Project Name: Web Scraping and Data Visualization Using Python

Objective

The objective of this project is to scrape race results data from a website, clean and manipulate the data, and perform data analysis and visualization to uncover insights. Specifically, the goals are:

1. **Web Scraping**: Extract race results data from an online source.
2. **Data Cleaning**: Clean and preprocess the scraped data for analysis.
3. **Data Analysis**: Analyse race times and demographics to understand performance trends.
4. **Data Visualization**: Create visualizations to represent the data and findings.

Summary

The project involves several steps, from scraping web data to visualizing it. Here's a breakdown of the process:

1. **Web Scraping**:
    - **URL Access**: Opened the webpage containing the race results using urlopen.
    - **HTML Parsing**: Used BeautifulSoup to parse the HTML content and extract relevant data.
2. **Data Extraction and Cleaning**:
    - **Extracting Table Data**: Retrieved data from HTML tables and converted it to a readable format.
    - **Data Cleaning**: Cleaned and formatted the extracted data into a Pandas DataFrame. Adjusted column names and removed unnecessary characters.
3. **Data Manipulation**:
    - **Time Conversion**: Converted race times into minutes for better analysis.
    - **Descriptive Statistics**: Generated summary statistics of the cleaned data.
4. **Data Visualization**:
    - **Box Plot**: Visualized the distribution of runner times using a box plot.
    - **Histogram**: Plotted histograms to show the distribution of race times.
    - **Gender-Based Analysis**: Created separate histograms for male and female runners and compared their performance.
    - **Box Plot by Gender**: Plotted box plots to compare race times across genders.

Results

1. **Data Extraction and Cleaning**:
    - Successfully extracted race results data and converted it into a usable DataFrame.
    - Cleaned the time data and adjusted column names for consistency.
2. **Descriptive Statistics**:
    - Provided a summary of the data, including mean, median, and standard deviation of race times.
3. **Visualizations**:
    - **Box Plot**: Showed the range and distribution of runner times, with visible outliers.
    - **Histogram**: Displayed the distribution of race times with a normal distribution curve, revealing the overall trends in runner performance.
    - **Gender-Based Analysis**: Histograms for male and female runners showed differences in performance, highlighting potential trends or disparities.

Conclusion

The Web Scraping and Data Visualization project provides valuable insights into race results:

- **Race Time Distribution**: Visualizations such as box plots and histograms reveal the overall distribution of race times, including the presence of outliers.
- **Gender Performance**: Gender-based histograms show differences in performance between male and female runners, which can be useful for understanding gender-specific trends.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from urllib.request import urlopen
from bs4 import BeautifulSoup

#Open the Home Page
url = "https://www.hubertiming.com/results/2017GPTR10K"
#url = "https://tatamumbaimarathon.procam.in/results/race-results"
html = urlopen(url)

soup = BeautifulSoup(html, 'lxml')
type(soup)

#Get The Titile
title = soup.title
print(title)

#Print out the text
text = soup.get_text()
#print(soup.text)

soup.find_all("a")

all_links = soup.find_all("a")
for link in all_links :
    print(link.get("href"))
```

```
#print the first 10 rows for sanity check
print("#Printing the first 10 rows for sanity check")
rows = soup.find_all('tr')
print(rows[:10])

for row in rows:
   row_td = row.find_all('td')
print(row_td)
type(row_td)



str_cells = str(row_td)
cleantext = BeautifulSoup(str_cells, "lxml").get_text()
print(cleantext)

import re

list_rows = []
for row in rows :
   cells = row.find_all('td')
   str_cells = str(cells)
   clean = re.compile('<.*?>')
   clean2 = (re.sub(clean,'',str_cells))
   list_rows.append(clean2)
print(clean2)
type(clean2)

df = pd.DataFrame(list_rows)
df.head(10)
'''
print("**********")
```

```
print(df.columns)
print("***********")
'''

#Data Manipulation and Cleaning

df1 = df[0].str.split(',', expand=True)
df1.head(10)



df1 = df[0].str.split(',', expand=True)
df1.head(10)



col_labels = soup.find_all('th')

all_header = []
col_str = str(col_labels)
cleantext2 = BeautifulSoup(col_str,"lxml").get_text()
all_header.append(cleantext2)
print(all_header)



df2 = pd.DataFrame(all_header)
print("*******df2.head*******")
print(df2.head())

df3 = df2[0].str.split(',', expand=True)
print("*******df3.head*******")
print(df3.head())
```

```
frames = [df3, df1]
df4 = pd.concat(frames)
print("*******df4.head*******")
df4.head(10)
#print(df4)


df5 = df4.rename(columns=df4.iloc[0])
print("*******df5.head*******")
print(df5.head())

print(df5.info())
print(df5.shape)

df6 = df5.dropna(axis=0, how='any')

df7 = df6.drop(df6.index[0])
print("*******df7.head*******")
print(df7.head())

df7.rename(columns={'[Place': 'Place'}, inplace=True)
df7.rename(columns={' Team]': 'Team'}, inplace=True)
print("*******df7.head*******")
print(df7.head())

df7['Team'] = df7['Team'].str.strip(']')
print("*******df7.head*******")
print(df7.head())

print("***********")
```

```
print(df7.columns)
print("***********")


#Data Analysis and Visualization

time_list = df7[' Time'].tolist()

#You can use a for loop to convert 'Chip Time' to minutes

time_mins = []
for i in time_list:
    if i.count(":")==1: #Check for : count
        m, s = i.split(':')
        math = ((int(m) * 60) + int(s))/60
    elif i.count(":")==2: #second also expected
        h ,m, s = i.split(':')
        math = (int(h) * 3600 +int(m) * 60 + int(s))/60
    else:
        print("Error occurred reading the data")
        math = 0
    time_mins.append(math)
#print(time_mins)

df7['Runner_mins'] = time_mins
print(df7.head())

print(df7.describe(include=[np.number]))

#BoxPlot
from pylab import rcParams
```

```
rcParams['figure.figsize']= 15,5

df7.boxplot(column='Runner_mins')

plt.grid(True, axis='y')

plt.ylabel('Chip Time')

plt.xticks([1],['Runners'])

plt.show()



#Normal distribution graph

x = df7['Runner_mins']

#ax = sns.displot(x, element='bars', kde=True, rug=False, color='m', bins=25, hist_kws={'edgecolor':
'black'}) giving error so updated

ax = sns.histplot(x, kde=True, color='m', bins=25, edgecolor='black')

#ax = sns.displot(x, kind='hist', kde=True, color='m', bins=25)

plt.show()


'''

#error

f_fuko = df7.loc[df7[' Gender']== ' F'] ['Runner_mins']

m_fuko = df7.loc[df7[' Gender']== ' M'] ['Runner_mins']


sns.displot(f_fuko, hist=True, kde=True, rug=False, hist_kws={'edgecolor' : 'black'}, label='Female')

sns.displot(f_fuko, hist=False, kde=True, rug=False, hist_kws={'edgecolor' : 'black'}, label='Male')


plt.legend()

plt.show()



#sns.histplot(m_fuko, kde=True, edgecolor='black', label='Male')

#sns.displot(f_fuko, hist=True, kde=True, rug=False, edgecolor='black', label='Female')

#sns.histplot(m_fuko, kde=True, edgecolor='black', label='Male')
```

```
#sns.displot(f_fuko, hist=False, kde=True, rug=False, edgecolor='black', label='Male')
#sns.histplot(f_fuko, kde=True, edgecolor='black', label='Female')
#sns.histplot(m_fuko, kde=True, edgecolor='black', label='Male')
'''

f_fuko = df7.loc[df7[' Gender']== ' F'] ['Runner_mins']
m_fuko = df7.loc[df7[' Gender']== ' M'] ['Runner_mins']

# Plotting
sns.histplot(f_fuko, kde=True, edgecolor='black', label='Female')
sns.histplot(m_fuko, kde=True, edgecolor='black', label='Male') #one error will solve it later

# Adding legend and showing plot
plt.legend()
plt.show()



g_stats = df7.groupby(" Gender", as_index=True).describe()
print(g_stats)



df7.boxplot(column='Runner_mins', by=' Gender')
plt.ylabel('Chip Time')
plt.suptitle("")
plt.show()
```