# Python_assign

February 13, 2025

[4]:
```python
import pandas as pd
```

[28]:
```python
import numpy as np
import scipy as stats
import matplotlib.pyplot as plt
import seaborn as sns
```

[6]:
```python
df=pd.read_csv("Simulated_DatasetPython.csv")
```

[7]:
```python
print(df.head())
```

```
   Age  Gender      Height     Weight       Income Region
0   56  Female  166.907115  63.213067  63274.85701  North
1   46  Female  176.731255  81.655747  43032.05377  North
2   32  Female  167.433698  85.683568  40666.44253  North
3   60    Male  166.321743  64.867874  44667.67039  North
4   25    Male  182.737336  56.109301  64280.16641  North
```

[9]:
```python
#Descriptive_Statistics
mean_Age= df['Age'].mean()
median_Age= df['Age'].median()
std_Age= df['Age'].std()
```

[11]:
```python
mean_Height= df['Height'].mean()
median_Height= df['Height'].median()
std_Height= df['Height'].std()
```

[12]:
```python
mean_Weight= df['Weight'].mean()
median_Weight= df['Weight'].median()
std_Weight= df['Weight'].std()
```

[13]:
```python
mean_Income= df['Income'].mean()
median_Income= df['Income'].median()
std_Income= df['Income'].std()
```

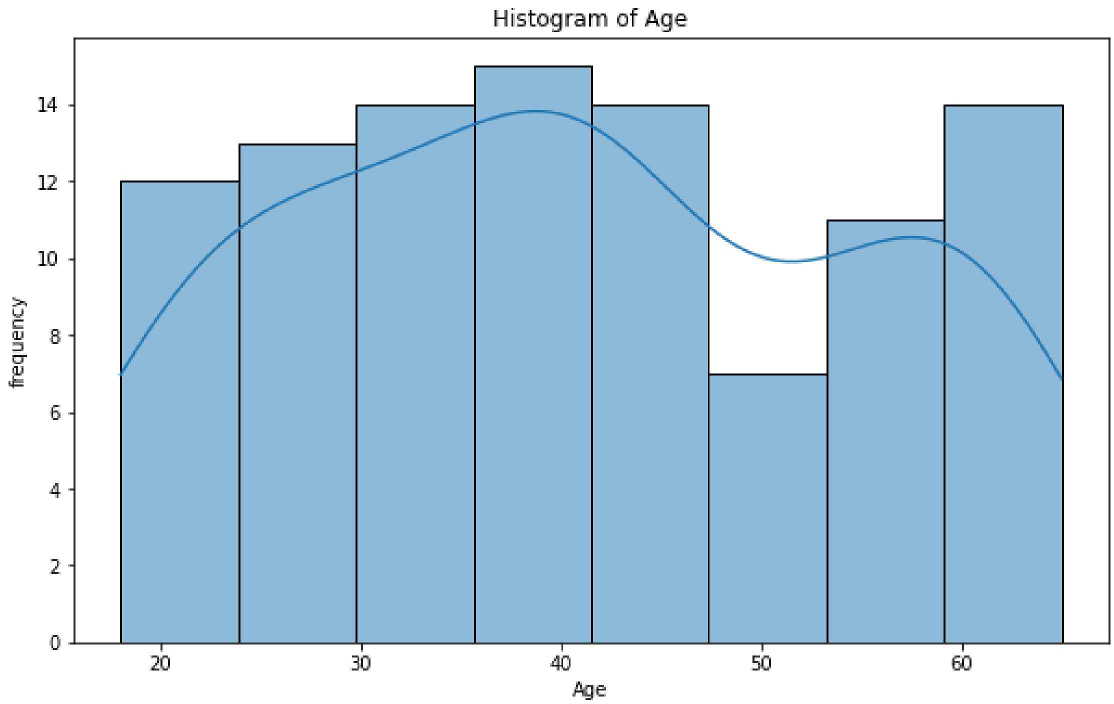
[14]:
```python
#Calculate_Summary_statistics
print(f"Mean Age: {mean_Age}, Median Age: {median_Age}, Std Age: {std_Age}")
```

```
print(f"Mean Height: {mean_Height}, Median Height: {median_Height}, Std Height:␣
↪{std_Height}")
print(f"Mean Weight: {mean_Weight}, Median Weight: {median_Weight}, Std Weight:␣
↪{std_Weight}")
print(f"Mean Income: {mean_Income}, Median Income: {median_Income}, Std Income:␣
↪{std_Income}")
print("\nFull Summary Statistics:")
print(df.describe())
```

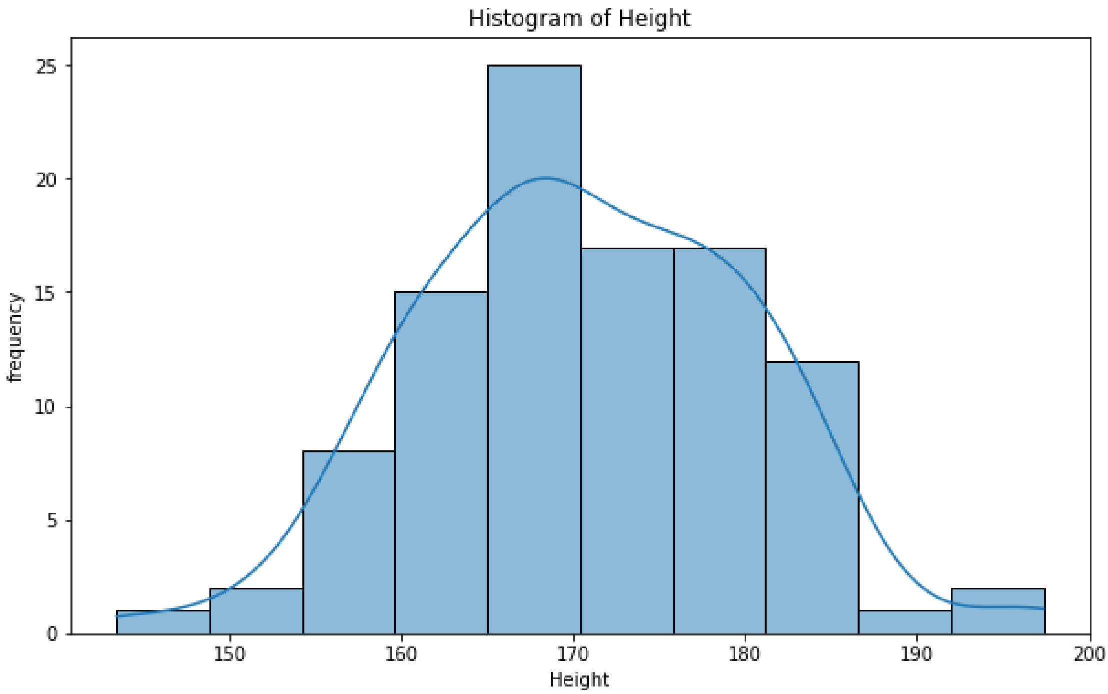
```
Mean Age: 40.92, Median Age: 41.0, Std Age: 14.054497105513555
Mean Height: 170.68712894, Median Height: 170.06540825000002, Std Height:
9.549196364724073
Mean Weight: 72.5263900181, Median Weight: 71.63930274500001, Std Weight:
15.132806469501524
Mean Income: 52492.42755600001, Median Income: 54029.436605, Std Income:
14505.766436328271

Full Summary Statistics:
              Age      Height      Weight        Income
count  100.000000  100.000000  100.000000    100.000000
mean    40.920000  170.687129   72.526390  52492.427556
std     14.054497    9.549196   15.132806  14505.766436
min     18.000000  143.448239   40.075576  18649.733770
25%     30.500000  164.041725   62.604011  43380.023233
50%     41.000000  170.065408   71.639303  54029.436605
75%     53.250000  177.679670   83.594140  63182.204065
max     65.000000  197.344222  105.835539  84136.419460
```

[16]:

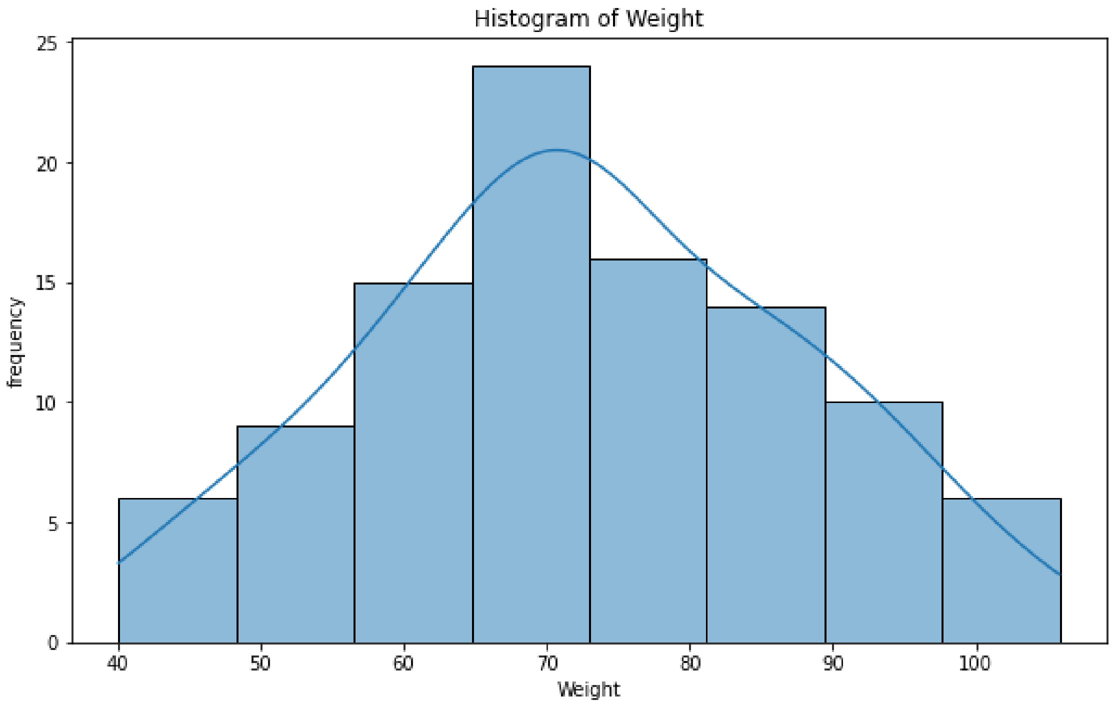
```
#Create_Histogram_for_Age
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], kde=True)
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('frequency')
plt.show()
```




[17]:
```
#Create_Histogram_for_Height
plt.figure(figsize=(10, 6))
sns.histplot(df['Height'], kde=True)
plt.title('Histogram of Height')
plt.xlabel('Height')
plt.ylabel('frequency')
plt.show()
```

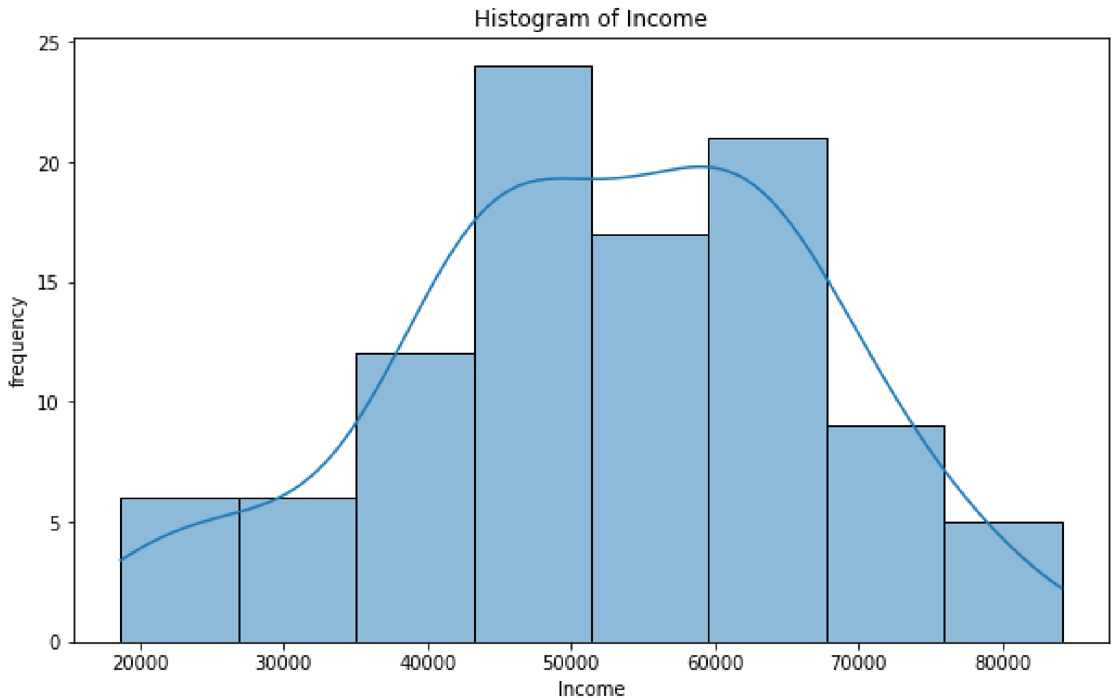



[18]:
```
#Create_Histogram_for_Weight
plt.figure(figsize=(10, 6))
sns.histplot(df['Weight'], kde=True)
plt.title('Histogram of Weight')
plt.xlabel('Weight')
plt.ylabel('frequency')
plt.show()
```

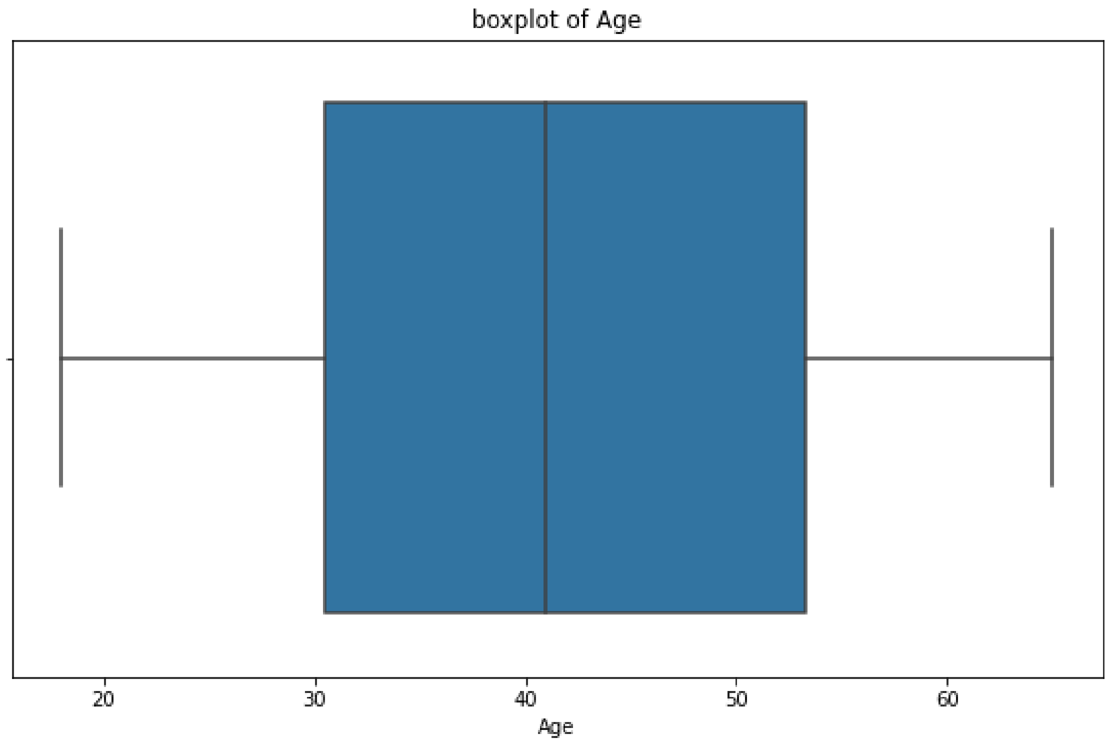



[19]:

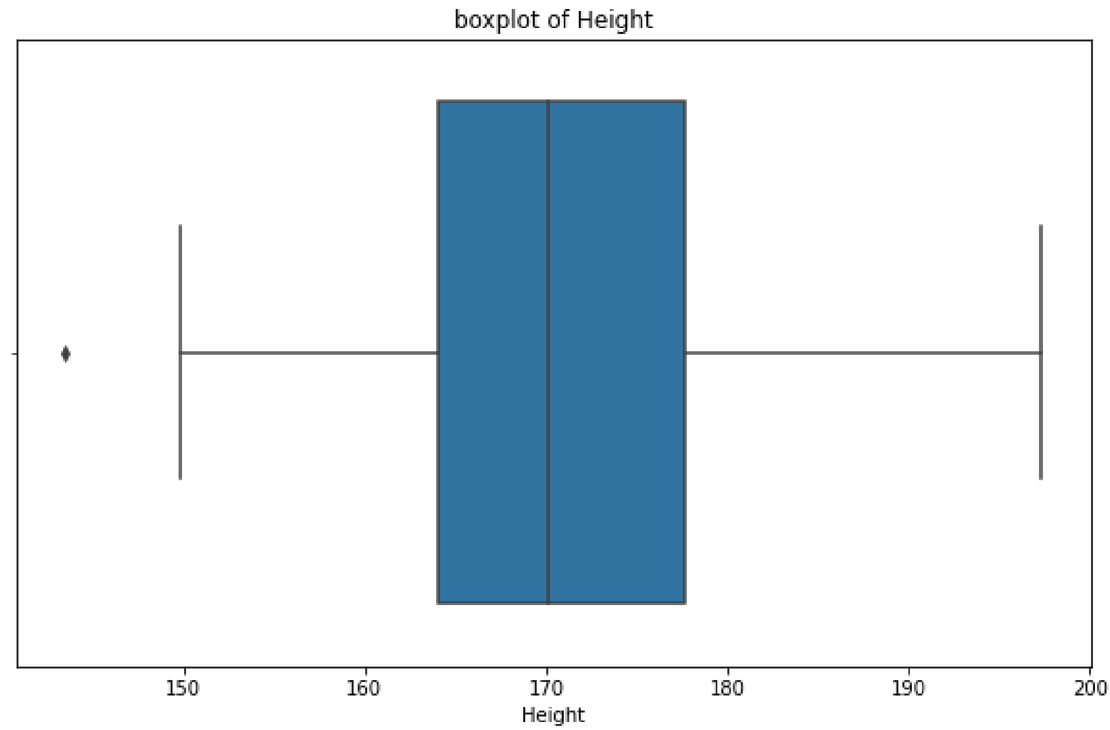
```
#Create_Histogram_for_Income
plt.figure(figsize=(10, 6))
sns.histplot(df['Income'], kde=True)
plt.title('Histogram of Income')
plt.xlabel('Income')
plt.ylabel('frequency')
plt.show()
```



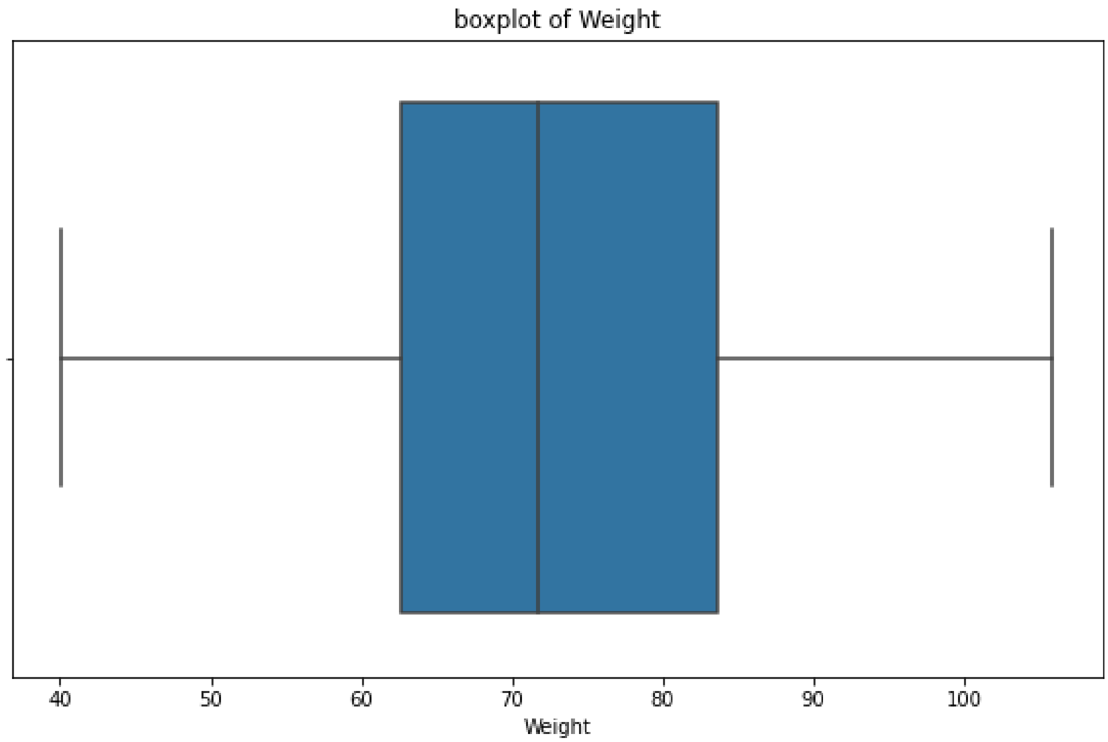

[21]:
```
#Create_Boxplots_Age
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Age'])
plt.title('boxplot of Age')
plt.xlabel('Age')
plt.show()
```



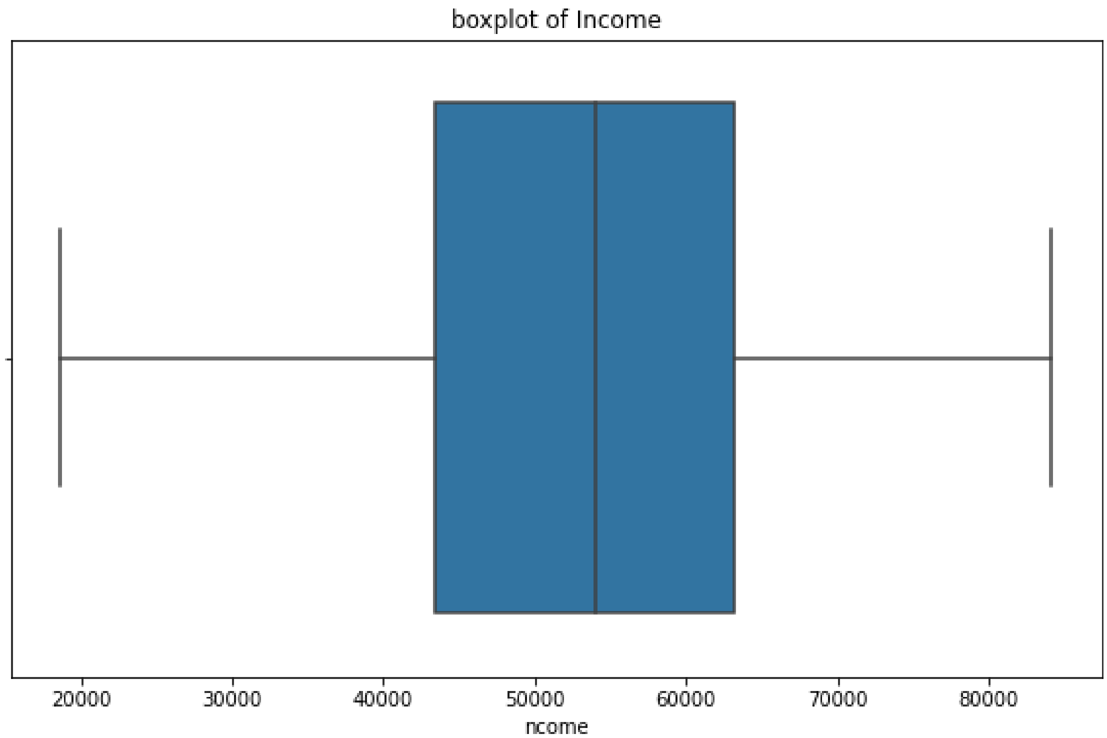

[22]:
```
#Create_Boxplots_Height
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Height'])
plt.title('boxplot of Height')
plt.xlabel('Height')
plt.show()
```




[23]:
```
#Create_Boxplots_Weight
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Weight'])
plt.title('boxplot of Weight')
plt.xlabel('Weight')
plt.show()
```




[25]:
```
#Create_Boxplots_Income
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Income'])
plt.title('boxplot of Income')
plt.xlabel('ncome')
plt.show()
```




[31]:
```
# T-Test
# Separate data by Gender

male_income = df[df['Gender'] == 'Male']['Income']
female_income = df[df['Gender'] == 'Female']['Income']

# Perform t-test
t_stat, p_value = stats.ttest_ind(male_income, female_income, equal_var=False)

# Print result
print(f"T-test result: t-statistic = {t_stat}, p-value = {p_value}")

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis: The mean income differs between males␣
 ↪and females.")
else:
    print("Fail to reject the null hypothesis: The mean income is the same␣
 ↪between males and females.")
```

```
T-test result: t-statistic = 0.9596258931638687, p-value = 0.339612081929773
```

Fail to reject the null hypothesis: The mean income is the same between males and females.

[57]:
```
# Z_Test



# Given population mean
population_mean = 50

# Choose a numerical column (e.g., 'Height' or 'Weight')
column_name = 'Height'  # Change this to 'Weight' or another numerical column␣
↪if needed

# Sample mean, standard deviation, and size
sample_mean = df[column_name].mean()
sample_std = df[column_name].std()
sample_size = len(df)

# Calculate the Z-statistic
z_stat = (sample_mean - population_mean) / (sample_std / np.sqrt(sample_size))

# Calculate the p-value for a two-tailed test
p_value_z = stats.norm.sf(abs(z_stat)) * 2  # Two-tailed test

# Print the result of the Z-test
print(f"Z-test result for {column_name}: z-statistic = {z_stat}, p-value =␣
↪{p_value_z}")

# Conclusion
if p_value_z < 0.05:
    print(f"Reject the null hypothesis: The mean {column_name} is significantly␣
↪different from 50.")
else:
    print(f"Fail to reject the null hypothesis: The mean {column_name} is not␣
↪significantly different from 50.")
```

Z-test result for Height: z-statistic = 126.3845923054147, p-value = 0.0
Reject the null hypothesis: The mean Height is significantly different from 50.

[55]:
```
#ANOVA



# Define the groups based on 'Region'
regions = df['Region'].unique()
grouped_data = [df[df['Region'] == region]['Income'] for region in regions]

# Perform one-way ANOVA
```

```
f_stat, p_value_anova = stats.f_oneway(*grouped_data)

# Print the result of the ANOVA
print(f"ANOVA result: F-statistic = {f_stat}, p-value = {p_value_anova}")

# Check the p-value to determine whether to reject or fail to reject the null␣
↪hypothesis
if p_value_anova < 0.05:
    print("Reject the null hypothesis: There is a significant difference in the␣
↪mean income across regions.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
↪difference in the mean income across regions.")
```

```
ANOVA result: F-statistic = 0.5700359127317277, p-value = 0.6360822046090806
Fail to reject the null hypothesis: There is no significant difference in the
mean income across regions.
```

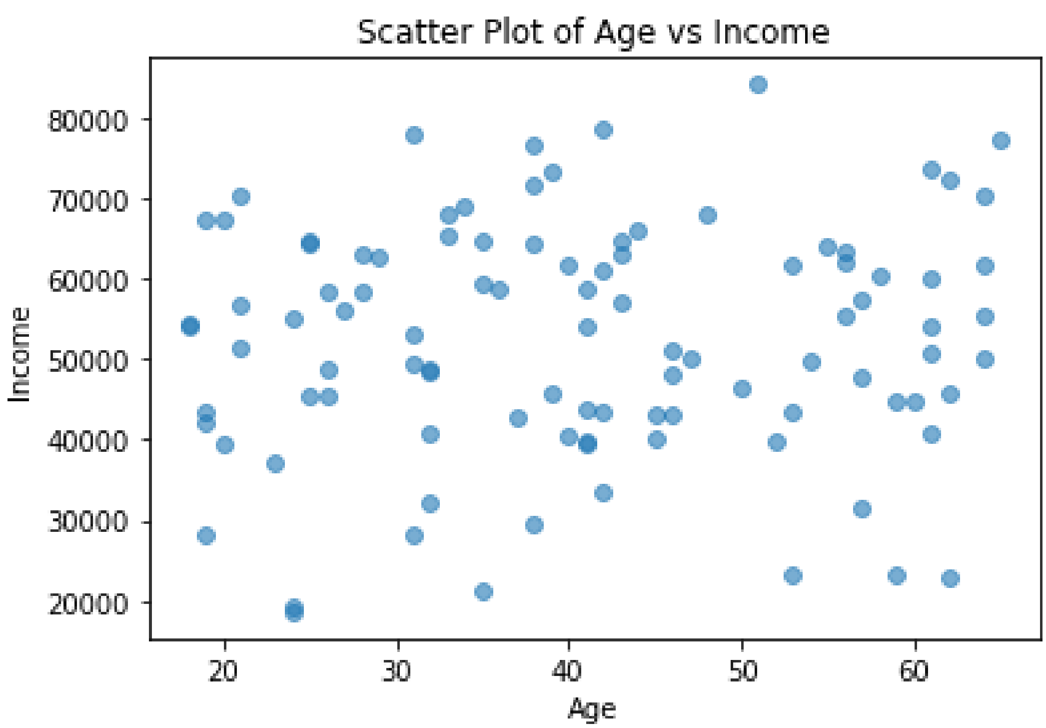
[54]:
```
# Confidence_Intervals



# Choose the numerical column (e.g., 'Height')
column_name = 'Height'  # You can change this to 'Weight' or 'Income' as needed

# Sample mean, standard deviation, and size
sample_mean = df[column_name].mean()
sample_std = df[column_name].std()
sample_size = len(df)

# Calculate the standard error (SE)
standard_error = sample_std / np.sqrt(sample_size)

# Z-score for 95% confidence interval (approximately 1.96)
z_score = 1.96

# Calculate the margin of error
margin_of_error = z_score * standard_error

# Calculate the confidence interval
confidence_interval = (sample_mean - margin_of_error, sample_mean +␣
↪margin_of_error)

# Print the confidence interval
print(f"95% Confidence Interval for the mean of {column_name}:␣
↪({confidence_interval[0]}, {confidence_interval[1]})")
```

```
95% Confidence Interval for the mean of Height: (168.81548645251408,
```

172.55877142748594)

[53]:
```
#Correlation

# Calculate Pearson correlation between 'Age' and 'Income'
correlation, _ = stats.pearsonr(df['Age'], df['Income'])

# Print the Pearson correlation coefficient
print(f"Pearson correlation between Age and Income: {correlation}")

# Visualize the relationship using a scatter plot
plt.scatter(df['Age'], df['Income'], alpha=0.6)
plt.title('Scatter Plot of Age vs Income')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()
```

Pearson correlation between Age and Income: 0.06599749922358968




[63]:
```
# Interpretation_Pearson_correlation
#The Pearson correlation of 0.066 indicates a very weak positive relationship␣
↪between `Age` and `Income`. This suggests that `Age` has little to no effect␣
↪on `Income` in this dataset.
```

[60]:

```
# regression_Analysis

from sklearn.linear_model import LinearRegression

# Reshape the data for the regression model
X = df[['Age']].values  # Predictor variable (Age)
y = df['Income'].values  # Response variable (Income)

# Create a linear regression model
model = LinearRegression()
model.fit(X, y)

# Get the regression coefficients
slope = model.coef_[0]
intercept = model.intercept_

# Calculate the R-squared value
r_squared = model.score(X, y)

# Print the results
print(f"Linear Regression: Slope = {slope}, Intercept = {intercept}")
print(f"R-squared value: {r_squared}")
```

```
Linear Regression: Slope = 68.1165823246414, Intercept = 49705.09700727568
R-squared value: 0.004355669903767789
```

[62]:

```
# Interpretation_Regression_Analysis
# The model shows a slight positive relationship between Age and Income, but␣
↪the low R-squared value (0.0044) indicates that Age explains very little of␣
↪the variation in Income. Other factors likely have a stronger impact.
```

[ ]:

```

```