Olympics_EDA

August 22, 2024

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
data = pd.read_csv(r"C:\Users\Anton Dominic\Downloads\dataset_olympics.
↪csv\dataset_olympics.csv")
```

[3]:
```
data.head()
```

[3]:
```
   ID                     Name Sex   Age  Height  Weight            Team  \
0   1                A Dijiang   M  24.0   180.0    80.0           China
1   2                 A Lamusi   M  23.0   170.0    60.0           China
2   3      Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
3   4     Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands

    NOC        Games  Year  Season       City          Sport  \
0  CHN  1992 Summer  1992  Summer  Barcelona     Basketball
1  CHN  2012 Summer  2012  Summer     London           Judo
2  DEN  1920 Summer  1920  Summer  Antwerpen       Football
3  DEN  1900 Summer  1900  Summer      Paris     Tug-Of-War
4  NED  1988 Winter  1988  Winter    Calgary  Speed Skating

                            Event Medal
0       Basketball Men's Basketball   NaN
1      Judo Men's Extra-Lightweight   NaN
2           Football Men's Football   NaN
3       Tug-Of-War Men's Tug-Of-War  Gold
4  Speed Skating Women's 500 metres   NaN
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      70000 non-null  int64
```

```
 1   Name    70000 non-null  object
 2   Sex     70000 non-null  object
 3   Age     67268 non-null  float64
 4   Height  53746 non-null  float64
 5   Weight  52899 non-null  float64
 6   Team    70000 non-null  object
 7   NOC     70000 non-null  object
 8   Games   70000 non-null  object
 9   Year    70000 non-null  int64
 10  Season  70000 non-null  object
 11  City    70000 non-null  object
 12  Sport   70000 non-null  object
 13  Event   70000 non-null  object
 14  Medal   9690 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 8.0+ MB
```

[5]:
```
data.describe()
```

[5]:

	ID	Age	Height	Weight	Year
count	70000.000000	67268.000000	53746.000000	52899.000000	70000.000000
mean	18081.846986	25.644645	175.505303	70.900216	1977.766457
std	10235.613253	6.485239	10.384203	14.217489	30.103306
min	1.000000	11.000000	127.000000	25.000000	1896.000000
25%	9325.750000	21.000000	168.000000	61.000000	1960.000000
50%	18032.000000	25.000000	175.000000	70.000000	1984.000000
75%	26978.000000	28.000000	183.000000	79.000000	2002.000000
max	35658.000000	88.000000	223.000000	214.000000	2016.000000

[6]:
```
data.describe(include =["object"])
```

[6]:

	Name	Sex	Team	NOC	\
count	70000	70000	70000	70000	
unique	35556	2	827	226	
top	Oksana Aleksandrovna Chusovitina	M	United States	USA	
freq	29	51877	4979	5216	

	Games	Season	City	Sport	Event	Medal
count	70000	70000	70000	70000	70000	9690
unique	51	2	42	65	744	3
top	2016 Summer	Summer	London	Athletics	Football Men's Football	Gold
freq	3675	58467	6034	10629	1738	3292

[7]:
```
data.isna().sum()
```

[7]:
```
ID            0
Name          0
Sex           0
```

```
Age        2732
Height    16254
Weight    17101
Team          0
NOC           0
Games         0
Year          0
Season        0
City          0
Sport         0
Event         0
Medal     60310
dtype: int64
```

[8]:
```
data.duplicated().sum()
```

[8]:
```
383
```

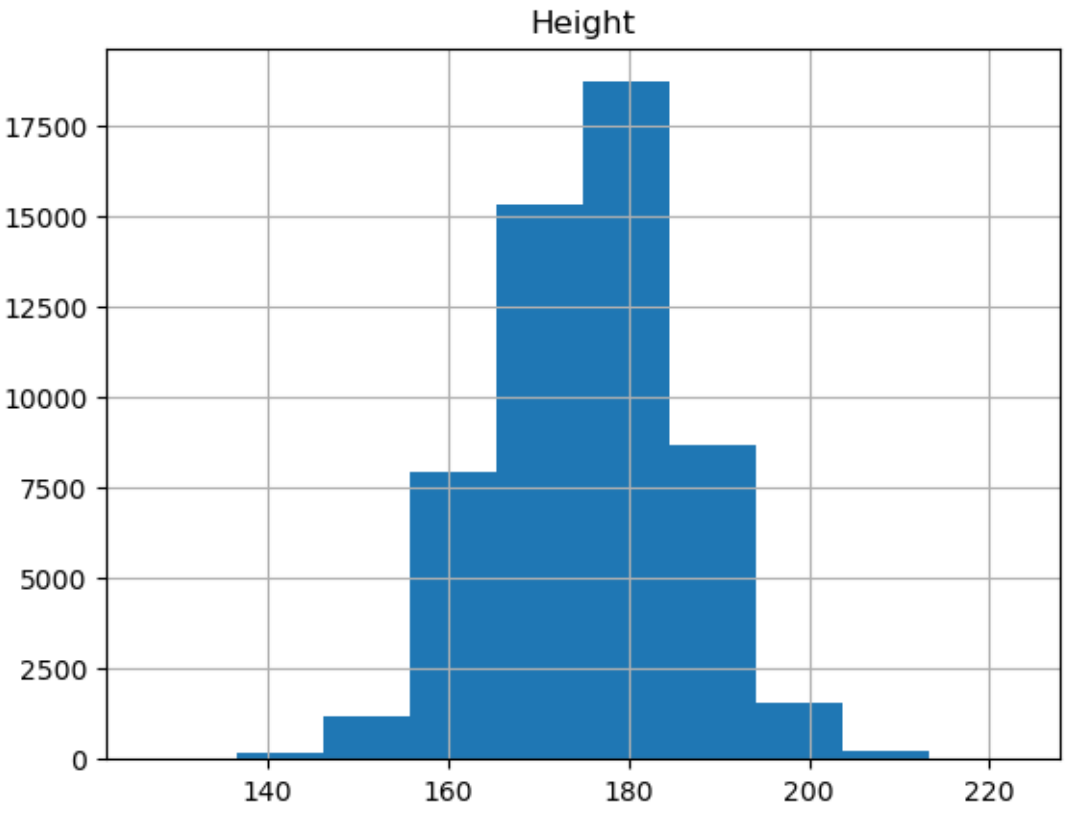
[9]:
```
data.drop_duplicates(inplace = True)
```

[10]:
```
data.duplicated().sum()
```

[10]:
```
0
```

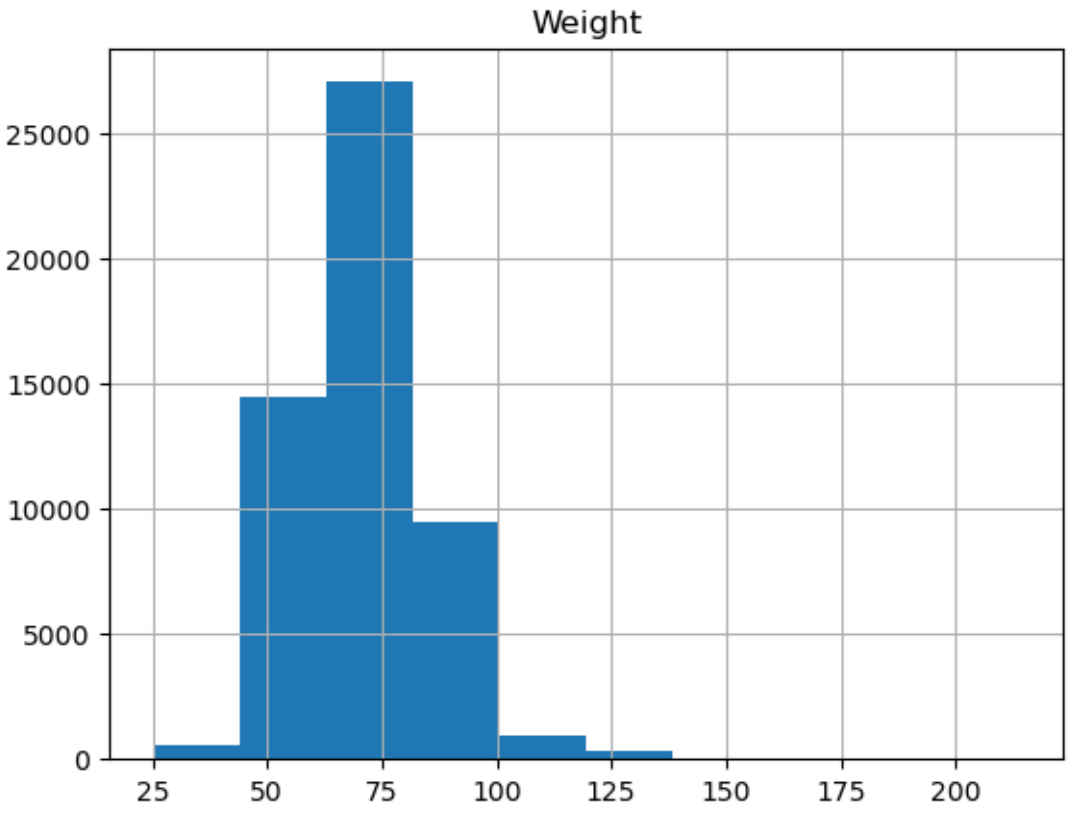
[11]:
```
data.hist(column="Height")
```

[11]:
```
array([[<Axes: title={'center': 'Height'}>]], dtype=object)
```



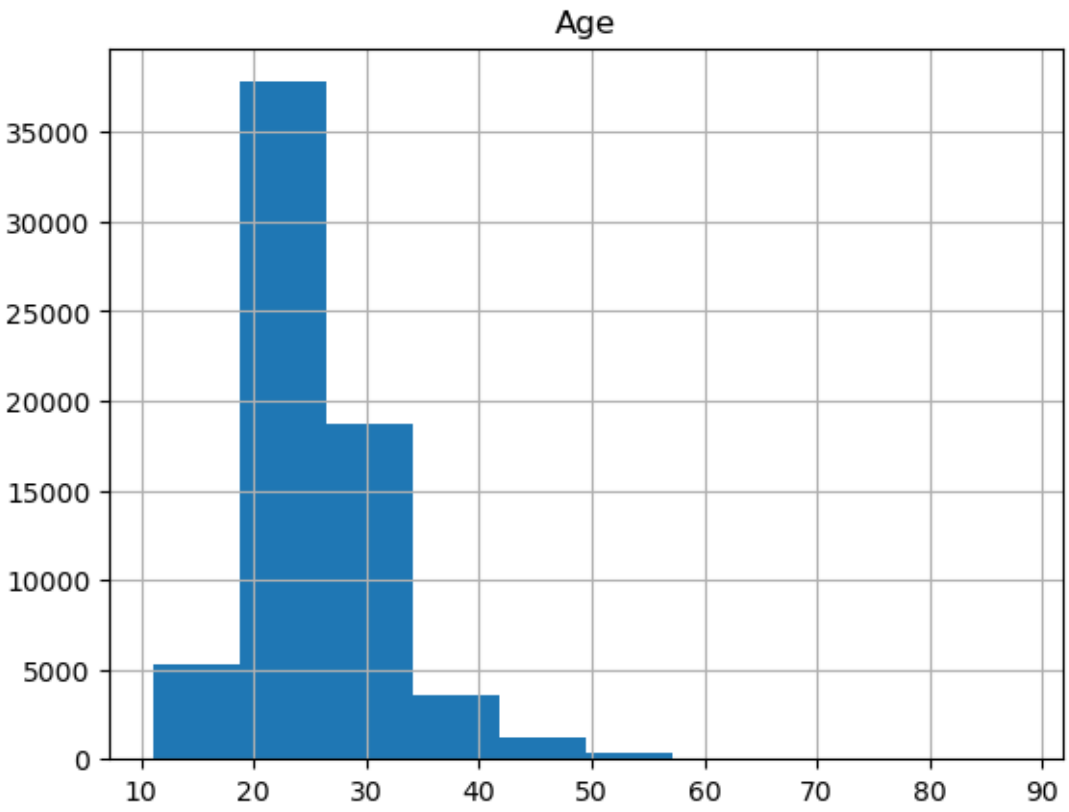
[12]:
```
data.hist(column="Weight" )
```

[12]:
```
array([[<Axes: title={'center': 'Weight'}>]], dtype=object)
```



[13]:
```
data.hist(column="Age" )
```

[13]:
```
array([[<Axes: title={'center': 'Age'}>]], dtype=object)
```



[14]:
```
mean_height = data.groupby(['NOC','Sport'])['Height'].transform('mean')

data['Height'] = data['Height'].fillna(mean_height)

mean_Weight = data.groupby(['NOC','Sport'])['Weight'].transform('mean')

data['Weight'] = data['Weight'].fillna(mean_height)

mean_age = data.groupby(['NOC','Sport'])['Age'].transform('mean')

data['Age'] = data['Age'].fillna(mean_age)
```

[15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 69617 entries, 0 to 69999
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      69617 non-null  int64
 1   Name    69617 non-null  object
```

```
 2   Sex     69617 non-null  object
 3   Age     69504 non-null  float64
 4   Height  68306 non-null  float64
 5   Weight  68346 non-null  float64
 6   Team    69617 non-null  object
 7   NOC     69617 non-null  object
 8   Games   69617 non-null  object
 9   Year    69617 non-null  int64
 10  Season  69617 non-null  object
 11  City    69617 non-null  object
 12  Sport   69617 non-null  object
 13  Event   69617 non-null  object
 14  Medal   9686 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 8.5+ MB
```

[16]:
```
mean_height = data.groupby('Sport')['Height'].transform('mean')

data['Height'] = data['Height'].fillna(mean_height)

mean_weight = data.groupby('Sport')['Weight'].transform('mean')

data['Weight'] = data['Weight'].fillna(mean_weight)

mean_age = data.groupby('Sport')['Age'].transform('mean')

data['Age'] = data['Age'].fillna(mean_weight)
```

[17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 69617 entries, 0 to 69999
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      69617 non-null  int64
 1   Name    69617 non-null  object
 2   Sex     69617 non-null  object
 3   Age     69613 non-null  float64
 4   Height  69527 non-null  float64
 5   Weight  69527 non-null  float64
 6   Team    69617 non-null  object
 7   NOC     69617 non-null  object
 8   Games   69617 non-null  object
 9   Year    69617 non-null  int64
 10  Season  69617 non-null  object
 11  City    69617 non-null  object
 12  Sport   69617 non-null  object
```

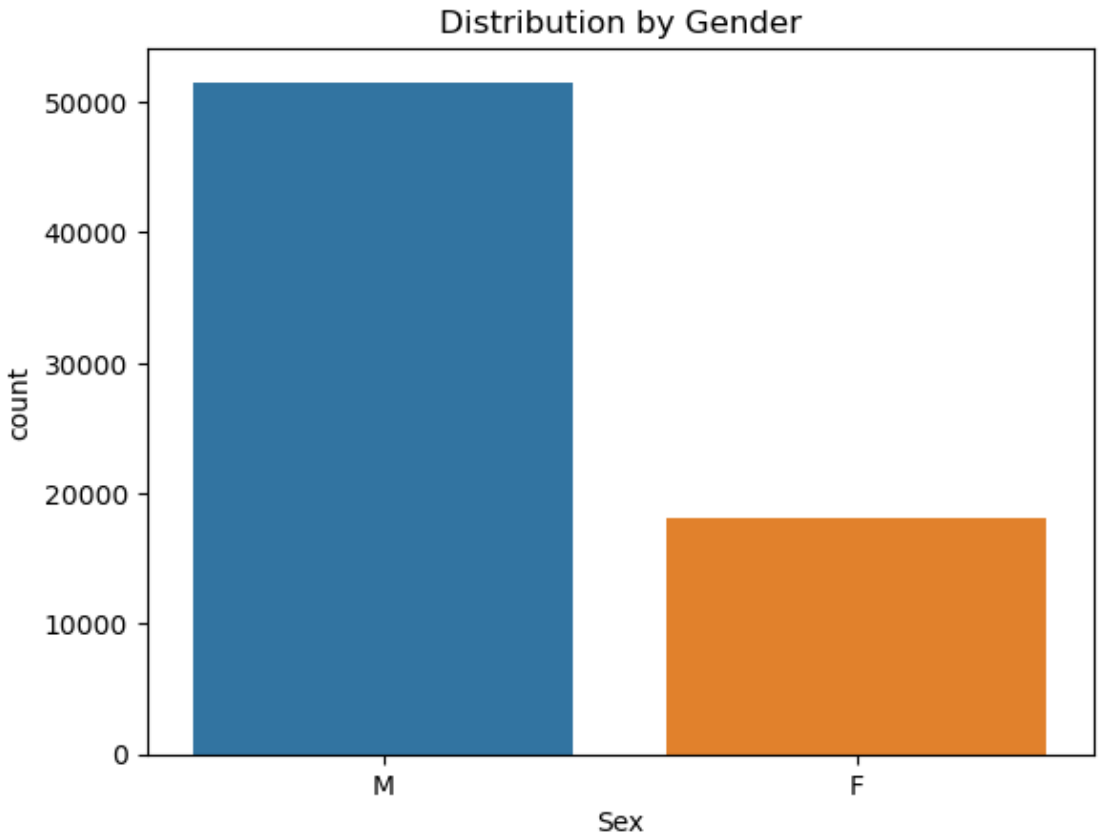
```
 13  Event   69617 non-null  object
 14  Medal   9686 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 8.5+ MB
```

[18]:
```
data['Height'].fillna(data['Height'].mean(), inplace=True)

data['Weight'].fillna(data['Weight'].mean(), inplace=True)

data['Age'].fillna(data['Age'].mean(), inplace=True)
```

[19]:
```
data.info()
```

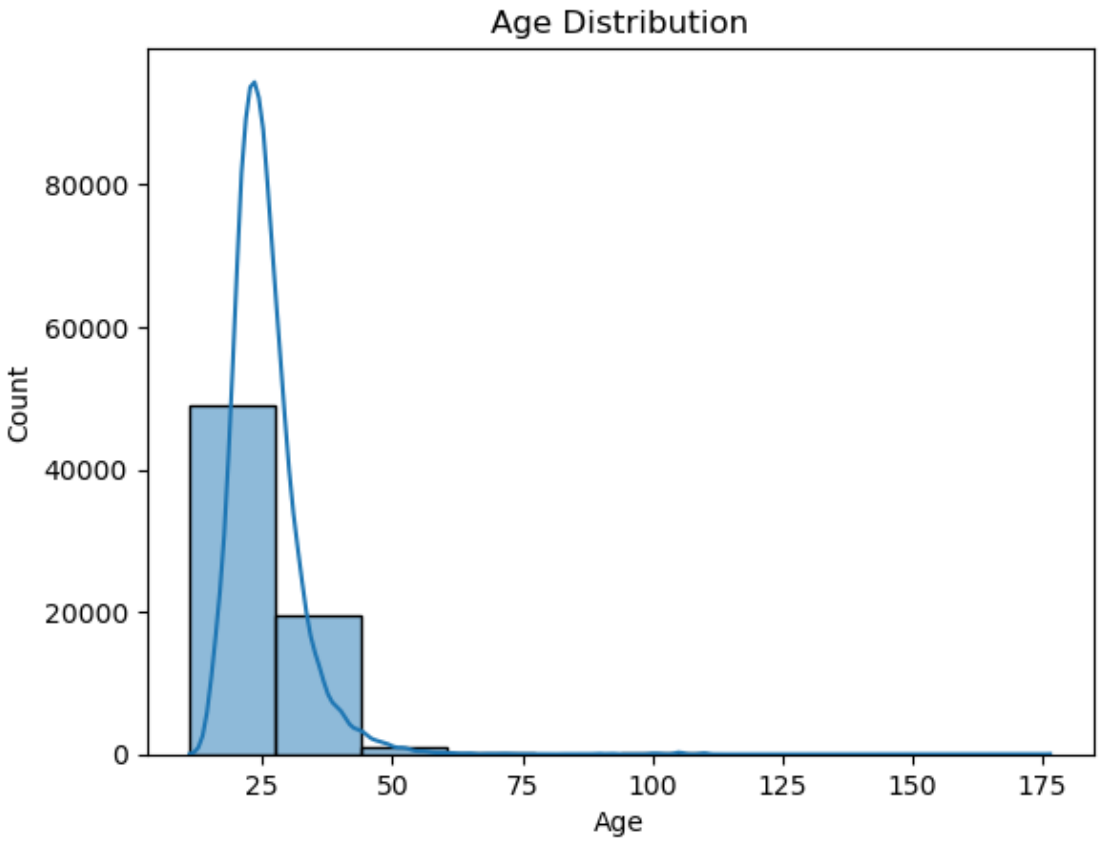
```
<class 'pandas.core.frame.DataFrame'>
Index: 69617 entries, 0 to 69999
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      69617 non-null  int64
 1   Name    69617 non-null  object
 2   Sex     69617 non-null  object
 3   Age     69617 non-null  float64
 4   Height  69617 non-null  float64
 5   Weight  69617 non-null  float64
 6   Team    69617 non-null  object
 7   NOC     69617 non-null  object
 8   Games   69617 non-null  object
 9   Year    69617 non-null  int64
 10  Season  69617 non-null  object
 11  City    69617 non-null  object
 12  Sport   69617 non-null  object
 13  Event   69617 non-null  object
 14  Medal   9686 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 8.5+ MB
```

[20]:
```
sns.countplot(data=data, x = "Sex")
plt.title("Distribution by Gender")
plt.show()
```



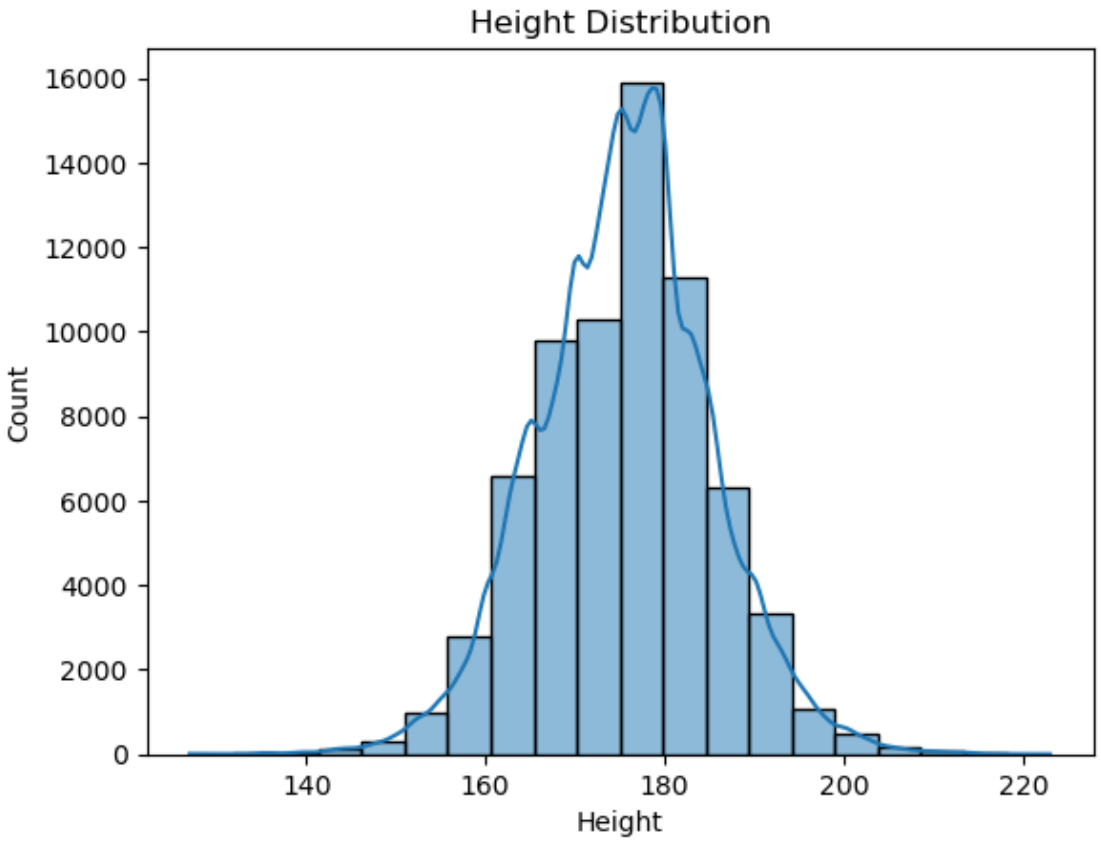
[21]:
```
sns.histplot(data=data, x="Age", bins=10, kde=True)
plt.title("Age Distribution")
plt.show()
```

```
C:\Users\Anton Dominic\anaconda3\2024\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



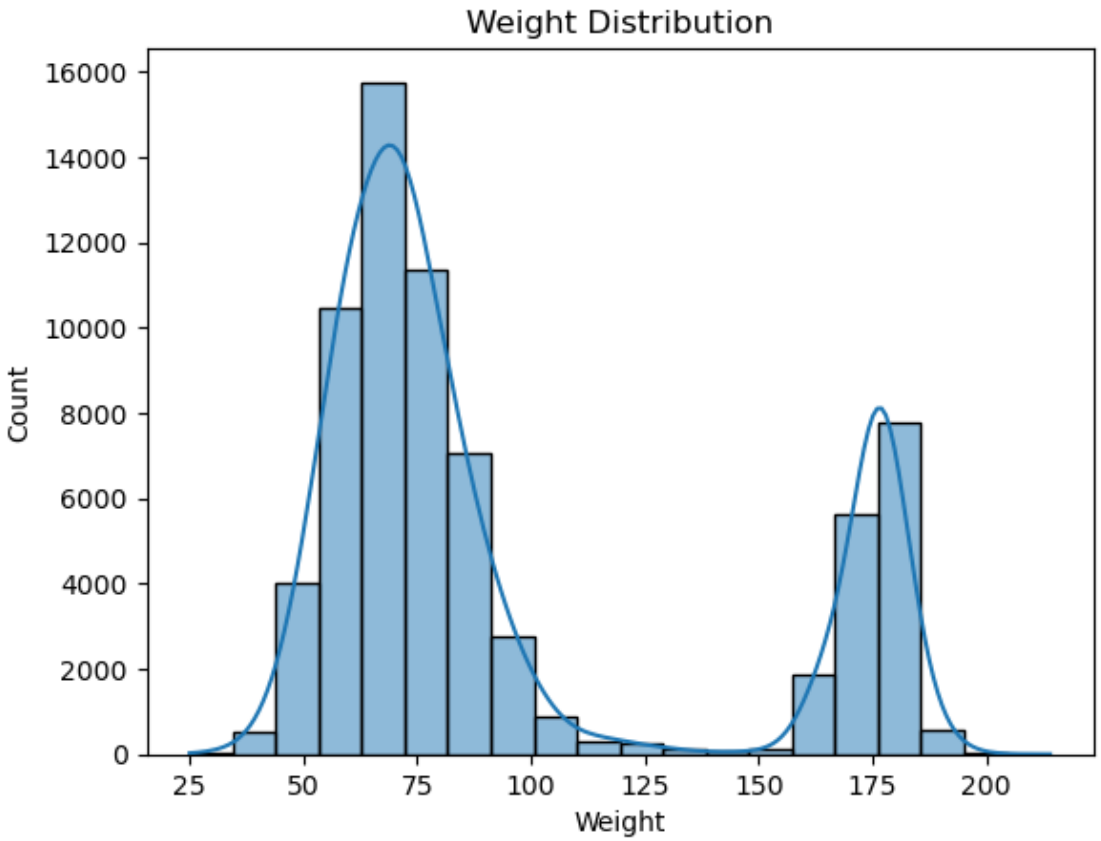
[22]:
```
sns.histplot(data=data, x="Height", bins = 20, kde = True)
plt.title("Height Distribution")
plt.show()
```

```
C:\Users\Anton Dominic\anaconda3\2024\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

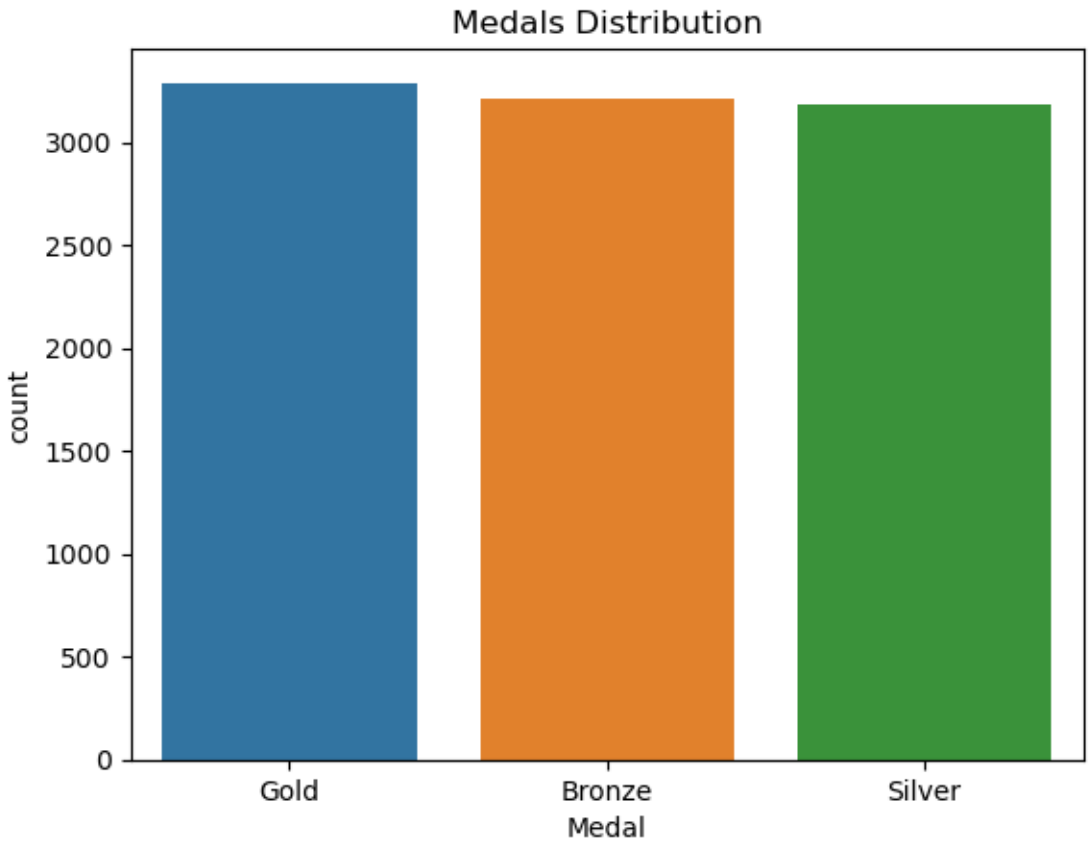


[23]:
```
sns.histplot(data=data, x="Weight", bins = 20, kde = True)
plt.title("Weight Distribution")
plt.show()
```

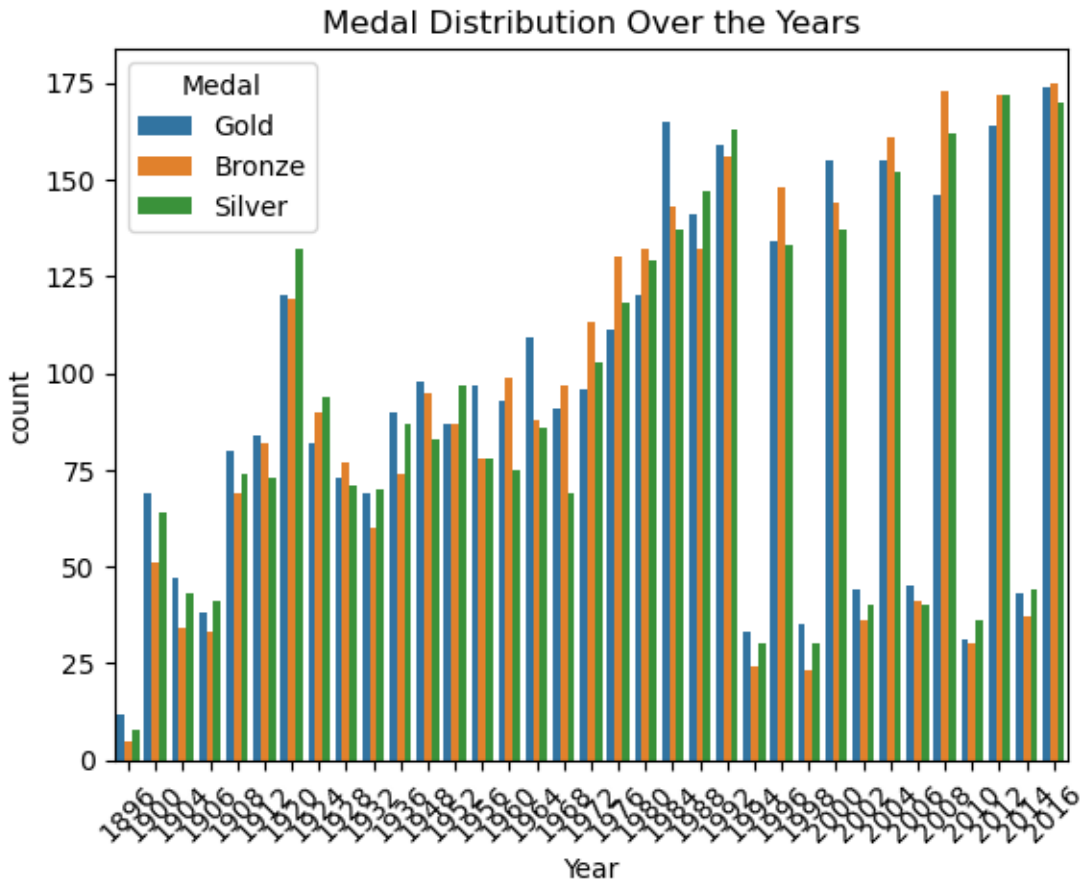
```
C:\Users\Anton Dominic\anaconda3\2024\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[24]:
```
sns.countplot(data=data, x="Medal")
plt.title("Medals Distribution")
plt.show()
```



[25]:
```
sns.countplot(data=data, x="Year", hue = "Medal")
plt.title("Medal Distribution Over the Years")
plt.xticks(rotation = 45)
plt.show()
```



[26]:
```
year_avg_age = data.groupby("Year")["Age"].mean()
year_avg_age
```

[26]:
```
Year
1896    24.614659
1900    28.508720
1904    27.587250
1906    28.061596
1908    26.914333
1912    28.174117
1920    28.974377
1924    28.712334
1928    27.705512
1932    29.838425
1936    27.353404
1948    28.448906
1952    26.246641
1956    27.259566
1960    25.540181
```

```
1964    24.872815
1968    24.391763
1972    24.295069
1976    23.704170
1980    23.599982
1984    24.125350
1988    24.419363
1992    24.760450
1994    24.487516
1996    25.338082
1998    25.143860
2000    25.435177
2002    26.029095
2004    25.780111
2006    26.091716
2008    25.681015
2010    26.150776
2012    25.993485
2014    26.082814
2016    26.259592
Name: Age, dtype: float64
```

[27]:
```
sport_median_height = data.groupby("Sport")["Height"].median()
sport_median_height
```

[27]:
```
Sport
Alpine Skiing       173.000000
Alpinism            175.506103
Archery             173.000000
Art Competitions    179.449123
Athletics           176.611364
                          …
Tug-Of-War          183.239130
Volleyball          187.285899
Water Polo          185.000000
Weightlifting       168.000000
Wrestling           173.000000
Name: Height, Length: 65, dtype: float64
```

[28]:
```
sport_median_height.max()
```

[28]:
```
190.5
```

[29]:
```
sport_median_height[sport_median_height == 190.5]
```

[29]:
```
Sport
Basketball    190.5
```

```
Name: Height, dtype: float64
```

[30]:
```
sport_median_height.min()
```

[30]:
```
164.0
```

[31]:
```
sport_median_height[sport_median_height == 164]
```

[31]:
```
Sport
Gymnastics    164.0
Name: Height, dtype: float64
```

[32]:
```
country_gender_count = data.groupby(["NOC", "Sex"])["ID"].count()
country_gender_count
```

[32]:
```
NOC  Sex
AFG  M       38
AHO  F        6
     M       27
ALB  F        4
     M        7
           ...
YUG  M      455
ZAM  F        3
     M       40
ZIM  F       41
     M       47
Name: ID, Length: 432, dtype: int64
```

[33]:
```
country_gold_medals = data[data["Medal"] == "Gold"].groupby("NOC")["Medal"].
 ↪count()
country_gold_medals
```

[33]:
```
NOC
ALG      1
ANZ      7
ARG     25
ARM      1
AUS     98
       ...
URU     13
USA    747
UZB      4
YUG     31
ZIM      7
Name: Medal, Length: 84, dtype: int64
```

[34]:
```
country_gold_medals.max()
```

[34]:
```
747
```

[35]:
```
country_gold_medals[country_gold_medals == 747]
```

[35]:
```
NOC
USA    747
Name: Medal, dtype: int64
```

[36]:
```
country_gold_medals.min()
```

[36]:
```
1
```

[37]:
```
country_gold_medals[country_gold_medals == 1]
```

[37]:
```
NOC
ALG    1
ARM    1
AZE    1
CIV    1
COL    1
DOM    1
GEO    1
IOA    1
IRI    1
JOR    1
LUX    1
MGL    1
NEP    1
POR    1
THA    1
UAE    1
UGA    1
Name: Medal, dtype: int64
```

[38]:
```
sport_gender_avg_weight = data.groupby(["Sport", "Sex"])["Weight"].mean()
sport_gender_avg_weight
```

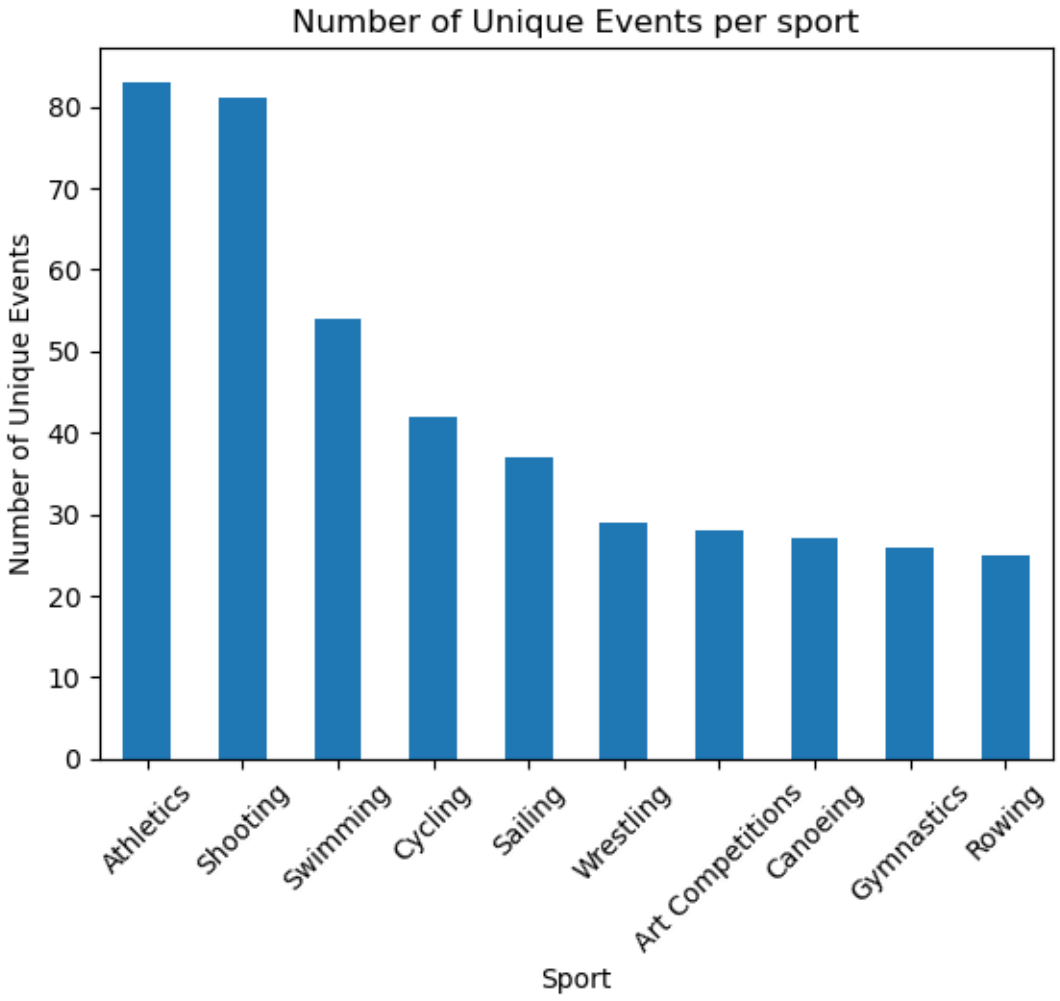
[38]:
```
Sport          Sex
Alpine Skiing  F       88.347534
               M      110.909026
Alpinism       F       95.164576
               M       95.164576
Archery        F       74.040303
                          …
Water Polo     M      119.208811
Weightlifting  F       66.189474
               M       82.459754
Wrestling      F       58.169014
```

```
                M      102.698392
Name: Weight, Length: 114, dtype: float64
```

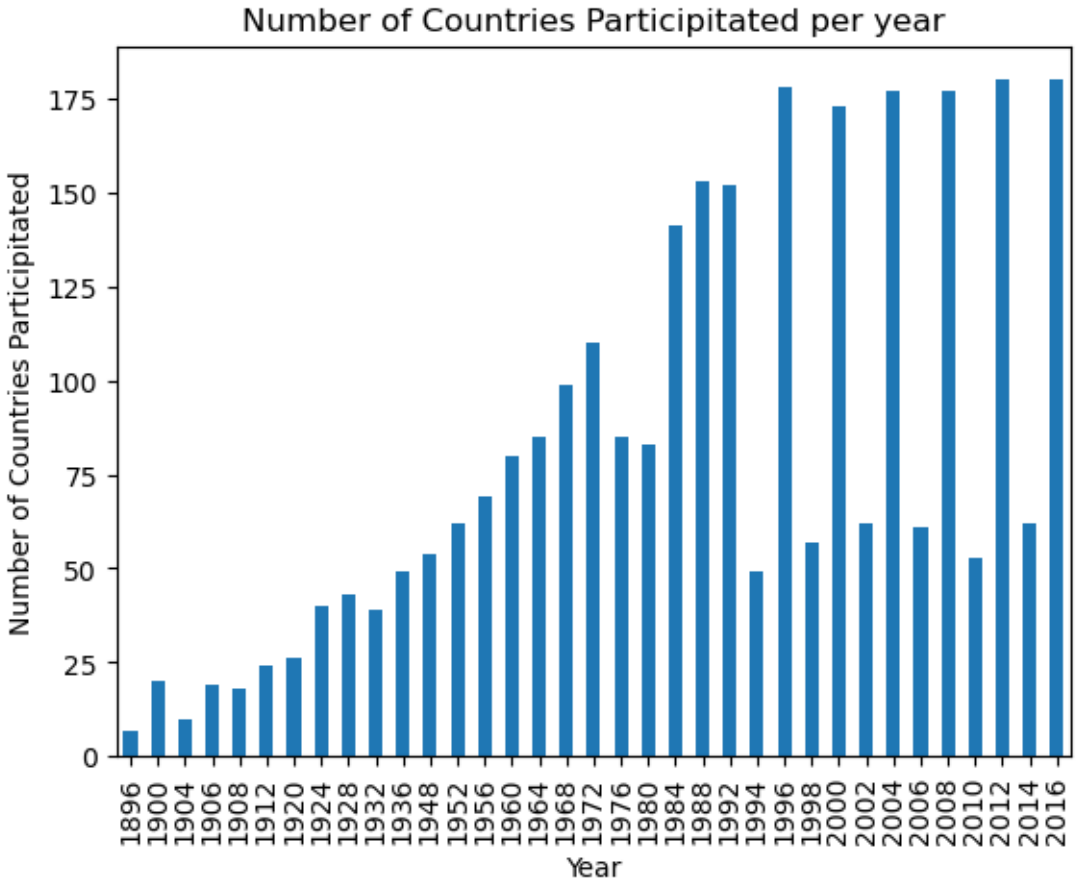
[39]:
```
sport_gender_avg_weight["Weightlifting"]["F"]
```

[39]:
```
66.18947368421053
```

[40]:
```
sport_event_count = data.groupby("Sport")["Event"].nunique().
 ↪sort_values(ascending = False)
sport_event_count.head(10).plot(kind = "bar")
plt.title("Number of Unique Events per sport")
plt.xlabel("Sport")
plt.ylabel("Number of Unique Events")
plt.xticks(rotation = 45)
plt.show()
```



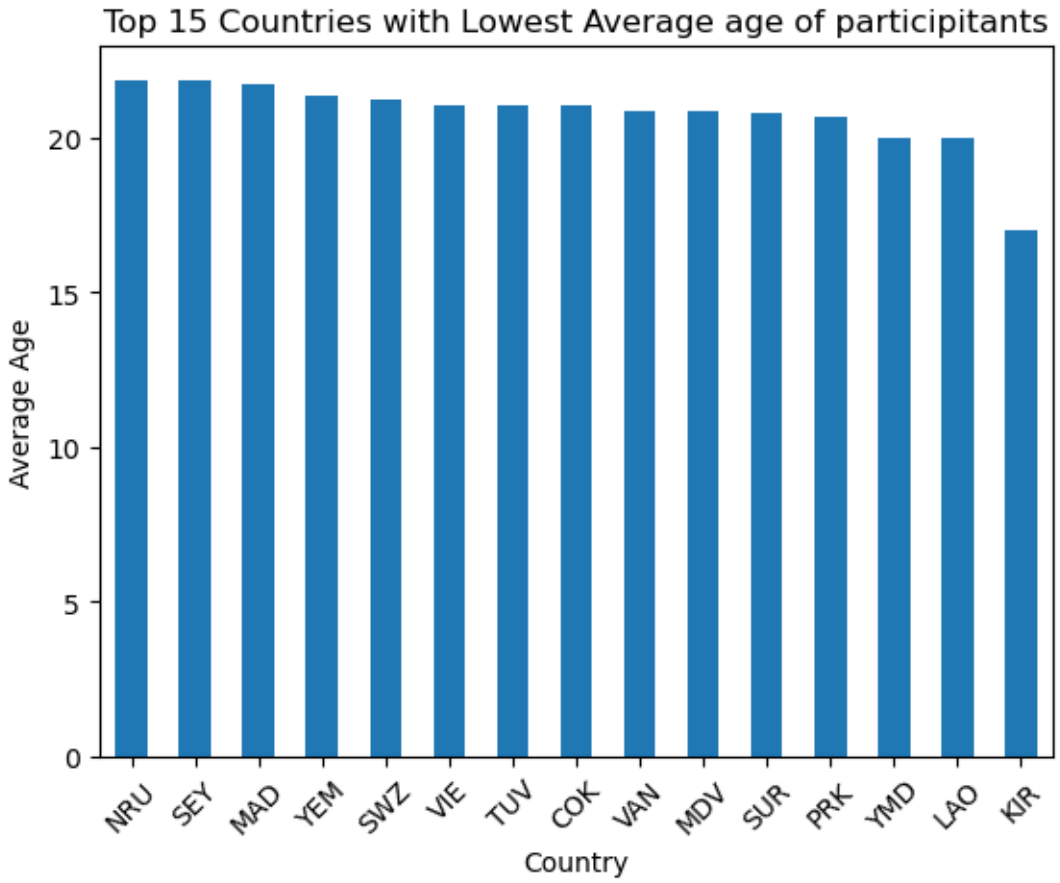
[41]:
```
year_participant_count = data.groupby("Year")["NOC"].nunique()
year_participant_count.plot(kind = "bar")
plt.title("Number of Countries Participitated per year")
plt.xlabel("Year")
plt.ylabel("Number of Countries Participitated")
plt.show()
```



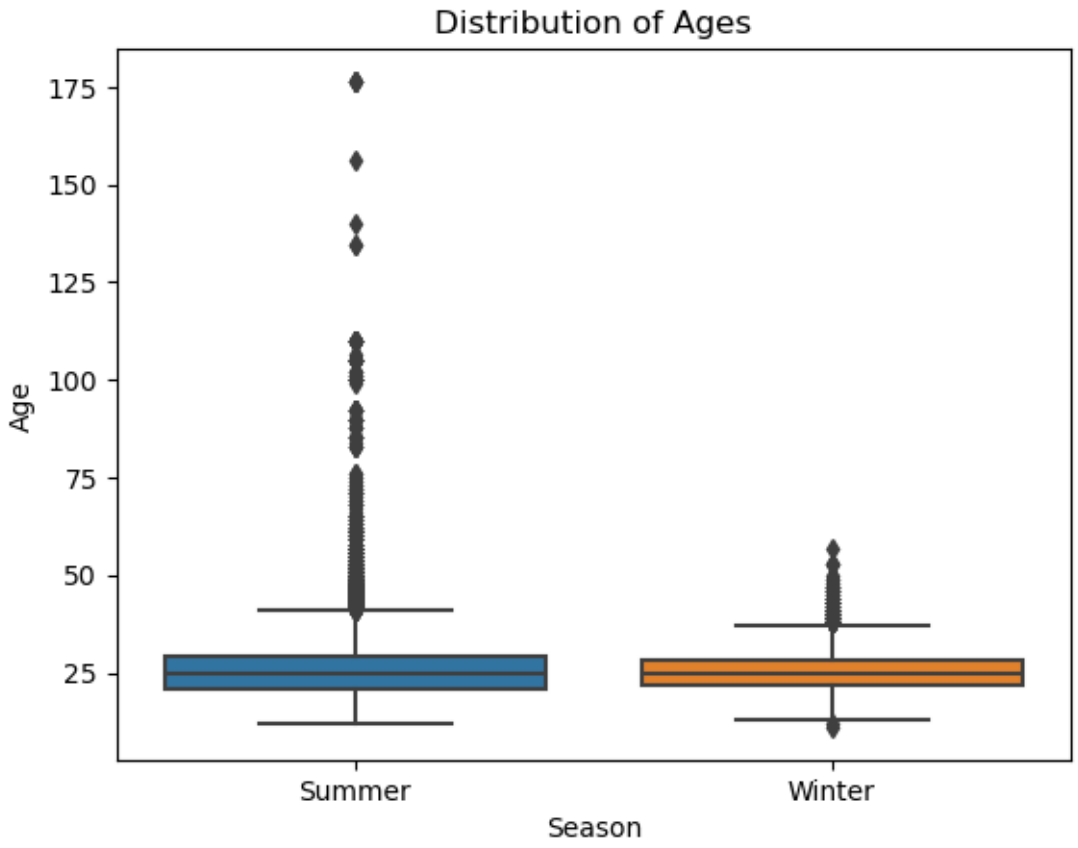
[42]:
```
country_avg_age = data.groupby("NOC")["Age"].mean().sort_values(ascending =␣
↪False)
country_avg_age.tail(15).plot(kind="bar")
plt.title("Top 15 Countries with Lowest Average age of participitants")
plt.xlabel("Country")
plt.ylabel("Average Age")
plt.xticks(rotation = 45)
```

[42]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(0, 0, 'NRU'),
  Text(1, 0, 'SEY'),
```

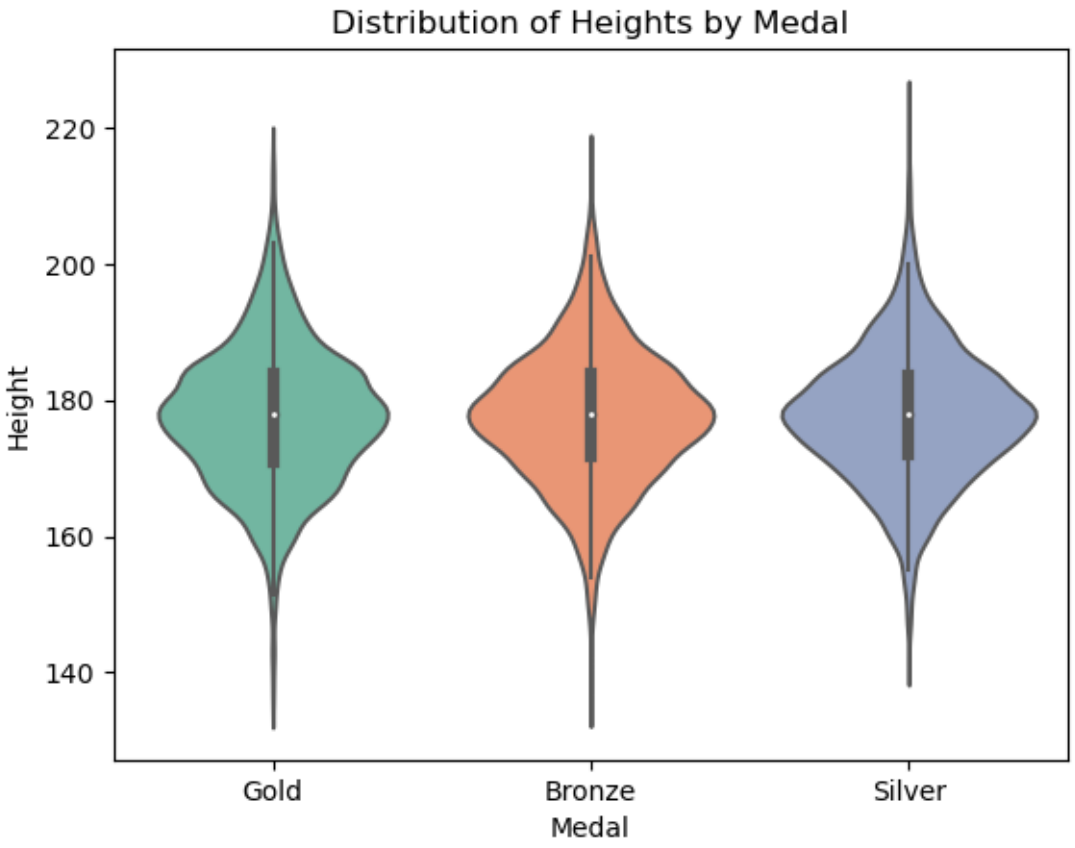
```
Text(2, 0, 'MAD'),
Text(3, 0, 'YEM'),
Text(4, 0, 'SWZ'),
Text(5, 0, 'VIE'),
Text(6, 0, 'TUV'),
Text(7, 0, 'COK'),
Text(8, 0, 'VAN'),
Text(9, 0, 'MDV'),
Text(10, 0, 'SUR'),
Text(11, 0, 'PRK'),
Text(12, 0, 'YMD'),
Text(13, 0, 'LAO'),
Text(14, 0, 'KIR')])
```



[43]:
```
sns.boxplot(data = data, x= "Season", y = "Age")
plt.title("Distribution of Ages")
plt.xlabel("Season")
plt.ylabel("Age")
plt.show()
```



[44]:
```
sns.violinplot(data=data, x= "Medal", y = "Height", palette = "Set2")
plt.title("Distribution of Heights by Medal")
plt.xlabel("Medal")
plt.ylabel("Height")
plt.show()
```



[45]:
```
most_medals_country = data["NOC"].value_counts().idxmax()
print("Most Medal-Winning Country: ", most_medals_country)
```

```
Most Medal-Winning Country:  USA
```

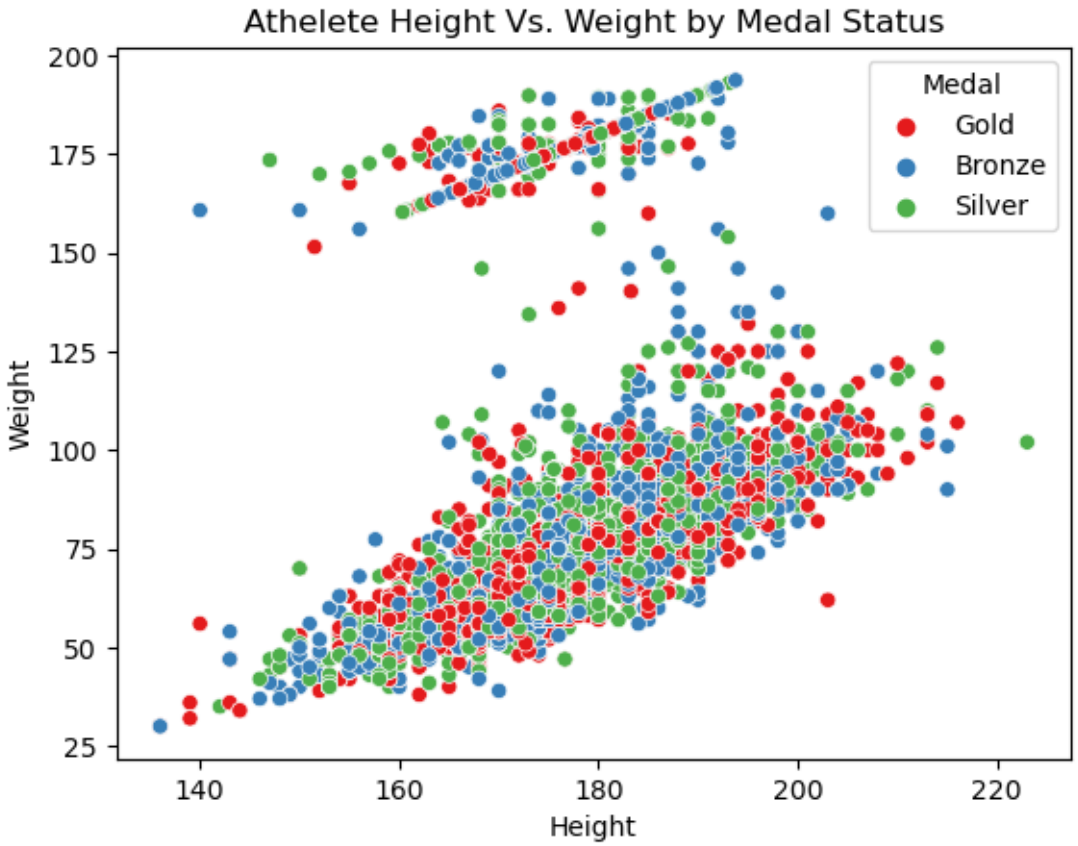
[46]:
```
tallest_athelete = data[data["Height"] == data["Height"].max()]
print("Tallest Athelete:")
print(tallest_athelete[["ID", "Name", "Height", "Sport"]])
```

```
Tallest Athelete:
          ID                   Name  Height       Sport
32376  16639  Tommy Loren Burleson   223.0  Basketball
```

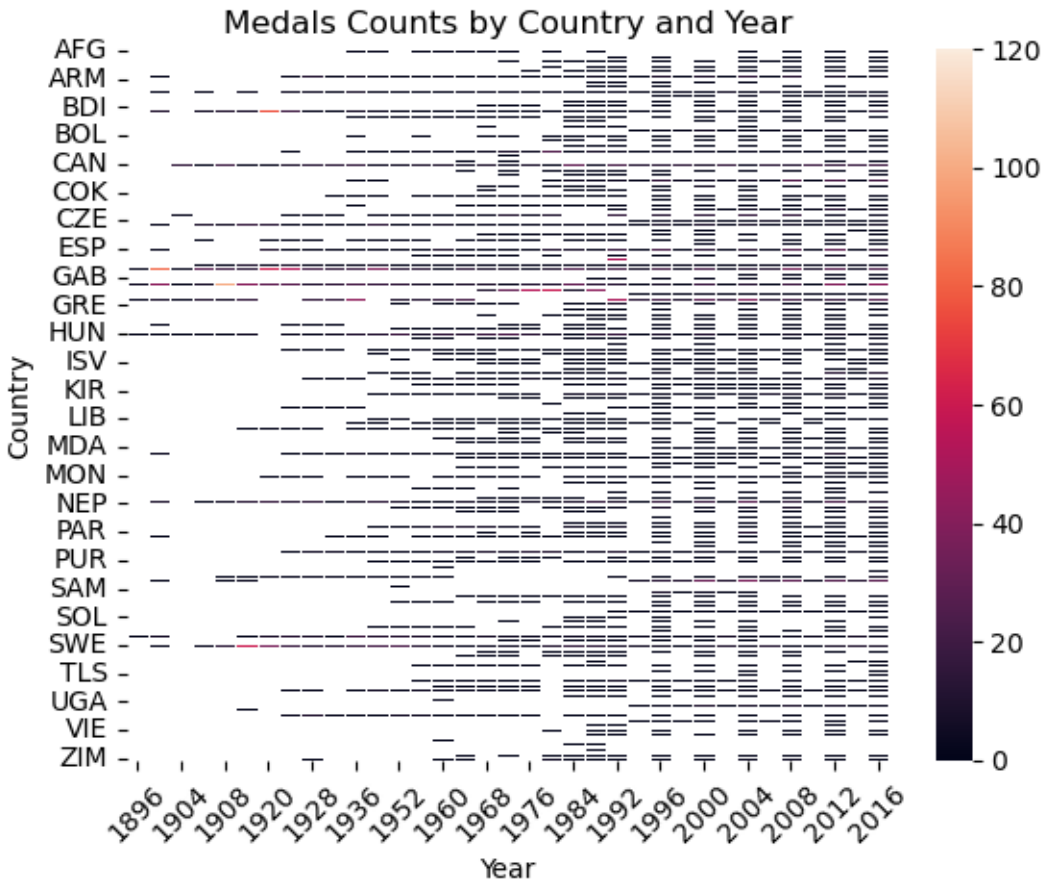
[47]:
```
heaviest_athelte = data[data["Weight"] == data["Weight"].max()]
print("Heaviest athelete: ")
print(heaviest_athelte[["ID", "Name", "Weight", "Sport"]])
```

```
Heaviest athelete:
          ID               Name  Weight Sport
23155  12177  Ricardo Blas, Jr.   214.0  Judo
23156  12177  Ricardo Blas, Jr.   214.0  Judo
```

[48]:
```
sns.scatterplot(data=data, x ="Height", y = "Weight", hue ="Medal", palette =␣
↪"Set1")
plt.title("Athelete Height Vs. Weight by Medal Status")
plt.xlabel("Height")
plt.ylabel("Weight")
plt.legend(title = "Medal")
plt.show()
```



[49]:
```
medals_by_country_year = data.pivot_table(index = "NOC", columns = "Year",␣
↪values = "Medal", aggfunc = "count")
sns.heatmap(medals_by_country_year, linewidths = 0.5)
plt.title("Medals Counts by Country and Year")
plt.xlabel("Year")
plt.ylabel("Country")
plt.xticks(rotation = 45)
plt.show()
```


Medals Counts by Country and Year
Country
AFG
ARM
BDI
BOL
CAN
COK
CZE
ESP
GAB
GRE
HUN
ISV
KIR
LIB
MDA
MON
NEP
PAR
PUR
SAM
SOL
SWE
TLS
UGA
VIE
ZIM
1896
1904
1908
1920
1928
1936
1952
1960
1968
1976
1984
1992
1996
2000
2004
2008
2012
2016
Year
120
100
80
60
40
20
0

[ ]:

[ ]: