In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv("C:\\Users\\kuppa\\Downloads\\Nomissingfinal_030123.csv")
data
```

Out[2]:

| | OBJECTID_12 | GEOID10 | NAME10 | INTPTLAT10 | INTPTLON10 | FIPS | NAME | STANDARD | FILEID | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1277 | 49025 | Kane | 37.275118 | -111.815413 | 49025 | Kane County | Kane County, UT | SF1US | 49( |
| **1** | 18 | 48271 | Kinney | 29.347086 | -100.417700 | 48271 | Kinney County | Kinney County, TX | SF1US | 48: |
| **2** | 1938 | 8053 | Hinsdale | 37.811625 | -107.383405 | 8053 | Hinsdale County | Hinsdale County, CO | SF1US | 8( |
| **3** | 2853 | 48301 | Loving | 31.844936 | -103.561229 | 48301 | Loving County | Loving County, TX | SF1US | 48: |
| **4** | 1878 | 16059 | Lemhi | 44.928789 | -113.887841 | 16059 | Lemhi County | Lemhi County, ID | SF1US | 16( |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3091** | 409 | 35033 | Mora | 35.982841 | -104.921898 | 35033 | Mora County | Mora County, NM | SF1US | 35( |
| **3092** | 1474 | 8039 | Elbert | 39.310817 | -104.117927 | 8039 | Elbert County | Elbert County, CO | SF1US | 8( |
| **3093** | 2819 | 51007 | Amelia | 37.336131 | -77.973218 | 51007 | Amelia County | Amelia County, VA | SF1US | 51( |
| **3094** | 1399 | 51101 | King William | 37.708260 | -77.091054 | 51101 | King William County | King William County, VA | SF1US | 51 |
| **3095** | 2250 | 51045 | Craig | 37.473129 | -80.231734 | 51045 | Craig County | Craig County, VA | SF1US | 51( |

3096 rows × 242 columns

In [4]:
```
data.shape
```

Out[4]: (3096, 242)

In [6]:
```
data.columns
```

Out[6]:
```
Index(['OBJECTID_12', 'GEOID10', 'NAME10', 'INTPTLAT10', 'INTPTLON10', 'FIPS',
       'NAME', 'STANDARD', 'FILEID', 'UI',
       ...
       'notproficientinEnglishApril2022', 'FemalesApril2022', 'RuralApril2022',
       'PovertyratApril2022', 'DisabilityApril2022', 'Republicannumber',
       'Republicanpercent', 'Democraticnumber', 'DemocraticPercent',
       'Name_12'],
      dtype='object', length=242)
```

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3096 entries, 0 to 3095
Columns: 242 entries, OBJECTID_12 to Name_12
dtypes: float64(127), int64(97), object(18)
memory usage: 5.7+ MB
```

In [10]:
```
data.dtypes
```

Out[10]:
```
OBJECTID_12            int64
GEOID10                int64
NAME10                object
INTPTLAT10           float64
INTPTLON10           float64
                      ...
Republicannumber       int64
Republicanpercent    float64
Democraticnumber       int64
DemocraticPercent    float64
Name_12               object
Length: 242, dtype: object
```

In [11]:
```
target_variable = 'Series_Complete_Pop_Pct_x'
```

In [12]:
```
# Step 1: Demographics Model
demographics = [
    'Census2019_5PlusPop_x', 'Census2019_5to17Pop_x', 'Census2019_12PlusPop_x',
    'Census2019_18PlusPop_x', 'Census2019_65PlusPop_x', 'below18yearsofage2019_x',
    'older65over2019_x', 'below18yearsofage2020', 'older65over2020',
    'below18yearsofageApril2022', 'older65overApril2022', 'Asian2019_x',
    'Asian2020', 'AsianApril2022', 'AmericanIndian_AlaskaNative2019',
    'AmericanIndian_AlaskaNative2020', 'AmericanIndian_AlaskaNativeApri',
    'Females2019_x', 'Females2020', 'FemalesApril2022'
]
```

In [13]:
```
import statsmodels.api as sm
```

In [14]:
```
x_demographics = sm.add_constant(data[demographics])
```

In [15]:
```
model_1 = sm.OLS(data[target_variable], x_demographics).fit()
```

In [16]:
```
print("Demographics_Rsquared:", model_1.rsquared)
```

```
Demographics_Rsquared: 0.26123261930550956
```

In [17]:
```
# Step 2: Demographics + Social Determinants Model
social_determinants = [
    'Highschoolcompletion2019_x', 'Highschoolcompletion2020',
    'HighschoolcompletionApril2022', 'notproficientinEnglish2019_x',
    'notproficientinEnglish2020', 'notproficientinEnglishApril2022',
    'Unemployment2019_x', 'Unemployment2020', 'UnemploymentApril2022',
    'Trafficvolume2019_x', 'Trafficvolume2020', 'TrafficvolumeApril2022',
    'Childreninpoverty2019_x', 'Childreninpoverty2020', 'ChildreninpovertyApril2022',
    'Childreninsingleparenthousehold', 'Childreninsingleparenthouseho_1',
    'Childreninsingleparenthouseho_2', 'Medianhouseholdincome2019_x',
    'Medianhouseholdincome2020', 'MedianhouseholdincomeApril2022'
]
```

In [18]:
```
x_sdoh = sm.add_constant(data[demographics+social_determinants])
```

In [19]:
```
model_2 = sm.OLS(data[target_variable], x_sdoh).fit()
```

In [20]:
```
print("SDOH_Rsquared:", model_2.rsquared)
```

```
SDOH_Rsquared: 0.4620042672867739
```

In [21]:
```
# Step 3: Demographics + Social Determinants + Health Factors Model
health_factors = [
    'Disability2019_x', 'Disability2022', 'DisabilityApril2022',
    'Lifeexpectancy2019_x', 'Lifeexpectancy2020', 'LifeexpectancyApril2022',
    'Prematureageadjustedmortality20', 'Prematureageadjustedmortality_1',
    'PrematureageadjustedmortalityAp'
]
```

In [22]:
```
x_health = sm.add_constant(data[demographics+social_determinants+health_factors])
```

In [24]:
```
model_3 = sm.OLS(data[target_variable], x_health).fit()
```

In [25]:
```
print("Health_Rsquared:", model_3.rsquared)
```

```
Health_Rsquared: 0.47193581944373986
```

In [26]:
```
political_leaning = [
    'Republicanpercent', 'DemocraticPercent'
]
```

In [28]:
```
x_political = sm.add_constant(data[demographics+social_determinants+health_factors+polit
```

In [29]:
```
model_4 = sm.OLS(data[target_variable], x_political).fit()
```

In [30]:
```
print("Political_Rsquared:", model_4.rsquared)
```

```
Political_Rsquared: 0.5056303078712133
```

In [32]:
```
# Print the statistical significance of political leaning factors
print("\nStatistical significance of Political Leaning factors:")
print(model_4.summary().tables[1])
```

```
+05      1.3e+05
AmericanIndian_AlaskaNative2020  3.846e+04   1.22e+05      0.315      0.753    -2.01e
+05     2.78e+05
AmericanIndian_AlaskaNativeApri -1.362e+04   4.31e+04     -0.316      0.752    -9.81e
+04     7.08e+04
Females2019_x                    1.398e+05   6.42e+05      0.218      0.828    -1.12e
+06      1.4e+06
Females2020                     -2.161e+05   9.93e+05     -0.218      0.828    -2.16e
+06     1.73e+06
FemalesApril2022                 7.625e+04    3.5e+05      0.218      0.828    -6.11e
+05     7.63e+05
Highschoolcompletion2019_x       1.248e+05   4.11e+05      0.303      0.762    -6.82e
+05     9.31e+05
Highschoolcompletion2020        -1.928e+05   6.36e+05     -0.303      0.762    -1.44e
+06     1.05e+06
HighschoolcompletionApril2022    6.805e+04   2.24e+05      0.303      0.762    -3.72e
+05     5.08e+05
notproficientinEnglish2019_x     3.359e+07    1.4e+08      0.240      0.811    -2.41e
+08     3.08e+08
```

# EDA

In [33]:
```
data.head()
```

Out[33]:

| | OBJECTID_12 | GEOID10 | NAME10 | INTPTLAT10 | INTPTLON10 | FIPS | NAME | STANDARD | FILEID | UI |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1277 | 49025 | Kane | 37.275118 | -111.815413 | 49025 | Kane County | Kane County, UT | SF1US | 49025 |
| **1** | 18 | 48271 | Kinney | 29.347086 | -100.417700 | 48271 | Kinney County | Kinney County, TX | SF1US | 48271 |
| **2** | 1938 | 8053 | Hinsdale | 37.811625 | -107.383405 | 8053 | Hinsdale County | Hinsdale County, CO | SF1US | 8053 |
| **3** | 2853 | 48301 | Loving | 31.844936 | -103.561229 | 48301 | Loving County | Loving County, TX | SF1US | 48301 |
| **4** | 1878 | 16059 | Lemhi | 44.928789 | -113.887841 | 16059 | Lemhi County | Lemhi County, ID | SF1US | 16059 |

5 rows × 242 columns

In [34]:
```
data.describe()
```

Out[34]:

| | OBJECTID_12 | GEOID10 | INTPTLAT10 | INTPTLON10 | FIPS | UI | FIPS_1 | Cc |
|---|---|---|---|---|---|---|---|---|
| **count** | 3096.000000 | 3096.000000 | 3096.000000 | 3096.000000 | 3096.000000 | 3096.000000 | 3096.000000 | |
| **mean** | 1553.906008 | 30749.049096 | 38.278969 | -91.601765 | 30749.049096 | 30749.049096 | 30744.074289 | |
| **std** | 897.614956 | 14962.975037 | 4.844293 | 11.394920 | 14962.975037 | 14962.975037 | 14959.939717 | |
| **min** | 1.000000 | 1001.000000 | 25.601043 | -124.211406 | 1001.000000 | 1001.000000 | 1001.000000 | |
| **25%** | 775.750000 | 19054.500000 | 34.675292 | -97.956729 | 19054.500000 | 19054.500000 | 19054.500000 | |
| **50%** | 1552.500000 | 29224.000000 | 38.319946 | -90.132265 | 29224.000000 | 29224.000000 | 29222.000000 | |
| **75%** | 2331.250000 | 46015.500000 | 41.709094 | -83.372924 | 46015.500000 | 46015.500000 | 46013.500000 | |
| **max** | 3108.000000 | 56045.000000 | 48.842653 | -67.609354 | 56045.000000 | 56045.000000 | 56045.000000 | |

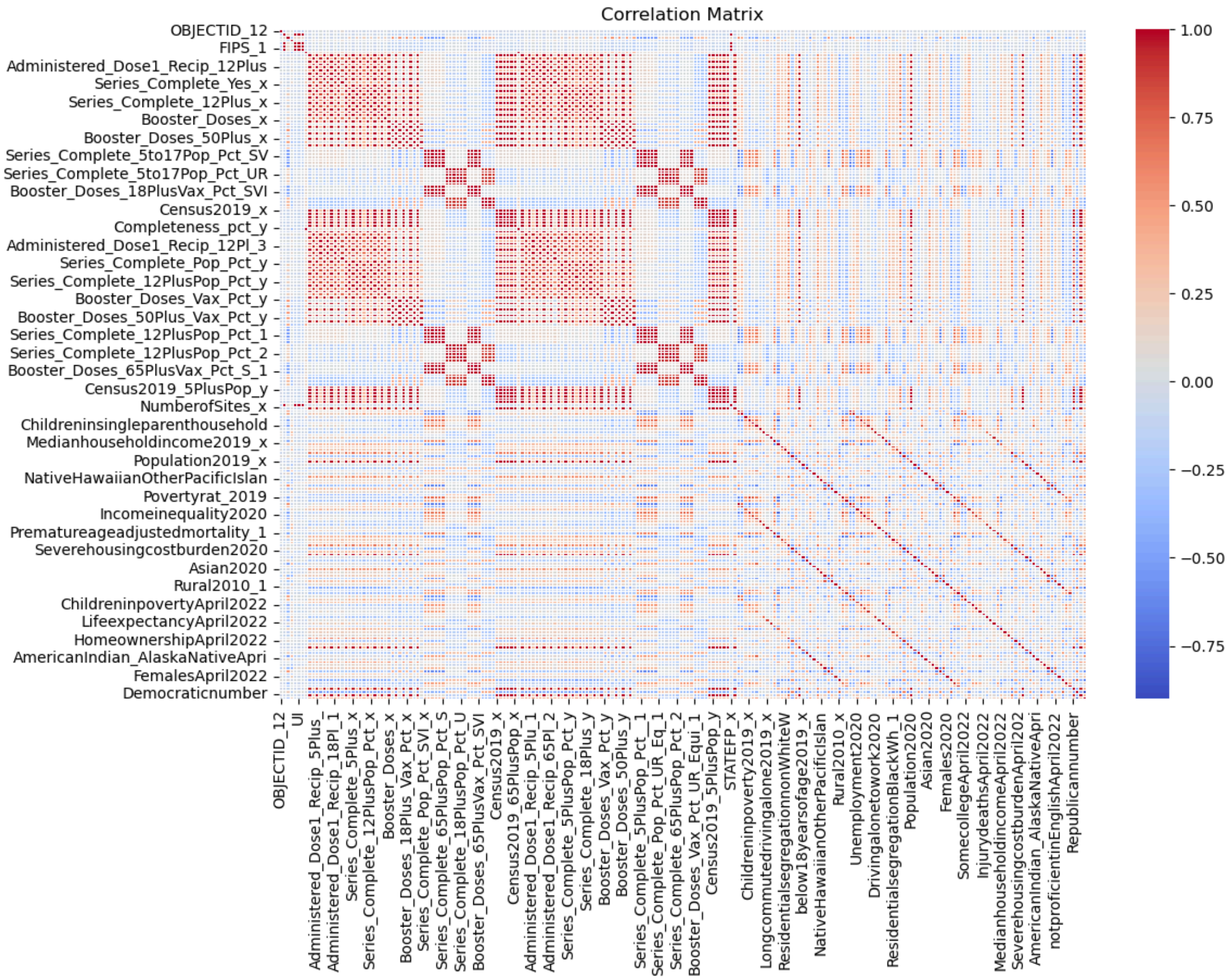
8 rows × 224 columns

In [35]:
```
#check for missing values
data.isnull().sum()
```

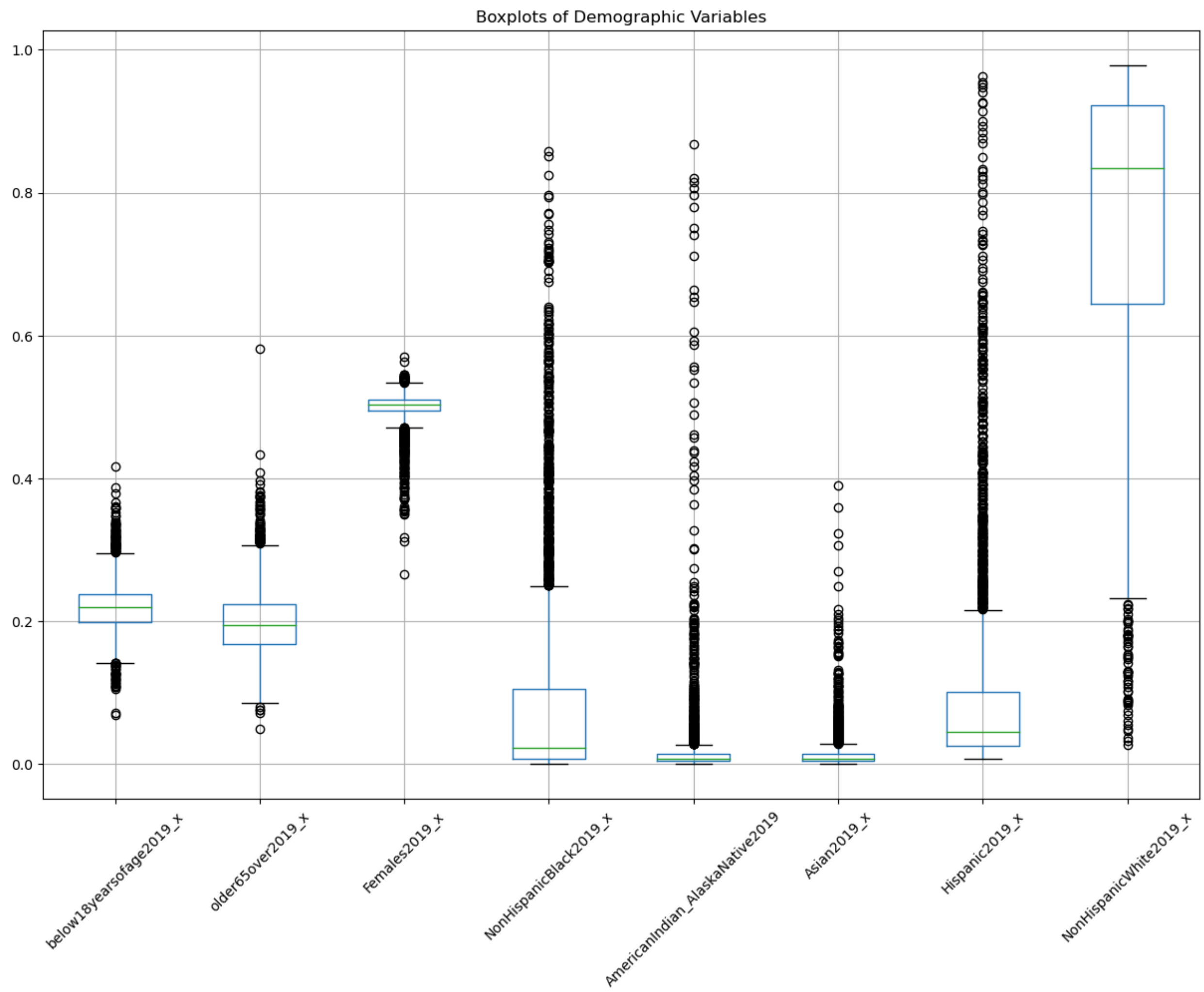
Out[35]:
```
OBJECTID_12          0
GEOID10              0
NAME10               0
INTPTLAT10           0
INTPTLON10           0
                    ..
Republicannumber     0
Republicanpercent    0
Democraticnumber     0
DemocraticPercent    0
Name_12              0
Length: 242, dtype: int64
```

In [36]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [40]:
```
# Select only numeric columns for the correlation matrix
numeric_data = data.select_dtypes(include=['float64', 'int64'])
plt.figure(figsize=(12,8))
corr_matrix = numeric_data.corr()
sns.heatmap(corr_matrix, cmap = "coolwarm", annot=False , linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

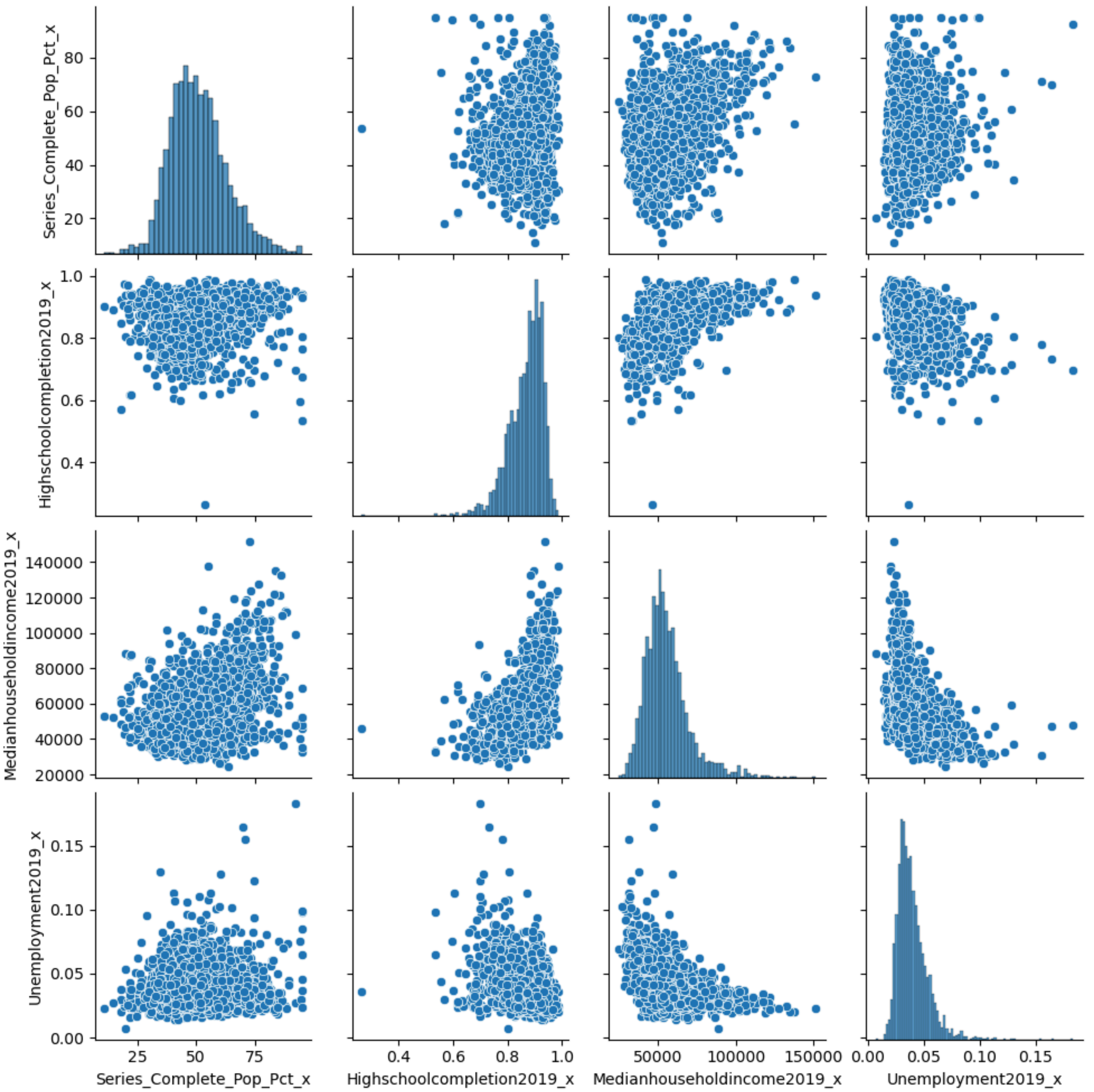


In [41]:
```
# Boxplots for demographic variables
demographic_columns = [
    'below18yearsofage2019_x', 'older65over2019_x', 'Females2019_x',
    'NonHispanicBlack2019_x', 'AmericanIndian_AlaskaNative2019',
    'Asian2019_x', 'Hispanic2019_x', 'NonHispanicWhite2019_x'
]
plt.figure(figsize=(15, 10))
data[demographic_columns].boxplot()
plt.title("Boxplots of Demographic Variables")
plt.xticks(rotation=45)
plt.show()
```



In [42]:
```
# Pairplot for a subset of relevant features
subset_columns = ['Series_Complete_Pop_Pct_x', 'Highschoolcompletion2019_x',
                  'Medianhouseholdincome2019_x', 'Unemployment2019_x']
sns.pairplot(data[subset_columns])
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fi
gure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




# ML

In [43]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [44]:
```
# Define the target and features
target = 'Series_Complete_Pop_Pct_x'
features = [
    'Highschoolcompletion2019_x', 'Medianhouseholdincome2019_x',
    'Unemployment2019_x', 'Disability2019_x', 'Lifeexpectancy2019_x',
    'Republicanpercent', 'DemocraticPercent'
]
```

In [45]:
```
X = data[features]
y = data[target]
```

In [47]:
```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
```

In [48]:
```
#Linear regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)
```

In [49]:
```
# Evaluation for Linear Regression
print("\nLinear Regression Results:")
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_lin))
print("R-squared:", r2_score(y_test, y_pred_lin))
```

```

Linear Regression Results:
Mean Squared Error: 89.69905960509034
R-squared: 0.37013731385508464
```

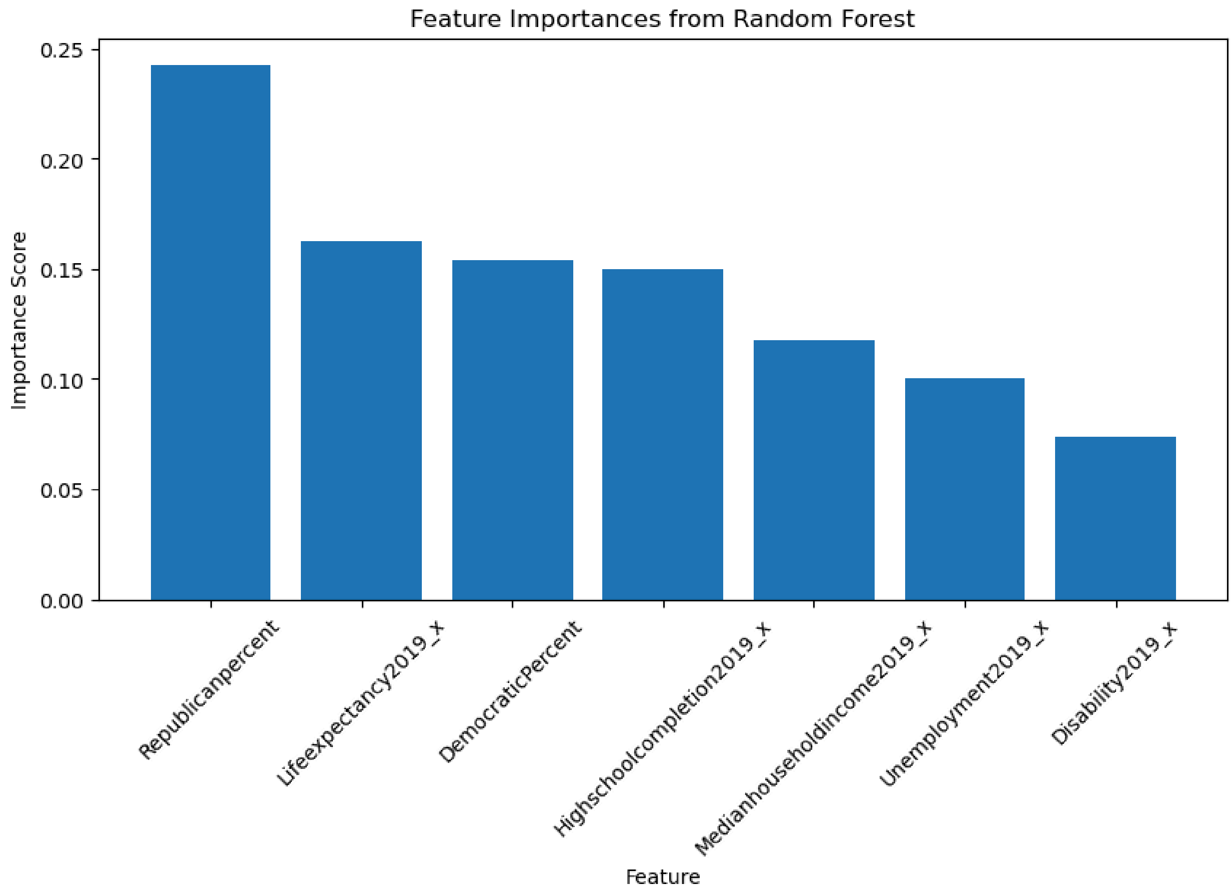
In [50]:
```
# 2. Random Forest Regressor
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)

# Evaluation for Random Forest
print("\nRandom Forest Results:")
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_rf))
print("R-squared:", r2_score(y_test, y_pred_rf))
```

```

Random Forest Results:
Mean Squared Error: 81.13850747039827
R-squared: 0.4302491186630668
```

In [51]:
```
# Feature Importance from Random Forest
plt.figure(figsize=(10, 5))
importances = rf_reg.feature_importances_
indices = sorted(range(len(importances)), key=lambda i: importances[i], reverse=True)
plt.bar([features[i] for i in indices], [importances[i] for i in indices])
plt.title("Feature Importances from Random Forest")
plt.xlabel("Feature")
plt.ylabel("Importance Score")
plt.xticks(rotation=45)
plt.show()
```




In [ ]: