Data Analysis for Olympic Game data.

In [182]:

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

In [183]:

```
data = pd.read_csv("athlete_events.csv")
data
```

Out[183]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Se
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Su
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Su
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Su
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Su
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	W
...	...	...	...	...	...	...	...	...	...	...	
271111	135569	Andrzej ya	M	29.0	179.0	89.0	Poland-1	POL	1976 Winter	1976	W
271112	135570	Piotr ya	M	27.0	176.0	59.0	Poland	POL	2014 Winter	2014	W
271113	135570	Piotr ya	M	27.0	176.0	59.0	Poland	POL	2014 Winter	2014	W
271114	135571	Tomasz Ireneusz ya	M	30.0	185.0	96.0	Poland	POL	1998 Winter	1998	W
271115	135571	Tomasz Ireneusz ya	M	34.0	185.0	96.0	Poland	POL	2002 Winter	2002	W

271116 rows × 15 columns

In [184]:

```
data.describe()
```

Out[184]:

	ID	Age	Height	Weight	Year
count	271116.000000	261642.000000	210945.000000	208241.000000	271116.000000
mean	68248.954396	25.556898	175.338970	70.702393	1978.378480
std	39022.286345	6.393561	10.518462	14.348020	29.877632
min	1.000000	10.000000	127.000000	25.000000	1896.000000
25%	34643.000000	21.000000	168.000000	60.000000	1960.000000
50%	68205.000000	24.000000	175.000000	70.000000	1988.000000
75%	102097.250000	28.000000	183.000000	79.000000	2002.000000
max	135571.000000	97.000000	226.000000	214.000000	2016.000000

In [185]:

```
data.shape
```

Out[185]:

```
(271116, 15)
```

1. First lets clean the dataframe; look for any null or "Nan" value.

In [186]:

```
#Look for null values
missing_data = data.isnull()
missing_data.head(10)
```

Out[186]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	True	True	False	False	False	False	False	False	False
3	False	False	False	False	True	True	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False	False	False	False	False

In [187]:

```
for column in missing_data:
    print(column)
    print(missing_data[column].value_counts())
    print("")
```

```
ID
False    271116
Name: ID, dtype: int64

Name
False    271116
Name: Name, dtype: int64

Sex
False    271116
Name: Sex, dtype: int64

Age
False    261642
True       9474
Name: Age, dtype: int64

Height
False    210945
True      60171
Name: Height, dtype: int64

Weight
False    208241
True      62875
Name: Weight, dtype: int64

Team
False    271116
Name: Team, dtype: int64

NOC
False    271116
Name: NOC, dtype: int64

Games
False    271116
Name: Games, dtype: int64

Year
False    271116
Name: Year, dtype: int64

Season
False    271116
Name: Season, dtype: int64

City
False    271116
Name: City, dtype: int64

Sport
False    271116
Name: Sport, dtype: int64

Event
False    271116
```

```
Name: Event, dtype: int64

Medal
True     231333
False     39783
Name: Medal, dtype: int64
```

The columns with missing data are "Age", "Weight", "Height", and "Medal". In this case we Find the mean value for each column to replace the "NaN" value, as the mean is the value that would most likely occur if we guessed it. For "Medal" we don't to find the mean as that would give inaccurate data on how many medals were really earned by contestants.

In [188]:

```
#Find mean of Age

avg_age = data["Age"].astype(float).mean(axis=0)
avg_age
```

Out[188]:

```
25.556898357297374
```

In [189]:

```
#Replace Nan in Age with the mean of age
data["Age"].replace(np.nan, avg_age, inplace=True)
```

In [190]:

```
#Find mean of Height
avg_height = data["Height"].astype(float).mean(axis=0)
avg_height
```

Out[190]:

```
175.33896987366376
```

In [191]:

```
data["Height"].replace(np.nan, avg_height, inplace=True)
```

In [192]:

```
#Find mean of weight
avg_weight = data["Weight"].astype(float).mean(axis=0)
avg_weight
```

Out[192]:

```
70.70239290053351
```

In [193]:

```
data["Weight"].replace(np.nan, avg_weight, inplace=True)
```

In [194]:

```
data
```

Out[194]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Yea
0	1	A Dijiang	M	24.0	180.00000	80.000000	China	CHN	1992 Summer	199
1	2	A Lamusi	M	23.0	170.00000	60.000000	China	CHN	2012 Summer	201
2	3	Gunnar Nielsen Aaby	M	24.0	175.33897	70.702393	Denmark	DEN	1920 Summer	192
3	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	190
4	5	Christine Jacoba Aaftink	F	21.0	185.00000	82.000000	Netherlands	NED	1988 Winter	198
...	...	...	...	...	...	...	...	...	...	.
271111	135569	Andrzej ya	M	29.0	179.00000	89.000000	Poland-1	POL	1976 Winter	197
271112	135570	Piotr ya	M	27.0	176.00000	59.000000	Poland	POL	2014 Winter	201
271113	135570	Piotr ya	M	27.0	176.00000	59.000000	Poland	POL	2014 Winter	201
271114	135571	Tomasz Ireneusz ya	M	30.0	185.00000	96.000000	Poland	POL	1998 Winter	199
271115	135571	Tomasz Ireneusz ya	M	34.0	185.00000	96.000000	Poland	POL	2002 Winter	200

271116 rows × 15 columns

In [195]:

```
#Now lets check if these corrects were made for the columns. Below shows all the columsn w
ith NaN data were cleaned except "M"
data.isnull().value_counts()
```

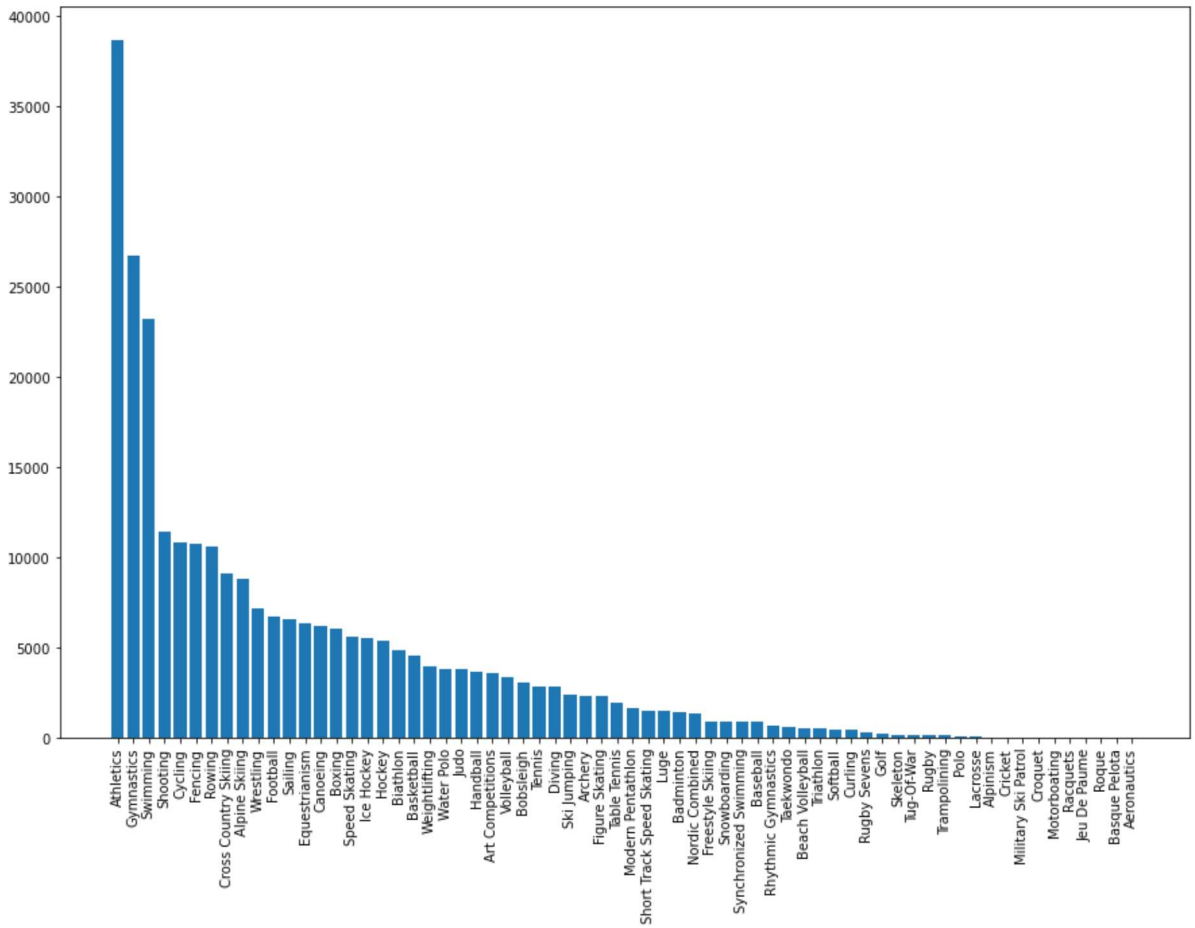
Out[195]:

```
ID     Name   Sex    Age    Height  Weight  Team   NOC    Games  Year   Seaso
n  City   Sport  Event  Medal
False  False  False  False  False   False   False  False  False  False  False
False  False  False  True     231333

False     39783
dtype: int64
```

How many persons competed in each sport event overall?

In [196]:

```
#How many persons competed in each Olympic sport; both medal winners and non-medal winner
s?

fit = plt.figure()

sport = data["Sport"].value_counts().to_dict()
x = sport.keys()
y = sport.values()
plt.xticks(rotation=90)
plt.bar(x, y)
plt.rcParams["figure.figsize"] = (15,10)
plt.show()
```



Now we see that we found the mean for Age, Weight, and Height, and calculated how many persons competed overall, let's fully clean the data by dropping all the rows with "NaN' for medal winners so we can have a full cleaned data set with just medal winners.

In [197]:

```
data.dropna(subset=["Medal"], axis=0, inplace=True)
data.reset_index(drop=True, inplace=True)
data
```

Out[197]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Y
0	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	1
1	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1
2	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1
3	16	Juhamatti Tapio Aaltonen	M	28.0	184.00000	85.000000	Finland	FIN	2014 Winter	2
4	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1
...	...	...	...	...	...	...	...	...	...	
39778	135553	Galina Ivanovna Zybina (-Fyodorova)	F	25.0	168.00000	80.000000	Soviet Union	URS	1956 Summer	1
39779	135553	Galina Ivanovna Zybina (-Fyodorova)	F	33.0	168.00000	80.000000	Soviet Union	URS	1964 Summer	1
39780	135554	Bogusaw Zych	M	28.0	182.00000	82.000000	Poland	POL	1980 Summer	1
39781	135563	Olesya Nikolayevna Zykina	F	19.0	171.00000	64.000000	Russia	RUS	2000 Summer	2
39782	135563	Olesya Nikolayevna Zykina	F	23.0	171.00000	64.000000	Russia	RUS	2004 Summer	2

39783 rows × 15 columns

2. Now we have a fully cleaned dataset, let's model the information below using barcharts, pie charts, and scatter plots.

In [198]:

```
#The dates are not in order, so let us rearrange them.
data.sort_values("Year", ascending=True, inplace=True)
data["Year"]
```

Out[198]:

```
3213     1896
35557    1896
31036    1896
31035    1896
31034    1896
         ...
36745    2016
3393     2016
3387     2016
27288    2016
34334    2016
Name: Year, Length: 39783, dtype: int64
```

1. Gold Medals for Gymnastics in "Age" range:

In [199]:

```
#We can do this by filtering the dataset to only gold medals and then plot it with the gym
nists

gold = data[data["Medal"] == "Gold"]
gold.head()
```

Out[199]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Yea
3213	12563	Conrad Helmut Fritz Bcker	M	25.000000	175.33897	70.702393	Germany	GER	1896 Summer	189
31036	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31035	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31034	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
28144	97712	Leonidas "Leon" Pyrgos	M	25.556898	175.33897	70.702393	Greece	GRE	1896 Summer	189

In [ ]:

In [200]:

```
#Now we filter the data set to just the gymnastic competitors
gymnastics = gold[gold["Sport"] == "Gymnastics"]
gymnastics.head()
```

Out[200]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Yea
3213	12563	Conrad Helmut Fritz Bcker	M	25.000000	175.33897	70.702393	Germany	GER	1896 Summer	189
31036	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31034	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
29630	102806	Richard Rstel	M	25.556898	175.33897	70.702393	Germany	GER	1896 Summer	189
29631	102806	Richard Rstel	M	25.556898	175.33897	70.702393	Germany	GER	1896 Summer	189

In [201]:

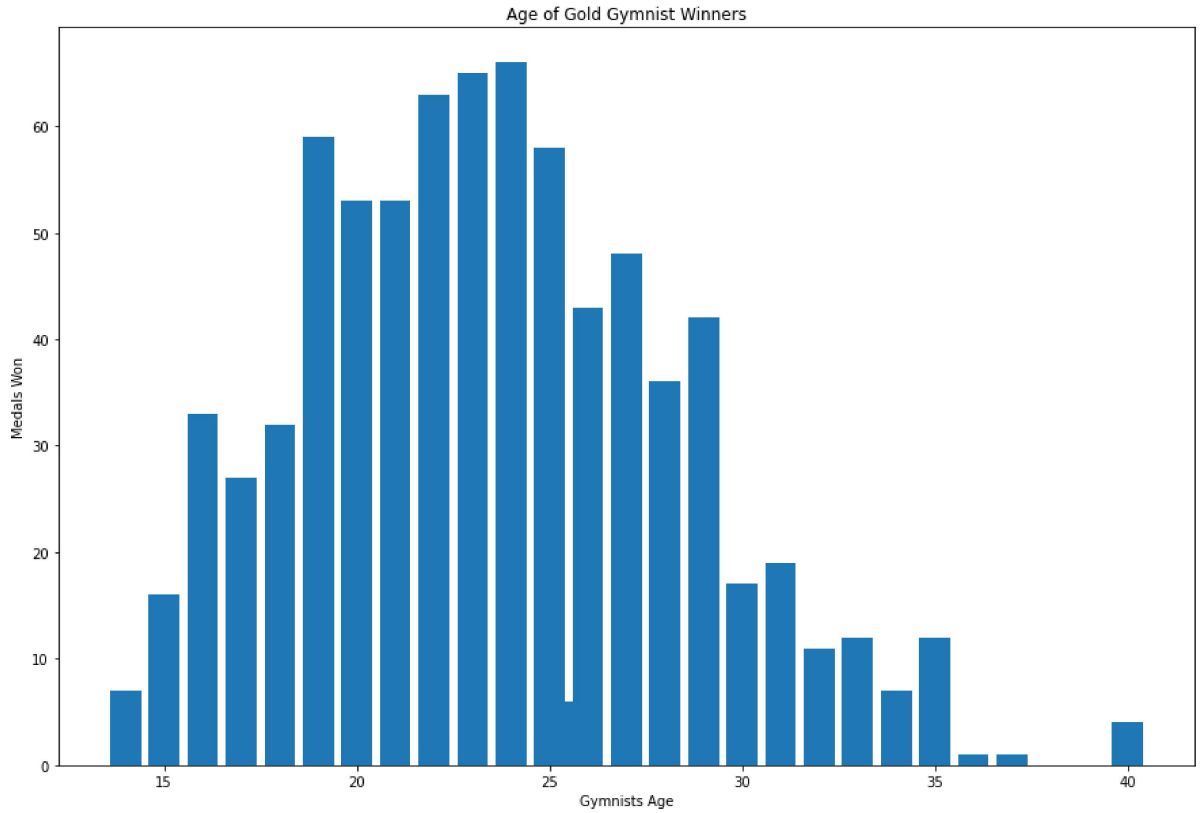
```
#Now that we have a dataset with only gymnists, we can count the different age ranges.
z = gymnastics["Age"].value_counts().to_dict()
z
```

Out[201]:

```
{24.0: 66,
 23.0: 65,
 22.0: 63,
 19.0: 59,
 25.0: 58,
 20.0: 53,
 21.0: 53,
 27.0: 48,
 26.0: 43,
 29.0: 42,
 28.0: 36,
 16.0: 33,
 18.0: 32,
 17.0: 27,
 31.0: 19,
 30.0: 17,
 15.0: 16,
 35.0: 12,
 33.0: 12,
 32.0: 11,
 34.0: 7,
 14.0: 7,
 25.556898357297374: 6,
 40.0: 4,
 37.0: 1,
 36.0: 1}
```

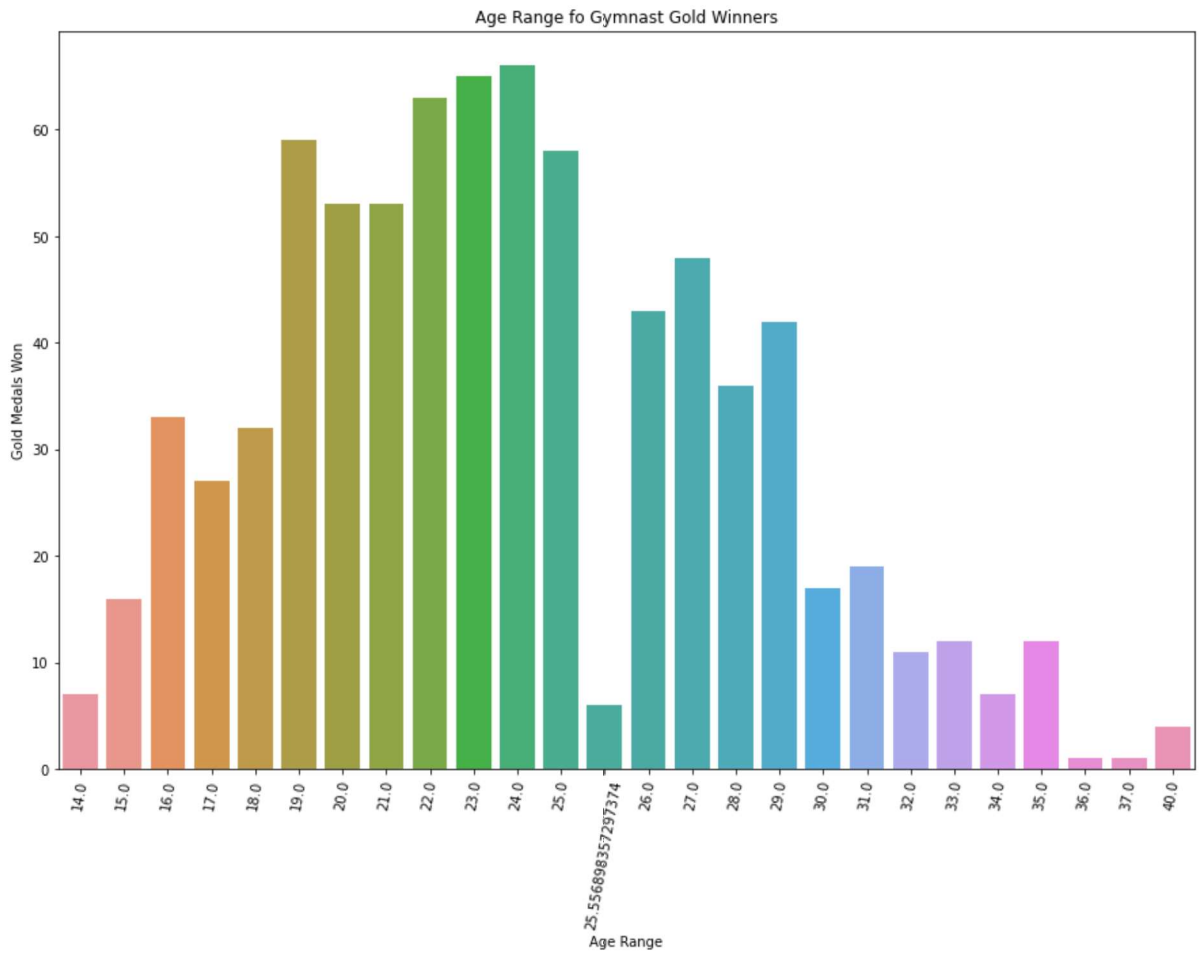
In [202]:

```
#Now that we have all the different age ranges and their values, we can plot it.

a = z.keys()
b = z.values()

plt.bar(a, b)
plt.title("Age of Gold Gymnist Winners")
plt.xlabel("Gymnists Age")
plt.ylabel("Medals Won")
plt.show()
```



In [203]:

```
#We can use the sns from the seaborn library as well
import seaborn as sns

sns_a = (list(z.keys()))
sns_b = (list(z.values()))

gymnast_age = sns.barplot(sns_a, sns_b, data=gymnastics)
gymnast_age.set(xlabel="Age Range", ylabel="Gold Medals Won", title="Age Range fo Gymnast
 Gold Winners")
gymnast_age.set_xticklabels(gymnast_age.get_xticklabels(), rotation="80")
plt.show()
```



1. How many medals were won by China over the years:

In [204]:

```
#We go to the cleaned "data" data frame
data
```

Out[204]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Yea
3213	12563	Conrad Helmut Fritz Bcker	M	25.000000	175.33897	70.702393	Germany	GER	1896 Summer	189
35557	122329	Georgios Tsitas	M	25.556898	175.33897	70.702393	Greece	GRE	1896 Summer	189
31036	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31035	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31034	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
...	...	...	...	...	...	...	...	...	...	..
36745	126330	Giuseppe Vicino	M	23.000000	195.00000	95.000000	Italy	ITA	2016 Summer	201
3393	13131	Bohdan Viktorovych Bondarenko	M	26.000000	198.00000	77.000000	Ukraine	UKR	2016 Summer	201
3387	13120	Hamish Byron Bond	M	30.000000	189.00000	89.000000	New Zealand	NZL	2016 Summer	201
27288	94685	Ccile Pieper	F	21.000000	166.00000	56.000000	Germany	GER	2016 Summer	201
34334	118426	Tang Mengni	F	22.000000	169.00000	59.000000	China	CHN	2016 Summer	201

39783 rows × 15 columns

In [205]:

```
#Now we filter the data set to just "China"
China = data[data["Team"] == "China"]
China.head()
```

Out[205]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	Cit
20106	69630	Liang Yan	F	22.0	177.0	68.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
20093	69555	Li Yanjun	F	21.0	179.0	73.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
20100	69579	Li Yuwei	M	19.0	167.0	63.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
20097	69572	Li Yuejiu	M	26.0	154.0	60.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
5395	20319	Chen Yongyan	F	21.0	158.0	44.0	China	CHN	1984 Summer	1984	Summer	Lo Angele

In [206]:

```
#Now we have a just "China" as the country team in this data, lets count how many bronze,
 silver,a nd gold medals were won each year.

bronze_medal = China[China["Medal"] == "Bronze"]
bronze_score = bronze_medal["Year"].value_counts().to_dict()

#Use "sorted" because we want the years and their values to tbe groups in descending order
from the lower earliest year to lastest.
bronze_score
```

Out[206]:

```
{2008: 49,
 1984: 37,
 2016: 35,
 1988: 30,
 2012: 29,
 2000: 15,
 1996: 15,
 1992: 15,
 2004: 13,
 2010: 7,
 2002: 7,
 2014: 5,
 1998: 5,
 2006: 4,
 1994: 2}
```

In [207]:

```
#Find silver medal winners

silver_medal = China[China["Medal"] == "Silver"]
silver_score = silver_medal["Year"].value_counts().to_dict()
silver_score
```

Out[207]:

```
{1996: 66,
 2008: 49,
 1992: 44,
 2012: 38,
 2016: 30,
 2004: 27,
 2000: 19,
 1988: 16,
 1984: 13,
 1998: 9,
 2002: 5,
 2014: 4,
 2006: 3,
 2010: 1,
 1994: 1}
```

In [208]:

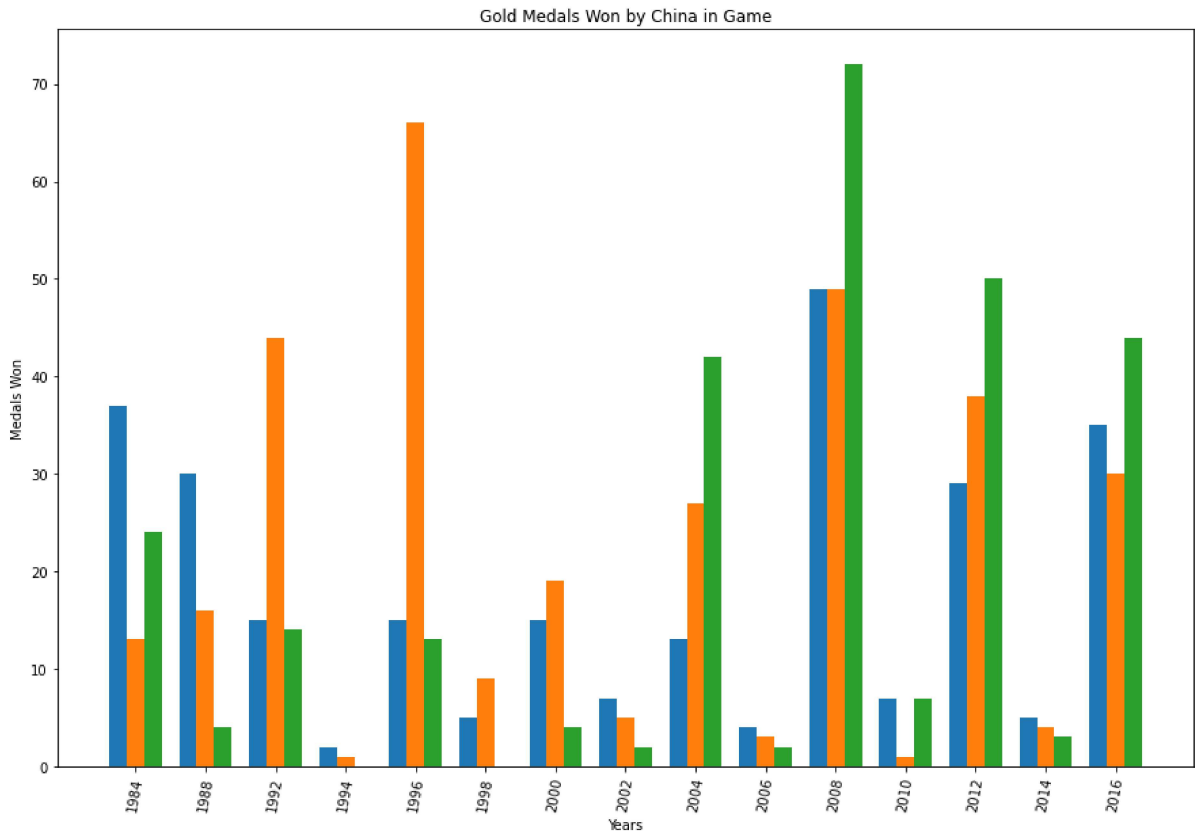
```
#Find gold medals.

gold_medal = China[China["Medal"] == "Gold"]
gold_score = gold_medal["Year"].value_counts().to_dict()
gold_score
```

Out[208]:

```
{2008: 72,
 2012: 50,
 2016: 44,
 2004: 42,
 2000: 31,
 1984: 24,
 1992: 14,
 1996: 13,
 2010: 7,
 1988: 4,
 2014: 3,
 2006: 2,
 2002: 2}
```

In [209]:

```
#Now we know how many bronze, silver, and gold medals won by China over the yeas, lets plo
t the information.

years = [1984, 1988, 1992, 1994, 1996, 1998, 2000, 2002, 2004, 2006, 2008, 2010, 2012, 201
4, 2016]

x_indexes = np.arange(len(years))
width = 0.25

bronze_y = [37, 30, 15, 2, 15, 5, 15, 7, 13, 4, 49, 7, 29, 5, 35]
plt.bar(x_indexes - width, bronze_y, width=width, label="Bronze", data=China)

silver_y = [13, 16, 44, 1, 66, 9, 19, 5, 27, 3, 49, 1, 38, 4, 30]
plt.bar(x_indexes, silver_y, width=width, label="Silver", data=China)

gold_y = [24, 4, 14, 0, 13, 0, 4, 2, 42, 2, 72, 7, 50, 3, 44]
plt.bar(x_indexes + width, gold_y, width=width, label="Gold")

plt.xticks(x_indexes,years, rotation=80)
plt.xlabel("Years")
plt.ylabel("Medals Won")
plt.title("Gold Medals Won by China in Game")
plt.show()
```



1. Number of medals won by China in Summer Games over the years:

In [210]:

```
#For this we go to the "China" data set because it only has medal winnrs from China, and t
hen filter it by the "Summer" games.
summer_season = China[China["Season"] == "Summer"]
summer_season.head()
```

Out[210]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	Cit
20106	69630	Liang Yan	F	22.0	177.0	68.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
20093	69555	Li Yanjun	F	21.0	179.0	73.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
20100	69579	Li Yuwei	M	19.0	167.0	63.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
20097	69572	Li Yuejiu	M	26.0	154.0	60.0	China	CHN	1984 Summer	1984	Summer	Lo Angele
5395	20319	Chen Yongyan	F	21.0	158.0	44.0	China	CHN	1984 Summer	1984	Summer	Lo Angele

In [211]:

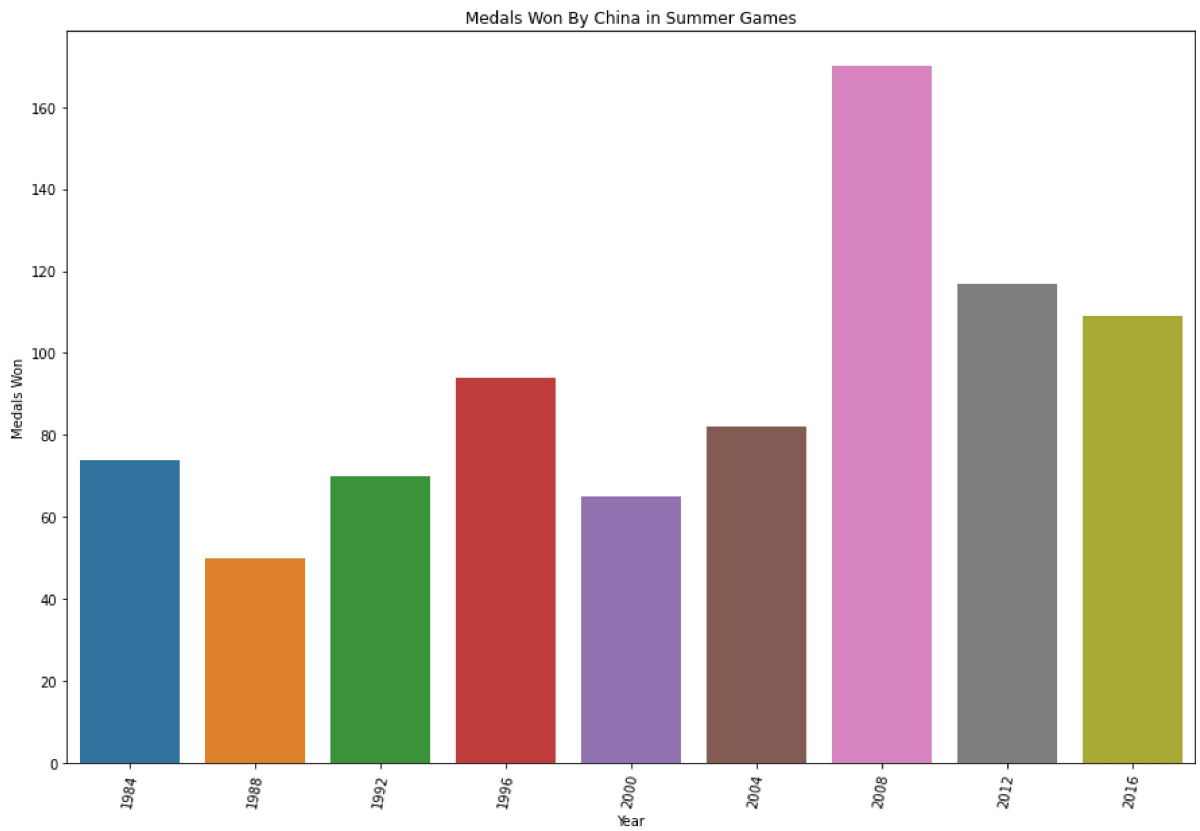
```
s = summer_season["Year"].value_counts().to_dict()
s
```

Out[211]:

```
{2008: 170,
 2012: 117,
 2016: 109,
 1996: 94,
 2004: 82,
 1984: 74,
 1992: 70,
 2000: 65,
 1988: 50}
```

In [212]:

```
sum_x = (list(s.keys()))
sum_y = (list(s.values()))

China_sum = sns.barplot(sum_x, sum_y, data=summer_season)
China_sum.set(xlabel="Year", ylabel="Medals Won", title="Medals Won By China in Summer Gam
es")
China_sum.set_xticklabels(China_sum.get_xticklabels(), rotation=80)
plt.show()
```



In [213]:

```
#Notice in the above that in "sns" the dates and their values were automatically arranged
 earliest to lastest year.
```

1. Top five countried with the most medals

In [214]:

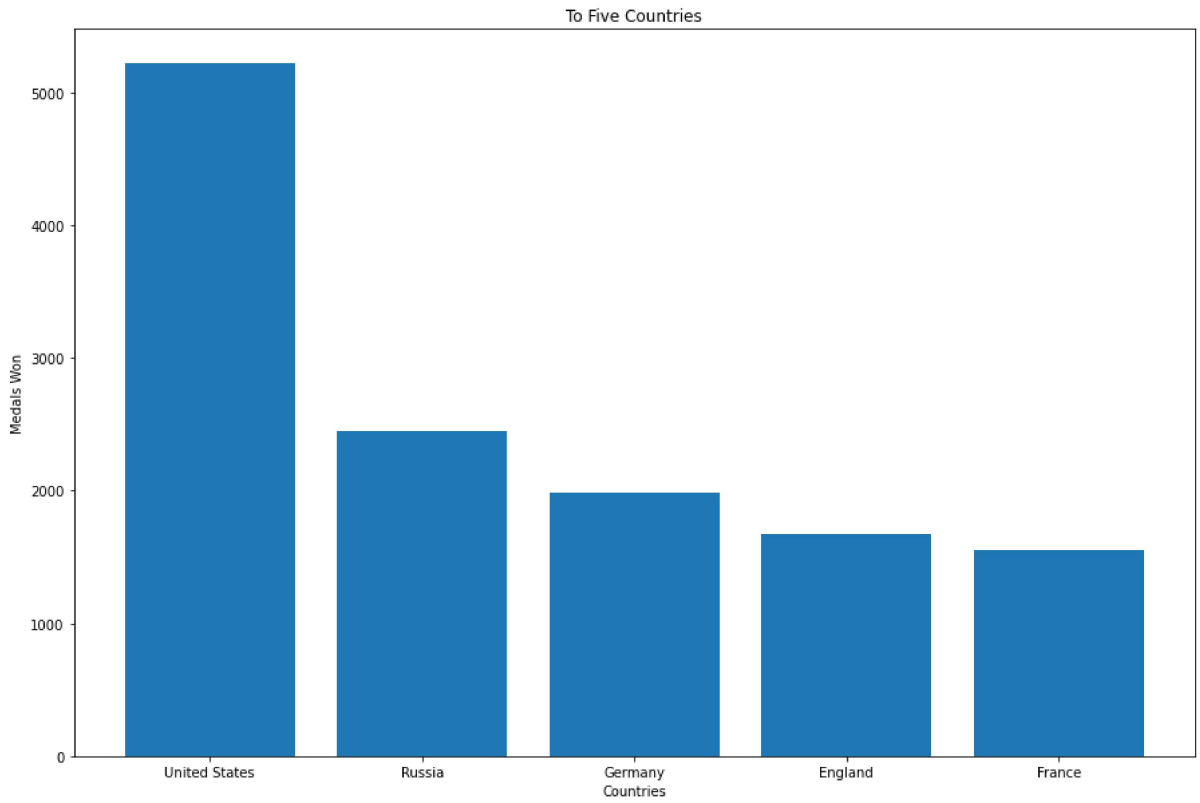
```
#For this we go back to the "data" dataset; it has only medal winners, so any country that
appears the most automatically has won a medal.

#This method counts the instances(number of times) a country appears from highest to lowes
t.
top_five = data["Team"].value_counts()

#Now we only want the top five.
top_five.head()
```

Out[214]:

```
United States    5219
Soviet Union     2451
Germany          1984
Great Britain    1673
France           1550
Name: Team, dtype: int64
```

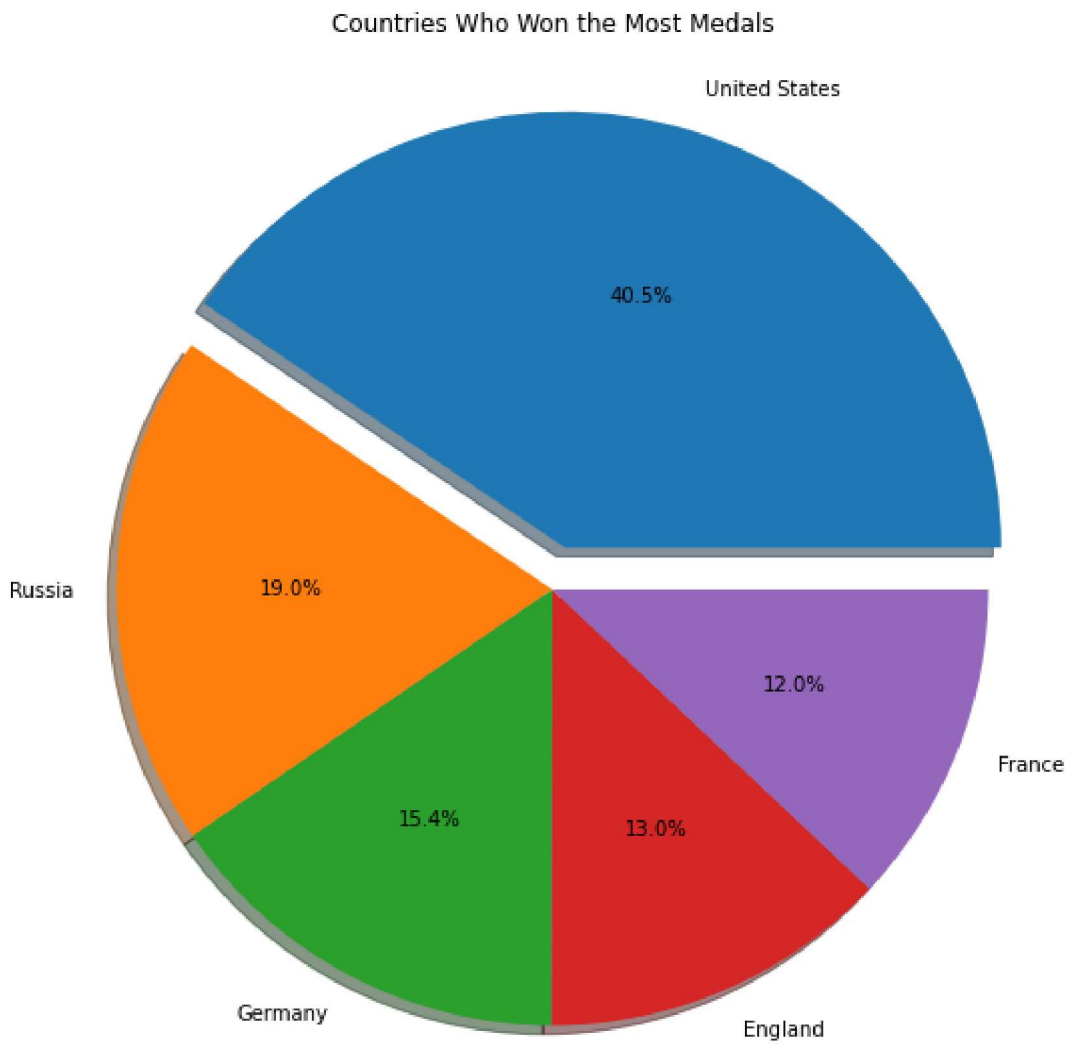
In [215]:

```
#Now we have the top five countries, we can simply plot the values in.

x = ["United States", "Russia", "Germany", "England", "France"]
y = [5219, 2451, 1984, 1673, 1550]

plt.bar(x, y)
plt.xlabel("Countries")
plt.ylabel("Medals Won")
plt.title("To Five Countries")
plt.show()
```


To Five Countries
Medals Won
5000
4000
3000
2000
1000
0
United States
Russia
Germany
England
France
Countries

In [216]:

```
#We can also plot the abobe information as a "pie chart."
labels = ["United States", "Russia", "Germany", "England", "France"]

slices = [5219, 2451, 1984, 1673, 1550]

explode = [0.1,0,0,0,0]

plt.pie(slices, labels=labels, explode=explode, autopct="%1.1f%%", shadow=True)
plt.title("Countries Who Won the Most Medals")
plt.show()
```



1. The height of male players over the years:

In [217]:

```
#For this we got to the "data" data set for this information and filter it to just the mal
e compepitors.
male = data[data["Sex"] == "M"]
male
```

Out[217]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Yea
3213	12563	Conrad Helmut Fritz Bcker	M	25.000000	175.33897	70.702393	Germany	GER	1896 Summer	189
35557	122329	Georgios Tsitas	M	25.556898	175.33897	70.702393	Greece	GRE	1896 Summer	189
31036	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31035	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
31034	107613	Carl Schuhmann	M	26.000000	159.00000	70.000000	Germany	GER	1896 Summer	189
...	...	...	...	...	...	...	...	...	...	..
36717	126248	Luca Vettori	M	25.000000	200.00000	95.000000	Italy	ITA	2016 Summer	201
10353	37688	Moritz Frste	M	31.000000	190.00000	89.000000	Germany	GER	2016 Summer	201
36745	126330	Giuseppe Vicino	M	23.000000	195.00000	95.000000	Italy	ITA	2016 Summer	201
3393	13131	Bohdan Viktorovych Bondarenko	M	26.000000	198.00000	77.000000	Ukraine	UKR	2016 Summer	201
3387	13120	Hamish Byron Bond	M	30.000000	189.00000	89.000000	New Zealand	NZL	2016 Summer	201

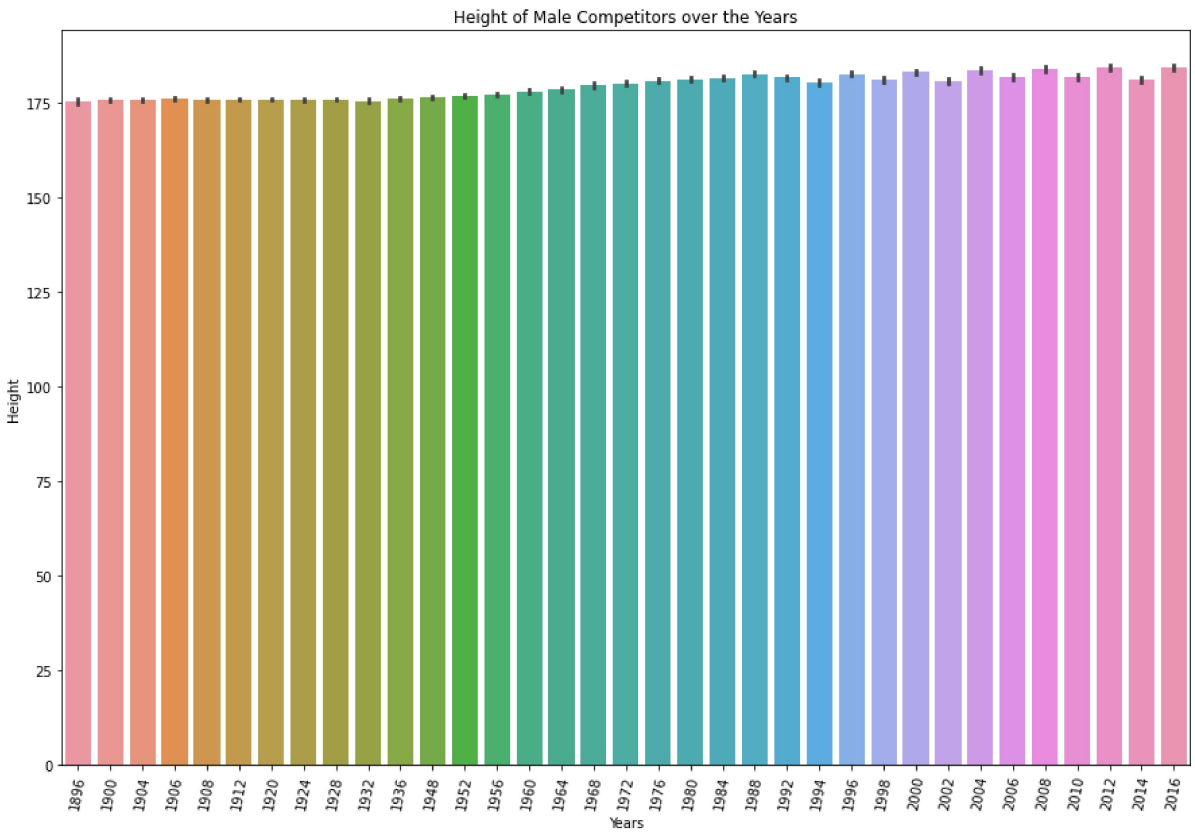
28530 rows × 15 columns

In [218]:

```
height = sns.barplot(male.Year, male.Height, data=male)
height.set(xlabel="Years", ylabel="Height", title="Height of Male Competitors over the Yea
rs")
height.set_xticklabels(height.get_xticklabels(), rotation=80)
height
```

Out[218]:

```
<AxesSubplot:title={'center':'Height of Male Competitors over the Years'}, xl
abel='Years', ylabel='Height'>
```



Fin