# Cleaning the Data

In [29]:
```
import pandas as pd
```

In [30]:
```
# This gets data for county off of gender, ethnicity and removing nation and state levels

master_df = pd.read_csv("Heart_Disease_Mortality_Data_Among_US_Adults__35___by_State_Territo

# removes the insufficient data columns
rem_null_overall_df = master_df[master_df['Data_Value_Footnote'].isnull()]

# gets only male/female
only_gen_overall_df = rem_null_overall_df[rem_null_overall_df['Stratification1'] != 'Overall']

# removes the overall for the ethnicity
only_eth_overall_df = only_gen_overall_df[only_gen_overall_df['Stratification2'] != 'Overall']

# only gets the county
only_county_overall_df = only_eth_overall_df[only_eth_overall_df['GeographicLevel'] == 'County']

# get the columns we are only using
desired_columns = ['LocationAbbr', 'LocationDesc', 'Data_Value', 'Stratification1', 'Stratification2']
cleaned_county_df = only_county_overall_df[desired_columns]

# Renamed the columns to better naming for the project
cleaned_county_df.columns = ['State', 'County', 'Heart Disease per 100k', 'Gender', 'Ethnicity']

# Validated the column total (I checked against the excel and made sure this was correct)
# print(len(cleaned_county_df))

# Checking the data
cleaned_county_df.head()
```

Out[30]:

| | State | County | Heart Disease per 100k | Gender | Ethnicity |
|---|---|---|---|---|---|
| **102** | AK | Anchorage | 317.5 | Male | White |
| **105** | AK | Denali | 400.7 | Male | White |
| **106** | AK | Fairbanks North Star | 401.0 | Male | White |
| **107** | AK | Haines | 385.5 | Male | White |
| **108** | AK | Juneau | 281.6 | Male | White |

In [31]:
```
# This block is to get the clean county overall data only

rem_null_overall_df = master_df[master_df['Data_Value_Footnote'].isnull()]

# gets overall for gender
only_gen_overall_df = rem_null_overall_df[rem_null_overall_df['Stratification1'] == 'Overall']

# gets overall for ethnicity
only_eth_overall_df = only_gen_overall_df[only_gen_overall_df['Stratification2'] == 'Overall']

# only gets the county
```

```
only_county_overall_df = only_eth_overall_df[only_eth_overall_df['GeographicLevel'] == 'County']

# get the columns we are only using
cleaned_county_overall_df = only_county_overall_df[desired_columns]

# Renamed the columns to better naming for the project
cleaned_county_overall_df.columns = ['State', 'County', 'Heart Disease per 100k', 'Gender', 'Ethnicity']

# Validated the column total (Verified the excel and it's correct)
# print(len(cleaned_county_overall_df))

cleaned_county_overall_df.head()
```

Out[31]:

| | State | County | Heart Disease per 100k | Gender | Ethnicity |
|---|---|---|---|---|---|
| **0** | AK | Aleutians East | 105.3 | Overall | Overall |
| **1** | AK | Aleutians West | 211.9 | Overall | Overall |
| **2** | AK | Anchorage | 257.9 | Overall | Overall |
| **3** | AK | Bethel | 351.6 | Overall | Overall |
| **5** | AK | Denali | 305.5 | Overall | Overall |

In [32]:

```
# This function finds the outliers using the interquartile range method
def find_outliers_iqr(df, column):
    # Extract the data column
    data = df[column]

    # Calculate the quartiles
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)

    # Calculate the interquartile range (IQR)
    IQR = Q3 - Q1

    # Calculate the lower bound and upper bound for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Find outliers based on the bounds
    outliers = (data < lower_bound) | (data > upper_bound)

    # Remove outliers from the DataFrame
    df = df[~outliers]

    return df

# Clean outliers from cleaned_county_df DataFrame
cleaned_county_df = find_outliers_iqr(cleaned_county_df, 'Heart Disease per 100k')

# Clean outliers from cleaned_county_overall_df DataFrame
cleaned_county_overall_df = find_outliers_iqr(cleaned_county_overall_df, 'Heart Disease per 100k')
```

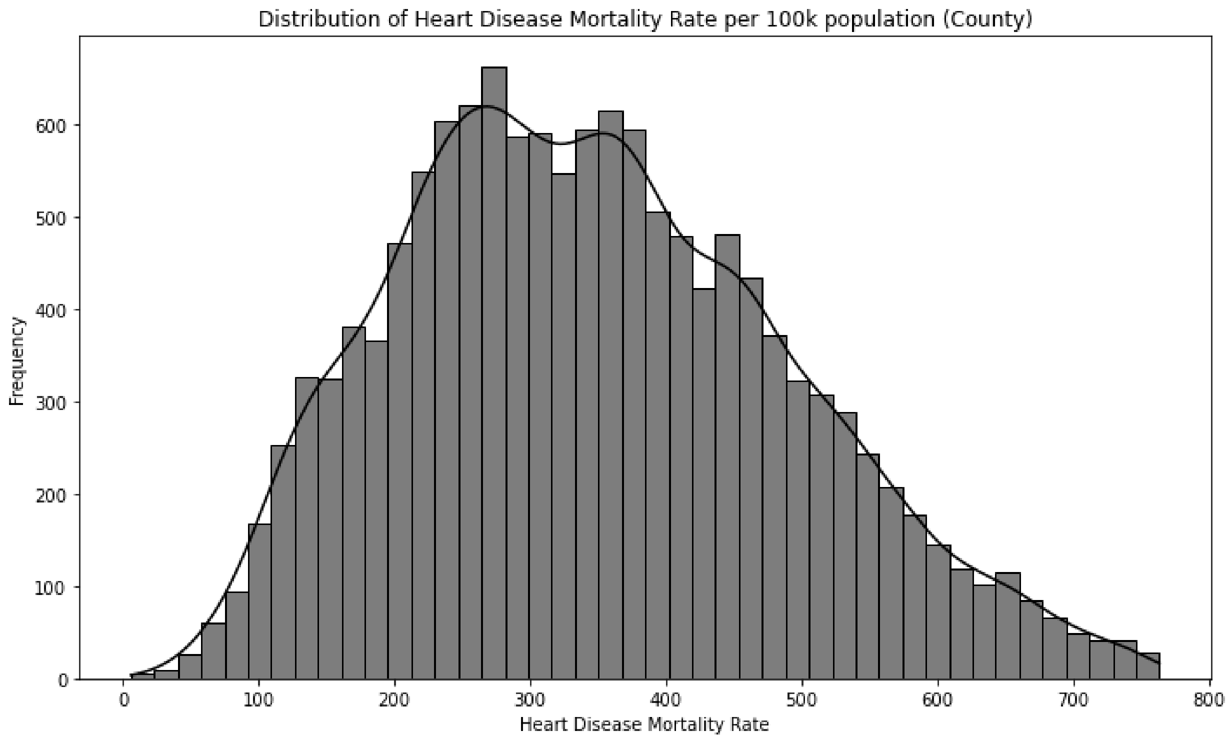
## Exploratory Data Analysis

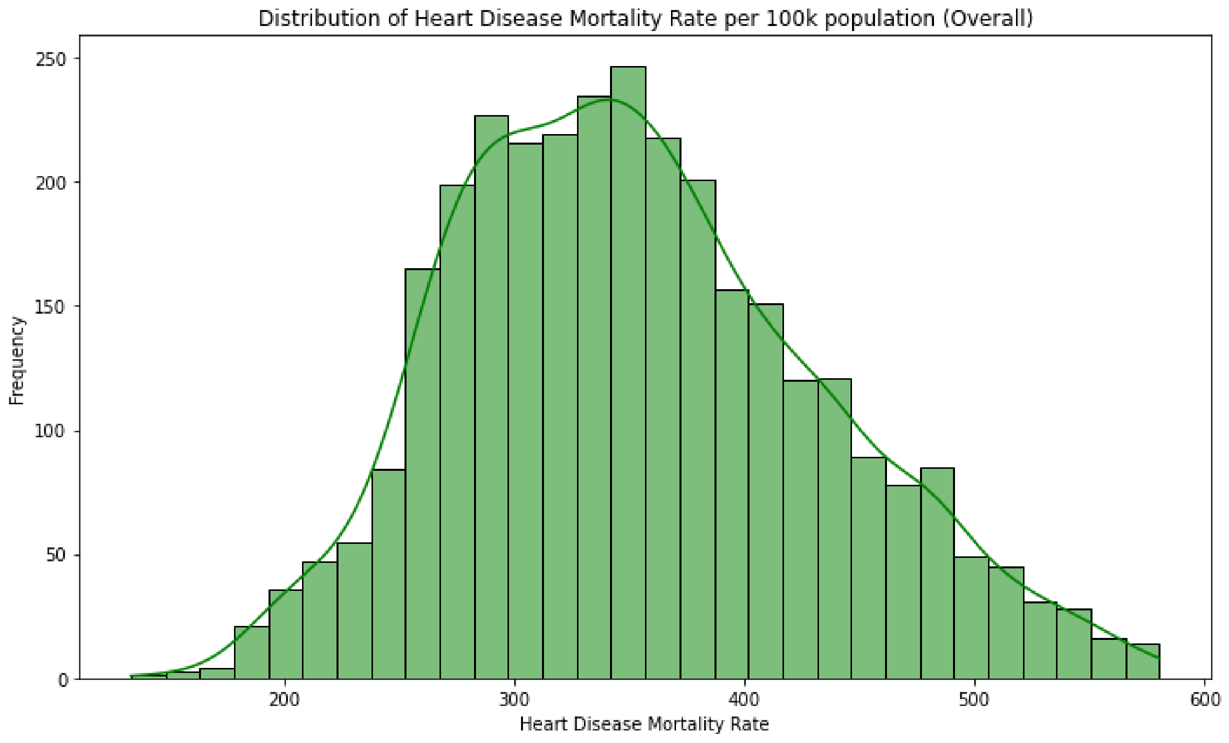
In [33]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from scipy import stats
```

In [34]:
```
# Print 5 Number Summary for cleaned data with no outliers
print("5 Number Summary for cleaned data with no outliers\n", cleaned_county_df.describe())

# Print 5 Number Summary for cleaned data overall with no outliers
print("\n5 Number Summary for cleaned data overall with no outliers\n", cleaned_county_overall_df.de

# Calculate statistics for overall cleaned data
overall_mean = np.mean(cleaned_county_overall_df['Heart Disease per 100k'])
overall_std = np.std(cleaned_county_overall_df['Heart Disease per 100k'], ddof=1)
overall_size = len(cleaned_county_overall_df)

# Calculate statistics for individual cleaned data
indv_mean = np.mean(cleaned_county_df['Heart Disease per 100k'])
indv_size = len(cleaned_county_df)

# Calculate standard error for the sample
se_indv = overall_std / np.sqrt(indv_size)

# Print calculated statistics
print("Population Mean:", overall_mean)
print("Sample Mean:", indv_mean)
print("Standard Error for the Sample:", se_indv)
```

```
5 Number Summary for cleaned data with no outliers
        Heart Disease per 100k
count            13484.000000
mean               347.002648
std                143.989750
min                  6.000000
25%                239.675000
50%                335.900000
75%                445.800000
max                763.500000

5 Number Summary for cleaned data overall with no outliers
        Heart Disease per 100k
count             3162.000000
mean               353.284756
std                 79.606042
min                133.500000
25%                294.025000
50%                345.850000
75%                404.575000
max                580.400000
Population Mean: 353.2847564832387
Sample Mean: 347.0026475823194
Standard Error for the Sample: 0.6855460914644189
```

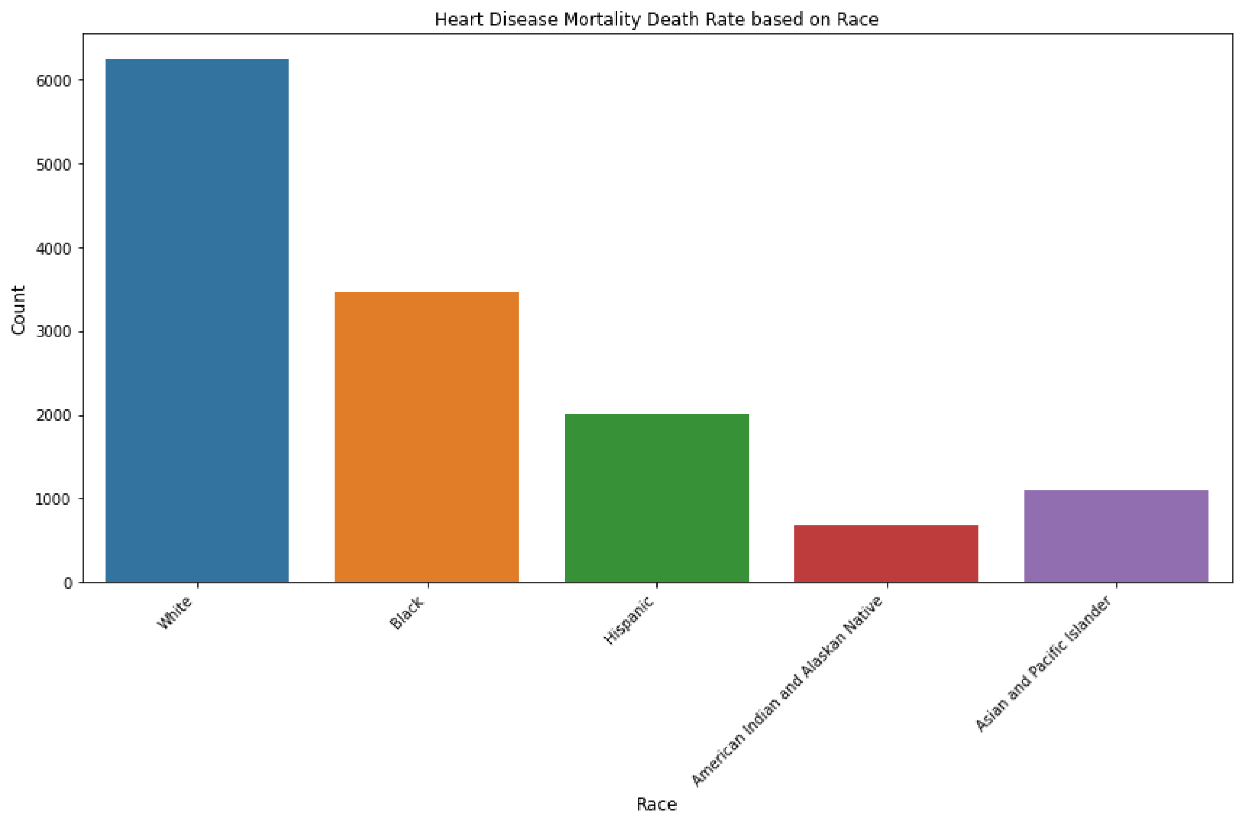
In [35]:
```
# Create a figure and axes with a 2x1 layout
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(10, 12))

# Subplot 1: Histogram for cleaned_county_df
sns.histplot(cleaned_county_df['Heart Disease per 100k'], bins='auto', kde=True, color='black', ax=axes[0
axes[0].set_title('Distribution of Heart Disease Mortality Rate per 100k population (County)')
axes[0].set_xlabel('Heart Disease Mortality Rate')
axes[0].set_ylabel('Frequency')

# Subplot 2: Histogram for cleaned_county_overall_df
sns.histplot(cleaned_county_overall_df['Heart Disease per 100k'], bins='auto', kde=True, color='green', ax
axes[1].set_title('Distribution of Heart Disease Mortality Rate per 100k population (Overall)')
axes[1].set_xlabel('Heart Disease Mortality Rate')
axes[1].set_ylabel('Frequency')

plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()
```

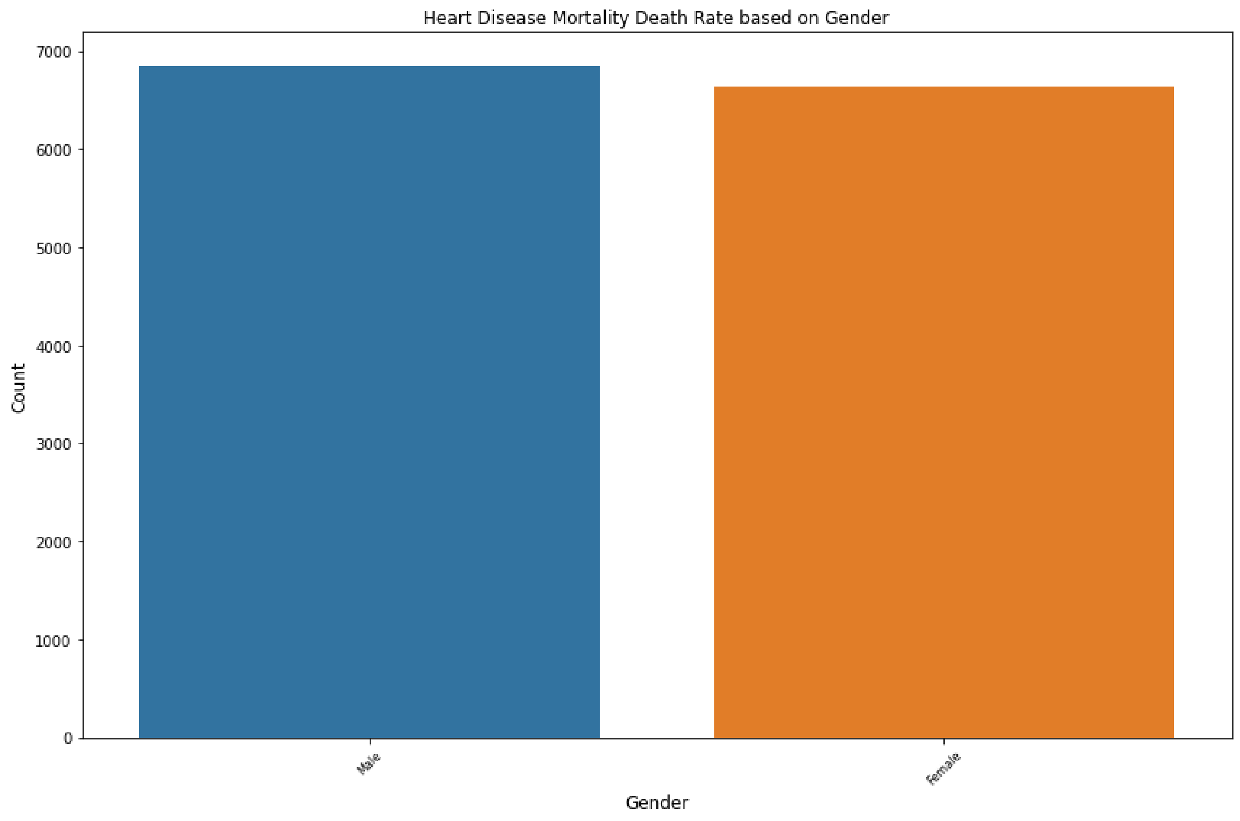


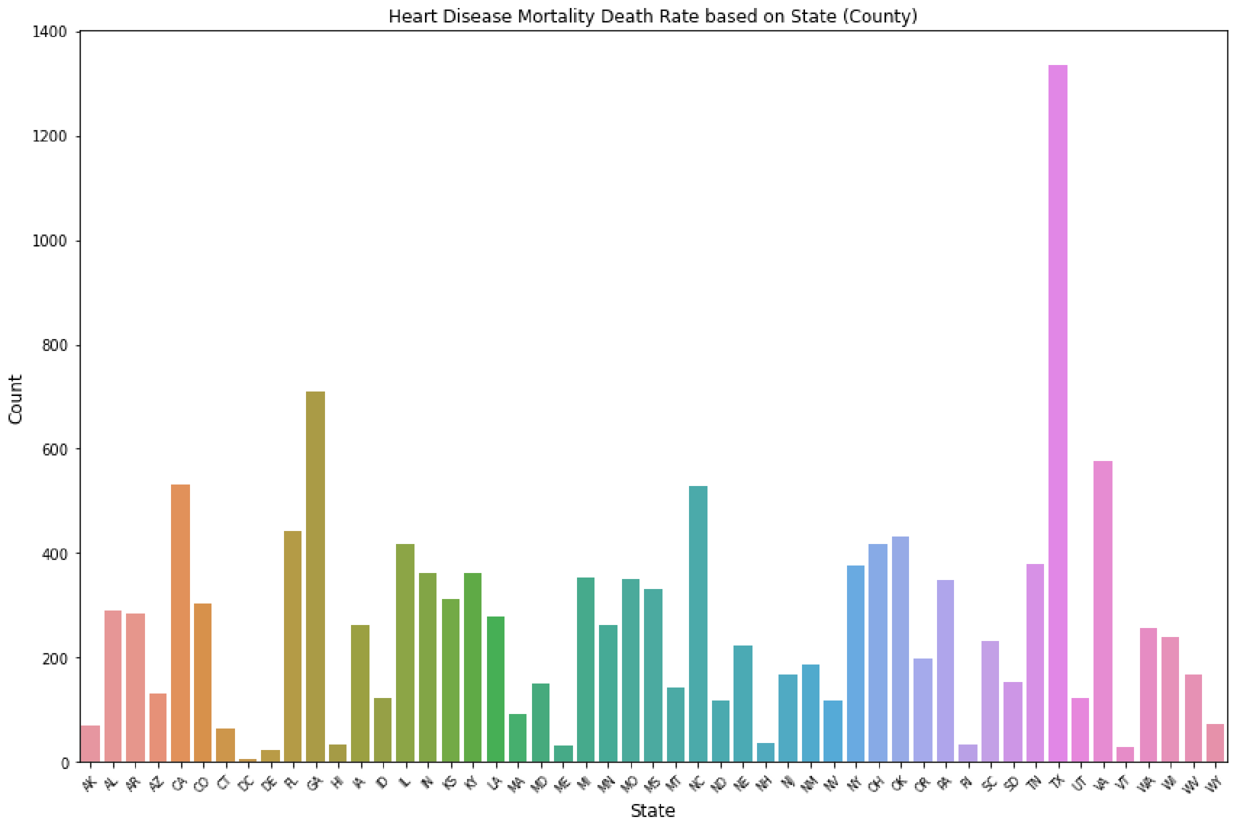


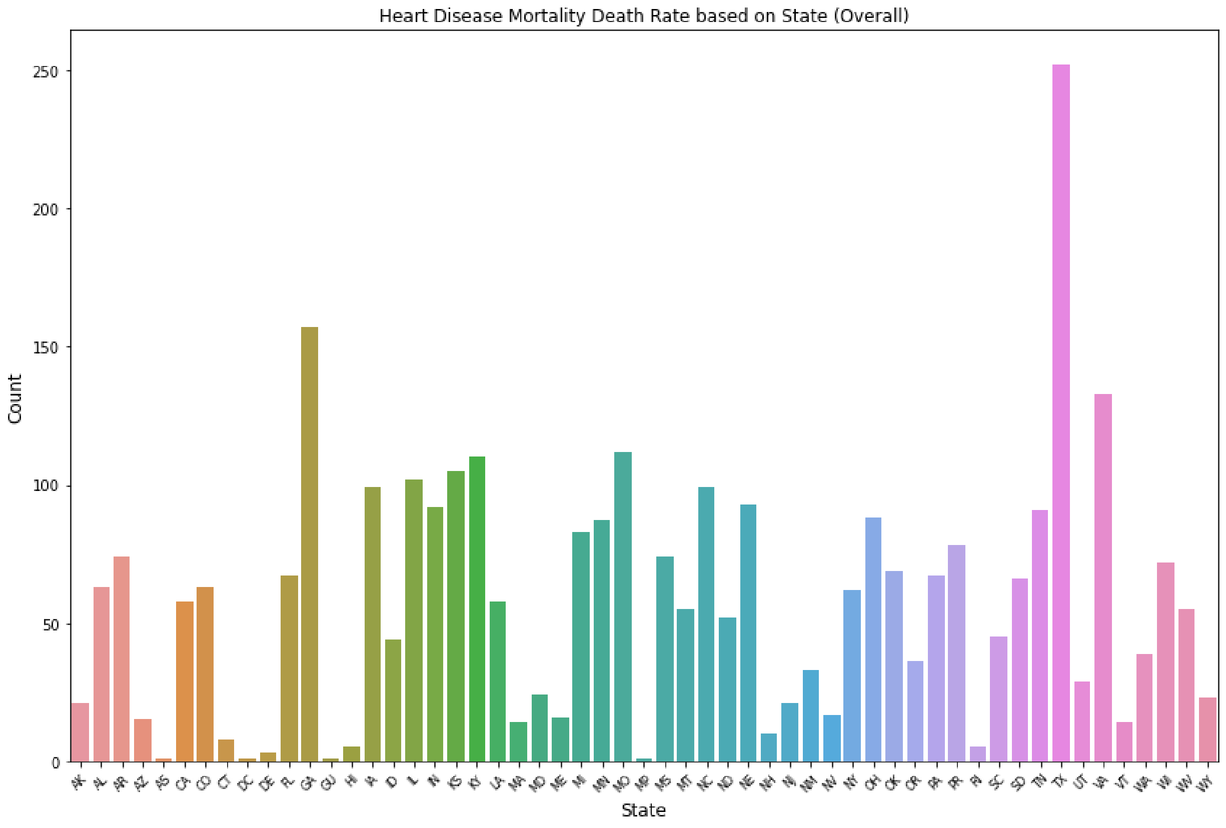
In [36]:
```
# Plot for Ethnicity
plt.figure(figsize=(12, 8))  # Countplot for Ethnicity
sns.countplot(data=cleaned_county_df, x='Ethnicity')
plt.title('Heart Disease Mortality Death Rate based on Race', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)  # Rotate labels by 45 degrees and align them to the rigl
plt.xlabel('Race', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()

# Plot for Gender
plt.figure(figsize=(12, 8))  # Countplot for Gender
sns.countplot(data=cleaned_county_df, x='Gender')
plt.title('Heart Disease Mortality Death Rate based on Gender', fontsize=12)
plt.xticks(rotation=45, fontsize=8)  # Decrease font size
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()

# Plot for States
fig, axs = plt.subplots(nrows=2, ncols=1, figsize=(12, 16))  # Create a figure and axes with a 2x1 layout

# Subplot 1: Countplot for State in cleaned_county_df
sns.countplot(data=cleaned_county_df, x='State', ax=axs[0])
axs[0].set_title('Heart Disease Mortality Death Rate based on State (County)', fontsize=12)
axs[0].tick_params(axis='x', labelrotation=45, labelsize=8)  # Rotate and decrease x-axis tick label size
axs[0].set_xlabel('State', fontsize=12)
axs[0].set_ylabel('Count', fontsize=12)

# Subplot 2: Countplot for State in cleaned_county_overall_df
sns.countplot(data=cleaned_county_overall_df, x='State', ax=axs[1])
axs[1].set_title('Heart Disease Mortality Death Rate based on State (Overall)', fontsize=12)
axs[1].tick_params(axis='x', labelrotation=45, labelsize=8)  # Rotate and decrease x-axis tick label size
axs[1].set_xlabel('State', fontsize=12)
axs[1].set_ylabel('Count', fontsize=12)

plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()

# Plot for top 10 counties
top_counties = cleaned_county_df['County'].value_counts().nlargest(10).index  # Calculate the top 10 cc
top_counties_overall = cleaned_county_overall_df['County'].value_counts().nlargest(10).index

top_county_data = cleaned_county_df[cleaned_county_df['County'].isin(top_counties)]  # Filter the dat
top_county_data_overall = cleaned_county_overall_df[cleaned_county_overall_df['County'].isin(top_cou

fig, axs = plt.subplots(nrows=2, ncols=1, figsize=(12, 16))  # Create a figure and axes with a 2x1 layout

# Subplot 1: Countplot for top 10 counties' heart disease mortality death rates
sns.countplot(data=top_county_data, x='County', order=top_counties, ax=axs[0])
axs[0].set_title('Top 10 Heart Disease Mortality Death Rate by County', fontsize=12)
axs[0].tick_params(axis='x', labelrotation=45, labelsize=8)  # Rotate and decrease x-axis tick label size
axs[0].set_xlabel('County', fontsize=12)
axs[0].set_ylabel('Count', fontsize=12)

# Subplot 2: Countplot for top 10 counties' heart disease mortality death rates (overall)
sns.countplot(data=top_county_data_overall, x='County', order=top_counties_overall, ax=axs[1])
axs[1].set_title('Top 10 Heart Disease Mortality Death Rate by County (Overall)', fontsize=12)
```

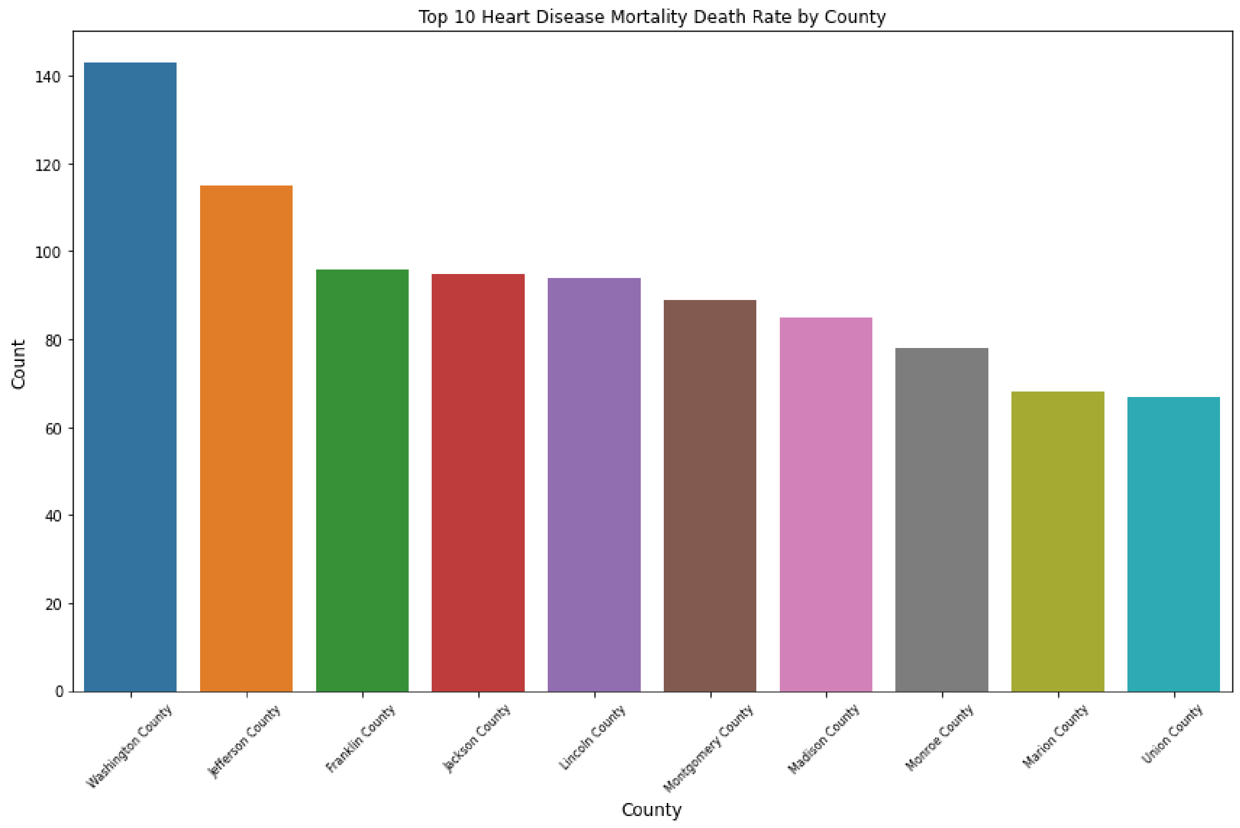
```
axs[1].tick_params(axis='x', labelrotation=45, labelsize=8)  # Rotate and decrease x-axis tick label size
axs[1].set_xlabel('County', fontsize=12)
axs[1].set_ylabel('Count', fontsize=12)

plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()
```

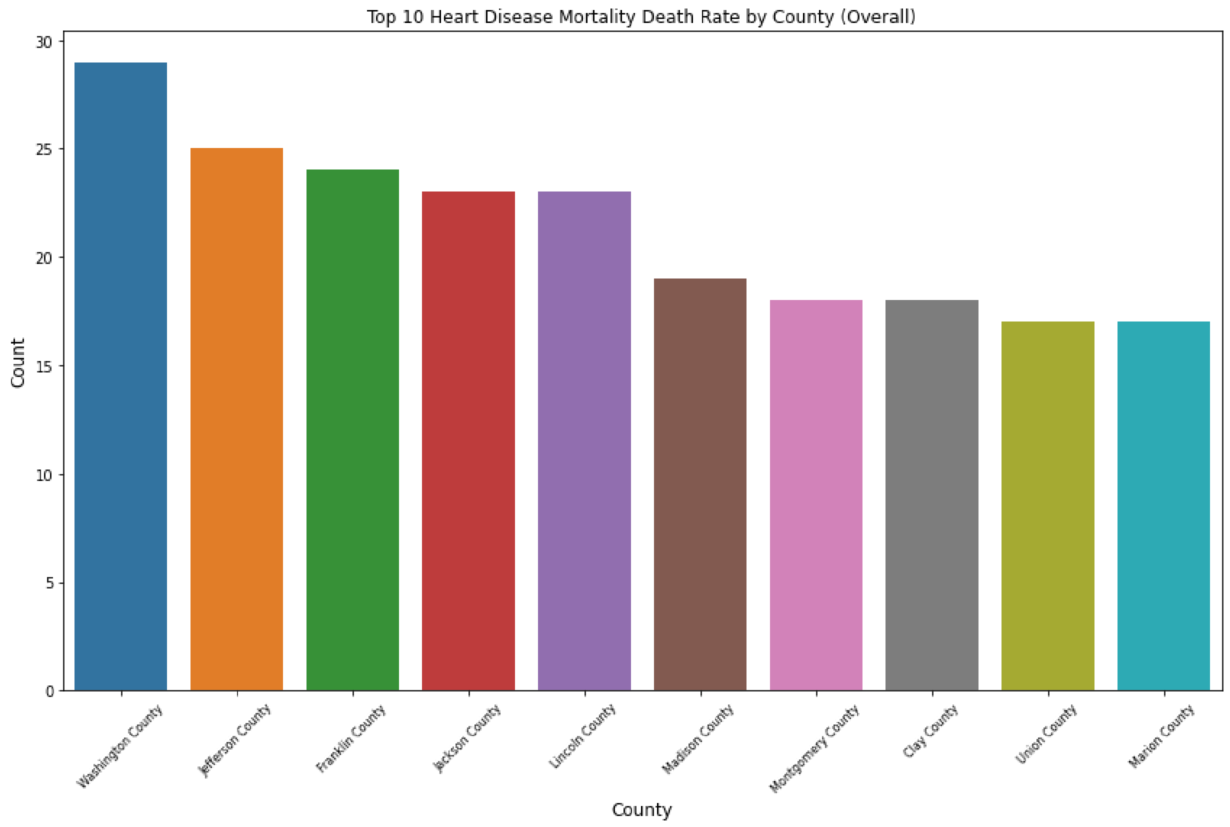





Heart Disease Mortality Death Rate based on State (County)
Count
State
AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC ND NE NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA VT WA WI WV WY

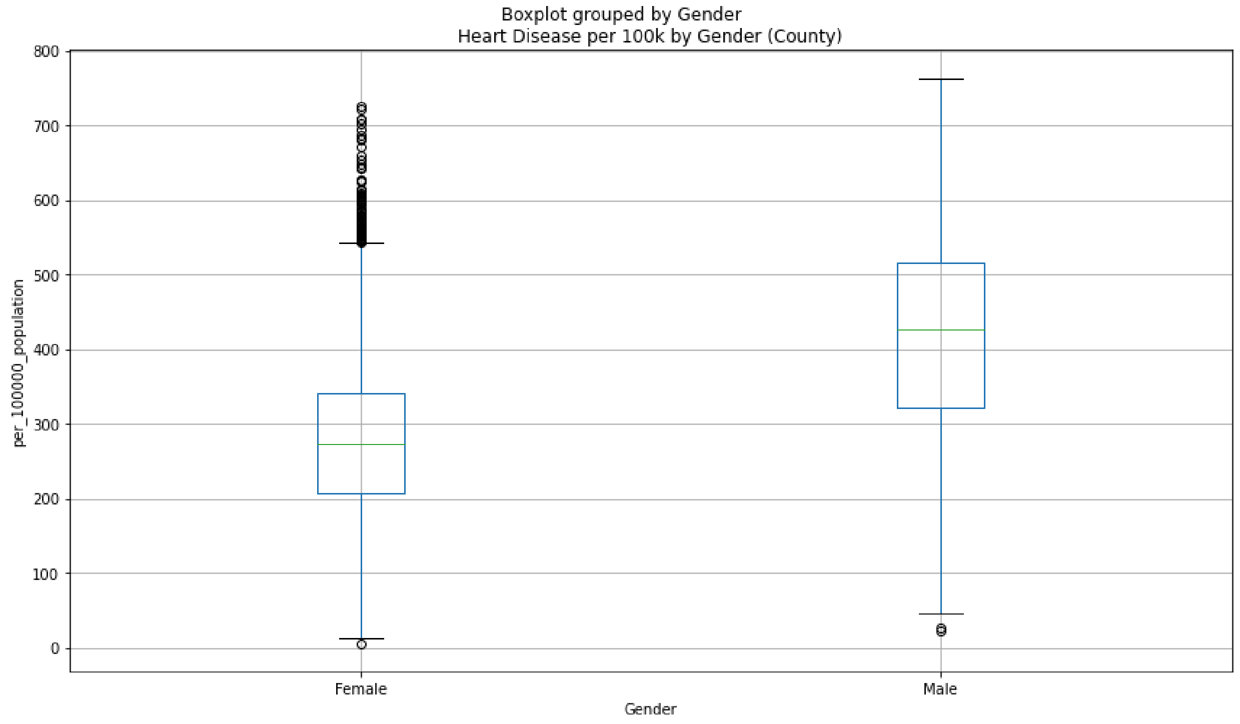

Heart Disease Mortality Death Rate based on State (Overall)
Count
State
AK AL AR AZ AS CA CO CT DC DE FL GA GU HI IA ID IL IN KS KY LA MA MD ME MI MN MO MP MS MT NC ND NE NH NJ NM NV NY OH OK OR PA PR RI SC SD TN TX UT VA VT WA WI WV WY

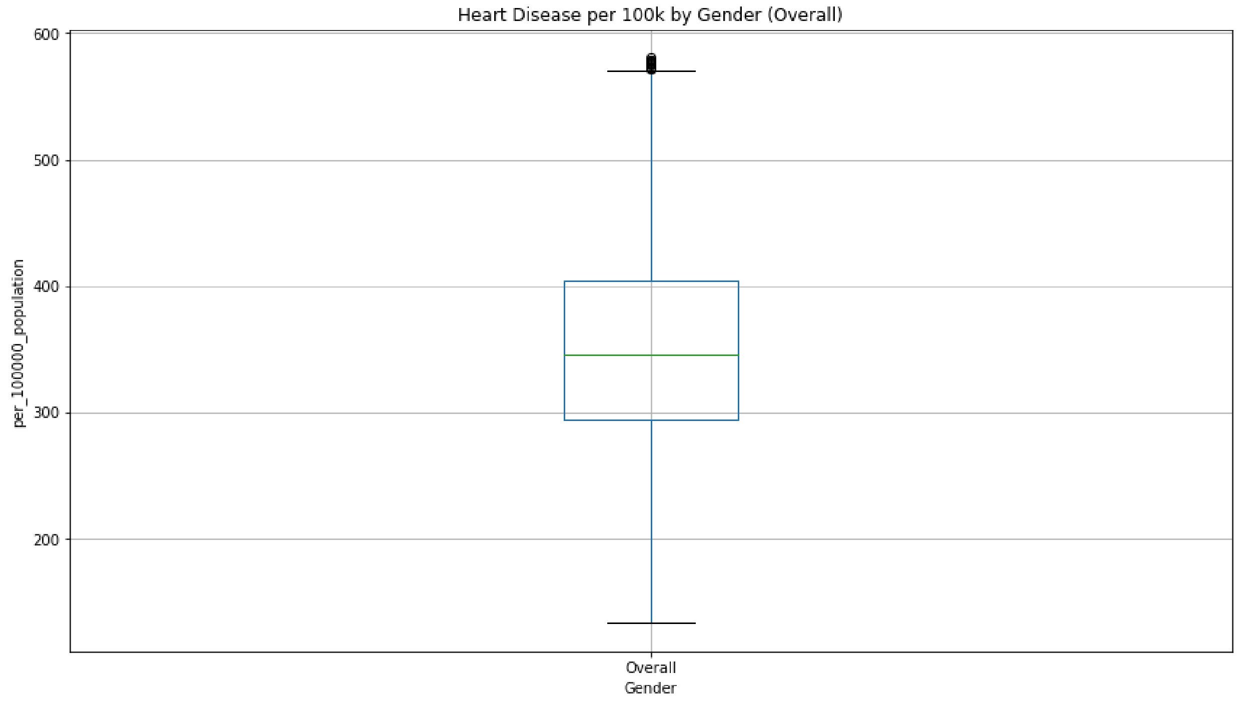


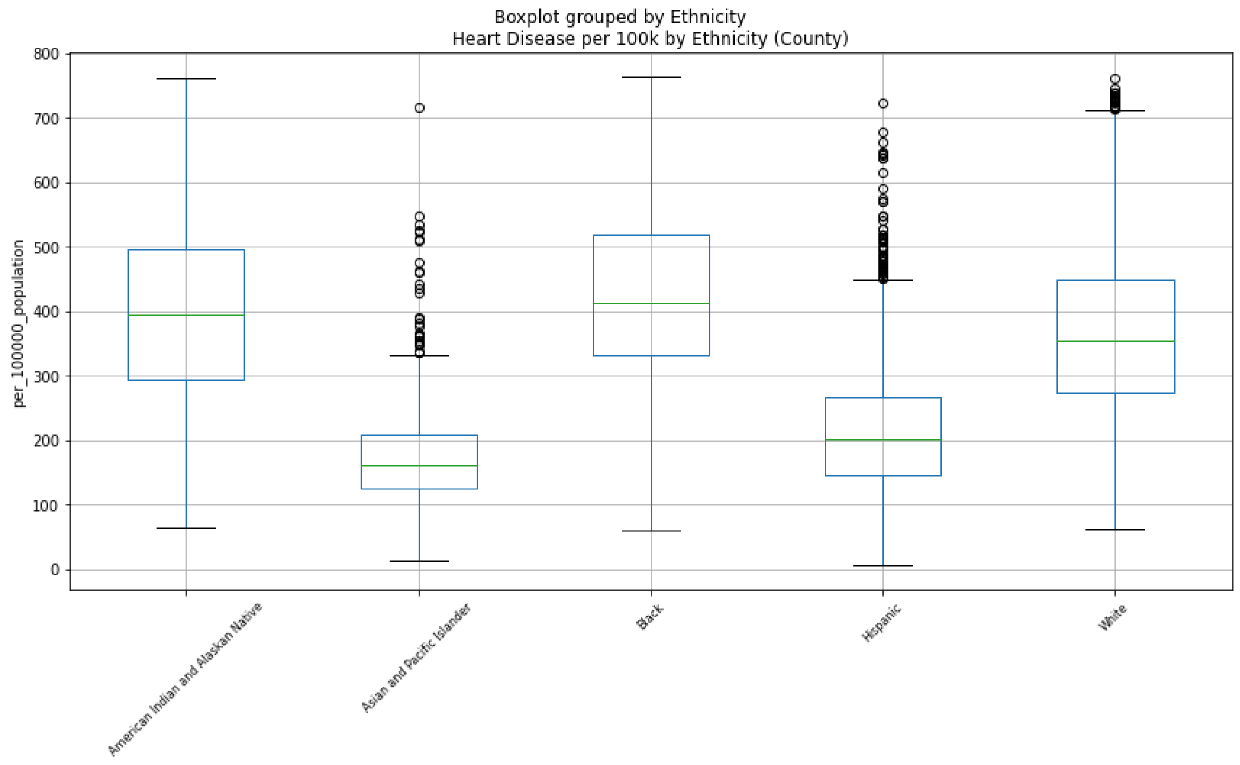


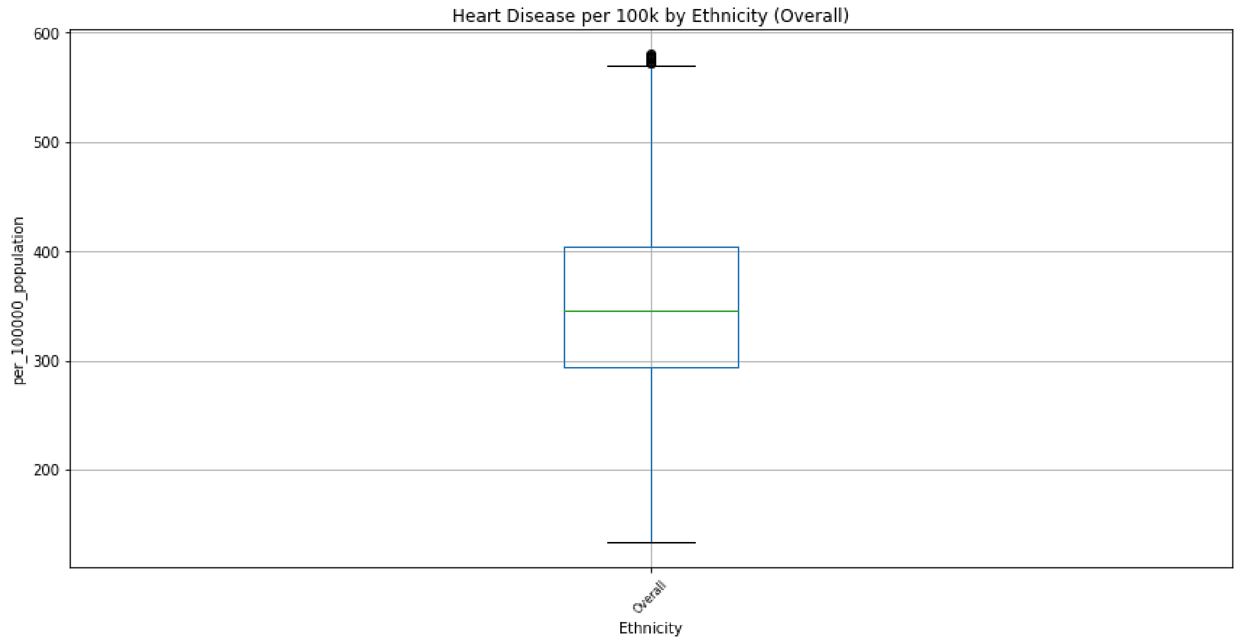
In [37]:
```
# Create a figure and axes for the first set of subplots
fig, axes1 = plt.subplots(nrows=2, ncols=1, figsize=(12, 14))

# Subplot 1: Box plot for Heart Disease per 100k by Gender in cleaned_county_df
cleaned_county_df.boxplot(column='Heart Disease per 100k', by='Gender', ax=axes1[0])
axes1[0].set_title('Heart Disease per 100k by Gender (County)')
axes1[0].set_ylabel('per_100000_population')
```

```
# Subplot 2: Box plot for Heart Disease per 100k by Gender in cleaned_county_overall_df
cleaned_county_overall_df.boxplot(column='Heart Disease per 100k', by='Gender', ax=axes1[1])
axes1[1].set_title('Heart Disease per 100k by Gender (Overall)')
axes1[1].set_ylabel('per_100000_population')

plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()

# Create a new figure and axes for the second set of subplots
fig, axes2 = plt.subplots(nrows=2, ncols=1, figsize=(12, 14))

# Subplot 1: Box plot for Heart Disease per 100k by Ethnicity in cleaned_county_df
cleaned_county_df.boxplot(column='Heart Disease per 100k', by='Ethnicity', ax=axes2[0])
axes2[0].set_title('Heart Disease per 100k by Ethnicity (County)')
axes2[0].set_ylabel('per_100000_population')
axes2[0].tick_params(axis='x', rotation=45, labelsize=8)  # Rotate and decrease x-axis tick label size

# Subplot 2: Box plot for Heart Disease per 100k by Ethnicity in cleaned_county_overall_df
cleaned_county_overall_df.boxplot(column='Heart Disease per 100k', by='Ethnicity', ax=axes2[1])
axes2[1].set_title('Heart Disease per 100k by Ethnicity (Overall)')
axes2[1].set_ylabel('per_100000_population')
axes2[1].tick_params(axis='x', rotation=45, labelsize=8)  # Rotate and decrease x-axis tick label size

plt.tight_layout()  # Adjust layout to prevent overlapping
plt.show()
```


Boxplot grouped by Gender
Heart Disease per 100k by Gender (County)
per_100000_population
0
100
200
300
400
500
600
700
800
Female
Male
Gender


Heart Disease per 100k by Gender (Overall)
per_100000_population
200
300
400
500
600
Overall
Gender





In [38]:
```
# Chi-square test for gender
contingency_gender = pd.crosstab(cleaned_county_df['Heart Disease per 100k'], cleaned_county_df['Ge
chi2_stat_gender, p_val_gender, _, _ = chi2_contingency(contingency_gender)

# Chi-square test for race
contingency_race = pd.crosstab(cleaned_county_df['Heart Disease per 100k'], cleaned_county_df['Ethni
chi2_stat_race, p_val_race, _, _ = chi2_contingency(contingency_race)

# Chi-square test for county
contingency_geo = pd.crosstab(cleaned_county_df['Heart Disease per 100k'], cleaned_county_df['County
chi2_stat_geo, p_val_geo, _, _ = chi2_contingency(contingency_geo)

# Chi-square test for state
contingency_state = pd.crosstab(cleaned_county_df['Heart Disease per 100k'], cleaned_county_df['State
chi2_stat_state, p_val_state, _, _ = chi2_contingency(contingency_state)
```

```
# Create a DataFrame for the chi-square statistics and p-values
data = {
    'Category': ['Gender', 'Ethnicity', 'County', 'State'],
    'Chi-square statistic': [chi2_stat_gender, chi2_stat_race, chi2_stat_geo, chi2_stat_state],
    'p-value': [p_val_gender, p_val_race, p_val_geo, p_val_state]
}

# Create the DataFrame for Chi-Square test
chi_square_df = pd.DataFrame(data)

# Print the DataFrame
print(chi_square_df)
```

```
    Category  Chi-square statistic        p-value
0     Gender          7.171499e+03   3.097805e-69
1  Ethnicity          2.560000e+04  2.355235e-111
2     County          9.512658e+06   1.000000e+00
3      State          2.557907e+05   9.999865e-01
```

In [39]:
```
# Calculate the overall mean heart disease rate
mean_heart_disease = cleaned_county_df['Heart Disease per 100k'].mean()

# Iterate through each state and perform the Z-test
for state in cleaned_county_df['State'].unique():
    heart_disease_state = cleaned_county_df[cleaned_county_df['State'] == state]['Heart Disease per 10

    # Performing the Z-test
    z_stat = (heart_disease_state.mean() - mean_heart_disease) / (heart_disease_state.std() / np.sqrt
    p_value = stats.norm.cdf(z_stat) * 2  # two-tailed test

    # Round Z-statistic and P-value to two decimals
    z_stat_rounded = round(z_stat, 2)
    p_value_rounded = round(p_value, 2)

    # Outputting the result
    print(f"Z-test for {state}:")
    print(f"Z-statistic: {z_stat_rounded}")
    print(f"P-value: {p_value_rounded}")
    if p_value < 0.05:
        print("The mean heart disease rate for this state is significantly different from the overall mean.
    else:
        print("The mean heart disease rate for this state is not significantly different from the overall m
    print()

# Z-test for Ethnicity
# Iterate through each ethnicity and perform the Z-test
for ethnicity in cleaned_county_df['Ethnicity'].unique():
    heart_disease_ethnicity = cleaned_county_df[cleaned_county_df['Ethnicity'] == ethnicity]['Heart Dis

    # Performing the Z-test
    z_stat = (heart_disease_ethnicity.mean() - mean_heart_disease) / (heart_disease_ethnicity.std() /
    p_value = stats.norm.cdf(z_stat) * 2  # two-tailed test

    # Round Z-statistic and P-value to two decimals
    z_stat_rounded = round(z_stat, 2)
    p_value_rounded = round(p_value, 2)

    # Outputting the result
    print(f"Z-test for {ethnicity}:")
    print(f"Z-statistic: {z_stat_rounded}")
    print(f"P-value: {p_value_rounded}")
    if p_value < 0.05:
        print("The mean heart disease rate for this ethnicity is significantly different from the overall m
    else:
        print("The mean heart disease rate for this ethnicity is not significantly different from the overa
    print()

# Z-test for Gender
# Iterate through each gender and perform the Z-test
for gender in cleaned_county_df['Gender'].unique():
    heart_disease_gender = cleaned_county_df[cleaned_county_df['Gender'] == gender]['Heart Disease

    # Performing the Z-test
    z_stat = (heart_disease_gender.mean() - mean_heart_disease) / (heart_disease_gender.std() / np.
    p_value = stats.norm.cdf(z_stat) * 2  # two-tailed test

    # Round Z-statistic and P-value to two decimals
    z_stat_rounded = round(z_stat, 2)
    p_value_rounded = round(p_value, 2)

    # Outputting the result
    print(f"Z-test for {gender}:")
    print(f"Z-statistic: {z_stat_rounded}")
    print(f"P-value: {p_value_rounded}")
    if p_value < 0.05:
        print("The mean heart disease rate for this gender is significantly different from the overall mea
    else:
        print("The mean heart disease rate for this gender is not significantly different from the overall
    print()
```

```
Z-test for AK:
Z-statistic: -1.62
P-value: 0.11
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for AL:
Z-statistic: 11.87
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for AR:
Z-statistic: 12.6
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for AZ:
Z-statistic: -7.56
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for CA:
Z-statistic: -9.3
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for CO:
Z-statistic: -19.56
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for CT:
Z-statistic: -7.8
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for DC:
Z-statistic: -1.08
P-value: 0.28
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for DE:
Z-statistic: -3.04
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for FL:
Z-statistic: -11.37
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for GA:
Z-statistic: 5.96
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for HI:
Z-statistic: -1.89
P-value: 0.06
The mean heart disease rate for this state is not significantly different from the overall mean.
```

```
Z-test for IA:
Z-statistic: -0.69
P-value: 0.49
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for ID:
Z-statistic: -8.2
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for IL:
Z-statistic: -0.52
P-value: 0.6
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for IN:
Z-statistic: 1.76
P-value: 1.92
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for KS:
Z-statistic: -2.77
P-value: 0.01
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for KY:
Z-statistic: 11.18
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for LA:
Z-statistic: 9.9
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for MA:
Z-statistic: -11.18
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for MD:
Z-statistic: -4.44
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for ME:
Z-statistic: -2.19
P-value: 0.03
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for MI:
Z-statistic: 2.14
P-value: 1.97
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for MN:
Z-statistic: -14.01
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.
```

```
Z-test for MO:
Z-statistic: 7.69
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for MS:
Z-statistic: 18.52
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for MT:
Z-statistic: 0.18
P-value: 1.15
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for NC:
Z-statistic: -4.3
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for ND:
Z-statistic: -2.35
P-value: 0.02
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for NE:
Z-statistic: -6.8
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for NH:
Z-statistic: -6.24
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for NJ:
Z-statistic: -6.82
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for NM:
Z-statistic: -6.32
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for NV:
Z-statistic: -0.86
P-value: 0.39
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for NY:
Z-statistic: -3.92
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for OH:
Z-statistic: 1.78
P-value: 1.92
The mean heart disease rate for this state is not significantly different from the overall mean.
```

```
Z-test for OK:
Z-statistic: 12.53
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for OR:
Z-statistic: -13.78
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for PA:
Z-statistic: -2.78
P-value: 0.01
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for RI:
Z-statistic: -5.72
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for SC:
Z-statistic: 0.34
P-value: 1.27
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for SD:
Z-statistic: -1.29
P-value: 0.2
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for TN:
Z-statistic: 10.22
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for TX:
Z-statistic: 3.04
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for UT:
Z-statistic: -9.95
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for VA:
Z-statistic: -3.89
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for VT:
Z-statistic: -2.24
P-value: 0.03
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for WA:
Z-statistic: -9.59
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.
```

```
Z-test for WI:
Z-statistic: -5.06
P-value: 0.0
The mean heart disease rate for this state is significantly different from the overall mean.

Z-test for WV:
Z-statistic: 5.87
P-value: 2.0
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for WY:
Z-statistic: -1.04
P-value: 0.3
The mean heart disease rate for this state is not significantly different from the overall mean.

Z-test for White:
Z-statistic: 14.14
P-value: 2.0
The mean heart disease rate for this ethnicity is not significantly different from the overall mean.

Z-test for Black:
Z-statistic: 37.39
P-value: 2.0
The mean heart disease rate for this ethnicity is not significantly different from the overall mean.

Z-test for Hispanic:
Z-statistic: -58.1
P-value: 0.0
The mean heart disease rate for this ethnicity is significantly different from the overall mean.

Z-test for American Indian and Alaskan Native:
Z-statistic: 9.97
P-value: 2.0
The mean heart disease rate for this ethnicity is not significantly different from the overall mean.

Z-test for Asian and Pacific Islander:
Z-statistic: -81.73
P-value: 0.0
The mean heart disease rate for this ethnicity is significantly different from the overall mean.

Z-test for Male:
Z-statistic: 39.28
P-value: 2.0
The mean heart disease rate for this gender is not significantly different from the overall mean.

Z-test for Female:
Z-statistic: -56.6
P-value: 0.0
The mean heart disease rate for this gender is significantly different from the overall mean.
```

## Model Selection and Analysis

### Linear Regression and Clustering

In [40]:
```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [41]:
```
# Create a copy of cleaned_county_df for regression analysis
gender_regression_df = cleaned_county_df.copy()

# Convert 'Gender' to dummy variables
# Now 'Gender' will be encoded as 1 for Male and 0 for Female
gender_regression_df['Gender'] = pd.get_dummies(gender_regression_df['Gender'], drop_first=True)

# Define the independent variable (X) and dependent variable (Y)
x_gender = gender_regression_df['Gender']
y_heart = gender_regression_df['Heart Disease per 100k']

# Add a constant term to the independent variable
x_gender = sm.add_constant(x_gender)

# Fit the regression model
gender_regression_model = sm.OLS(y_heart, x_gender).fit()

# Print the summary of the regression model
print(gender_regression_model.summary())

# Notes for the presentation
# R-Squared shows 23.6% of variability of the heart disease is explained by gender
# F statistic 4167 the model is significantly better fit than a model with no predictors
# prob of F statistics is close to 0 which proves that gender is related to heart disease
# Log-likelihood is for model comparison. Higher is better
# AIC, BIC are for other model comparisons. The lower is better

# males = 1
# females 0
# Const coef: this is to show when all values are 0 (Gender = 0 = female) which shows female average
# Gender Coef: males have a higher disease mortality rate by 139.94 units
# t stat: shows gender is statistically significant
# P>|t|: shows the p-values are close to .00 so are significant
# omnibus: this is small so it is normally distributed
# prob(omnibus): higher values show it is normal
# Durbin-Watson: Since it is not close to two this shows significant autocorrelation
# Cond. No. : This measures multicollinearity. Values greater than 30 indicate multicollinearity

# MODEL AND DATA IS SIGNIFICANT
```

```
                             OLS Regression Results
==============================================================================
Dep. Variable:     Heart Disease per 100k   R-squared:                       0.236
Model:                            OLS   Adj. R-squared:                  0.236
Method:                 Least Squares   F-statistic:                     4167.
Date:                Sat, 24 Feb 2024   Prob (F-statistic):               0.00
Time:                        18:18:23   Log-Likelihood:                -84329.
No. Observations:               13484   AIC:                         1.687e+05
Df Residuals:                   13482   BIC:                         1.687e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        275.9017      1.545    178.542      0.000     272.873     278.931
Gender       139.9395      2.168     64.550      0.000     135.690     144.189
==============================================================================
Omnibus:                        0.196   Durbin-Watson:                   0.726
Prob(Omnibus):                  0.907   Jarque-Bera (JB):                0.217
Skew:                          -0.005   Prob(JB):                        0.897
Kurtosis:                       2.983   Cond. No.                         2.64
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
c:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a futur
e version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [42]:

```
# Create a copy of cleaned_county_df for regression analysis
ethnicity_regression_df = cleaned_county_df.copy()

# One-hot encode the 'Ethnicity' column
ethnicity_dummies = pd.get_dummies(ethnicity_regression_df['Ethnicity'])

# Concatenate the dummy variables with the original DataFrame
ethnicity_regression_dummies = pd.concat([ethnicity_regression_df, ethnicity_dummies], axis=1)

# Define the independent variables (X) and dependent variable (Y)
x_ethnicity = ethnicity_regression_dummies[['White', 'Black', 'Hispanic', 'American Indian and Alaskan Na
y_heart = ethnicity_regression_dummies['Heart Disease per 100k']

# Add a constant term to the independent variables
x_ethnicity = sm.add_constant(x_ethnicity)

# Fit the regression model
ethnicity_regression_model = sm.OLS(y_heart, x_ethnicity).fit()

# Print the summary of the regression model
print(ethnicity_regression_model.summary())

# R-Squared: 28% of the data is explained by ethnicity
# F statistic: 1329 shows the model is significant
# Prob of F statistics: is close to 0 which shows it's significant

# Log-likelihood: is for model comparison. Higher is better
# AIC, BIC: are for other model comparisons. The lower is better

# Const coef: the average when no one has ethnicity (the default is assumed white)
# Rest of Coef: average heart disease for each ethnicity
# t stat: larger absolute values indicate greater evidence against the null hypothesis
# P>|t|: no significance since close to 1

# MODEL is significant but the data is not
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Heart Disease per 100k   R-squared:                       0.340
Model:                                OLS   Adj. R-squared:                  0.340
Method:                     Least Squares   F-statistic:                     1737.
Date:                    Sat, 24 Feb 2024   Prob (F-statistic):               0.00
Time:                            18:18:23   Log-Likelihood:                -83342.
No. Observations:                   13484   AIC:                         1.667e+05
Df Residuals:                       13479   BIC:                         1.667e+05
Df Model:                               4
Covariance Type:                nonrobust
===================================================================================
=====
                                        coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
---
const                               265.1469      1.126    235.436      0.000     262.939     267.354
White                               103.0378      1.653     62.343      0.000      99.798     106.277
Black                               163.3917      1.976     82.673      0.000     159.518     167.266
Hispanic                            -47.6185      2.408    -19.779      0.000     -52.338     -42.899
American Indian and Alaskan Native   138.5605      3.845     36.036      0.000     131.024     146.097
Asian and Pacific Islander           -92.2246      3.092    -29.824      0.000     -98.286     -86.163
==============================================================================
Omnibus:                      570.105   Durbin-Watson:                   0.821
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              644.453
Skew:                           0.534   Prob(JB):                    1.14e-140
Kurtosis:                       3.081   Cond. No.                     1.44e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 8.69e-27. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
c:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a futur
e version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [43]:

```
# Make a copy of the dummy data without the 'White' column to remove multicollinearity
ethnicity_regression_dummies_noCol = ethnicity_regression_dummies.drop(columns=['White'])

# Define the predictor variables after removing 'White' to address multicollinearity
x_no_white = ethnicity_regression_dummies_noCol[['Black', 'Hispanic', 'American Indian and Alaskan Nati

# Define the target variable
y_heart = ethnicity_regression_dummies['Heart Disease per 100k']

# Calculate Variance Inflation Factor (VIF) to detect multicollinearity
vif_data = sm.add_constant(x_no_white)
vif = pd.DataFrame()
vif["Variable"] = vif_data.columns
vif["VIF"] = [variance_inflation_factor(vif_data.values, i) for i in range(vif_data.shape[1])]

# Identify variables with VIF greater than 10 (common threshold indicating multicollinearity)
high_vif_variables = vif[vif["VIF"] > 10]["Variable"].tolist()
```

```
# Drop high VIF variables from the predictor variables
x_no_white = x_no_white.drop(columns=high_vif_variables)
x_no_white = sm.add_constant(x_no_white)

# Fit Ordinary Least Squares (OLS) regression model using the updated predictor variables
# and the target variable y_heart
model = sm.OLS(y_heart, x_no_white).fit()

# Print the summary of the regression model
print("\nModel Summary After Addressing Multicollinearity:")
print(model.summary())

# R-Squared: 31% of the data is explained by ethnicity
# F statistic: 1529 shows the model is significant
# Prob of F statistics: is close to 0 which shows it's significant

# Log-likelihood (negative does not matter): is for model comparison. Higher is better
# AIC, BIC: are for other model comparisons. The lower is better

# Const coef: the average when no one has ethnicity (the default is assumed white)
# black coef: higher than white
# hispanic coef: lower than white
# indian coef: higher than white
# asian coef: worse than white
# t stat: larger absolute values indicate greater evidence against the null hypothesis
# P>|t|: significance since close to .00

# MODEL AND DATA IS SIGNIFICANT
```

```
Model Summary After Addressing Multicollinearity:
                              OLS Regression Results
==============================================================================
Dep. Variable:     Heart Disease per 100k   R-squared:                       0.340
Model:                                OLS   Adj. R-squared:                  0.340
Method:                     Least Squares   F-statistic:                     1737.
Date:                    Sat, 24 Feb 2024   Prob (F-statistic):               0.00
Time:                            18:18:23   Log-Likelihood:                -83342.
No. Observations:                   13484   AIC:                         1.667e+05
Df Residuals:                       13479   BIC:                         1.667e+05
Df Model:                               4
Covariance Type:                nonrobust
===================================================================================
=====
                                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
---
const                             368.1847      1.482    248.516      0.000     365.281     371.089
Black                              60.3540      2.480     24.334      0.000      55.492      65.216
Hispanic                         -150.6563      2.998    -50.256      0.000    -156.532    -144.780
American Indian and Alaskan Native    35.5227      4.740      7.494      0.000      26.231      44.814
Asian and Pacific Islander      -195.2623      3.826    -51.039      0.000    -202.761    -187.763
==============================================================================
Omnibus:                      570.105   Durbin-Watson:                   0.821
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              644.453
Skew:                           0.534   Prob(JB):                    1.14e-140
Kurtosis:                       3.081   Cond. No.                         5.19
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
c:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a futur
e version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [44]:
```
# Make a copy of the original dataframe
state_regression_df = cleaned_county_df.copy()

# Create hot encoded data, dropping the first state (AK)
state_regression_encode = pd.get_dummies(state_regression_df, columns=['State'], drop_first=True)

# Define predictor variables (hot encoded states) and target variable (heart disease rate)
x_hot_encoded_state = state_regression_encode.drop(['Heart Disease per 100k', 'County', 'Gender', 'Ethr
y_heart = state_regression_encode['Heart Disease per 100k']

# Calculate Variance Inflation Factor (VIF) to detect multicollinearity
vif_data_state = sm.add_constant(x_hot_encoded_state)
vif_state = pd.DataFrame()
vif_state["Variable"] = vif_data_state.columns
vif_state["VIF"] = [variance_inflation_factor(vif_data_state.values, i) for i in range(vif_data_state.shape

# Identify variables with VIF greater than 10 (common threshold indicating multicollinearity)
high_vif_variables = vif_state[vif_state["VIF"] > 10]["Variable"].tolist()

# Remove constant from high VIF variables list
high_vif_variables.remove('const')

# Drop variables with high VIF
x_hot_encoded_state = x_hot_encoded_state.drop(high_vif_variables, axis=1)

# Add constant
x_hot_encoded_state = sm.add_constant(x_hot_encoded_state)

# Fit Ordinary Least Squares (OLS) regression model using the updated predictor variables and the tar
model = sm.OLS(y_heart, x_hot_encoded_state).fit()

# Print the summary of the regression model
print("\nModel Summary After Addressing Multicollinearity:")
print(model.summary())


# R-Squared: 16% of the data is explained by the ethnicity
# F statistic: 54 shows the model is significant
# Prob of F statistics: is close to 0 which shows it's significant

# Log-likelihood (negative does not matter): is for model comparison. Higher is better
# AIC, BIC: are for other model comparisons. The lower is better

# Const coef: the average when no one has the state (the default is assumed Alaska)
# t stat: larger absolute values indicate greater evidence against the null hypothesis
# P>|t|: Depends on the state, some of them are not significant. These would be the states to study

# MODEL AND DATA ARE SIGNIFICANT (depending on state)

# Based on the chi-square test, these results for significant contribution to heart disease mortality
# can be due to random chance. It is best to examine the counties that do not have significance if you
# deep dive more and go under the assumption that this is not due to random chance.

# Texas and Georgia were removed due to high VIF.
```

```
c:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a futur
e version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

```
Model Summary After Addressing Multicollinearity:
                            OLS Regression Results
==============================================================================
Dep. Variable:     Heart Disease per 100k   R-squared:                   0.165
Model:                            OLS   Adj. R-squared:              0.162
Method:                 Least Squares   F-statistic:                 55.22
Date:                Sat, 24 Feb 2024   Prob (F-statistic):           0.00
Time:                        18:18:24   Log-Likelihood:            -84931.
No. Observations:               13484   AIC:                     1.700e+05
Df Residuals:                   13435   BIC:                     1.703e+05
Df Model:                          48
Covariance Type:            nonrobust
==============================================================================
                coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       364.7796      2.867    127.256      0.000     359.161     370.398
State_AL     88.6038      8.267     10.717      0.000      72.398     104.809
State_AR     88.5869      8.331     10.633      0.000      72.257     104.917
State_AZ    -90.5651     11.869     -7.630      0.000    -113.830     -67.300
State_CA    -66.5237      6.399    -10.396      0.000     -79.066     -53.981
State_CO   -116.3073      8.098    -14.363      0.000    -132.180    -100.435
State_CT   -115.3390     16.726     -6.896      0.000    -148.124     -82.554
State_DC    -94.7171     46.696     -2.028      0.043    -186.248      -3.186
State_DE   -102.2251     28.252     -3.618      0.000    -157.602     -46.848
State_FL    -82.7003      6.882    -12.017      0.000     -96.189     -69.211
State_HI    -52.1943     22.789     -2.290      0.022     -96.865      -7.524
State_IA    -22.6363      8.649     -2.617      0.009     -39.589      -5.683
State_ID    -85.7739     12.227     -7.015      0.000    -109.741     -61.807
State_IL    -20.9818      7.063     -2.970      0.003     -34.827      -7.136
State_IN     -5.6106      7.507     -0.747      0.455     -20.326       9.104
State_KS    -35.5576      7.984     -4.454      0.000     -51.207     -19.908
State_KY     61.3907      7.507      8.178      0.000      46.676      76.106
State_LA     76.7923      8.410      9.131      0.000      60.307      93.277
State_MA   -136.2924     13.896     -9.808      0.000    -163.530    -109.055
State_MD    -70.5372     11.104     -6.352      0.000     -92.303     -48.771
State_ME    -50.2948     23.127     -2.175      0.030     -95.626      -4.963
State_MI     -1.7622      7.561     -0.233      0.816     -16.583      13.059
State_MN   -107.0522      8.619    -12.420      0.000    -123.948     -90.157
State_MO     39.4462      7.589      5.198      0.000      24.571      54.321
State_MS    121.7279      7.782     15.642      0.000     106.474     136.982
State_MT    -15.8614     11.391     -1.393      0.164     -38.189       6.466
State_NC    -44.9266      6.413     -7.005      0.000     -57.498     -32.356
State_ND    -42.9216     12.420     -3.456      0.001     -67.267     -18.577
State_NE    -62.2146      9.282     -6.703      0.000     -80.408     -44.021
State_NH   -124.1877     21.861     -5.681      0.000    -167.039     -81.337
State_NJ    -85.2237     10.567     -8.065      0.000    -105.936     -64.511
State_NM    -67.7342     10.057     -6.735      0.000     -87.448     -48.020
State_NV    -29.6679     12.420     -2.389      0.017     -54.013      -5.323
State_NY    -42.7607      7.386     -5.789      0.000     -57.239     -28.282
State_OH     -5.9818      7.056     -0.848      0.397     -19.813       7.850
State_OK     72.8691      6.954     10.479      0.000      59.239      86.499
State_OR   -117.0711      9.775    -11.977      0.000    -136.231     -97.911
State_PA    -38.1785      7.626     -5.006      0.000     -53.126     -23.231
State_RI   -109.8884     22.789     -4.822      0.000    -154.559     -65.218
State_SC    -14.3324      9.100     -1.575      0.115     -32.169       3.504
State_SD    -31.6919     10.970     -2.889      0.004     -53.194     -10.190
State_TN     56.9666      7.353      7.747      0.000      42.553      71.380
State_UT    -98.2617     12.227     -8.036      0.000    -122.229     -74.295
State_VA    -38.7069      6.200     -6.243      0.000     -50.860     -26.554
State_VT    -50.6996     24.238     -2.092      0.036     -98.210      -3.189
```

```
State_WA      -91.4170       8.708    -10.497      0.000    -108.487     -74.347
State_WI      -58.0087       8.963     -6.472      0.000     -75.577     -40.441
State_WV       30.1527      10.596      2.846      0.004       9.383      50.923
State_WY      -31.8337      15.590     -2.042      0.041     -62.393      -1.274
==============================================================================
Omnibus:                      137.350   Durbin-Watson:                   0.697
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              131.983
Skew:                           0.215   Prob(JB):                     2.19e-29
Kurtosis:                       2.775   Cond. No.                         41.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [45]:
```python
# Define states of interest
states_of_interest = ['AZ', 'CA', 'CO', 'CT', 'DE', 'FL', 'IA', 'ID', 'LA', 'MA',
                      'MD', 'ME', 'MN', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV',
                      'NY', 'OR', 'PA', 'RI', 'UT', 'VA', 'VT', 'WA', 'WI']

# Initialize dictionaries to store coefficients and p-values
coefficients = {}
p_values = {}

# Extract coefficients and p-values from Table 1
table_data = model.summary().tables[1].data
for row in table_data[2:]:  # Skip the first two rows as they contain headers
    state = row[0].split('_')[1]  # Extract state abbreviation
    if state in states_of_interest:
        coef = float(row[1])  # Extract coefficient value
        p_val = float(row[4])  # Extract p-value
        coefficients[state] = coef
        p_values[state] = p_val

# Print coefficients and p-values for the specified states
for state in states_of_interest:
    print(f"State: {state}, Coefficient: {coefficients[state]}, P-value: {p_values[state]}")

# Comments:
# The code extracts coefficients and p-values from Table 1 of the regression model summary and prints
# The specified states of interest are those in the list 'states_of_interest'.
# Coefficients are stored in the 'coefficients' dictionary, and p-values are stored in the 'p_values' dictio
# The code ensures that only coefficients and p-values for the specified states are extracted and print
# The loop iterates through the table data, skipping the first two rows (which contain headers), and spl
# It then checks if the state is in the list of states of interest and extracts the coefficient and p-value
```

```
State: AZ, Coefficient: -90.5651, P-value: 0.0
State: CA, Coefficient: -66.5237, P-value: 0.0
State: CO, Coefficient: -116.3073, P-value: 0.0
State: CT, Coefficient: -115.339, P-value: 0.0
State: DE, Coefficient: -102.2251, P-value: 0.0
State: FL, Coefficient: -82.7003, P-value: 0.0
State: IA, Coefficient: -22.6363, P-value: 0.009
State: ID, Coefficient: -85.7739, P-value: 0.0
State: LA, Coefficient: 76.7923, P-value: 0.0
State: MA, Coefficient: -136.2924, P-value: 0.0
State: MD, Coefficient: -70.5372, P-value: 0.0
State: ME, Coefficient: -50.2948, P-value: 0.03
State: MN, Coefficient: -107.0522, P-value: 0.0
State: NC, Coefficient: -44.9266, P-value: 0.0
State: ND, Coefficient: -42.9216, P-value: 0.001
State: NE, Coefficient: -62.2146, P-value: 0.0
State: NH, Coefficient: -124.1877, P-value: 0.0
State: NJ, Coefficient: -85.2237, P-value: 0.0
State: NM, Coefficient: -67.7342, P-value: 0.0
State: NV, Coefficient: -29.6679, P-value: 0.017
State: NY, Coefficient: -42.7607, P-value: 0.0
State: OR, Coefficient: -117.0711, P-value: 0.0
State: PA, Coefficient: -38.1785, P-value: 0.0
State: RI, Coefficient: -109.8884, P-value: 0.0
State: UT, Coefficient: -98.2617, P-value: 0.0
State: VA, Coefficient: -38.7069, P-value: 0.0
State: VT, Coefficient: -50.6996, P-value: 0.036
State: WA, Coefficient: -91.417, P-value: 0.0
State: WI, Coefficient: -58.0087, P-value: 0.0
```

In [46]:
```python
# Drop the constant column from x_no_white as it's not needed in this context
default_white_race = x_no_white.drop(columns='const')

# Combine gender and race with heart disease data
combined = pd.concat([gender_regression_df[['Gender', 'Heart Disease per 100k']], default_white_race],

# Separate predictors (x_comb) and target (y_comb)
x_comb = combined.drop('Heart Disease per 100k', axis=1)
y_comb = combined['Heart Disease per 100k']

# Add constant for the intercept term
x_comb = sm.add_constant(x_comb)

# Fit Ordinary Least Squares (OLS) regression model
model = sm.OLS(y_comb, x_comb).fit()

# Print model summary
print(model.summary())


# R-Squared: 58% of the data is explained by ethnicity
# F statistic: 3816 show model is significant
# prob of F statistics: is close to 0 which shows it significant

# Log-likelohood(neg does not matter): is for model comparison. Higher is better
# AIC, BIC: are for other model comparisons. the lower is better

# Const coef: the average when someone is a white female (all other refs are 0)
# Gender coef: being male increases 142 units
# black coef: being black incerease by 60
# hispanic coef: lowers by 155
# indian coef: higher by 28
# asian coef: being asian lowers by 196
# t stat: larger absolutes values indicate greater evidence against the null hypothesis
# P>|t|: significance since close to .00

# MODEL AND DATA IS SIGNIFICANT
```

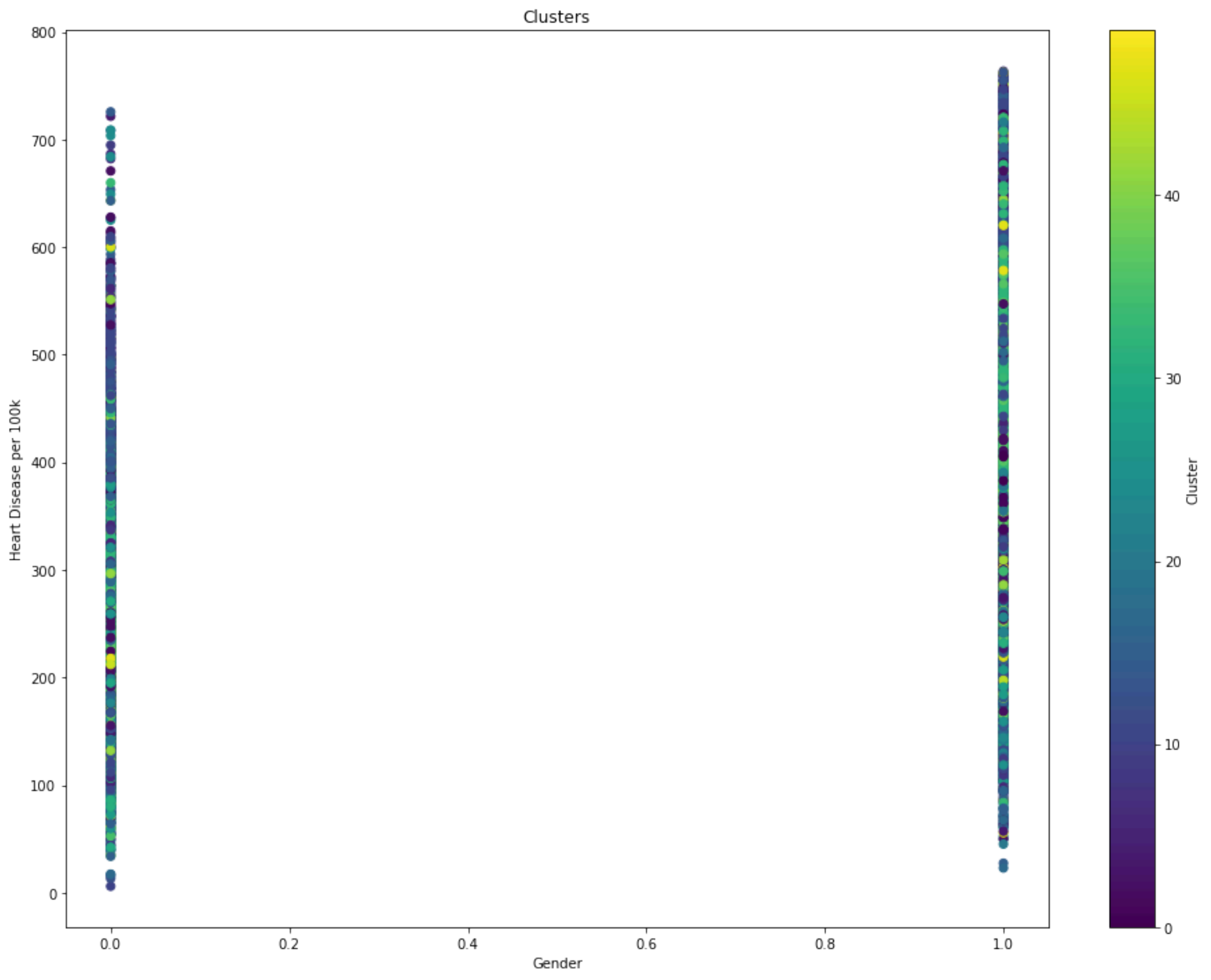
```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Heart Disease per 100k   R-squared:                       0.586
Model:                                OLS   Adj. R-squared:                  0.586
Method:                     Least Squares   F-statistic:                     3816.
Date:                    Sat, 24 Feb 2024   Prob (F-statistic):               0.00
Time:                            18:18:24   Log-Likelihood:                -80198.
No. Observations:                   13484   AIC:                         1.604e+05
Df Residuals:                       13478   BIC:                         1.605e+05
Df Model:                               5
Covariance Type:                nonrobust
========================================================================================
=====
                                      coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
---
const                             296.8629      1.419    209.274      0.000     294.082     299.643
Gender                            142.8956      1.597     89.485      0.000     139.766     146.026
Black                              60.2072      1.964     30.648      0.000      56.357      64.058
Hispanic                         -155.7819      2.375    -65.589      0.000    -160.437    -151.126
American Indian and Alaskan Native    28.7282      3.755      7.650      0.000      21.367      36.089
Asian and Pacific Islander       -196.9472      3.030    -64.994      0.000    -202.887    -191.008
==============================================================================
Omnibus:                      518.354   Durbin-Watson:                   1.229
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              753.898
Skew:                           0.378   Prob(JB):                    1.96e-164
Kurtosis:                       3.878   Cond. No.                         5.87
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

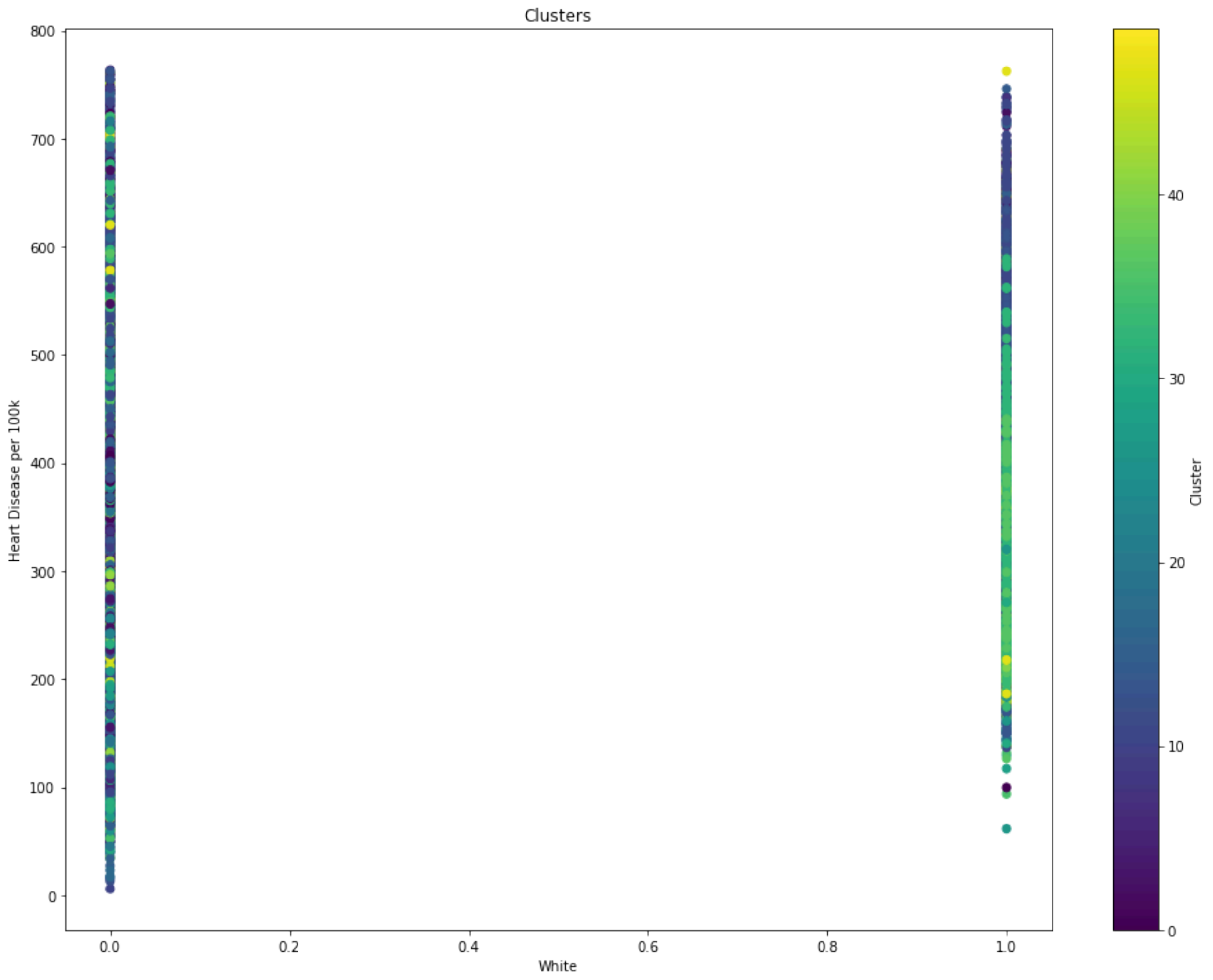
```
c:\Users\User\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a futur
e version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

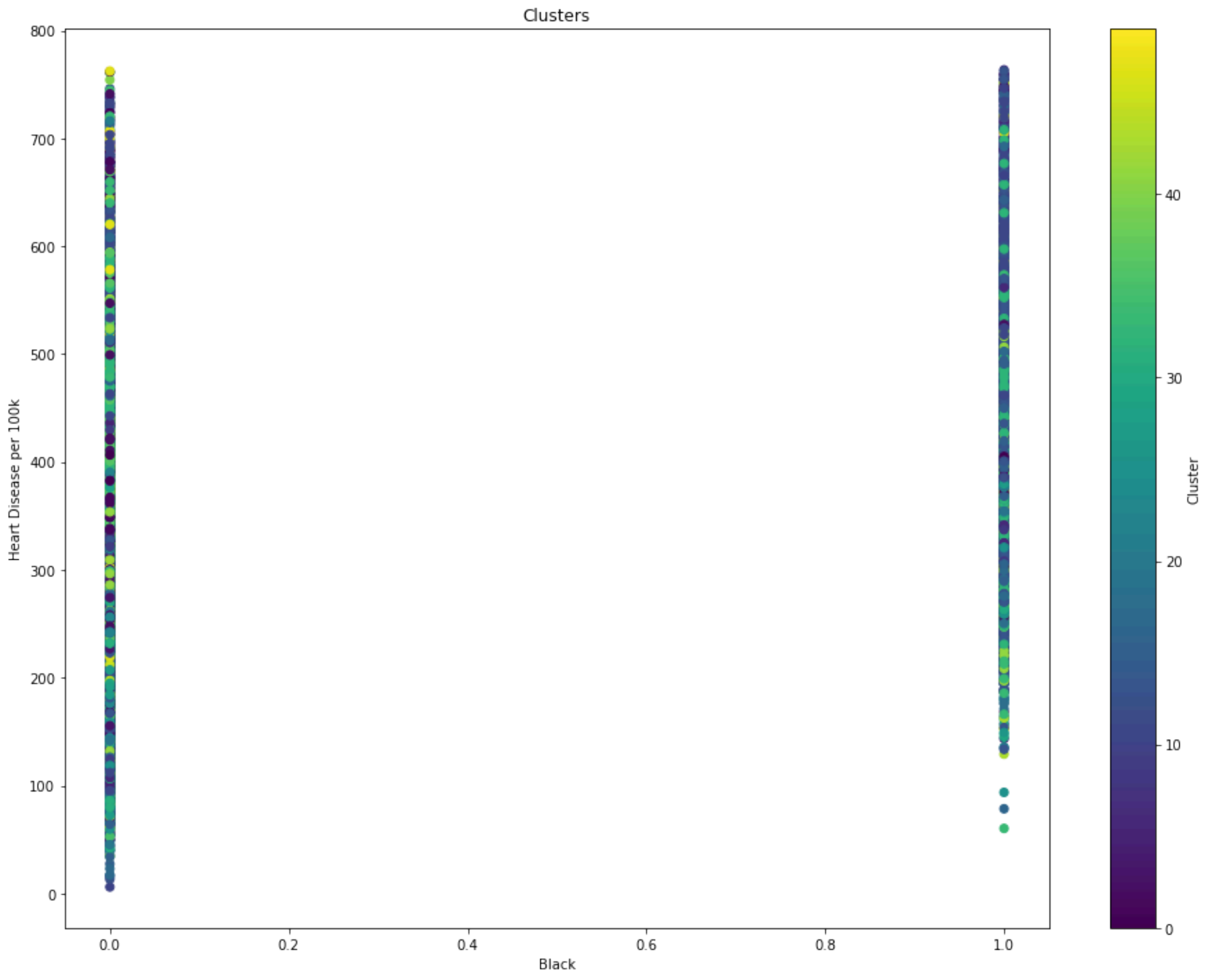
In [47]:
```python
# Extracting necessary data
all_races = ethnicity_regression_dummies.copy()
all_races_only = all_races[['Black', 'Hispanic', 'American Indian and Alaskan Native', 'Asian and Pacific Isl

gender_bi = gender_regression_df.copy()
gender_bi = gender_bi[['Gender', 'Heart Disease per 100k']]

all_state = pd.get_dummies(state_regression_df, columns=['State'])
all_state_only = all_state.drop(['Heart Disease per 100k', 'County', 'Gender', 'Ethnicity'], axis=1)

# Combine all data for clustering
combined_cluster_no_state = combined.copy()
default_state = x_hot_encoded_state.drop(columns='const')
combined_cluster = pd.concat([all_races_only, gender_bi, all_state_only], axis=1)

# Standardize the features
scaler = StandardScaler()
```

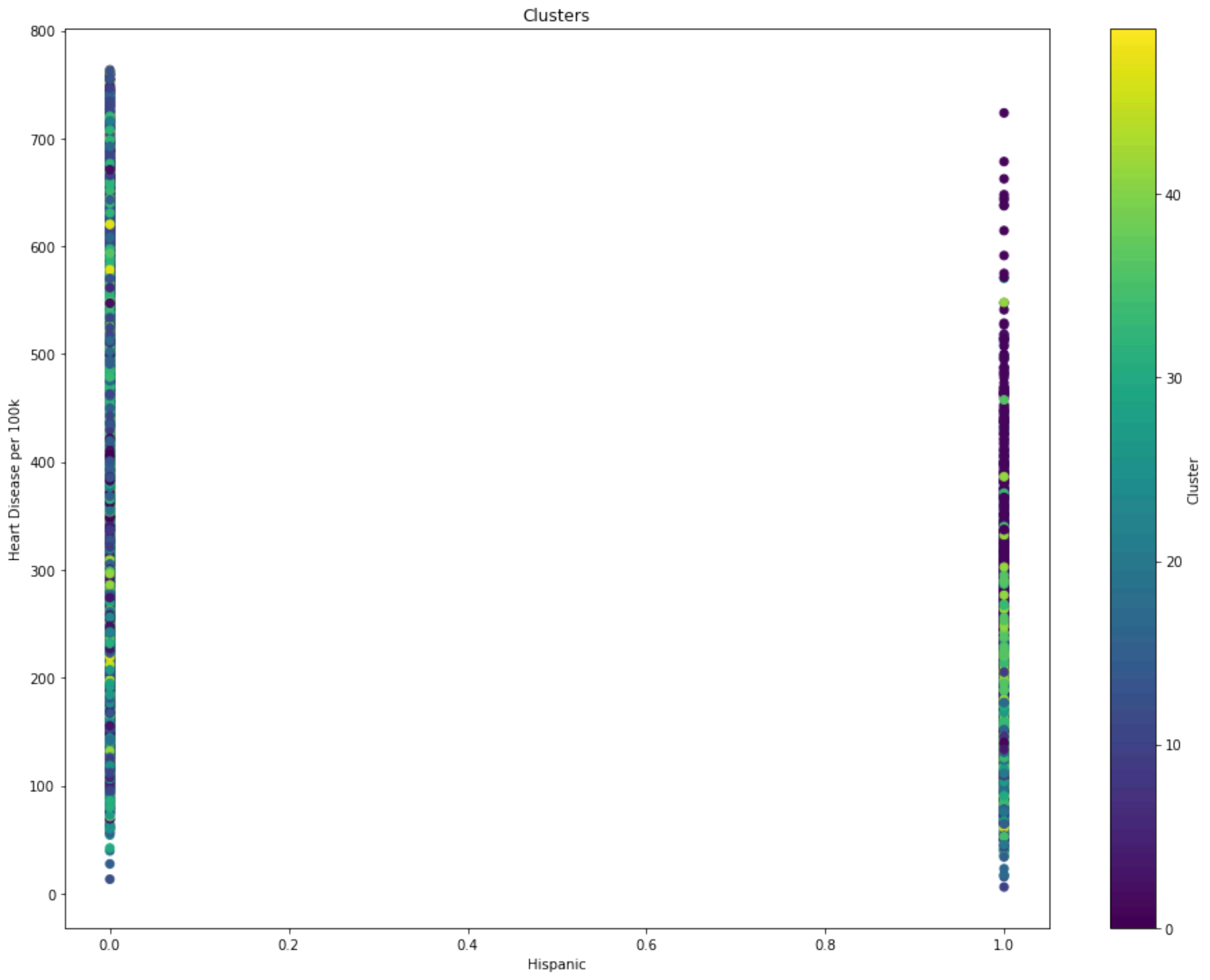
```
combined_cluster_scaled = scaler.fit_transform(combined_cluster)

# Choose the number of clusters
num_clusters = 50

# Initialize and fit the KMeans model
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(combined_cluster_scaled)

# Get cluster labels for each data point
cluster_labels = kmeans.labels_

# Add cluster labels to the DataFrame
combined_cluster['Cluster'] = cluster_labels

# Get centroids
centroids = kmeans.cluster_centers_

# Create a DataFrame to display centroid values
centroid_df = pd.DataFrame(centroids, columns=combined_cluster.columns[:-1])  # Exclude the 'Cluster' 

# Display centroid values
#print("Centroid Values for Each Cluster:")
#print(centroid_df)

# Scatter plot for Gender
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['Gender'], combined_cluster['Heart Disease per 100k'], c=cluster_labels, cm
plt.title('Clusters')
plt.xlabel('Gender')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()

# Scatter plot for White race
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['White'], combined_cluster['Heart Disease per 100k'], c=cluster_labels, cma
plt.title('Clusters')
plt.xlabel('White')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()

# Scatter plot for Black race
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['Black'], combined_cluster['Heart Disease per 100k'], c=cluster_labels, cmap
plt.title('Clusters')
plt.xlabel('Black')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()

# Scatter plot for Hispanic race
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['Hispanic'], combined_cluster['Heart Disease per 100k'], c=cluster_labels, cm
plt.title('Clusters')
plt.xlabel('Hispanic')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()
```

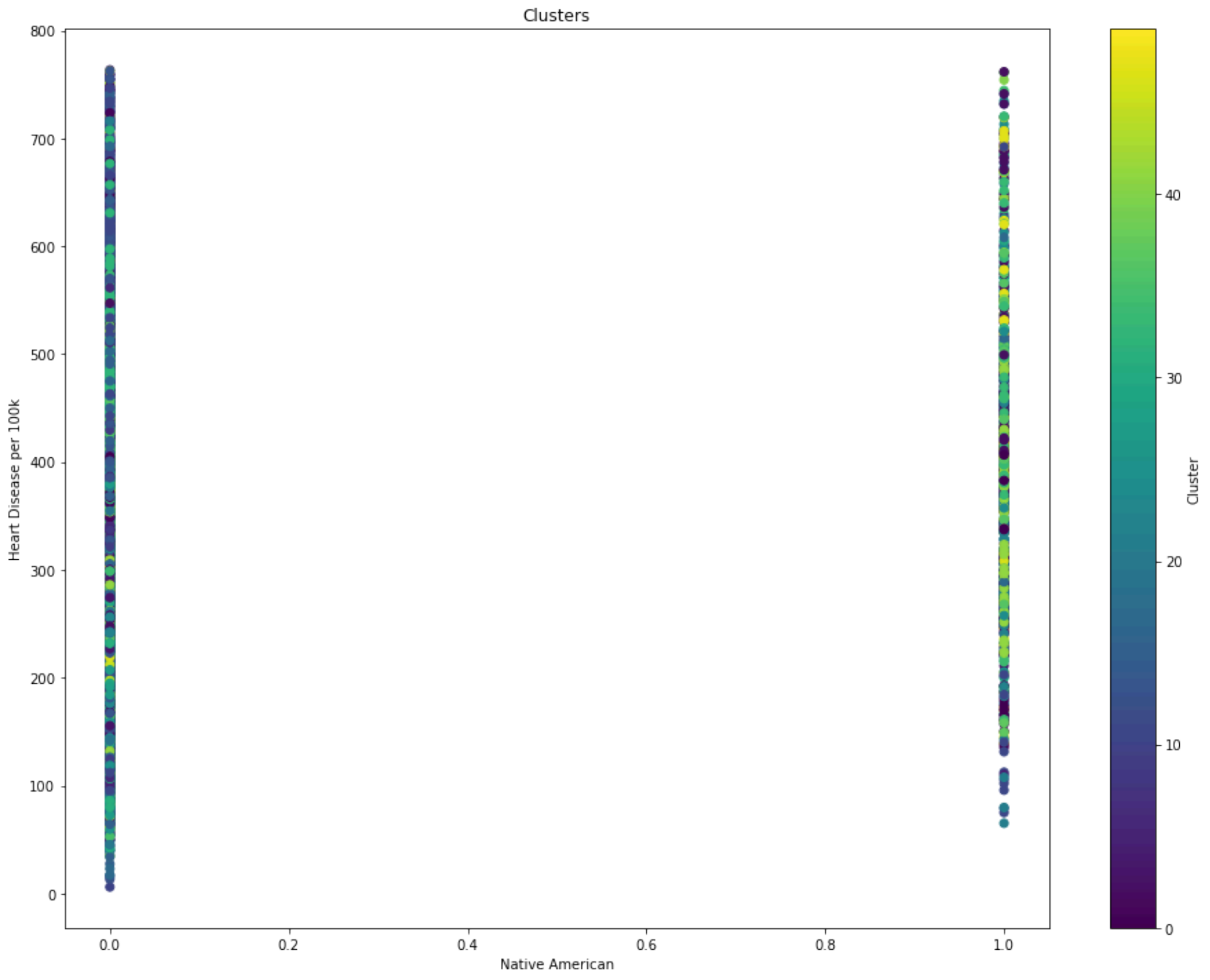
```
# Scatter plot for American Indian and Alaskan Native race
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['American Indian and Alaskan Native'], combined_cluster['Heart Disease per
plt.title('Clusters')
plt.xlabel('Native American')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()

# Scatter plot for Asian and Pacific Islander race
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['Asian and Pacific Islander'], combined_cluster['Heart Disease per 100k'], c=
plt.title('Clusters')
plt.xlabel('Asian')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()

# Scatter plot for State (Hawaii)
plt.figure(figsize=(16, 12))
plt.scatter(combined_cluster['State_HI'], combined_cluster['Heart Disease per 100k'], c=cluster_labels, c
plt.title('Clusters')
plt.xlabel('Hawaii')
plt.ylabel('Heart Disease per 100k')
plt.colorbar(label='Cluster')
plt.show()
```

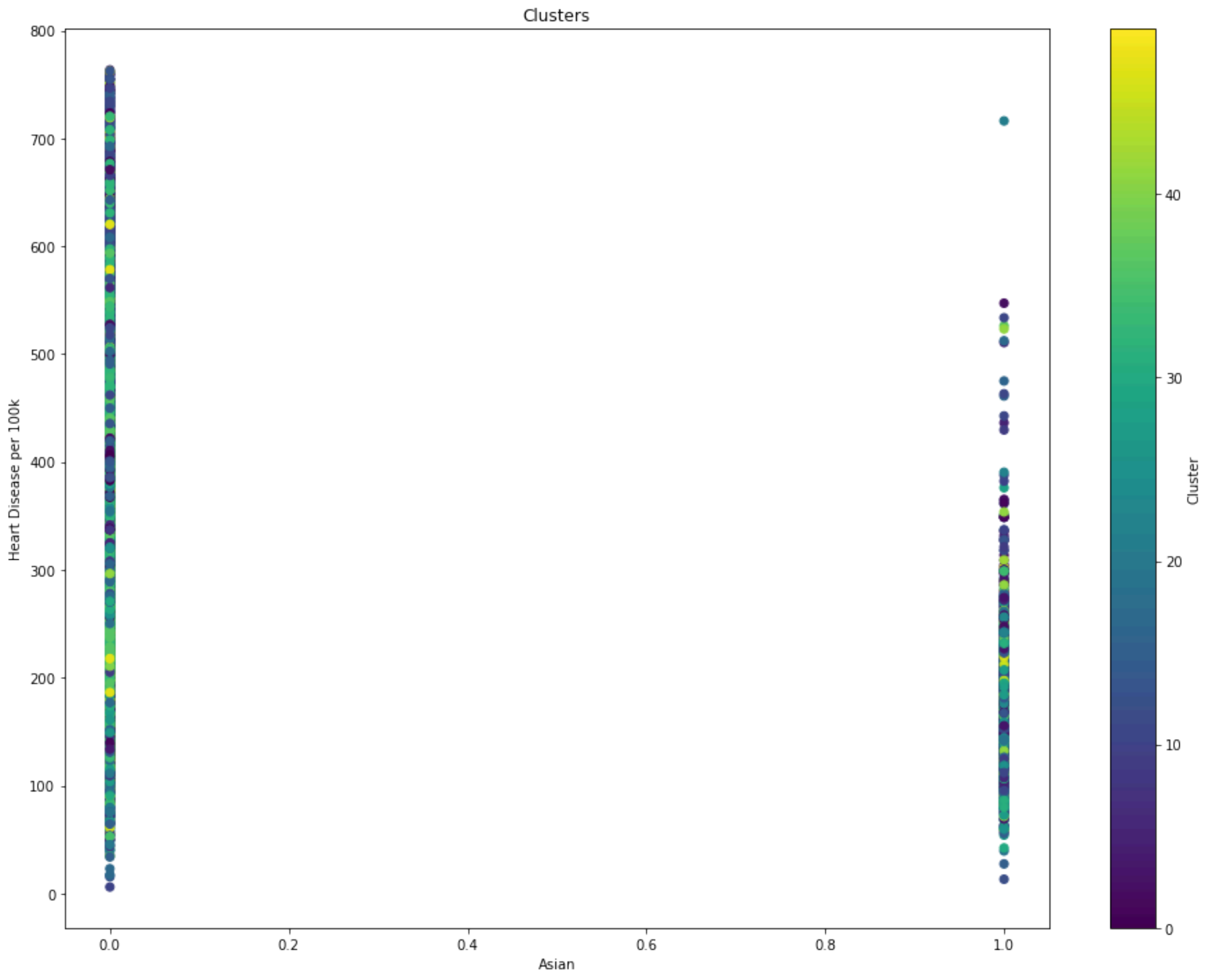


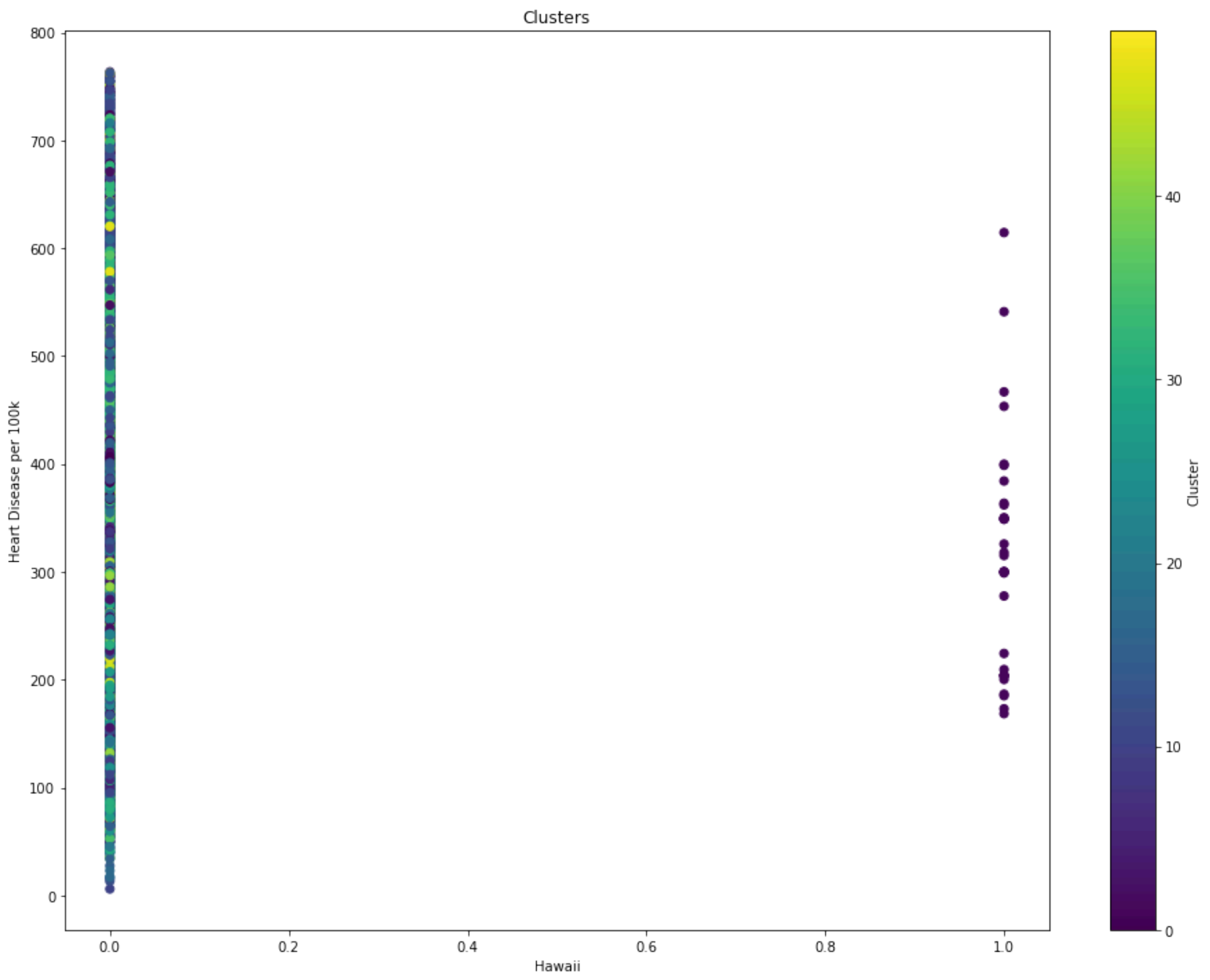

Clusters
Heart Disease per 100k
White
Cluster
0
100
200
300
400
500
600
700
800
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40


Clusters
Heart Disease per 100k
Black
Cluster
0
100
200
300
400
500
600
700
800
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40


Clusters
Heart Disease per 100k
0
100
200
300
400
500
600
700
800
Hispanic
0.0
0.2
0.4
0.6
0.8
1.0
Cluster
0
10
20
30
40


Clusters
Heart Disease per 100k
0
100
200
300
400
500
600
700
800
Native American
0.0
0.2
0.4
0.6
0.8
1.0
Cluster
0
10
20
30
40


Clusters
Heart Disease per 100k
Asian
Cluster
0
100
200
300
400
500
600
700
800
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40

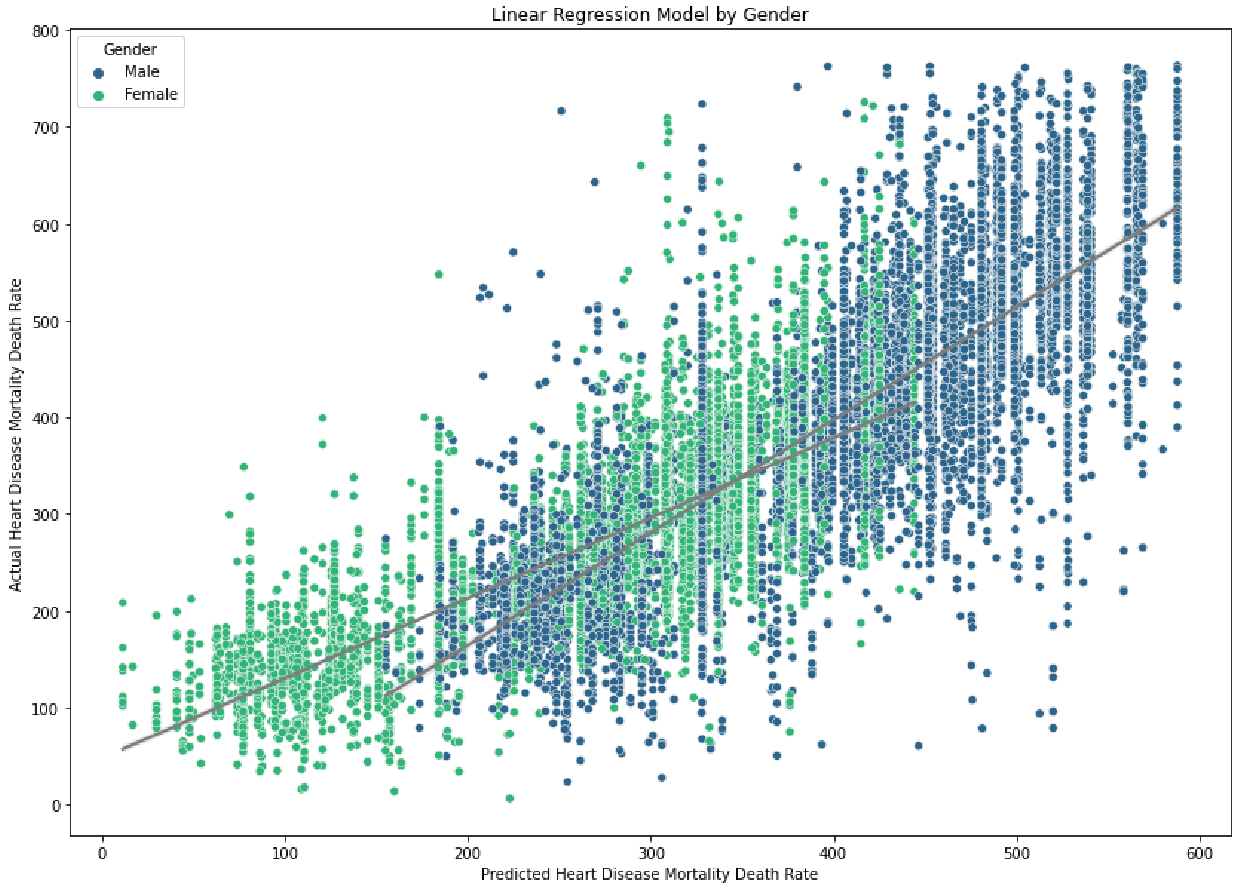


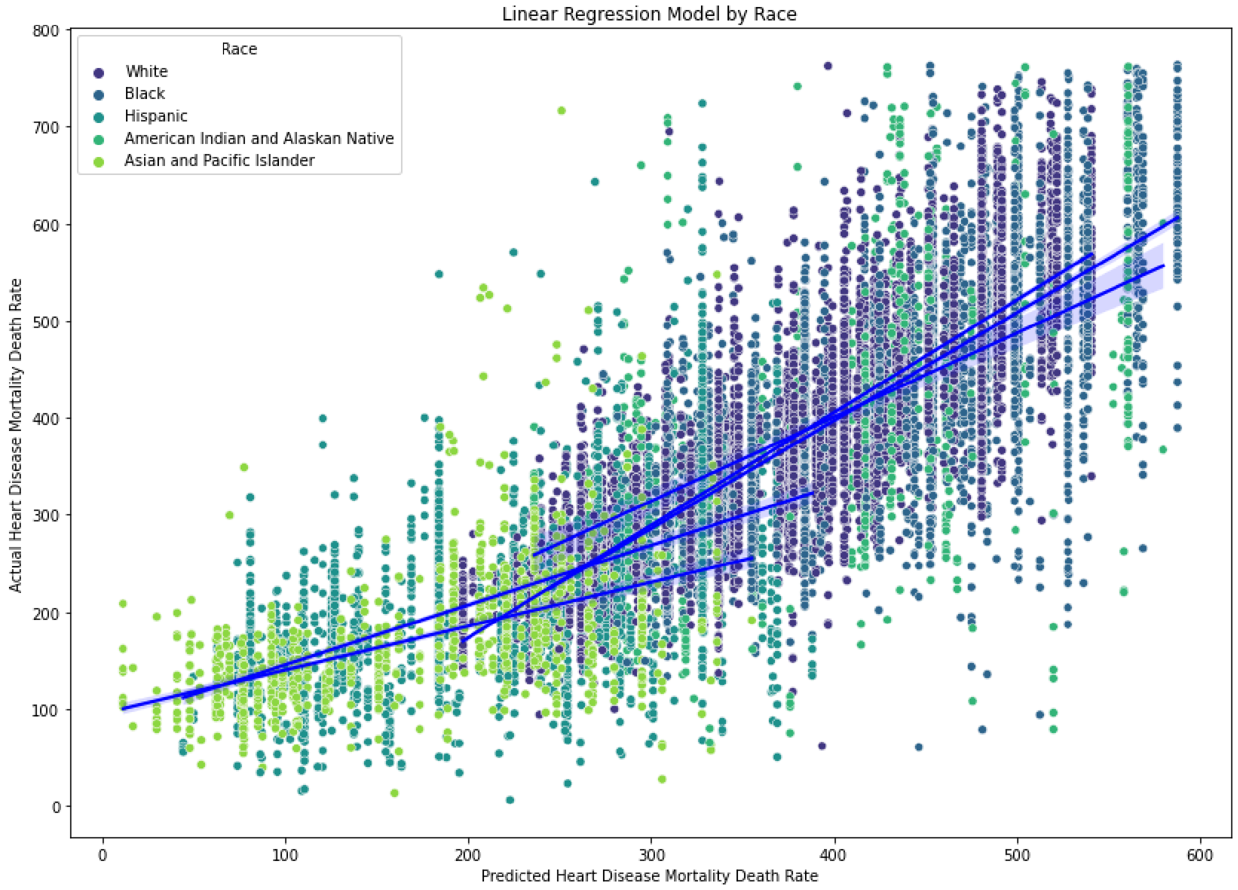
## Linear Regression Visual Modeling

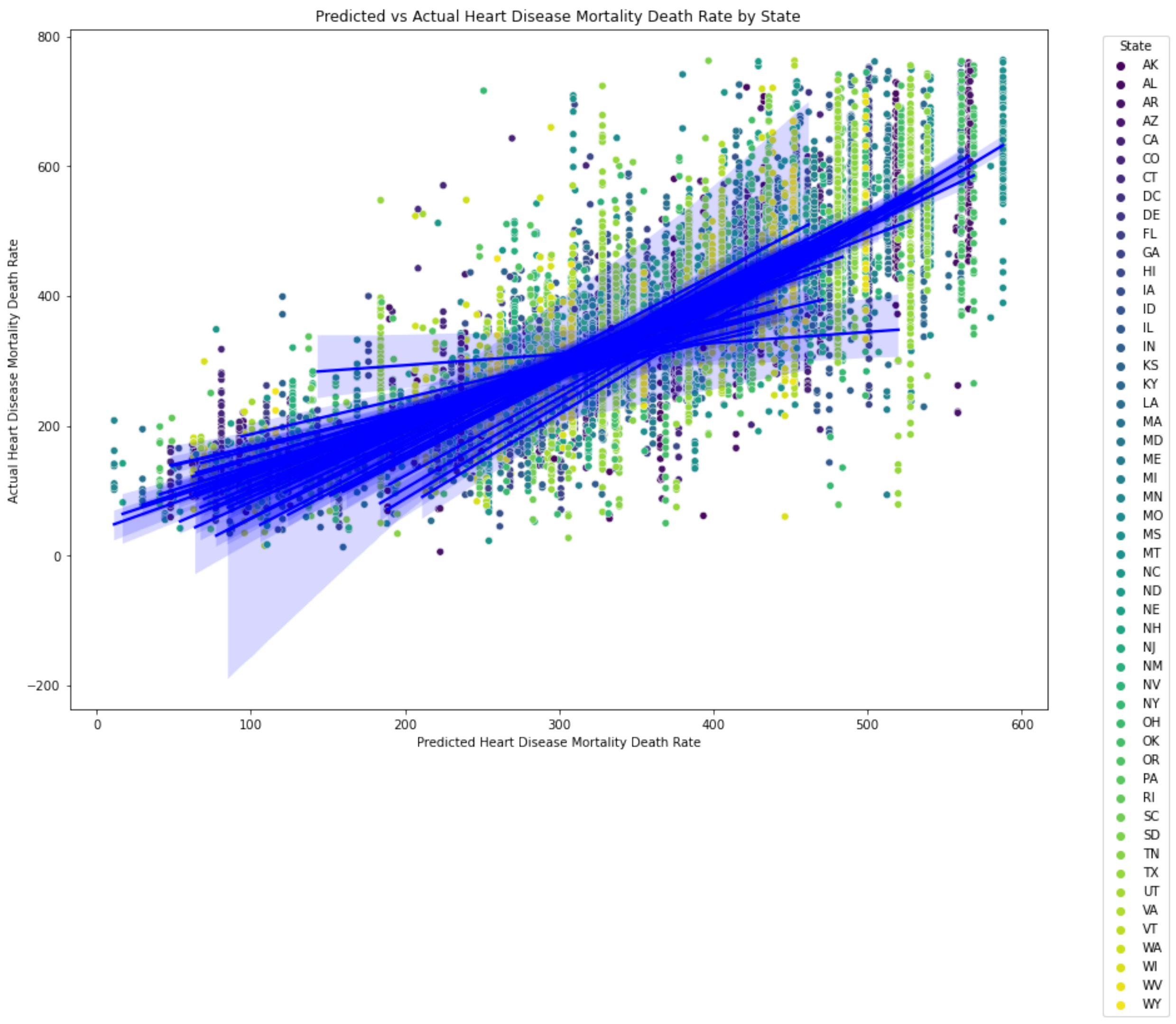
In [48]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

In [49]:
```
# Selecting relevant columns
X = cleaned_county_df[['Gender', 'Ethnicity', 'State']]
y = cleaned_county_df['Heart Disease per 100k']

# Define preprocessing steps for encoding categorical variables
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['Gender', 'Ethnicity', 'State'])  # One-hot encode categorical variables
    ],
    remainder='passthrough'  # Pass through any remaining columns
)

# Create a pipeline with preprocessing and linear regression model
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())  # Linear regression model
])

# Fit the pipeline on the data
```

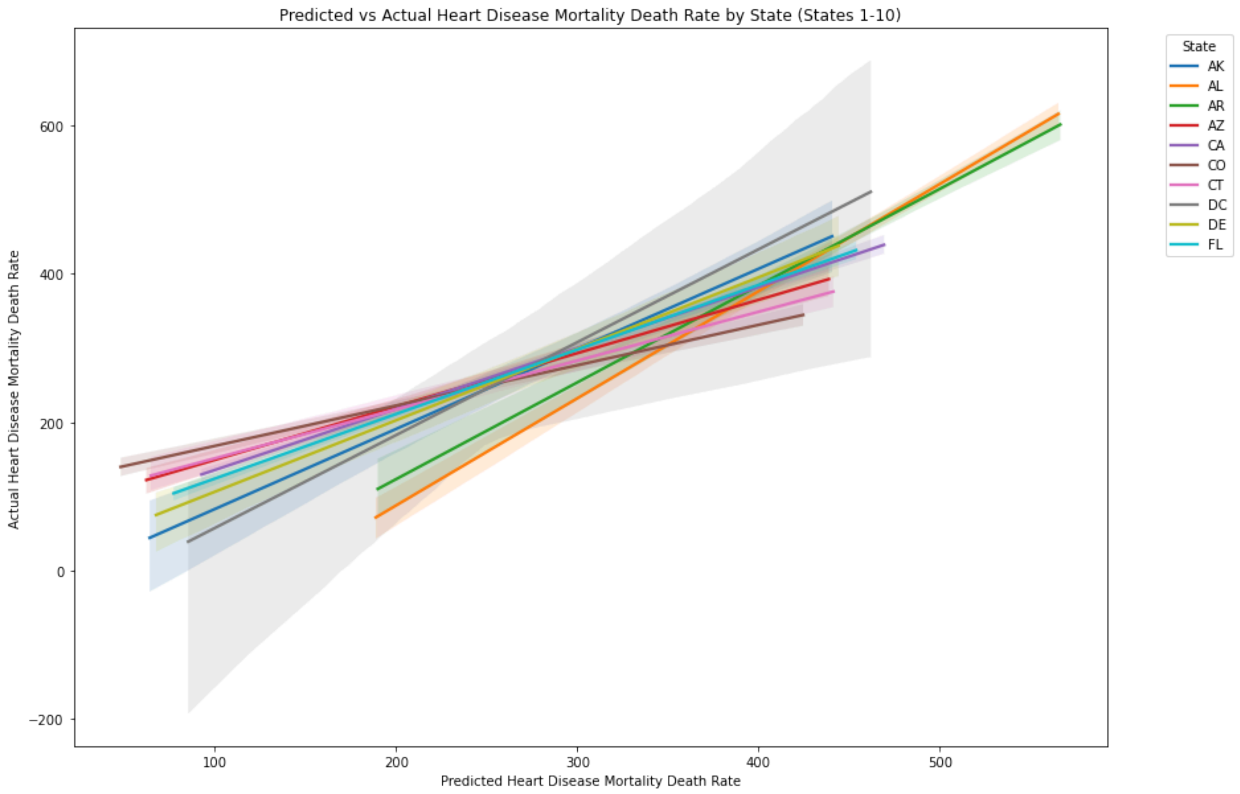
```python
pipeline.fit(X, y)

# Predict heart disease mortality death rate
y_pred = pipeline.predict(X)


# Plot for Gender
plt.figure(figsize=(14, 10))
sns.scatterplot(data=cleaned_county_df, x=y_pred, y=y, hue='Gender', palette='viridis', legend='full')
for category in cleaned_county_df['Gender'].unique():
    category_mask = (cleaned_county_df['Gender'] == category)
    sns.regplot(x=y_pred[category_mask], y=y[category_mask], scatter=False, color='gray')
plt.xlabel('Predicted Heart Disease Mortality Death Rate')
plt.ylabel('Actual Heart Disease Mortality Death Rate')
plt.title('Linear Regression Model by Gender')
plt.legend(title='Gender')
plt.show()

# Plot for Race
plt.figure(figsize=(14, 10))
sns.scatterplot(data=cleaned_county_df, x=y_pred, y=y, hue='Ethnicity', palette='viridis', legend='full')
for category in cleaned_county_df['Ethnicity'].unique():
    category_mask = (cleaned_county_df['Ethnicity'] == category)
    sns.regplot(x=y_pred[category_mask], y=y[category_mask], scatter=False, color='blue')
plt.xlabel('Predicted Heart Disease Mortality Death Rate')
plt.ylabel('Actual Heart Disease Mortality Death Rate')
plt.title('Linear Regression Model by Race')
plt.legend(title='Race')
plt.show()

# Plot for State
plt.figure(figsize=(14, 10))
sns.scatterplot(data=cleaned_county_df, x=y_pred, y=y, hue='State', palette='viridis')
for category in cleaned_county_df['State'].unique():
    category_mask = (cleaned_county_df['State'] == category)
    sns.regplot(x=y_pred[category_mask], y=y[category_mask], scatter=False, color='blue')
plt.xlabel('Predicted Heart Disease Mortality Death Rate')
plt.ylabel('Actual Heart Disease Mortality Death Rate')
plt.title('Predicted vs Actual Heart Disease Mortality Death Rate by State')
plt.legend(title='State', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

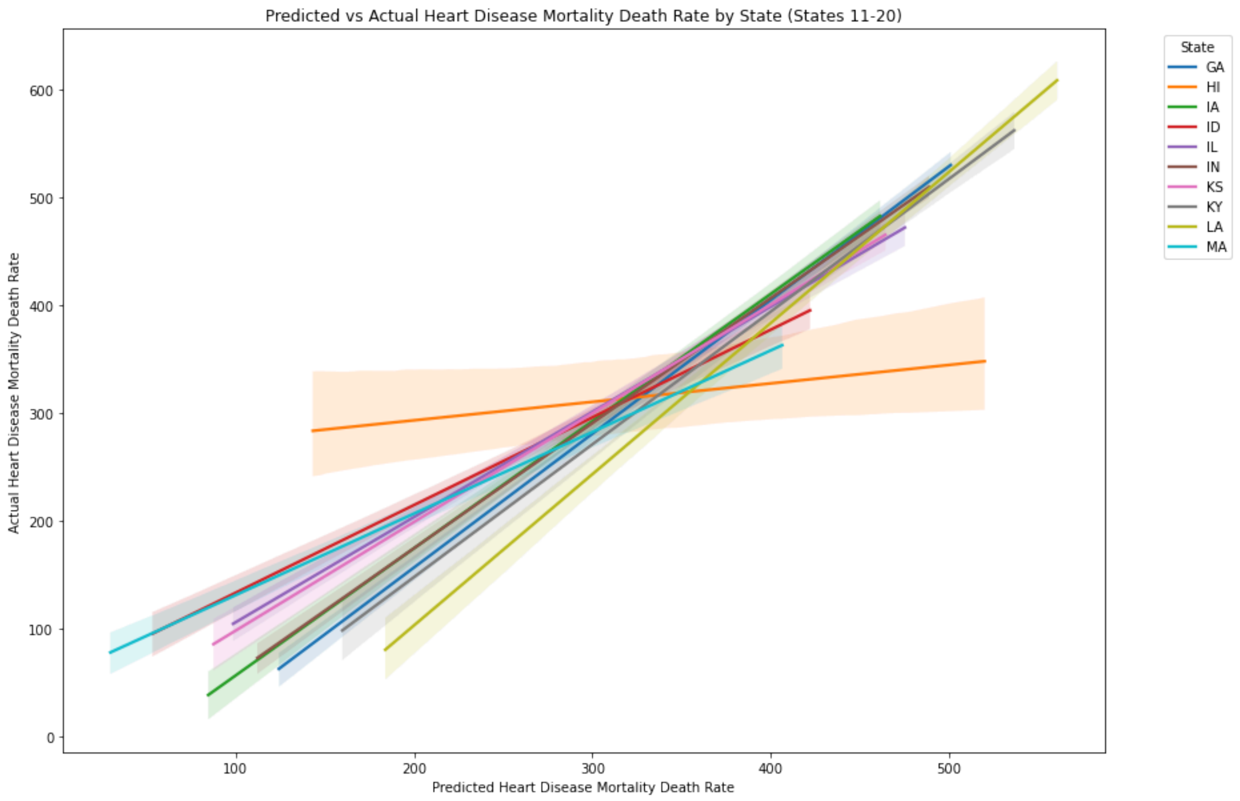

Linear Regression Model by Gender
Gender
Male
Female
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
0
100
200
300
400
500
600
700
800

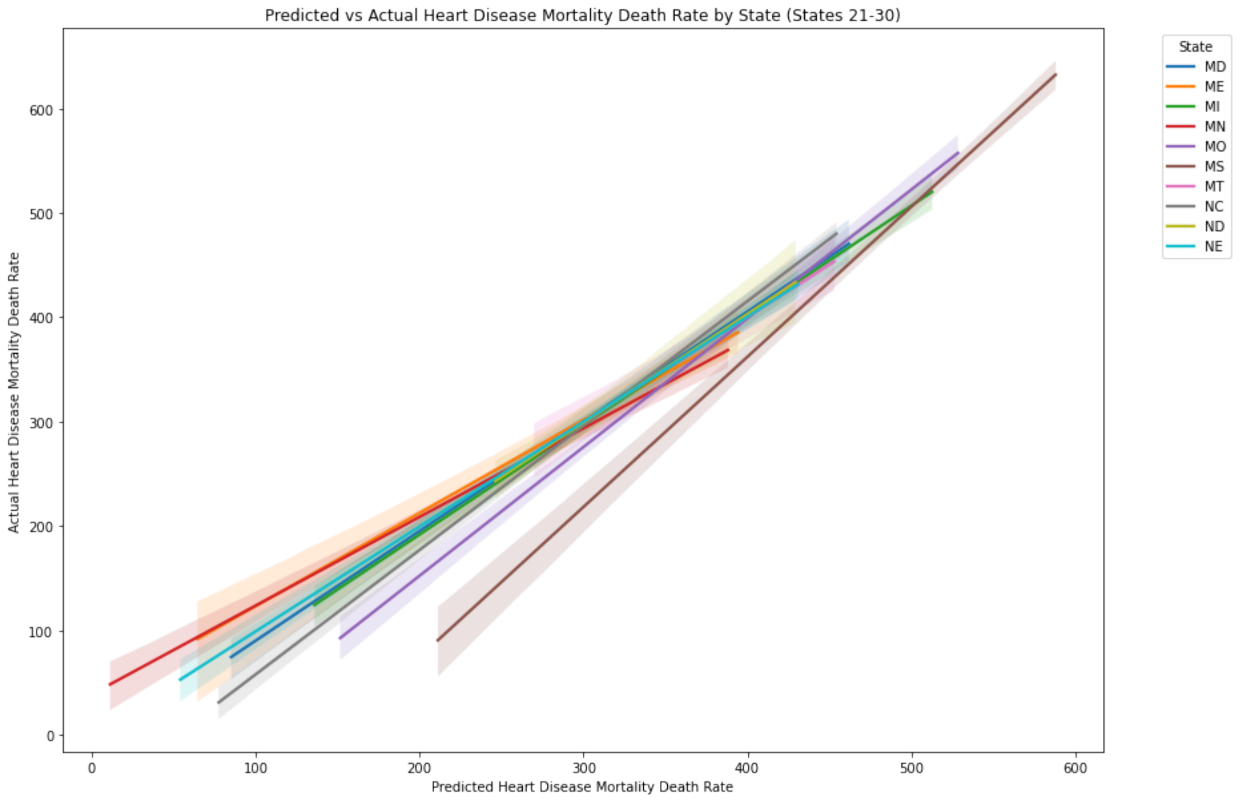

Linear Regression Model by Race
Race
White
Black
Hispanic
American Indian and Alaskan Native
Asian and Pacific Islander
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
0
100
200
300
400
500
600
700
800

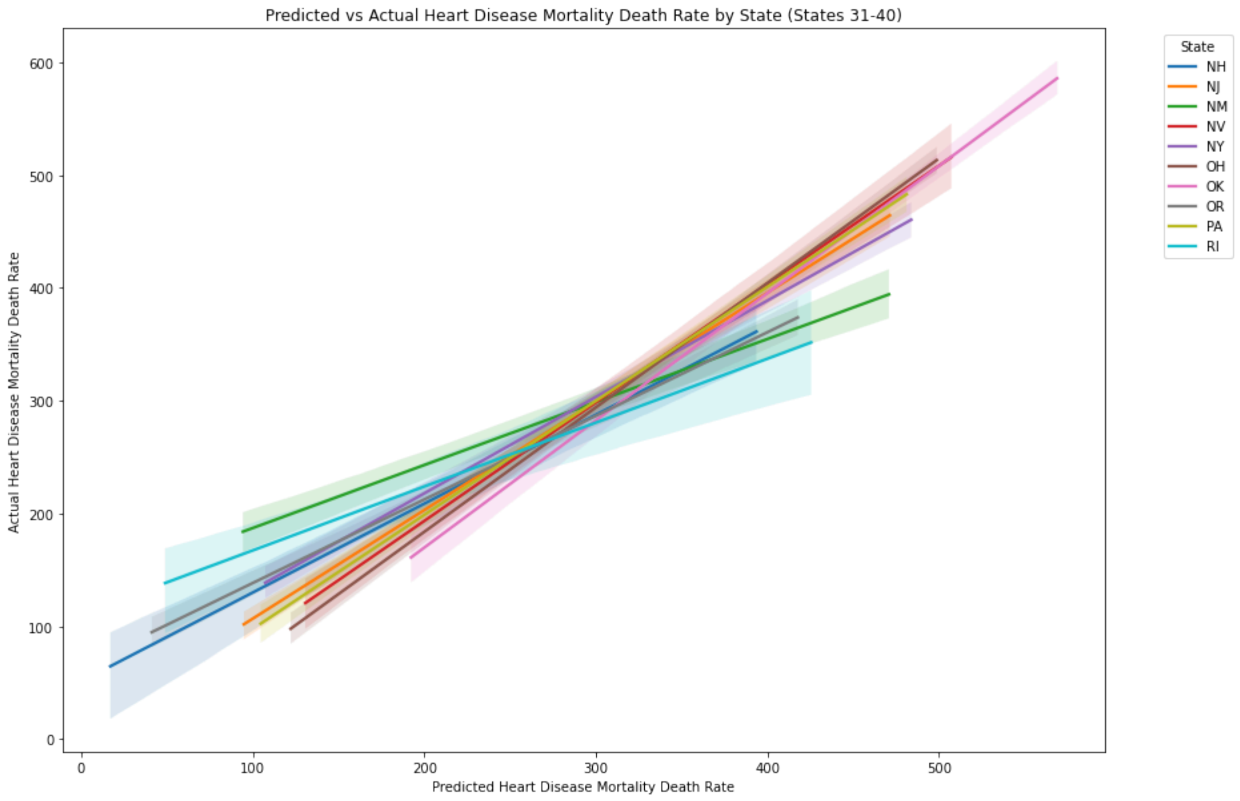


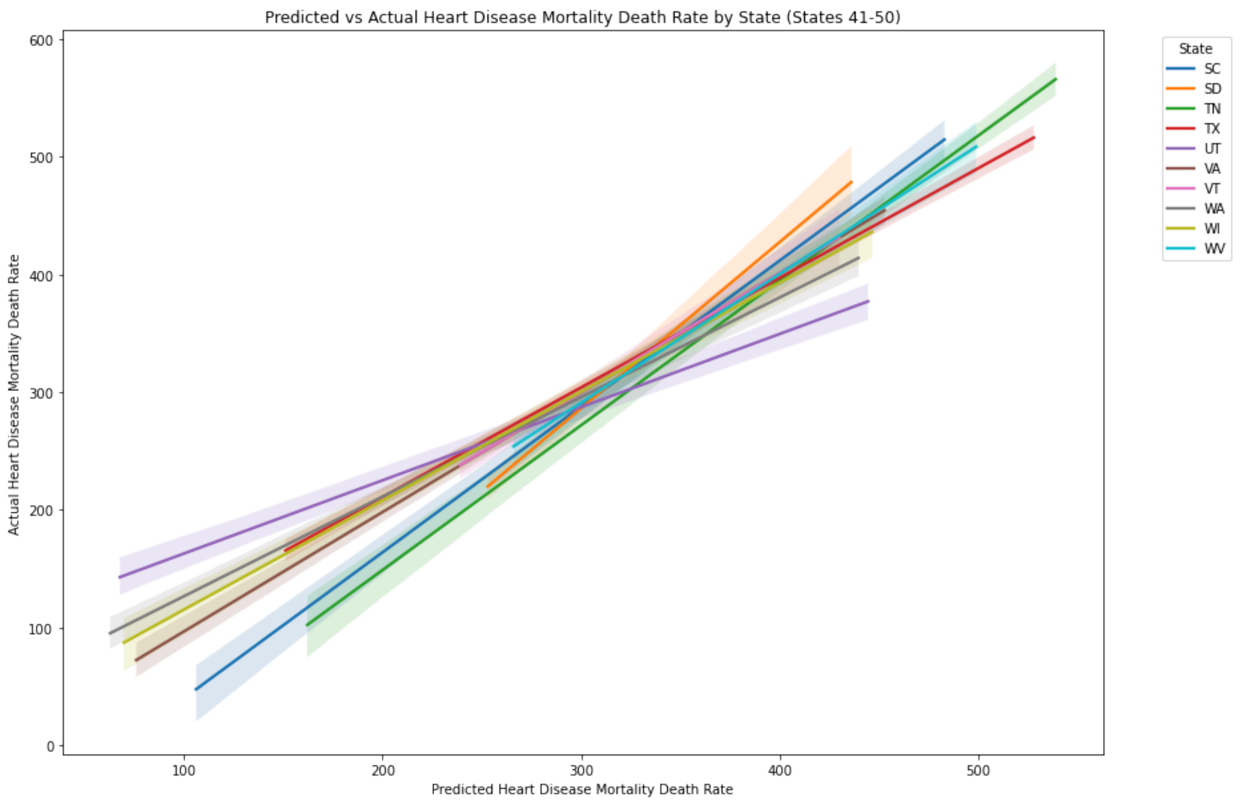
In [50]:
```
# Determine the number of groups (graphs) needed
num_states = len(cleaned_county_df['State'].unique())
num_groups = int(np.ceil(num_states / 10))  # Round up to the nearest integer

# Plot for each group of states
for i in range(num_groups):
    start_index = i * 10
    end_index = min((i + 1) * 10, num_states)  # Ensure not to exceed the number of states
    states_subset = list(cleaned_county_df['State'].unique())[start_index:end_index]

    plt.figure(figsize=(14, 10))
    for category in states_subset:
        category_mask = (cleaned_county_df['State'] == category)
        sns.regplot(x=y_pred[category_mask], y=y[category_mask], scatter=False, label=category)

    plt.xlabel('Predicted Heart Disease Mortality Death Rate')
    plt.ylabel('Actual Heart Disease Mortality Death Rate')
    plt.title(f'Predicted vs Actual Heart Disease Mortality Death Rate by State (States {start_index+1}-{e
    plt.legend(title='State', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.show()

    # Assumption for Hawaii. It is the amount of data collected (seen in clustering) and
    # assuming the population is majority Asian
```

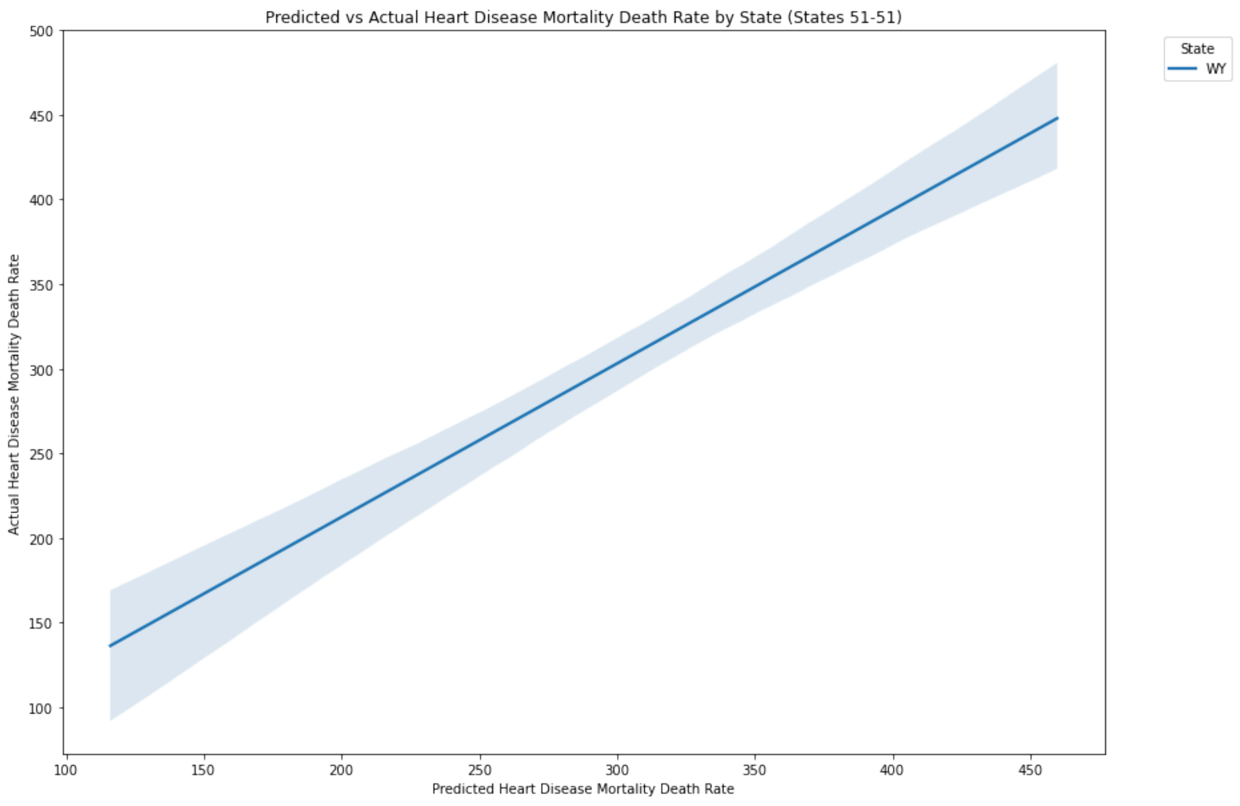

Predicted vs Actual Heart Disease Mortality Death Rate by State (States 1-10)
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
600
400
200
0
−200
100
200
300
400
500
State
AK
AL
AR
AZ
CA
CO
CT
DC
DE
FL


Predicted vs Actual Heart Disease Mortality Death Rate by State (States 11-20)
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
600
500
400
300
200
100
0
100
200
300
400
500
State
GA
HI
IA
ID
IL
IN
KS
KY
LA
MA


Predicted vs Actual Heart Disease Mortality Death Rate by State (States 21-30)
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
0
100
200
300
400
500
600
State
MD
ME
MI
MN
MO
MS
MT
NC
ND
NE


Predicted vs Actual Heart Disease Mortality Death Rate by State (States 31-40)
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
0
100
200
300
400
500
600
State
NH
NJ
NM
NV
NY
OH
OK
OR
PA
RI


Predicted vs Actual Heart Disease Mortality Death Rate by State (States 41-50)
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
0
100
200
300
400
500
600
100
200
300
400
500
State
SC
SD
TN
TX
UT
VA
VT
WA
WI
WV


Predicted vs Actual Heart Disease Mortality Death Rate by State (States 51-51)
Actual Heart Disease Mortality Death Rate
Predicted Heart Disease Mortality Death Rate
100
150
200
250
300
350
400
450
500
100
150
200
250
300
350
400
450
State
WY