Title: What are the reasons behind the disparity in health insurance coverage among demographic groups and geographic regions in the United States?

Authors : Aakanksha Maheshwari, Mansi Gopani, Het Trivedi (Group 7B)

Motivation and Background

In the USA, access to health insurance coverage not only determines financial security, it is the ability of individuals to access health care services. In the healthcare sector, we see continuous efforts to make healthcare accessible to every individual but despite the efforts, we can persistently see disparities across different geographic regions & demographic groups based on their age and income. These disparities have the most implications on people's health outcomes and well-being. We aim to identify those disparities and find some factors that influenced the most.

Importance of Addressing Disparities

The disparities will show the inequalities within the healthcare system and understanding the root cause behind these disparities is crucial for government and policymakers to reduce them. Analysing the data and contribution of different factors will be important in healthcare sectors to tailor their efforts and reach out to underserved populations more effectively.

Relevance in the Current Context

The COVID-19 pandemic has highlighted the importance of healthcare access and the pandemic has majorly affected communities with limited access to healthcare and since then the condition has been deteriorating. As the nation realized the importance and fighting the pandemic with limited facilities, now it is alarming to build a more resilient healthcare infrastructure that addresses the factors causing the disparities in health insurance coverage. Therefore, in the future, we can be better prepared for any such pandemic.

Summary of Research Questions & Result :-

Main Research Question

What are the reasons behind the disparity in health insurance coverage among demographic groups and geographic regions in the United States?

The data showed that racial, income, and sexual categories have a major impact on the number of people without health insurance. More people lack insurance, namely those at or below 400% of the poverty line, men, and members of specific racial or ethnic groups, such Hispanic or Latino people. Addressing disparities in health insurance coverage requires multifaceted interventions that consider both demographic and geographic dimensions.

1.What are the key factors affecting uninsured counts using Regression Model?

Compute:- Identify key factors influencing uninsured counts using Regression Model.

Why?:- Understanding these factors can provide insights into the root causes of disparities in health insurance coverage.

Answer:- Race, Income & Sex category were the most significant variables affecting uninsured counts.

2.How does the uninsured population vary among U.S. states, specifically focusing on the race , age, income and gender category?

Compute:- Analyze uninsured population variation across states, with a focus on demographic categories.

Why?:- This analysis helps pinpoint which demographic groups are most affected by lack of insurance coverage in different states.

Answer:- a.Individuals at or below 400% of the poverty level have the highest uninsured counts. b.Males have higher uninsured counts compared to females. c.Hispanic or Latino individuals show lower rates of insurance coverage. d.Individuals aged 21 to 64 years have the highest uninsured counts.

3)How does the relationship between income and uninsured counts vary across different age categories?

Compute:- Examine the relationship between income and uninsured counts across age groups.

Why?:- Understanding how income impacts uninsured counts across different age categories can inform targeted interventions.

Answer:- Lower uninsured counts are observed among income levels at or below 138% of the poverty line in every age group.

Dataset Description

The Small Area Health Insurance Estimates (SAHIE) dataset is taken from the official site of the United States Census Bureau. It provides valuable information about health insurance coverage in different areas of the United States. This dataset contains over 30,000 rows and 23 columns.

URL of the Dataset : https://www.census.gov/data/datasets/time-series/demo/sahie/estimates-acs.htm

The proposed project aims to leverage the SAHIE dataset for the year 2021 to conduct a comprehensive analysis of health insurance coverage disparities. By cleaning and transforming the dataset we wanted to use the data for meaningful analysis. We have followed the below steps to perform our analysis.

1. Examined the dataset's form and the first and last few rows. It provides us with an overview of the dataset's rows and columns.
2. We checked the data type or information in the columns and subsequently changed some of them to be of the numerical kind.
3. To improve comprehension, we have now modified the names of the columns.
4. Looked for any missing data and entered 0 in place of them because using median or mean values in those specific columns is inappropriate.
5. Remove the leading and following white spaces from the column labeled "county name" and count the instances of each value in the column labeled "county." In cases where the county code is zero, replace the county name with "not defined.
6. Two columns, year and version, were eliminated since they were not relevant to the analysis.
7. Following data transformation and cleansing, we got to work on our exploratory analysis. To determine which variables are relevant for the investigation, we have employed a correlational matrix of numerical values.
8. To better comprehend the situation, we conducted an exploratory study in which we first looked at the graphs before looking at the research questions. (This is attached, and the file ends with it.)
9. We used the XG boost model to determine each variable's significance. As an alternative, we have used linear regression to determine the relative importance of the variables.
10. Next, we grouped by state code and added the total count for income category 0 for our primary study analysis. Added the names of the states to the aggregated data, identified the Top 5 and Bottom 5 states, and then utilized a heatmap. For each of the seven racial groupings we had, we have these heat maps.
11. In a similar vein, we use the heat map to identify the Uninsured Counts for the Highest and Lowest 5 States with Age Categories 1 to 5, as well as the lowest 5 States by classifying them based on income and gender (sex) categories.

Results :-

1)What are the key factors affecting uninsured counts using Regression Model?

The model exhibits a 74% accuracy, indicating that a significant portion of the variation in uninsurance count can be explained by the other variables included in the model. **Feature Importance:** Race Category, Income Category, Gender and Age Category. When you predict these

variables, that's how they hold their importance.

a)Every additional race category increases the uninsured count by approximately 619 units:- This indicates that the uninsured count is predicted to rise by roughly 619 units for every extra racial category included in the analysis. It implies that different racial groups might have differing degrees of access to health insurance, and that the number of uninsured people may rise with the inclusion of each new racial group.

b) Every additional income category increases the uninsured count by approximately 410 units:- Approximately 410 units are added to the uninsured population for each additional income category. This means that the number of uninsured people is predicted to increase by about 410 units for every extra income category considered. It suggests that the number of people without health insurance is impacted by changes in income levels across categories, with higher income levels being linked to lower rates of uninsured people.

c) Every additional age category increases the uninsured count by approximately 123 units:- This suggests that for each additional age category included in the analysis, the uninsured count increases by approximately 123 units. It implies that different age groups may have varying rates of health insurance coverage, with each additional age category contributing to a higher uninsured count.

d)Every additional sex category decreases the uninsured count by approximately 139 units:- This statement indicates that for each additional sex category considered (e.g., male and female), the uninsured count is expected to decrease by approximately 139 units. It suggests that there may be differences in health insurance coverage between sexes, with each additional sex category contributing to a reduction in the uninsured count.

2.How does the uninsured population vary among U.S. states, specifically focusing on the race , age, income and gender category?

Texas has a significant population of undocumented immigrants who might not buy insurance. Low wages and high living expenses could be reasons for this analysis. Moreover, limited availability of healthcare providers, particularly in rural regions, compounds the issue by making it difficult for individuals to access necessary care. At or below 400% of the poverty level, we interpret the highest uninsured counts in all age categories.

3.How does the relationship between income and uninsured counts vary across different age categories?

Lower uninsured counts are observed among income levels at or below 138% of the poverty line in every age group.

The most surprising analysis we got was that almost in every analysis, Texas , Florida and California respectively, exhibit the highest numbers of uninsured individuals. Notably, these states are coastal regions adjacent to oceans. The cost of insurance premiums can

be notably high, particularly in areas susceptible to natural calamities such as wildfires, and floods. We feel external factors like type of diseases, climate changes, amount covered, or areas in these locations would be helpful to find more insights.

Reflection:-

We now know how to use several models, libraries, and visualization tools in conjunction with the Jupyter notebook. Above all, we now know how to do exploratory analysis prior to answering the research questions.Thanks to Professor, Angela Usha Ramnarine-Rieks who introduced us to various concepts & its application.

Additionally, we believe that our study might benefit from a slightly larger dataset in columns or additional details about the individuals. Our analysis would provide additional depth and insights if the data could hold more information.

Work Done by Teammates

Aakanksha Maheshwari:- Data cleaning & transforming,Modelling ,Visualization, Writing Report & Analysis

Mansi Gopani:- Exploratory analysis, Project Proposal, Research question 2,Visualization, Writing Report & Analysis

Het Trivedi:- Data Selection, Project Progress reports, Research Question 3, Poster Presentation, Visualization

In [105…
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install scikit-learn
!pip install statsmodels
!pip install xgboost
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
```

In [106…
```
# Read the CSV file, skipping the first few rows
df = pd.read_csv('sahie_2021.csv', skiprows=83)
```

```
/tmp/ipykernel_226/1148478942.py:2: DtypeWarning: Columns (9,10,11,12,13,14,15,16,17,18,19,20,21,22) have mixed types. Specify
dtype option on import or set low_memory=False.
  df = pd.read_csv('sahie_2021.csv', skiprows=83)
```

In [107…

```
# Display the shape of the DataFrame
df.shape
print("Shape of the DataFrame:", df.shape)
```

```
Shape of the DataFrame: (338754, 25)
```

There are 338754 rows & 25 columns.

In [108…

```
# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())
```

```
First few rows of the dataset:
   year   version  statefips  countyfips  geocat  agecat  racecat  sexcat  \
0  2021                    1           0      40       0        0       0
1  2021                    1           0      40       0        0       0
2  2021                    1           0      40       0        0       0
3  2021                    1           0      40       0        0       0
4  2021                    1           0      40       0        0       0

   iprcat     NIPR  ... PCTUI pctui_moe PCTIC pctic_moe PCTELIG pctelig_moe  \
0       0  4018412  ...  11.7       0.4  88.3       0.4    11.7         0.4
1       1  1418268  ...  19.0       0.7  81.0       0.7     6.7         0.3
2       2  1763104  ...  18.1       0.6  81.9       0.6     7.9         0.3
3       3   970079  ...  20.0       0.8  80.0       0.8     4.8         0.2
4       4  2666721  ...  15.3       0.5  84.7       0.5    10.1         0.3

  PCTLIIC pctliic_moe                                       state_name  \
0    88.3         0.4  Alabama                                     ...
1    28.6         0.4  Alabama                                     ...
2    35.9         0.4  Alabama                                     ...
3    19.3         0.3  Alabama                                     ...
4    56.2         0.5  Alabama                                     ...

                                  county_name
0
1
2
3
4

[5 rows x 25 columns]
```

In [109…

```
# Display the last few rows of the dataset
df.tail()
```

Out[109…

	year	version	statefips	countyfips	geocat	agecat	racecat	sexcat	iprcat	NIPR	…	PCTUI	pctui_moe	PCTIC	pctic_moe	PC
338749	2021		56	45	50	5	0	2	1	408	…	29.4	6.9	70.6	6.9	
338750	2021		56	45	50	5	0	2	2	534	…	28.1	6.2	71.9	6.2	
338751	2021		56	45	50	5	0	2	3	271	…	31.7	7.9	68.3	7.9	
338752	2021		56	45	50	5	0	2	4	915	…	23.5	4.7	76.5	4.7	
338753	2021		56	45	50	5	0	2	5	644	…	20.0	4.0	80.0	4.0	

5 rows × 25 columns

In [110…

```
# Get information about the dataset
print("\nInformation about the dataset:")
print(df.info())
```

```
Information about the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 338754 entries, 0 to 338753
Data columns (total 25 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   year         338754 non-null  int64 
 1   version      338754 non-null  object
 2   statefips    338754 non-null  int64 
 3   countyfips   338754 non-null  int64 
 4   geocat       338754 non-null  int64 
 5   agecat       338754 non-null  int64 
 6   racecat      338754 non-null  int64 
 7   sexcat       338754 non-null  int64 
 8   iprcat       338754 non-null  int64 
 9   NIPR         338754 non-null  object
 10  nipr_moe     338754 non-null  object
 11  NUI          338754 non-null  object
 12  nui_moe      338754 non-null  object
 13  NIC          338754 non-null  object
 14  nic_moe      338754 non-null  object
 15  PCTUI        338754 non-null  object
 16  pctui_moe    338754 non-null  object
 17  PCTIC        338754 non-null  object
 18  pctic_moe    338754 non-null  object
 19  PCTELIG      338754 non-null  object
 20  pctelig_moe  338754 non-null  object
 21  PCTLIIC      338754 non-null  object
 22  pctliic_moe  338754 non-null  object
 23  state_name   338754 non-null  object
 24  county_name  338754 non-null  object
dtypes: int64(8), object(17)
memory usage: 64.6+ MB
None
```

There are total 17 objects. Now we're converting 15 objects to numerical values.

Method 1: Data type Conversion

In [111…

```
# Convert specified columns to numeric data types, replacing non-numeric values with NaN

df['NIPR'] = pd.to_numeric(df['NIPR'], errors = 'coerce')
df['nipr_moe'] = pd.to_numeric(df['nipr_moe'], errors='coerce')
df['NUI'] = pd.to_numeric(df['NUI'], errors='coerce')
df['nui_moe'] = pd.to_numeric(df['nui_moe'], errors='coerce')
df['NIC'] = pd.to_numeric(df['NIC'], errors='coerce')
df['nic_moe'] = pd.to_numeric(df['nic_moe'], errors='coerce')
df['PCTUI'] = pd.to_numeric(df['PCTUI'], errors='coerce')
df['pctui_moe'] = pd.to_numeric(df['pctui_moe'], errors='coerce')
df['PCTIC'] = pd.to_numeric(df['PCTIC'], errors='coerce')
df['pctic_moe'] = pd.to_numeric(df['pctic_moe'], errors='coerce')
df['PCTELIG'] = pd.to_numeric(df['PCTELIG'], errors='coerce')
df['pctelig_moe'] = pd.to_numeric(df['pctelig_moe'], errors='coerce')
df['PCTLIIC'] = pd.to_numeric(df['PCTLIIC'], errors='coerce')
df['pctliic_moe'] = pd.to_numeric(df['pctliic_moe'], errors='coerce')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 338754 entries, 0 to 338753
Data columns (total 25 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   year         338754 non-null  int64  
 1   version      338754 non-null  object 
 2   statefips    338754 non-null  int64  
 3   countyfips   338754 non-null  int64  
 4   geocat       338754 non-null  int64  
 5   agecat       338754 non-null  int64  
 6   racecat      338754 non-null  int64  
 7   sexcat       338754 non-null  int64  
 8   iprcat       338754 non-null  int64  
 9   NIPR         338658 non-null  float64
 10  nipr_moe     338658 non-null  float64
 11  NUI          338658 non-null  float64
 12  nui_moe      338658 non-null  float64
 13  NIC          338658 non-null  float64
 14  nic_moe      338658 non-null  float64
 15  PCTUI        338648 non-null  float64
 16  pctui_moe    338648 non-null  float64
 17  PCTIC        338648 non-null  float64
 18  pctic_moe    338648 non-null  float64
 19  PCTELIG      338658 non-null  float64
 20  pctelig_moe  338658 non-null  float64
 21  PCTLIIC      338658 non-null  float64
 22  pctliic_moe  338658 non-null  float64
 23  state_name   338754 non-null  object 
 24  county_name  338754 non-null  object 
dtypes: float64(14), int64(8), object(3)
memory usage: 64.6+ MB
```

Method 2: Changing column names

In [112…

```python
# Changing column names in DataFrame
df.rename(columns={'statefips': 'state_code'}, inplace=True)
df.rename(columns={'countyfips': 'county_code'}, inplace=True)
df.rename(columns={'iprcat': 'income_category'}, inplace=True)
df.rename(columns={'NIPR': 'group_size'}, inplace=True)
```

```python
df.rename(columns={'nipr_moe': 'group_size_error'}, inplace=True)
df.rename(columns={'NUI': 'uninsured_count'}, inplace=True)
df.rename(columns={'nui_moe': 'uninsured_count_error'}, inplace=True)
df.rename(columns={'NIC': 'insured_count'}, inplace=True)
df.rename(columns={'nic_moe': 'insured_count_error'}, inplace=True)
df.rename(columns={'PCTUI': '%_uninsured_count'}, inplace=True)
df.rename(columns={'pctui_moe': '%_uninsured_count_error'}, inplace=True)
df.rename(columns={'PCTIC': '%_insured_count'}, inplace=True)
df.rename(columns={'pctic_moe': '%_insured_count_error'}, inplace=True)
df.rename(columns={'PCTELIG': '%_uninsured_count_all_income'}, inplace=True)
df.rename(columns={'pctelig_moe': '%_uninsured_count_all_income_error'}, inplace=True)
df.rename(columns={'PCTLIIC': '%_insured_count_all_income'}, inplace=True)
df.rename(columns={'pctliic_moe': '%_insured_count_all_income_error'}, inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 338754 entries, 0 to 338753
Data columns (total 25 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   year                                  338754 non-null  int64
 1   version                               338754 non-null  object
 2   state_code                            338754 non-null  int64
 3   county_code                           338754 non-null  int64
 4   geocat                                338754 non-null  int64
 5   agecat                                338754 non-null  int64
 6   racecat                               338754 non-null  int64
 7   sexcat                                338754 non-null  int64
 8   income_category                       338754 non-null  int64
 9   group_size                            338658 non-null  float64
 10  group_size_error                      338658 non-null  float64
 11  uninsured_count                       338658 non-null  float64
 12  uninsured_count_error                 338658 non-null  float64
 13  insured_count                         338658 non-null  float64
 14  insured_count_error                   338658 non-null  float64
 15  %_uninsured_count                     338648 non-null  float64
 16  %_uninsured_count_error               338648 non-null  float64
 17  %_insured_count                       338648 non-null  float64
 18  %_insured_count_error                 338648 non-null  float64
 19  %_uninsured_count_all_income          338658 non-null  float64
 20  %_uninsured_count_all_income_error    338658 non-null  float64
 21  %_insured_count_all_income            338658 non-null  float64
 22  %_insured_count_all_income_error      338658 non-null  float64
 23  state_name                            338754 non-null  object
 24  county_name                           338754 non-null  object
dtypes: float64(14), int64(8), object(3)
memory usage: 64.6+ MB
```

Method 3: Checking Missing values in the dataset

In [113…

```python
# Check for missing values
print("\nMissing values in the dataset:")
print(df.isna().sum())
```

```
Missing values in the dataset:
year                                    0
version                                 0
state_code                              0
county_code                             0
geocat                                  0
agecat                                  0
racecat                                 0
sexcat                                  0
income_category                         0
group_size                             96
group_size_error                       96
uninsured_count                        96
uninsured_count_error                  96
insured_count                          96
insured_count_error                    96
%_uninsured_count                     106
%_uninsured_count_error               106
%_insured_count                       106
%_insured_count_error                 106
%_uninsured_count_all_income           96
%_uninsured_count_all_income_error     96
%_insured_count_all_income             96
%_insured_count_all_income_error       96
state_name                              0
county_name                             0
dtype: int64
```

Method 4: Fill missing values in specified columns with 0

In [114…

```python
# Fill missing values in specified columns with 0

columns_to_fill = ['group_size', 'group_size_error', 'uninsured_count', 'uninsured_count_error', 'insured_count', 'insured_cou

df[columns_to_fill] = df[columns_to_fill].fillna(0)

print(df.isna().sum())
```

```
year                                0
version                             0
state_code                          0
county_code                         0
geocat                              0
agecat                              0
racecat                             0
sexcat                              0
income_category                     0
group_size                          0
group_size_error                    0
uninsured_count                     0
uninsured_count_error               0
insured_count                       0
insured_count_error                 0
%_uninsured_count                   0
%_uninsured_count_error             0
%_insured_count                     0
%_insured_count_error               0
%_uninsured_count_all_income        0
%_uninsured_count_all_income_error  0
%_insured_count_all_income          0
%_insured_count_all_income_error    0
state_name                          0
county_name                         0
dtype: int64
```

Method 5 : Replacing blank values in the category county_name with not defined

In [115…

```
# Strip leading and trailing white spaces from 'county_name' column
df['county_name'] = df['county_name'].str.strip()

# checking unique values in the 'county_name' column
print("Unique values in the 'county' column:")
print(df['county_name'].unique())
```

```
Unique values in the 'county' column:
['' 'Autauga County' 'Baldwin County' ... 'Uinta County' 'Washakie County'
 'Weston County']
```

In [116…

```
# Count occurrences of each value in the 'county' column
print("\nValue counts in the 'county' column:")
print(df['county_name'].value_counts())
```

```
Value counts in the 'county' column:
county_name
                          37026
Washington County          2880
Jefferson County           2400
Franklin County            2304
Lincoln County             2208
                          ...
Winn Parish                  96
West Feliciana Parish        96
West Carroll Parish          96
West Baton Rouge Parish      96
Weston County                96
Name: count, Length: 1879, dtype: int64
```

In [117…

```
# Replace county name with 'not defined' where county_code is 0
df.loc[df['county_code'] == 0, 'county_name'] = 'not defined'

# Display the updated DataFrame
print(df)
```

	year	version	state_code	county_code	geocat	agecat	racecat	\
0	2021		1	0	40	0	0	
1	2021		1	0	40	0	0	
2	2021		1	0	40	0	0	
3	2021		1	0	40	0	0	
4	2021		1	0	40	0	0	
...	...	...	...	...	...	...	...	
338749	2021		56	45	50	5	0	
338750	2021		56	45	50	5	0	
338751	2021		56	45	50	5	0	
338752	2021		56	45	50	5	0	
338753	2021		56	45	50	5	0	

	sexcat	income_category	group_size	...	%_uninsured_count	\
0	0	0	4018412.0	...	11.7	
1	0	1	1418268.0	...	19.0	
2	0	2	1763104.0	...	18.1	
3	0	3	970079.0	...	20.0	
4	0	4	2666721.0	...	15.3	
...	...	...	...	...	...	
338749	2	1	408.0	...	29.4	
338750	2	2	534.0	...	28.1	
338751	2	3	271.0	...	31.7	
338752	2	4	915.0	...	23.5	
338753	2	5	644.0	...	20.0	

	%_uninsured_count_error	%_insured_count	%_insured_count_error	\
0	0.4	88.3	0.4	
1	0.7	81.0	0.7	
2	0.6	81.9	0.6	
3	0.8	80.0	0.8	
4	0.5	84.7	0.5	
...	...	...	...	
338749	6.9	70.6	6.9	
338750	6.2	71.9	6.2	
338751	7.9	68.3	7.9	
338752	4.7	76.5	4.7	
338753	4.0	80.0	4.0	

	%_uninsured_count_all_income	%_uninsured_count_all_income_error	\
0	11.7	0.4	

```
1                              6.7                                0.3
2                              7.9                                0.3
3                              4.8                                0.2
4                             10.1                                0.3
...                            ...                                ...
338749                         7.3                                2.0
338750                         9.2                                2.3
338751                         5.3                                1.6
338752                        13.1                                2.8
338753                         7.9                                1.7

        %_insured_count_all_income  %_insured_count_all_income_error  \
0                             88.3                                0.4
1                             28.6                                0.4
2                             35.9                                0.4
3                             19.3                                0.3
4                             56.2                                0.5
...                            ...                                ...
338749                        17.6                                2.7
338750                        23.5                                3.1
338751                        11.3                                2.2
338752                        42.8                                3.9
338753                        31.5                                3.1

                                          state_name    county_name
0       Alabama                                  ...    not defined
1       Alabama                                  ...    not defined
2       Alabama                                  ...    not defined
3       Alabama                                  ...    not defined
4       Alabama                                  ...    not defined
...                                              ...            ...
338749  Wyoming                                  ...  Weston County
338750  Wyoming                                  ...  Weston County
338751  Wyoming                                  ...  Weston County
338752  Wyoming                                  ...  Weston County
338753  Wyoming                                  ...  Weston County

[338754 rows x 25 columns]
```

Method 6 : Drop Columns version and year

In [118…

```
# Drop the 'version' and 'year' columns from the DataFrame
df.drop(columns=['version', 'year'], inplace=True)

# Print the DataFrame to confirm the deletion
print(df)
```

```
        state_code  county_code  geocat  agecat  racecat  sexcat  \
0                1            0      40       0        0       0
1                1            0      40       0        0       0
2                1            0      40       0        0       0
3                1            0      40       0        0       0
4                1            0      40       0        0       0
...            ...          ...     ...     ...      ...     ...
338749          56           45      50       5        0       2
338750          56           45      50       5        0       2
338751          56           45      50       5        0       2
338752          56           45      50       5        0       2
338753          56           45      50       5        0       2

        income_category  group_size  group_size_error  uninsured_count  ...  \
0                     0   4018412.0               0.0         469887.0  ...
1                     1   1418268.0           15141.0         269331.0  ...
2                     2   1763104.0           15533.0         319016.0  ...
3                     3    970079.0           13885.0         193882.0  ...
4                     4   2666721.0           15752.0         406730.0  ...
...                 ...         ...               ...              ...  ...
338749                1       408.0              49.0            120.0  ...
338750                2       534.0              54.0            150.0  ...
338751                3       271.0              43.0             86.0  ...
338752                4       915.0              63.0            215.0  ...
338753                5       644.0              56.0            129.0  ...

        %_uninsured_count  %_uninsured_count_error  %_insured_count  \
0                    11.7                      0.4             88.3
1                    19.0                      0.7             81.0
2                    18.1                      0.6             81.9
3                    20.0                      0.8             80.0
4                    15.3                      0.5             84.7
...                   ...                      ...              ...
338749               29.4                      6.9             70.6
338750               28.1                      6.2             71.9
338751               31.7                      7.9             68.3
338752               23.5                      4.7             76.5
338753               20.0                      4.0             80.0

        %_insured_count_error  %_uninsured_count_all_income  \
0                         0.4                          11.7
```

```
1                       0.7                        6.7
2                       0.6                        7.9
3                       0.8                        4.8
4                       0.5                       10.1
...                     ...                        ...
338749                  6.9                        7.3
338750                  6.2                        9.2
338751                  7.9                        5.3
338752                  4.7                       13.1
338753                  4.0                        7.9

        %_uninsured_count_all_income_error  %_insured_count_all_income  \
0                                      0.4                        88.3
1                                      0.3                        28.6
2                                      0.3                        35.9
3                                      0.2                        19.3
4                                      0.3                        56.2
...                                    ...                         ...
338749                                 2.0                        17.6
338750                                 2.3                        23.5
338751                                 1.6                        11.3
338752                                 2.8                        42.8
338753                                 1.7                        31.5

        %_insured_count_all_income_error  \
0                                    0.4
1                                    0.4
2                                    0.4
3                                    0.3
4                                    0.5
...                                  ...
338749                               2.7
338750                               3.1
338751                               2.2
338752                               3.9
338753                               3.1

                                            state_name    county_name
0       Alabama                                    ...    not defined
1       Alabama                                    ...    not defined
2       Alabama                                    ...    not defined
```

```
3       Alabama                                          ...    not defined
4       Alabama                                          ...    not defined
...                                                      ...            ...
338749  Wyoming                                          ...  Weston County
338750  Wyoming                                          ...  Weston County
338751  Wyoming                                          ...  Weston County
338752  Wyoming                                          ...  Weston County
338753  Wyoming                                          ...  Weston County

[338754 rows x 23 columns]
```

In [119…

```python
# Calculate the lower  & Upper limit of uninsured individuals
df['lower_uninsured'] = df['uninsured_count'] - df['uninsured_count_error']
print('Lower Uninsured:', df['lower_uninsured'])

df['upper_uninsured'] = df['uninsured_count'] + df['uninsured_count_error']
print('Upper_uninsured', df['upper_uninsured'])

# Calculate the lower & Upper limit of insured individuals
df['lower_insured'] = df['insured_count'] - df['insured_count_error']
print('Lower_insured', df['lower_insured'])

df['upper_insured'] = df['insured_count'] + df['insured_count_error']
print('Upper_insured', df['upper_uninsured'])
```

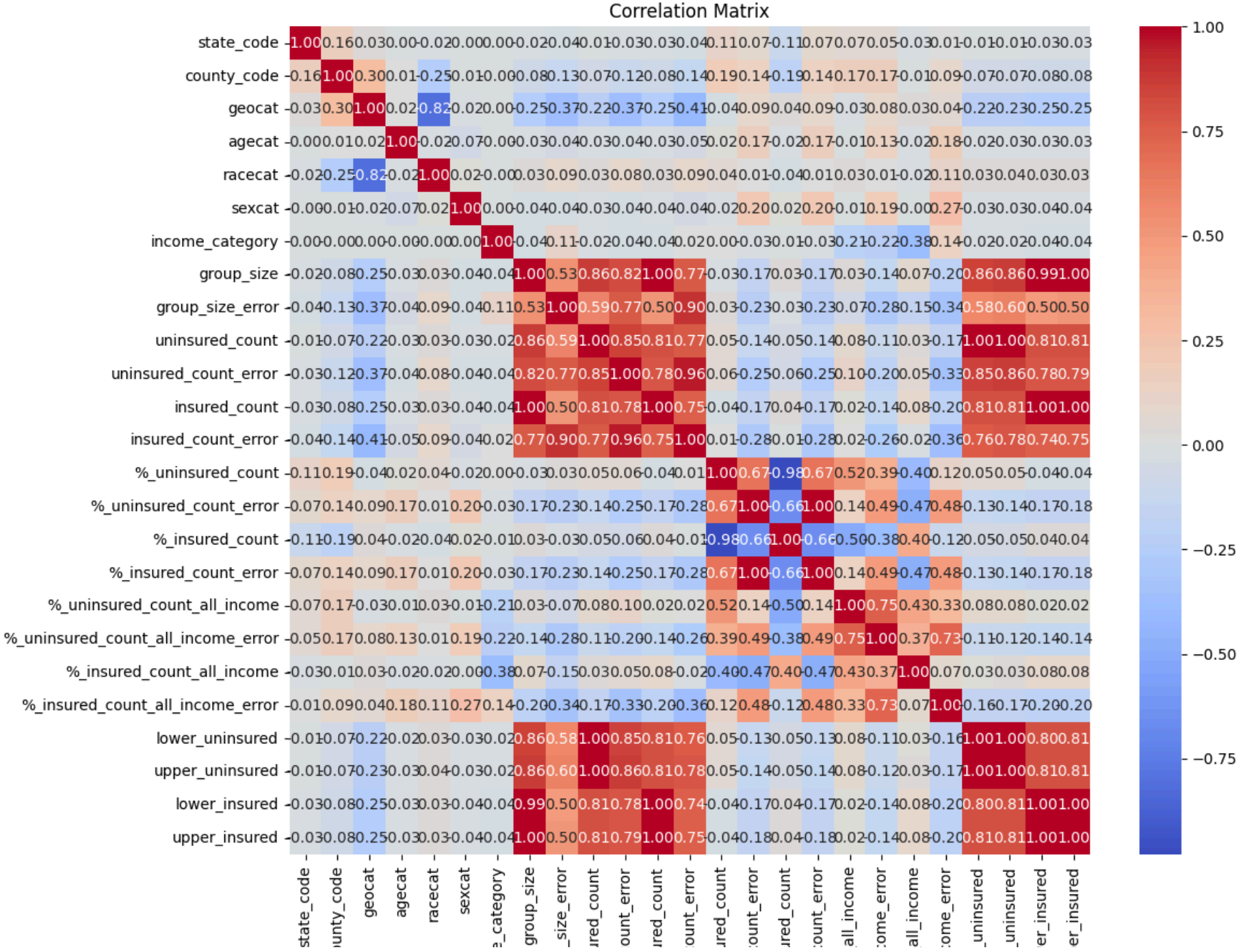
```
Lower Uninsured: 0         455340.0
1         259172.0
2         307970.0
3         185447.0
4         393773.0
            ...
338749        88.0
338750       113.0
338751        60.0
338752       169.0
338753       101.0
Name: lower_uninsured, Length: 338754, dtype: float64
Upper_uninsured 0         484434.0
1         279490.0
2         330062.0
3         202317.0
4         419687.0
            ...
338749       152.0
338750       187.0
338751       112.0
338752       261.0
338753       157.0
Name: upper_uninsured, Length: 338754, dtype: float64
Lower_insured 0         3533978.0
1         1133414.0
2         1427724.0
3          762459.0
4         2241823.0
            ...
338749        244.0
338750        334.0
338751        149.0
338752        636.0
338753        464.0
Name: lower_insured, Length: 338754, dtype: float64
Upper_insured 0         484434.0
1         279490.0
2         330062.0
3         202317.0
4         419687.0
```

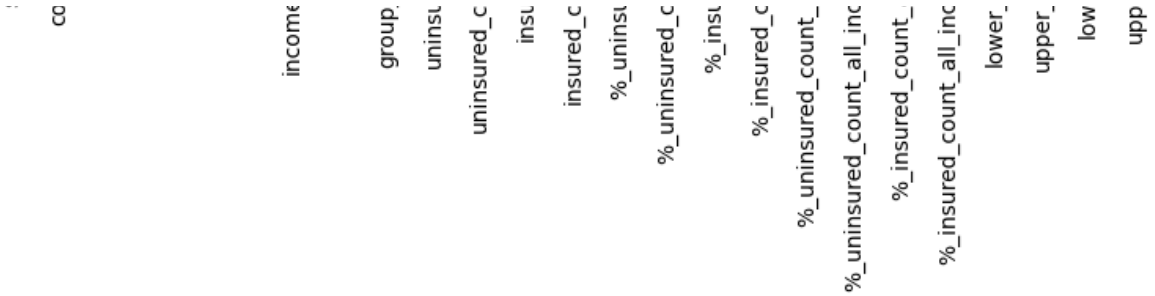
```
           ...
338749      152.0
338750      187.0
338751      112.0
338752      261.0
338753      157.0
Name: upper_uninsured, Length: 338754, dtype: float64
```

Using correlation matrix to identify correlations and dependencies between numeric variables.

In [120…

```python
# Select only numeric columns
numeric_df = df.select_dtypes(include=['number'])

# Calculate correlation matrix
correlation_matrix = numeric_df.corr()

# Increase figure size
plt.figure(figsize=(12, 10))

# Plot correlation matrix
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
state_code
county_code
geocat
agecat
racecat
sexcat
income_category
group_size
group_size_error
uninsured_count
uninsured_count_error
insured_count
insured_count_error
%_uninsured_count
%_uninsured_count_error
%_insured_count
%_insured_count_error
%_uninsured_count_all_income
%_uninsured_count_all_income_error
%_insured_count_all_income
%_insured_count_all_income_error
lower_uninsured
upper_uninsured
lower_insured
upper_insured
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



We can observe a significant negative correlation between the percentage of insured individuals and percentage of uninsured individuals. This implies that as the percentage of insured individuals increases,percentage of uninsured individuals decreases, suggesting a potential relationship worth exploring further in our analysis.

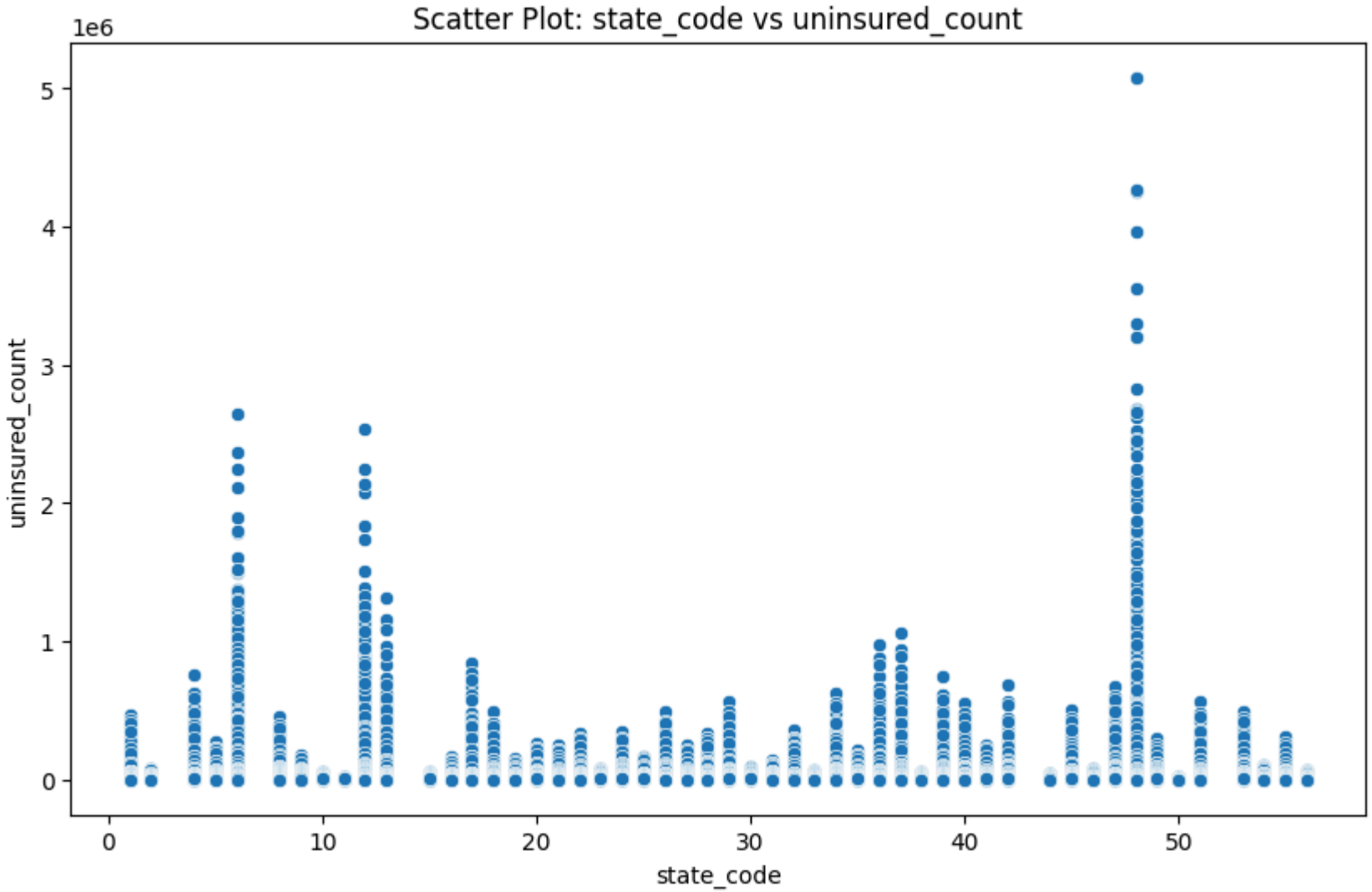
Additionally, there is a strong positive correlation between the error in insured and uninsured counts. This correlation likely arises from the relationship between the actual counts of insured and uninsured individuals.

Moderate correlations are also present with the 'racecat' and 'geocat' variables, indicating that these variables may also be worth investigating in our analysis.

How does uninsured counts vary across states?

In [121…

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='state_code', y='uninsured_count', data=df)
plt.title('Scatter Plot: state_code vs uninsured_count')
plt.xlabel('state_code')
plt.ylabel('uninsured_count')
plt.show()
```



States with codes 6, 12, and 48, namely California, Florida, and Texas, respectively, exhibit the highest numbers of uninsured individuals. Notably, these states are coastal regions adjacent to oceans. The cost of insurance premiums can be notably high, particularly in areas susceptible to natural calamities such as wildfires, and floods.

Evaluating the Performance of XGBoost Regressor for Predicting Uninsured Counts

In [122…

```
# Split data into training and testing sets
X_reg = df[['county_code', 'state_code','agecat', 'racecat', 'sexcat', 'income_category','geocat','group_size']]
y_reg = df['uninsured_count']

X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)

# Train the model
xgb_reg = XGBRegressor(random_state=42)
xgb_reg.fit(X_train_reg, y_train_reg)

# Evaluate the model
xgb_mse = mean_squared_error(y_test_reg, xgb_reg.predict(X_test_reg))
print(f"XGBoost Regressor Mean Squared Error: {xgb_mse}")

# Make predictions
y_pred_reg = xgb_reg.predict(X_test_reg)

# Calculate R-squared
r_squared = r2_score(y_test_reg, y_pred_reg)
print(f"XGBoost Regressor R-squared: {r_squared}")
```

```
XGBoost Regressor Mean Squared Error: 66796343.207252175
XGBoost Regressor R-squared: 0.9569597339550899
```

In [123…

```
# Get feature importance
importance = xgb_reg.feature_importances_
feature_names = X_train_reg.columns

# Create a DataFrame to store the feature importance values
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importance
})

# Sort the features by their importance
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

print("Feature Importance:")
print(feature_importance_df)
```

```
# Basic summary of the XGBoost model
print("\nXGBoost Model Summary:")
print("XGBoost Regressor Mean Squared Error:", xgb_mse)
print("XGBoost Regressor R-squared:", r_squared)
```

```
Feature Importance:
           Feature  Importance
7       group_size    0.435383
1       state_code    0.166062
3          racecat    0.108476
2           agecat    0.081974
4           sexcat    0.080665
0      county_code    0.070542
5  income_category    0.056898
6           geocat    0.000000

XGBoost Model Summary:
XGBoost Regressor Mean Squared Error: 66796343.207252175
XGBoost Regressor R-squared: 0.9569597339550899
```

Reasearch Question 1: What are the key factors affecting uninsured counts using Regression Model?

In [124…

```
# Selecting features and target variable
X = df[['state_code', 'county_code', 'geocat', 'agecat', 'racecat', 'sexcat', 'income_category', 'group_size']]
y = df['uninsured_count']

# Add constant to the features
X = sm.add_constant(X)

# Fitting the linear regression model
model = sm.OLS(y, X).fit()

# Getting summary of the regression model
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        uninsured_count   R-squared:                       0.745
Model:                            OLS   Adj. R-squared:                  0.745
Method:                 Least Squares   F-statistic:                 1.236e+05
Date:                Sun, 12 May 2024   Prob (F-statistic):               0.00
Time:                        00:24:18   Log-Likelihood:            -3.8971e+06
No. Observations:              338754   AIC:                         7.794e+06
Df Residuals:                  338745   BIC:                         7.794e+06
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -8284.0933   1267.520     -6.536      0.000   -1.08e+04   -5799.791
state_code        26.5734      2.745      9.681      0.000      21.193      31.953
county_code        2.1600      0.411      5.257      0.000       1.355       2.965
geocat           121.6611     25.387      4.792      0.000      71.903     171.419
agecat           122.7670     24.011      5.113      0.000      75.706     169.828
racecat          618.8023     57.113     10.835      0.000     506.862     730.742
sexcat          -138.9356     50.080     -2.774      0.006    -237.090     -40.781
income_category  410.9528     24.157     17.012      0.000     363.607     458.299
group_size         0.1458      0.000    911.898      0.000       0.145       0.146
==============================================================================
Omnibus:                   768985.040   Durbin-Watson:                   0.723
Prob(Omnibus):                  0.000   Jarque-Bera (JB):      54725101507.225
Skew:                          20.749   Prob(JB):                         0.00
Kurtosis:                    1971.612   Cond. No.                     8.77e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.77e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The model exhibits a 74% accuracy, indicating that a significant portion of the variation in uninsurance count can be explained by the other variables included in the model.

Feature Importance: Race Category, Income Category, Gender and Age Category

Research Question 2: How does the uninsured population vary among U.S. states, specifically focusing on the race , age, income and gender category?

Race Category

In [125…

```
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Group by state code and sum the total count for race category 1: White only
state_total_count = df[df['racecat'] == 1].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count['state_name'] = state_total_count['state_code'].map(state_names)

print(state_total_count)

# Highest 5 states
highest_5_states = state_total_count.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - White Alone :")
print(highest_5_states[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states = state_total_count.nsmallest(5, 'uninsured_count')
print("\nLeast 5 states based on race category - White Alone :")
print(least_5_states[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states = pd.concat([highest_5_states, least_5_states])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data = top_bottom_states.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category White only')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	6820056.0	
1	2	749372.0	
2	4	4980004.0	
3	5	3671380.0	
4	6	9984248.0	
5	8	4143372.0	
6	9	1332878.0	
7	10	536496.0	
8	11	69176.0	
9	12	24302468.0	
10	13	12631618.0	
11	15	219920.0	
12	16	2326476.0	
13	17	6980866.0	
14	18	7325116.0	
15	19	2276084.0	
16	20	3582554.0	
17	21	4254980.0	
18	22	3762154.0	
19	23	1630246.0	
20	24	1827428.0	
21	25	1752074.0	
22	26	7226338.0	
23	27	3067790.0	
24	28	4032666.0	
25	29	9622118.0	
26	30	1499542.0	
27	31	1697162.0	
28	32	2339676.0	
29	33	1137576.0	
30	34	3432884.0	
31	35	866994.0	
32	36	7042850.0	
33	37	11837936.0	
34	38	802802.0	
35	39	11659922.0	
36	40	6491300.0	
37	41	3434308.0	
38	42	9096668.0	
39	44	454190.0	

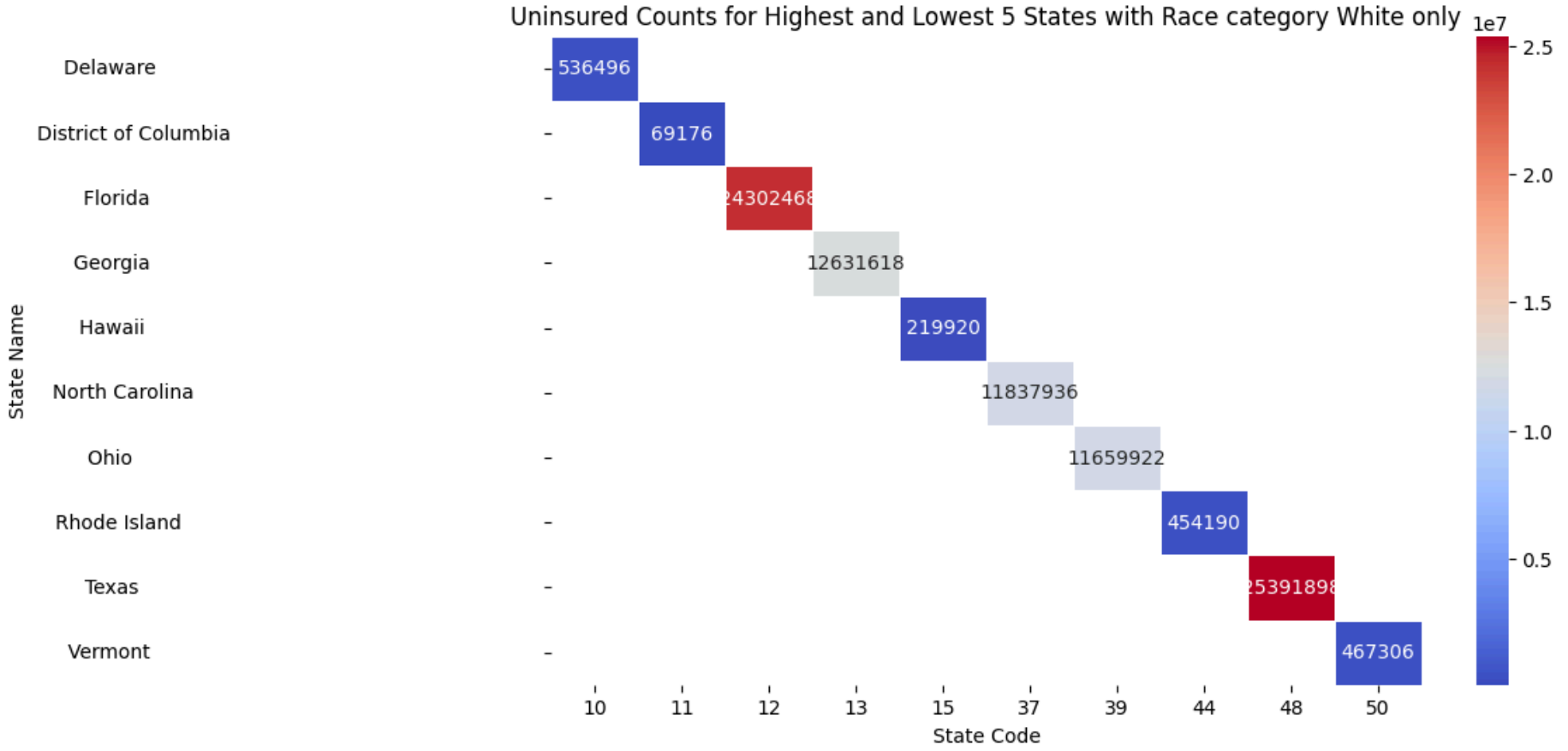
```
40          45        6374654.0
41          46        1096230.0
42          47       10782676.0
43          48       25391898.0
44          49        3095886.0
45          50         467306.0
46          51        5048130.0
47          53        4641196.0
48          54        2415368.0
49          55        4099254.0
50          56        1094810.0
```

	state_name
0	Alabama ...
1	Alaska ...
2	Arizona ...
3	Arkansas ...
4	California ...
5	Colorado ...
6	Connecticut ...
7	Delaware ...
8	District of Columbia ...
9	Florida ...
10	Georgia ...
11	Hawaii ...
12	Idaho ...
13	Illinois ...
14	Indiana ...
15	Iowa ...
16	Kansas ...
17	Kentucky ...
18	Louisiana ...
19	Maine ...
20	Maryland ...
21	Massachusetts ...
22	Michigan ...
23	Minnesota ...
24	Mississippi ...
25	Missouri ...
26	Montana ...
27	Nebraska ...

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - White Alone :
    state_code  uninsured_count  \
43          48       25391898.0
9           12       24302468.0
10          13       12631618.0
33          37       11837936.0
35          39       11659922.0

                                          state_name
43  Texas                                       ...
9   Florida                                     ...
10  Georgia                                     ...
33  North Carolina                              ...
35  Ohio                                        ...

Least 5 states based on race category - White Alone :
    state_code  uninsured_count  \
8           11          69176.0
```

```
11          15          219920.0
39          44          454190.0
45          50          467306.0
7           10          536496.0
```

```
                    state_name
8   District of Columbia    ...
11  Hawaii                  ...
39  Rhode Island            ...
45  Vermont                 ...
7   Delaware                ...
```



In [126…

```
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()
```

```python
# Group by state code and sum the total count for race category 2: BLACK OR AFRICAN AMERICAN ALONE
state_total_count2 = df[df['racecat'] == 2].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count2['state_name'] = state_total_count2['state_code'].map(state_names)

print(state_total_count2)

# Highest 5 states
highest_5_states2 = state_total_count2.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - BLACK OR AFRICAN AMERICAN ALONE :")
print(highest_5_states2[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states2 = state_total_count2.nsmallest(5, 'uninsured_count')
print("\nLeast 5 states based on race category -  BLACK OR AFRICAN AMERICAN ALONE :")
print(least_5_states2[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states2 = pd.concat([highest_5_states2, least_5_states2])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data2 = top_bottom_states2.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(15, 6))
sns.heatmap(heatmap_data2, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category BLACK OR AFRICAN AMERICAN ALONE')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	3730322.0	
1	2	62528.0	
2	4	718034.0	
3	5	1038266.0	
4	6	2587554.0	
5	8	409616.0	
6	9	461154.0	
7	10	275004.0	
8	11	207210.0	
9	12	10840846.0	
10	13	10588192.0	
11	15	22802.0	
12	16	36414.0	
13	17	3135760.0	
14	18	1387186.0	
15	19	211000.0	
16	20	462548.0	
17	21	624758.0	
18	22	2882000.0	
19	23	42904.0	
20	24	1976018.0	
21	25	386794.0	
22	26	2018312.0	
23	27	549130.0	
24	28	3778900.0	
25	29	2050302.0	
26	30	16774.0	
27	31	202056.0	
28	32	714818.0	
29	33	32622.0	
30	34	1882690.0	
31	35	83274.0	
32	36	3515682.0	
33	37	5910808.0	
34	38	71318.0	
35	39	2847890.0	
36	40	1005718.0	
37	41	129338.0	
38	42	1995170.0	
39	44	63472.0	

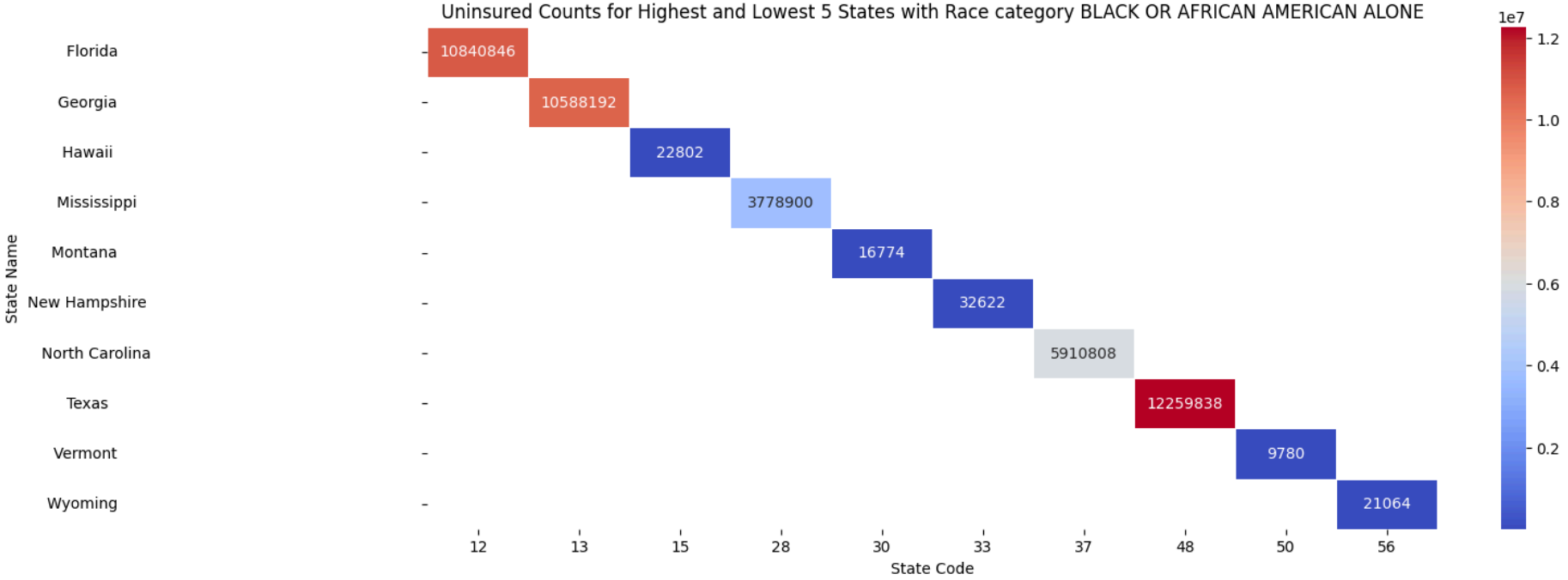
```
40          45        3600900.0
41          46          68126.0
42          47        3206262.0
43          48       12259838.0
44          49         101964.0
45          50           9780.0
46          51        2353530.0
47          53         517312.0
48          54         107256.0
49          55         655116.0
50          56          21064.0
```

	state_name	...
0	Alabama	...
1	Alaska	...
2	Arizona	...
3	Arkansas	...
4	California	...
5	Colorado	...
6	Connecticut	...
7	Delaware	...
8	District of Columbia	...
9	Florida	...
10	Georgia	...
11	Hawaii	...
12	Idaho	...
13	Illinois	...
14	Indiana	...
15	Iowa	...
16	Kansas	...
17	Kentucky	...
18	Louisiana	...
19	Maine	...
20	Maryland	...
21	Massachusetts	...
22	Michigan	...
23	Minnesota	...
24	Mississippi	...
25	Missouri	...
26	Montana	...
27	Nebraska	...

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - BLACK OR AFRICAN AMERICAN ALONE :
    state_code  uninsured_count  \
43          48       12259838.0
9           12       10840846.0
10          13       10588192.0
33          37        5910808.0
24          28        3778900.0

                                          state_name
43  Texas                                       ...
9   Florida                                     ...
10  Georgia                                     ...
33  North Carolina                              ...
24  Mississippi                                 ...

Least 5 states based on race category -  BLACK OR AFRICAN AMERICAN ALONE :
    state_code  uninsured_count  \
45          50           9780.0
```

```
26          30          16774.0
50          56          21064.0
11          15          22802.0
29          33          32622.0
```

```
                                state_name
45  Vermont                           ...
26  Montana                           ...
50  Wyoming                           ...
11  Hawaii                            ...
29  New Hampshire                     ...
```



In [127…

```python
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Group by state code and sum the total count for race category 3: Hisapanic or Latino
state_total_count3 = df[df['racecat'] == 3].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count3['state_name'] = state_total_count3['state_code'].map(state_names)
```

```
print(state_total_count3)

# Highest 5 states
highest_5_states3 = state_total_count3.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - Hisapanic or Latino :")
print(highest_5_states3[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states3 = state_total_count3.nsmallest(5, 'uninsured_count')
print("\nLeast 5 states based on race category -Hisapanic or Latino :")
print(least_5_states3[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states3 = pd.concat([highest_5_states3, least_5_states3])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data3 = top_bottom_states3.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(15, 6))
sns.heatmap(heatmap_data3, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category Hisapanic or Latino')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	1553678.0	
1	2	223316.0	
2	4	9986606.0	
3	5	1568932.0	
4	6	45241106.0	
5	8	4936086.0	
6	9	1927974.0	
7	10	370616.0	
8	11	107298.0	
9	12	25043230.0	
10	13	7805426.0	
11	15	223388.0	
12	16	1178630.0	
13	17	9128802.0	
14	18	2011438.0	
15	19	728324.0	
16	20	1883150.0	
17	21	877876.0	
18	22	1470848.0	
19	23	92674.0	
20	24	3066580.0	
21	25	1123404.0	
22	26	1504524.0	
23	27	1100536.0	
24	28	693276.0	
25	29	1417654.0	
26	30	195256.0	
27	31	1069560.0	
28	32	4547406.0	
29	33	183568.0	
30	34	8313512.0	
31	35	3427836.0	
32	36	9461442.0	
33	37	7599040.0	
34	38	139400.0	
35	39	1818866.0	
36	40	2980168.0	
37	41	1947350.0	
38	42	3314146.0	
39	44	424090.0	

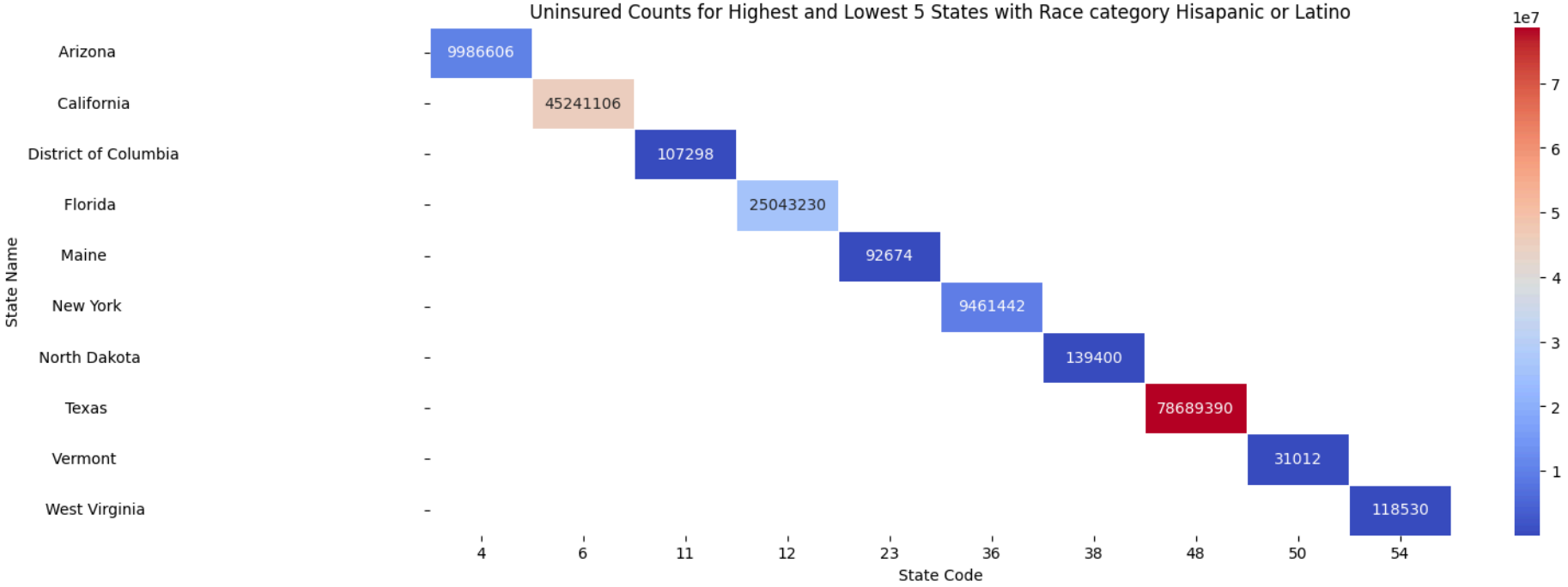
```
40          45        2210584.0
41          46         185086.0
42          47        2928646.0
43          48       78689390.0
44          49        2452610.0
45          50          31012.0
46          51        3861588.0
47          53        4153838.0
48          54         118530.0
49          55        1682062.0
50          56         331674.0
```

	state_name
0	Alabama ...
1	Alaska ...
2	Arizona ...
3	Arkansas ...
4	California ...
5	Colorado ...
6	Connecticut ...
7	Delaware ...
8	District of Columbia ...
9	Florida ...
10	Georgia ...
11	Hawaii ...
12	Idaho ...
13	Illinois ...
14	Indiana ...
15	Iowa ...
16	Kansas ...
17	Kentucky ...
18	Louisiana ...
19	Maine ...
20	Maryland ...
21	Massachusetts ...
22	Michigan ...
23	Minnesota ...
24	Mississippi ...
25	Missouri ...
26	Montana ...
27	Nebraska ...

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - Hisapanic or Latino :
    state_code  uninsured_count  \
43          48       78689390.0
4            6       45241106.0
9           12       25043230.0
2            4        9986606.0
32          36        9461442.0

                                          state_name
43  Texas                                       ...
4   California                                  ...
9   Florida                                     ...
2   Arizona                                     ...
32  New York                                    ...

Least 5 states based on race category -Hisapanic or Latino :
    state_code  uninsured_count  \
45          50          31012.0
```

```
19          23          92674.0
8           11         107298.0
48          54         118530.0
34          38         139400.0
```

```
                                state_name
45  Vermont                            ...
19  Maine                              ...
8   District of Columbia               ...
48  West Virginia                      ...
34  North Dakota                       ...
```



In [128…

```python
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Group by state code and sum the total count for race category 4: American Indian & Alaska Native Alone
state_total_count4 = df[df['racecat'] == 4].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count4['state_name'] = state_total_count4['state_code'].map(state_names)
```

```
print(state_total_count4)

# Highest 5 states
highest_5_states4 = state_total_count4.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - American Indian & Alaska Native Alone :")
print(highest_5_states4[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states4 = state_total_count4.nsmallest(5, 'uninsured_count')
print("\nLeast 5 states based on race category - American Indian & Alaska Native Alone :")
print(least_5_states4[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states4 = pd.concat([highest_5_states4, least_5_states4])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data4 = top_bottom_states4.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(15, 6))
sns.heatmap(heatmap_data4, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category American Indian & Alaska Native Alone')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	70086.0	
1	2	413244.0	
2	4	1172100.0	
3	5	60752.0	
4	6	249412.0	
5	8	69236.0	
6	9	6664.0	
7	10	3634.0	
8	11	814.0	
9	12	162628.0	
10	13	74308.0	
11	15	2352.0	
12	16	53026.0	
13	17	30690.0	
14	18	33408.0	
15	19	15466.0	
16	20	65362.0	
17	21	15376.0	
18	22	73918.0	
19	23	14306.0	
20	24	17730.0	
21	25	6850.0	
22	26	113436.0	
23	27	127036.0	
24	28	56016.0	
25	29	88304.0	
26	30	248784.0	
27	31	36120.0	
28	32	76538.0	
29	33	3548.0	
30	34	16396.0	
31	35	698932.0	
32	36	78366.0	
33	37	399976.0	
34	38	125400.0	
35	39	43784.0	
36	40	1807076.0	
37	41	70264.0	
38	42	26930.0	
39	44	3794.0	

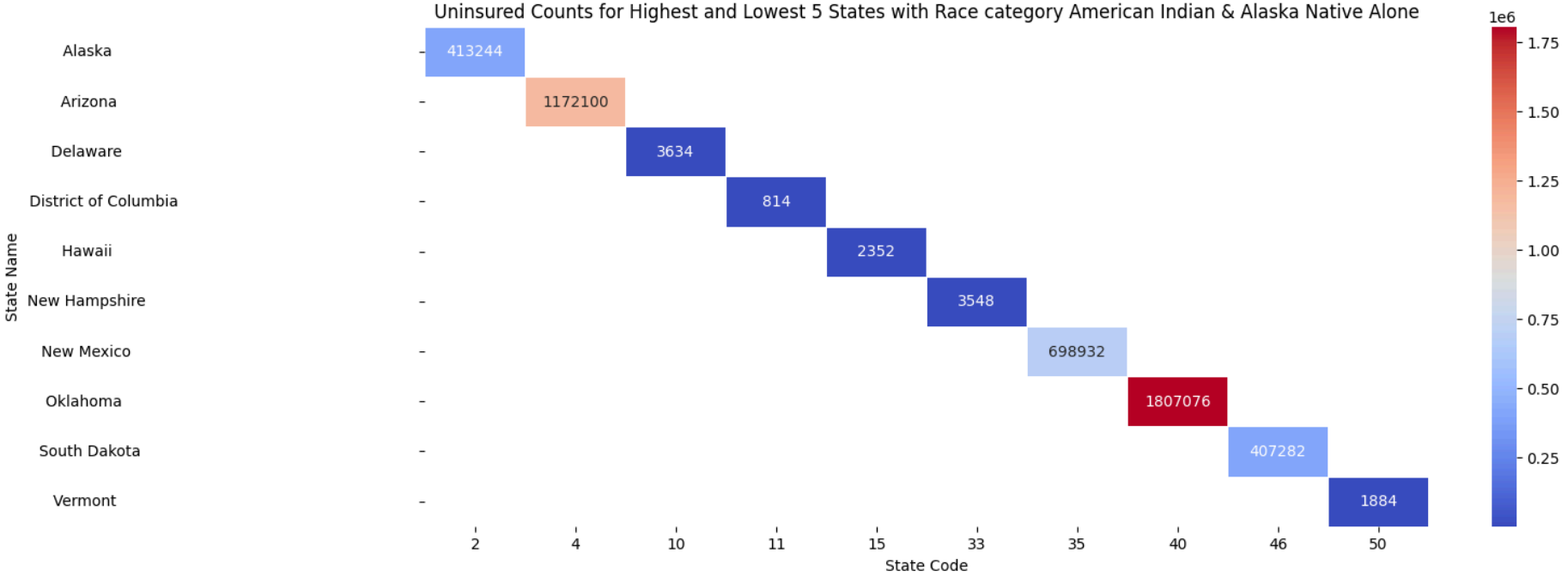
```
40          45          62834.0
41          46         407282.0
42          47          55526.0
43          48         305908.0
44          49          90050.0
45          50           1884.0
46          51          32722.0
47          53         206062.0
48          54           6794.0
49          55         107260.0
50          56          49730.0
```

```
                                    state_name
0   Alabama                                ...
1   Alaska                                 ...
2   Arizona                                ...
3   Arkansas                               ...
4   California                             ...
5   Colorado                               ...
6   Connecticut                            ...
7   Delaware                               ...
8   District of Columbia                   ...
9   Florida                                ...
10  Georgia                                ...
11  Hawaii                                 ...
12  Idaho                                  ...
13  Illinois                               ...
14  Indiana                                ...
15  Iowa                                   ...
16  Kansas                                 ...
17  Kentucky                               ...
18  Louisiana                              ...
19  Maine                                  ...
20  Maryland                               ...
21  Massachusetts                          ...
22  Michigan                               ...
23  Minnesota                              ...
24  Mississippi                            ...
25  Missouri                               ...
26  Montana                                ...
27  Nebraska                               ...
```

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - American Indian & Alaska Native Alone :
    state_code  uninsured_count  \
36          40        1807076.0
2            4        1172100.0
31          35         698932.0
1            2         413244.0
41          46         407282.0

                                          state_name
36  Oklahoma                                      ...
2   Arizona                                       ...
31  New Mexico                                    ...
1   Alaska                                        ...
41  South Dakota                                  ...

Least 5 states based on race category - American Indian & Alaska Native Alone :
    state_code  uninsured_count  \
8           11            814.0
```

```
45          50          1884.0
11          15          2352.0
29          33          3548.0
7           10          3634.0
```

	state_name	...
8	District of Columbia	...
45	Vermont	...
11	Hawaii	...
29	New Hampshire	...
7	Delaware	...



In [129…

```python
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Group by state code and sum the total count for race category 5: Asian Alone
state_total_count5 = df[df['racecat'] == 5].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count5['state_name'] = state_total_count5['state_code'].map(state_names)
```

```
print(state_total_count5)

# Highest 5 states
highest_5_states5 = state_total_count5.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - Asian Alone :")
print(highest_5_states5[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states5 = state_total_count5.nsmallest(5, 'uninsured_count')
print("\nLeast 5 states based on race category - Asian Alone :")
print(least_5_states5[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states5 = pd.concat([highest_5_states5, least_5_states5])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data5 = top_bottom_states5.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(15, 6))
sns.heatmap(heatmap_data5, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category Asian Alone')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	159322.0	
1	2	115280.0	
2	4	403542.0	
3	5	111380.0	
4	6	4575582.0	
5	8	247014.0	
6	9	146946.0	
7	10	39638.0	
8	11	13500.0	
9	12	1315682.0	
10	13	1016972.0	
11	15	289960.0	
12	16	56392.0	
13	17	912124.0	
14	18	303450.0	
15	19	122156.0	
16	20	168874.0	
17	21	111152.0	
18	22	168696.0	
19	23	26076.0	
20	24	409578.0	
21	25	239892.0	
22	26	305276.0	
23	27	314876.0	
24	28	101310.0	
25	29	249976.0	
26	30	23142.0	
27	31	88150.0	
28	32	546742.0	
29	33	52100.0	
30	34	791760.0	
31	35	55734.0	
32	36	2242914.0	
33	37	561282.0	
34	38	23240.0	
35	39	476376.0	
36	40	259616.0	
37	41	212432.0	
38	42	657584.0	
39	44	29764.0	

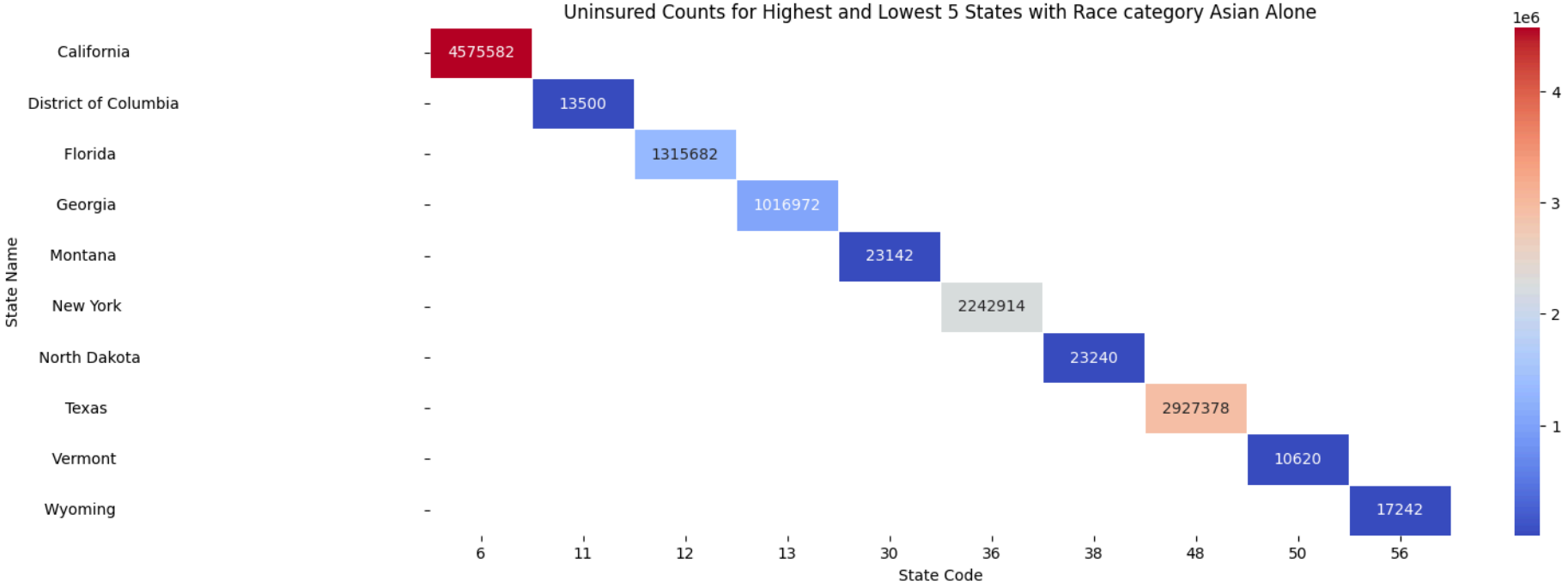
```
40          45         243328.0
41          46          36116.0
42          47         282146.0
43          48        2927378.0
44          49         152534.0
45          50          10620.0
46          51         681950.0
47          53         721428.0
48          54          24966.0
49          55         244890.0
50          56          17242.0
```

```
                                    state_name
0   Alabama                                ...
1   Alaska                                 ...
2   Arizona                                ...
3   Arkansas                               ...
4   California                             ...
5   Colorado                               ...
6   Connecticut                            ...
7   Delaware                               ...
8   District of Columbia                   ...
9   Florida                                ...
10  Georgia                                ...
11  Hawaii                                 ...
12  Idaho                                  ...
13  Illinois                               ...
14  Indiana                                ...
15  Iowa                                   ...
16  Kansas                                 ...
17  Kentucky                               ...
18  Louisiana                              ...
19  Maine                                  ...
20  Maryland                               ...
21  Massachusetts                          ...
22  Michigan                               ...
23  Minnesota                              ...
24  Mississippi                            ...
25  Missouri                               ...
26  Montana                                ...
27  Nebraska                               ...
```

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - Asian Alone :
    state_code  uninsured_count  \
4            6        4575582.0
43          48        2927378.0
32          36        2242914.0
9           12        1315682.0
10          13        1016972.0

                                           state_name
4   California                                     ...
43  Texas                                          ...
32  New York                                       ...
9   Florida                                        ...
10  Georgia                                        ...

Least 5 states based on race category - Asian Alone :
    state_code  uninsured_count  \
45          50          10620.0
```

```
8           11          13500.0
50          56          17242.0
26          30          23142.0
34          38          23240.0
```

```
                                state_name
45  Vermont                           ...
8   District of Columbia              ...
50  Wyoming                           ...
26  Montana                           ...
34  North Dakota                      ...
```



In [130…

```python
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Group by state code and sum the total count for race category 6: Native Hawaiian & Other
state_total_count6 = df[df['racecat'] == 6].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count6['state_name'] = state_total_count6['state_code'].map(state_names)
```

```
print(state_total_count6)

# Highest 5 states
highest_5_states6 = state_total_count6.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - Native Hawaiian & Other :")
print(highest_5_states6[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states6 = state_total_count6.nsmallest(5, 'uninsured_count')
print("\nLeast 5 states based on race category - Native Hawaiian & Other :")
print(least_5_states6[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states6 = pd.concat([highest_5_states6, least_5_states6])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data6 = top_bottom_states6.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(15, 6))
sns.heatmap(heatmap_data6, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category Native Hawaiian & Other')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	8018.0	
1	2	34042.0	
2	4	34858.0	
3	5	47292.0	
4	6	235512.0	
5	8	16290.0	
6	9	1396.0	
7	10	672.0	
8	11	142.0	
9	12	44762.0	
10	13	26796.0	
11	15	170072.0	
12	16	9736.0	
13	17	6546.0	
14	18	6822.0	
15	19	11018.0	
16	20	8838.0	
17	21	7536.0	
18	22	4792.0	
19	23	686.0	
20	24	4594.0	
21	25	1828.0	
22	26	4796.0	
23	27	4688.0	
24	28	4818.0	
25	29	29422.0	
26	30	2274.0	
27	31	2854.0	
28	32	67398.0	
29	33	680.0	
30	34	4590.0	
31	35	3278.0	
32	36	14040.0	
33	37	22610.0	
34	38	1474.0	
35	39	11612.0	
36	40	38032.0	
37	41	31698.0	
38	42	6320.0	
39	44	642.0	

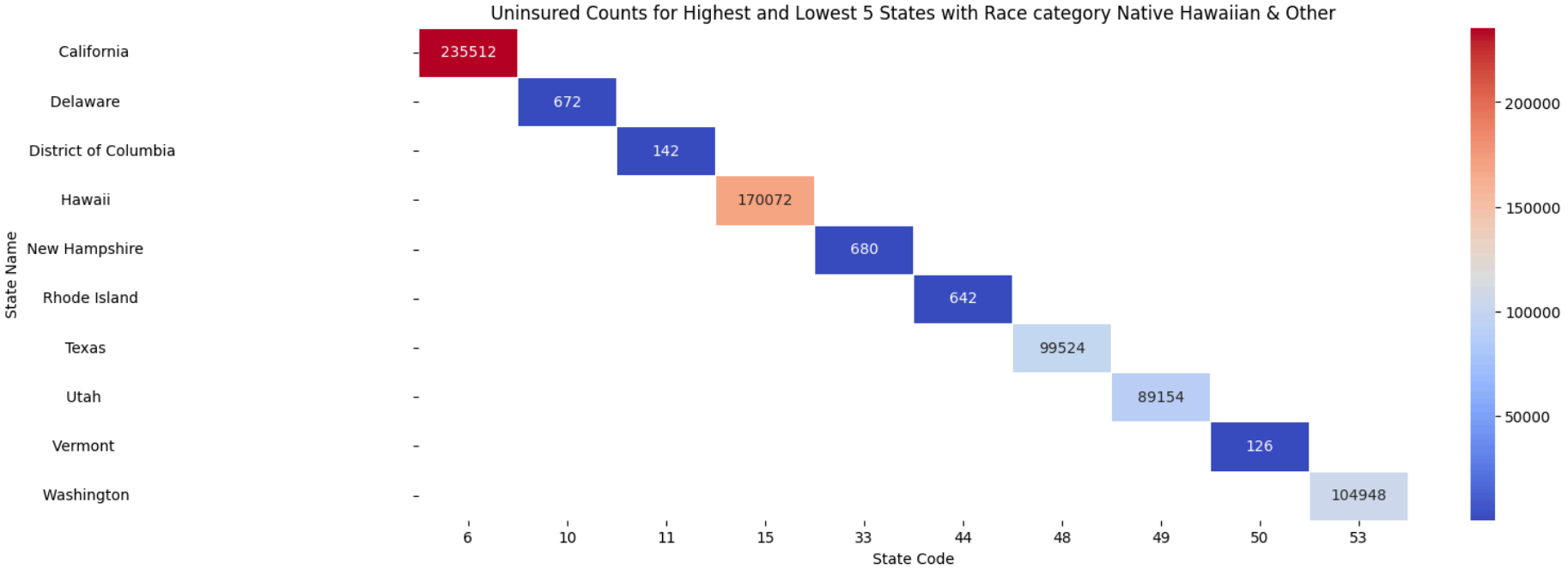
```
40          45          10736.0
41          46           1806.0
42          47          12748.0
43          48          99524.0
44          49          89154.0
45          50            126.0
46          51          11328.0
47          53         104948.0
48          54            734.0
49          55           3992.0
50          56           1472.0
```

```
                                    state_name
0   Alabama                                ...
1   Alaska                                 ...
2   Arizona                                ...
3   Arkansas                               ...
4   California                             ...
5   Colorado                               ...
6   Connecticut                            ...
7   Delaware                               ...
8   District of Columbia                   ...
9   Florida                                ...
10  Georgia                                ...
11  Hawaii                                 ...
12  Idaho                                  ...
13  Illinois                               ...
14  Indiana                                ...
15  Iowa                                   ...
16  Kansas                                 ...
17  Kentucky                               ...
18  Louisiana                              ...
19  Maine                                  ...
20  Maryland                               ...
21  Massachusetts                          ...
22  Michigan                               ...
23  Minnesota                              ...
24  Mississippi                            ...
25  Missouri                               ...
26  Montana                                ...
27  Nebraska                               ...
```

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - Native Hawaiian & Other :
    state_code  uninsured_count  \
4            6         235512.0
11          15         170072.0
47          53         104948.0
43          48          99524.0
44          49          89154.0

                                          state_name
4   California                                   ...
11  Hawaii                                       ...
47  Washington                                   ...
43  Texas                                        ...
44  Utah                                         ...

Least 5 states based on race category - Native Hawaiian & Other :
    state_code  uninsured_count  \
45          50            126.0
```

```
8           11          142.0
39          44          642.0
7           10          672.0
29          33          680.0
```

```
                                state_name
45  Vermont                           ...
8   District of Columbia              ...
39  Rhode Island                      ...
7   Delaware                          ...
29  New Hampshire                     ...
```



In [131…

```python
#Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Group by state code and sum the total count for race category 7: Two or more Races
state_total_count7 = df[df['racecat'] == 7].groupby('state_code')['uninsured_count'].sum().reset_index()

# Add state names to the grouped data
state_total_count7['state_name'] = state_total_count7['state_code'].map(state_names)
```

```
print(state_total_count7)

# Highest 5 states
highest_5_states7 = state_total_count7.nlargest(5, 'uninsured_count')
print("Highest 5 states based on race category - Two or more Races :")
print(highest_5_states7[['state_code', 'uninsured_count', 'state_name']])

# Least 5 states
least_5_states7 = state_total_count7.nsmallest(7, 'uninsured_count')
print("\nLeast 5 states based on race category -Two or more Races :")
print(least_5_states7[['state_code', 'uninsured_count', 'state_name']])

# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states7 = pd.concat([highest_5_states7, least_5_states7])

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data7 = top_bottom_states7.pivot(index='state_name', columns='state_code', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(15, 6))
sns.heatmap(heatmap_data7, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Race category Two or more Races')
plt.xlabel('State Code')
plt.ylabel('State Name')
plt.show()
```

	state_code	uninsured_count	\
0	1	252648.0	
1	2	155520.0	
2	4	403188.0	
3	5	186952.0	
4	6	1328676.0	
5	8	239410.0	
6	9	74366.0	
7	10	39736.0	
8	11	10542.0	
9	12	1326548.0	
10	13	706932.0	
11	15	250866.0	
12	16	116430.0	
13	17	340536.0	
14	18	300190.0	
15	19	88770.0	
16	20	269588.0	
17	21	154088.0	
18	22	165646.0	
19	23	51874.0	
20	24	211358.0	
21	25	71820.0	
22	26	433844.0	
23	27	198624.0	
24	28	142058.0	
25	29	514380.0	
26	30	86264.0	
27	31	82214.0	
28	32	332520.0	
29	33	37276.0	
30	34	203110.0	
31	35	86446.0	
32	36	511876.0	
33	37	693352.0	
34	38	38630.0	
35	39	552820.0	
36	40	1176394.0	
37	41	253376.0	
38	42	350754.0	
39	44	22104.0	

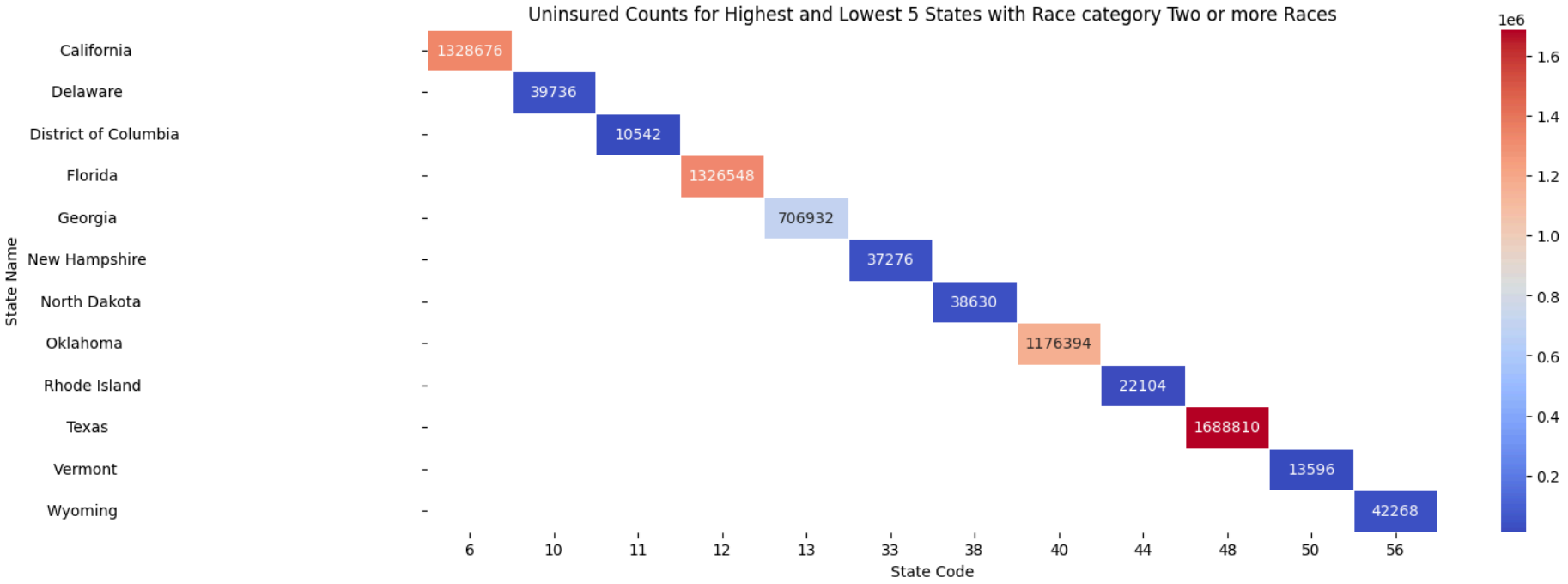
```
40          45         324460.0
41          46          66592.0
42          47         391150.0
43          48        1688810.0
44          49         176172.0
45          50          13596.0
46          51         479334.0
47          53         533892.0
48          54          68044.0
49          55         172906.0
50          56          42268.0
```

	state_name	...
0	Alabama	...
1	Alaska	...
2	Arizona	...
3	Arkansas	...
4	California	...
5	Colorado	...
6	Connecticut	...
7	Delaware	...
8	District of Columbia	...
9	Florida	...
10	Georgia	...
11	Hawaii	...
12	Idaho	...
13	Illinois	...
14	Indiana	...
15	Iowa	...
16	Kansas	...
17	Kentucky	...
18	Louisiana	...
19	Maine	...
20	Maryland	...
21	Massachusetts	...
22	Michigan	...
23	Minnesota	...
24	Mississippi	...
25	Missouri	...
26	Montana	...
27	Nebraska	...

```
28  Nevada                                          ...
29  New Hampshire                                   ...
30  New Jersey                                      ...
31  New Mexico                                      ...
32  New York                                        ...
33  North Carolina                                  ...
34  North Dakota                                    ...
35  Ohio                                            ...
36  Oklahoma                                        ...
37  Oregon                                          ...
38  Pennsylvania                                    ...
39  Rhode Island                                    ...
40  South Carolina                                  ...
41  South Dakota                                    ...
42  Tennessee                                       ...
43  Texas                                           ...
44  Utah                                            ...
45  Vermont                                         ...
46  Virginia                                        ...
47  Washington                                      ...
48  West Virginia                                   ...
49  Wisconsin                                       ...
50  Wyoming                                         ...
Highest 5 states based on race category - Two or more Races :
    state_code  uninsured_count  \
43          48        1688810.0
4            6        1328676.0
9           12        1326548.0
36          40        1176394.0
10          13         706932.0

                                          state_name
43  Texas                                       ...
4   California                                  ...
9   Florida                                     ...
36  Oklahoma                                    ...
10  Georgia                                     ...

Least 5 states based on race category -Two or more Races :
    state_code  uninsured_count  \
8           11          10542.0
```

```
45          50          13596.0
39          44          22104.0
29          33          37276.0
34          38          38630.0
7           10          39736.0
50          56          42268.0
```

	state_name	...
8	District of Columbia	...
45	Vermont	...
39	Rhode Island	...
29	New Hampshire	...
34	North Dakota	...
7	Delaware	...
50	Wyoming	...

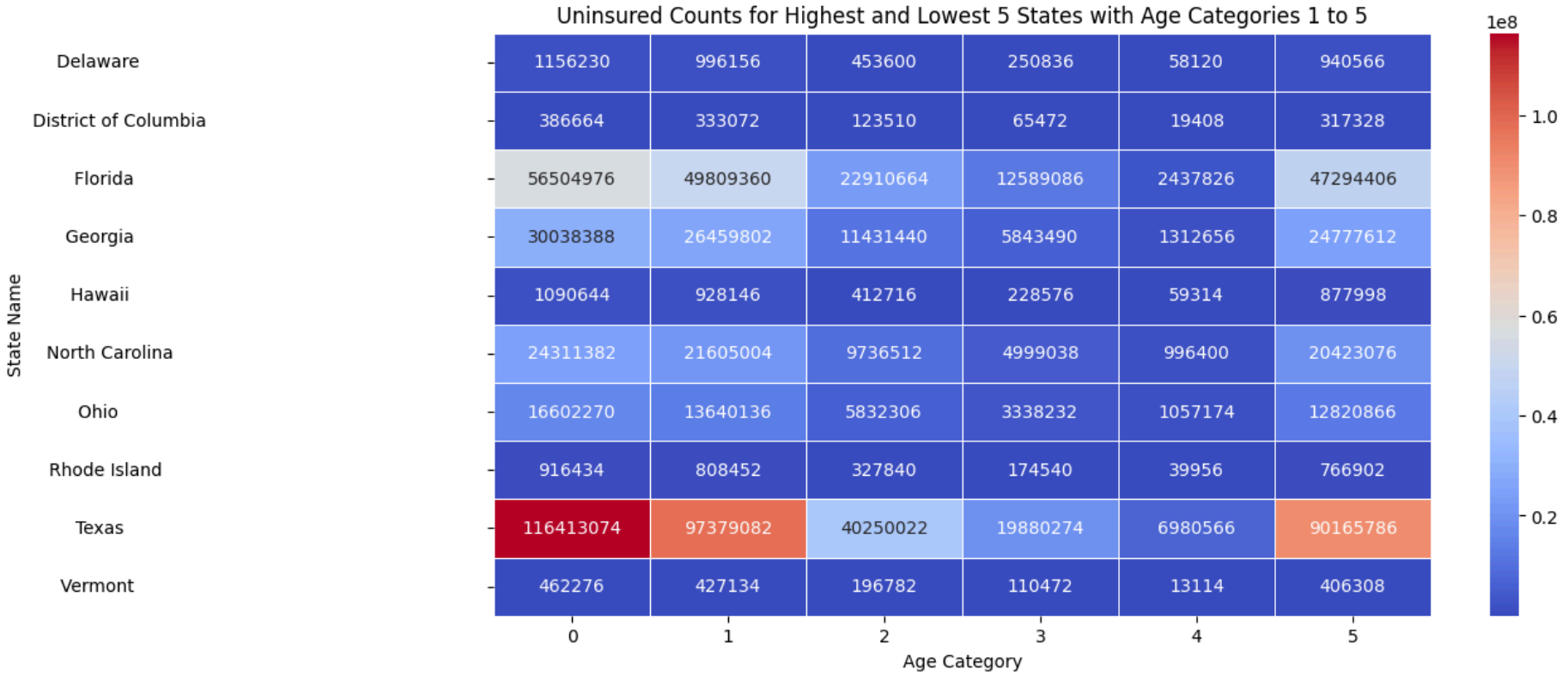


Age Category

In [132…

```
# Create a DataFrame containing the highest and lowest 5 states
top_bottom_states = pd.concat([highest_5_states, least_5_states])
```

```
# Get the state names for the highest and lowest 5 states
state_names_highest = top_bottom_states.nlargest(5, 'uninsured_count')['state_name']
state_names_lowest = top_bottom_states.nsmallest(5, 'uninsured_count')['state_name']
selected_states = pd.concat([state_names_highest, state_names_lowest])

# Filter the data for the selected states
filtered_data = df[df['state_name'].isin(selected_states)]

# Group by state name and age category, summing the total uninsured count
state_age_total_count = filtered_data.groupby(['state_name', 'agecat'])['uninsured_count'].sum().reset_index()

# Pivot the DataFrame to create a matrix for the heatmap
heatmap_data = state_age_total_count.pivot(index='state_name', columns='agecat', values='uninsured_count')

# Plot the heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(heatmap_data, cmap='coolwarm', annot=True, fmt=".0f", linewidths=0.5)
plt.title('Uninsured Counts for Highest and Lowest 5 States with Age Categories 1 to 5')
plt.xlabel('Age Category')
plt.ylabel('State Name')
plt.show()
```

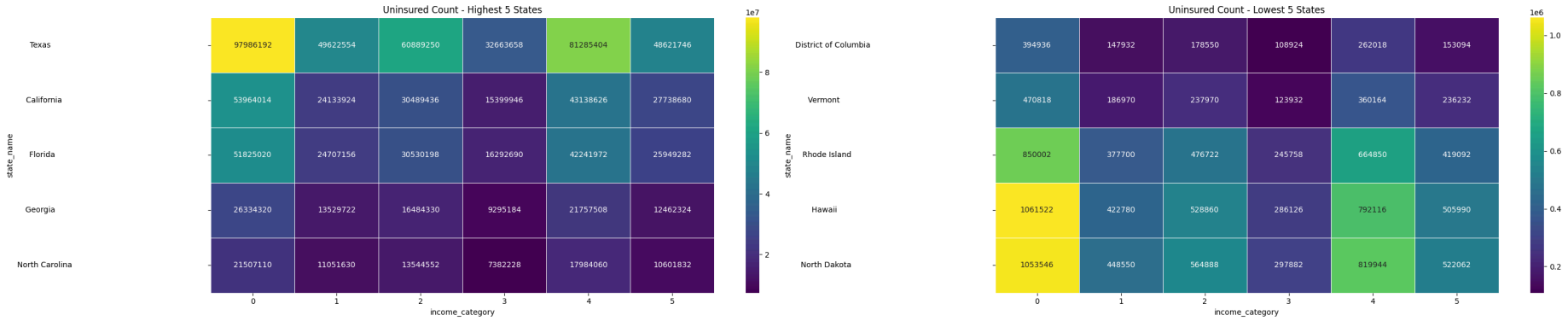


Income Category

In [133…

```python
# Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Filter the data for income categories 0 to 5
income_categories = list(range(6))
filtered_data = df[df['income_category'].isin(income_categories)]

# Group by state code and race category, then sum the uninsured counts
grouped_data = filtered_data.groupby(['state_code', 'income_category'])['uninsured_count'].sum().reset_index()

# Replace state codes with state names
grouped_data['state_name'] = grouped_data['state_code'].map(state_names)
```

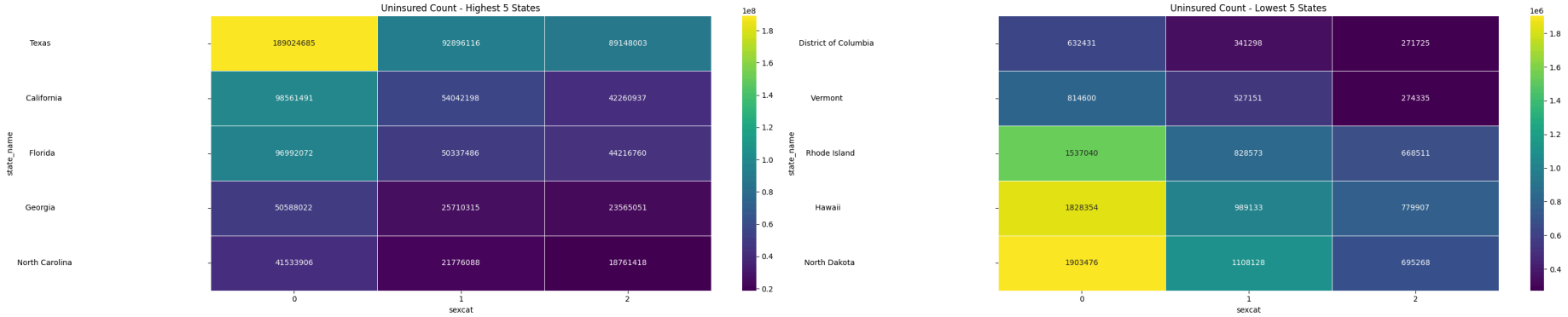
```
# Pivot the data to have race categories as columns
pivot_data = grouped_data.pivot(index='state_name', columns='income_category', values='uninsured_count').fillna(0)

# Get the states with highest and lowest uninsured counts
highest_5_states = pivot_data.sum(axis=1).nlargest(5).index
lowest_5_states = pivot_data.sum(axis=1).nsmallest(5).index

# Filter the pivot data for highest and lowest 5 states
highest_5_data = pivot_data.loc[highest_5_states]
lowest_5_data = pivot_data.loc[lowest_5_states]

# Plotting
plt.figure(figsize=(30, 6))

# Highest 5 states
plt.subplot(1, 2, 1)
sns.heatmap(highest_5_data, cmap='viridis', annot=True, fmt='.0f', linewidths=0.5)
plt.title('Uninsured Count - Highest 5 States')

# Lowest 5 states
plt.subplot(1, 2, 2)
sns.heatmap(lowest_5_data, cmap='viridis', annot=True, fmt='.0f', linewidths=0.5)
plt.title('Uninsured Count - Lowest 5 States')

plt.tight_layout()
plt.show()
```



Gender Category

In [134…

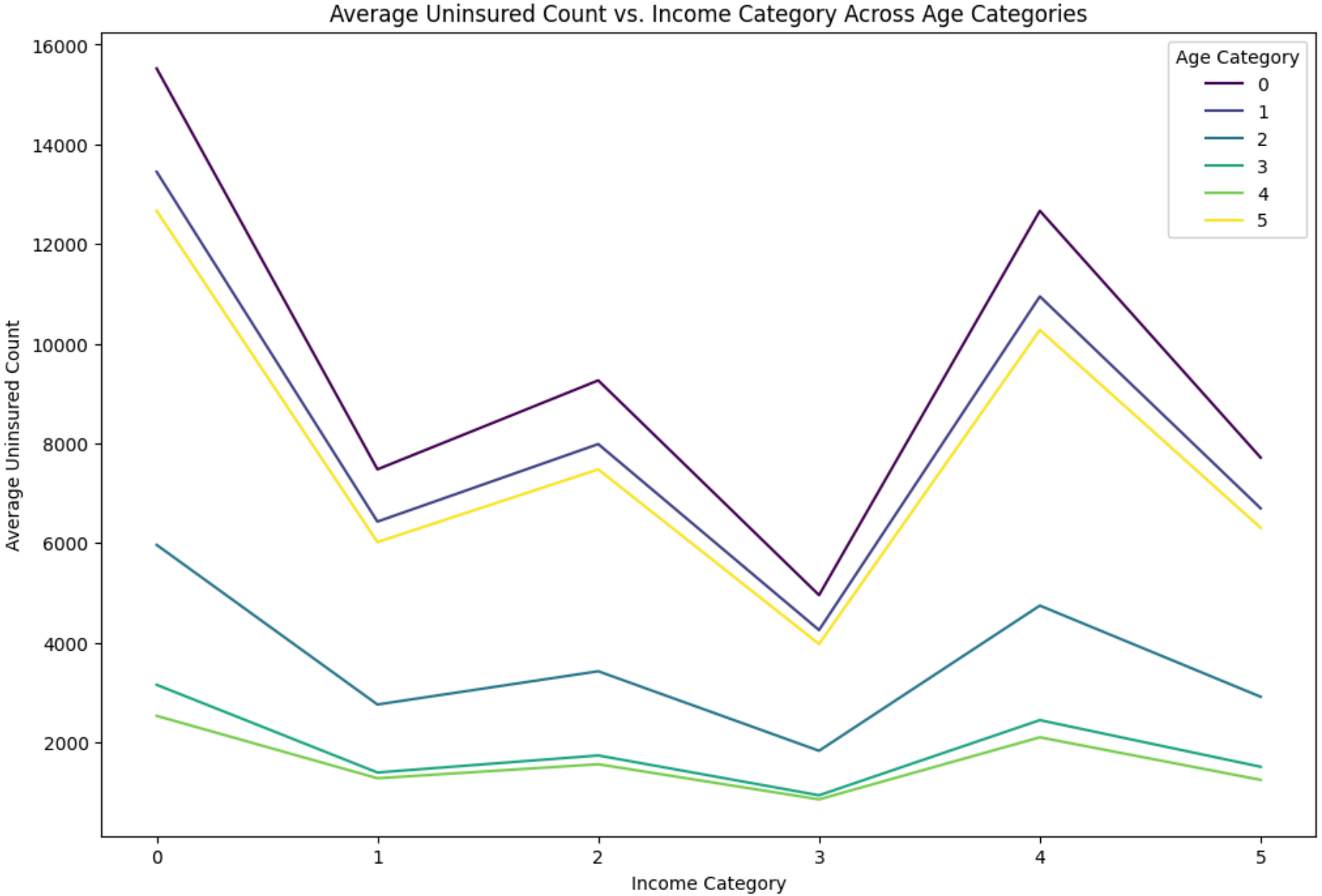
```
# Create a dictionary mapping state codes to state names
state_names = df[['state_code', 'state_name']].drop_duplicates().set_index('state_code')['state_name'].to_dict()

# Filter the data for income categories 0 to 2
sex_categories = list(range(3))
filtered_data = df[df['sexcat'].isin(sex_categories)]

# Group by state code and race category, then sum the uninsured counts
grouped_data = filtered_data.groupby(['state_code', 'sexcat'])['uninsured_count'].sum().reset_index()

# Replace state codes with state names
grouped_data['state_name'] = grouped_data['state_code'].map(state_names)

# Pivot the data to have race categories as columns
pivot_data = grouped_data.pivot(index='state_name', columns='sexcat', values='uninsured_count').fillna(0)

# Get the states with highest and lowest uninsured counts
highest_5_states = pivot_data.sum(axis=1).nlargest(5).index
lowest_5_states = pivot_data.sum(axis=1).nsmallest(5).index

# Filter the pivot data for highest and lowest 5 states
highest_5_data = pivot_data.loc[highest_5_states]
lowest_5_data = pivot_data.loc[lowest_5_states]

# Plotting
plt.figure(figsize=(30, 6))

# Highest 5 states
plt.subplot(1, 2, 1)
sns.heatmap(highest_5_data, cmap='viridis', annot=True, fmt='.0f', linewidths=0.5)
plt.title('Uninsured Count - Highest 5 States')

# Lowest 5 states
plt.subplot(1, 2, 2)
sns.heatmap(lowest_5_data, cmap='viridis', annot=True, fmt='.0f', linewidths=0.5)
plt.title('Uninsured Count - Lowest 5 States')

plt.tight_layout()
plt.show()
```



Research Question 3: How does the relationship between income and uninsured counts vary across different age categories?

In [135…

```
# Filter the data for relevant columns
data_plot = df[['agecat', 'income_category', 'uninsured_count']]

# Calculate the average uninsured count for each age category and income category
average_uninsured_count = data_plot.groupby(['agecat', 'income_category'])['uninsured_count'].mean().reset_index()

# Create a line plot
plt.figure(figsize=(12, 8))
sns.lineplot(x='income_category', y='uninsured_count', hue='agecat', data=average_uninsured_count, palette='viridis')
plt.title('Average Uninsured Count vs. Income Category Across Age Categories')
plt.xlabel('Income Category')
plt.ylabel('Average Uninsured Count')
plt.legend(title='Age Category')
plt.show()
```

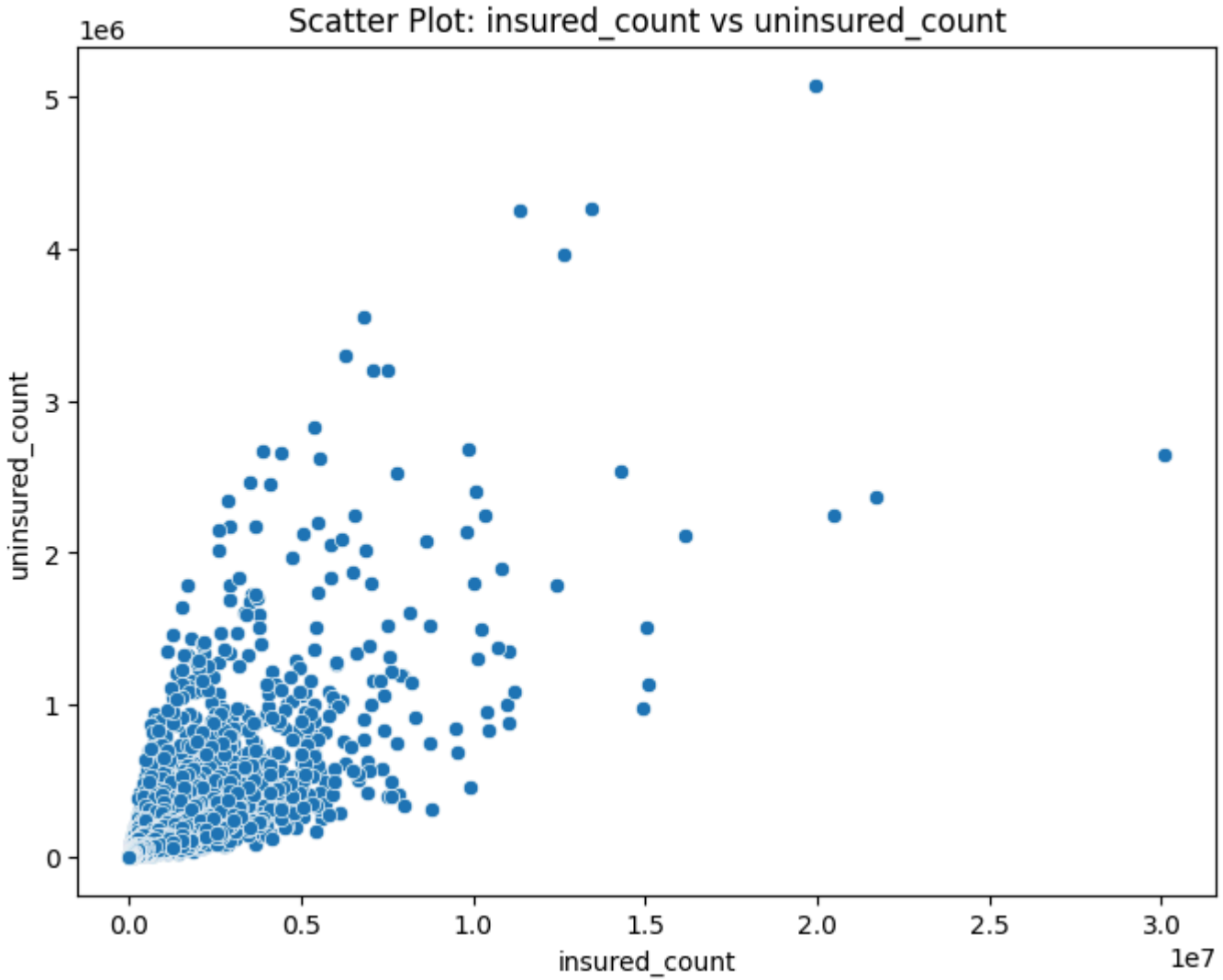


At or below 400% of the poverty level, we interpret the highest uninsured counts in all age categories.

Exploratory Data Analysis which were done earlier but not used for our project analysis

In [136…

```
# Example scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='insured_count', y='uninsured_count', data=df)
plt.title('Scatter Plot: insured_count vs uninsured_count')
plt.xlabel('insured_count')
plt.ylabel('uninsured_count')
plt.show()
```

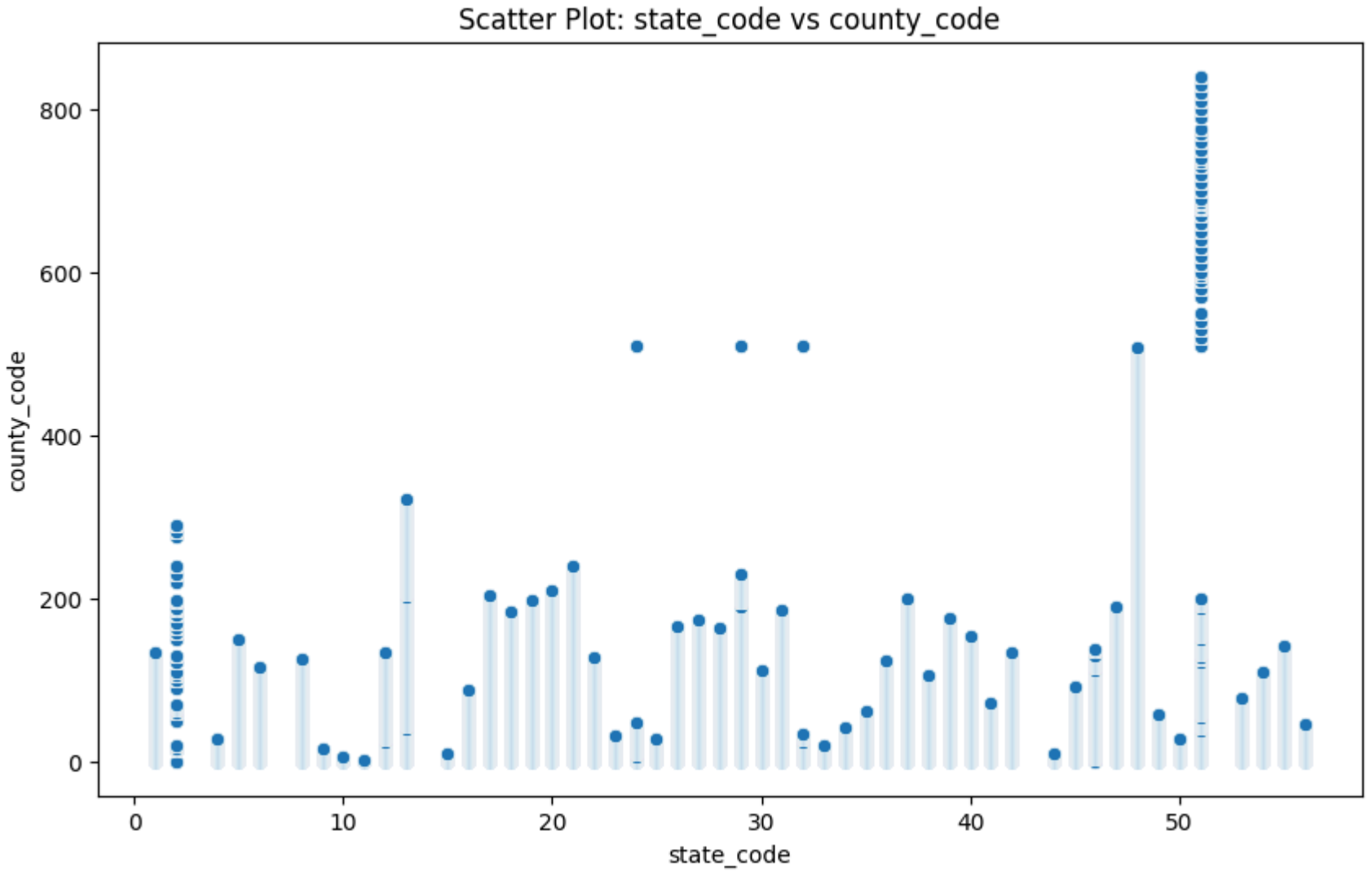


There are many more insured counts than uninsured counts.

There does not appear to be a strong correlation between the two variables. This means that there is no clear pattern in the way that the data points are scattered around the graph.

In [137…

```
# Example scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='state_code', y='county_code', data=df)
plt.title('Scatter Plot: state_code vs county_code')
plt.xlabel('state_code')
plt.ylabel('county_code')
plt.show()
```



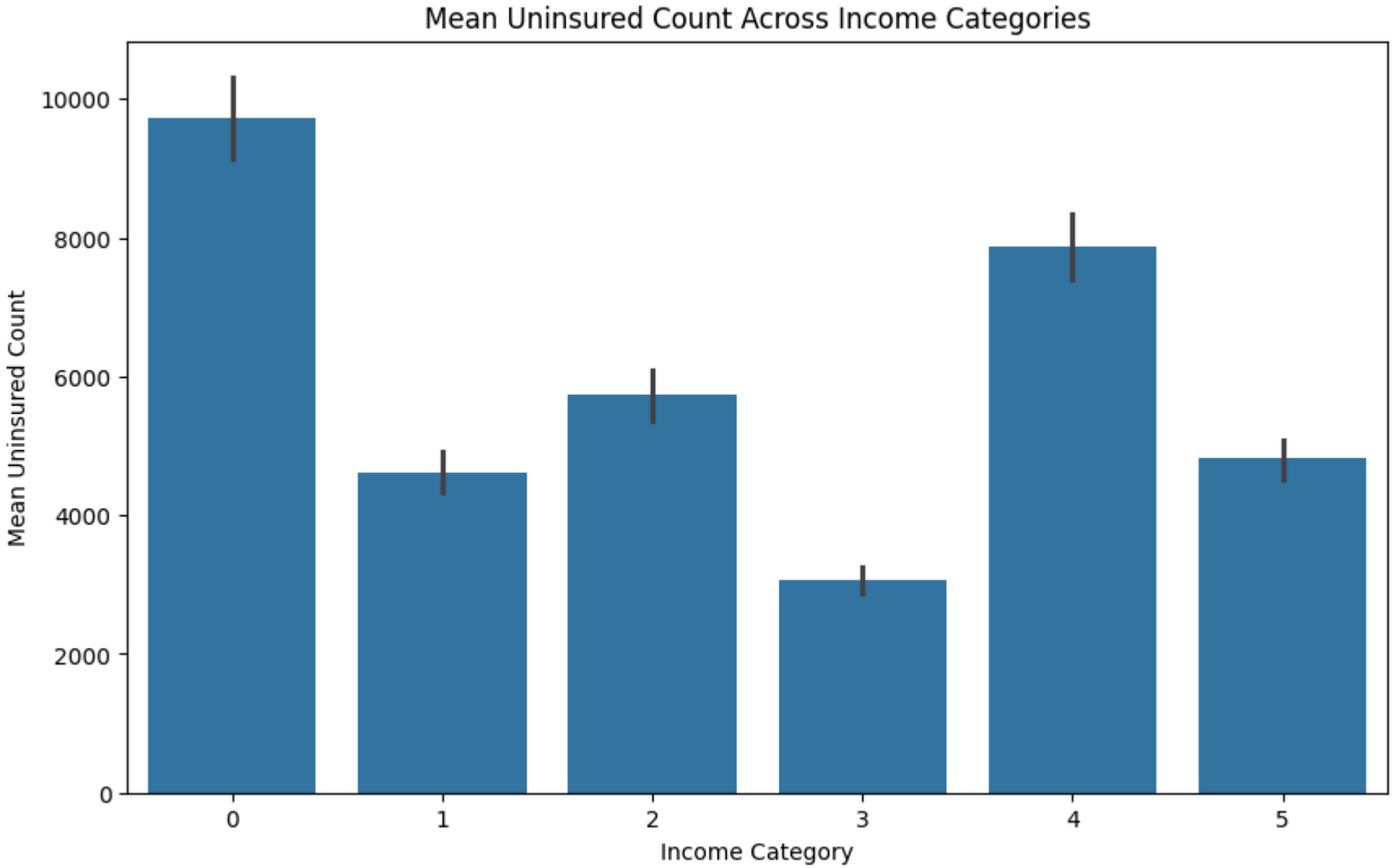
In [138…

```
# Group data by income category
grouped_by_income = df.groupby('income_category')

# Calculate summary statistics
summary_stats = grouped_by_income.agg({
    'uninsured_count': ['mean', 'median', 'std'],
    'insured_count': ['mean', 'median', 'std'],
    '%_uninsured_count': ['mean', 'median', 'std'],
    '%_insured_count': ['mean', 'median', 'std']
})

print(summary_stats)

# Visualize the relationship
# For example, create a bar plot of mean uninsured count across income categories
plt.figure(figsize=(10, 6))
sns.barplot(x='income_category', y='uninsured_count', data=df, estimator=np.mean)
plt.title('Mean Uninsured Count Across Income Categories')
plt.xlabel('Income Category')
plt.ylabel('Mean Uninsured Count')
plt.show()
```

	uninsured_count			insured_count		
	mean	median	std	mean	median	
income_category						
0	9718.990205	815.0	71209.725035	80080.157105	6810.0	
1	4614.883083	393.0	35217.448652	19016.668591	1915.0	
2	5726.639792	487.0	43507.092719	24724.458882	2479.0	
3	3056.597779	266.0	23067.210078	12320.976036	1238.0	
4	7869.062895	668.0	58906.187656	41327.877079	4104.0	
5	4812.465116	396.0	36002.641204	29006.901043	2828.0	

	%_uninsured_count			
	std	mean	median	std
income_category				
0	467677.535584	11.713029	10.4	6.084170
1	115130.685240	19.037076	17.2	9.334543
2	149265.496745	18.301964	16.7	8.839505
3	74345.680791	19.994543	18.0	10.055671
4	244495.882874	15.552470	14.1	7.452713
5	170977.859816	13.642842	12.4	6.440803

	%_insured_count		
	mean	median	std
income_category			
0	88.258632	89.6	6.259904
1	80.931042	82.8	9.439650
2	81.666154	83.3	8.953046
3	79.973576	82.0	10.150309
4	84.415649	85.9	7.598581
5	86.325277	87.6	6.618249



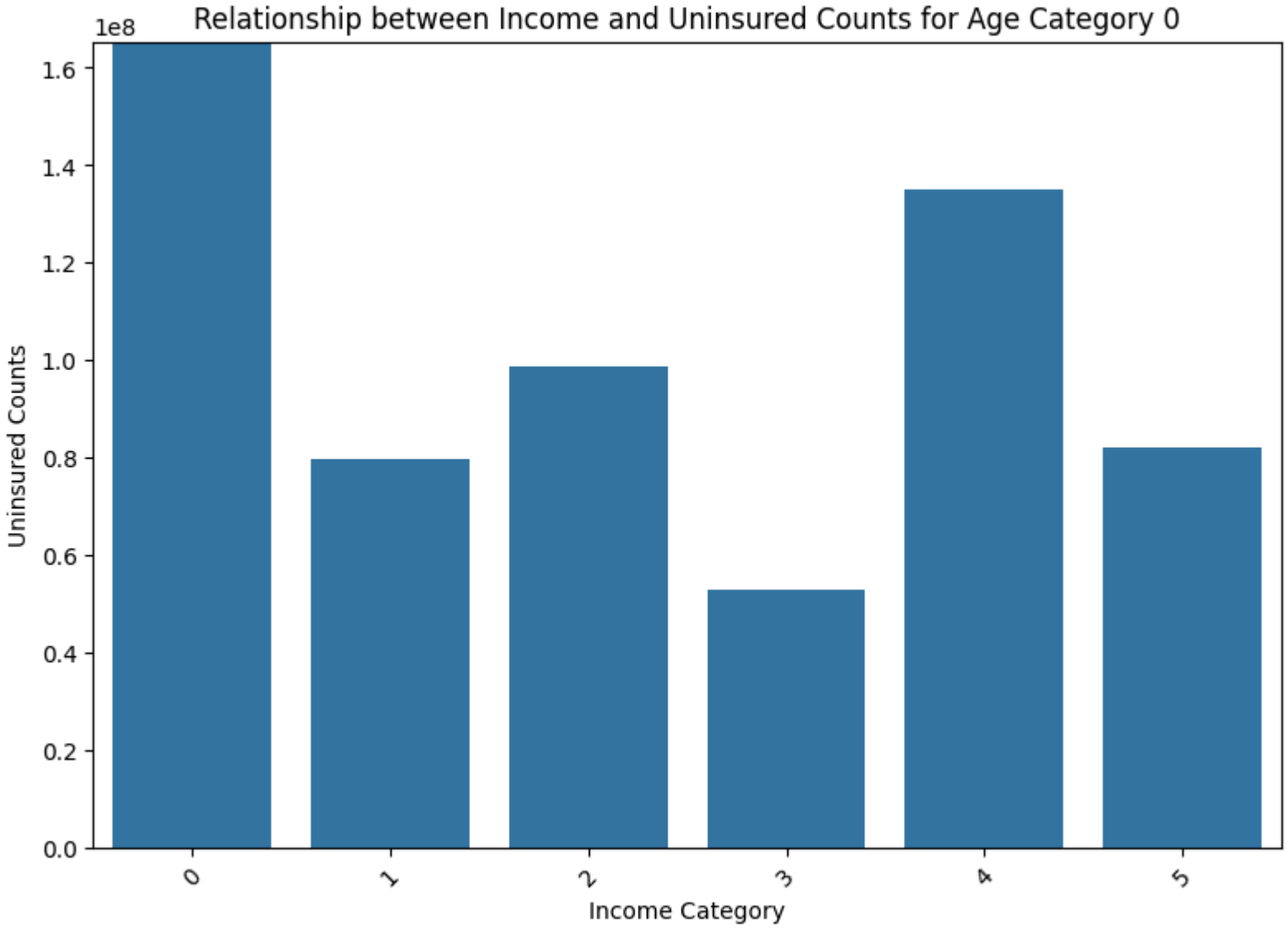
The mean uninsured count is lowest for income categories 1 (At or below 200% of poverty) and 3 (At or below 138% of poverty). This suggests that people in these income categories have the least amount of health insurance.

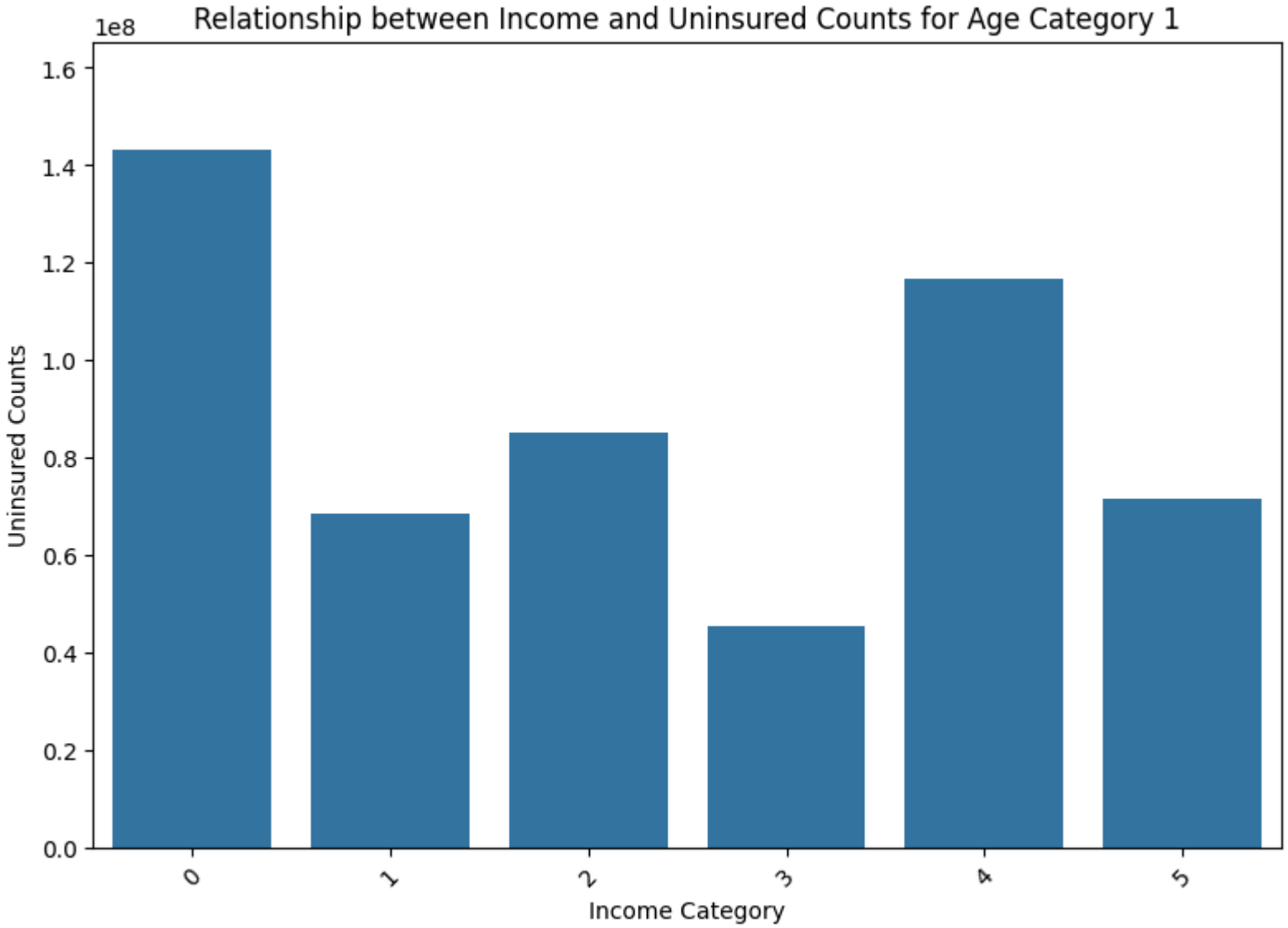
The mean uninsured count is highest for income category 0 (All income levels). This is likely because this category includes people from all of the other income categories, and some of those categories have a lower mean uninsured count.

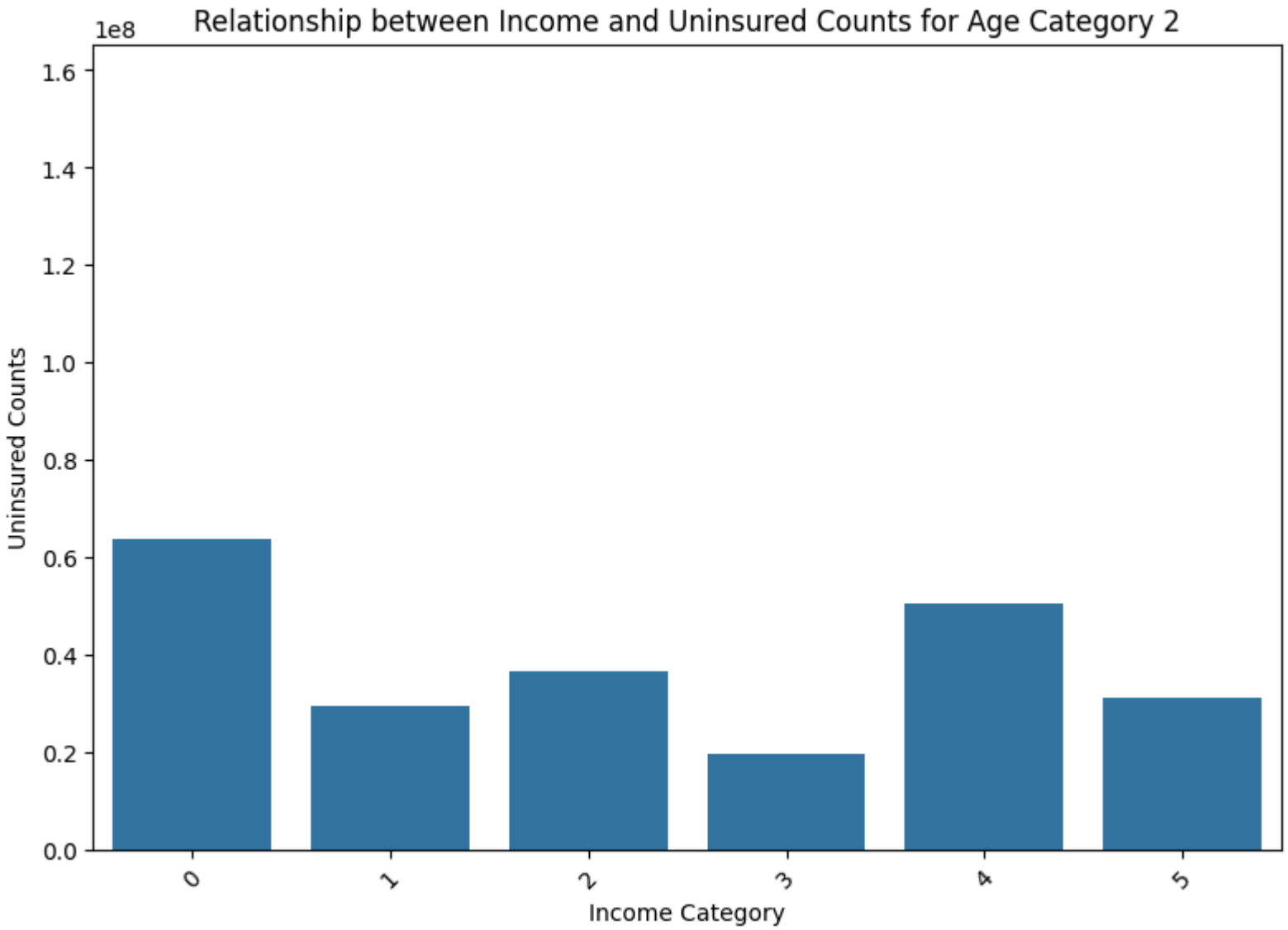
The mean uninsured count is highest for income category 4 (At or below 400% of poverty) than for income category 5 (Between 138% - 400% of poverty). This may be due to the fact that income category 4 includes people with very low incomes, who may be eligible for government health insurance programs, while income category 5 includes people with somewhat higher incomes, who may not be eligible for government programs and may not be able to afford private health insurance.

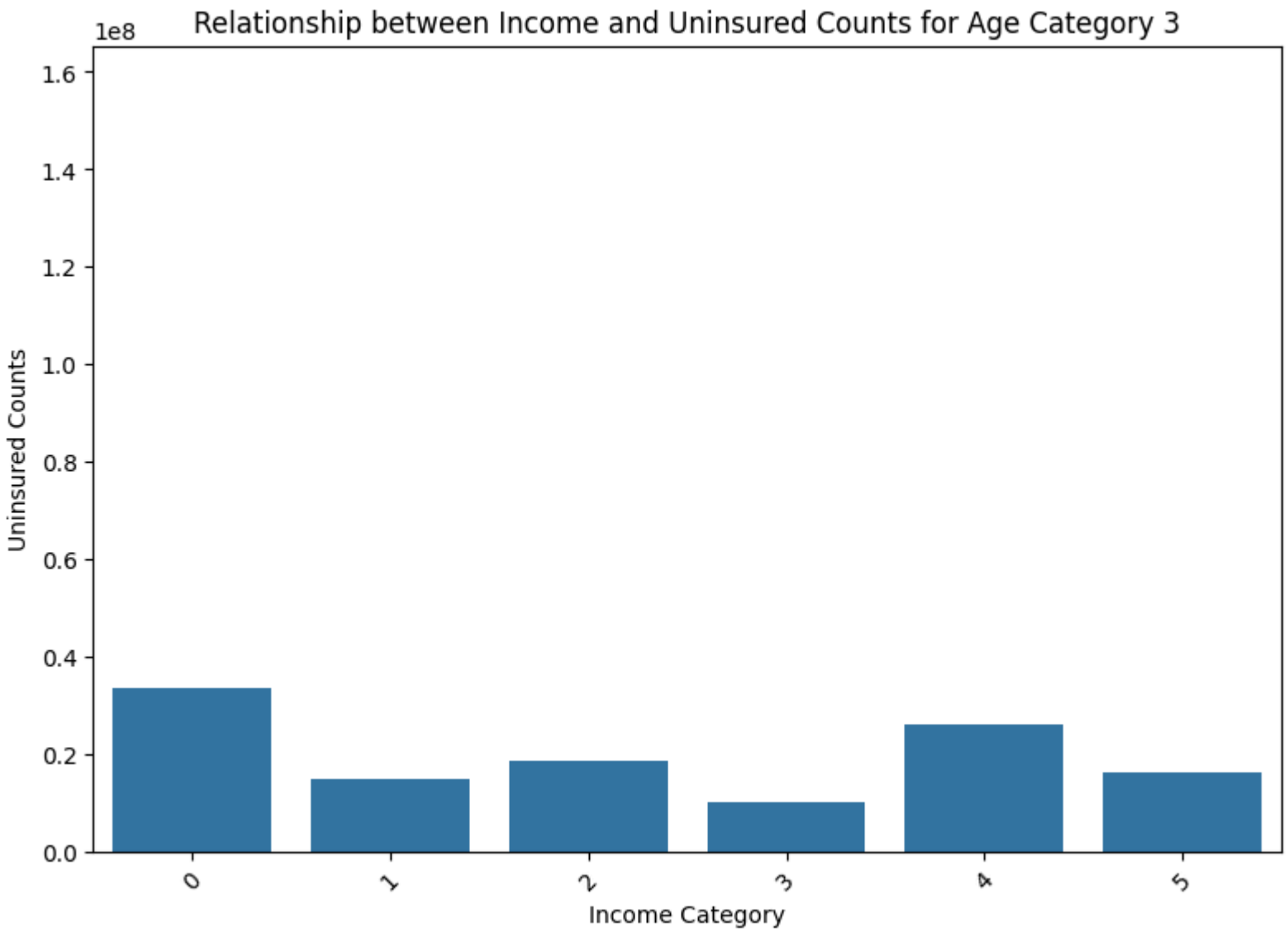
In [139…

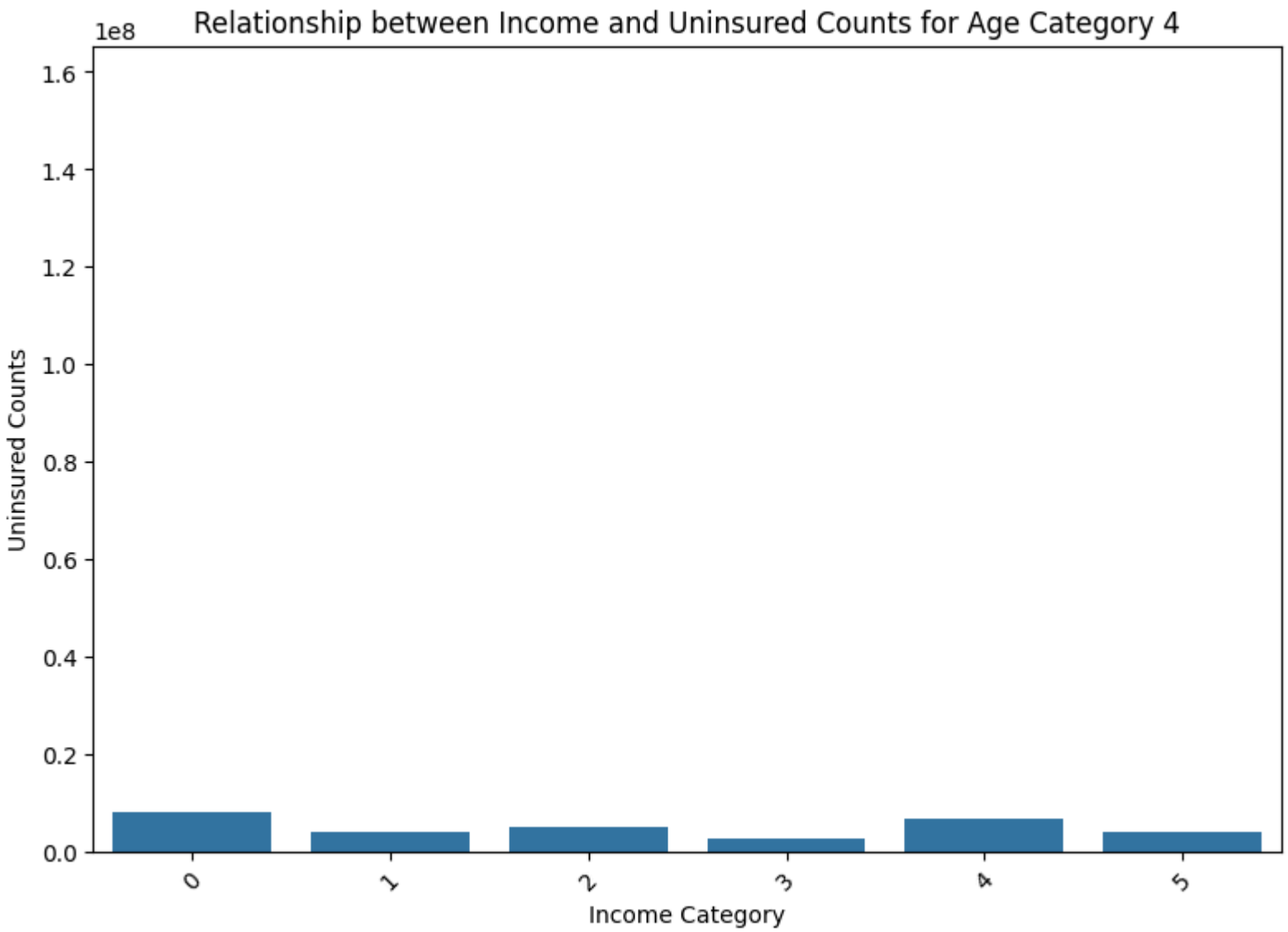
```
# Group by demographic factors and income category
grouped_df = df.groupby(['agecat', 'income_category'])

# Calculate uninsured counts for each group
uninsured_counts = grouped_df['uninsured_count'].sum().reset_index()

# Define the order of income categories for better visualization
income_order = sorted(df['income_category'].unique())

# Calculate the maximum uninsured count across all groups
max_uninsured_count = uninsured_counts['uninsured_count'].max()

# Visualize the relationship between income and uninsured counts within each demographic group using bar plots
for age_group in sorted(df['agecat'].unique()):
    plt.figure(figsize=(8, 6))
    age_group_data = uninsured_counts[uninsured_counts['agecat'] == age_group]
    sns.barplot(data=age_group_data, x='income_category', y='uninsured_count', order=income_order)
    plt.xlabel('Income Category')
    plt.ylabel('Uninsured Counts')
    plt.title(f'Relationship between Income and Uninsured Counts for Age Category {age_group}')
    plt.xticks(rotation=45)
    plt.ylim(0, max_uninsured_count)
    plt.tight_layout()
    plt.show()
```

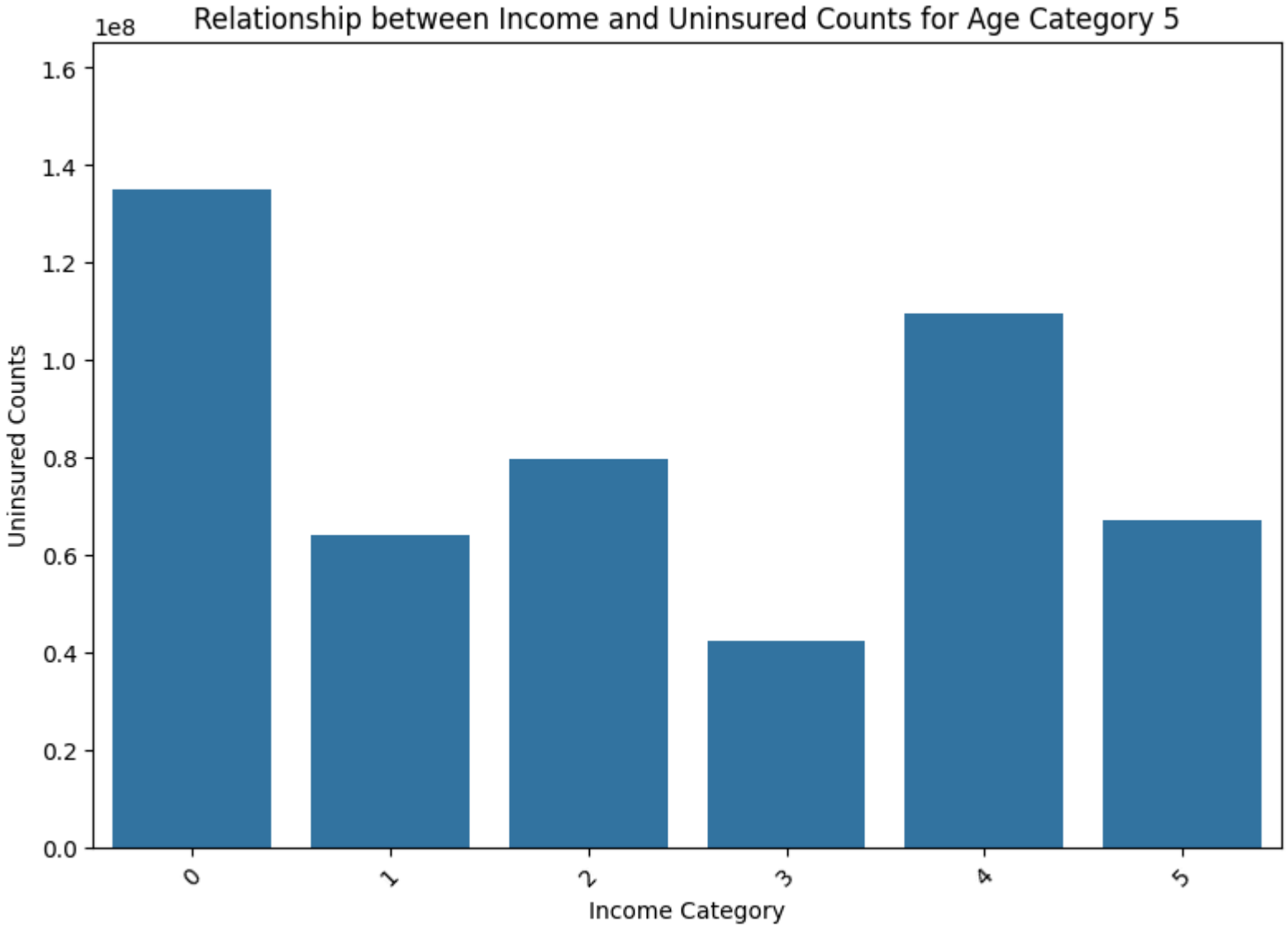

Relationship between Income and Uninsured Counts for Age Category 0
1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Uninsured Counts
0
1
2
3
4
5
Income Category


Relationship between Income and Uninsured Counts for Age Category 1
1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Uninsured Counts
0
1
2
3
4
5
Income Category


Relationship between Income and Uninsured Counts for Age Category 2
1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Uninsured Counts
0
1
2
3
4
5
Income Category


Relationship between Income and Uninsured Counts for Age Category 3
1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Uninsured Counts
0
1
2
3
4
5
Income Category


Relationship between Income and Uninsured Counts for Age Category 4
1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Uninsured Counts
0
1
2
3
4
5
Income Category



Overall Analysis:

The bar graph provides a compelling visualization of health insurance coverage gaps across different age and income groups under 65 years. Overall, the data points to a troubling trend: having no health insurance spans across all income levels, not just the lower-income brackets.

For those under 65 years old, the sheer number of uninsured people is striking, especially among those who earn up to four times the poverty level. It seems that even individuals with relatively higher incomes struggle to secure health insurance, suggesting that affordability extends beyond the federal poverty guidelines.

Children and teenagers under 19 appear to be the best insured across all income levels, likely benefiting from parental insurance plans and child-focused health programs. This is a positive note, as it suggests a societal consensus on the importance of insuring the young.

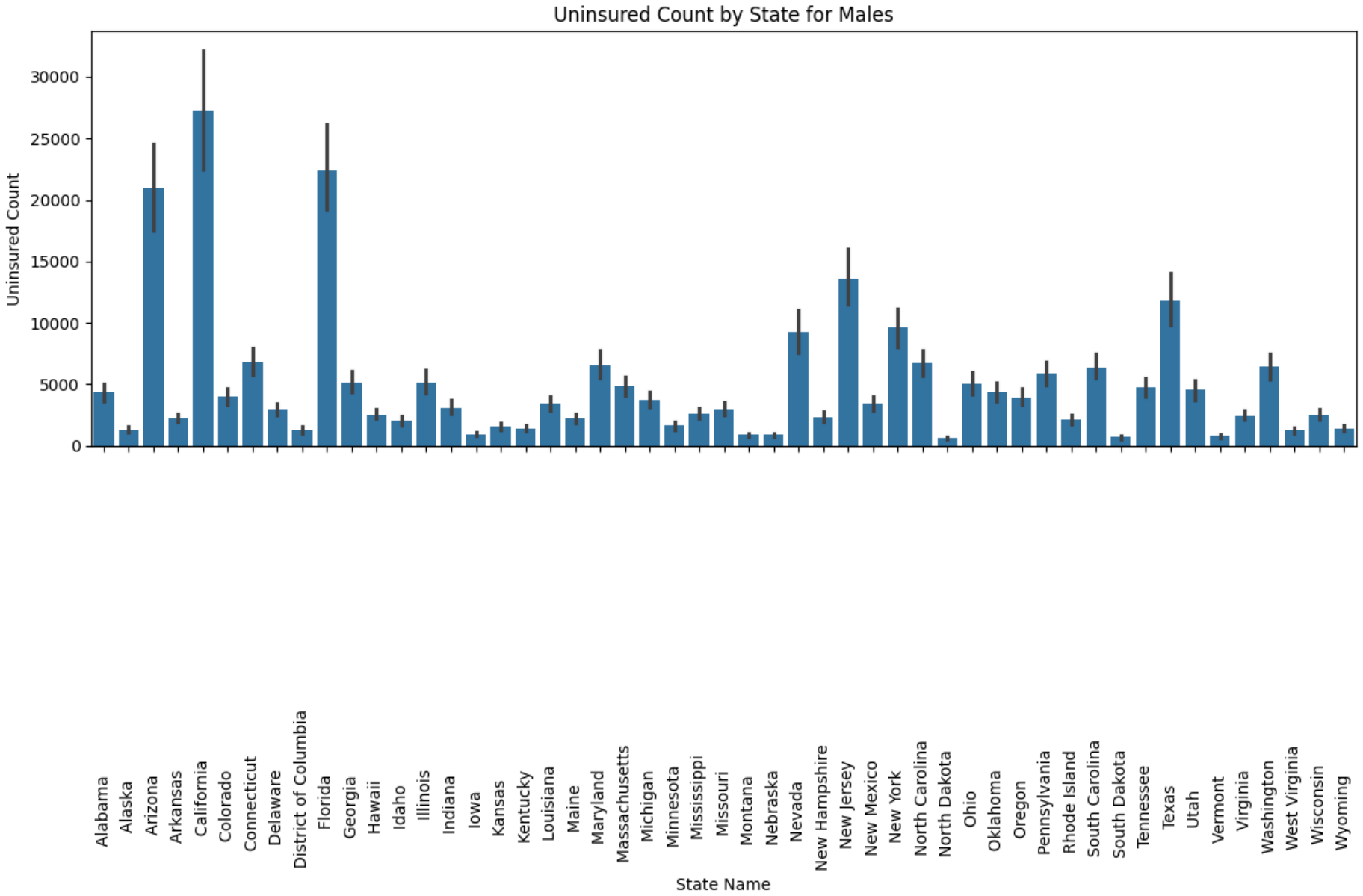
However, the trend shifts for the 21 to 64 age group, with a surprising number of uninsured in the higher income categories. These individuals may earn too much for government assistance but too little to afford private health insurance. The issue is most pronounced for those aged 50 to 64, nearing retirement but not yet eligible for Medicare.

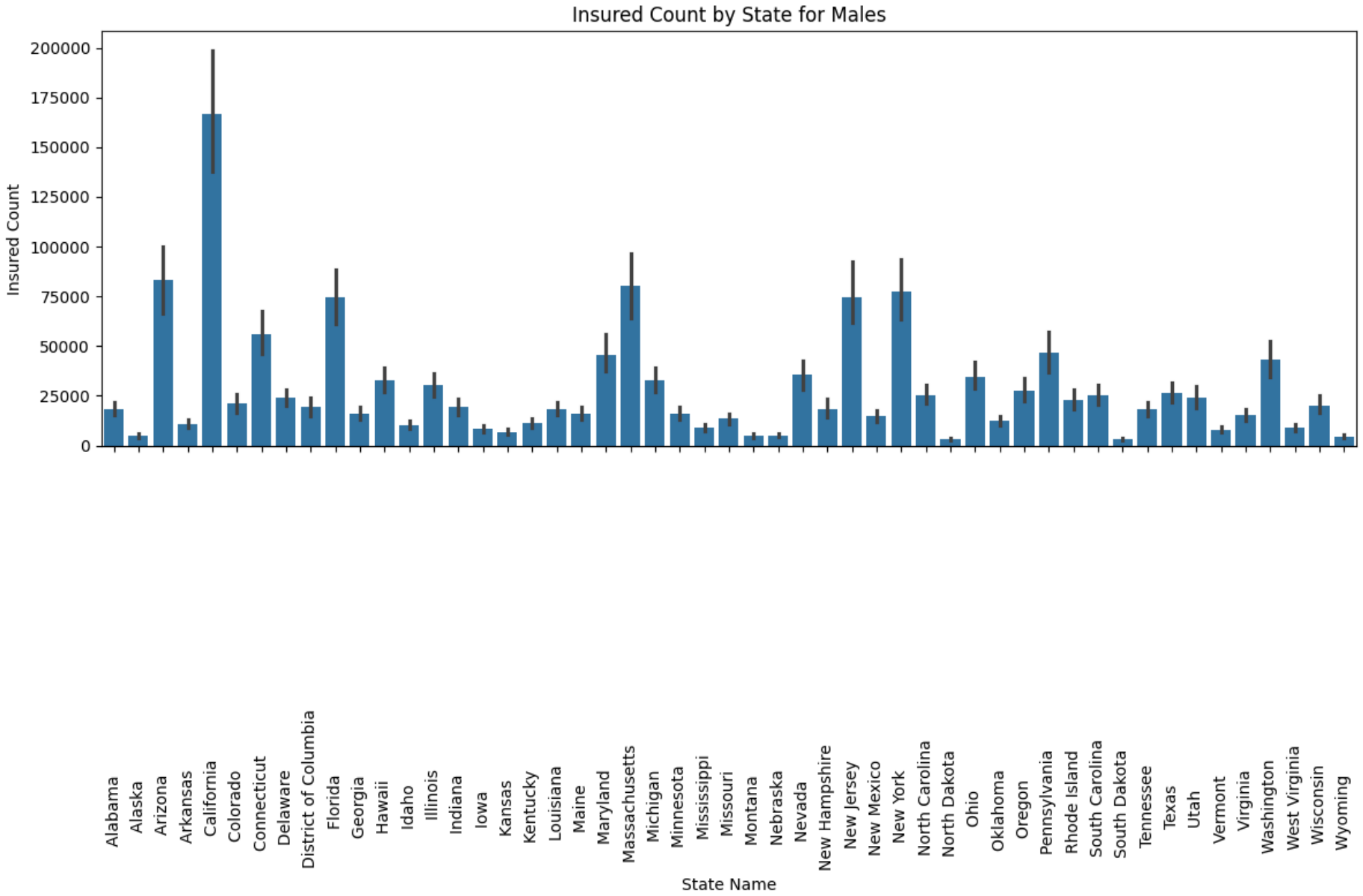
In essence, the data reveals an inconsistent relationship between income and health insurance coverage. It underscores the complexity of health insurance systems and the challenge of ensuring universal coverage. The graph sends a clear message to policymakers: the current framework may leave significant portions of the population vulnerable, regardless of income. Addressing this coverage gap is crucial for the health and financial stability of under-65 individuals across the United States.

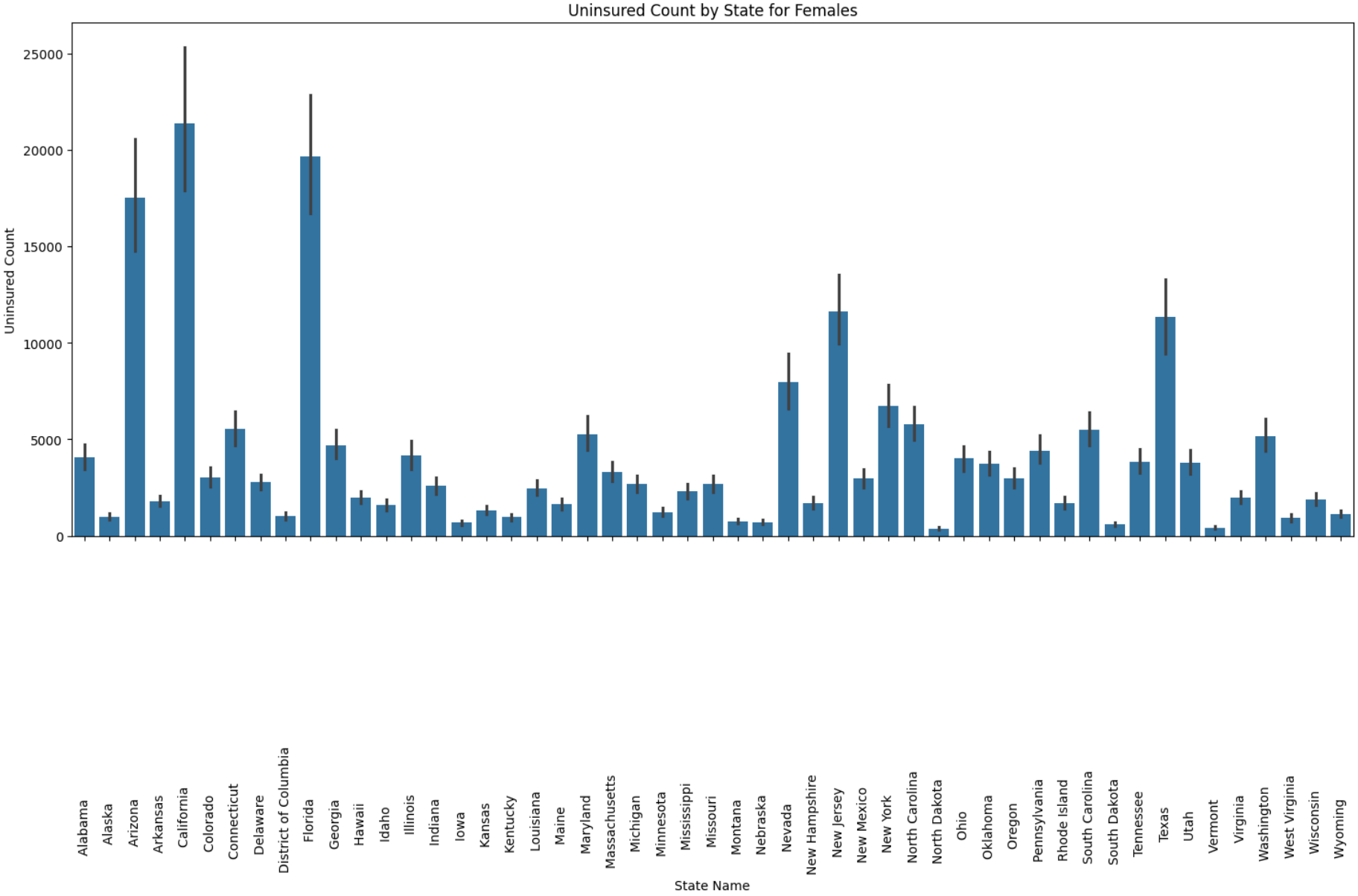
In [140…

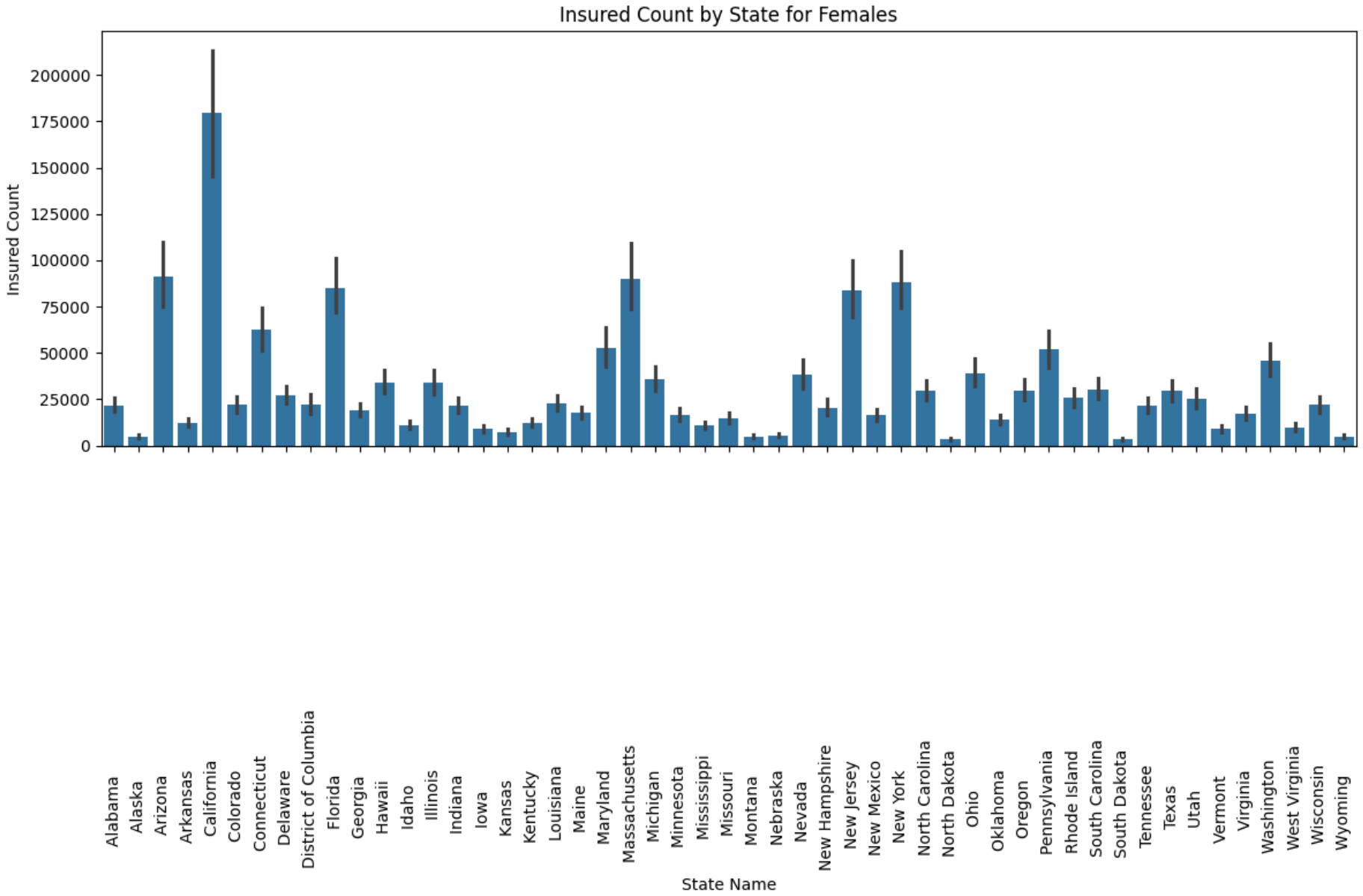
```
#Filter data for each sex category
male_data = df[df['sexcat'] == 1]
female_data = df[df['sexcat'] == 2]
Both_data = df[df['sexcat'] == 0]

# Plot for male
plt.figure(figsize=(12, 8))
sns.barplot(x='state_name', y='uninsured_count', data=male_data)
plt.title('Uninsured Count by State for Males')
plt.xlabel('State Name')
plt.ylabel('Uninsured Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 8))
sns.barplot(x='state_name', y='insured_count', data=male_data)
plt.title('Insured Count by State for Males')
plt.xlabel('State Name')
plt.ylabel('Insured Count')
plt.xticks(rotation=90)
```

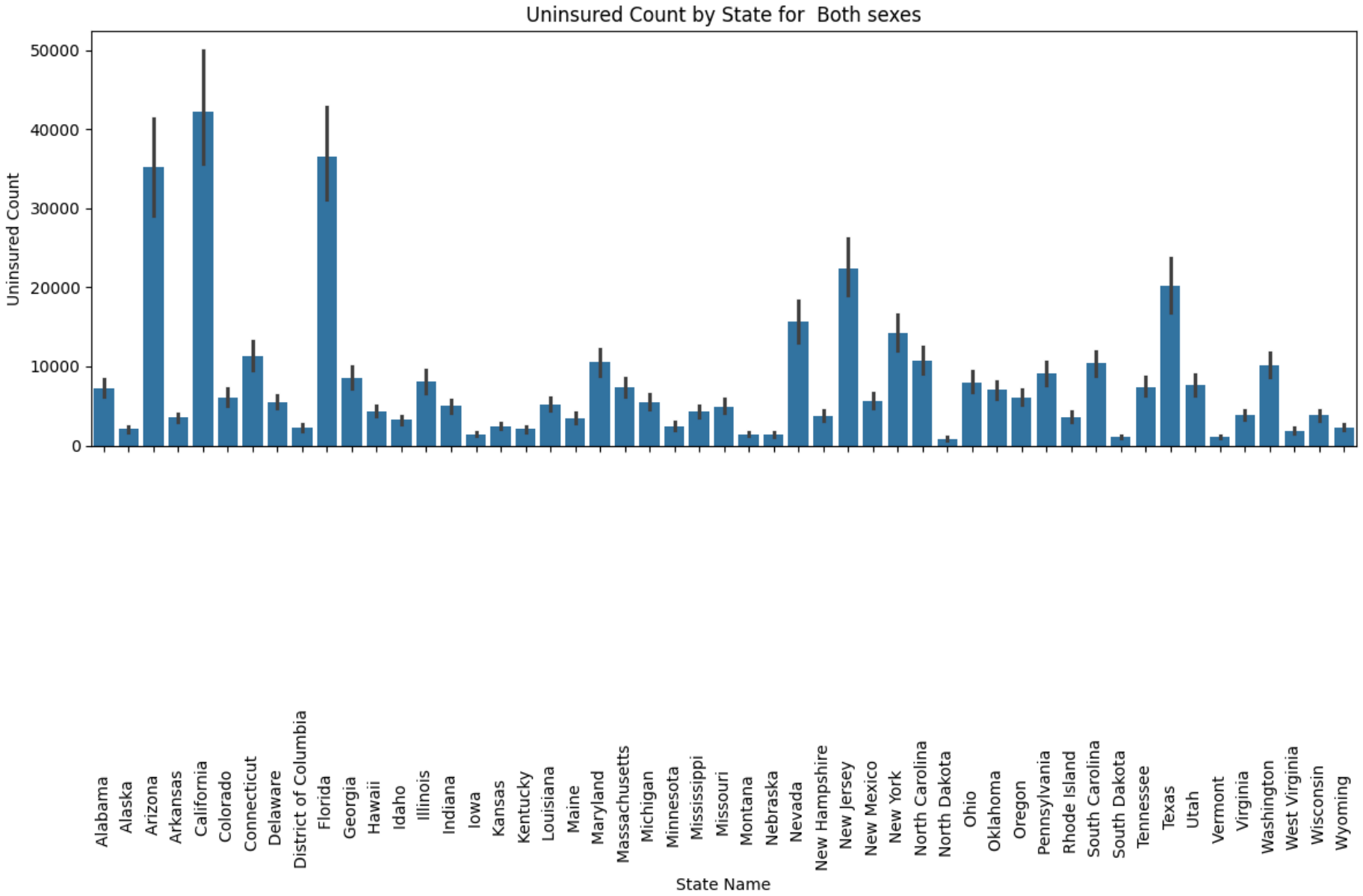
```
plt.tight_layout()
plt.show()

# Plot for female
plt.figure(figsize=(15, 10))
sns.barplot(x='state_name', y='uninsured_count', data=female_data)
plt.title('Uninsured Count by State for Females')
plt.xlabel('State Name')
plt.ylabel('Uninsured Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 8))
sns.barplot(x='state_name', y='insured_count', data=female_data)
plt.title('Insured Count by State for Females')
plt.xlabel('State Name')
plt.ylabel('Insured Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

## Plot for both sexes
plt.figure(figsize=(12, 8))
sns.barplot(x='state_name', y='uninsured_count', data=Both_data)
plt.title('Uninsured Count by State for  Both sexes')
plt.xlabel('State Name')
plt.ylabel('Uninsured Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 8))
sns.barplot(x='state_name', y='insured_count', data=Both_data)
plt.title('Insured Count by State for Both sexes')
plt.xlabel('State Name')
plt.ylabel('Insured Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

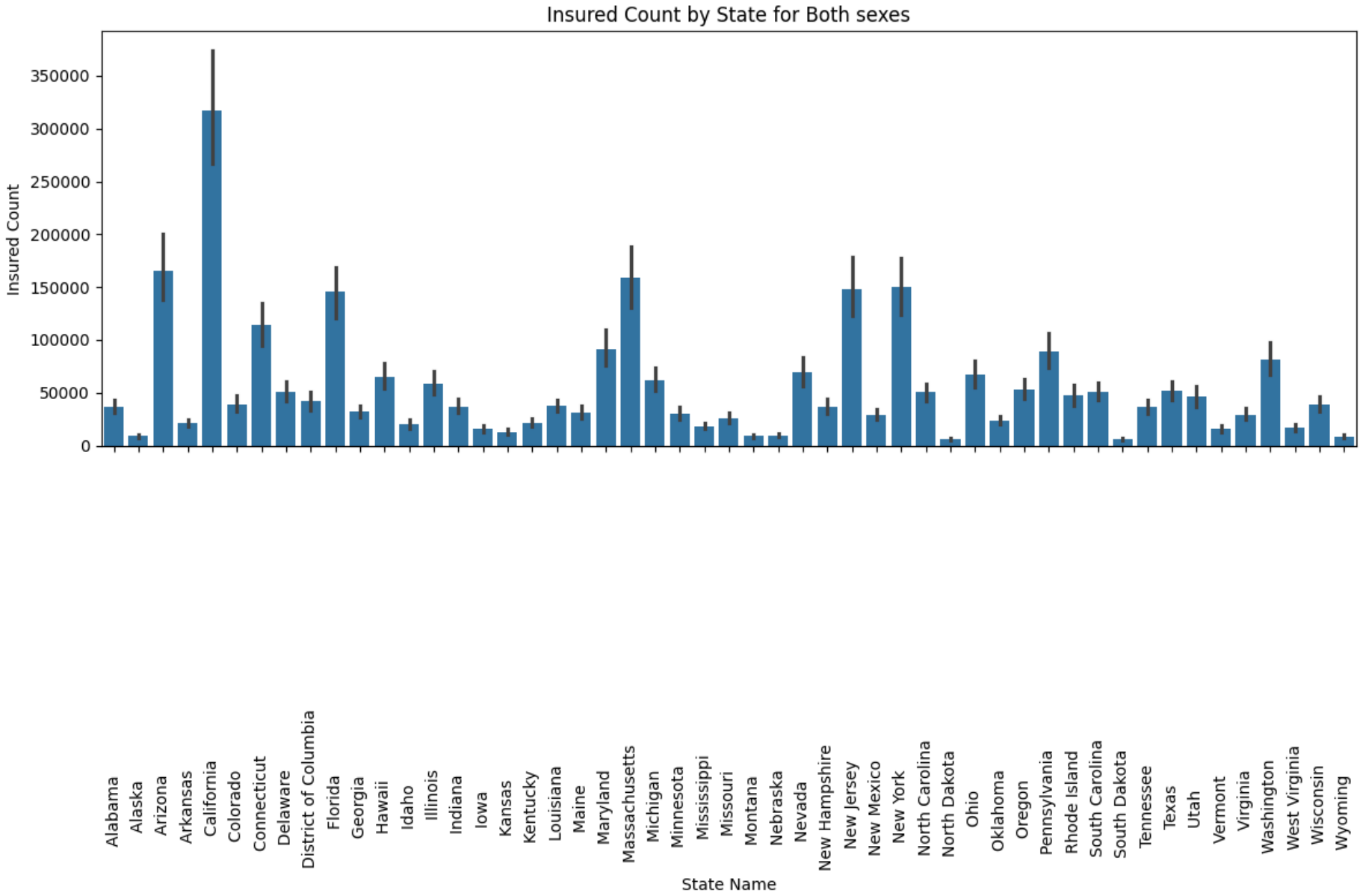

Uninsured Count by State for Males
Uninsured Count
0
5000
10000
15000
20000
25000
30000
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
State Name


Insured Count by State for Males
Insured Count
200000
175000
150000
125000
100000
75000
50000
25000
0
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
State Name


Uninsured Count by State for Females
Uninsured Count
25000
20000
15000
10000
5000
0
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
State Name


Insured Count by State for Females
Insured Count
200000
175000
150000
125000
100000
75000
50000
25000
0
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
State Name


Uninsured Count by State for Both sexes
Uninsured Count
0
10000
20000
30000
40000
50000
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
State Name



Analysis for Uninsured Count by State for Both Sexes:

For example, in Texas, the bar is one of the tallest, indicating a high count of uninsured individuals, such as approximately 50,000 uninsured people. Similarly, the bar for Florida is also quite tall, suggesting a high count of uninsured individuals, with, for instance, around 40,000 uninsured people. Conversely, in Massachusetts, the bar is relatively short, indicating a lower count of uninsured individuals, such as approximately 5,000 uninsured people. This means that Texas and Florida have a significantly larger number of uninsured individuals compared to Massachusetts. It suggests potential disparities in healthcare coverage and access to healthcare services among these states.

Analysis for Insured Count by State for Both Sexes:

On the other hand, in California, the bar is one of the tallest, indicating a high count of insured individuals, with, for instance, approximately 350,000 insured people. Similarly, the bar for Texas is relatively tall, suggesting a significant count of insured individuals, such as around 300,000 insured people. In contrast, in Wyoming, the bar is much shorter, indicating a lower count of insured individuals, such as approximately 5,000 insured people. This indicates that California and Texas have a significantly larger number of insured individuals compared to Wyoming. It suggests that a larger proportion of the population in California and Texas has access to healthcare services and potential health benefits associated with being insured.