In [1]:

```
%matplotlib inline
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:

```
file = "Athlete_events..csv"

data = pd.read_csv(file)

data
```

Out[2]:

	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 [3]:

```
data.columns
```

Out[3]:

```
Index(['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight', 'Team', 'NOC', 'Game
s',
       'Year', 'Season', 'City', 'Sport', 'Event', 'Medal'],
      dtype='object')
```

In [4]:

```
data.describe()
```

Out[4]:

	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

1.Find out if there are any null values

In [5]:

```
missing_data = data.isnull()

missing_data
```

Out[5]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City
0	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
2	False	False	False	False	True	True	False	False	False	False	False	False
3	False	False	False	False	True	True	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
271111	False	False	False	False	False	False	False	False	False	False	False	False
271112	False	False	False	False	False	False	False	False	False	False	False	False
271113	False	False	False	False	False	False	False	False	False	False	False	False
271114	False	False	False	False	False	False	False	False	False	False	False	False
271115	False	False	False	False	False	False	False	False	False	False	False	False

271116 rows × 15 columns

In [6]:

```
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
```

Columns with missing data are: "Age" "Height" "Weight" "Medal"

Now we calulate the mean of each columns to replace the "NaN" value with, as the mean would be the most likely value to occur if we guessed it. For the medal value we don't calculate the mean as it is "object" class and not a "int" or "float".

In [7]:

```
#Find mean for "Age"
```

In [8]:

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

avg_age
```

Out[8]:

```
25.556898357297374
```

In [9]:

```
data["Age"].replace(np.nan, avg_age, inplace=True)
```

In [10]:

```
#Find mean for "Height"
```

In [11]:

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

avg_height
```

Out[11]:

```
175.33896987366376
```

In [12]:

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

In [13]:

```
#Find mean for "Weight"
```

In [14]:

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

avg_weight
```

Out[14]:

```
70.70239290053351
```

In [15]:

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

In [16]:

```
#Lets check now if we have "NaN" values:
```

In [17]:

```
data.isnull()
```

Out[17]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City
0	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
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
271111	False	False	False	False	False	False	False	False	False	False	False	False
271112	False	False	False	False	False	False	False	False	False	False	False	False
271113	False	False	False	False	False	False	False	False	False	False	False	False
271114	False	False	False	False	False	False	False	False	False	False	False	False
271115	False	False	False	False	False	False	False	False	False	False	False	False

271116 rows × 15 columns

2.Show us all of the people who won medals. We cannot drop the "Medal" column even though it has more "NaN" values because we still need information on the medal winnbers. So, we need to remove just the rows of people who won no medals.

In [18]:

```
data.head()
```

Out[18]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Seasoı
0	1	A Dijiang	M	24.0	180.00000	80.000000	China	CHN	1992 Summer	1992	Summe
1	2	A Lamusi	M	23.0	170.00000	60.000000	China	CHN	2012 Summer	2012	Summe
2	3	Gunnar Nielsen Aaby	M	24.0	175.33897	70.702393	Denmark	DEN	1920 Summer	1920	Summe
3	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	1900	Summe
4	5	Christine Jacoba Aaftink	F	21.0	185.00000	82.000000	Netherlands	NED	1988 Winter	1988	Winte

In [19]:

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

data.reset_index(drop=True, inplace=True)

data
```

Out[19]:

	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

In [20]:

```
#Now we have a data set with just people with medals won with their ages, weight and heigh
t specified.
```

Now we will visualize the information we need using graphs, bar charts, pie charts etc.

3.Plotting Data

In [21]:

```
gold = data[data["Medal"] == "Gold"]

gold
```

Out[21]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Ye
0	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	19
5	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	19
6	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	19
7	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	19
9	20	Kjetil Andr Aamodt	M	20.0	176.00000	85.000000	Norway	NOR	1992 Winter	19
...	...	...	...	...	...	...	...	...	...	
39764	135503	Zurab Zviadauri	M	23.0	182.00000	90.000000	Georgia	GEO	2004 Summer	20
39767	135520	Julia Zwehl	F	28.0	167.00000	60.000000	Germany	GER	2004 Summer	20
39771	135523	Ronald Ferdinand "Ron" Zwerver	M	29.0	200.00000	93.000000	Netherlands	NED	1996 Summer	19
39776	135545	Henk Jan Zwolle	M	31.0	197.00000	93.000000	Netherlands	NED	1996 Summer	19
39777	135553	Galina Ivanovna Zybina (-Fyodorova)	F	21.0	168.00000	80.000000	Soviet Union	URS	1952 Summer	19

13372 rows × 15 columns

1. Golds Medals for Gymnastic in "Age" range

In [22]:

```
fig = plt.figure()

gymnastics = gold[gold["Sport"] == "Gymnastics"]

gymnastics.head(5)
```

Out[22]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season
5	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1948	Summer
6	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1948	Summer
7	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1948	Summer
106	521	Isak Abrahamsen	M	21.0	175.33897	70.702393	Norway	NOR	1912 Summer	1912	Summer
134	697	Fausto Alesio Acke (Padovini-)	M	22.0	175.33897	70.702393	Sweden	SWE	1920 Summer	1920	Summer

```
<Figure size 432x288 with 0 Axes>
```

In [23]:

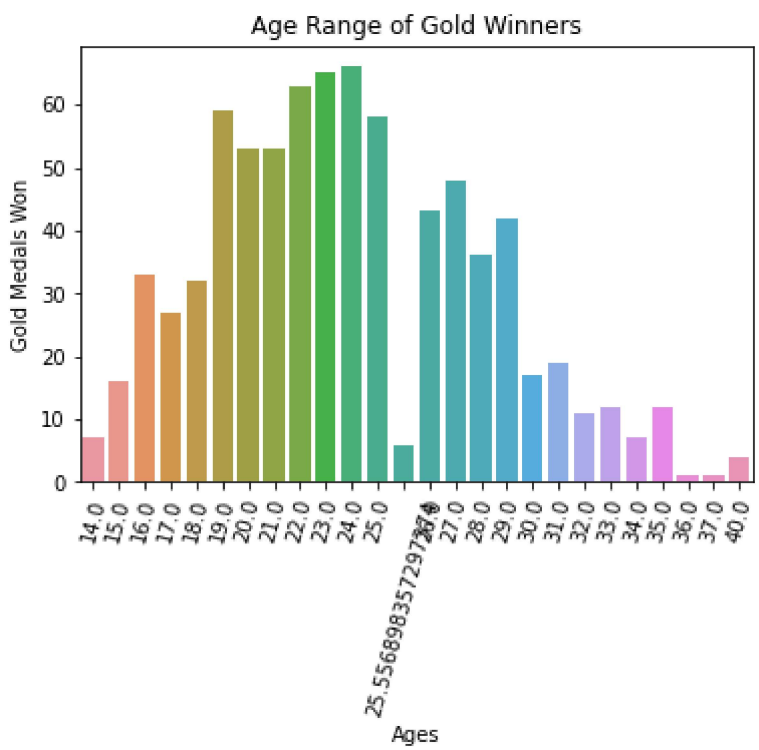
```
#Now that we filtered the data with just the gymnastic gold medalists,
#let's see what the age group of who wins the most gold medals.
#Since all medals in the gymnastics data set are filtered by gold, all ages groups automat
ically have won gold.

g = gymnastics["Age"].value_counts().to_dict()
g
```

Out[23]:

```
{24.0: 66,
 23.0: 65,
 22.0: 63,
 19.0: 59,
 25.0: 58,
 21.0: 53,
 20.0: 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,
 36.0: 1,
 37.0: 1}
```

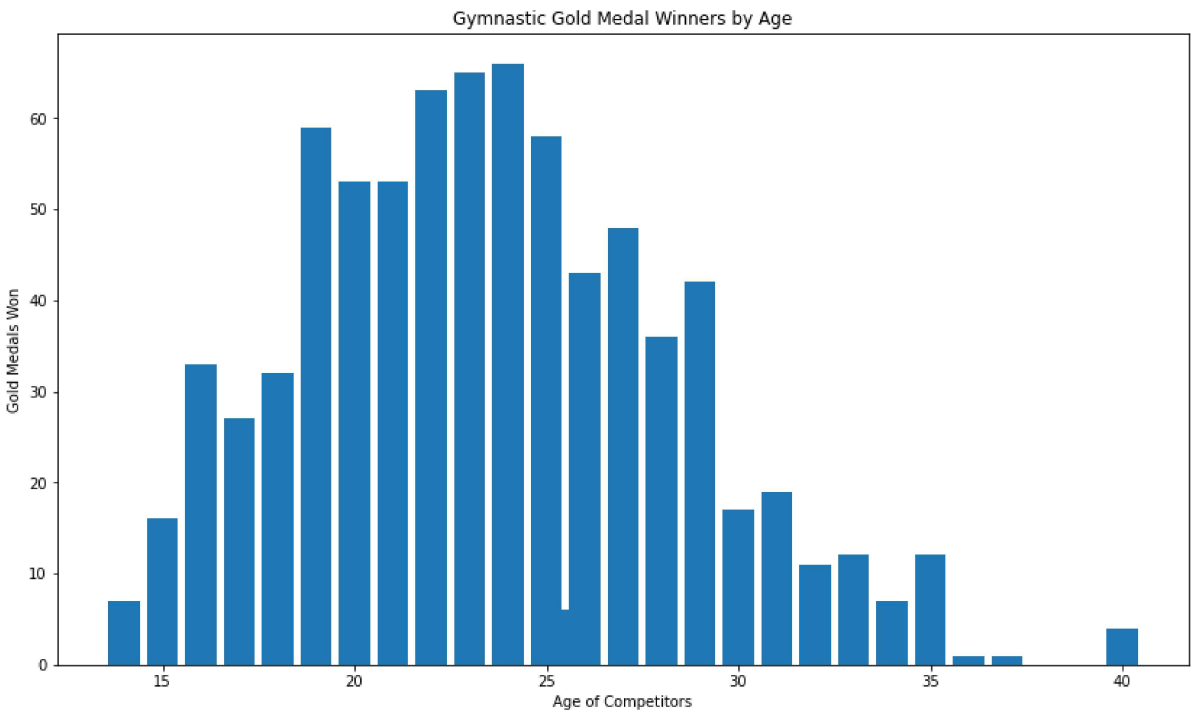
In [24]:

```
#This method shows how many times that age group appeared.
#And since the ages above are all gold medal winners, that tells how many times that age g
roup won medals.

# Now we can just plot the group and the rate. In this case we'll use "sns" from the "seab
orn" library

#This turns the key values of "g" dictionary (the ages) into lists.
age_x = (list(g.keys()))

age_x

#This turns the value values of "g" dictionary (the rates of occurence) into lists.
rate_y = (list(g.values()))

rate_y


age_graph = sns.barplot(age_x, rate_y, data=gymnastics)
age_graph.set(xlabel="Ages", ylabel="Gold Medals Won", title="Age Range of Gold Winners")
age_graph.set_xticklabels(age_graph.get_xticklabels(), rotation=75)
plt.rcParams["figure.figsize"] = (14,8)
plt.show()
```



In [25]:

```
#We can plot the same graph using "plt.bar" from the "matplotlib" library.

a = gymnastics["Age"].value_counts().to_dict()

key = a.keys()

value = a.values()

x = key
y = value

plt.bar(x, y,)
plt.title("Gymnastic Gold Medal Winners by Age")
plt.xlabel("Age of Competitors")
plt.rcParams["figure.figsize"] = (15,8)
plt.ylabel("Gold Medals Won")
plt.show()
```



1. Medals won by China over the years

In [36]:

```
#To do this, we go to the original cleaned "data" data frame, but filter it by the "Team"
 China

China = data[data["Team"] == "China"]

China.head()
```

Out[36]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City
701	3610	An Yulong	M	19.0	173.0	70.0	China	CHN	1998 Winter	1998	Winter	Nagano
702	3610	An Yulong	M	19.0	173.0	70.0	China	CHN	1998 Winter	1998	Winter	Nagano
703	3610	An Yulong	M	23.0	173.0	70.0	China	CHN	2002 Winter	2002	Winter	Salt Lake City
704	3611	An Zhongxin	F	23.0	170.0	65.0	China	CHN	1996 Summer	1996	Summer	Atlanta
1452	6381	Ba Yan	F	21.0	183.0	78.0	China	CHN	1984 Summer	1984	Summer	Los Angeles

In [37]:

```
#Now we know this column is just for China, all medals are automatically won by China for
 the data set "China",
#But now we have to see how many medals are won each years.

#Find Bronze Medals earned.
bronze = China[China["Medal"] == "Bronze"]

#Now we have the data for the Bronze medals earned throught the years for China.
bronze_value = bronze["Year"].value_counts().to_dict()

bronze_value
```

Out[37]:

```
{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 [38]:

```
#Silver Medals

silver = China[China["Medal"] == "Silver"]

silver_value = silver["Year"].value_counts().to_dict()

silver_value
```

Out[38]:

```
{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 [39]:

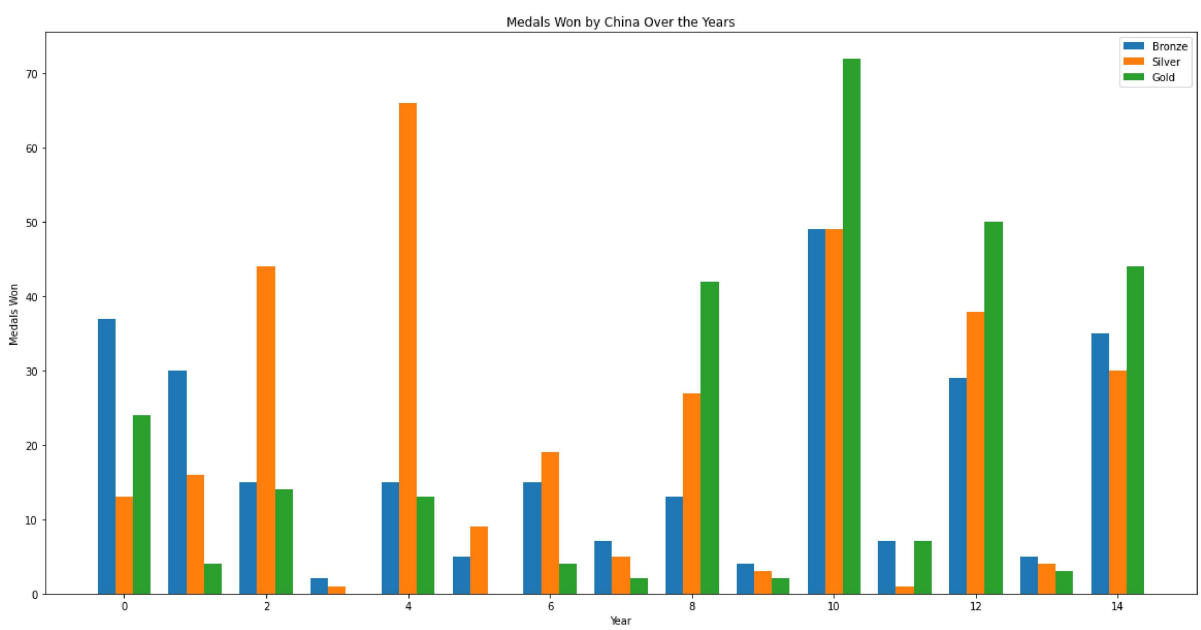
```
#Gold Medals
gold = China[China["Medal"] == "Gold"]

gold_value = gold["Year"].value_counts().to_dict()

gold_value
```

Out[39]:

```
{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 [40]:

```
#Now that we have the "Years" where bronze, silver, and gold were won for each year, lets
 group the values and plot them.

# We will group the rates for bronze, silver and gold to correspond with each year.

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")

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")

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.title("Medals Won by China Over the Years")
plt.xlabel("Year")
plt.ylabel("Medals Won")
plt.rcParams["figure.figsize"] = (15,8)
plt.legend()
plt.show()
```



1. Number of medals won by China in Summer Games

In [41]:

```
#We call the "gold" data set which alread has the Chinese competitors that have won only g
old, and specify from it the "Summer" season.
gold.head()
```

Out[41]:

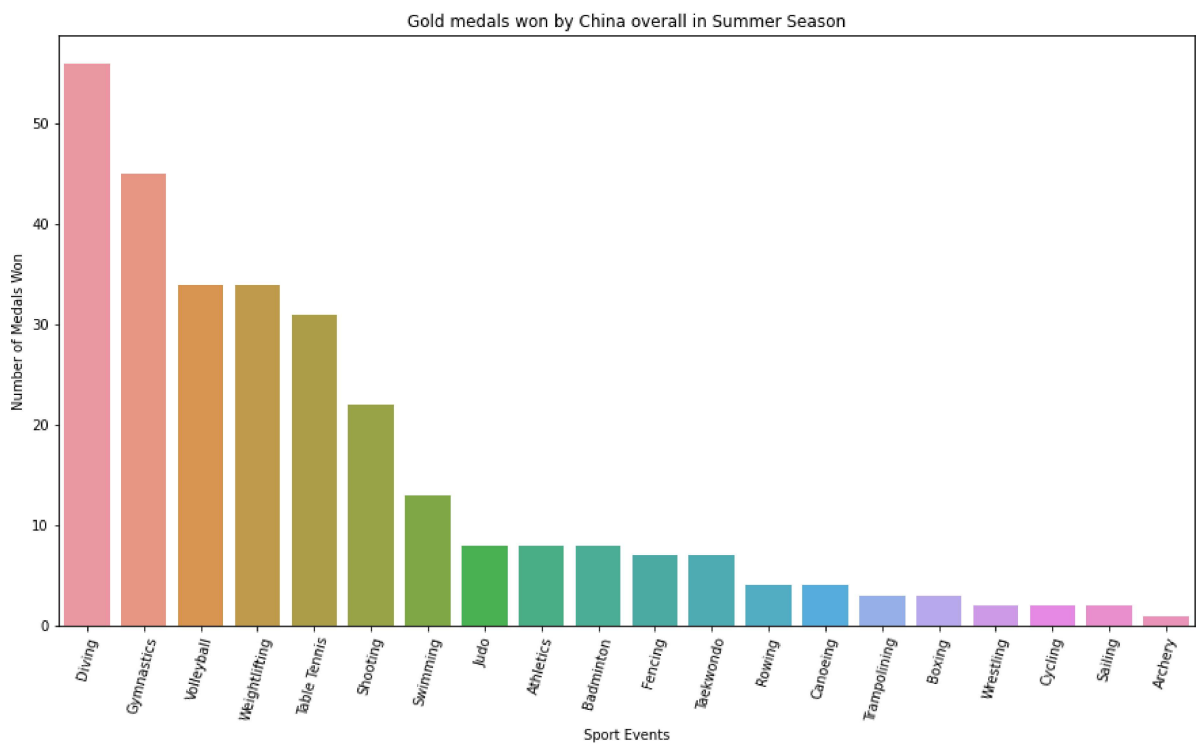
	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	
4640	17294	Cai Yalin	M	23.0	174.0	60.0	China	CHN	2000 Summer	2000	Summer	Sydney	
4824	17995	Cao Lei	F	24.0	168.0	75.0	China	CHN	2008 Summer	2008	Summer	Beijing	V
4826	18005	Cao Yuan	M	17.0	160.0	42.0	China	CHN	2012 Summer	2012	Summer	London	
4827	18005	Cao Yuan	M	21.0	160.0	42.0	China	CHN	2016 Summer	2016	Summer	Rio de Janeiro	
5351	20150	Chen Aisen	M	20.0	168.0	60.0	China	CHN	2016 Summer	2016	Summer	Rio de Janeiro	

In [42]:

```
summer = gold[gold["Season"] == "Summer"]

sum_sport = summer["Sport"].value_counts().to_dict()

summer_x = (list(sum_sport.keys()))

summer_x

summer_y = (list(sum_sport.values()))

summer_y

summer_games = sns.barplot(summer_x, summer_y, data=summer)
summer_games.set(xlabel="Sport Events", ylabel="Number of Medals Won", title=("Gold medals
won by China overall in Summer Season"))
summer_games.set_xticklabels(summer_games.get_xticklabels(), rotation=75)
summer_games
```

Out[42]:

```
<AxesSubplot:title={'center':'Gold medals won by China overall in Summer Seas
on'}, xlabel='Sport Events', ylabel='Number of Medals Won'>
```



1. Height of male atheles over the years

In [43]:

```
data.head()
```

Out[43]:

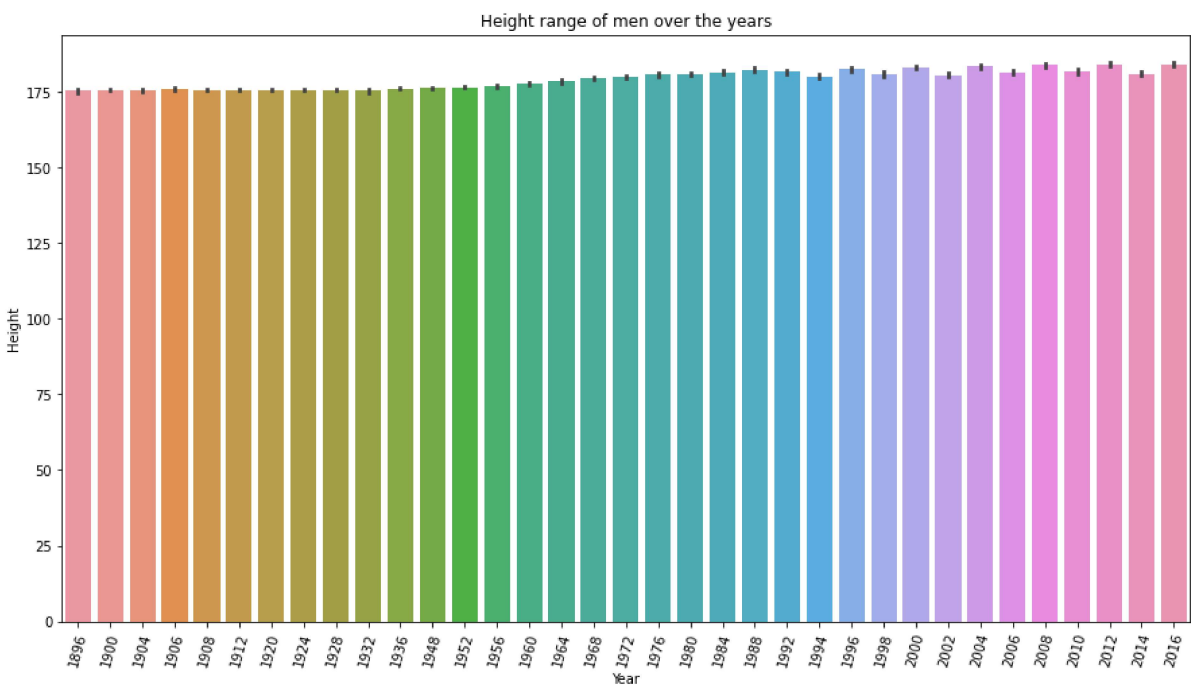
	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Seasc
0	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	1900	Summ
1	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1920	Summ
2	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1920	Summ
3	16	Juhamatti Tapio Aaltonen	M	28.0	184.00000	85.000000	Finland	FIN	2014 Winter	2014	Wint
4	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1948	Summ

In [44]:

```
#For this we go to the original filtered "data" set

male = data[data["Sex"] == "M"]

#Now we have a data set filtered by the male sex
male

#Now that we know the only gender in "male" are men, we can plot their height over the yea
rs.

male_height = sns.barplot("Year", "Height", data=male)
male_height.set(title="Height range of men over the years")
plt.rcParams["figure.figsize"] = (20,10)
male_height.set_xticklabels(male_height.get_xticklabels(), rotation=75)
male_height
```

Out[44]:

```
<AxesSubplot:title={'center':'Height range of men over the years'}, xlabel='Y
ear', ylabel='Height'>
```



1. Top five countries with the most medals

In [45]:

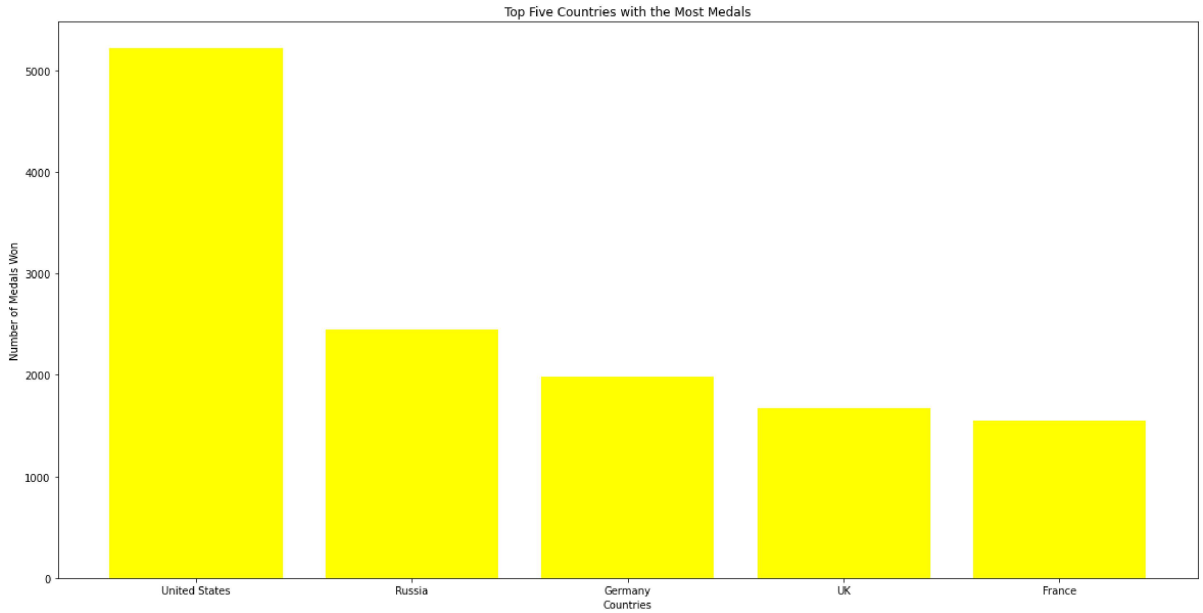
```
data.head(5)
```

Out[45]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Seasc
0	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	1900	Summ
1	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1920	Summ
2	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1920	Summ
3	16	Juhamatti Tapio Aaltonen	M	28.0	184.00000	85.000000	Finland	FIN	2014 Winter	2014	Wint
4	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1948	Summ

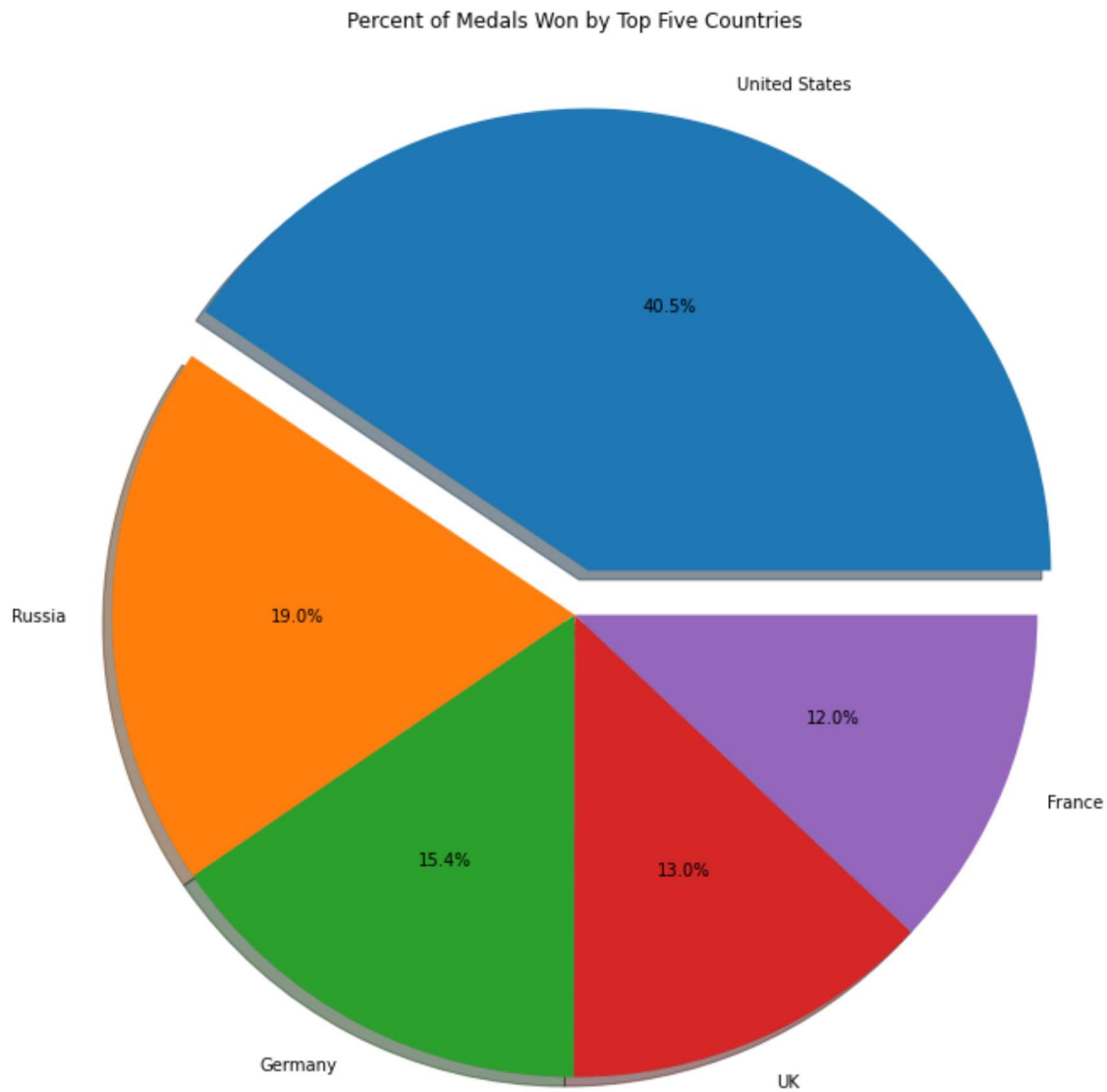
In [46]:

```
#We go back to the original "data" data set.
#We know this row has stricty medal winners, so we simple count have many times a country
 appears to know how many medals they won

top_five = data["Team"].value_counts()

top_five.head()

#The above os the top five countries with the most medals, now we use a "barplot" to displ
ay it.
#For a set so small, we can simple write in the values into a list.

x = ["United States", "Russia", "Germany", "UK", "France"]

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

plt.bar(x, y, color="yellow")
plt.xlabel("Countries")
plt.ylabel("Number of Medals Won")
plt.title("Top Five Countries with the Most Medals")
plt.show()
```



In [47]:

```
#We can also plot the above information in a "pie chart"

labels = ["United States", "Russia", "Germany", "UK", "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("Percent of Medals Won by Top Five Countries")
plt.rcParams["figure.figsize"] = (12,5)
plt.tight_layout()
plt.show()
```



In [ ]:

```
```

1. Number of Atheletes in each olympic games

In [48]:

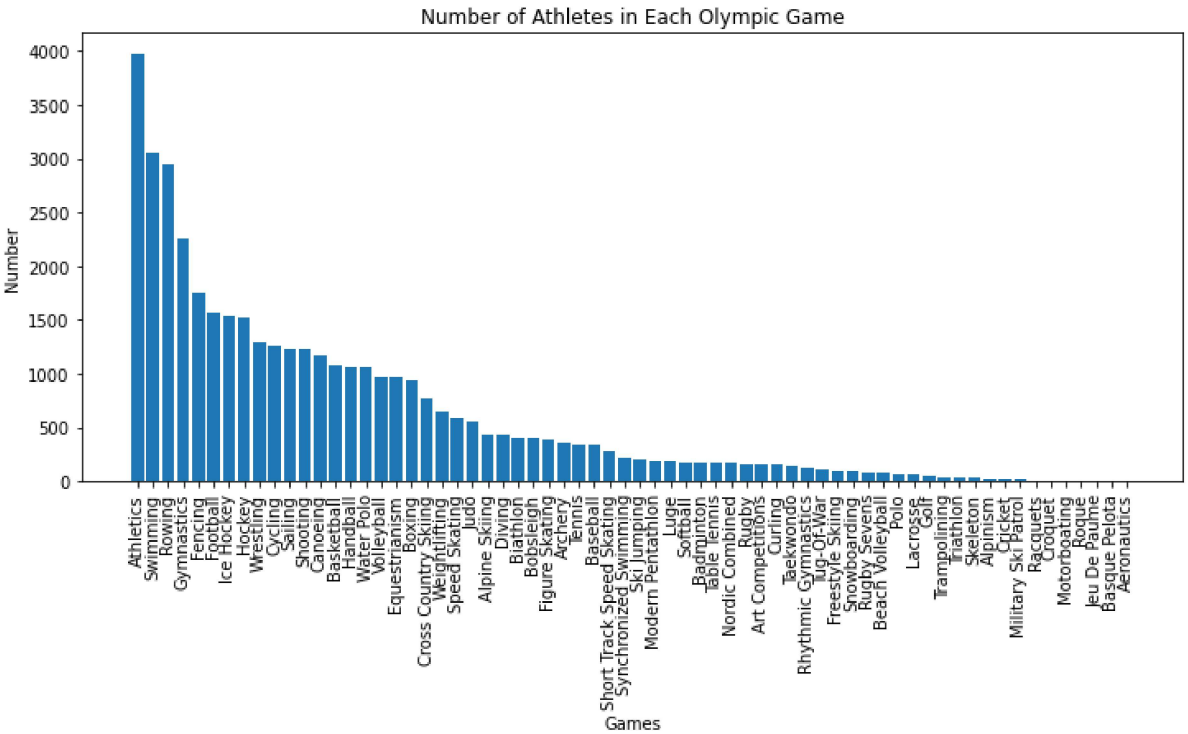
```
data.head()
```

Out[48]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Seasc
0	4	Edgar Lindenau Aabye	M	34.0	175.33897	70.702393	Denmark/Sweden	DEN	1900 Summer	1900	Summ
1	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1920	Summ
2	15	Arvo Ossian Aaltonen	M	30.0	175.33897	70.702393	Finland	FIN	1920 Summer	1920	Summ
3	16	Juhamatti Tapio Aaltonen	M	28.0	184.00000	85.000000	Finland	FIN	2014 Winter	2014	Wint
4	17	Paavo Johannes Aaltonen	M	28.0	175.00000	64.000000	Finland	FIN	1948 Summer	1948	Summ

In [49]:

```
event = data["Sport"].value_counts().to_dict()

event

x = (list(event.keys()))

y = (list(event.values()))

plt.bar(x, y)
plt.xticks(rotation=90)
plt.title("Number of Athletes in Each Olympic Game")
plt.xlabel("Games")
plt.ylabel("Number")
plt.rcParams["figure.figsize"] = (17,9)
plt.show()
```



Fin