Olympics Analysis 1896 to 2016, A deep Dive

By Obira Daniel,

Data from SportsStats

Learn SQL Basics for Data Science Specialization Capstone, Cousera, UC Davis

1.0 Data Description, Importing and Checking the Data

I created a parquet file from the csv file, reduced the file from ~40MB to ~5.6MB, it downloads and loads faster using pyarrow

Data is read from the web and loaded in to a sqlite database then all analysis stems from SQL queries ***sqlite3***, ***ipython-sql*** *and* ***pyarrow*** *are required for SQL and parquet interfacing accordingly*

1.1 Data Description

The file athlete_events.csv or olympics.parquet, contains 271,116 rows and 15 columns.
Each row/record corresponds to an individual athlete competing in an individual Olympic event (athlete-events).
The columns are the following:

ID - Unique ID for each individual athlete e.g 55881 for Michael Jordan, 13029 for Usain Bolt;
Name - Athlete's Full name;
Sex - M or F;
Age - 64 bit Float of Age in Years;
Height - 64 bit Float of height In centimeters;
Weight - 64 bit Float of Mass In kilograms;
Team - Team name;
NOC - National Olympic Committee 3-letter code;
Games - Year and season;
Year - Integer of Year of Event;
Season - Summer or Winter;
City - Host city;
Sport - Sport;
Event - Event;
Medal - Gold, Silver, Bronze, or NA.

1.2 Data Import, Quick Exploration and Quality Checks

1.21 Data Import/Loading

In [1]:
```
import os, datetime
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
```

```
import matplotlib.ticker as mticker
import numpy as np
import seaborn as sns
import pyarrow
import sqlite3
import scipy.stats

startTime = datetime.datetime.now()
#file='athlete_events.csv'
#Printing Styling
class style():
    BLACK = '\033[30m'
    RED = '\033[31m'
    GREEN = '\033[32m'
    YELLOW = '\033[33m'
    BLUE = '\033[34m'
    MAGENTA = '\033[35m'
    CYAN = '\033[36m'
    WHITE = '\033[37m'
    UNDERLINE = '\033[4m'
    RESET = '\033[0m'
    BOLD = '\033[1m'

#csvfile='https://raw.githubusercontent.com/obiradaniel/od_olympics/main/athlete_events.cs

parquetfile='https://raw.githubusercontent.com/obiradaniel/od_olympics/main/olympics.parqu

worldcountries = 'https://raw.githubusercontent.com/obiradaniel/od_olympics/main/world_cou

nocfile = "https://raw.githubusercontent.com/obiradaniel/od_olympics/main/noc_regions.csv"

olympics = pd.read_parquet(parquetfile, engine="pyarrow")
noc = pd.read_csv(nocfile)
countries = pd.read_csv(worldcountries)

olympics.head()
```

Out[1]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating

In [2]:

```
noc.describe(include='all')
```

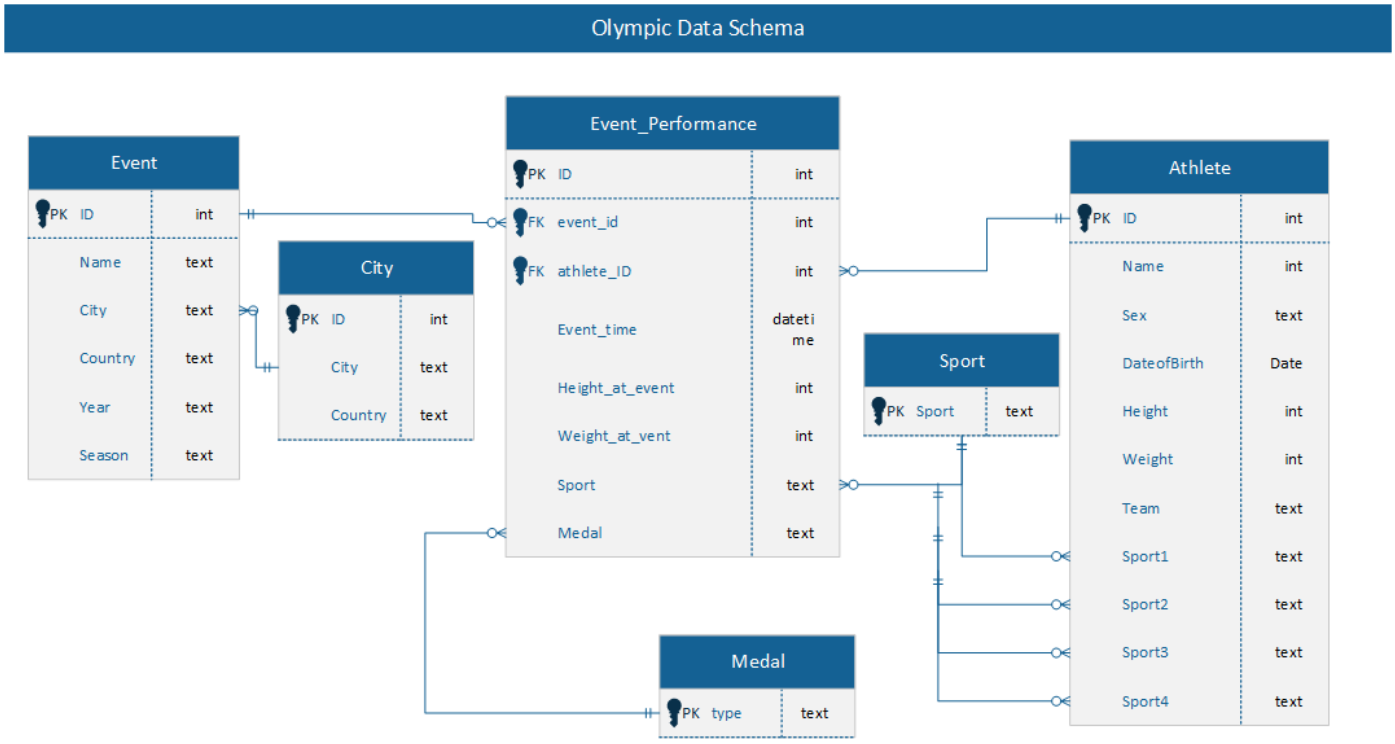
Out[2]:

	NOC	Country	notes

	NOC	Country	notes
count	232	232	21
unique	232	209	21
top	AFG	Germany	Netherlands Antilles
freq	1	4	1

They are 230 Unique NOCS and 206 uniqure reions or countries,
this data will be joined to the main olympic dataset after some basic cleaning

Olympic Data Schema



1.3 Checking the data for consistency and errors

In [3]:
```
records = olympics.shape[0]
columns = olympics.shape[1]

print("\nThe data has", records, "records and ", columns, "columns.\n")
print("Data Information Summary")
print(olympics.info())
```

```
The data has 271116 records and  15 columns.

Data Information Summary
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      271116 non-null  int64
 1   Name    271116 non-null  object
 2   Sex     271116 non-null  object
 3   Age     261642 non-null  float64
 4   Height  210945 non-null  float64
 5   Weight  208241 non-null  float64
 6   Team    271116 non-null  object
```

```
 7   NOC     271116 non-null  object
 8   Games   271116 non-null  object
 9   Year    271116 non-null  int64
 10  Season  271116 non-null  object
 11  City    271116 non-null  object
 12  Sport   271116 non-null  object
 13  Event   271116 non-null  object
 14  Medal   39783 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
None
```

In [4]:
```
print("Null Value Summary")
olympics.isnull().sum(axis = 0)
```

Out[4]:
```
Null Value Summary
ID              0
Name            0
Sex             0
Age          9474
Height      60171
Weight      62875
Team            0
NOC             0
Games           0
Year            0
Season          0
City            0
Sport           0
Event           0
Medal      231333
dtype: int64
```

In [5]:
```
print("Range of years is from", min(olympics.Year), "to ", max(olympics.Year))
```

```
Range of years is from 1896 to  2016
```

1.3.1 Data Description before Cleaning

In [6]:
```
print("Data Description for all Variables before cleaning")
olympics.describe(include='all')
```

```
Data Description for all Variables before cleaning
```

Out[6]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	
count	271116.000000	271116	271116	261642.000000	210945.000000	208241.000000	271116	271116	271116	2
unique	NaN	134732	2	NaN	NaN	NaN	1184	230	51	
top	NaN	Robert Tait McKenzie	M	NaN	NaN	NaN	United States	USA	2000 Summer	
freq	NaN	58	196594	NaN	NaN	NaN	17847	18853	13821	
mean	68248.954396	NaN	NaN	25.556898	175.338970	70.702393	NaN	NaN	NaN	
std	39022.286345	NaN	NaN	6.393561	10.518462	14.348020	NaN	NaN	NaN	
min	1.000000	NaN	NaN	10.000000	127.000000	25.000000	NaN	NaN	NaN	
25%	34643.000000	NaN	NaN	21.000000	168.000000	60.000000	NaN	NaN	NaN	
50%	68205.000000	NaN	NaN	24.000000	175.000000	70.000000	NaN	NaN	NaN	

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games
75%	102097.250000	NaN	NaN	28.000000	183.000000	79.000000	NaN	NaN	NaN
max	135571.000000	NaN	NaN	97.000000	226.000000	214.000000	NaN	NaN	NaN

Observations

1. Numeric Variables
   A. ID seems to show 135,571 unique athletes have ever attended the Olympics
   B. Age has extreme value of 97 years and so does weight with 214 Kg, this will have to be explored
   C. The rest seem fine
2. Text Variables
   A. Names shows 134732 unique values
   B. City, 42 Different Cities, a map will show them well
   C. Rest seem fine

Checking for Usain Bolt in that data

In [7]:
```
olympics.loc[olympics['Name'].str.contains('usain', case=False)].head()
```

Out[7]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	
3344	1883	Mamdooh Husain Isa Al-Doseri	M	20.0	NaN	NaN	Bahrain	BRN	1992 Summer	1992	Summer	Barcelona	Cycling	N k
3595	2061	Husain Al-Mutairi	M	18.0	170.0	52.0	Kuwait	KUW	1988 Summer	1988	Summer	Seoul	Boxing	
24876	13029	Usain St. Leo Bolt	M	17.0	196.0	95.0	Jamaica	JAM	2004 Summer	2004	Summer	Athina	Athletics	N
24877	13029	Usain St. Leo Bolt	M	21.0	196.0	95.0	Jamaica	JAM	2008 Summer	2008	Summer	Beijing	Athletics	N
24878	13029	Usain St. Leo Bolt	M	21.0	196.0	95.0	Jamaica	JAM	2008 Summer	2008	Summer	Beijing	Athletics	N

1.3.2 Data Checks

Test 1: Checking for Double Spaces in Athlete Names, should give blank Values

In [8]:
```
olympics.loc[olympics['Name'].str.contains('  ', case=False)]['Name'].value_counts()#Name
```

Out[8]:
```
Jean Honor  Gounot                          17
Paolo Francesco "Paul"  Radmilovic          13
Florence  Andre Isabelle Baverel-Robert     12
Mladenka  Malenia (-Vidovic)                 6
Isabella Minna Veronica  Wagner              5
                                            ..
Ana Cludia  da Costa Gonalves                1
Virginia Anne  "Ginny" Gilder                1
Petru Ionel  Gavril                          1
```

```
Jorge Ignacio  Garbey Castillo              1
Edmund Roman  Zieliski                      1
Name: Name, Length: 128, dtype: int64
```

128 Athletes Names have double spaces

In [9]:
```
olympics.loc[olympics['Team'].str.contains('  ', case=False)]['Team'].value_counts()#Team
```

Out[9]:
```
Series([], Name: Team, dtype: int64)
```

In [10]:
```
olympics.loc[olympics['Sport'].str.contains('  ', case=False)]['Sport'].value_counts()#Spo
```

Out[10]:
```
Series([], Name: Sport, dtype: int64)
```

In [11]:
```
olympics.loc[olympics['Event'].str.contains('  ', case=False)]['Event'].value_counts()#Eve
```

Out[11]:
```
Series([], Name: Event, dtype: int64)
```

In [12]:
```
olympics.loc[olympics['City'].str.contains('  ', case=False)]['City'].value_counts()#Event
```

Out[12]:
```
Series([], Name: City, dtype: int64)
```

Test 2: Checking for Consinstent Single Spaces and Uniform Cases, should give consitent output, should easily show inconsitency

In [13]:
```
olympics.loc[olympics['Name'].str.contains(' ', case=False)]['Name'].value_counts()#Name V
```

Out[13]:
```
Robert Tait McKenzie           58
Heikki Ilmari Savolainen       39
Joseph "Josy" Stoffel          38
Ioannis Theofilakis            36
Takashi Ono                    33
                               ..
Meng Fanlong                    1
Jennifer Susan "Jenny" Duck     1
Peter Ducke                     1
Marcela Menezes                 1
A Dijiang                       1
Name: Name, Length: 134448, dtype: int64
```

In [14]:
```
olympics.loc[olympics['Team'].str.contains(' ', case=False)]['Team'].value_counts()#Team V
```

Out[14]:
```
United States       17847
Great Britain       11404
Soviet Union         5535
South Korea          4344
West Germany         3199
                    ...
Quando Quando           1
Whitini Star            1
Pop Art                 1
Pierre et Jean-3        1
Dow Jones               1
Name: Team, Length: 435, dtype: int64
```

In [15]:
```
olympics.loc[olympics['Sport'].str.contains(' ', case=False)]['Sport'].value_counts()#Spor
```

Out[15]:
```
Cross Country Skiing          9133
Alpine Skiing                 8829
Speed Skating                 5613
Ice Hockey                    5516
Water Polo                    3846
Art Competitions              3578
Ski Jumping                   2401
Figure Skating                2298
Table Tennis                  1955
Modern Pentathlon             1677
Short Track Speed Skating     1534
Nordic Combined               1344
Freestyle Skiing               937
Synchronized Swimming          909
Rhythmic Gymnastics            658
Beach Volleyball               564
Rugby Sevens                   299
Military Ski Patrol             24
Jeu De Paume                    11
Basque Pelota                    2
Name: Sport, dtype: int64
```

In [16]:
```
olympics.loc[olympics['Event'].str.contains(' ', case=False)]['Event'].value_counts()#Ever
```

Out[16]:
```
Football Men's Football                                  5733
Ice Hockey Men's Ice Hockey                              4762
Hockey Men's Hockey                                      3958
Water Polo Men's Water Polo                              3358
Basketball Men's Basketball                              3280
                                                         ...
Croquet Mixed Doubles                                       2
Archery Men's Target Archery, 50 metres, Individual         2
Archery Men's Target Archery, 33 metres, Individual         2
Archery Men's Target Archery, 28 metres, Individual         2
Aeronautics Mixed Aeronautics                               1
Name: Event, Length: 765, dtype: int64
```

In [17]:
```
olympics.loc[olympics['City'].str.contains(' ', case=False)]['City'].value_counts()#Event
```

Out[17]:
```
Rio de Janeiro       13688
Los Angeles          12423
Mexico City           8588
Salt Lake City        4109
Lake Placid           2098
Sankt Moritz          1657
Cortina d'Ampezzo     1307
St. Louis             1301
Squaw Valley          1116
Name: City, dtype: int64
```

By and Large, the case seems to be consitent

In [18]:
```
#olympics.loc[olympics['Name'].str.contains('usain', case=False)]
#olympics.loc[olympics['Team'].str.contains(' ', case=False)]
```

1.3.4 Data Cleanup Summary

The file athlete_events.csv or olympics.parquet, contains 271,116 rows and 15 columns.
Each row/record corresponds to an individual athlete competing in an individual Olympic event (athlete-events).
The columns are the following:

ID - Unique ID, assumed to be clean, no Nulls ***will check would be to see if the same athlete from one NOC has 2 Ids***
Name - Athlete's Full name,134732, unique values ***will be trimmed and double spaces replace with single space***
Sex - M or F; Clean
Age - Integer of years, 9,474 Nulls, ***nothing can be done here, may be check if some athletes declared age in later years then backfill***
Height - In centimeters, okay, 60,171 Nulls
Weight - In kilograms, okay, 62,875 Nulls
Team - Team name, will pick the Country name from the NOC file;***checked, clean***
NOC - National Olympic Committee 3-letter code;
Games - Year and season;
Year - Integer;
Season - Summer or Winter;
City - Host city, ***checked, clean***
Sport - Sport;***checked, clean***
Event - Event;***checked, clean***
Medal - Gold, Silver, Bronze, or NA.

Will add EventScore, Gold ->10, Silver ->7.5, Bronze->5, or NA->1 for qualifying.
Will add medal binary, Gold ->1, Silver ->1, Bronze->1, or NA-0

Data will be joined to NOC to get Country

Performing the cleanups

In [19]:
```
olympics['Name'] = olympics['Name'].str.replace('  ', ' ')#Replace double space with singl
olympics['Name'] = olympics.Name.str.strip()
```

Checking the cleaned columns

In [20]:
```
olympics.loc[olympics['Name'].str.contains('  ', case=False)]['Name'].value_counts()#Name
```

Out[20]:
```
Series([], Name: Name, dtype: int64)
```

In [21]:
```
olympics.loc[olympics['Name'].str.contains(' ', case=False)]['Name'].value_counts()#Name
```

Out[21]:
```
Robert Tait McKenzie          58
Heikki Ilmari Savolainen      39
Joseph "Josy" Stoffel         38
Ioannis Theofilakis           36
Takashi Ono                   33
                              ..
Maria Ulrika Kalte             1
Josef Kalt (-Arnet)            1
Katri Johanna Kalpala          1
Dorothea "Dora" Kalpakidou     1
Georges Marcel Lecointe        1
Name: Name, Length: 134447, dtype: int64
```

1.3.4 Data Transfer to SQL, SQLite

This uses ipython-sql, if missing please install

Joinining the Cleaned Data to NOC

In [22]:
```
#olympics_merge = pd.merge(olympics, noc, on='NOC', how='left')
```

In [23]:
```
#pip install ipython-sql
```

In [24]:
```
%load_ext sql
```

In [25]:
```
conn = sqlite3.connect('Olympics.db')
cur = conn.cursor()
```

In [26]:
```
%sql sqlite:///Olympics.db
```

In [27]:
```
noc.to_sql("NOC", conn, if_exists='replace', index=False,method="multi")
countries.to_sql("Countries", conn, if_exists='replace', index=False,method="multi")
olympics.to_sql("Olympics", conn, if_exists='replace', index=False,method=None);
#None method is used for Olympics since they are 271,116 rows to enable using insert and a
#OperationalError: too many SQL variables, caused by trying to write all rows at Once.
#chunksize can also be used
```

In [28]:
```
#list of all tables in the database and a basic description and SQL Create Code
%sql select name from sqlite_master WHERE type='table';
```

```
 * sqlite:///Olympics.db
Done.
```

Out[28]:

name
NOC
Countries
Olympics

In [29]:
```
#%sql SELECT name,type,length(type) FROM PRAGMA_TABLE_INFO('Olympics');
```

In [30]:
```
%sql SELECT * FROM Olympics LIMIT 5;
```

```
 * sqlite:///Olympics.db
Done.
```

Out[30]:

ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	
1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	B
2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Ju Lig
3	Gunnar Nielsen Aaby	M	24.0	None	None	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	
4	Edgar	M	34.0	None	None	Denmark/Sweden	DEN	1900	1900	Summer	Paris	Tug-Of-	

	Lindenau Aabye									Summer					War	V
5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating				5(

1.3.5 Data Description After Cleaning

Overvwriting the old olympics data from parquet with this loadded from the Database.

In [31]:
```
olympics=pd.read_sql_query("SELECT * FROM Olympics LEFT JOIN NOC ON Olympics.NOC= NOC.NOC;
```

In [32]:
```
#olympics_all.rename(columns = {'region':'Country'}, inplace = True)
len(olympics) #Should be 271,116
```

Out[32]:
```
271116
```

In [33]:
```
olympics[olympics['Country'].isnull()]
```

Out[33]:

ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal	NOC	Cou

In [34]:
```
print("Data Description for all Variables After cleaning")
olympics.describe(include='all')
```

```
Data Description for all Variables After cleaning
```

Out[34]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	
count	271116.000000	271116	271116	261642.000000	210945.000000	208241.000000	271116	271116	271116	2
unique	NaN	134731	2	NaN	NaN	NaN	1184	230	51	
top	NaN	Robert Tait McKenzie	M	NaN	NaN	NaN	United States	USA	2000 Summer	
freq	NaN	58	196594	NaN	NaN	NaN	17847	18853	13821	
mean	68248.954396	NaN	NaN	25.556898	175.338970	70.702393	NaN	NaN	NaN	
std	39022.286345	NaN	NaN	6.393561	10.518462	14.348020	NaN	NaN	NaN	
min	1.000000	NaN	NaN	10.000000	127.000000	25.000000	NaN	NaN	NaN	
25%	34643.000000	NaN	NaN	21.000000	168.000000	60.000000	NaN	NaN	NaN	
50%	68205.000000	NaN	NaN	24.000000	175.000000	70.000000	NaN	NaN	NaN	
75%	102097.250000	NaN	NaN	28.000000	183.000000	79.000000	NaN	NaN	NaN	
max	135571.000000	NaN	NaN	97.000000	226.000000	214.000000	NaN	NaN	NaN	

In [35]:
```
y=%sql SELECT * FROM Olympics LEFT JOIN NOC ON Olympics.NOC= NOC.NOC;
```

```
 * sqlite:///Olympics.db
Done.
```

In [36]:
```
len(y) #Should alse  be 271,116 to match the pandas query
```

Out[36]:
```
271116
```

2.0 Data Exploration and Visualization

Global Seaborn Settings

In [37]:
```
#sns.set_style("darkgrid")
sns.set_theme(context='notebook', style='darkgrid', palette='deep', font_scale=1, color_co
sns.set(rc = {'figure.figsize':(15,8)})
'''
ax.spines['left'].set_color('black')        # setting up Y-axis tick color to red
ax.spines['top'].set_color('black')
ax.spines['bottom'].set_color('black')        # setting up Y-axis tick color to red
ax.spines['right'].set_color('black')
'''
#.grid(color='black')
```

Out[37]:
```
"\nax.spines['left'].set_color('black')        # setting up Y-axis tick color to red\nax.s
pines['top'].set_color('black') \nax.spines['bottom'].set_color('black')        # setting
up Y-axis tick color to red\nax.spines['right'].set_color('black') \n"
```

Spliting Data into Summer and Winter Oylpmics

The reason for this is most African Countries don't participate in Winter Olympics and don't even have winter

In [38]:
```
summer = olympics[olympics["Season"]=="Summer"]
winter = olympics[olympics["Season"]=="Winter"]
```

In [39]:
```
olympics['ID'].value_counts().describe()
```

Out[39]:
```
count    135571.000000
mean          1.999808
std           1.990221
min           1.000000
25%           1.000000
50%           1.000000
75%           2.000000
max          58.000000
Name: ID, dtype: float64
```

Function to show seaborn plot values

In [172…
```
def show_values(axs, orient="v", space=.02):
    """Function to show plot values"""
    def _single(ax):
        if orient == "v":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() / 2
                h= p.get_height()
                if pd.isna(h) or h==0:
                    value = ''
                    _y=0
                else:
                    _y = p.get_y() + p.get_height() + (p.get_height()*0.01)
                    #value = '{:.1f}'.format(p.get_height())
                    value = '{:,d}'.format(int(p.get_height()))
```

```
                ax.text(_x, _y, value, ha="center")
        elif orient == "h":
            for p in ax.patches:
                x = p.get_x()
                if pd.isna(x) or x==0:
                    value = ''
                    _x=0
                else:
                    _x = p.get_x() + p.get_width() + float(space)
                    _y = p.get_y() + p.get_height() - (p.get_height()*0.5)
                #value = '{:.1f}'.format(p.get_width())
                value = '{:,d}'.format(p.get_width())
                ax.text(_x, _y, value, ha="left")

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _single(ax)
    else:
        _single(axs)
```

2. 1 Olympics Athlete Attendance by Year, All Olympics

In [41]:
```
all_athletes_yearly = %sql select Year, COUNT(*) AS Athletes from Olympics GROUP BY Year ;
```

```
 * sqlite:///Olympics.db
Done.
```

In [42]:
```
#Getting the Data Ready
all_athletes_yearly = pd.read_sql_query("select Year, COUNT(*) AS Athletes from Olympics (
```

In [43]:
```
unique_athletes_yearly = olympics.groupby(['Year'])[['ID']].nunique().reset_index()
```

In [44]:
```
#olympics.groupby('Year')[['ID']].count().reset_index()
```

2.1 Olympics Athlete Attendance by Year by Winter/Summer

In [148…
```
#Data preparation
#all_athletes_year_season =olympics.groupby(['Year','Season'])[['ID']].count().reset_index
unique_athletes_year_season = olympics.groupby(['Year','Season'])[['ID']].nunique().reset_
records = unique_athletes_year_season.shape[0]
```

In [149…
```
years = unique_athletes_year_season.Year.values.tolist()
```

In [150…
```
missing_summer_years = [ i for i in list(range(1896,2020,4)) if i not in years]
```

In [132…
```
unique_athletes_year_season.head()
```

Out[132…

	Year	Season	ID
0	1896	Summer	176
1	1900	Summer	1224
2	1904	Summer	650
3	1906	Summer	841

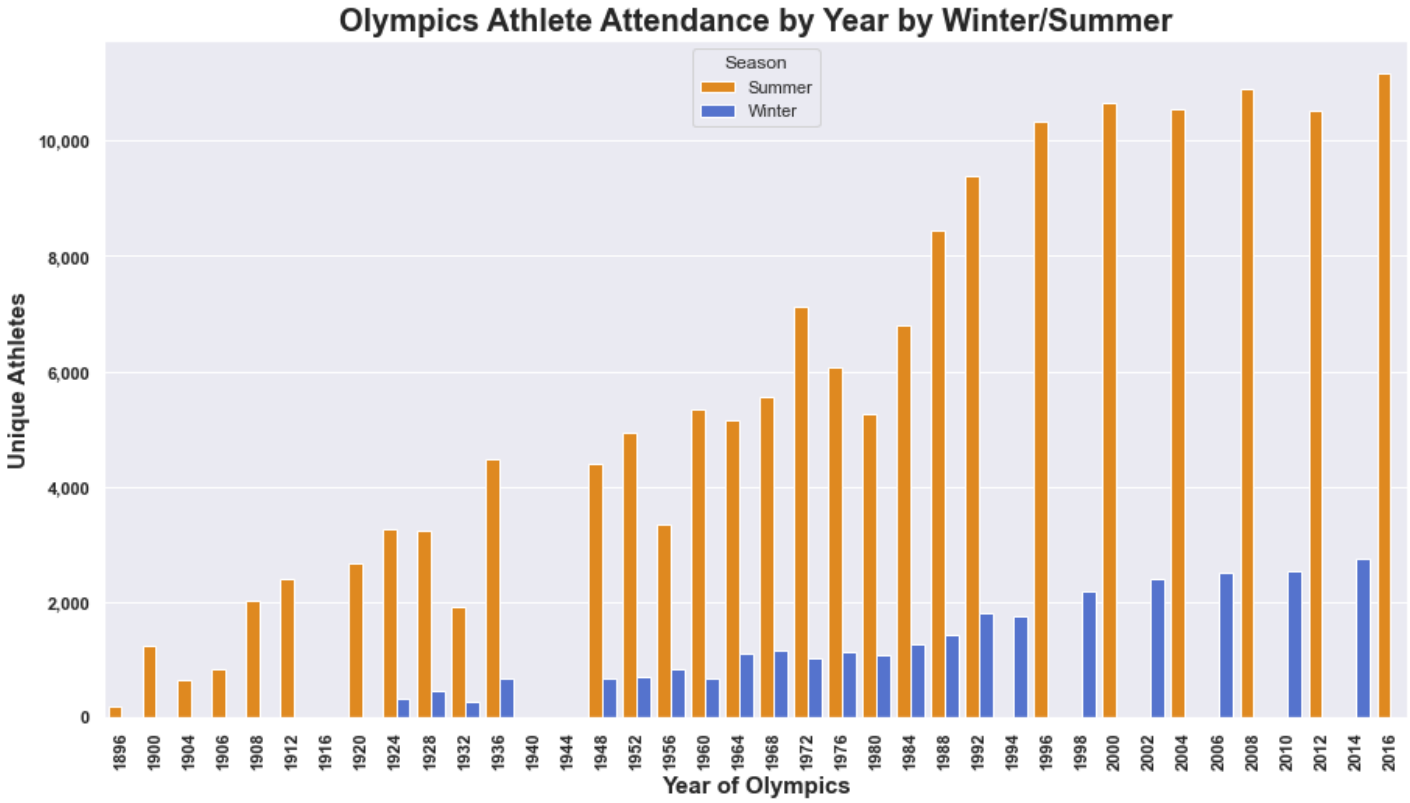
	Year	Season	ID
4	1908	Summer	2024

In [133…
```
unique_athletes_year_season.columns
```

Out[133…
```
Index(['Year', 'Season', 'ID'], dtype='object')
```

In [155…
```
for i in range(len(missing_summer_years)):
    unique_athletes_year_season.loc[records +i] =[missing_summer_years[i], 'Summer', 0]
```

In [163…
```
#ax=sns.lineplot(x='Year', y='ID', hue = 'Season', data=unique_athletes_year_season, palet
ax=sns.barplot(x='Year', y='ID', hue = 'Season', data=unique_athletes_year_season, palette
#ax=sns.countplot(x='Year', hue = 'Season', data=olympics, palette=['darkorange', 'royalbl
ax.set_title('Olympics Athlete Attendance by Year by Winter/Summer', fontsize = 20,weight=
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Unique Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

#plt.yticks(list(range(0,100,10)));
#plt.xticks(list(range(1896,2020,4)));

#ax.set_xticklabels(list(range(1896,2020,4)))

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
#show_values(ax)
```



2.3 All Olympics Athlete Attendance by Year, Gender

In [173…
```
#Data preparation
#all_athletes_year_sex = olympics.groupby(['Year','Sex'])[['ID']].count().reset_index()
```

```
unique_athletes_year_sex = olympics.groupby(['Year','Sex'])[['ID']].nunique().reset_index
records = unique_athletes_year_sex.shape[0]
```

In [164…

```
unique_athletes_year_sex.head()
```

Out[164…

	Year	Sex	ID
0	1896	M	176
1	1900	F	23
2	1900	M	1201
3	1904	F	6
4	1904	M	644

In [165…

```
unique_athletes_year_sex.columns
```

Out[165…

```
Index(['Year', 'Sex', 'ID'], dtype='object')
```

In [166…

```
unique_athletes_year_sex.shape[0]
```

Out[166…

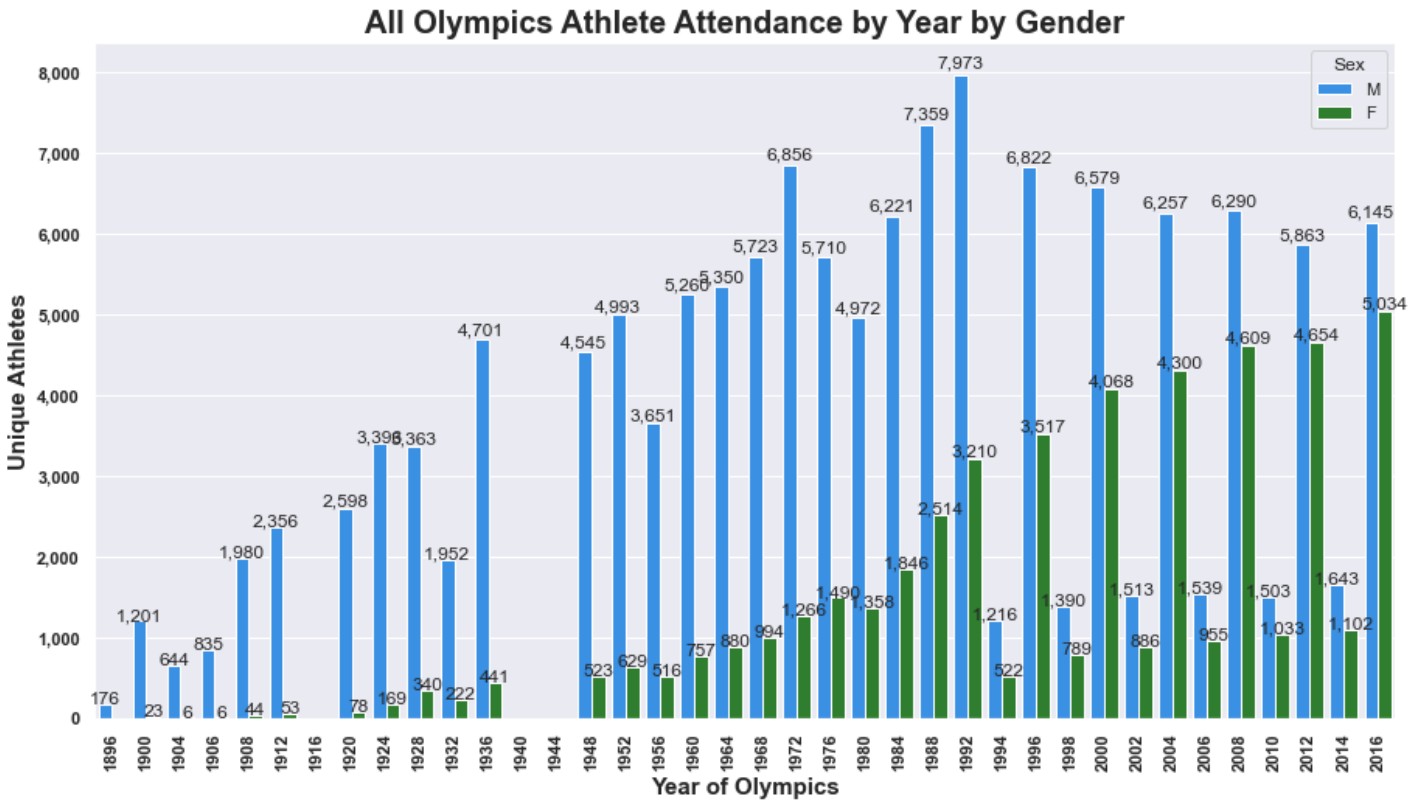
```
69
```

In [174…

```
for i in range(len(missing_summer_years)):
    unique_athletes_year_sex.loc[records +i] =[missing_summer_years[i], 'M', 0]

for i in range(len(missing_summer_years)):
    unique_athletes_year_sex.loc[records +i+3] =[missing_summer_years[i], 'F', 0]
```

In [175…

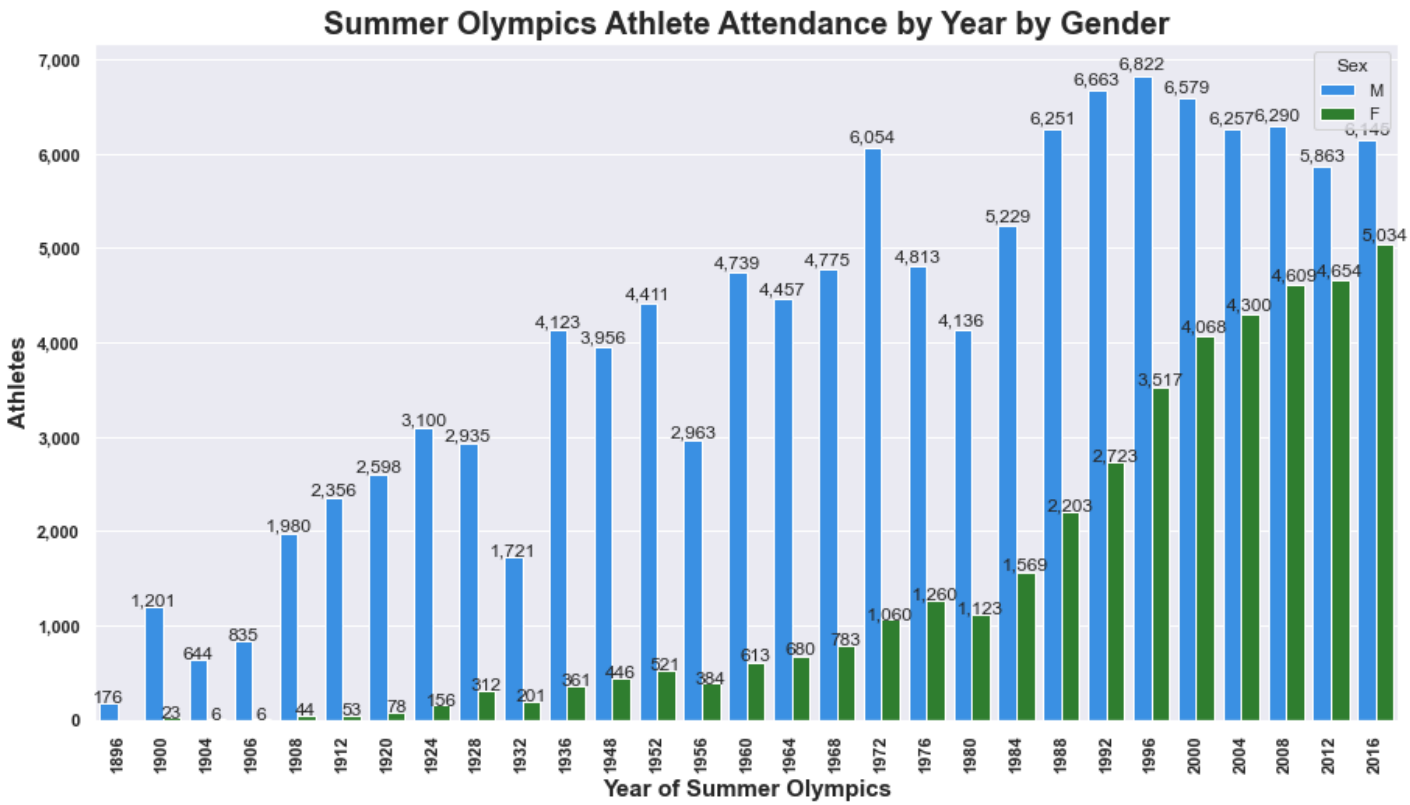
```
ax=sns.barplot(x='Year', y='ID', hue = 'Sex', data=unique_athletes_year_sex, palette=['doc
ax.set_title('All Olympics Athlete Attendance by Year by Gender', fontsize = 20,weight='bo
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Unique Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
show_values(ax)
```



2.4 Summer Olympics Athlete Attendance by Year, Gender

In [50]:
```
#Data preparation
all_athletes_summer_sex = summer.groupby(['Year','Sex'])[['ID']].count().reset_index()
unique_athletes_summer_sex = summer.groupby(['Year','Sex'])[['ID']].nunique().reset_index
```

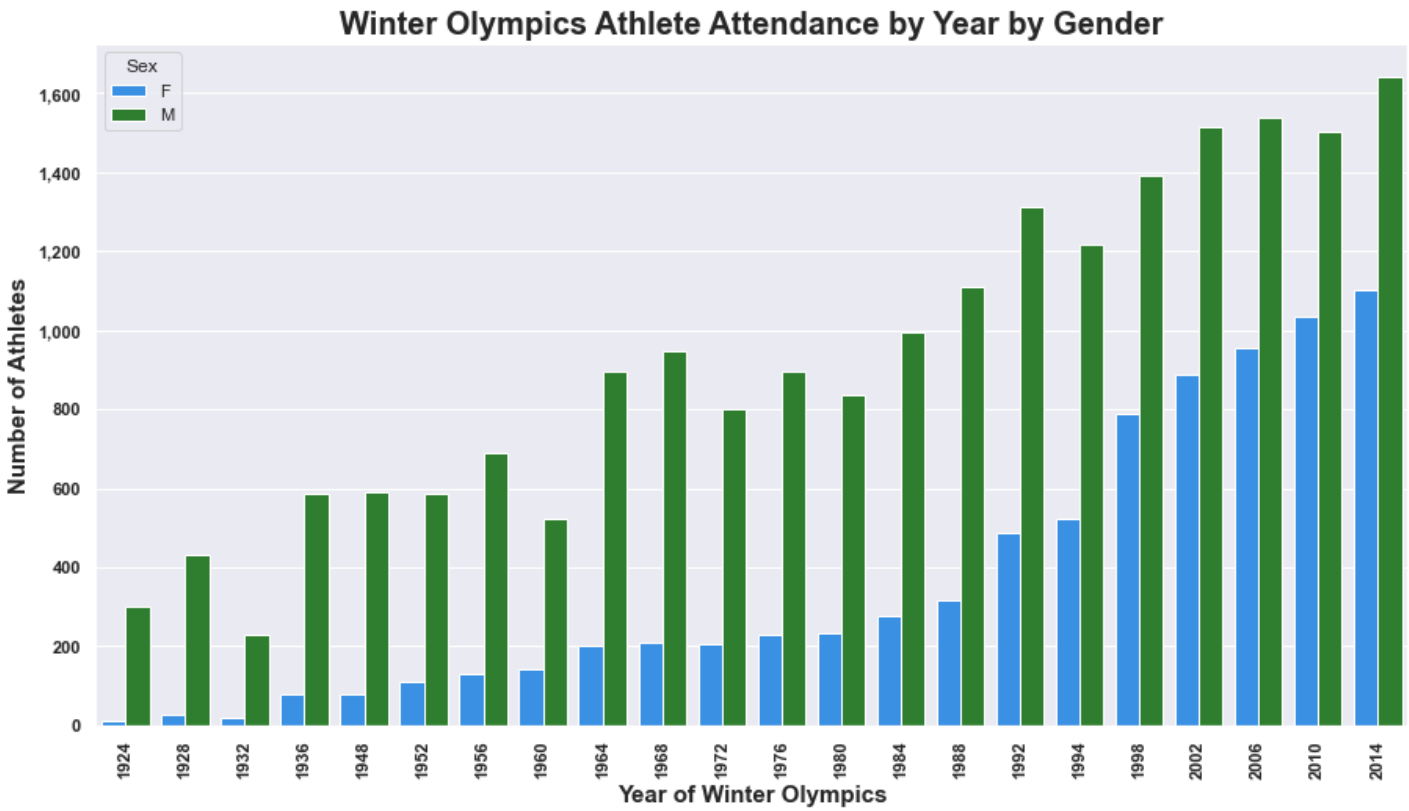
In [51]:
```
#ax=sns.barplot(x='Year', y='ID', hue = 'Sex', data=all_athletes_summer_sex, palette=['doc
ax=sns.barplot(x='Year',y='ID', hue = 'Sex', data=unique_athletes_summer_sex, palette=['dc
ax.set_title('Summer Olympics Athlete Attendance by Year by Gender', fontsize = 20,weight=
ax.set_xlabel("Year of Summer Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
show_values(ax)
```



2.5 Winter Olympics Athlete Attendance by Year, Gender

In [52]:
```
#Data preparation
all_athletes_winter_sex = winter.groupby(['Year','Sex'])[['ID']].count().reset_index()
unique_athletes_winter_sex = winter.groupby(['Year','Sex'])[['ID']].nunique().reset_index
```

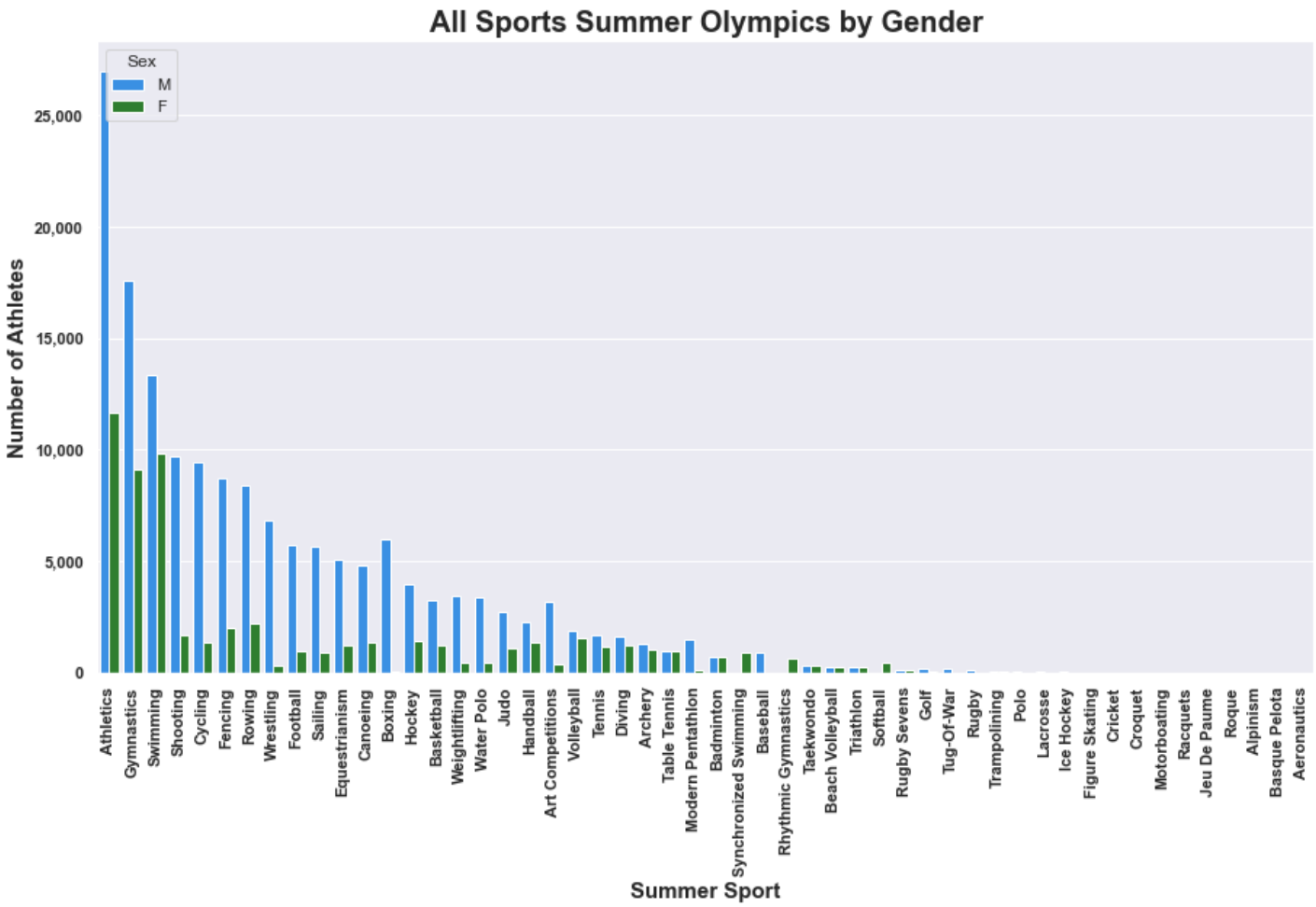
In [53]:
```
#ax=sns.countplot(x='Year', hue = 'Sex', data=winter, palette=['forestgreen', 'dodgerblue
ax=sns.barplot(x='Year',y='ID', hue = 'Sex', data=unique_athletes_winter_sex, palette=['do
ax.set_title('Winter Olympics Athlete Attendance by Year by Gender', fontsize = 20,weight=
ax.set_xlabel("Year of Winter Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
#show_values(ax)
```



2.6 All Summer Olympics Sports by Gender

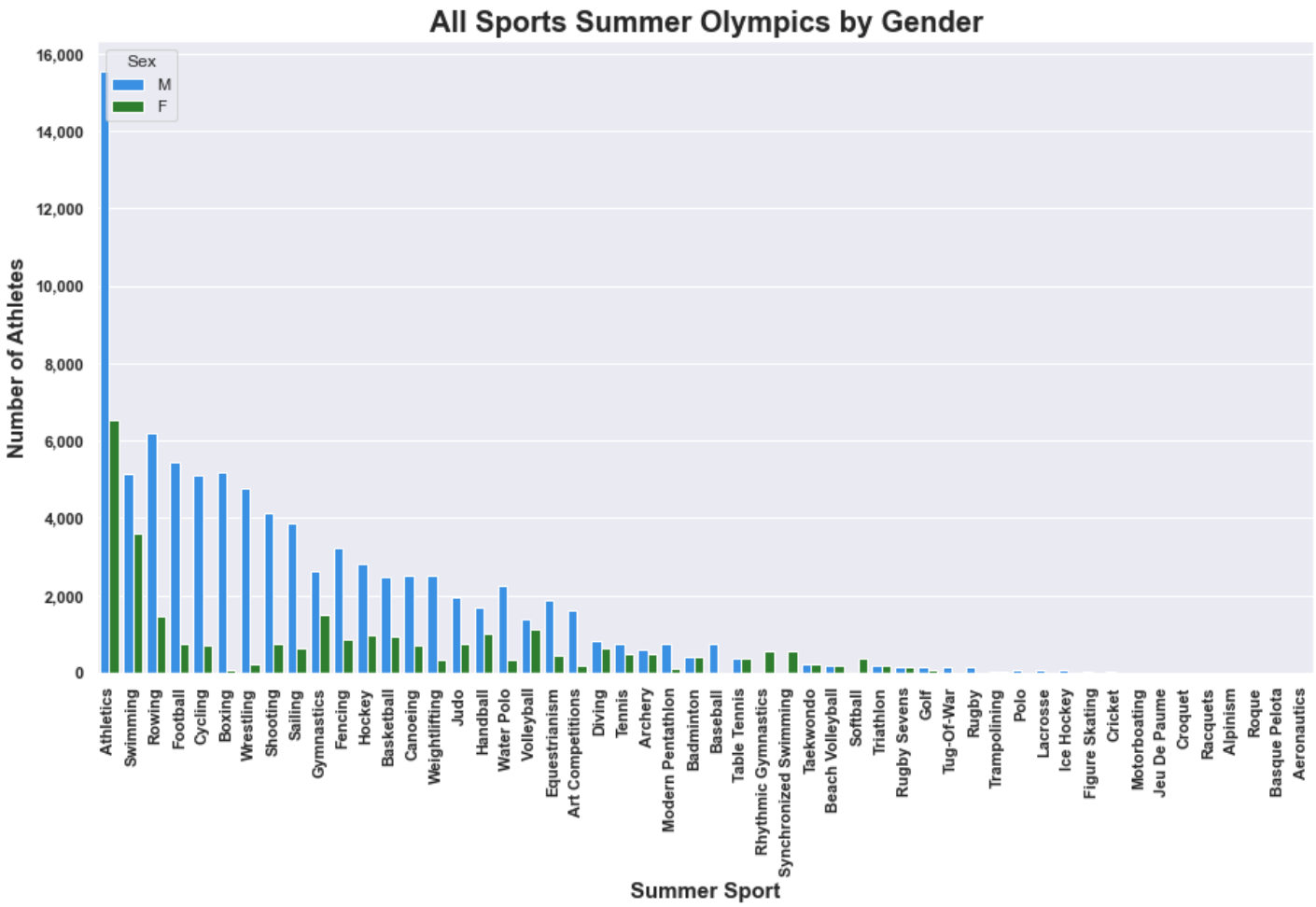
In [54]:
```
#Data preparation
#all_athletes_summer_sex = summer.groupby(['Year','Sex'])[['ID']].count().reset_index()
unique_athletes_summer_sport = summer.groupby(['Sport','Sex'])[['ID']].nunique().reset_inc
```

In [55]:
```
ax=sns.countplot(x='Sport',hue = 'Sex', data=summer, palette=['dodgerblue', 'forestgreen']
#sortorder = summer.groupby(['Sport'])[['ID']].nunique().sort_values(by='ID', ascending=Fa
#ax=sns.barplot(x='Sport',y='ID', hue = 'Sex', data=unique_athletes_summer_sport, palette=
ax.set_title('All Sports Summer Olympics by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Summer Sport",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
#show_values(ax)
```



In [56]:

```
#ax=sns.countplot(x='Sport',hue = 'Sex', data=summer, palette=['dodgerblue', 'forestgreen
sortorder = summer.groupby(['Sport'])[['ID']].nunique().sort_values(by='ID', ascending=Fal
ax=sns.barplot(x='Sport',y='ID', hue = 'Sex', data=unique_athletes_summer_sport, palette=
ax.set_title('All Sports Summer Olympics by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Summer Sport",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
#show_values(ax)
```



In [57]:
```
unique_athletes_summer_sport.sort_values(by='ID', ascending=False)['Sport'].unique
```

Out[57]:
```
<bound method Series.unique of 8         Athletics
7         Athletics
58           Rowing
35          Football
18            Boxing
           ...
22           Croquet
14     Basque Pelota
1           Alpinism
51      Motorboating
0        Aeronautics
Name: Sport, Length: 89, dtype: object>
```

In [58]:
```
summer.groupby(['Sport'])[['ID']].nunique().sort_values(by='ID', ascending=False).reset_ir
```

Out[58]:
```
0                 Athletics
1                  Swimming
2                    Rowing
3                  Football
4                   Cycling
5                    Boxing
6                 Wrestling
7                  Shooting
8                   Sailing
9                 Gymnastics
10                   Fencing
11                    Hockey
12                Basketball
13                  Canoeing
14             Weightlifting
```

```
15                    Judo
16                 Handball
17               Water Polo
18               Volleyball
19            Equestrianism
20          Art Competitions
21                   Diving
22                   Tennis
23                  Archery
24         Modern Pentathlon
25                 Badminton
26                  Baseball
27              Table Tennis
28       Rhythmic Gymnastics
29     Synchronized Swimming
30                 Taekwondo
31          Beach Volleyball
32                  Softball
33                 Triathlon
34              Rugby Sevens
35                      Golf
36               Tug-Of-War
37                    Rugby
38              Trampolining
39                      Polo
40                 Lacrosse
41               Ice Hockey
42            Figure Skating
43                   Cricket
44              Motorboating
45              Jeu De Paume
46                   Croquet
47                  Racquets
48                  Alpinism
49                     Roque
50              Basque Pelota
51                Aeronautics
Name: Sport, dtype: object
```

In [59]:

```
sports = summer['Sport'].value_counts()
y2=summer.groupby(['Sport'])[['ID']].nunique().reset_index()
y2
```

Out[59]:

	Sport	ID
0	Aeronautics	1
1	Alpinism	4
2	Archery	1113
3	Art Competitions	1814
4	Athletics	22071
5	Badminton	811
6	Baseball	761
7	Basketball	3413
8	Basque Pelota	2
9	Beach Volleyball	383
10	Boxing	5262

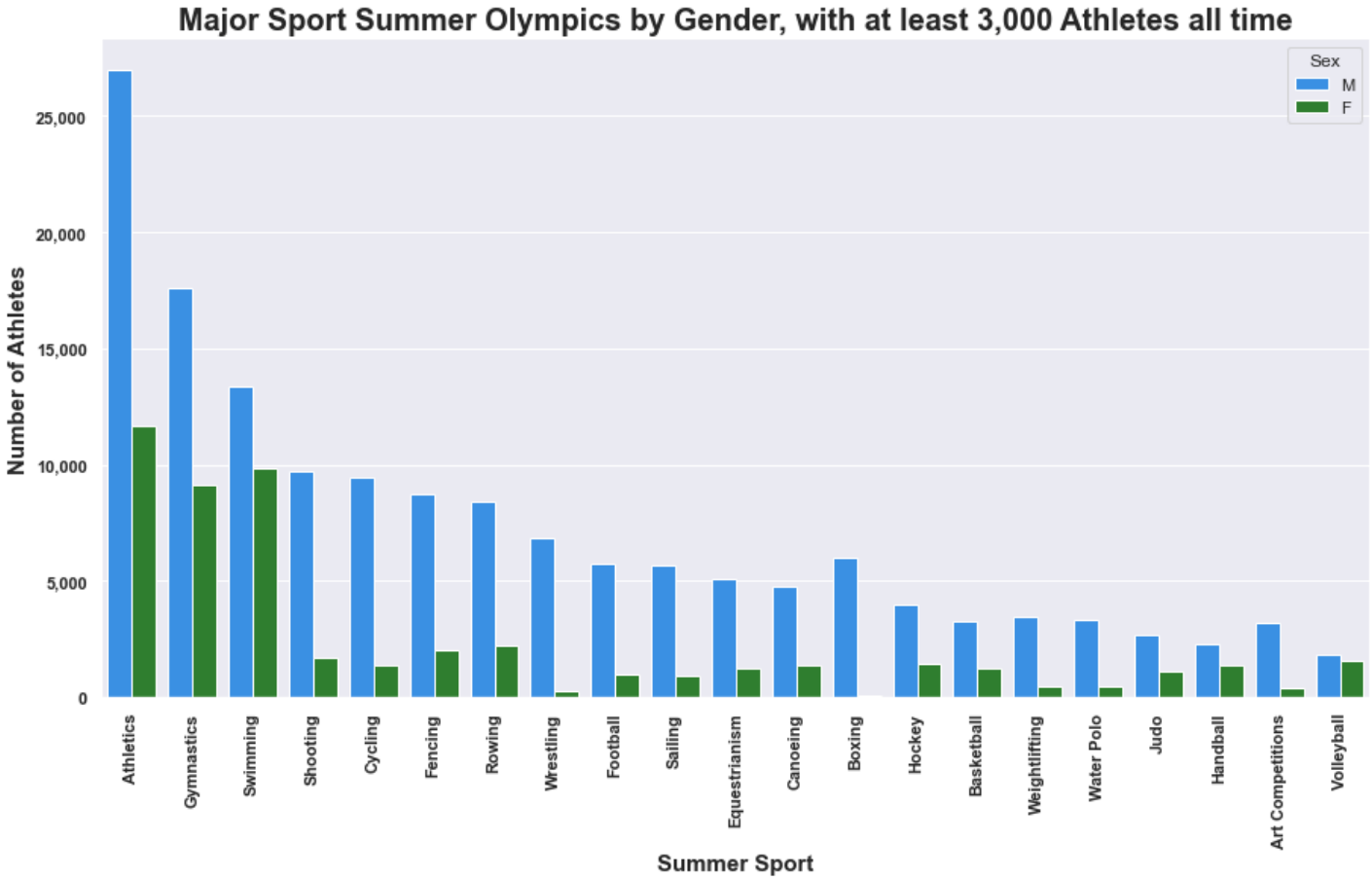
	Sport	ID
11	Canoeing	3206
12	Cricket	24
13	Croquet	10
14	Cycling	5819
15	Diving	1466
16	Equestrianism	2345
17	Fencing	4123
18	Figure Skating	45
19	Football	6161
20	Golf	218
21	Gymnastics	4134
22	Handball	2702
23	Hockey	3825
24	Ice Hockey	60
25	Jeu De Paume	11
26	Judo	2724
27	Lacrosse	60
28	Modern Pentathlon	864
29	Motorboating	14
30	Polo	87
31	Racquets	7
32	Rhythmic Gymnastics	567
33	Roque	4
34	Rowing	7687
35	Rugby	155
36	Rugby Sevens	299
37	Sailing	4480
38	Shooting	4882
39	Softball	367
40	Swimming	8765
41	Synchronized Swimming	550
42	Table Tennis	749
43	Taekwondo	470
44	Tennis	1246
45	Trampolining	93
46	Triathlon	355

	Sport	ID
47	Tug-Of-War	160
48	Volleyball	2503
49	Water Polo	2599
50	Weightlifting	2882
51	Wrestling	4988

In [60]:
```
sports_3k=sports[sports > 3000]
sports_3k_less=sports[sports <= 3000]
```

2.7 Major Sport Summer Olympics by Gender, with at least 3,000 Athletes all time

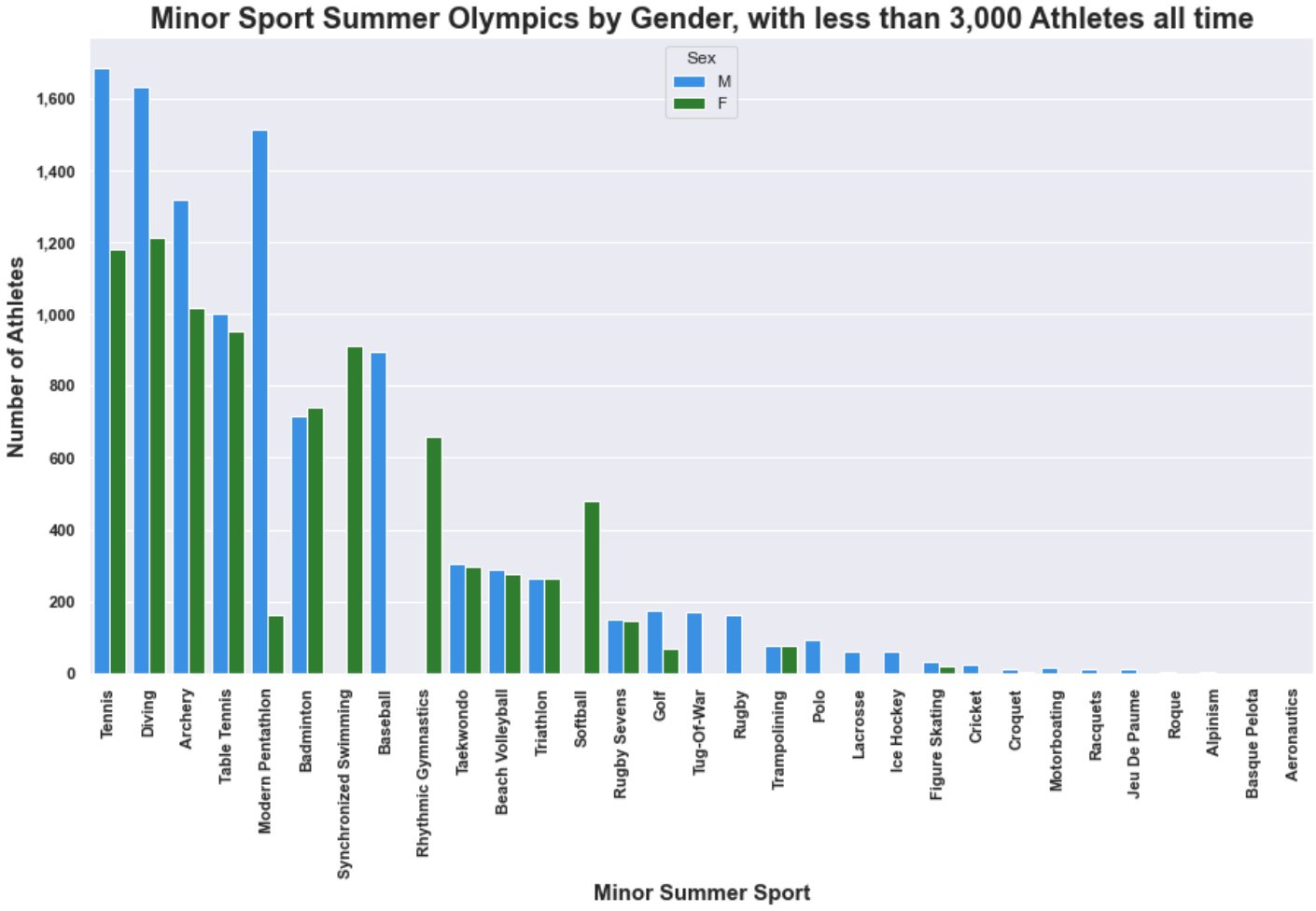
In [61]:
```
ax=sns.countplot(x='Sport',hue = 'Sex', data=summer[summer['Sport'].isin(sports_3k.index)]
ax.set_title('Major Sport Summer Olympics by Gender, with at least 3,000 Athletes all time
ax.set_xlabel("Summer Sport",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
#show_values(ax)
```



2.8 Minor Sport Summer Olympics by Gender, with less than 3,000 Athletes all time

In [62]:
```
ax=sns.countplot(x='Sport',hue = 'Sex', data=summer[summer['Sport'].isin(sports_3k_less.ir
ax.set_title('Minor Sport Summer Olympics by Gender, with less than 3,000 Athletes all tir
ax.set_xlabel("Minor Summer Sport",fontsize = 15, weight='bold')
```

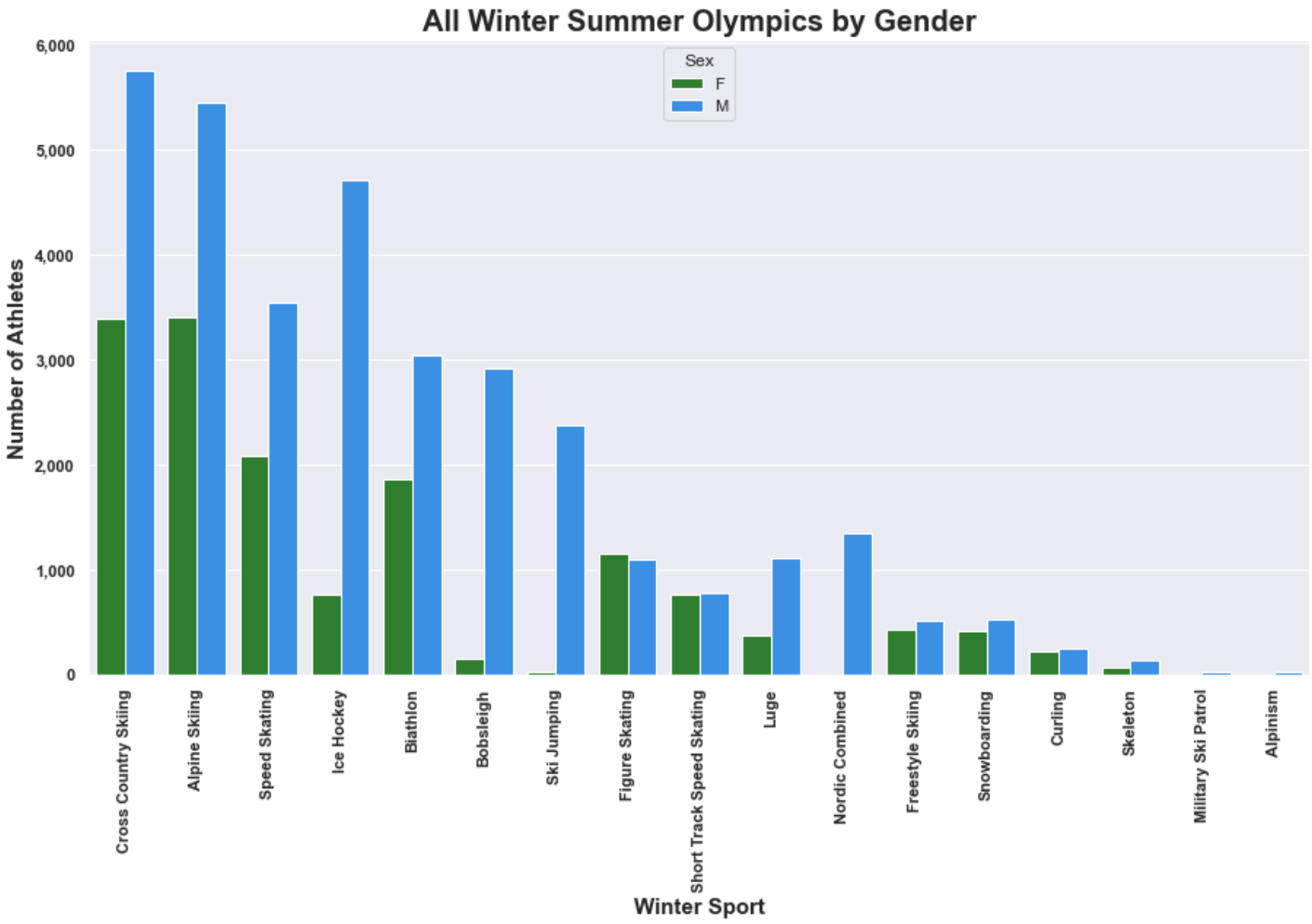
```
ax.set_ylabel('Number of Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
#show_values(ax)
```



2.9 All Winter Olympics Sports by Gender

In [63]:

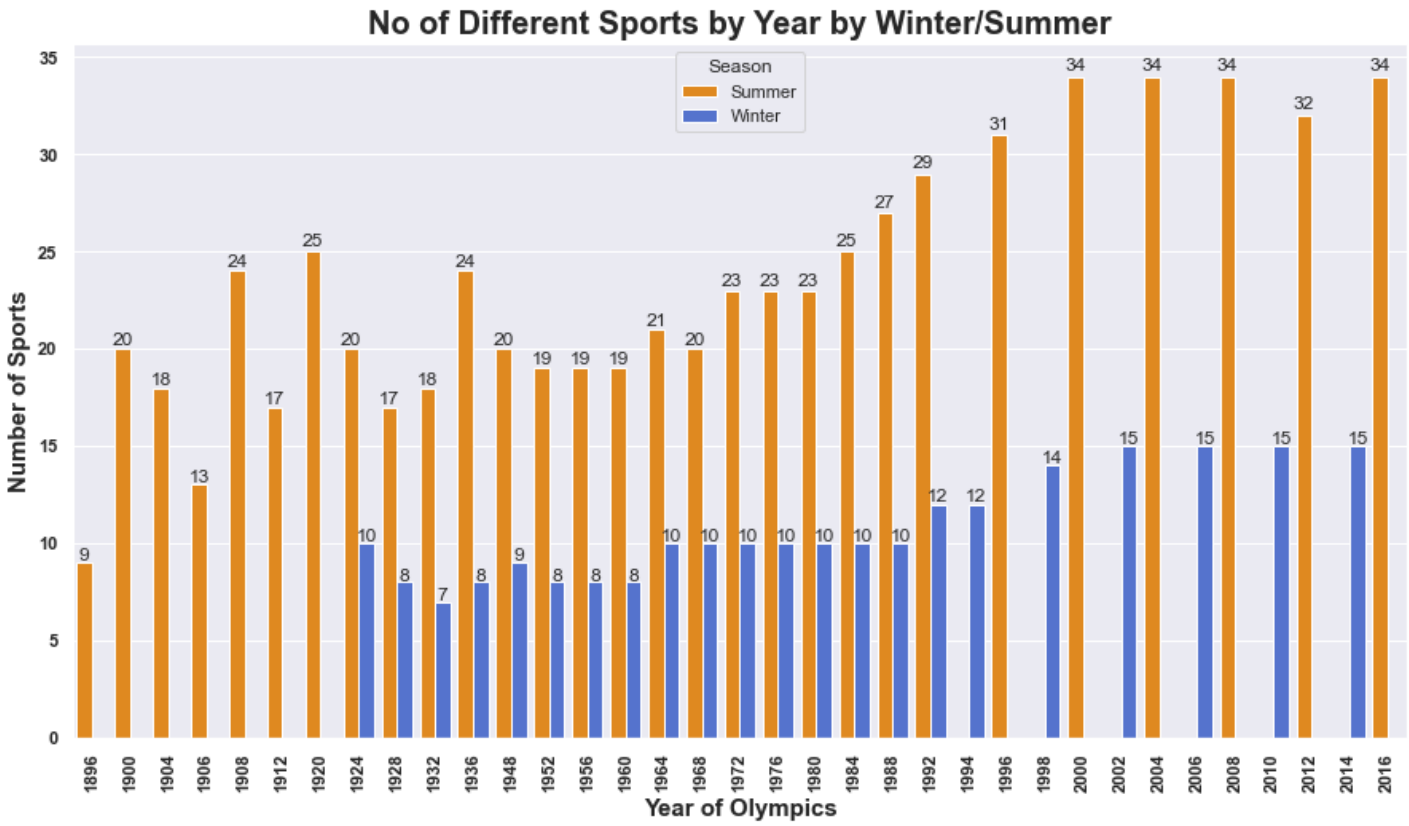
```
ax=sns.countplot(x='Sport',hue = 'Sex', data=winter, palette=['forestgreen', 'dodgerblue']
ax.set_title('All Winter Summer Olympics by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Winter Sport",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Athletes',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc],weight='bold');
#show_values(ax)
```



2.9 No of Unique Sports over the years

In [64]:
```
yearly_sports = pd.read_sql_query("select Year, Season, COUNT(DISTINCT(Sport)) AS NoOfSpor
```

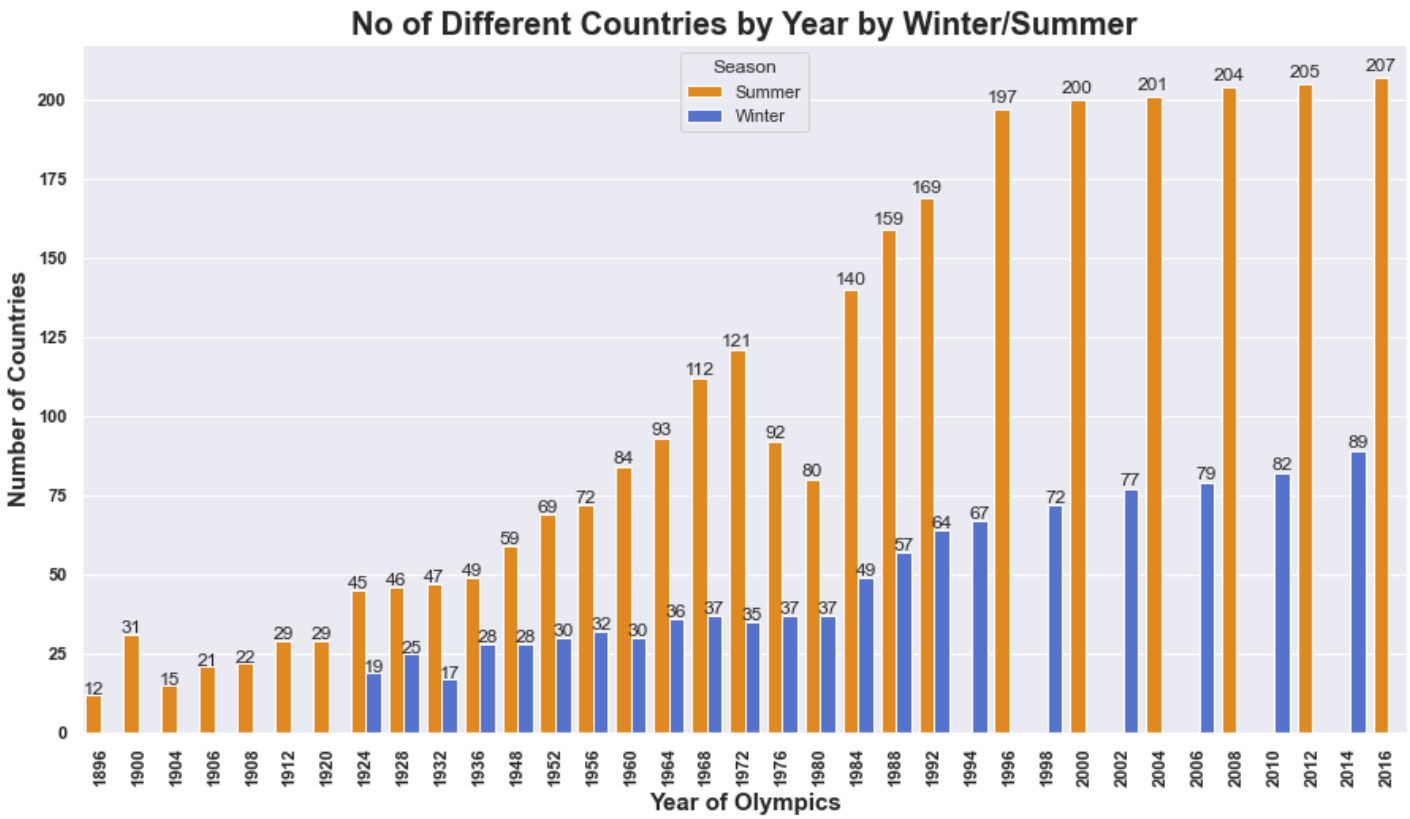
In [65]:
```
ax=sns.barplot(x='Year', y='NoOfSports', hue = 'Season', data=yearly_sports, palette=['dar
#ax=sns.countplot(x='Year', hue = 'Season', data=olympics, palette=['darkorange', 'royalb
ax.set_title('No of Different Sports by Year by Winter/Summer', fontsize = 20,weight='bold
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Sports',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
show_values(ax)
```



2.9 No of Unique Countries participating over the years

In [66]:
```
yearly_countries = pd.read_sql_query("select Year, Season, COUNT(DISTINCT(NOC)) AS NoOfCou
```

In [67]:
```
ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, palette
#ax=sns.countplot(x='Year', hue = 'Season', data=olympics, palette=['darkorange', 'royalb
ax.set_title('No of Different Countries by Year by Winter/Summer', fontsize = 20,weight='b
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Countries',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
show_values(ax)
```



In [68]:
```
#yearly_sports = olympics.groupby(['Year', 'Season'])[['Sport']].nunique().reset_index()
```

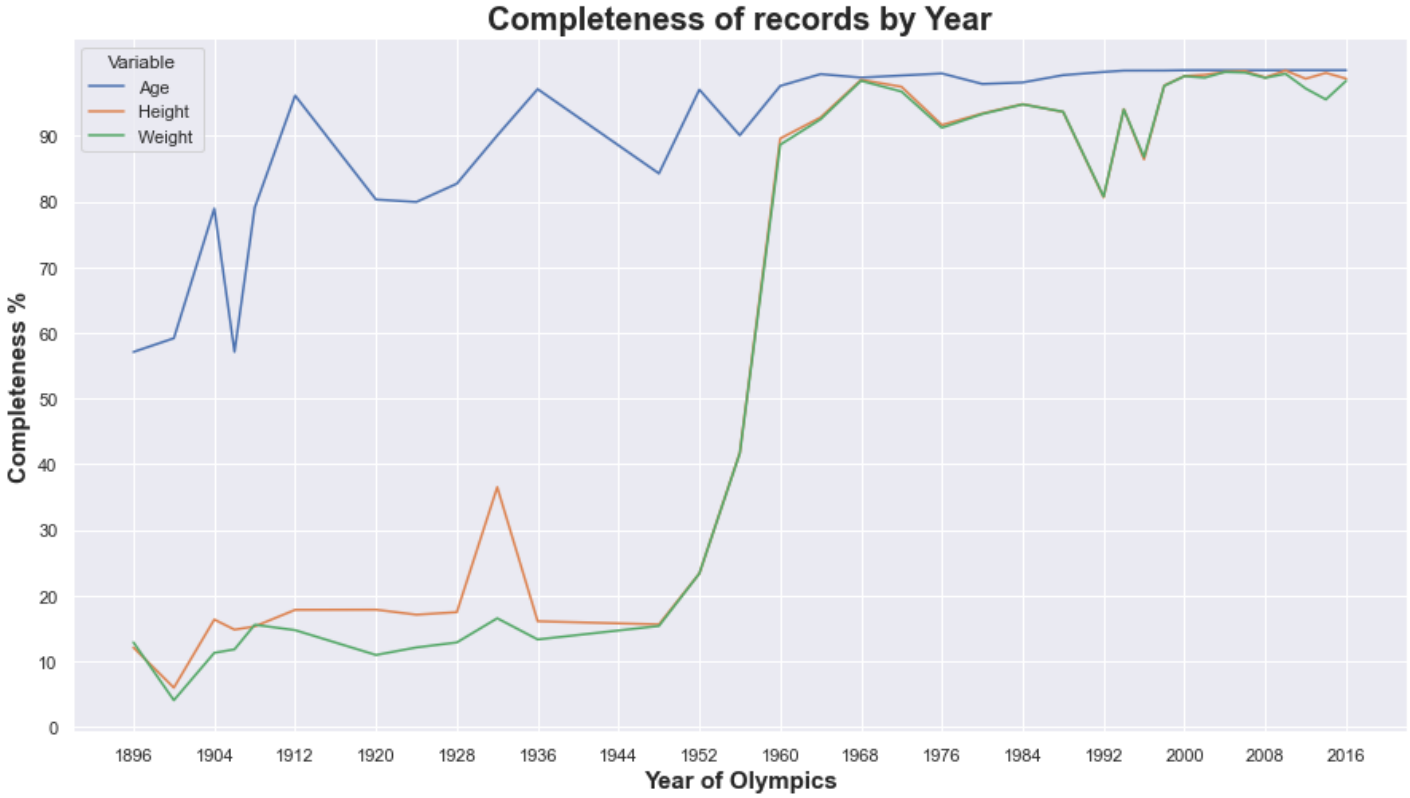
Individuals

Completeness of Individual Data

In [69]:
```
indv_complete = pd.read_sql_query("""
select Year, 'Age ' AS Variable,
    COUNT(*) AS Athletes, COUNT(Age) AS CompleteRecords
    from Olympics
GROUP BY Year
UNION
select Year, 'Weight ' AS Variable,
    COUNT(*) AS Athletes, COUNT(Weight) AS CompleteRecords
    from Olympics
GROUP BY Year
UNION
select Year, 'Height ' AS Variable,
        COUNT(*) AS Athletes, COUNT(Height) AS CompleteRecords
        from Olympics
    GROUP BY Year""", conn)
indv_complete['Completeness'] = 100 * indv_complete['CompleteRecords']/indv_complete['Athl
```

In [70]:
```
#ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, palett
ax=sns.lineplot(x='Year', y='Completeness',hue = 'Variable', data=indv_complete)
ax.set_title('Completeness of records by Year', fontsize = 20,weight='bold')
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Completeness %',fontsize = 15,weight='bold' );
#ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

plt.yticks(list(range(0,100,10)));
plt.xticks(list(range(1896,2020,8)));

#ticks_loc = ax.get_yticks().tolist();
```

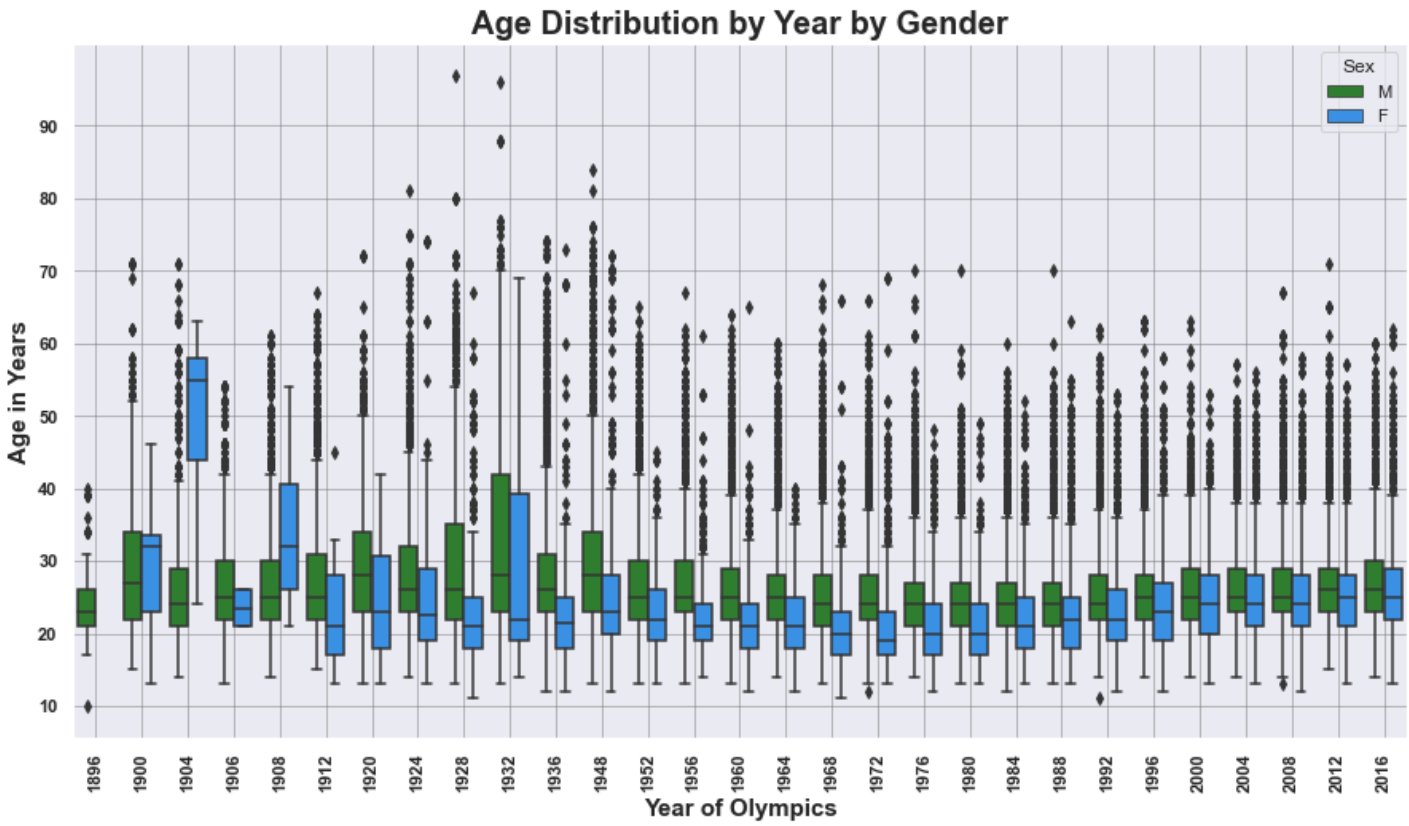
```
#ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
#ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
#show_values(ax)
```



Age Over time

In [71]:

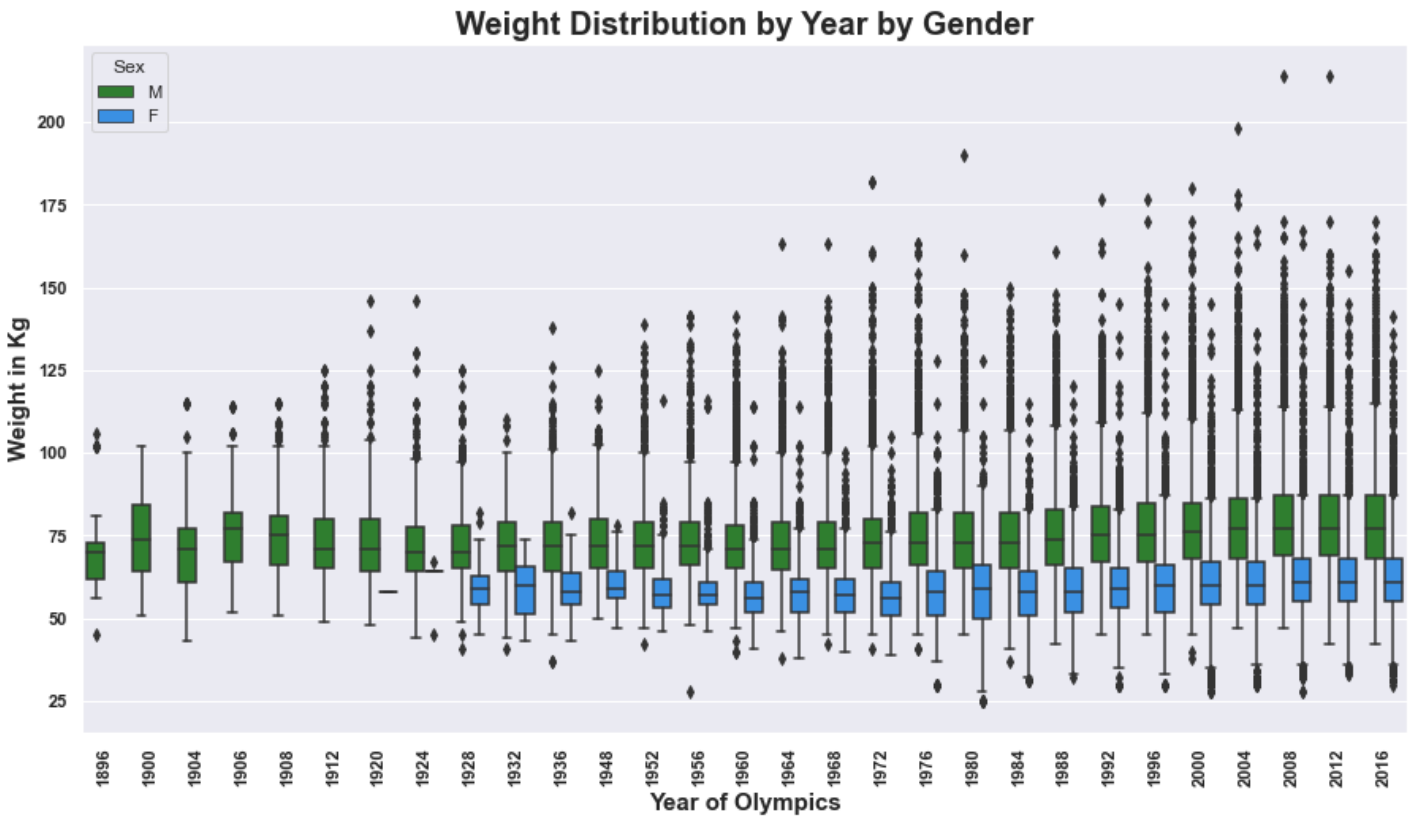
```
#ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, palett
ax=sns.boxplot(x='Year', y='Age',hue = 'Sex', data=summer, palette=['forestgreen', 'dodger
ax.set_title('Age Distribution by Year by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Age in Years',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');
plt.yticks(list(range(10,100,10)))
ax.grid(color='grey', linewidth=0.5)

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
#show_values(ax)
```



Weight Over time

In [72]:

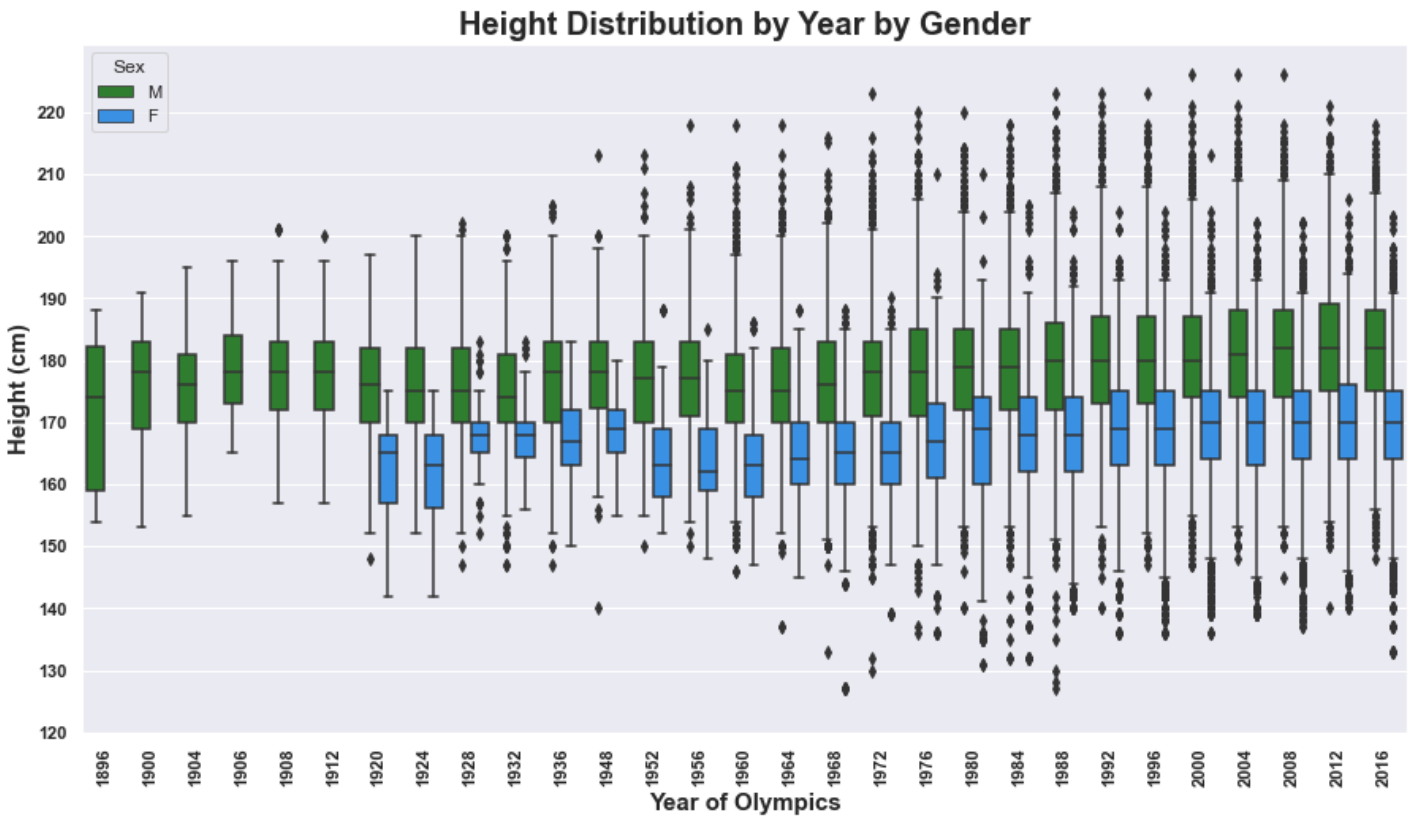
```
#ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, paletu
ax=sns.boxplot(x='Year', y='Weight',hue = 'Sex', data=summer, palette=['forestgreen', 'doc
ax.set_title('Weight Distribution by Year by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Weight in Kg',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
#show_values(ax)
```



Height Over time

In [73]:

```
#ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, paletu
ax=sns.boxplot(x='Year', y='Height',hue = 'Sex', data=summer, palette=['forestgreen', 'doc
ax.set_title('Height Distribution by Year by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Height (cm)',fontsize = 15,weight='bold' )
ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');
plt.yticks(list(range(120,230,10)))

ticks_loc = ax.get_yticks().tolist();
ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
#show_values(ax)
```



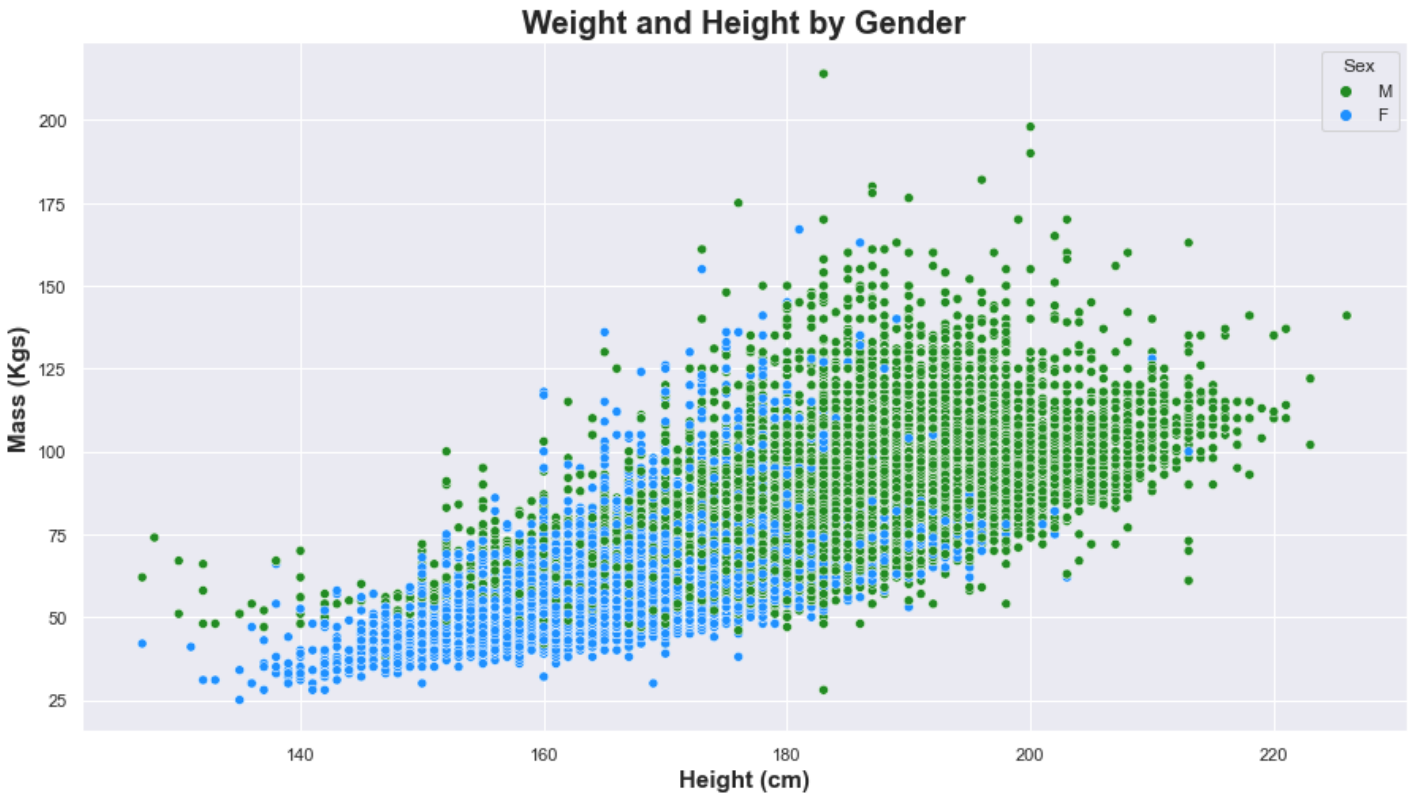
Scatter Plot of Height and Weight by Gender

In [74]:
```
unique_ath = pd.read_sql_query("select * from Olympics GROUP BY Year, ID;", conn)
```

In [75]:
```
unique_ath.loc[unique_ath['Name'].str.contains('usain', case=False)];
```

In [76]:
```
summer.loc[summer['Name'].str.contains('usain', case=False)];
```

In [77]:
```
#ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, palett
ax=sns.scatterplot(x='Height', y='Weight',hue = 'Sex', data=unique_ath, palette=['forestgr
ax.set_title('Weight and Height by Gender', fontsize = 20,weight='bold')
ax.set_xlabel("Height (cm)",fontsize = 15, weight='bold')
ax.set_ylabel('Mass (Kgs)',fontsize = 15,weight='bold' );
#ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

#ticks_loc = ax.get_yticks().tolist();
#ax.yaxis.set_major_locator(mticker.FixedLocator(ticks_loc));
#ax.set_yticklabels(["{:,}".format(int(x)) for x in ticks_loc], weight='bold');
#show_values(ax)
```

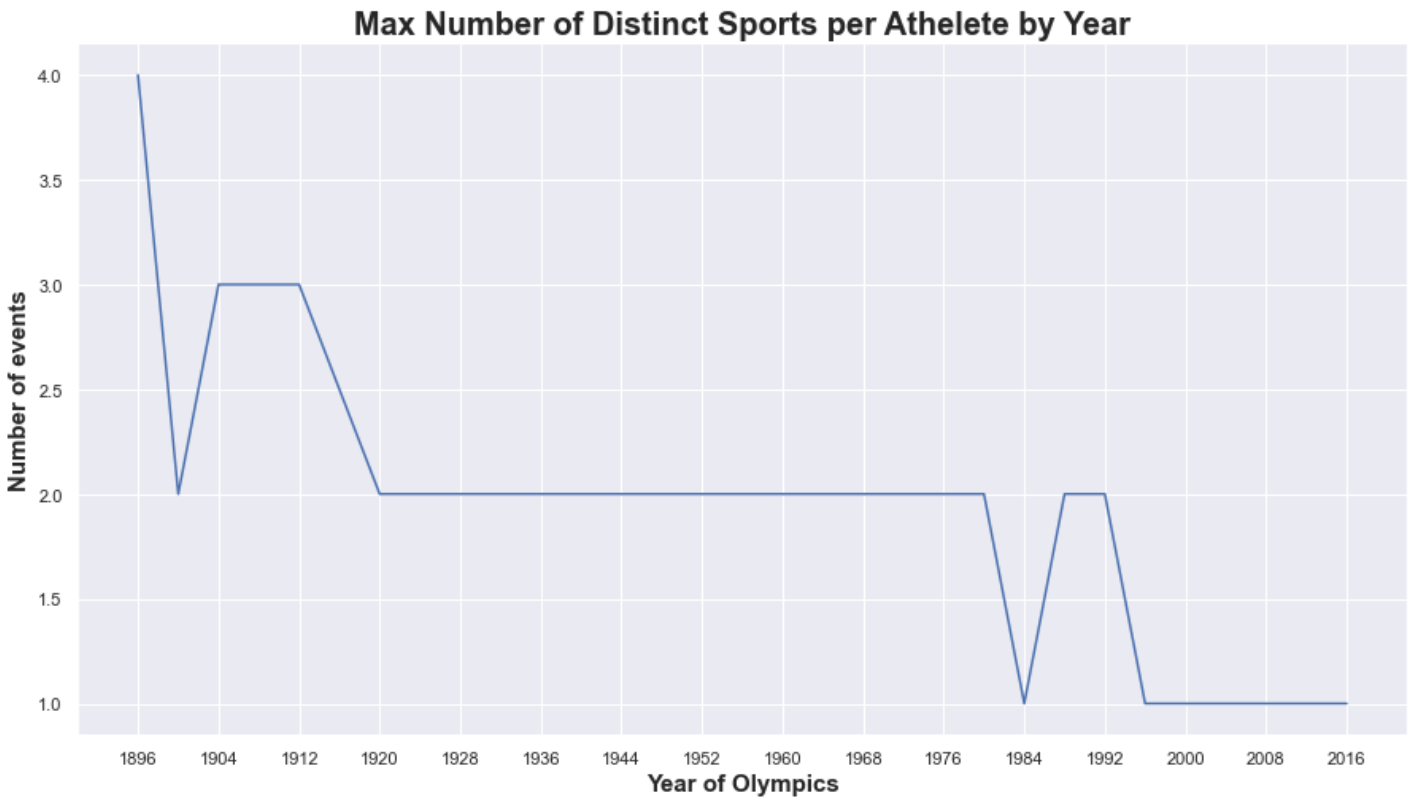


Max Number of Distinct Sports per Athelete Over time

In [78]:
```
max_indiv_sports = pd.read_sql_query("""
select Year, Name, MAX(NoOfInvSports) AS MaxIndSports
    from (select Year, Name, COUNT(DISTINCT(Sport))
            AS NoOfInvSports from Olympics
            WHERE Season = 'Summer'
            GROUP BY Year, ID)
    GROUP BY Year""", conn)
```

In [79]:
```
max_indiv_sports;
```

In [80]:
```
ax=sns.lineplot(x='Year', y='MaxIndSports', data=max_indiv_sports)
ax.set_title('Max Number of Distinct Sports per Athelete by Year', fontsize = 20,weight='b
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Number of Sports',fontsize = 15,weight='bold' );
ax.set_ylabel('Number of events',fontsize = 15,weight='bold' );
plt.xticks(list(range(1896,2020,8)));
#ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');
```



Max Number of Distinct Events per Athelete Over time

In [81]:
```
max_indiv_events = pd.read_sql_query("""
select Year, Name, MAX(NoOfInvEvents) AS MaxIndEvents
    from (select Year, Name, COUNT(DISTINCT(Event))
            AS NoOfInvEvents from Olympics
            GROUP BY Year, ID)
    GROUP BY Year""", conn)
```

In [82]:
```
max_indiv_events
```

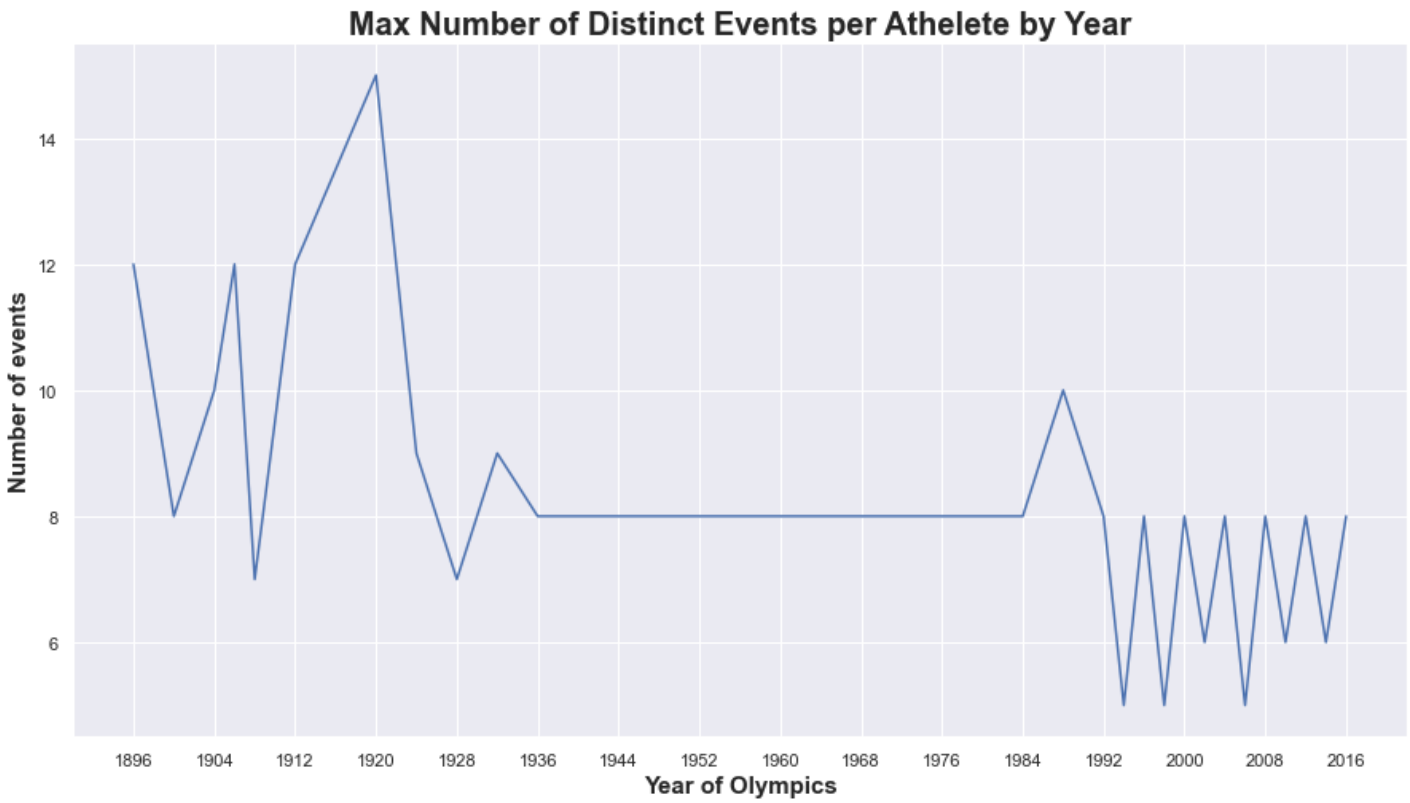
Out[82]:

	Year	Name	MaxIndEvents
0	1896	Carl Schuhmann	12
1	1900	Lon Ernest Moreaux	8
2	1904	George Louis Eyser	10
3	1906	Marie Joseph "Raoul" le Borgne de Boigne	12
4	1908	Gustaf Eric Carlberg	7
5	1912	Ioannis Theofilakis	12
6	1920	Willis Augustus Lee, Jr.	15
7	1924	Harold Brown	9
8	1928	Edvard Antonijevi	7
9	1932	Istvn Pelle	9
10	1936	Andrei Abraham	8
11	1948	Paavo Johannes Aaltonen	8
12	1952	Paavo Johannes Aaltonen	8
13	1956	Nobuyuki Aihara	8

	Year	Name	MaxIndEvents
14	1960	Ismail Abdallah	8
15	1964	Georgi Mirchev Adamov	8
16	1968	Georgi Mirchev Adamov	8
17	1972	Nikolay Yefimovich Andrianov	8
18	1976	Nikolay Yefimovich Andrianov	8
19	1980	Nikolay Yefimovich Andrianov	8
20	1984	Laurent Barbiri	8
21	1988	Mohamed Bin Abid	10
22	1992	Yutaka Aihara	8
23	1994	Kjetil Andr Aamodt	5
24	1996	Aleksandr Gennadyevich Belanovsky	8
25	1998	Thomas Alsgaard (Alsgrd-)	5
26	2000	Maksim Nikolayevich Alyoshin	8
27	2002	Elin Maria Ek	6
28	2004	Alejandro Barrenechea Jayo	8
29	2006	Albina Khamitovna Akhatova	5
30	2008	Thomas Bouhail	8
31	2010	Yelena Vladimirovna Kolomina	6
32	2012	Kim Su-Myeon	8
33	2014	Lowell Conrad Bailey	6
34	2016	Nikita Vladimirovich Nagorny	8

In [83]:
```
ax=sns.lineplot(x='Year', y='MaxIndEvents', data=max_indiv_events)
ax.set_title('Max Number of Distinct Events per Athelete by Year', fontsize = 20,weight='b
ax.set_xlabel("Year of Olympics",fontsize = 15, weight='bold')
ax.set_ylabel('Number of events',fontsize = 15,weight='bold' );
plt.xticks(list(range(1896,2020,8)));
#ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');

'''
for i in range(0,max_indiv_events.shape[0]):
     ax.text(max_indiv_events.Year[i]+0.01, max_indiv_events.MaxIndEvents[i],
     max_indiv_events.Name[i], horizontalalignment='left',
     size='medium', color='black', weight='semibold')
'''
```

Out[83]:
```
"\nfor i in range(0,max_indiv_events.shape[0]):\n     ax.text(max_indiv_events.Year[i]+0.0
1, max_indiv_events.MaxIndEvents[i], \n     max_indiv_events.Name[i], horizontalalignment
='left', \n     size='medium', color='black', weight='semibold')\n"
```



Countries

Attendance Over time Top Countries

In [84]:
```
all_athletes_yearly = %sql select NOC, Year, COUNT(DISTINCT(ID)) AS Athletes from Olympics
```

```
 * sqlite:///Olympics.db
Done.
```

In [85]:
```
#Getting the Data Ready
all_athletes_yearly = pd.read_sql_query("select Year, COUNT(*) AS Athletes from Olympics (
```

In [86]:
```
yu = pd.pivot_table(summer, values = 'ID', index=['Country'], columns = 'Year', aggfunc=la
yu = yu.fillna(0)
```

In [87]:
```
yu.columns[1:]
```

Out[87]:
```
Index([1896, 1900, 1904, 1906, 1908, 1912, 1920, 1924, 1928, 1932, 1936, 1948,
       1952, 1956, 1960, 1964, 1968, 1972, 1976, 1980, 1984, 1988, 1992, 1996,
       2000, 2004, 2008, 2012, 2016],
      dtype='object', name='Year')
```

In [88]:
```
yu['Total'] = yu[yu.columns[1:]].sum(axis=1)
```

In [89]:
```
yu = yu.sort_values(by=['Total'], ascending=False)
yu
```

Out[89]:

Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	...	1984	1988	1992	1996	2000	2(
194	USA	14.0	75.0	524.0	38.0	122.0	174.0	288.0	299.0	280.0	...	522.0	527.0	545.0	648.0	586.0	53
67	Germany	19.0	76.0	22.0	49.0	82.0	185.0	0.0	0.0	295.0	...	390.0	606.0	463.0	466.0	422.0	44

Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	...	1984	1988	1992	1996	2000	20
193	UK	10.0	104.0	6.0	48.0	735.0	274.0	234.0	267.0	232.0	...	337.0	345.0	371.0	300.0	310.0	26
63	France	12.0	720.0	1.0	56.0	208.0	119.0	304.0	401.0	255.0	...	238.0	266.0	339.0	299.0	336.0	30
152	Russia	0.0	4.0	0.0	0.0	6.0	159.0	0.0	3.0	0.0	...	0.0	481.0	475.0	390.0	435.0	44
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
185	Timor-Leste	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
95	Kosovo	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
192	Tuvalu	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
172	South Sudan	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
198	Unknown	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

209 rows × 31 columns

In [90]:
```
yu.head(15)
```

Out[90]:

Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	...	1984	1988	1992	1996	2000
194	USA	14.0	75.0	524.0	38.0	122.0	174.0	288.0	299.0	280.0	...	522.0	527.0	545.0	648.0	586.0
67	Germany	19.0	76.0	22.0	49.0	82.0	185.0	0.0	0.0	295.0	...	390.0	606.0	463.0	466.0	422.0
193	UK	10.0	104.0	6.0	48.0	735.0	274.0	234.0	267.0	232.0	...	337.0	345.0	371.0	300.0	310.0
63	France	12.0	720.0	1.0	56.0	208.0	119.0	304.0	401.0	255.0	...	238.0	266.0	339.0	299.0	336.0
152	Russia	0.0	4.0	0.0	0.0	6.0	159.0	0.0	3.0	0.0	...	0.0	481.0	475.0	390.0	435.0
87	Italy	1.0	23.0	1.0	76.0	66.0	66.0	174.0	200.0	174.0	...	268.0	253.0	304.0	340.0	361.0
10	Australia	1.0	2.0	2.0	4.0	30.0	25.0	13.0	36.0	18.0	...	242.0	252.0	279.0	417.0	617.0
33	Canada	0.0	4.0	57.0	3.0	87.0	37.0	52.0	65.0	69.0	...	408.0	328.0	295.0	303.0	294.0
90	Japan	0.0	0.0	0.0	0.0	0.0	2.0	15.0	19.0	40.0	...	226.0	255.0	256.0	306.0	266.0
178	Sweden	1.0	10.0	0.0	39.0	168.0	444.0	260.0	159.0	100.0	...	174.0	185.0	187.0	177.0	150.0
78	Hungary	7.0	18.0	4.0	35.0	65.0	121.0	0.0	89.0	109.0	...	0.0	188.0	217.0	213.0	178.0
129	Netherlands	0.0	35.0	0.0	16.0	113.0	33.0	130.0	177.0	266.0	...	136.0	147.0	201.0	239.0	231.0
173	Spain	0.0	9.0	0.0	0.0	0.0	0.0	58.0	95.0	80.0	...	179.0	229.0	422.0	289.0	321.0
39	China	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	262.0	321.0	282.0	317.0	302.0
145	Poland	0.0	0.0	0.0	0.0	0.0	1.0	0.0	65.0	93.0	...	0.0	143.0	201.0	165.0	187.0

15 rows × 31 columns

In [91]:
```
yu.columns
```

Out[91]:
```
Index(['Country',       1896,       1900,       1904,       1906,       1908,
            1912,       1920,       1924,       1928,       1932,       1936,
            1948,       1952,       1956,       1960,       1964,       1968,
            1972,       1976,       1980,       1984,       1988,       1992,
            1996,       2000,       2004,       2008,       2012,       2016,
```

```
       'Total'],
      dtype='object', name='Year')
```

In [92]:
```
yu[yu['Country'].isin(['Uganda', 'Kenya', 'Tanzania', 'Rwanda'])]
```

Out[92]:

	Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	...	1984	1988	1992	1996	2000	2004
93	Kenya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	61.0	74.0	49.0	52.0	56.0	46.0
195	Uganda	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	26.0	24.0	8.0	10.0	13.0	11.0
183	Tanzania	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	18.0	10.0	9.0	7.0	4.0	8.0
153	Rwanda	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3.0	6.0	10.0	4.0	5.0	5.0

4 rows × 31 columns

In [93]:
```
ug = yu[yu['Country']=='Uganda']
```

In [94]:
```
ug
```

Out[94]:

	Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	...	1984	1988	1992	1996	2000	2004
195	Uganda	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	26.0	24.0	8.0	10.0	13.0	11.0

1 rows × 31 columns

In [95]:
```
ug[1896]
```

Out[95]:
```
195    0.0
Name: 1896, dtype: float64
```

In [96]:
```
y=ug.columns
```

In [97]:
```
y[1:]
```

Out[97]:
```
Index([   1896,    1900,    1904,    1906,    1908,    1912,    1920,    1924,
          1928,    1932,    1936,    1948,    1952,    1956,    1960,    1964,
          1968,    1972,    1976,    1980,    1984,    1988,    1992,    1996,
          2000,    2004,    2008,    2012,    2016, 'Total'],
      dtype='object', name='Year')
```

In [98]:
```
(ug[ug.columns[1:]]>0).idxmax(axis=1)
```

Out[98]:
```
195    1956
dtype: int64
```

In [99]:
```
list(ug.columns).index(1956)
```

Out[99]:
```
14
```

In [100…
```
t=ug[ug.columns[14:-1]]
```

In [101…
```
list(ug.columns[14:-1])
```

Out[101…
```
[1956,
 1960,
 1964,
 1968,
 1972,
 1976,
 1980,
 1984,
 1988,
 1992,
 1996,
 2000,
 2004,
 2008,
 2012,
 2016]
```

In [102…
```
len(t.values.tolist()[0])
```

Out[102…
```
16
```

In [103…
```
len(list(t.values))
```

Out[103…
```
1
```

In [104…
```
ug2  =pd.DataFrame({'Years':list(ug.columns[14:-1]), 'Athletes':t.values.tolist()[0]})
```

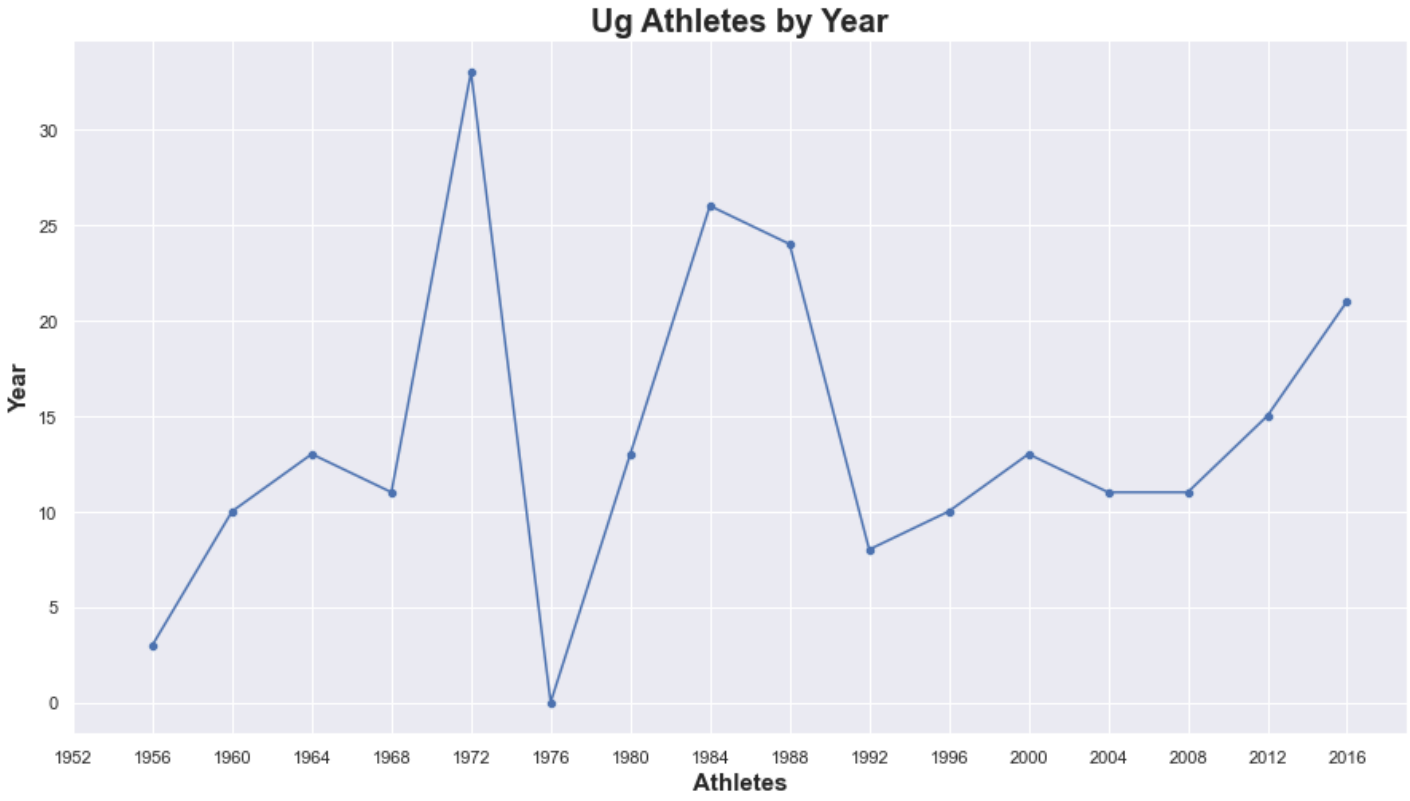
In [105…
```
ug2
```

Out[105…

	Years	Athletes
0	1956	3.0
1	1960	10.0
2	1964	13.0
3	1968	11.0
4	1972	33.0
5	1976	0.0
6	1980	13.0
7	1984	26.0
8	1988	24.0
9	1992	8.0
10	1996	10.0
11	2000	13.0
12	2004	11.0
13	2008	11.0
14	2012	15.0
15	2016	21.0

In [106…

```
#ax=sns.barplot(x='Year', y='NoOfCountries', hue = 'Season', data=yearly_countries, palett
ax=sns.scatterplot(x='Years', y='Athletes', data=ug2)
ax=sns.lineplot(x='Years', y='Athletes', data=ug2)
ax.set_title('Ug Athletes by Year', fontsize = 20,weight='bold')
ax.set_xlabel("Athletes",fontsize = 15, weight='bold')
ax.set_ylabel('Year',fontsize = 15,weight='bold' );
plt.xticks(list(range(1952,2020,4)));
#ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,weight='bold');
```



In [107…

```
scipy.stats.pearsonr(ug2['Years'], ug2['Athletes'])
```

Out[107…

```
(0.17078771128025388, 0.5271198764497287)
```

In [108…

```
ke = yu[yu['Country']=='Kenya']
```

In [109…

```
ke
```

Out[109…

	Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	…	1984	1988	1992	1996	2000	2004
93		Kenya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	61.0	74.0	49.0	52.0	56.0	46.0

1 rows × 31 columns

In [110…

```
yu.tail(20)
```

Out[110…

	Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	…	1984	1988	1992	1996	2000	20
118		Micronesia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	5.0	5
168		Solomon Islands	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	3.0	4.0	1.0	4.0	2.0	2

Year	Country	1896	1900	1904	1906	1908	1912	1920	1924	1928	...	1984	1988	1992	1996	2000	200
155	Saint Lucia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	5.0	2
138	Palau	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	5.0	4
139	Palestine	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	2.0	3
74	Guinea-Bissau	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	3.0	3
41	Comoros	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	4.0	2.0	3
34	Cape Verde	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	2.0	3
52	Dominica	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	4.0	2
114	Marshall Islands	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
159	Sao Tome and Principe	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	2.0	2.0	2
127	Nauru	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	2.0	3
94	Kiribati	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	3
27	Brunei	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	2.0	1
149	Refugee Olympic Team	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
185	Timor-Leste	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2
95	Kosovo	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
192	Tuvalu	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
172	South Sudan	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
198	Unknown	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0

20 rows × 31 columns

In [ ]:

Attendance Over time Improving Countries

Attendance Over time Declining Countries

Total Medals Over time Top Countries

Total Gold Medals Over time Top Countries

Total Silver Medals Over time Top Countries

Total Bronze Medals Over time Top Countries

Total Medals per Athelte, Top 10 Countries

Medals Over time Improving Countries

Attendance Over time Declining Countries

Max Number of Distinct Events per Athelete Over time

Max Number of Distinct Events per Athelete Over time

Sports Performance

Attendance Over time Top Sports

Attendance Over time Improving Sports

Attendance Over time Declining Sports

Hypothesis

In [111…

```
(pd.crosstab(olympics['Year'], olympics['City'])>0).astype(int).sum()
```

Out[111…

```
City
Albertville                1
Amsterdam                  1
Antwerpen                  1
Athina                     3
Atlanta                    1
Barcelona                  1
Beijing                    1
Berlin                     1
Calgary                    1
Chamonix                   1
Cortina d'Ampezzo          1
Garmisch-Partenkirchen     1
Grenoble                   1
Helsinki                   1
Innsbruck                  2
Lake Placid                2
Lillehammer                1
London                     3
Los Angeles                2
Melbourne                  1
Mexico City                1
Montreal                   1
Moskva                     1
Munich                     1
Nagano                     1
Oslo                       1
Paris                      2
Rio de Janeiro             1
Roma                       1
Salt Lake City             1
Sankt Moritz               2
Sapporo                    1
Sarajevo                   1
Seoul                      1
Sochi                      1
Squaw Valley               1
St. Louis                  1
Stockholm                  2
Sydney                     1
Tokyo                      1
Torino                     1
Vancouver                  1
dtype: int64
```

In [112…

```
city_hosts = olympics.groupby(['City','Season','Year'])[['ID']].count()
```

In [113…
```
olympics['Country'].value_counts()
```

Out[113…
```
USA            18853
Germany        15883
France         12758
UK             12256
Russia         11692
               ...
Timor-Leste        9
Kosovo             8
Tuvalu             7
South Sudan        3
Unknown            2
Name: Country, Length: 209, dtype: int64
```

In [114…
```
#df = pd.read_sql_query("select * from Olympics LIMIT 15;", conn)

df = %sql select * from Olympics LIMIT 15;

#print the dataframe
#df
```

```
 * sqlite:///Olympics.db
Done.
```

1.3.6 Adding New Variables

Will add EventScore, Gold ->10, Silver ->7, Bronze->5, or NA->1 for qualifying.

Will add medal binary, Gold ->1, Silver ->1, Bronze->1, or NA-0

Rename region to Country

In [115…
```
#Gold 10 points
olympics.loc[olympics['Medal']=='Gold', 'EventScore'] = 10
#Silver 7 points
olympics.loc[olympics['Medal']=='Silver', 'EventScore'] = 7.5
#Bronze 4 points
olympics.loc[olympics['Medal']=='Bronze', 'EventScore'] = 5
#Qualification and attendance 1 points
olympics.loc[olympics['Medal'].isnull(), 'EventScore'] = 1
#olympics["region"]=np.where(olympics["region"].isnull(), olympics["notes"],olympics["regi
```

In [116…
```
olympics['Medal'].value_counts()
```

Out[116…
```
Gold      13372
Bronze    13295
Silver    13116
Name: Medal, dtype: int64
```

In [117…
```
olympics['EventScore'].value_counts()
```

Out[117…
```
1.0     231333
10.0     13372
5.0      13295
7.5      13116
Name: EventScore, dtype: int64
```

In [118…
```
olympics.columns
```

```
Index(['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight', 'Team', 'NOC', 'Games',
```

Out[118…
```
       'Year', 'Season', 'City', 'Sport', 'Event', 'Medal', 'NOC', 'Country',
       'notes', 'EventScore'],
      dtype='object')
```

In [119…
```
#olympics.rename(columns = {'region':'Country'}, inplace = True)
```

In [120…
```
olympics.columns
```

Out[120…
```
Index(['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight', 'Team', 'NOC', 'Games',
       'Year', 'Season', 'City', 'Sport', 'Event', 'Medal', 'NOC', 'Country',
       'notes', 'EventScore'],
      dtype='object')
```

In [121…
```
# Close the connection to the SQLite Database
conn.close()
print("SQLite Database Connection Closed Sucessfully")
```

```
SQLite Database Connection Closed Sucessfully
```

In [122…
```
print(style.GREEN + style.BOLD,"\n",datetime.datetime.now() - startTime, "Time running\n")
```

0:01:08.217844 Time running