In [1]:
```
#Library and data imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import geopandas as gp
import shapely
import shapefile
import plotly.figure_factory as ff
election_data = pd.read_csv('election_train.csv')
demographics_data = pd.read_csv('demographics_train.csv')
```

In [2]:
```
#Task 1: Reshaping election_train into wide format

# Reshaping the data by pivoting along the County and unstacking the Party column of the original data set
reshaped_data = election_data.pivot_table(index=['County', 'State', 'Year', 'Office'], values='Votes',columns
='Party' )

reshaped_data.reset_index(inplace=True) # we must reshape the index to get rid of multi-level pivot table

reshaped_data.columns.name = None # here we get rid of the index column name

print(reshaped_data)
```

```
             County State  Year      Office  Democratic  Republican
0      Adams County    IN  2018  US Senator      3146.0      7511.0
1      Adams County    ND  2018  US Senator       364.0       796.0
2      Adams County    NE  2018  US Senator      3334.0      6487.0
3      Adams County    OH  2018  US Senator      2635.0      6000.0
4      Adams County    PA  2018  US Senator     14880.0     23419.0
...             ...   ...   ...         ...         ...         ...
1200    York County    ME  2018  US Senator     51387.0     32849.0
1201    York County    NE  2018  US Senator      1281.0      3659.0
1202    York County    PA  2018  US Senator     69272.0     95814.0
1203   Young County    TX  2018  US Senator       821.0      5543.0
1204  Zapata County    TX  2018  US Senator      1392.0       821.0

[1205 rows x 6 columns]
```

In [3]:
```
#Task 2: Merging reshaped election data with demographics data

# dictionary to convert state names to abbreviations, sourced from github.com/rogerallen
state_dict = {
    'Alabama': 'AL',
    'Alaska': 'AK',
    'American Samoa': 'AS',
    'Arizona': 'AZ',
    'Arkansas': 'AR',
    'California': 'CA',
    'Colorado': 'CO',
    'Connecticut': 'CT',
    'Delaware': 'DE',
    'District of Columbia': 'DC',
    'Florida': 'FL',
    'Georgia': 'GA',
    'Guam': 'GU',
    'Hawaii': 'HI',
    'Idaho': 'ID',
    'Illinois': 'IL',
    'Indiana': 'IN',
    'Iowa': 'IA',
    'Kansas': 'KS',
    'Kentucky': 'KY',
    'Louisiana': 'LA',
    'Maine': 'ME',
    'Maryland': 'MD',
    'Massachusetts': 'MA',
    'Michigan': 'MI',
    'Minnesota': 'MN',
    'Mississippi': 'MS',
    'Missouri': 'MO',
    'Montana': 'MT',
    'Nebraska': 'NE',
    'Nevada': 'NV',
    'New Hampshire': 'NH',
    'New Jersey': 'NJ',
    'New Mexico': 'NM',
    'New York': 'NY',
    'North Carolina': 'NC',
    'North Dakota': 'ND',
    'Northern Mariana Islands':'MP',
```

```
    'Ohio': 'OH',
    'Oklahoma': 'OK',
    'Oregon': 'OR',
    'Pennsylvania': 'PA',
    'Puerto Rico': 'PR',
    'Rhode Island': 'RI',
    'South Carolina': 'SC',
    'South Dakota': 'SD',
    'Tennessee': 'TN',
    'Texas': 'TX',
    'Utah': 'UT',
    'Vermont': 'VT',
    'Virgin Islands': 'VI',
    'Virginia': 'VA',
    'Washington': 'WA',
    'West Virginia': 'WV',
    'Wisconsin': 'WI',
    'Wyoming': 'WY'
}

"""
Fixing up the election data

"""

# handling inconsistencies in the names of counties
reshaped_data['County'] = reshaped_data['County'].str.replace('County', '')
reshaped_data['County'] = reshaped_data['County'].str.strip();

# removing rows where there is null republican or democratic data
reshaped_data = reshaped_data.dropna()

"""
Fixing up the demographics data

"""

# replacing state names with abbreivations
demographics_data = demographics_data.replace({"State": state_dict})
demographics_data = demographics_data.sort_values(['County', 'State'])

# resetting the indexs after sorting to match election data
demographics_data = demographics_data.reset_index(drop=True)
```

```
# doing a left join on the data results in a dataset having rows where the demographics data matches one of o
ur 1205 rows
merged_data = pd.merge(reshaped_data, demographics_data, on=['State', 'County'], how='left')

print(merged_data.iloc[:, 0:9]) #not printing all columns for cleaner output
```

```
      County State  Year      Office  Democratic  Republican     FIPS  \
0      Adams    IN  2018  US Senator      3146.0      7511.0  18001.0
1      Adams    ND  2018  US Senator       364.0       796.0  38001.0
2      Adams    NE  2018  US Senator      3334.0      6487.0  31001.0
3      Adams    OH  2018  US Senator      2635.0      6000.0  39001.0
4      Adams    PA  2018  US Senator     14880.0     23419.0  42001.0
...      ...   ...   ...         ...         ...         ...      ...
1195    York    ME  2018  US Senator     51387.0     32849.0  23031.0
1196    York    NE  2018  US Senator      1281.0      3659.0  31185.0
1197    York    PA  2018  US Senator     69272.0     95814.0  42133.0
1198   Young    TX  2018  US Senator       821.0      5543.0  48503.0
1199  Zapata    TX  2018  US Senator      1392.0       821.0  48505.0

      Total Population  Citizen Voting-Age Population
0              34813.0                            0.0
1               2348.0                            0.0
2              31536.0                            0.0
3              28111.0                            0.0
4             101759.0                        78370.0
...                ...                            ...
1195          200536.0                            0.0
1196           13842.0                        10570.0
1197          440604.0                       334780.0
1198           18275.0                            0.0
1199           14335.0                            0.0

[1200 rows x 9 columns]
```

In [4]:
```
#Task 3: Exploring the merged data

# printing out the shape
print("Size of dataset: ", merged_data.shape)
print("*****")

# displaying the info for the columns
merged_data.info()
print("*****")

# inquiring about redundant and irrelevant variables
print("Unique years: ", merged_data['Year'].unique())
print("*****")
print("Unique offices: ", merged_data['Office'].unique())
print("*****")


# we decide to drop these columns because they are all the same value
merged_data = merged_data.drop(columns=['Year', 'Office'])

print(merged_data.iloc[:, 0:6]) #not printing all columns for cleaner output
```

```
Size of dataset:  (1200, 21)
*****
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1200 entries, 0 to 1199
Data columns (total 21 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   County                                1200 non-null   object 
 1   State                                 1200 non-null   object 
 2   Year                                  1200 non-null   int64  
 3   Office                                1200 non-null   object 
 4   Democratic                            1200 non-null   float64
 5   Republican                            1200 non-null   float64
 6   FIPS                                  1188 non-null   float64
 7   Total Population                      1188 non-null   float64
 8   Citizen Voting-Age Population         1188 non-null   float64
 9   Percent White, not Hispanic or Latino 1188 non-null   float64
 10  Percent Black, not Hispanic or Latino 1188 non-null   float64
 11  Percent Hispanic or Latino            1188 non-null   float64
 12  Percent Foreign Born                  1188 non-null   float64
 13  Percent Female                        1188 non-null   float64
 14  Percent Age 29 and Under              1188 non-null   float64
 15  Percent Age 65 and Older              1188 non-null   float64
 16  Median Household Income               1188 non-null   float64
 17  Percent Unemployed                    1188 non-null   float64
 18  Percent Less than High School Degree  1188 non-null   float64
 19  Percent Less than Bachelor's Degree   1188 non-null   float64
 20  Percent Rural                         1188 non-null   float64
dtypes: float64(17), int64(1), object(3)
memory usage: 206.2+ KB
*****
Unique years:  [2018]
*****
Unique offices:  ['US Senator']
*****
      County State  Democratic  Republican     FIPS  Total Population
0      Adams    IN      3146.0      7511.0  18001.0           34813.0
1      Adams    ND       364.0       796.0  38001.0            2348.0
2      Adams    NE      3334.0      6487.0  31001.0           31536.0
3      Adams    OH      2635.0      6000.0  39001.0           28111.0
4      Adams    PA     14880.0     23419.0  42001.0          101759.0
...      ...   ...         ...         ...      ...               ...
1195    York    ME     51387.0     32849.0  23031.0          200536.0
```

```
1196    York    NE    1281.0     3659.0  31185.0        13842.0
1197    York    PA   69272.0    95814.0  42133.0       440604.0
1198   Young    TX     821.0     5543.0  48503.0        18275.0
1199  Zapata    TX    1392.0      821.0  48505.0        14335.0

[1200 rows x 6 columns]
```

Task 3 Response

The merged dataset has 21 variables, as shown at the top of the output. The County, State, and Office columns are strings, the year is a int64, and all other columns are float64.

It is hard to tell if any of the demographic data is irrelevent at this point in the project, since all of it can be used to make meaningful comparisions based on income, population, age distribution, etc. The join used for Task 2 eliminated the duplicate County and State columns, and the only two irrelevent/redudant columns are the Year and Office ones. After analyzing their values they both only have one unique value (2018 for year, US Senator for office) and we dealt with them by dropping them to reduce the number of columns. This means our dataset now has 19 columns.

In [5]:
```
#Task 4: searching for missing values

print("Columns containing missing values and their counts:\n")
for c in merged_data:
    if merged_data[c].isnull().any():
        print('\t{0} has {1} null values'.format(c, merged_data[c].isnull().sum()))


# drop duplicates in the data
merged_data = merged_data.drop_duplicates()
print("\nSize of dataset after dropping duplicates: ", merged_data.shape)

# drop rows containing missing data
merged_data = merged_data.dropna()
print("\nSize of dataset after dropping rows with missing values: ", merged_data.shape)
```

```
Columns containing missing values and their counts:

	FIPS has 12 null values
	Total Population has 12 null values
	Citizen Voting-Age Population has 12 null values
	Percent White, not Hispanic or Latino has 12 null values
	Percent Black, not Hispanic or Latino has 12 null values
	Percent Hispanic or Latino has 12 null values
	Percent Foreign Born has 12 null values
	Percent Female has 12 null values
	Percent Age 29 and Under has 12 null values
	Percent Age 65 and Older has 12 null values
	Median Household Income has 12 null values
	Percent Unemployed has 12 null values
	Percent Less than High School Degree has 12 null values
	Percent Less than Bachelor's Degree has 12 null values
	Percent Rural has 12 null values

Size of dataset after dropping duplicates:  (1200, 19)

Size of dataset after dropping rows with missing values:  (1188, 19)
```

Task 4 Response

As shown above, the merged data is missing demographic information for 12 counties. Since the merged set has 1200 entries, we can afford to drop the 12 rows containing missing values and still be able to accurately interpret the data.

There were no duplicate values.

In [6]:
```
#Task 5: assigning a party to each county based on majority vote
import numpy as np

# value of 'Party' is 1 if  # Democratic votes > # Republican votes, 0 otherwise
merged_data['Party'] = np.where(merged_data['Democratic'] > merged_data['Republican'] , 1, 0)

print(merged_data.loc[:, ['Democratic', 'Republican', 'Party']])
```

```
      Democratic  Republican  Party
0         3146.0      7511.0      0
1          364.0       796.0      0
2         3334.0      6487.0      0
3         2635.0      6000.0      0
4        14880.0     23419.0      0
...          ...         ...    ...
1195     51387.0     32849.0      1
1196      1281.0      3659.0      0
1197     69272.0     95814.0      0
1198       821.0      5543.0      0
1199      1392.0       821.0      1

[1188 rows x 3 columns]
```

In [28]:
```
#Task 6:  Computing the mean median household income for Democratic counties and Republican counties. And per
forming hypothesis
#test to determine whether this difference is statistically significant at the α = 0.05 significance level.

import scipy.stats as st
Democrats=pd.DataFrame()
Republican=pd.DataFrame()

Democrats['Median household income'] = np.where(merged_data['Party'] == 1 , merged_data['Median Household Inc
ome'], np.nan)
Republican['Median household income'] = np.where(merged_data['Party'] == 0 , merged_data['Median Household In
come'], np.nan)

Democrats_Income=Democrats.dropna()
print("Mean median household income of Democratic counties is",Democrats_Income['Median household income'].me
an())

Republican_Income=Republican.dropna()
print("Mean median household income of Republican counties is",Republican_Income['Median household income'].m
ean())


[statistic, pvalue]=st.ttest_ind(Democrats_Income['Median household income'],
                                 Republican_Income['Median household income'], equal_var = False)

print("t-test statistic",statistic)
print("pvalue",pvalue)
```

```
Mean median household income of Democratic counties is 53816.12037037037
Mean median household income of Republican counties is 48708.913194444445
t-test statistic 5.521703490870819
pvalue 5.708990935722737e-08
```

Task 6 Response

Hence we can see mean median household income of Democratic counties is greater than Republican.

Hypothesis test:

$\bar{x}1$ = 53816.120 ,that is the mean median household income of Democratic counties, $\bar{x}2$ = 48708.913 ,that is the mean median household income of Republican counties.

H0: μ1 = μ2 , Hα: μ1 ≠ μ2

Now since, t-test statistic = 5.521 and pvalue = 5.708990935722737e-08 So as pvalue < α, we reject H0: Null hypothesis.

In [8]:
```
#Task 7:  Computing the mean Population for Democratic counties and Republican counties. And performing hypot
hesis
#test to determine whether this difference is statistically significant at the α = 0.05 significance level.


Democrats['Population'] = np.where(merged_data['Party'] == 1 , merged_data['Total Population'], np.nan)
Republican['Population'] = np.where(merged_data['Party'] == 0 , merged_data['Total Population'], np.nan)

Democrats_Population=Democrats.dropna(subset=['Population'])
print("Mean Population of Democratic counties is",Democrats_Population['Population'].mean())

Republican_Population=Republican.dropna(subset=['Population'])
print("Mean Population of Republican counties is",Republican_Population['Population'].mean())

[statistic, pvalue]=st.ttest_ind(Democrats_Population['Population'],Republican_Population['Population'], equa
l_var = False)

print("t-test statistic",statistic)
print("p-value",pvalue)
```

```
Mean Population of Democratic counties is 301584.7530864198
Mean Population of Republican counties is 54033.41087962963
t-test statistic 7.9945970576664305
p-value 2.2089383479337377e-14
```

Task 7 Response

Hence we can see mean Population of Democratic counties is greater than Republican.

Hypothesis test:

$\bar{x}1$ = 301584.753 ,that is the mean Population of Democratic counties, $\bar{x}2$ = 54033.410 ,that is the mean Population of Republican counties

H0: $\mu1 = \mu2$, Hα: $\mu1 \neq \mu2$

Now since, t-test statistic = 7.994 and pvalue = 2.2089383479337377e-14 So as pvalue < α, we reject H0: Null hypothesis.

In [18]:
```
#Task 8:  Compare Democratic counties and Republican counties in terms of age, gender, race and ethnicity,
#and education by computing descriptive statistics and creating plots to visualize the results.
DescriptiveAgeCols = ['Percent Age 29 and Under','Percent Age 65 and Older']
DescriptiveRaceCols = ['Percent White, not Hispanic or Latino','Percent Black, not Hispanic or Latino'
                       ,'Percent Hispanic or Latino']
DescriptiveGenCols = ['Percent Female']
DescriptiveEthCols = ['Percent Foreign Born']
DescriptiveEduCols = ['Percent Less than High School Degree','Percent Less than Bachelor\'s Degree']
AgeByParties = merged_data.dropna().groupby('Party')[DescriptiveAgeCols].describe().transpose()
RaceByParties = merged_data.dropna().groupby('Party')[DescriptiveRaceCols].describe().transpose()
GenByParties = merged_data.dropna().groupby('Party')[DescriptiveGenCols].describe().transpose()
EthByParties = merged_data.dropna().groupby('Party')[DescriptiveEthCols].describe().transpose()
EduByParties = merged_data.dropna().groupby('Party')[DescriptiveEduCols].describe().transpose()
print(AgeByParties)
print(RaceByParties)
print(GenByParties)
print(EthByParties)
print(EduByParties)
```

Party		0	1
Percent Age 29 and Under	count	864.000000	324.000000
	mean	35.998412	38.732313
	std	5.173301	6.261712
	min	11.842105	23.156452
	25%	32.974578	34.486626
	50%	35.846532	38.076169
	75%	38.532906	42.175497
	max	58.749116	67.367823
Percent Age 65 and Older	count	864.000000	324.000000
	mean	18.839527	16.196314
	std	4.741228	4.288962
	min	6.954387	6.653188
	25%	15.791656	13.101127
	50%	18.379757	15.672478
	75%	21.124413	18.806606
	max	37.622759	31.642106

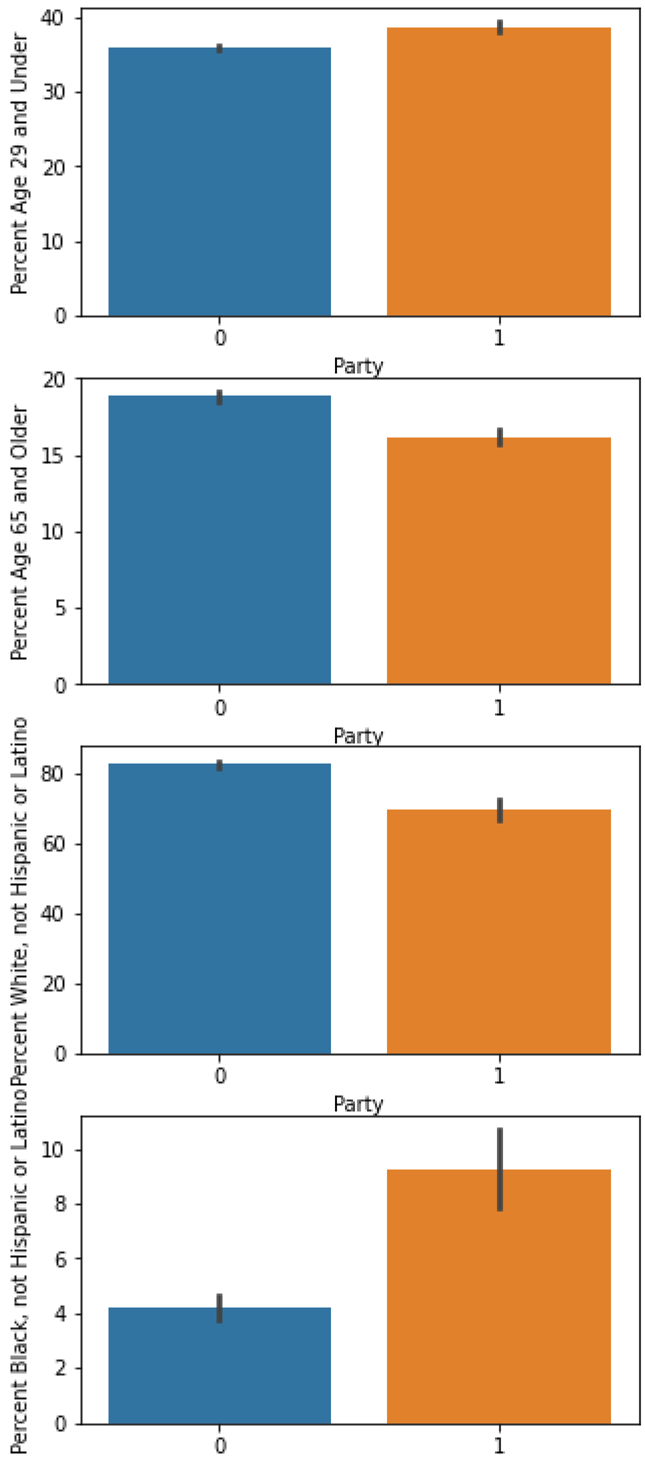
Party		0	1
Percent White, not Hispanic or Latino	count	864.000000	324.000000
	mean	82.648662	69.651454
	std	16.063086	25.013340
	min	18.758977	2.776702
	25%	75.054536	53.118027
	50%	89.388832	77.773724
	75%	94.467740	90.331700
	max	99.627329	98.063495
Percent Black, not Hispanic or Latino	count	864.000000	324.000000
	mean	4.180656	9.237679
	std	6.708644	13.371690
	min	0.000000	0.000000
	25%	0.467673	0.831120
	50%	1.321870	3.478789
	75%	4.747062	11.260282
	max	41.563041	63.953279
Percent Hispanic or Latino	count	864.000000	324.000000
	mean	9.742904	12.607479
	std	14.064943	19.601953
	min	0.000000	0.193349
	25%	1.704293	2.524541
	50%	3.427435	5.034558
	75%	10.772700	11.893419
	max	78.397012	95.479801

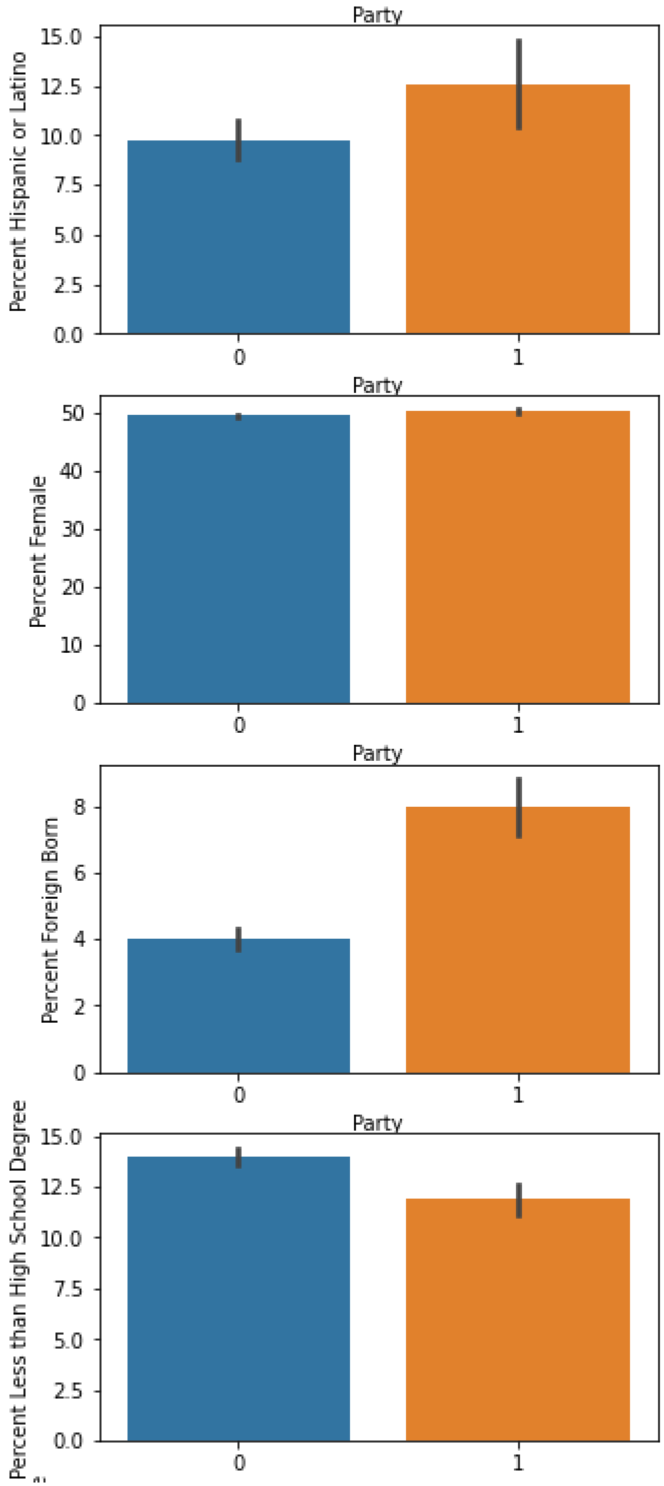
Party	0	1

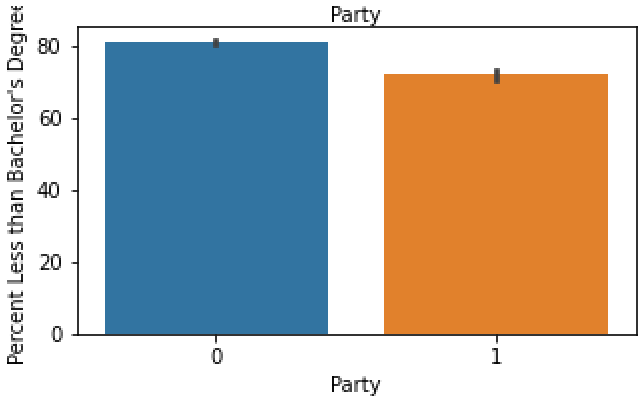
```
Percent Female count  864.000000  324.000000
               mean    49.632268   50.391860
               std      2.434885    2.149553
               min     21.513413   34.245291
               25%     49.228148   49.863006
               50%     50.176792   50.658513
               75%     50.832124   51.492427
               max     55.885023   56.418468
Party                          0           1
Percent Foreign Born count  864.000000  324.000000
                     mean     4.002532    8.001366
                     std      4.520393    8.339208
                     min      0.000000    0.179769
                     25%      1.318889    2.456684
                     50%      2.334546    5.106662
                     75%      5.175071   10.162906
                     max     37.058317   52.229868
Party                                            0           1
Percent Less than High School Degree count  864.000000  324.000000
                                     mean    13.992217   11.881962
                                     std      6.273239    6.515595
                                     min      2.134454    3.215803
                                     25%      9.650973    7.863450
                                     50%     12.572435   10.360797
                                     75%     17.442117   13.654224
                                     max     47.812773   49.673777
Percent Less than Bachelor's Degree  count  864.000000  324.000000
                                     mean    81.085492   71.935989
                                     std      6.825780   11.194596
                                     min     43.419470   26.335440
                                     25%     78.091930   65.703788
                                     50%     82.360830   72.728019
                                     75%     85.554266   79.791402
                                     max     97.014925   94.849957
```

In [27]:
```
#Task 8 Cont.: Plot the data
fig, axes = plt.subplots(9, 1, figsize = (5,30))
sns.barplot(x = 'Party', y = 'Percent Age 29 and Under', data = merged_data, ax = axes[0])
sns.barplot(x = 'Party', y = 'Percent Age 65 and Older', data = merged_data, ax = axes[1])
sns.barplot(x = 'Party', y = 'Percent White, not Hispanic or Latino', data = merged_data, ax = axes[2])
sns.barplot(x = 'Party', y = 'Percent Black, not Hispanic or Latino', data = merged_data, ax = axes[3])
sns.barplot(x = 'Party', y = 'Percent Hispanic or Latino', data = merged_data, ax = axes[4])
sns.barplot(x = 'Party', y = 'Percent Female', data = merged_data, ax = axes[5])
sns.barplot(x = 'Party', y = 'Percent Foreign Born', data = merged_data, ax = axes[6])
sns.barplot(x = 'Party', y = 'Percent Less than High School Degree', data = merged_data, ax = axes[7])
sns.barplot(x = 'Party', y = 'Percent Less than Bachelor\'s Degree', data = merged_data, ax = axes[8])
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x224263eebe0>


40
30
20
10
0
Percent Age 29 and Under
0
1
Party
20
15
10
5
0
Percent Age 65 and Older
0
1
Party
80
60
40
20
0
Percent White, not Hispanic or Latino
0
1
Party
10
8
6
4
2
0
Percent Black, not Hispanic or Latino
0
1


Party
Percent Hispanic or Latino
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
Party
Percent Female
50
40
30
20
10
0
0
1
Party
Percent Foreign Born
8
6
4
2
0
0
1
Party
Percent Less than High School Degree
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1



Task 9 Based on your results for tasks 6-8, which variables in the dataset do you think are more important to determine whether a county is labeled as Democratic or Republican? Justify your answer.

Task 9 Response The most important variables when determining if a county is Democratic or Republican are Population, Median Household Income, Percent white, not hispanic or latino, Percent Less than Bachelor's degree. These are have a much large percentages and a large difference in there mean. While others like, Hispanic or Latino have a big difference in means, they are at a much lower percent level.

In [17]:
```
#Task 10 Create a map of Democratic counties and Republican counties using the
#counties’ FIPS codes and Python’s Plotly library (plot.ly/python/county-choropleth/).
#Note that this dataset does not include all United States counties.

states = list(set(merged_data['State'].tolist()))
values = merged_data['Party'].tolist()
fips = merged_data['FIPS'].tolist()
colorscale = ['rgb(255.0, 0.0, 0.0)', 'rgb(0.0, 0.0, 255.0)']

fig = ff.create_choropleth(
    fips=fips, values=values,colorscale=colorscale,
    scope=states, county_outline={'color': 'rgb(255,255,255)', 'width': 0.5},
    legend_title='Party by County'

)
fig.update_layout(
    legend_x = 0,
    annotations = {'x': -0.12, 'xanchor': 'left'}
)

fig.layout.template = None
fig.show()
```

```
In [ ]:
```