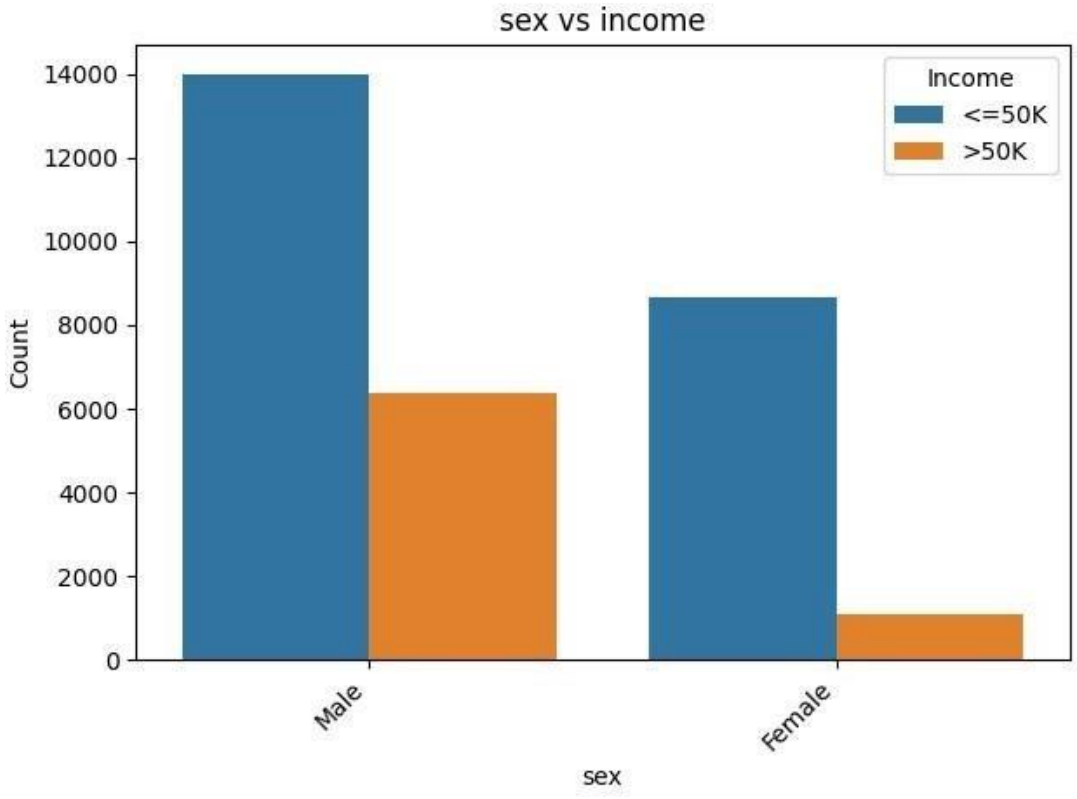
I. APPENDIX

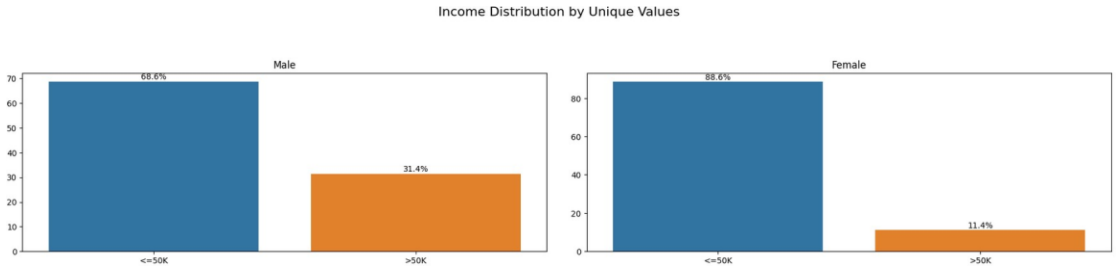
UNIVARIABLE

```
import pandas as pd
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
import math

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

# Removing rows containing "?"
df = df[~df.isin(['?']).any(axis=1)]

below = df[df["income"] == "<=50K"]
above = df[df["income"] == ">50K"]

#Counting number of rows in each category
print("(>50K) = " + str(len(above.index)))
print("(<=50K) = " + str(len(below.index)))

(>50K) = 7508
(<=50K) = 22654

# Function to visualize income distribution for numerical data
def Numerical_data(column):
    above_50k = above[column]
    below_50k = below[column]

    # Visualize income distribution for numerical data
    plt.close()
    fig, ax = plt.subplots(figsize=(10, 5))
    sns.boxplot(data=[above_50k, below_50k], palette=['salmon',
'skyblue'], showfliers=False)
    ax.set_title("Income Distribution by " + column)
    ax.set_ylabel(column)
    ax.set_xlabel("Income")
    ax.set_xticklabels(['>50K', '<=50K'])

    plt.show()
```

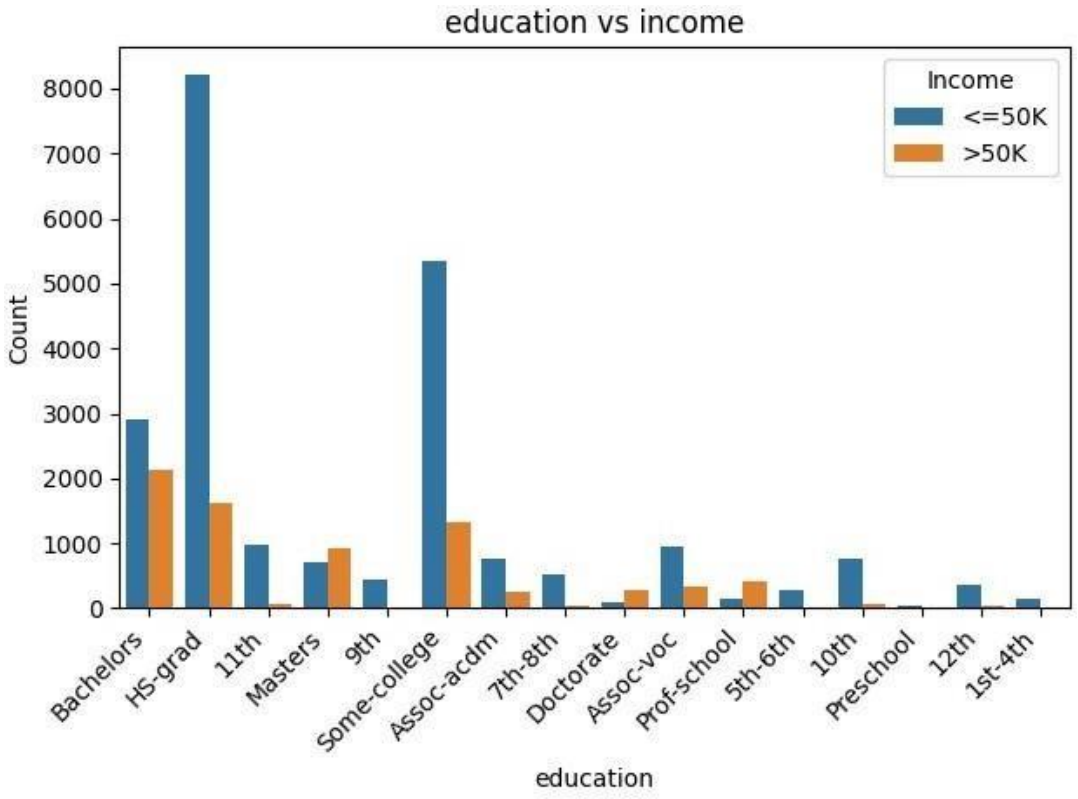
```
# Function to visualize income distribution for Categorial data
def Categorical_data(column):
    plt.close()
    sns.countplot(data=df, x=column, hue='income')
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.title(column + ' vs income')
    plt.legend(title='Income')
    plt.xticks(rotation=45, ha='right')

    # Calculate percentages for each category
    #total_count = len(df)
    #ax = plt.gca()
    #for p in ax.patches:
        #height = p.get_height()
        #percentage = (height / total_count) * 100
        #ax.text(p.get_x() + p.get_width() / 2, height,
f'{percentage:.1f}%', ha='center', va='bottom')

    plt.tight_layout()
    plt.show()



# Function to visualize income distribution of Unique values in a
categorial data
def Unique_value(column):
    unique_values = df[column].unique()
    num_plots = len(unique_values)
    num_cols = min(num_plots, 3)
    num_rows = math.ceil(num_plots / num_cols)

    plt.close()
    fig, axes = plt.subplots(ncols=num_cols, nrows=num_rows,
figsize=(10 * num_cols, 5 * num_rows))

    for i, val in enumerate(unique_values):
        valdf = df[df[column] == val]
        above_50k = valdf[valdf["income"] == ">50K"]
        below_50k = valdf[valdf["income"] == "<=50K"]

        # For Calculation of percentages
        total_count = len(valdf.index)
        below_50k_percent = (len(below_50k.index) / total_count) * 100
        above_50k_percent = (len(above_50k.index) / total_count) * 100

        # For Creating a bar plot using seaborn
        row = i // num_cols
        col = i % num_cols
```

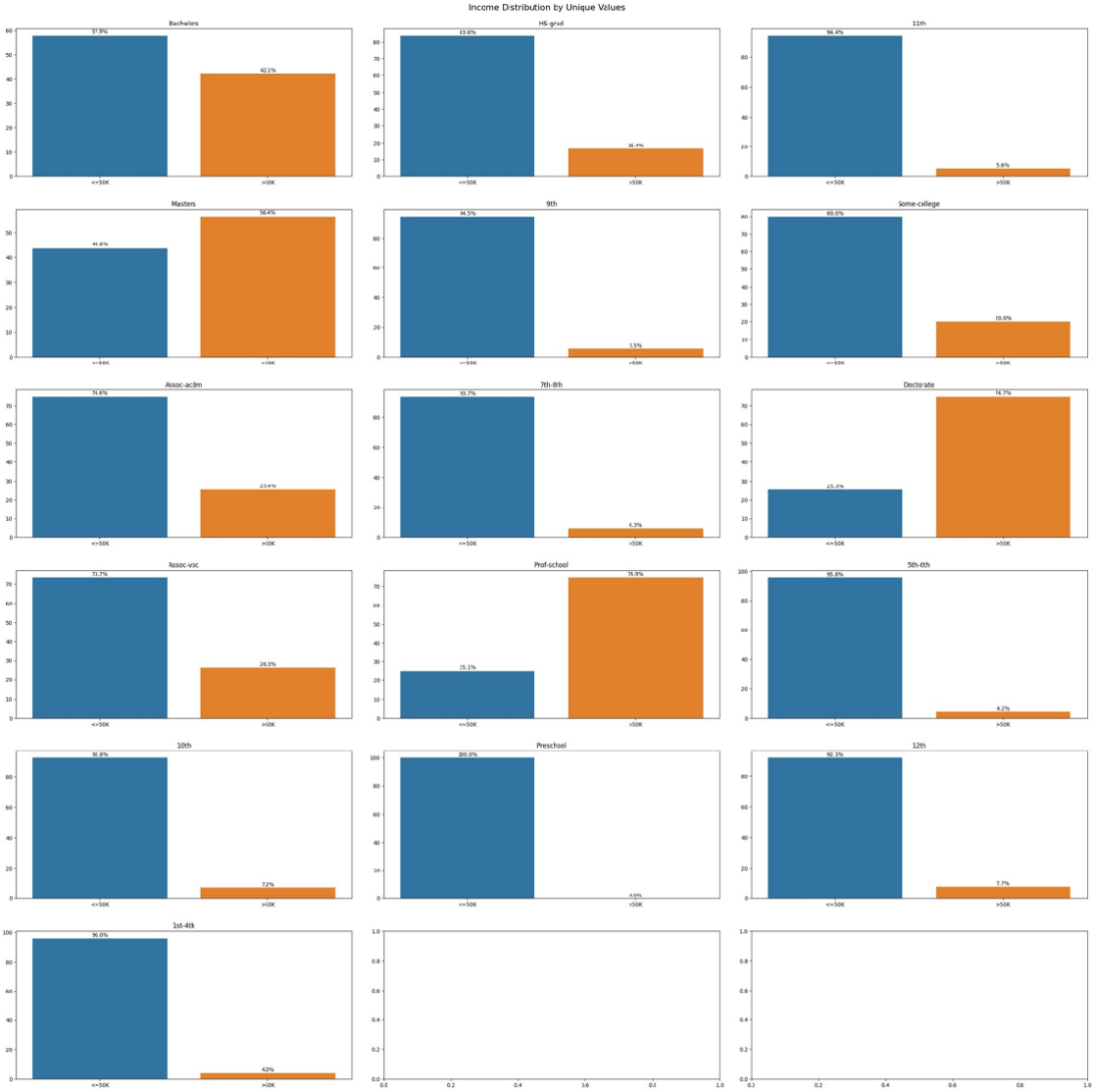
```
        ax = axes[row, col] if num_rows > 1 else axes[col]
        sns.barplot(x=["<=50K", ">50K"], y=[below_50k_percent,
above_50k_percent], ax=ax)
        ax.set_title(val)

        # For Adding labels to the bars
        for j, (label, value) in enumerate(zip(["<=50K", ">50K"],
[below_50k_percent, above_50k_percent])):
            ax.text(j, value + 1, f"{value:.1f}%", ha='center',
fontsize=10)

    # For Adding title to the figure
    fig.suptitle("Income Distribution by Unique Values", fontsize=16)

    # For Adjusting the spacing between subplots
    fig.tight_layout(pad=3.0)

    plt.show()

# Calling required functions
Categorical_data("sex")
Unique_value("sex")
```



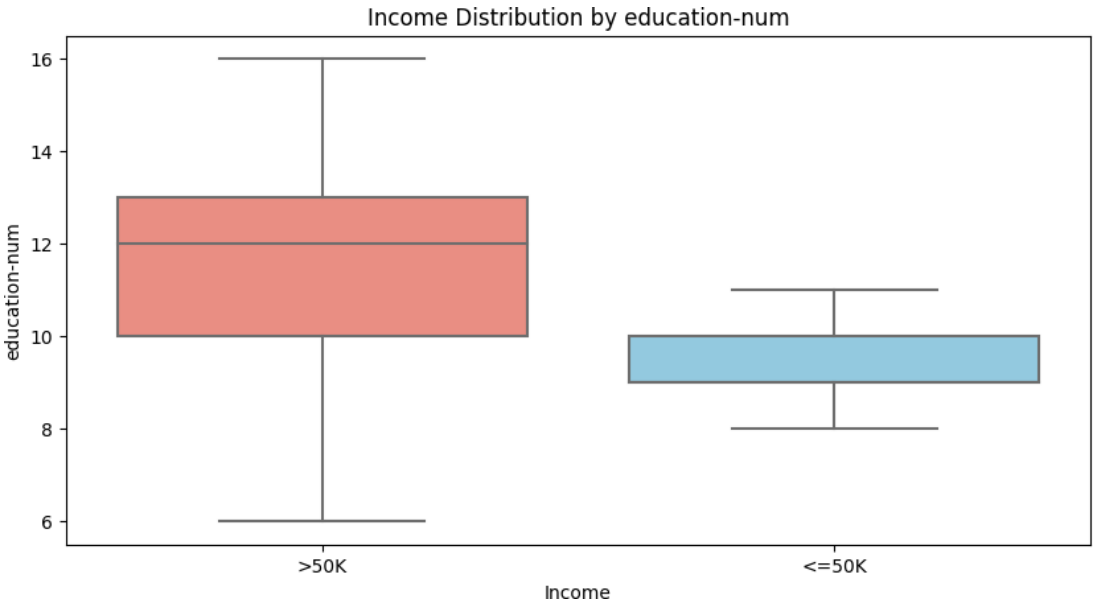



```
# Calling required functions
Categorical_data("education")
Unique_value("education")
```




Income Distribution by Unique Values
Bachelors
57.9%
42.1%
HS-grad
83.6%
16.4%
11th
94.4%
5.6%
Masters
43.6%
56.4%
9th
94.5%
5.5%
Some-college
80.0%
20.0%
Assoc-acdm
74.6%
25.4%
7th-8th
93.7%
6.3%
Doctorate
25.3%
74.7%
Assoc-voc
73.7%
26.3%
Prof-school
75.1%
24.9%
5th-6th
95.8%
4.2%
10th
92.8%
7.2%
Preschool
100.0%
0.0%
12th
92.3%
7.7%
1st-4th
96.0%
4.0%
<=50K
>50K

```
# I printed this plot just to show an example for Numerical_data()
# Calling required functions
Numerical_data("education-num")
```

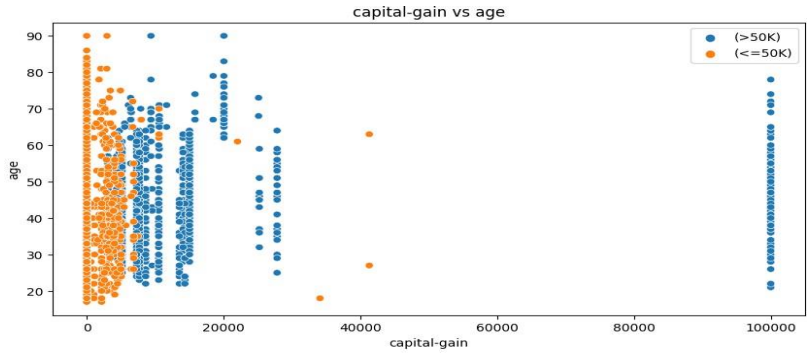


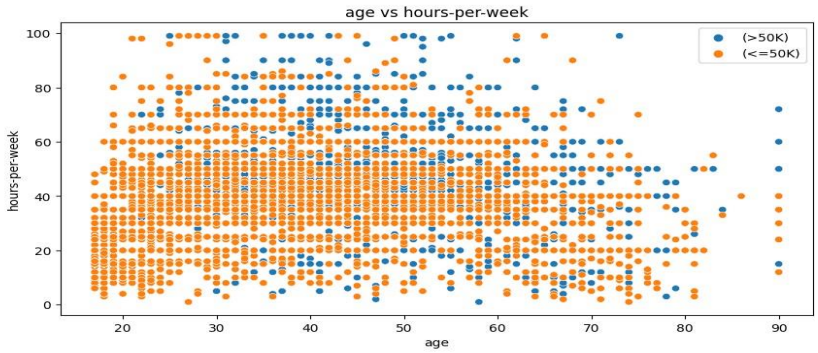
MULTIVARIABLE

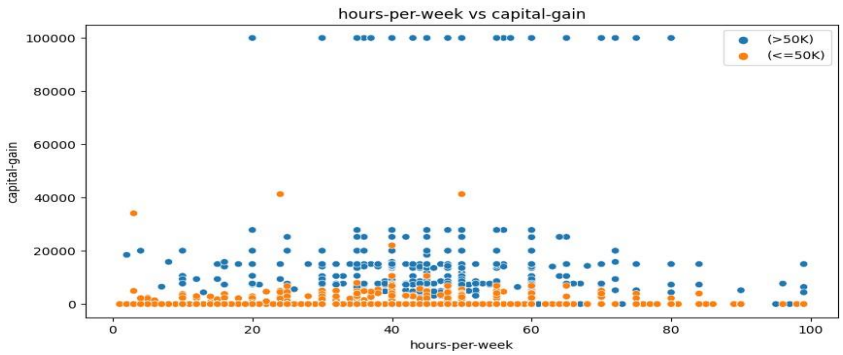
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.graphics.mosaicplot import mosaic
from collections import Counter
from sklearn.preprocessing import MinMaxScaler


df = pd.read_csv("adult.data", header=None, sep=", ", engine='python')
df.columns = ["age", "workclass", "fnlwgt", "education", "education-num",
"marital-status", "occupation", "relationship",
              "race", "sex", "capital-gain", "capital-loss", "hours-per-
week", "native-country", "salary"]

# Removing rows containing "?"
df = df[~df.isin(['?']).any(axis=1)]
# Separate the data based on salary range
below_50K = df[df["salary"] == "<=50K"].sample(n=7841)
above_50K = df[df["salary"] == ">50K"]

# Concatenate the balanced dataframes
df = pd.concat([above_50K, below_50K])

# Assign the class labels
df['income'] = (df["salary"] == ">50K").astype(int)

def plot_Numerical_scatter(column1, column2, column3):
    plt.close()
    fig, axes = plt.subplots(ncols=1, nrows=3, figsize=(10, 20))
    fig.subplots_adjust(hspace=.5)

    colors = ['(>50K)' if income else '(<=50K)' for income in df['income']]

    # Scatter plot for column1 vs column2
    sns.scatterplot(data=df, x=column1, y=column2, hue=colors, legend=True,
ax=axes[0])
    axes[0].set_title(column1 + " vs " + column2)
    axes[0].set_xlabel(column1)
    axes[0].set_ylabel(column2)

    # Scatter plot for column2 vs column3
    sns.scatterplot(data=df, x=column2, y=column3, hue=colors, legend=True,
ax=axes[1])
    axes[1].set_title(column2 + " vs " + column3)
    axes[1].set_xlabel(column2)
    axes[1].set_ylabel(column3)

    # Scatter plot for column3 vs column1
```

```
    sns.scatterplot(data=df, x=column3, y=column1, hue=colors, legend=True,
ax=axes[2])
    axes[2].set_title(column3 + " vs " + column1)
    axes[2].set_xlabel(column3)
    axes[2].set_ylabel(column1)

    plt.show()


def plot_Categorical_heatmap(column1, column2):
    plt.close()
    fig, axes = plt.subplots(ncols=2, figsize=(12, 10))
    fig.subplots_adjust(wspace=0.4)

    # Create contingency tables for below_50K and above_50K
    below_50K_table = pd.crosstab(below_50K[column1], below_50K[column2])
    above_50K_table = pd.crosstab(above_50K[column1], above_50K[column2])

    # Generate the heatmap for below_50K
    sns.heatmap(below_50K_table, cmap='coolwarm', annot=True, fmt='d',
cbar=True, ax=axes[0])
    axes[0].set_title("Below 50K")
    axes[0].set_xlabel(column2)
    axes[0].set_ylabel(column1)

    # Generate the heatmap for above_50K
    sns.heatmap(above_50K_table, cmap='coolwarm', annot=True, fmt='d',
cbar=True, ax=axes[1])
    axes[1].set_title("Above 50K")
    axes[1].set_xlabel(column2)
    axes[1].set_ylabel(column1)

    plt.show()



def Parallel_coordinates_Numerical(column1, column2, column3):
    frame_pc = df[[column1, column2, column3, 'income']].copy()
    frame_np_array = MinMaxScaler().fit_transform(frame_pc.values)
    frame_pc = pd.DataFrame(frame_np_array)
    df.index = frame_pc.index
    frame_pc['salary'] = df['salary']
    frame_pc.columns = [column1, column2, column3, 'income', 'salary']
    frame_pc_below_50K = frame_pc[frame_pc["income"] == 0.0].sample(n=30)
    frame_pc_above_50K = frame_pc[frame_pc["income"] == 1.0].sample(n=30)

    # Concatenating the balanced dataframes
    frame_pc = pd.concat([frame_pc_below_50K, frame_pc_above_50K])
```

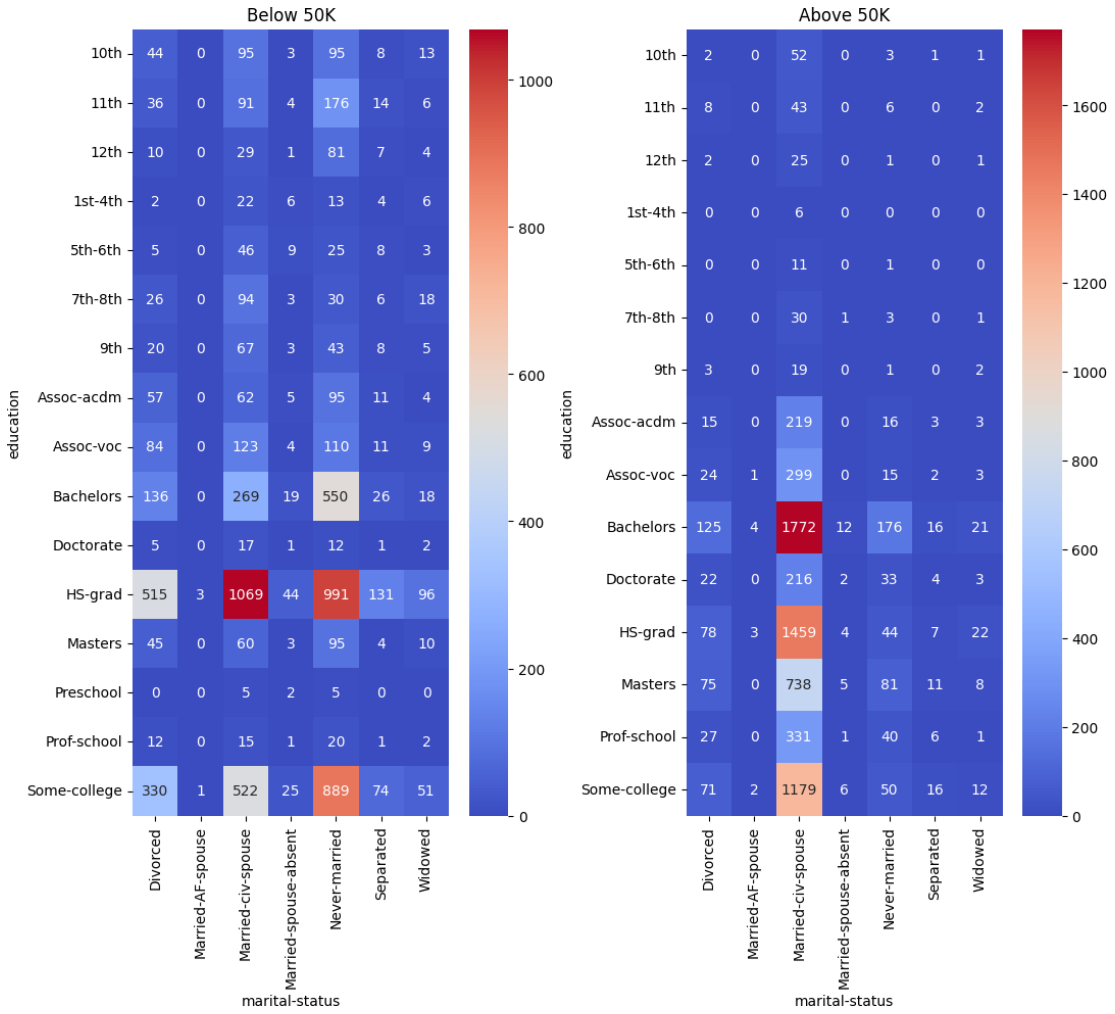
```
    # Visualize income distribution
    plt.figure(figsize=(10, 6))
    pd.plotting.parallel_coordinates(frame_pc, 'salary', cols=[column1,
column2, column3], color=('yellow', 'purple'))
    plt.xlabel("Features")
    plt.ylabel("Normalized Values")
    plt.title("Parallel Coordinates Plot")
    plt.legend(loc='upper right')
    plt.show()

# Calling required functions
plot_Numerical_scatter('capital-gain', 'age', 'hours-per-week')
```







```
# Calling required functions
plot_Categorical_heatmap('education', 'marital-status')
```



```
# Calling required functions
Parallel_coordinates_Numerical('age', 'capital-gain', 'education-num')
```