**QUESTION 1- Which colleges are producing the most NBA players for players in 2022-2023?**

# Import the requests library for making HTTP requests

import requests

# Import the BeautifulSoup library for web scraping and parsing HTML

from bs4 import BeautifulSoup

# Import the pandas library for data manipulation and analysis

import pandas as pd

# Import the pyplot module of the matplotlib library for data visualization

import matplotlib.pyplot as plt

# Import the NumPy library for numerical computing with Python

import numpy as np

# Import the random module for generating random numbers

import random

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Make a GET request to the desired URL and retrieve the HTML content

webpage3 = requests.get("https://www.basketball-reference.com/teams/MIA/2023.html")

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(webpage3.content, 'html.parser')

# Find all the HTML td tags that have a "data-stat" attribute with a value of "college"

players = soup.find\_all("td", {"data-stat" : "college"})

# Create an empty list to store the college names of the players

players\_colleges\_more = []

# Loop through each player's HTML tag and find any "a" tags (which contain the college name)

for i in range(0, len(players)):

for item in players[i].find\_all("a"):

# Append the college name to the list of player colleges

players\_colleges\_more.append(item.text)

# Print the list of player colleges

print(players\_colleges\_more)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Create an empty string to store the new URL

new\_web\_links = ""

# Create an empty list to store the college names of the players

players\_colleges\_more = []

# Construct the new URL for the desired team and season

new\_web\_links = "https://www.basketball-reference.com/teams/" + 'ATL' + "/2023.html"

# Make a GET request to the desired URL and retrieve the HTML content

webpage4 = requests.get(new\_web\_links)

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(webpage4.content, 'html.parser')

# Find all the HTML td tags that have a "data-stat" attribute with a value of "college"

players = soup.find\_all("td", {"data-stat" : "college"})

# Loop through each player's HTML tag and find any "a" tags (which contain the college name)

for i in range(0, len(players)):

for item in players[i].find\_all("a"):

# Append the college name to the list of player colleges

players\_colleges\_more.append(item.text)

# Print the list of player colleges

print(players\_colleges\_more)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Make a GET request to the desired URL and retrieve the HTML content

webpage = requests.get("https://geojango.com/pages/list-of-nba-teams")

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(webpage.content, 'html.parser')

# Find all the HTML td tags that have a "class" attribute with a value of "shogun-table-row"

table\_data = soup.find\_all("td", {"class" : "shogun-table-row"})

# Create an empty list to store the NBA team names

nba\_teams = []

# Loop through each row of the table data and extract the team names

for i in range(0, len(table\_data)):

# Check if the current row is a team name row (every 5th row starting from the first)

if i % 5 == 0:

# Find the span tag containing the team name and append it to the list of team names

for item in table\_data[i].find\_all("span"):

nba\_teams.append(item.text)

# Print the list of NBA team names

print(nba\_teams)

#print(table\_data)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Create an empty list to store the NBA team names with only their first 3 letters in uppercase

nba\_teams\_three\_letters = []

# Loop through each NBA team name and add the first 3 letters (in uppercase) to the list

for item in nba\_teams:

nba\_teams\_three\_letters.append(item[:3].upper())

# Print the list of NBA team names with only their first 3 letters in uppercase

print(nba\_teams\_three\_letters)

# Loop through each NBA team abbreviation and replace any abbreviations that are incorrect or outdated

for i in range(0, len(nba\_teams\_three\_letters)):

if nba\_teams\_three\_letters[i] == "BRO":

nba\_teams\_three\_letters[i] = "BRK"

if nba\_teams\_three\_letters[i] == "CHA":

nba\_teams\_three\_letters[i] = "CHO"

if nba\_teams\_three\_letters[i] == "GOL":

nba\_teams\_three\_letters[i] = "GSW"

if nba\_teams\_three\_letters[i] == "LOS":

nba\_teams\_three\_letters[i] = "LAL"

if nba\_teams\_three\_letters[i] == "LOS":

nba\_teams\_three\_letters[i] = "LAC"

if nba\_teams\_three\_letters[i] == "NEW":

nba\_teams\_three\_letters[i] = "NYK"

if nba\_teams\_three\_letters[i] == "NEW":

nba\_teams\_three\_letters[i] = "NOP"

if nba\_teams\_three\_letters[i] == "OKL":

nba\_teams\_three\_letters[i] = "OKC"

if nba\_teams\_three\_letters[i] == "SAN":

nba\_teams\_three\_letters[i] = "SAS"

# Print the updated list of NBA team abbreviations

print(nba\_teams\_three\_letters)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Initializing variables for web scraping

new\_web\_links = ""

players\_colleges\_more = []

# Iterating over the NBA teams and dynamically generating URLs to the webpages that contain their respective rosters

for i in range(0, len(nba\_teams\_three\_letters)):

new\_web\_links = "https://www.basketball-reference.com/teams/" + nba\_teams\_three\_letters[i] + "/2023.html"

webpage2 = requests.get(new\_web\_links)

soup = BeautifulSoup(webpage2.content, 'html.parser')

# Checking if the webpage was found

if soup.find("p").text == "We apologize, but we could not find the page requested by your device.":

print(new\_web\_links)

# Extracting the colleges of all the players on the team and appending them to the list players\_colleges\_more

players = soup.find\_all("td", {"data-stat" : "college"})

for i in range(0, len(players)):

for item in players[i].find\_all("a"):

players\_colleges\_more.append(item.text)

print(players\_colleges\_more)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Create an empty dictionary called players\_colleges\_more\_dict

players\_colleges\_more\_dict = {}

# Iterate through the list players\_colleges\_more

for i in players\_colleges\_more:

# If the current element i is already a key in the dictionary, increment the associated value by 1

if i in players\_colleges\_more\_dict:

players\_colleges\_more\_dict[i] += 1

# If the current element i is not a key in the dictionary, add a new key-value pair with a value of 1

else:

players\_colleges\_more\_dict[i] = 1

# Print out the resulting dictionary

print(players\_colleges\_more\_dict)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Sort the players\_colleges\_more\_dict in descending order based on the values of the dictionary items

sorted\_players\_colleges = sorted(players\_colleges\_more\_dict.items(), key=lambda x:x[1], reverse=True)

# Convert the sorted list of tuples back into a dictionary

sorted\_players\_colleges = dict(sorted\_players\_colleges)

# Extract the top 10 items as a new dictionary

top\_10\_sorted\_players\_colleges = {k: sorted\_players\_colleges[k] for k in list(sorted\_players\_colleges)[:10]}

# Extract the top 5 items as a new dictionary

top\_5\_sorted\_players\_colleges = {k: sorted\_players\_colleges[k] for k in list(sorted\_players\_colleges)[:5]}

# Print out the top 10 and top 5 dictionaries

print(top\_10\_sorted\_players\_colleges)

print(top\_5\_sorted\_players\_colleges)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Extract the keys and values from the top\_10\_sorted\_players\_colleges dictionary and store them in the names and values lists

names = list(top\_10\_sorted\_players\_colleges.keys())

values = list(top\_10\_sorted\_players\_colleges.values())

# Generate a list of 10 random hexadecimal color codes and store them in the colors list

colors = []

for i in range(len(top\_10\_sorted\_players\_colleges)):

colors.append('#%06X' % random.randint(0, 0xFFFFFF))

# Add text labels to each bar in the chart using the plt.text() function

for i, v in enumerate(values):

plt.text(i, v, str(v), color='black', ha='center', fontweight='bold')

# Rotate the x-axis labels by 45 degrees

plt.xticks(rotation=45)

# Set the title, x-axis label, and y-axis label of the chart

plt.title('Top 10 Producing Colleges for NBA Players')

plt.xlabel('Universities')

plt.ylabel('Amount of Players')

# Create a bar chart with the tick\_label parameter set to the names list and the color parameter set to the colors list

plt.bar(range(len(top\_10\_sorted\_players\_colleges)), values, tick\_label=names, color=colors)

# Display the chart

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Extract the keys and values from the top\_5\_sorted\_players\_colleges dictionary and store them in the names and values lists

names = list(top\_5\_sorted\_players\_colleges.keys())

values = list(top\_5\_sorted\_players\_colleges.values())

# Generate a list of 5 random hexadecimal color codes and store them in the colors list

colors = []

for i in range(len(top\_5\_sorted\_players\_colleges)):

colors.append('#%06X' % random.randint(0, 0xFFFFFF))

# Add text labels to each bar in the chart using the plt.text() function

for i, v in enumerate(values):

plt.text(i, v, str(v), color='black', ha='center', fontweight='bold')

# Rotate the x-axis labels by 45 degrees

plt.xticks(rotation=45)

# Set the title, x-axis label, and y-axis label of the chart

plt.title('Top 5 Producing Colleges for NBA Players')

plt.xlabel('Universities')

plt.ylabel('Amount of Players')

# Create a bar chart with the tick\_label parameter set to the names list and the color parameter set to the colors list

plt.bar(range(len(top\_5\_sorted\_players\_colleges)), values, tick\_label=names, color = colors)

# Display the chart

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import pandas as pd

# Create a sample DataFrame

data = {'Country': ['USA', 'USA', 'USA', 'Canada', 'Canada', 'Mexico'],

'Year': [2019, 2020, 2021, 2019, 2020, 2021],

'GDP': [21.4, 21.5, 22.3, 1.7, 1.8, 1.9],

'Population': [328.2, 331.0, 333.7, 37.6, 37.9, 130.2]}

df = pd.DataFrame(data)

# Set the hierarchical index

df = df.set\_index(['Country', 'Year'])

# Print the DataFrame with hierarchical indexing

print(df)

**QUESTION 2- Which age group has scored the highest amount of average points this 2022-2023 season?**

# Import the requests library for making HTTP requests

import requests

# Import the BeautifulSoup library for web scraping and parsing HTML

from bs4 import BeautifulSoup

# Import the pandas library for data manipulation and analysis

import pandas as pd

# Import the pyplot module of the matplotlib library for data visualization

import matplotlib.pyplot as plt

# Import the NumPy library for numerical computing with Python

import numpy as np

# Import the random module for generating random numbers

import random

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# URL of the web page to scrape

url = 'https://www.basketball-reference.com/leagues/NBA\_2023\_totals.html'

# Send an HTTP request to the URL

response = requests.get(url)

# Parse the HTML content of the page with BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Find the HTML table containing the player stats and extract the rows

table = soup.find('table', {'id': 'totals\_stats'})

rows = table.find\_all('tr')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# URL of the web page to scrape

url = 'https://www.basketball-reference.com/leagues/NBA\_2023\_totals.html'

# Send an HTTP request to the URL

response = requests.get(url)

# Parse the HTML content of the page with BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')

# Find the HTML table containing the player stats and extract the rows

table = soup.find('table', {'id': 'totals\_stats'})

rows = table.find\_all('tr')

# Initialize arrays to store the total points and number of players for each age group

age\_groups\_points = np.zeros(50, dtype=int)

age\_groups\_players = np.zeros(50, dtype=int)

# Iterate over the rows in the table (skipping the header row)

for row in rows[1:]:

# Check that the row has at least 29 columns

if len(row.find\_all('td')) >= 29:

# Extract the player's age as an integer

age = int(row.find\_all('td')[2].text)

# Extract the player's total points as an integer

points = int(row.find\_all('td')[28].text)

# Add the player's points and increment the player count for the corresponding age group

age\_groups\_points[age] += points

age\_groups\_players[age] += 1

# Compute the average points for each age group and print the results

max\_age = 0

max\_avg\_points = 0

for age in range(len(age\_groups\_points)):

if age\_groups\_players[age] > 0:

avg\_points = age\_groups\_points[age] / age\_groups\_players[age]

print(f"Age {age}: {avg\_points:.1f} points per player ({age\_groups\_players[age]} players)")

if avg\_points > max\_avg\_points:

max\_avg\_points = avg\_points

max\_age = age

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Create a bar graph of the total points for each age group

plt.bar(range(50), age\_groups\_points, color='blue')

max\_value = max(age\_groups\_points)

# Iterate over each bar in the chart

for i in range(len(age\_groups\_points)):

# If the bar's height matches the highest value

if age\_groups\_points[i] == max\_value:

# Set its color to red

plt.gca().get\_children()[i].set\_color('red')

# Add labels and title to the graph

plt.xlabel('Age')

plt.ylabel('Total points')

plt.title('Total points scored by age group during the 22-23 NBA season')

# Set the x-ticks

plt.xticks(np.arange(0, 52, 2))

# Show the graph

plt.show()

# Print the age group that scores the highest average points

print(f"Players age {max\_age} scored the highest average points ({max\_avg\_points:.1f} points per player).")

**QUESTION 3**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*