Importing the required libraries

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import geopandas as gpd
import seaborn as sns
from pandas.plotting import parallel_coordinates
from matplotlib.patches import Patch
import warnings
warnings.filterwarnings('ignore')

columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num',
'marital-status', 'occupation', 'relationship', 'race', 'sex',
'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
'income']
data = pd.read_csv('data.txt', sep=', ', header=None, names=columns)

# Drop the fnlwgt column
data = data.drop('fnlwgt', axis=1)

# Replace the ? characters with NaN
data = data.replace('?', np.nan)
data.dropna(inplace=True)
```

Planning for Visualizations

Feature Selection

We need to choose the number of features that we are continuing with (Need to be at least 8 variables)

1. Age
2. education
3. Hours per week
4. Maritial Status
5. Sex
6. Relationship Status
7. Capital Loss
8. Capital Gain

User Stories

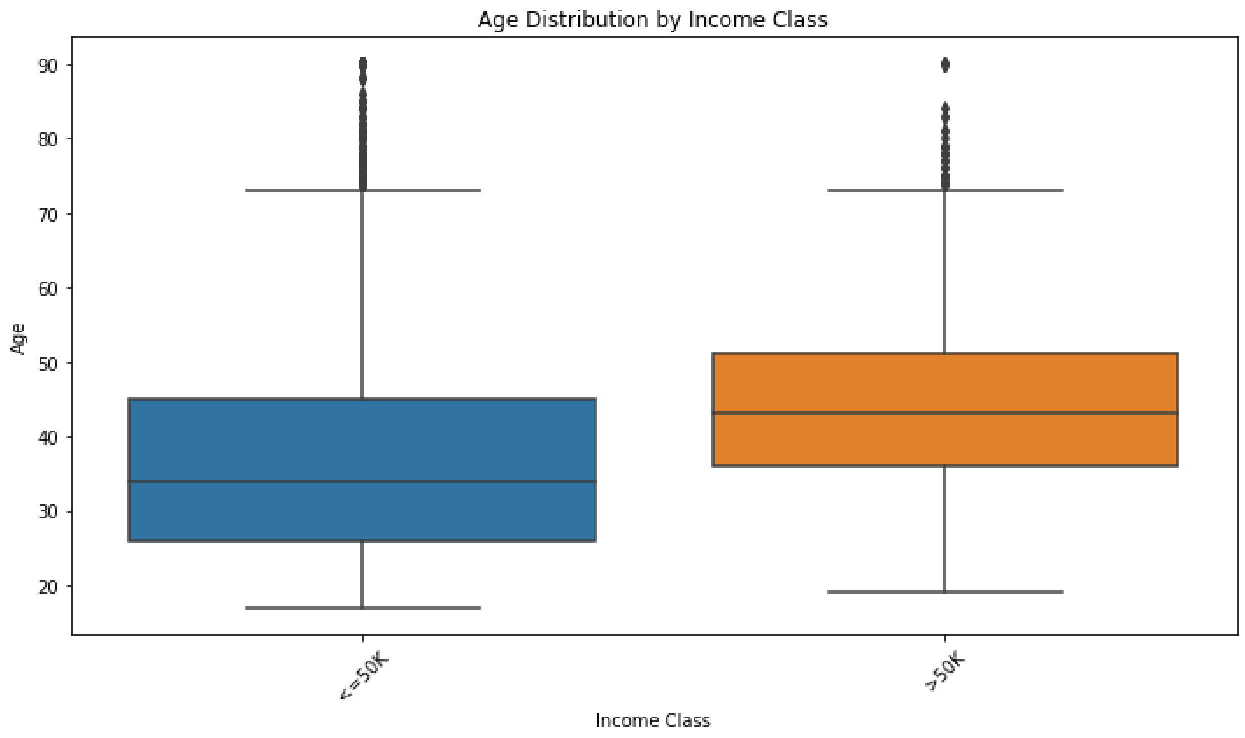
Define the 5 user stories that I want to use for the project. (At least, 3 of them must be mutli-variate)

1. Grouped Bar Chart → (Capital Gain & Capital Loss and Income)
2. Heat Map Plot → (Relationship and Martial Status and Income)
3. Parallel Coordinates Plot → (education and Income)
4. Scatter Plot → (Sex and HPW VS Income)
5. Box and Whisker plot → (Age and Income)

```
# Split the data into the 2 dataframes one for the incomes less than
and equal 50k and the other for the incomes greater than 50k
data_less_50k = data[data['income'] == '<=50K']
data_greater_50k = data[data['income'] == '>50K']
```

Box and Whisker plot → (Age and Income)

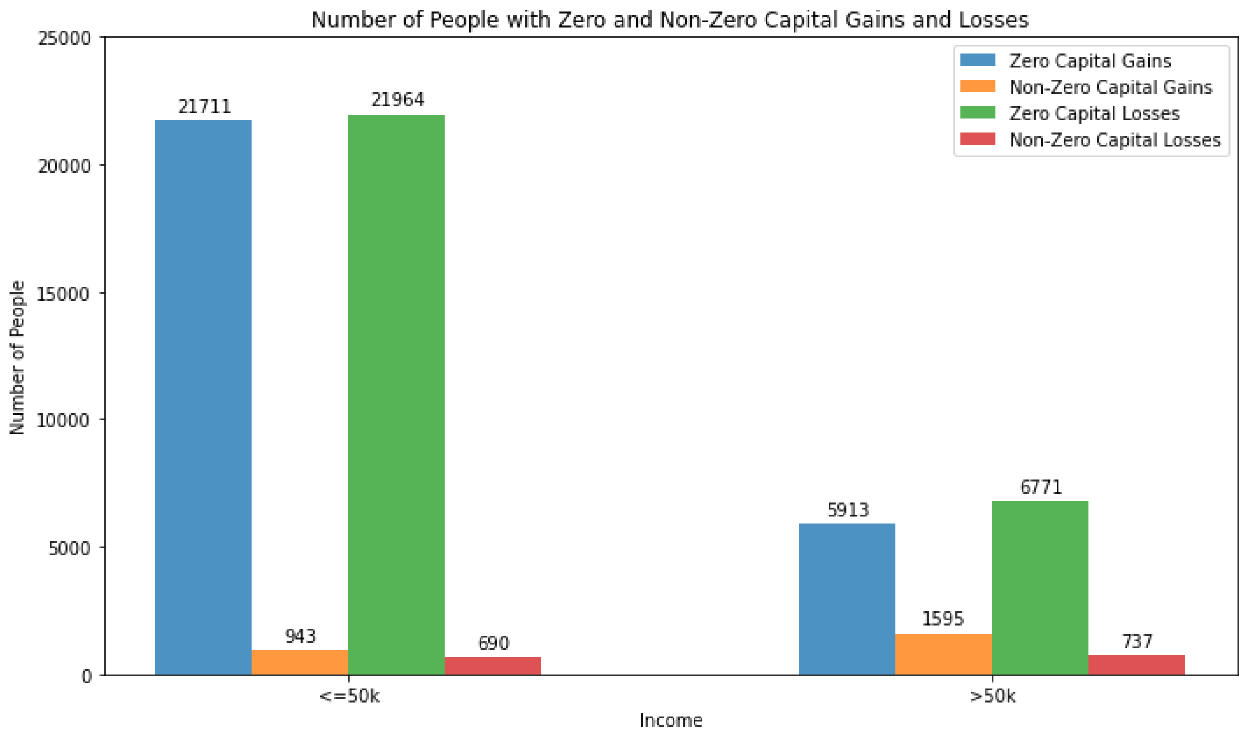
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='income', y='age', data=data)

plt.title('Age Distribution by Income Class')
plt.xlabel('Income Class')
plt.ylabel('Age')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Grouped Bar Chart (Capital Gain & Capital Loss VS. Income)

```
categories = ('<=50k', '>50k')
bars = {'Zero Capital Gains': [0, 0], 'Non-Zero Capital Gains': [0,
0],
        'Zero Capital Losses': [0, 0], 'Non-Zero Capital Losses': [0,
0]}

# Count the number of zero and non-zero capital gains for the incomes
less than and equal 50k
bars['Zero Capital Gains'][0] = data_less_50k[data_less_50k['capital-
gain'] == 0].shape[0]
bars['Non-Zero Capital Gains'][0] =
data_less_50k[data_less_50k['capital-gain'] != 0].shape[0]
bars['Zero Capital Losses'][0] = data_less_50k[data_less_50k['capital-
loss'] == 0].shape[0]
bars['Non-Zero Capital Losses'][0] =
data_less_50k[data_less_50k['capital-loss'] != 0].shape[0]

# Count the number of zero and non-zero capital gains for the incomes
greater than 50k
bars['Zero Capital Gains'][1] =
data_greater_50k[data_greater_50k['capital-gain'] == 0].shape[0]
bars['Non-Zero Capital Gains'][1] =
data_greater_50k[data_greater_50k['capital-gain'] != 0].shape[0]
bars['Zero Capital Losses'][1] =
data_greater_50k[data_greater_50k['capital-loss'] == 0].shape[0]
bars['Non-Zero Capital Losses'][1] =
data_greater_50k[data_greater_50k['capital-loss'] != 0].shape[0]

fig, ax = plt.subplots(figsize=(10, 6))
index = np.arange(len(categories))
bar_width = 0.15
opacity = 0.8

# Calculate positions for each group to ensure they don't overlap
positions = [index + bar_width * i for i in range(len(bars))]

# To store the bar containers for adding labels
bars_plotted = []
for i, (label, counts) in enumerate(bars.items()):
    bar_container = ax.bar(positions[i], counts, bar_width,
alpha=opacity, label=label)
    bars_plotted.append(bar_container)

plt.xlabel('Income')
plt.ylabel('Number of People')
plt.title('Number of People with Zero and Non-Zero Capital Gains and
Losses')
plt.xticks(index + bar_width * len(bars) / 2 - bar_width / 2,
categories)
plt.legend()
ax.set_ylim(0, 25000)  # Adjust as necessary
```

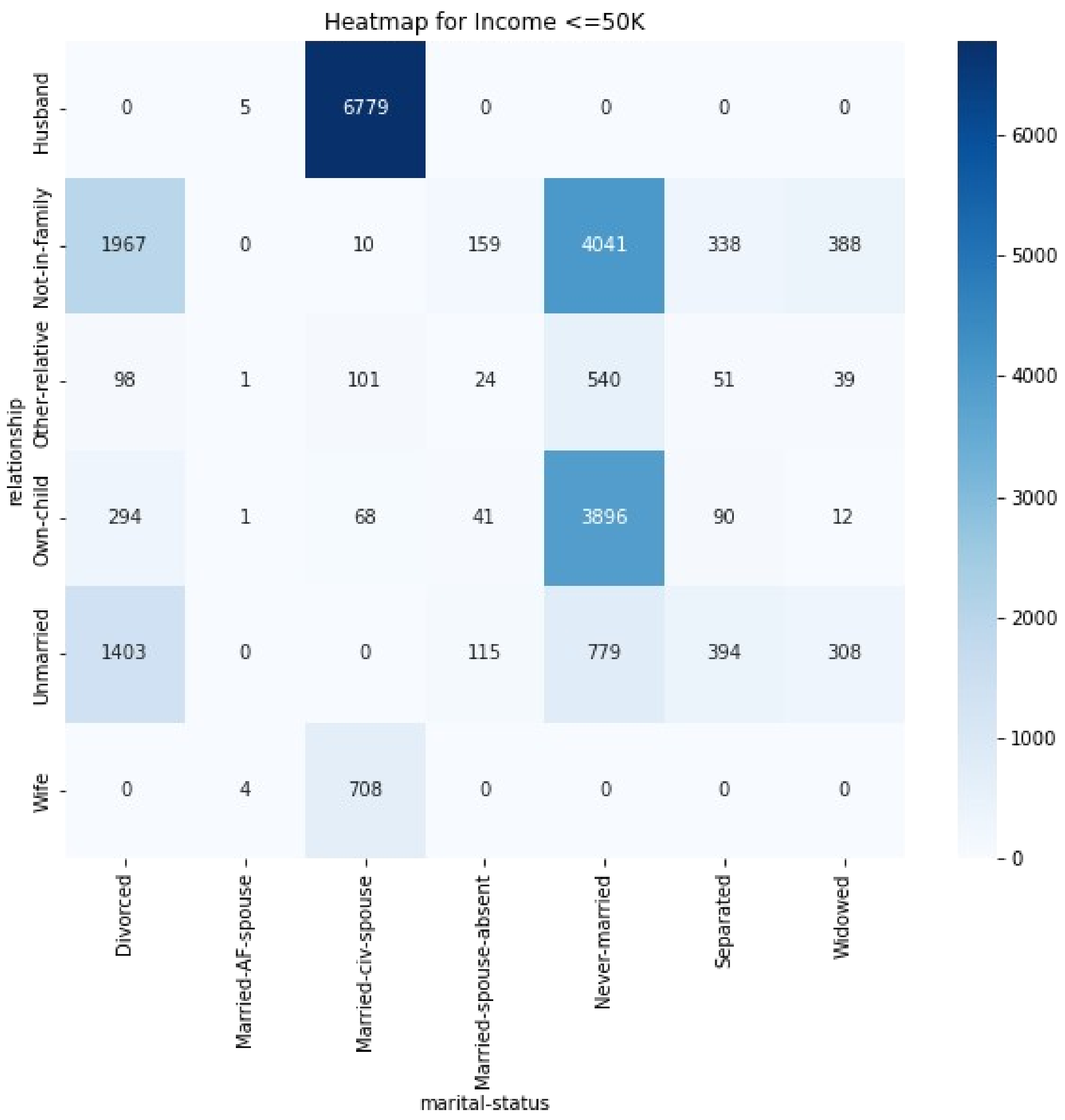
```
# Function to add numbers on top of each bar
def autolabel(bars):
    for bar in bars:
        height = bar.get_height()
        ax.annotate('{}'.format(int(height)),
                    xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

# Apply the function to each bar container
for bars in bars_plotted:
    autolabel(bars)

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

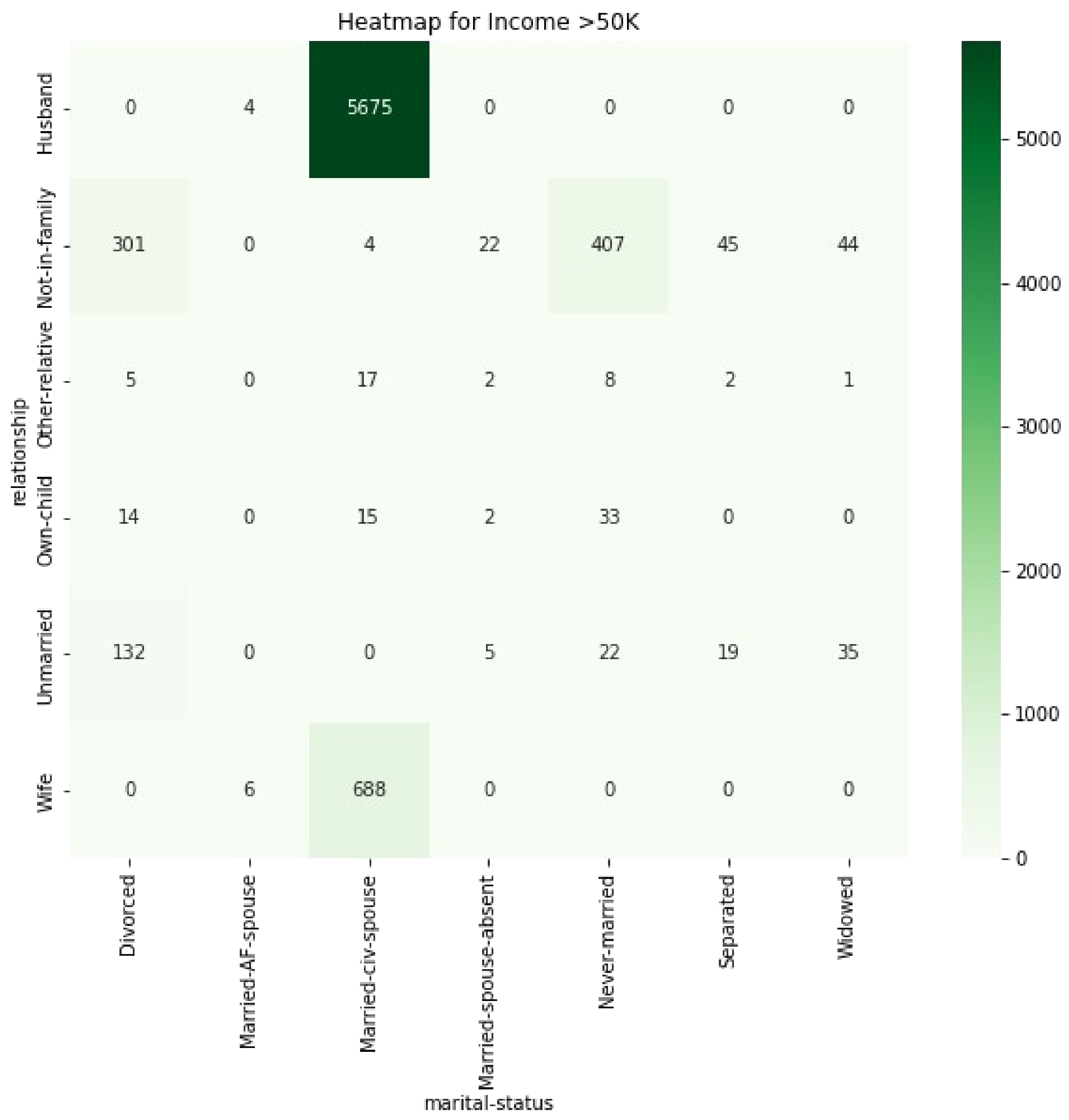


Heat Map Plot → (Relationship and Martial Status and Income)

```
# Copy the DataFrame
df = data.copy()
df['income'] = df['income'].apply(lambda x: 0 if x == "<=50K" else 1)

# Filter the DataFrame for income status 0 and 1
df_income_0 = df[df['income'] == 0]
df_income_1 = df[df['income'] == 1]
```

```python
# Pivot the data for heatmap for income status 0
pivot_0 = df_income_0.pivot_table(index='relationship',
columns='marital-status', values='income', aggfunc='count',
fill_value=0)

# Pivot the data for heatmap for income status 1
pivot_1 = df_income_1.pivot_table(index='relationship',
columns='marital-status', values='income', aggfunc='count',
fill_value=0)

# Check if pivot_0 is empty; if it is, don't attempt to plot it
if not pivot_0.empty:
    # Heatmap for income status 0
    plt.figure(figsize=(10, 8))
    sns.heatmap(pivot_0, annot=True, fmt="d", cmap='Blues')
    plt.title('Heatmap for Income <=50K')
    plt.show()

# Check if pivot_1 is empty; if it is, don't attempt to plot it
if not pivot_1.empty:
    # Heatmap for income status 1
    plt.figure(figsize=(10, 8))
    sns.heatmap(pivot_1, annot=True, fmt="d", cmap='Greens')
    plt.title('Heatmap for Income >50K')
    plt.show()
```


Heatmap for Income <=50K
relationship
Husband
Not-in-family
Other-relative
Own-child
Unmarried
Wife
marital-status
Divorced
Married-AF-spouse
Married-civ-spouse
Married-spouse-absent
Never-married
Separated
Widowed
0 5 6779 0 0 0 0
1967 0 10 159 4041 338 388
98 1 101 24 540 51 39
294 1 68 41 3896 90 12
1403 0 0 115 779 394 308
0 4 708 0 0 0 0
6000
5000
4000
3000
2000
1000
0

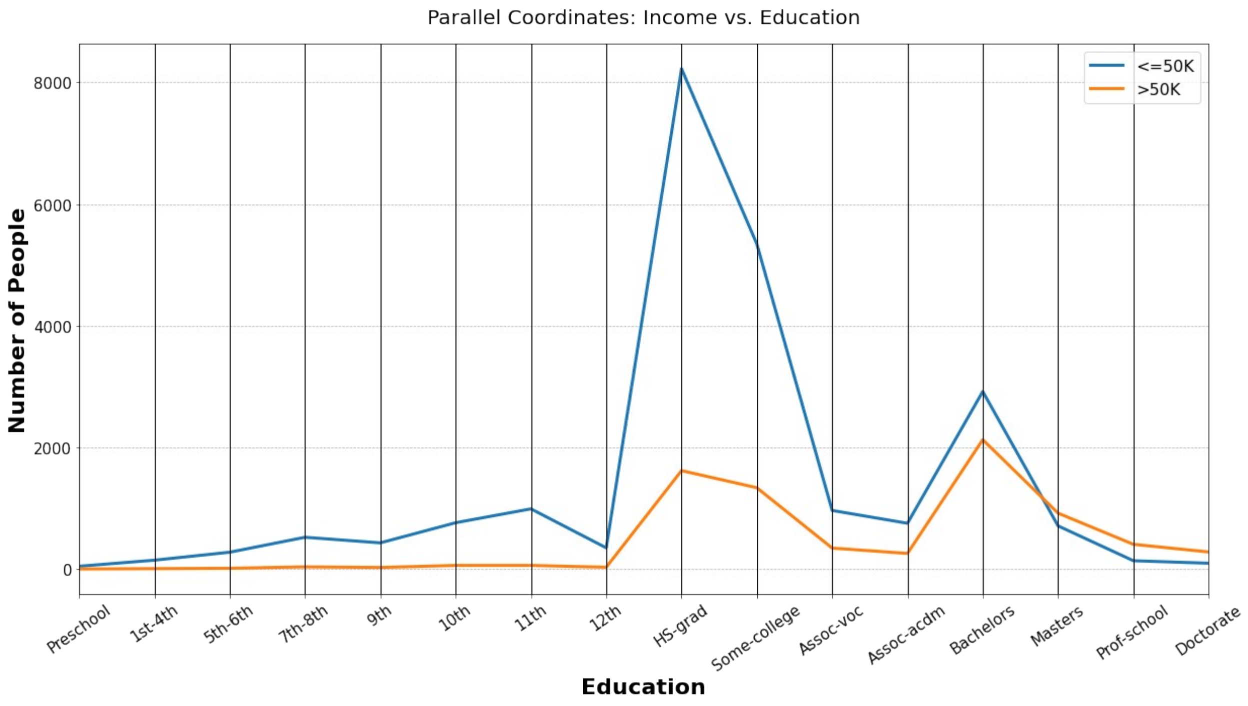


relationship	Divorced	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	Widowed
Husband	0	4	5675	0	0	0	0
Not-in-family	301	0	4	22	407	45	44
Other-relative	5	0	17	2	8	2	1
Own-child	14	0	15	2	33	0	0
Unmarried	132	0	0	5	22	19	35
Wife	0	6	688	0	0	0	0

Parallel Coordinates Plot → (Education and Income)

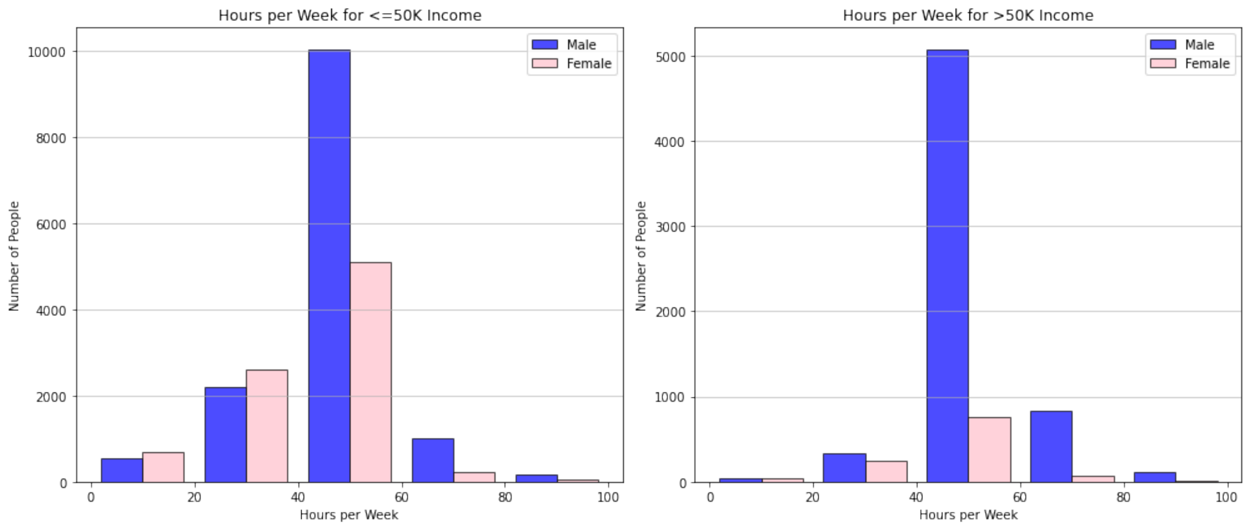
```
# Defining the education levels in order
education = ['Preschool','1st-4th','5th-6th','7th-8th','9th','10th','11th','12th','HS-grad','Some-college','Assoc-voc','Assoc-acdm','Bachelors','Masters','Prof-school','Doctorate']

# Counting the values in each category for the incomes less than and equal 50k
counts_less = data_less_50k['education'].value_counts().to_dict()
counts_greater =
```

```python
data_greater_50k['education'].value_counts().to_dict()

# Making sure that each category has a value in the dictionary
for category in education:
    if category not in counts_less:
        counts_less[category] = 0
    if category not in counts_greater:
        counts_greater[category] = 0

# Sorting the dictionary by the education level
counts_less = {k: counts_less[k] for k in education}
counts_greater = {k: counts_greater[k] for k in education}

# Storing the values in a list which'll be used for plotting
values_less = list(counts_less.values())
values_greater = list(counts_greater.values())

# Adding each list into a dataframe
df_less = pd.DataFrame(data = [values_less], columns= education)
df_greater = pd.DataFrame(data = [values_greater], columns= education)

df_less['income'] = '<=50K'
df_greater['income'] = '>50K'

# Concatenating the two dataframes
df_together = pd.concat([df_less, df_greater])

# Plotting data
plt.figure(figsize=(20,10))
plt.title('Parallel Coordinates: Income vs. Education', fontsize=20,
pad=20)

parallel_coordinates(df_together,'income', color=('#1f77b4',
'#ff7f0e'), linewidth=3)
plt.xlabel("Education", fontsize=22, fontweight='bold')
plt.ylabel("Number of People", fontsize=22, fontweight='bold')
plt.xticks(rotation=35, fontsize=15)
plt.yticks(fontsize=15)
plt.legend(fontsize=16, loc='upper right')
plt.grid(linestyle='--', linewidth=0.7, axis='y')

plt.show()
```



Scatterplot HPW and Sex VS Income

```
# Bins for the histogram
bins = [0, 20, 40, 60, 80, 100]

# Split the dataset by income bracket
low_income_data = data[data['income'] == '<=50K']
high_income_data = data[data['income'] == '>50K']

# Plot histograms for low income bracket
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.hist([low_income_data[low_income_data['sex'] == 'Male']['hours-
per-week'],
          low_income_data[low_income_data['sex'] == 'Female']['hours-
per-week']],
         bins=bins, label=['Male', 'Female'], color=['blue', 'pink'],
edgecolor='black', alpha=0.7)
plt.title('Hours per Week for <=50K Income')
plt.xlabel('Hours per Week')
plt.ylabel('Number of People')
plt.legend()
plt.grid(axis='y', alpha=0.75)

# Plot histograms for high income bracket
plt.subplot(1, 2, 2)
plt.hist([high_income_data[high_income_data['sex'] == 'Male']['hours-
per-week'],
          high_income_data[high_income_data['sex'] == 'Female']
['hours-per-week']],
         bins=bins, label=['Male', 'Female'], color=['blue', 'pink'],
edgecolor='black', alpha=0.7)
plt.title('Hours per Week for >50K Income')
plt.xlabel('Hours per Week')
plt.ylabel('Number of People')
plt.legend()
plt.grid(axis='y', alpha=0.75)

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



Useloss plots & Reason

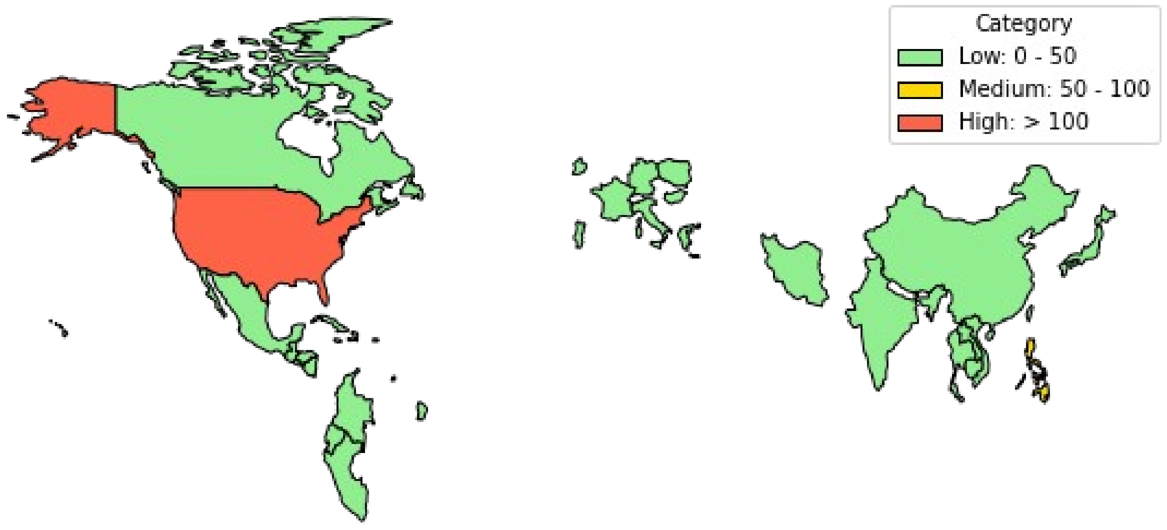
Heat Map (Native Country & Income >50k)

USA has most points, so it doesn't really tell a story.

```
# These are the countries that we are keeping from the GeoDataFrame
countries_to_keep = [
    'United States of America', 'Cambodia', 'Puerto Rico', 'Canada',
'Germany',
    'India', 'Japan', 'Greece', 'China', 'Cuba', 'Iran', 'Honduras',
    'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico',
    'Portugal', 'Ireland', 'France', 'Dominican Republic', 'Laos',
    'Ecuador', 'Taiwan', 'Haiti', 'Colombia', 'Hungary', 'Guatemala',
    'Nicaragua', 'Scotland', 'Thailand', 'Yugoslavia', 'El Salvador',
    'Trinidad and Tobago', 'Peru', 'Hong Kong']

# Load the world map
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

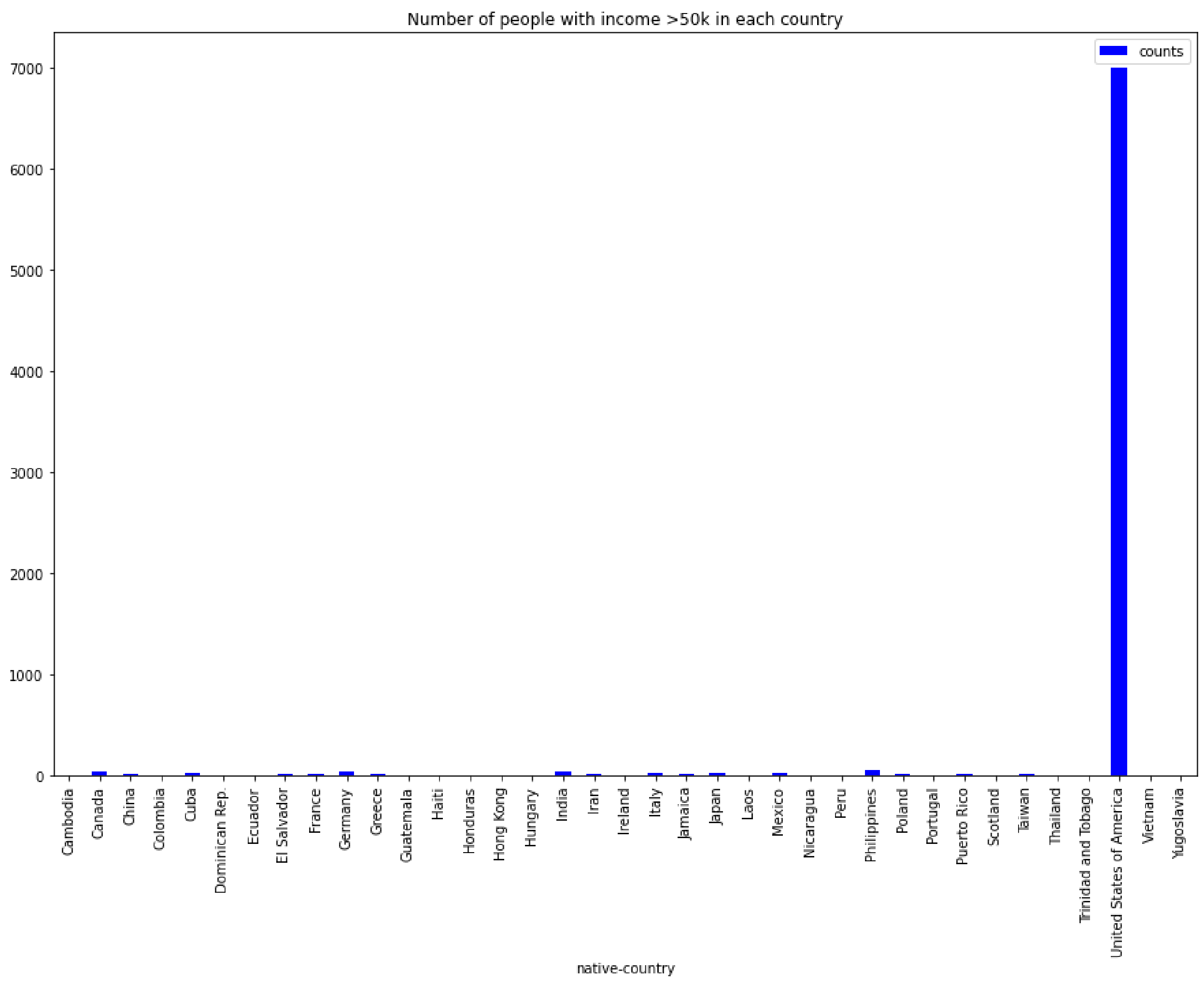
```
# Filter the GeoDataFrame to keep only specified countries
filtered_world = world[world['name'].isin(countries_to_keep)]

# Take the greater than 50k data and group it by the native country
and count the number of people in each country
greater_50k_grouped = data_greater_50k.groupby('native-
country').size().reset_index(name='counts')

# Rename United-States to United States
greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'United-States', 'native-country'] = 'United States of America'
greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'Dominican-Republic', 'native-country'] = 'Dominican Rep.'
greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'Hong', 'native-country'] = 'Hong Kong'
greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'Trinadad&Tobago', 'native-country'] = 'Trinidad and Tobago'
# greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'Holand-Netherlands', 'native-country'] = 'Netherlands'
greater_50k_grouped.loc[greater_50k_grouped['native-country'] == 'El-
Salvador', 'native-country'] = 'El Salvador'
greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'Puerto-Rico', 'native-country'] = 'Puerto Rico'
greater_50k_grouped.loc[greater_50k_grouped['native-country'] ==
'Columbia', 'native-country'] = 'Colombia'

# Drop the rows with the countries that are not in not in the
GeoDataFrame --> South, Outlying-US(Guam-USVI-etc), England
greater_50k_grouped = greater_50k_grouped[greater_50k_grouped['native-
country'] != 'South']
greater_50k_grouped = greater_50k_grouped[greater_50k_grouped['native-
country'] != 'Outlying-US(Guam-USVI-etc)']
greater_50k_grouped = greater_50k_grouped[greater_50k_grouped['native-
country'] != 'England']

# Merging the datasets
filtered_world = filtered_world.rename(columns={'name': 'native-
country'})  # Ensuring column names match for the merge
merged = filtered_world.merge(greater_50k_grouped, on='native-
country', how='left')

import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.patches import Patch

# Example of categorizing into 3 bins with labels
bin_edges = [0, 50, 100, max(merged['counts'].max(), 200)]  # Adjust
the bin edges as needed
bin_labels = ['Low', 'Medium', 'High']
merged['category'] = pd.cut(merged['counts'], bins=bin_edges,
labels=bin_labels, include_lowest=True)

# Define colors for each category
category_colors = {
    'Low': 'lightgreen',
    'Medium': 'gold',
    'High': 'tomato'
}

# Plotting with a larger figure size, black borders, and a custom
legend
fig, ax = plt.subplots(1, 1, figsize=(10, 6))

legend_handles = []  # List to store legend handles

for i, (category, color) in enumerate(category_colors.items()):
    subset = merged[merged['category'] == category]

    # Create a custom label with category name and bin range
    if i < len(bin_labels) - 1:
        custom_label = f"{category}: {bin_edges[i]} -
{bin_edges[i+1]}"
    else:
        # Handle the last category differently if necessary
        custom_label = f"{category}: > {bin_edges[i]}"

    # Plot each category
    subset.plot(ax=ax, label=custom_label, color=color,
edgecolor='black')

    # Create a legend handle for the category
    legend_handles.append(Patch(facecolor=color, edgecolor='black',
label=custom_label))

# Create custom legend
plt.legend(handles=legend_handles, title="Category")
plt.title('Number of people with income >50k in each country')
plt.axis('off')
plt.show()
```

Number of people with income >50k in each country



```python
# Showing that most data points are located in the United States
greater_50k_grouped.plot.bar(x='native-country', y='counts',
color='blue', figsize=(15, 10))
plt.title('Number of people with income >50k in each country')
```

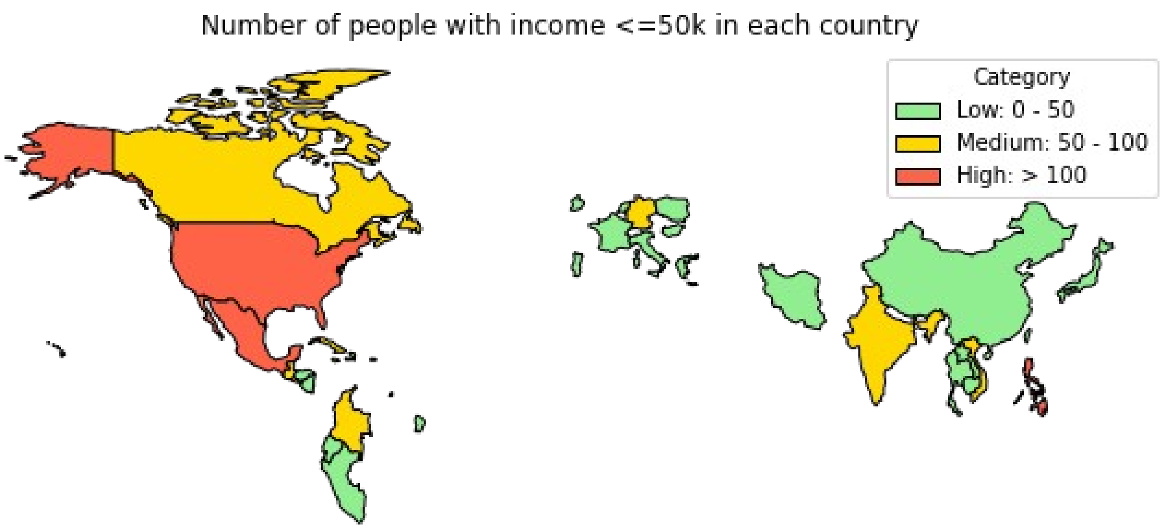
```
Text(0.5, 1.0, 'Number of people with income >50k in each country')
```



Heat Map (Native Country vs Income <=50k)

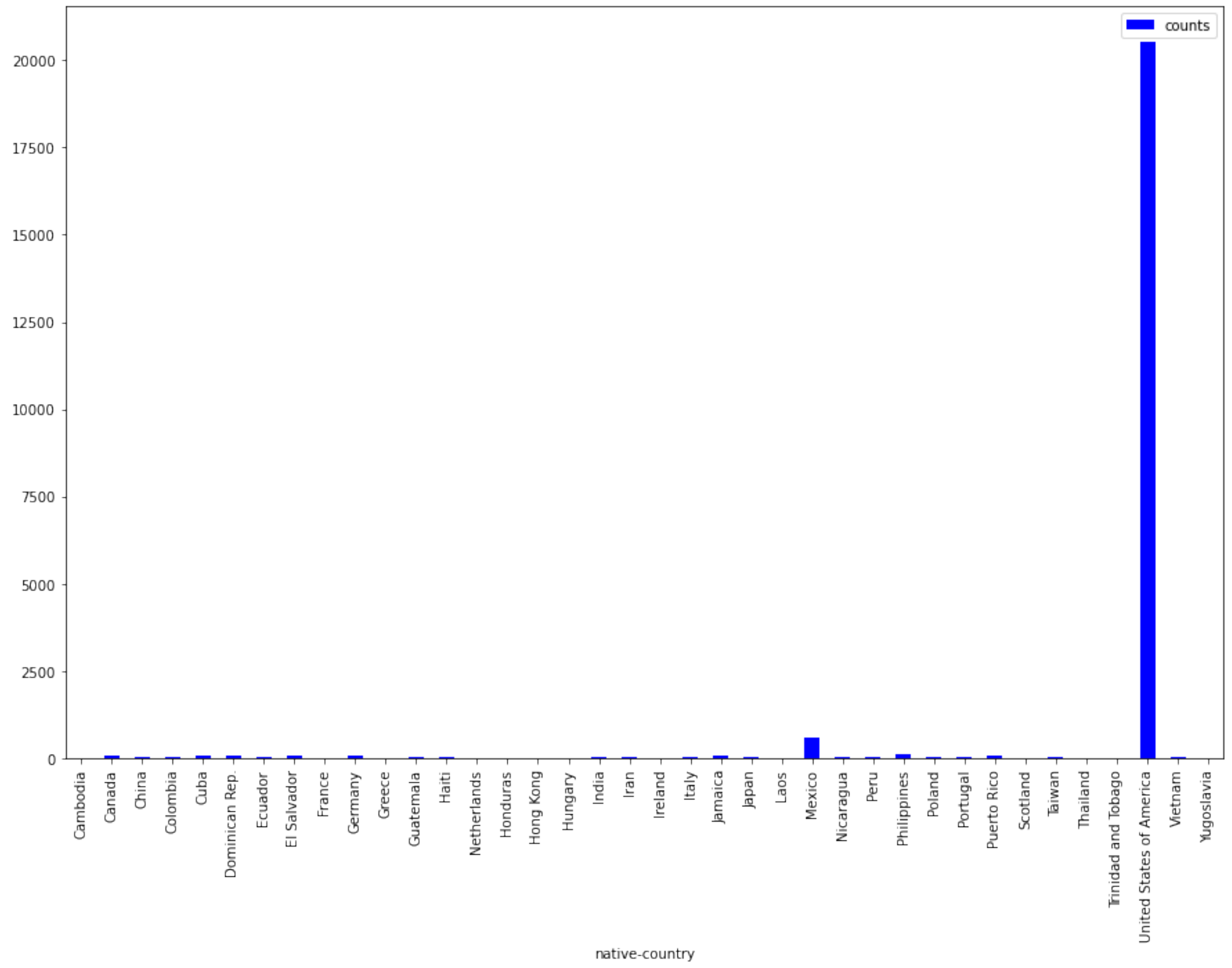
```
# These are the countries that we are keeping from the GeoDataFrame
countries_to_keep = [
    'United States of America', 'Cambodia', 'Puerto Rico', 'Canada',
'Germany',
    'India', 'Japan', 'Greece', 'China', 'Cuba', 'Iran', 'Honduras',
    'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico',
    'Portugal', 'Ireland', 'France', 'Dominican Republic', 'Laos',
    'Ecuador', 'Taiwan', 'Haiti', 'Colombia', 'Hungary', 'Guatemala',
    'Nicaragua', 'Scotland', 'Thailand', 'Yugoslavia', 'El Salvador',
    'Trinidad and Tobago', 'Peru', 'Hong Kong', 'Netherlands']

# Load the world map
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
filtered_world = world[world['name'].isin(countries_to_keep)]

# Take the greater than 50k data and group it by the native country
and count the number of people in each country
```

```
less_50k_grouped = data_less_50k.groupby('native-
country').size().reset_index(name='counts')

# Rename the columns according to the GeoDataFrame

# Rename United-States to United States
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'United-
States', 'native-country'] = 'United States of America'
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'Dominican-
Republic', 'native-country'] = 'Dominican Rep.'
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'Hong',
'native-country'] = 'Hong Kong'
less_50k_grouped.loc[less_50k_grouped['native-country'] ==
'Trinadad&Tobago', 'native-country'] = 'Trinidad and Tobago'
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'Holand-
Netherlands', 'native-country'] = 'Netherlands'
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'El-
Salvador', 'native-country'] = 'El Salvador'
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'Puerto-
Rico', 'native-country'] = 'Puerto Rico'
less_50k_grouped.loc[less_50k_grouped['native-country'] == 'Columbia',
'native-country'] = 'Colombia'


# Drop the rows with the countries that are not in not in the
GeoDataFrame --> South, Outlying-US(Guam-USVI-etc), England
less_50k_grouped = less_50k_grouped[less_50k_grouped['native-country']
!= 'South']
less_50k_grouped = less_50k_grouped[less_50k_grouped['native-country']
!= 'Outlying-US(Guam-USVI-etc)']
less_50k_grouped = less_50k_grouped[less_50k_grouped['native-country']
!= 'England']

# Merging the datasets
filtered_world = filtered_world.rename(columns={'name': 'native-
country'})  # Ensuring column names match for the merge
less_merged = filtered_world.merge(less_50k_grouped, on='native-
country', how='left')

# Example of categorizing into 3 bins with labels
bin_edges = [0, 50, 100, max(less_merged['counts'].max(), 200)]  #
Adjust the bin edges as needed
bin_labels = ['Low', 'Medium', 'High']
less_merged['category'] = pd.cut(less_merged['counts'],
bins=bin_edges, labels=bin_labels, include_lowest=True)

# Define colors for each category
category_colors = {
    'Low': 'lightgreen',
    'Medium': 'gold',
    'High': 'tomato'
}

# Plotting with a larger figure size, black borders, and a custom
legend
fig, ax = plt.subplots(1, 1, figsize=(10, 6))

legend_handles = []  # List to store legend handles

for i, (category, color) in enumerate(category_colors.items()):
    subset = less_merged[less_merged['category'] == category]

    # Create a custom label with category name and bin range
    if i < len(bin_labels) - 1:
        custom_label = f"{category}: {bin_edges[i]} -
{bin_edges[i+1]}"
    else:
        # Handle the last category differently if necessary
        custom_label = f"{category}: > {bin_edges[i]}"

    # Plot each category
    subset.plot(ax=ax, label=custom_label, color=color,
edgecolor='black')

    # Create a legend handle for the category
    legend_handles.append(Patch(facecolor=color, edgecolor='black',
label=custom_label))

# Create custom legend
plt.legend(handles=legend_handles, title="Category")
plt.title('Number of people with income <=50k in each country')
plt.axis('off')
plt.show()
```



```
# Give me a bar plot of the number of people in each country with
income greater than 50k without using the GeoDataFrame
less_50k_grouped.plot.bar(x='native-country', y='counts',
color='blue', figsize=(15, 10))
```

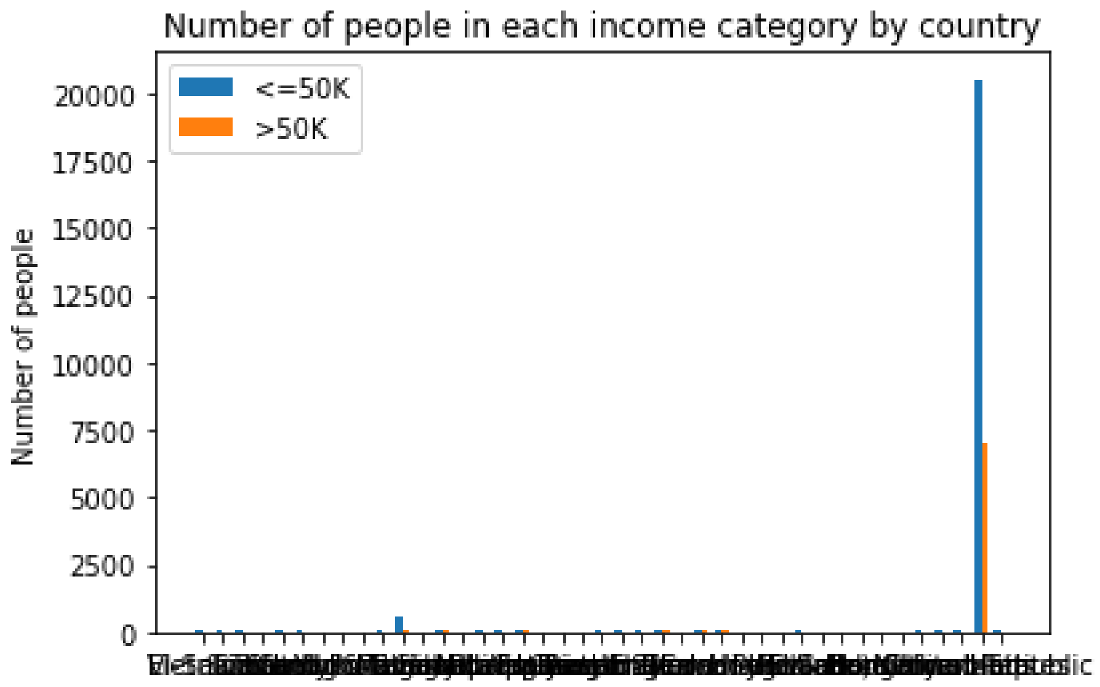
```
<AxesSubplot:xlabel='native-country'>
```



Stacked Bar Chart (Native Country VS Income)

```
# Set of countries in the dataset
countries = set(data['native-country'])

# Dictionary to store the counts for each country
country_income = {}
for country in countries:
    country_income[country] = [data_less_50k[data_less_50k['native-
country'] == country].shape[0],
data_greater_50k[data_greater_50k['native-country'] ==
country].shape[0]]

x = np.arange(len(countries))
width = 0.35

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, [country[0] for country in
country_income.values()], width, label='<=50K')
rects2 = ax.bar(x + width/2, [country[1] for country in
country_income.values()], width, label='>50K')
```

```
ax.set_ylabel('Number of people')
ax.set_title('Number of people in each income category by country')
ax.set_xticks(x)
ax.set_xticklabels(countries)
ax.legend()

plt.show()
```


Number of people in each income category by country
<=50K
>50K
Number of people