'Brazil': 8, 'Canada': 1, 'Chile': 1, 'China': 5, 'Col
```

In [4]:
```
europe = {'Austria': 3, 'Belgium': 1, 'Denmark': 4, 'England': 63, 'France': 47, 'Germany': 30, 'Greece': 3, 'I
    'Italy': 3, 'Malta': 2, 'Poland': 4, 'San Marino': 1,'Spain': 22, 'Sweden': 2, 'Switzerland': 5, 'Wales': 2

asia = {'China': 5, 'India': 25, 'Indonesia': 1, 'Iran': 1, 'Israel': 3, 'Japan': 9, 'Jordan': 13, 'Lebanon': 2
        'Philippines': 3, 'Syria': 1, 'Thailand': 1, 'Turkey': 3, 'Vietnam': 3}

north_america = {'Canada': 1, 'Cuba': 4, 'Grenada': 2, 'Guatemala': 1, 'Jamaica': 4, 'Mexico': 8}

south_america = {'Brazil': 8, 'Chile': 1, 'Colombia': 1, 'Peru': 5, 'Suriname': 1}

africa = {'Egypt': 3, 'Liberia': 1, 'Madagascar': 1, 'Morocco': 1, 'South Africa': 5, 'Tunisia': 1}

oceania = {'Australia': 1,'Fiji': 1,'New Zealand': 1}
```

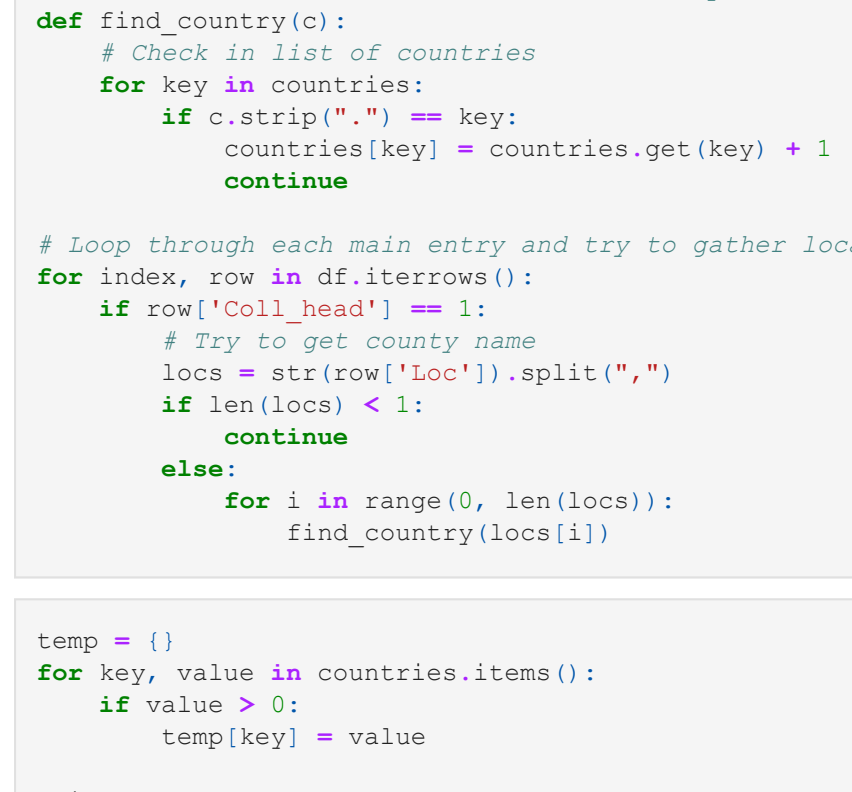
Looks like we have cards from several different countries, mainly from Europe, but we also have a few from South America, Africa, Asia, and Oceania. We have a total of **318 cards from non-USA countries**. Let's visualize how they are distributed.

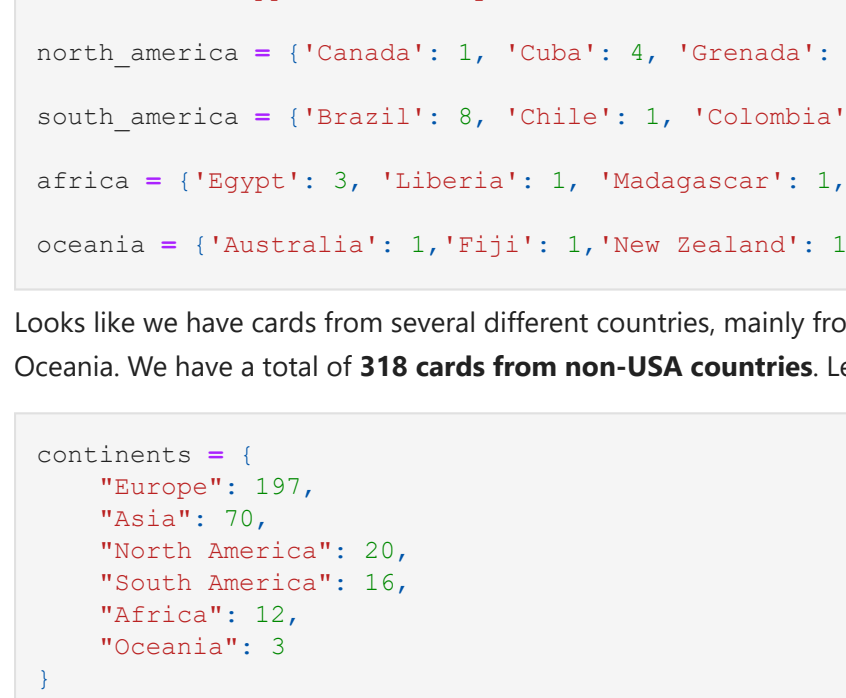
In [23]:
```
continents = {
    "Europe": 197,
    "Asia": 70,
    "North America": 20,
    "South America": 16,
    "Africa": 12,
    "Oceania": 3
}
```

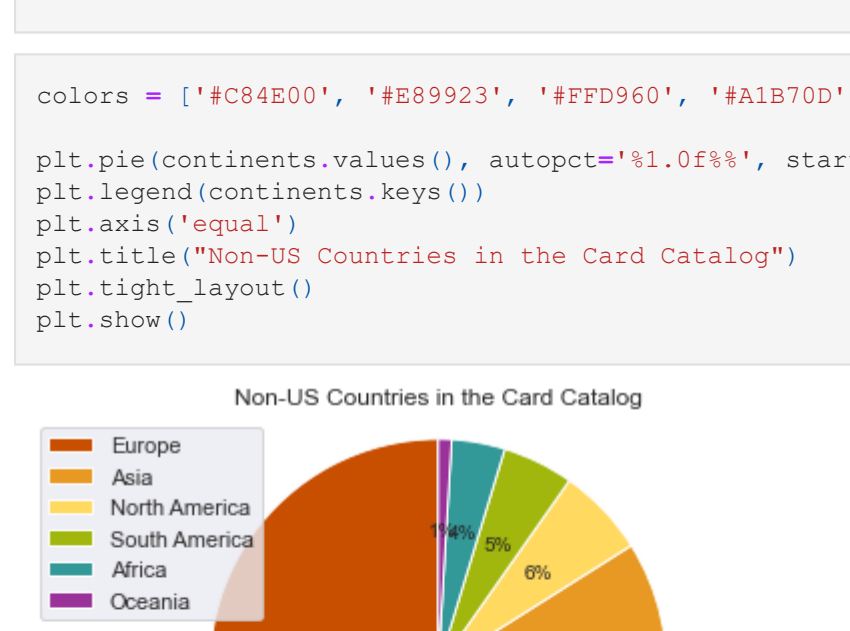
In [24]:
```
colors = ['#C84E00', '#E89923', '#FFD960', '#A1B70D', '#339898', '#993399']

plt.pie(continents.values(), autopct='%1.0f%%', startangle = 90, colors=colors)
plt.legend(continents.keys())
plt.axis('equal')
plt.title("Non-US Countries in the Card Catalog")
plt.tight_layout()
plt.show()
```

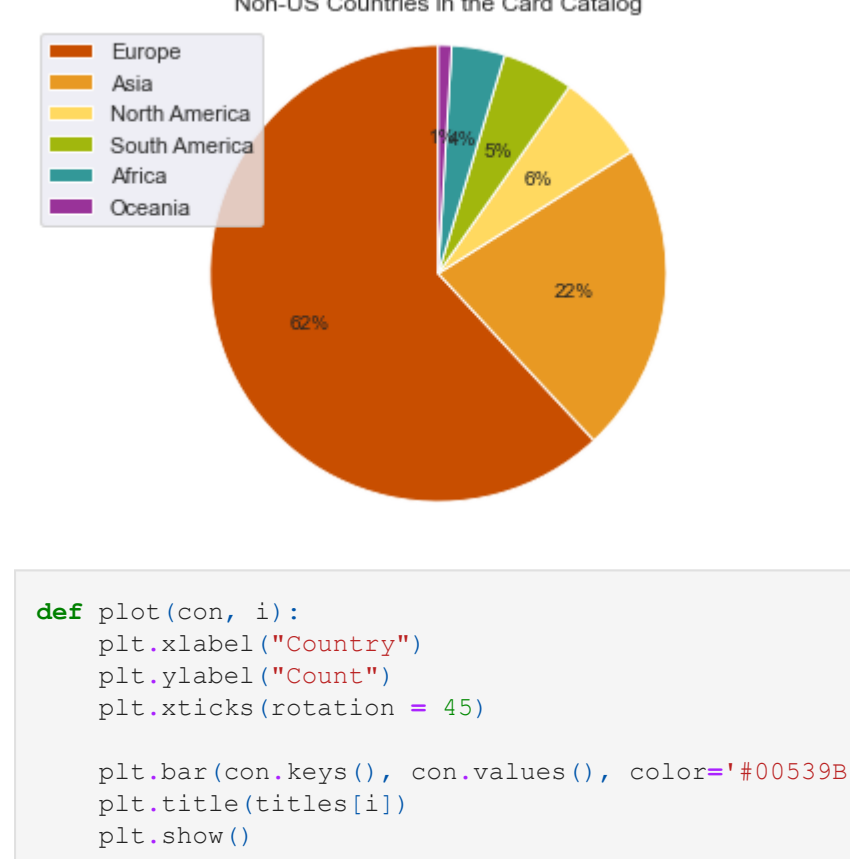


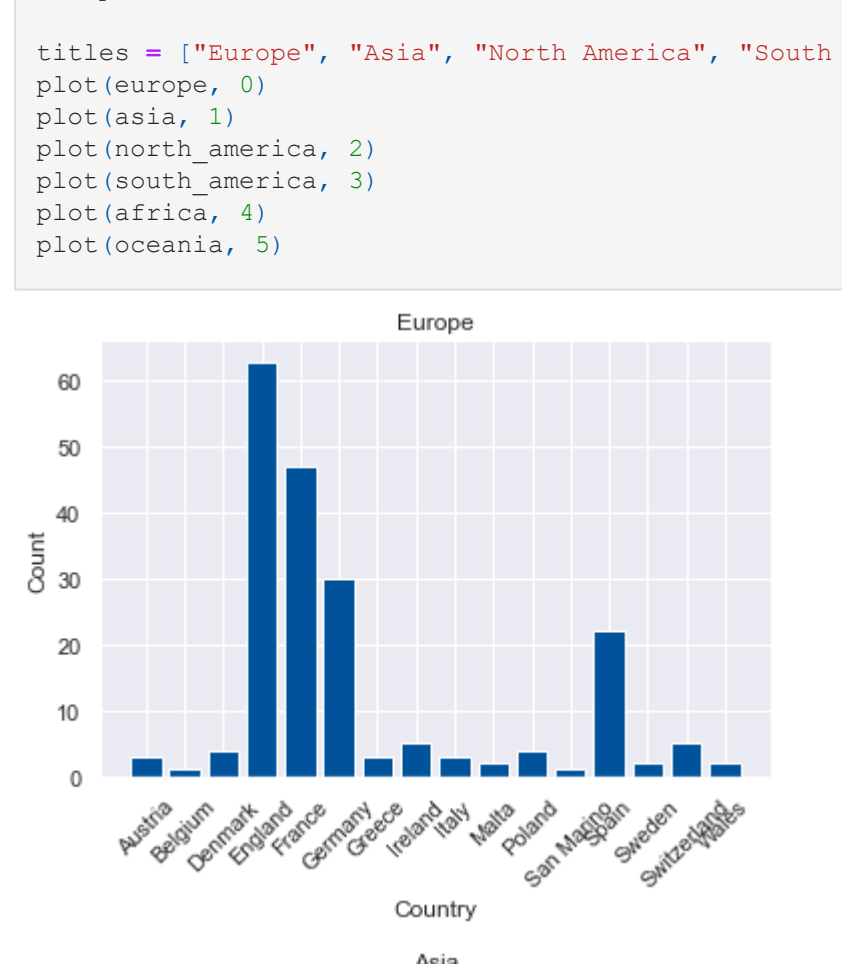
In [25]:
```
def plot(con, i):
    plt.xlabel("Country")
    plt.ylabel("Count")
    plt.xticks(rotation = 45)

    plt.bar(con.keys(), con.values(), color='#00539B')
    plt.title(titles[i])
    plt.show()

titles = ["Europe", "Asia", "North America", "South America", "Africa", "Oceania"]
plot(europe, 0)
plot(asia, 1)
plot(north_america, 2)
plot(south_america, 3)
plot(africa, 4)
plot(oceania, 5)
```











The above visualizations compare the quantities of international card collection between and within continents. We have cards hailing from every continent, save Antarctica!

Adding Country and Continent Columns to Dataset

In [ ]:
```
# Add country name column to dataset
df = pd.read_csv('C:/Users/heidi/Downloads/main_entry_dataset-csv.csv')
countries_in_order = []

# Function to check if location is a country and add to list
def find_country(c):
    # Check in list of countries
    for key in countries:
        if c.strip(".") == key:
            countries_in_order.append(c)
            return

# Loop through each main entry and try to gather location
for index, row in df.iterrows():
    if row['Coll_head'] == 1:
        # Try to get county name
        if pd.isna(row['Loc']):
            countries_in_order.append('NaN')
        else:
            locs = str(row['Loc']).split(",")
            if len(locs) > 0:
                for i in range(0, len(locs)):
                    prev_len = len(countries_in_order)
                    find_country(locs[i])
                    if len(countries_in_order) != prev_len:
                        break
            if len(countries_in_order) == index:
                countries_in_order.append("USA")
    else:
        countries_in_order.append('NaN')
    if len(countries_in_order) == index:
        countries_in_order.append('NaN')
```

In [ ]:
```
df['Country'] = countries_in_order
```

In [6]:
```
# Add continent column to dataset
df = pd.read_csv('C:/Users/heidi/Desktop/rl/Data--Rubenstein-Library-Card-Catalog/main_file_dataset.csv')
continents_in_order = []

for index, row in df.iterrows():
    if row['Coll_head'] == 1:
        if pd.isna(row['Country']):
            continents_in_order.append('NaN')
        else:
            c = row['Country']
            if c == 'USA' or c in north_america.keys():
                continents_in_order.append("North America")
            elif c in europe:
                continents_in_order.append("Europe")
            elif c in asia:
                continents_in_order.append("Asia")
            elif c in south_america:
                continents_in_order.append("South America")
            elif c in africa:
                continents_in_order.append("Africa")
            elif c in oceania:
                continents_in_order.append("Oceania")
            else:
                continents_in_order.append("Unknown")
    else:
        continents_in_order.append('NaN')
```

In [7]:
```
df['Continent'] = continents_in_order
```

In [9]:
```
df.to_csv('main_file_dataset.csv', index=False)
```