TITLE: OLYMPIC DATA ANALYSIS

by

NAME: AYUSH SINGH **REGISTER NUMBER:** 19BCE1813
AAYUSH KUMAR SINGH 19BCE1113
ROHIT ARVIND MISHRA 19BCE1049

A project report submitted to

Dr. Vijaykumar K P

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

in partial fulfilment of the requirements for the course of

CSE3020 – Data Visualization

in

B. Tech. COMPUTER SCIENCE AND ENGINEERING

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

DECEMBER 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**OLYMPIC DATA ANALYSIS"** is a bonafide work of **AYUSH SINGH 19BCE1813, AAYUSH KUMAR SINGH 19BCE1113 AND ROHIT ARVIND MISHRA 19BCE1049** who carried out the Project work under my supervision and guidance for **CSE3020- DATA VISUALIZATION.**

Dr. VIJAYKUMAR KP

Associate Professor (Sr)

School of Computer Science and Engineering (SCOPE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

The Sportspersons from various countries participate in competitions and make their countries proud of their excellence in sports. Despite massive population, many most populous countries fail to grab many medals at the Olympic games. The primary objective of this project to analyses the Olympic dataset using python to compare overall performance of countries and to evaluate the contribution of each country in Olympics. These analyses will give deeper insight into the performance of countries in Olympics over the years and helps sportspersons to quickly analyses their own and the competitor's performance.

INTRODUCTION

Olympics is considered as most important event worldwide, which provides common platform to players from various nations to show their talents. Olympics has been started at 1896, which is being conducted once in every four years. The goal of this project is to analyze performance and participation of nations in Olympics. In addition, the field of sports of particular country in particular year, in which they have contributed the maximum can be identified. The comparison of the performance of each sport with other can be done. The field of sports that has to have more participation can be identified and necessary action can be taken by players and nations to enhance themselves in future contributions towards Olympics.

PROBLEM STATEMENT

The Olympic Games, considered to be the world's foremost sports competition has more than 200 nations participating across the Summer and Winter Games alternating by occurring every four years but two years apart. Throughout this project, we will explore the Olympics dataset, look at some interesting statistics and then try to find out which country is the King of the Olympic Games

LITERATURE REVIEW

The advantage of host country in any sporting activity is well known, as the participants will have familiarity of the field, and also there is a great support from the home crowd. Host countries are expected to win 3 times the medals that they were winning while playing as away (Clarke, 2000).

- Age factor is also one of the important ones when it comes to sports and even among the athletes of the same age, relative age effect (RAE) comes into factor which determines who triumphs (Fletcher & Sarkar, 2012).

- Being a host nation and also having a communist background is also going to have a positive effect in the number of medals won (Bian, 2005).

- RAE states that an athlete can have more advantage as compared to another who is younger by almost a year with respect to maturity, experience and early specialization (Neill, Cotton, Cuadros & Connor, 2016)

PROPOSED SYSTEM

The Olympics dataset had the names of participants, their demographics, which sport they participated in and on which Olympic games.

- A custom dataset was also created that maps the cities mentioned in the Olympics dataset to country names.
- Visualizations were created in Tableau and Python. Initial data cleaning was done on Excel, and visualization specific data manipulation were carried out as needed on Tableau and Python.

Data preparation and cleaning

Data Analyst spend most of their time in preparing and cleaning the data. This normally includes below activities

1. Load the dataset into a data frame using Pandas.
2. Explore the number of rows & columns, ranges of values etc.
3. Format data (text to number conversion, removing additional formatting such as ',' from numbers, currency symbol, date format etc.)
4. Handle missing, incorrect and invalid data (either remove it or fill it with appropriate data)
5. Perform any additional steps (parsing dates, creating additional columns, merging multiple datasets etc.

Exploratory analysis and visualization

After the data is checked, prepared and cleaned, we should visualize the data. Picture speaks more than words and hence a good visualization can help data analysts to present their analysis without using much write-up. Normally, data visualization can be used to highlight below:

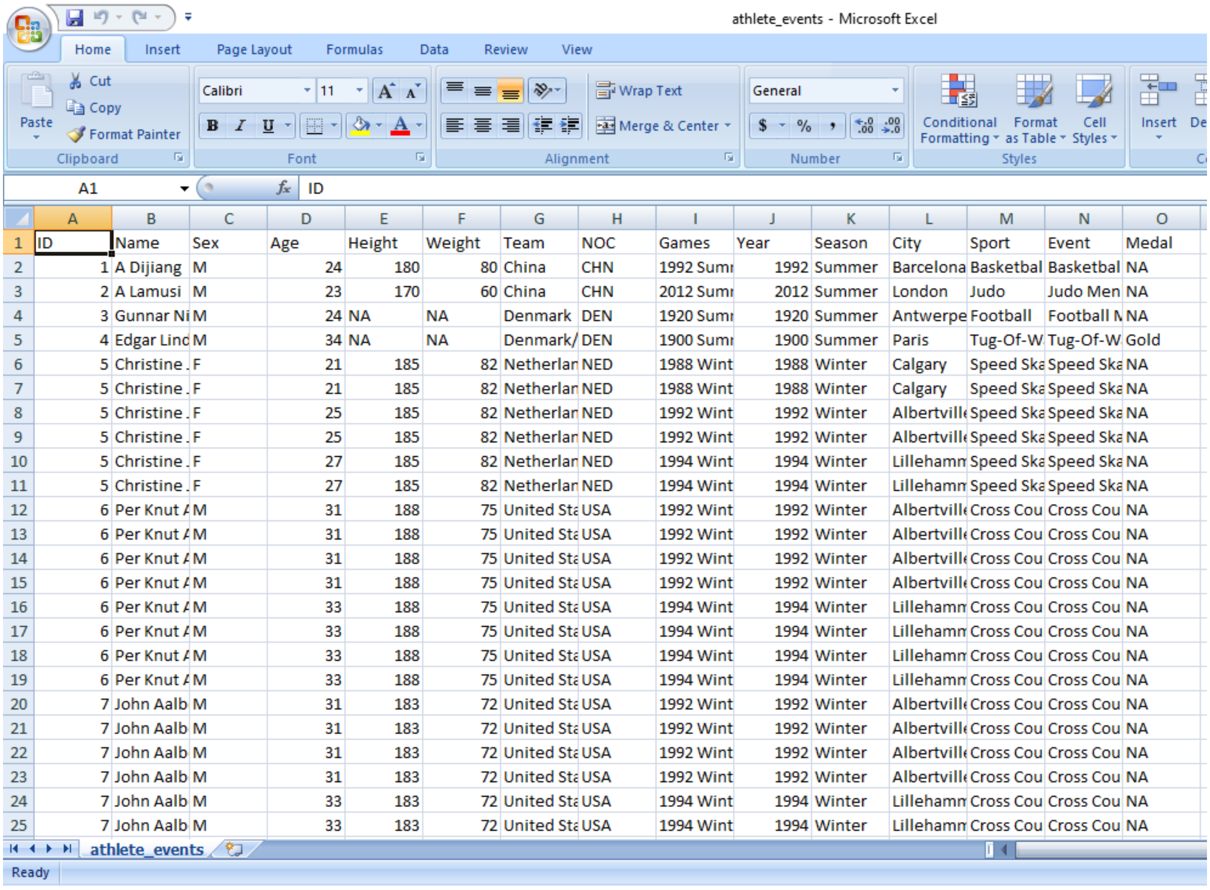
1.Statistics such as mean, sum, range and other interesting parameters for numeric columns
2.Explore distributions of numeric columns using histograms, distribution etc.
3.Explore relationship between columns using scatter plots, bar charts etc.
4.Make a note of interesting insights from the exploratory analysis.

Asking and answering questions

In this section, we will be asking few questions to imitate real life scenario. Various Python packages/functions may be used to answer the questions.

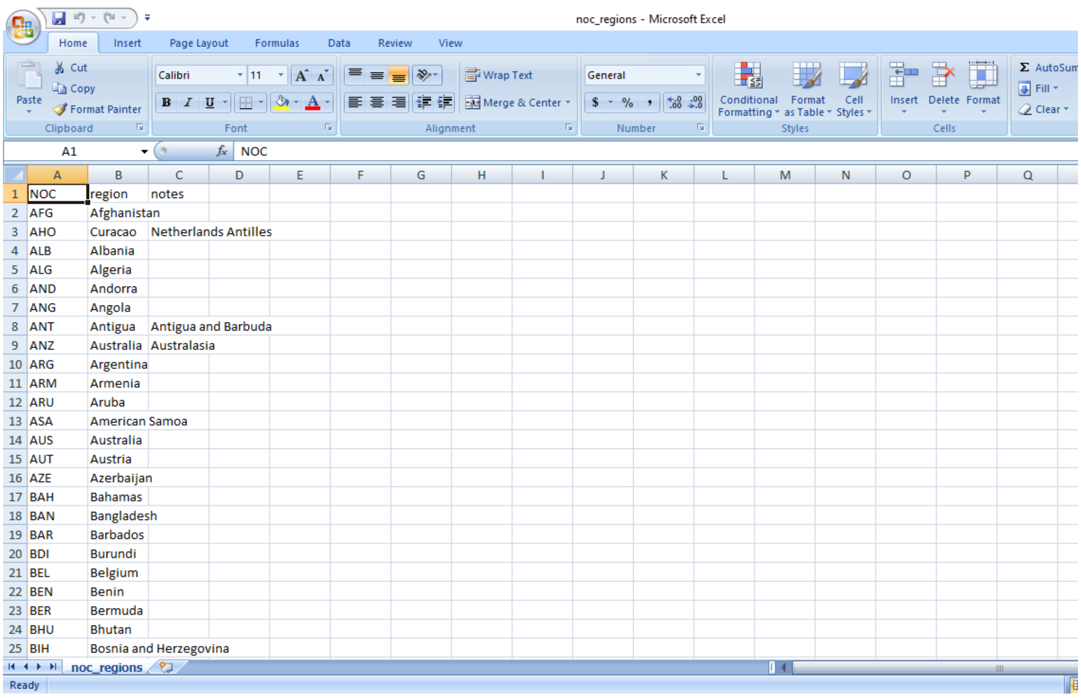
Tools used

- Jupyter Notebook
- Tableau
- Plotly: The front end for ML and data science models

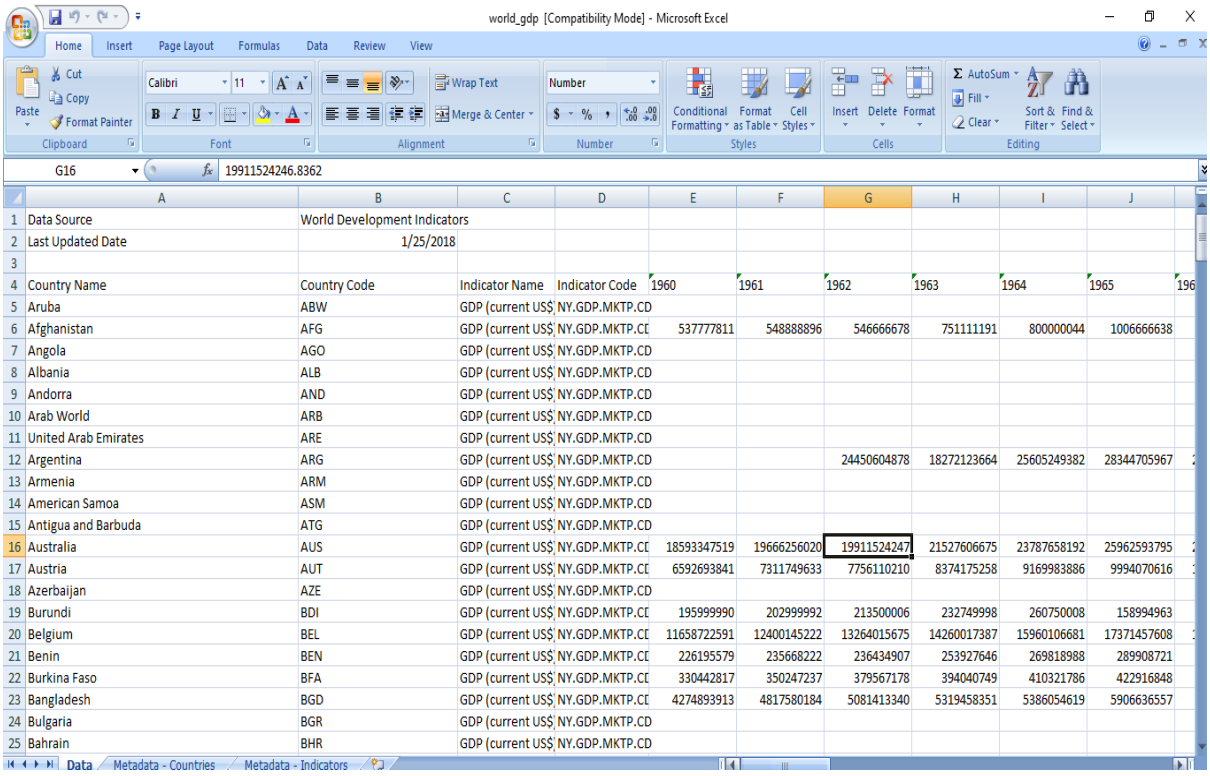


	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
2	1	A Dijiang	M	24	180	80	China	CHN	1992 Sumı	1992	Summer	Barcelona	Basketbal	Basketbal	NA
3	2	A Lamusi	M	23	170	60	China	CHN	2012 Sumı	2012	Summer	London	Judo	Judo Men	NA
4	3	Gunnar Ni	M	24	NA	NA	Denmark	DEN	1920 Sumı	1920	Summer	Antwerpe	Football	Football N	NA
5	4	Edgar Linc	M	34	NA	NA	Denmark/	DEN	1900 Sumı	1900	Summer	Paris	Tug-Of-W	Tug-Of-W	Gold
6	5	Christine .	F	21	185	82	Netherlar	NED	1988 Wint	1988	Winter	Calgary	Speed Ska	Speed Ska	NA
7	5	Christine .	F	21	185	82	Netherlar	NED	1988 Wint	1988	Winter	Calgary	Speed Ska	Speed Ska	NA
8	5	Christine .	F	25	185	82	Netherlar	NED	1992 Wint	1992	Winter	Albertville	Speed Ska	Speed Ska	NA
9	5	Christine .	F	25	185	82	Netherlar	NED	1992 Wint	1992	Winter	Albertville	Speed Ska	Speed Ska	NA
10	5	Christine .	F	27	185	82	Netherlar	NED	1994 Wint	1994	Winter	Lillehamn	Speed Ska	Speed Ska	NA
11	5	Christine .	F	27	185	82	Netherlar	NED	1994 Wint	1994	Winter	Lillehamn	Speed Ska	Speed Ska	NA
12	6	Per Knut /	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
13	6	Per Knut /	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
14	6	Per Knut /	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
15	6	Per Knut /	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
16	6	Per Knut /	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Cou	Cross Cou	NA
17	6	Per Knut /	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Cou	Cross Cou	NA
18	6	Per Knut /	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Cou	Cross Cou	NA
19	6	Per Knut /	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Cou	Cross Cou	NA
20	7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
21	7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
22	7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
23	7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertville	Cross Cou	Cross Cou	NA
24	7	John Aalb	M	33	183	72	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Cou	Cross Cou	NA
25	7	John Aalb	M	33	183	72	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Cou	Cross Cou	NA

It includes the testing dataset which has been trained multiple times and is ready to be tested alongside all other athlete to give us an accurate result and helps us to find out the top five countries which won the most medals in the Olympics.



	NOC	region	notes
2	AFG	Afghanistan	
3	AHO	Curacao	Netherlands Antilles
4	ALB	Albania	
5	ALG	Algeria	
6	AND	Andorra	
7	ANG	Angola	
8	ANT	Antigua	Antigua and Barbuda
9	ANZ	Australia	Australasia
10	ARG	Argentina	
11	ARM	Armenia	
12	ARU	Aruba	
13	ASA	American Samoa	
14	AUS	Australia	
15	AUT	Austria	
16	AZE	Azerbaijan	
17	BAH	Bahamas	
18	BAN	Bangladesh	
19	BAR	Barbados	
20	BDI	Burundi	
21	BEL	Belgium	
22	BEN	Benin	
23	BER	Bermuda	
24	BHU	Bhutan	
25	BIH	Bosnia and Herzegovina	



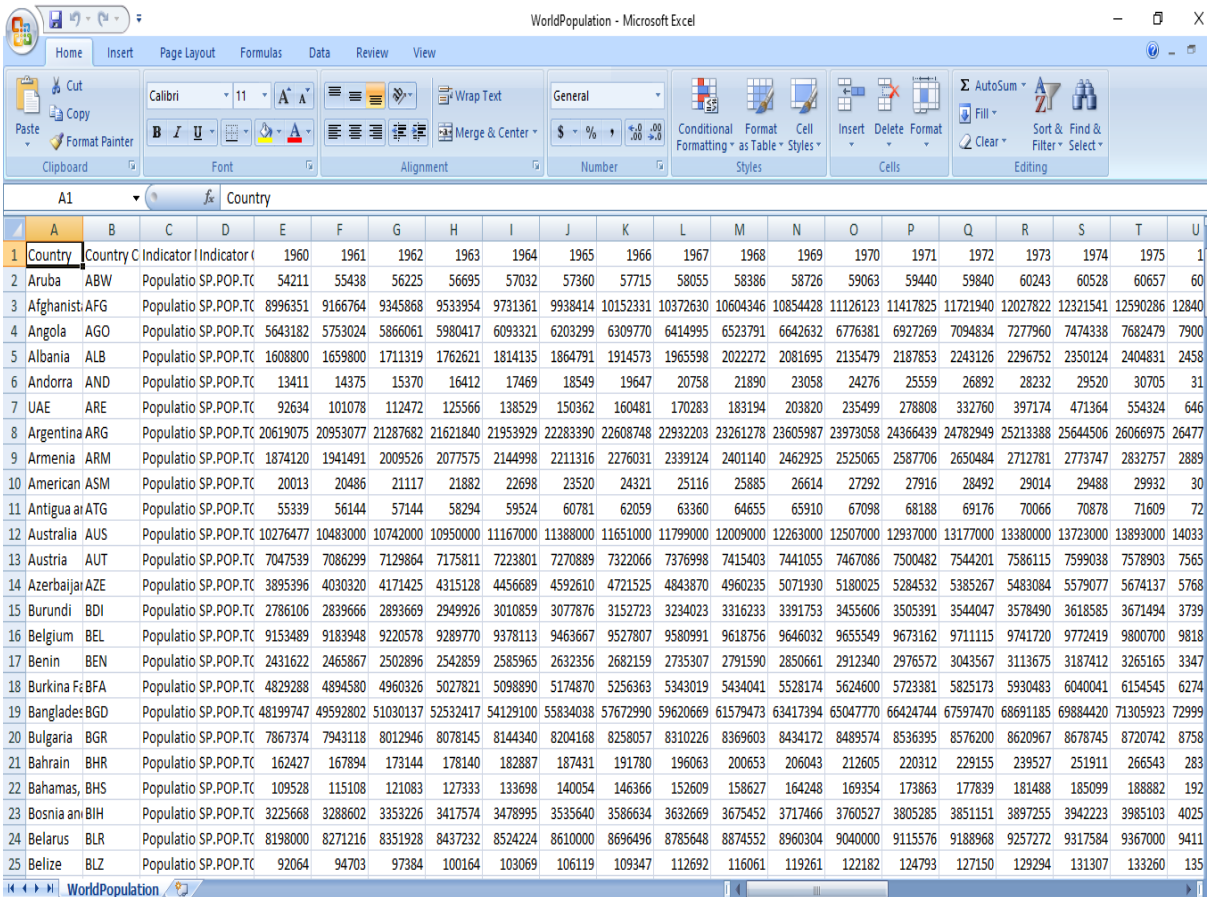
world_gdp [Compatibility Mode] - Microsoft Excel

Home | Insert | Page Layout | Formulas | Data | Review | View

G16: 19911524246.8362

	A	B	C	D	E	F	G	H	I	J
1	Data Source	World Development Indicators								
2	Last Updated Date	1/25/2018								
3										
4	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965
5	Aruba	ABW	GDP (current US$)	NY.GDP.MKTP.CD						
6	Afghanistan	AFG	GDP (current US$)	NY.GDP.MKTP.CD	537777811	548888896	546666678	751111191	800000044	1006666638
7	Angola	AGO	GDP (current US$)	NY.GDP.MKTP.CD						
8	Albania	ALB	GDP (current US$)	NY.GDP.MKTP.CD						
9	Andorra	AND	GDP (current US$)	NY.GDP.MKTP.CD						
10	Arab World	ARB	GDP (current US$)	NY.GDP.MKTP.CD						
11	United Arab Emirates	ARE	GDP (current US$)	NY.GDP.MKTP.CD						
12	Argentina	ARG	GDP (current US$)	NY.GDP.MKTP.CD			24450604878	18272123664	25605249382	28344705967
13	Armenia	ARM	GDP (current US$)	NY.GDP.MKTP.CD						
14	American Samoa	ASM	GDP (current US$)	NY.GDP.MKTP.CD						
15	Antigua and Barbuda	ATG	GDP (current US$)	NY.GDP.MKTP.CD						
16	Australia	AUS	GDP (current US$)	NY.GDP.MKTP.CD	18593347519	19666256020	19911524247	21527606675	23787658192	25962593795
17	Austria	AUT	GDP (current US$)	NY.GDP.MKTP.CD	6592693841	7311749633	7756110210	8374175258	9169983886	9994070616
18	Azerbaijan	AZE	GDP (current US$)	NY.GDP.MKTP.CD						
19	Burundi	BDI	GDP (current US$)	NY.GDP.MKTP.CD	195999990	202999992	213500006	232749998	260750008	158994963
20	Belgium	BEL	GDP (current US$)	NY.GDP.MKTP.CD	11658722591	12400145222	13264015675	14260017387	15960106681	17371457608
21	Benin	BEN	GDP (current US$)	NY.GDP.MKTP.CD	226195579	235668222	236434907	253927646	269818988	289908721
22	Burkina Faso	BFA	GDP (current US$)	NY.GDP.MKTP.CD	330442817	350247237	379567178	394040749	410321786	422916848
23	Bangladesh	BGD	GDP (current US$)	NY.GDP.MKTP.CD	4274893913	4817580184	5081413340	5319458351	5386054619	5906636557
24	Bulgaria	BGR	GDP (current US$)	NY.GDP.MKTP.CD						
25	Bahrain	BHR	GDP (current US$)	NY.GDP.MKTP.CD						

Data | Metadata - Countries | Metadata - Indicators



WorldPopulation - Microsoft Excel

Home | Insert | Page Layout | Formulas | Data | Review | View

A1: Country

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Country	Country C	Indicator I	Indicator (	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1
2	Aruba	ABW	Populatio	SP.POP.TC	54211	55438	56225	56695	57032	57360	57715	58055	58386	58726	59063	59440	59840	60243	60528	60657	60
3	Afghanist	AFG	Populatio	SP.POP.TC	8996351	9166764	9345868	9533954	9731361	9938414	10152331	10372630	10604346	10854428	11126123	11417825	11721940	12027822	12321541	12590286	12840
4	Angola	AGO	Populatio	SP.POP.TC	5643182	5753024	5866061	5980417	6093321	6203299	6309770	6414995	6523791	6642632	6776381	6927269	7094834	7277960	7474338	7682479	7900
5	Albania	ALB	Populatio	SP.POP.TC	1608800	1659800	1711319	1762621	1814135	1864791	1914573	1965598	2022272	2081695	2135479	2187853	2243126	2296752	2350124	2404831	2458
6	Andorra	AND	Populatio	SP.POP.TC	13411	14375	15370	16412	17469	18549	19647	20758	21890	23058	24276	25559	26892	28232	29520	30705	31
7	UAE	ARE	Populatio	SP.POP.TC	92634	101078	112472	125566	138529	150362	160481	170283	183194	203820	235499	278808	332760	397174	471364	554324	646
8	Argentina	ARG	Populatio	SP.POP.TC	20619075	20953077	21287682	21621840	21953929	22283390	22608748	22932203	23261278	23605987	23973058	24366439	24782949	25213388	25644506	26066975	26477
9	Armenia	ARM	Populatio	SP.POP.TC	1874120	1941491	2009526	2077575	2144998	2211316	2276031	2339124	2401140	2462925	2525065	2587706	2650484	2712781	2773747	2832757	2889
10	American	ASM	Populatio	SP.POP.TC	20013	20486	21117	21882	22698	23520	24321	25116	25885	26614	27292	27916	28492	29014	29488	29932	30
11	Antigua a	ATG	Populatio	SP.POP.TC	55339	56144	57144	58294	59524	60781	62059	63360	64655	65910	67098	68188	69176	70066	70878	71609	72
12	Australia	AUS	Populatio	SP.POP.TC	10276477	10483000	10742000	10950000	11167000	11388000	11651000	11799000	12009000	12263000	12507000	12937000	13177000	13380000	13723000	13893000	14033
13	Austria	AUT	Populatio	SP.POP.TC	7047539	7086299	7129864	7175811	7223801	7270889	7322066	7376998	7415403	7441055	7467086	7500482	7544201	7586115	7599038	7578903	7565
14	Azerbaija	AZE	Populatio	SP.POP.TC	3895396	4030320	4171425	4315128	4456689	4592610	4721525	4843870	4960235	5071930	5180025	5284532	5385267	5483084	5579077	5674137	5768
15	Burundi	BDI	Populatio	SP.POP.TC	2786106	2839666	2893669	2949926	3010859	3077876	3152723	3234023	3316233	3391753	3455606	3505391	3544047	3578490	3618585	3671494	3739
16	Belgium	BEL	Populatio	SP.POP.TC	9153489	9183948	9220578	9289770	9378113	9463667	9527807	9580991	9618756	9646032	9655549	9673162	9711115	9741720	9772419	9800700	9818
17	Benin	BEN	Populatio	SP.POP.TC	2431622	2465867	2502896	2542859	2585965	2632356	2682159	2735307	2791590	2850661	2912340	2976572	3043567	3113675	3187412	3265165	3347
18	Burkina Fa	BFA	Populatio	SP.POP.TC	4829288	4894580	4960326	5027821	5098890	5174870	5256363	5343019	5434041	5528174	5624600	5723381	5825173	5930483	6040041	6154545	6274
19	Banglades	BGD	Populatio	SP.POP.TC	48199747	49592802	51030137	52532417	54129100	55834038	57672990	59620669	61579473	63417394	65047770	66424744	67597470	68691185	69884420	71305923	72999
20	Bulgaria	BGR	Populatio	SP.POP.TC	7867374	7943118	8012946	8078145	8144340	8204168	8258057	8310226	8369603	8434172	8489574	8536395	8576200	8620967	8678745	8720742	8758
21	Bahrain	BHR	Populatio	SP.POP.TC	162427	167894	173144	178140	182887	187431	191780	196063	200653	206043	212605	220312	229155	239527	251911	266543	283
22	Bahamas,	BHS	Populatio	SP.POP.TC	109528	115108	121083	127333	133698	140054	146366	152609	158627	164248	169354	173863	177839	181488	185099	188882	192
23	Bosnia an	BIH	Populatio	SP.POP.TC	3225668	3288602	3353226	3417574	3478995	3535640	3586634	3632669	3675452	3717466	3760527	3805285	3851151	3897255	3942223	3985103	4025
24	Belarus	BLR	Populatio	SP.POP.TC	8198000	8271216	8351928	8437232	8524224	8610000	8696496	8785648	8874552	8960304	9040000	9115576	9188968	9257272	9317584	9367000	9411
25	Belize	BLZ	Populatio	SP.POP.TC	92064	94703	97384	100164	103069	106119	109347	112692	116061	119261	122182	124793	127150	129294	131307	133260	135

WorldPopulation

ALGORITHM AND METHODOLOGY

IMPORTING LIBRARIES

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import tensorflow as tf
from plotly.subplots import make_subplots
from sklearn.linear_model import LinearRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import mean_squared_error as mse
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import statsmodels.formula.api as smf
```

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error as mse
import statsmodels.formula.api as smf
```

LOADING THE DATASET

```
In [2]: #Load the dataset
        athletes=pd.read_csv('athlete_events.csv')
        regions=pd.read_csv('noc_regions.csv')
```

```
In [2]: df = pd.read_csv('athlete_events.csv')
        noc = pd.read_csv('noc_regions.csv')
```

```
In [12]: gdp = pd.read_excel('world_gdp.xls',skiprows=3)
```

```
In [24]: pop = pd.read_csv('WorldPopulation.csv',
                           usecols=['Country','1988','1992','1996','2000','2004','2008','2012','2016'])
         pop.head()
```

Out[24]:

	Country	1988	1992	1996	2000	2004	2008	2012	2016
0	Aruba	61079.0	68235.0	83200.0	90853	98737.0	101353.0	102577.0	104822
1	Afghanistan	11540888.0	13981231.0	17822884.0	20093756	24118979.0	27294031.0	30696958.0	34656032
2	Angola	11513968.0	12968345.0	14682284.0	16440924	18865716.0	21759420.0	25096150.0	28813463
3	Albania	3142336.0	3247039.0	3168033.0	3089027	3026939.0	2947314.0	2900401.0	2876101
4	Andorra	50434.0	58888.0	64360.0	65390	76244.0	83861.0	82431.0	77281

DATA PREPROCESSING

In [5]:
```
summer['Medal'].fillna('DNW', inplace = True)
summer = summer.drop_duplicates()
```

```
D:\CSE 3rd year\anaconda\lib\site-packages\pandas\core\series.py:4463: 
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
rsus-a-copy
  return super().fillna(
```

In [6]:
```
summer.loc[summer['region'].isnull(),['NOC', 'Team']].drop_duplicates()
```

Out[6]:

	NOC	Team
578	SGP	Singapore
6267	ROT	Refugee Olympic Athletes
44376	SGP	June Climene
61080	UNK	Unknown
64674	TUV	Tuvalu
80986	SGP	Rika II
108582	SGP	Singapore-2
235895	SGP	Singapore-1

In [14]:
```
gdp = pd.melt(gdp,
              id_vars='Country',
              value_vars=list(gdp.columns[1:]),
              var_name='Year',
              value_name='GDP')
gdp.sort_values(['Country','Year'],ascending = [True,True],inplace=True)
gdp.head()
```

Out[14]:

	Country	Year	GDP
1	Afghanistan	1988	NaN
265	Afghanistan	1992	NaN
529	Afghanistan	1996	NaN
793	Afghanistan	2000	NaN
1057	Afghanistan	2004	5.285466e+09

In [15]:
```
to_replace = ['Bahamas, The','Egypt, Arab Rep.','Iran, Islamic Rep.',"Cote d'Ivoire",'Kyrgyz Republic','North Macedonia',
              'Korea, Dem. People's Rep.','Russian Federation','Slovak Republic','Korea, Rep.','Syrian Arab Republic',
               'Trinidad and Tobago','United Kingdom','United States','Venezuela, RB','Virgin Islands (U.S.)']


new_countries =   ['Bahamas','Egypt', 'Iran', 'Ivory Coast','Kyrgyzstan','Macedonia','North Korea','Russia','Slovakia',
                   'South Korea','Syria','Trinidad','UK','USA','Venezuela','Virgin Islands, US']

gdp.replace(to_replace,new_countries,inplace=True)
```

In [16]: medals['Medal_Won'] = 1

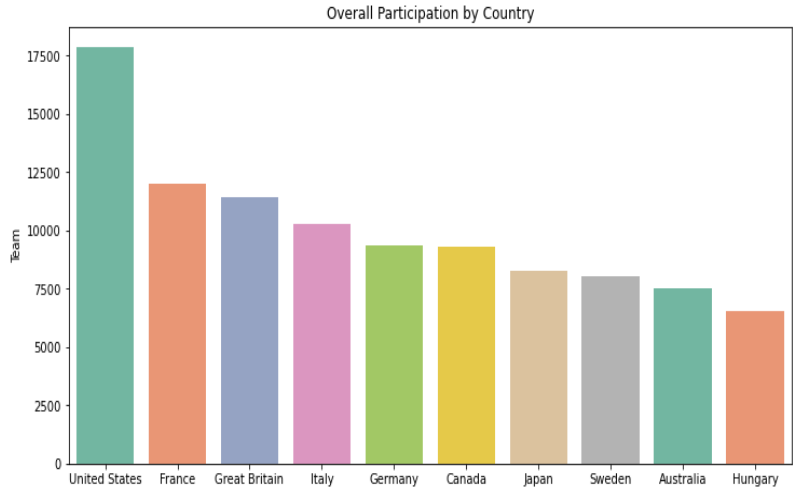
In [8]:
```
summer[['Year','City']].drop_duplicates().sort_values('Year')
```

Out[8]:

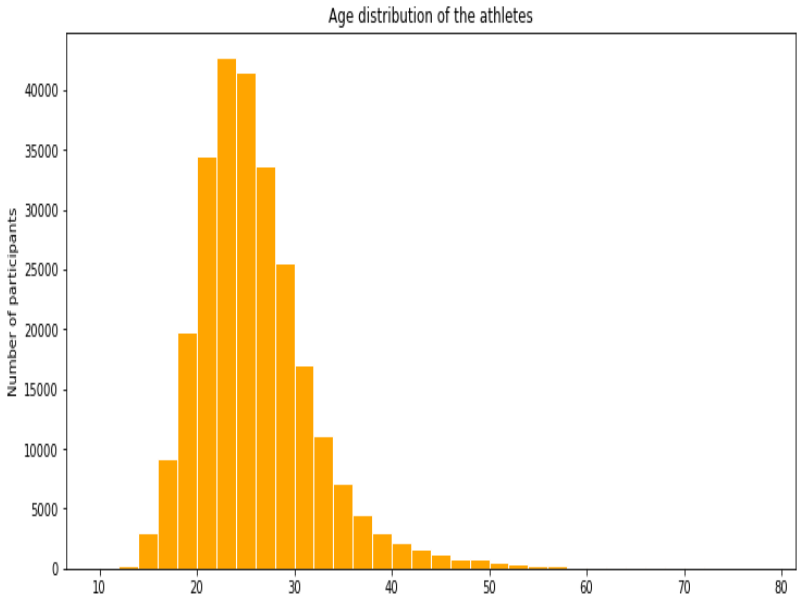
	Year	City
3079	1896	Athina
3	1900	Paris
711	1904	St. Louis
268	1906	Athina
1149	1908	London
35	1912	Stockholm
2	1920	Antwerpen
39	1924	Paris
133	1928	Amsterdam
26	1932	Los Angeles
94	1936	Berlin
41	1948	London
29	1952	Helsinki
6194	1956	Stockholm
128	1956	Melbourne
129	1960	Roma
192	1964	Tokyo
89	1968	Mexico City

DATA ANALYSIS

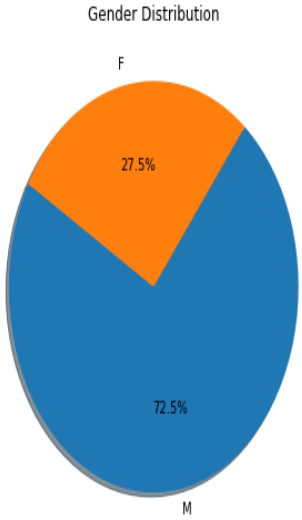
In [14]:
```
plt.figure(figsize=(12,6))
plt.title('Overall Participation by Country')
sns.barplot(x=top_10_countries.index, y=top_10_countries, palette='Set2');
```



In [15]:
```
#Age Distribution of the participants
plt.figure(figsize=(12, 6))
plt.title("Age distribution of the athletes")
plt.xlabel('Age')
plt.ylabel('Number of participants')
plt.hist (athletes_df. Age, bins = np.arange (10,80,2), color='orange', edgecolor = 'white');
```

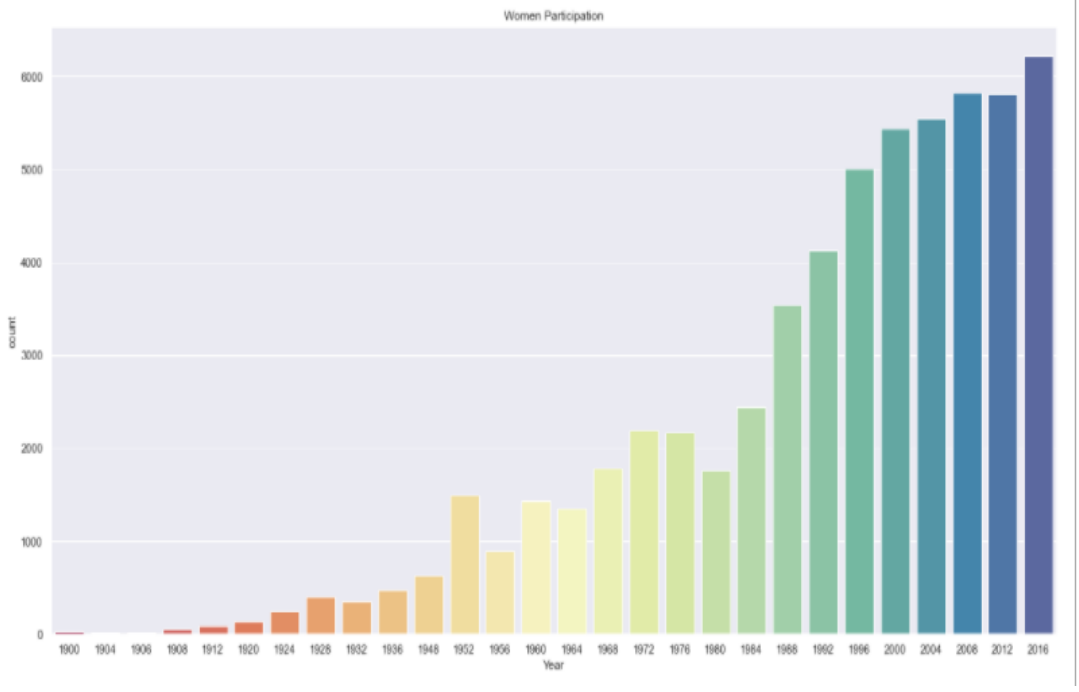


In [19]:
```
plt.figure(figsize=(12,6))
plt.title('Gender Distribution')
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=150, shadow=True);
```



In [23]:
```
sns.set(style="darkgrid")
plt.figure(figsize=(20,10))
sns.countplot(x='Year', data=womenOlympics, palette="Spectral")
plt.title('Women Participation')
```

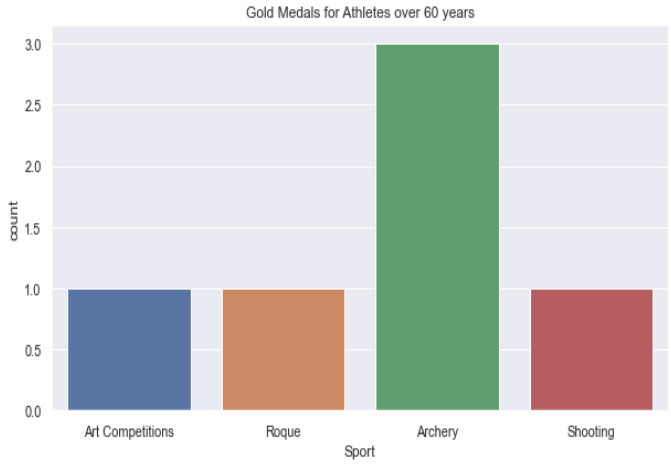
Out[23]: Text(0.5, 1.0, 'Women Participation')



In [27]:
```
plt.figure(figsize=(10,5))
plt.tight_layout()
sns.countplot(sporting_event)
plt.title('Gold Medals for Athletes over 60 years')
```

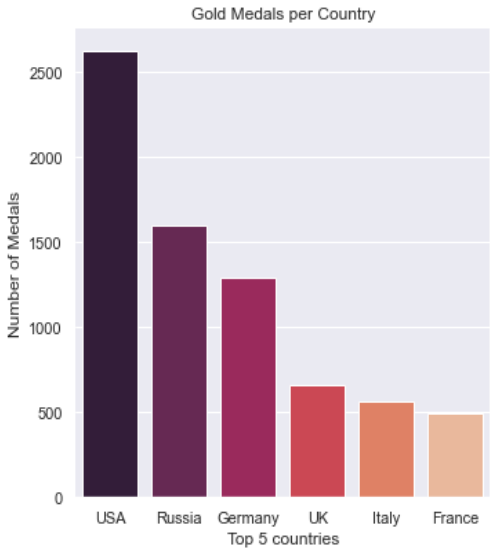
```
D:\CSE 3rd year\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]: Text(0.5, 1.0, 'Gold Medals for Athletes over 60 years')

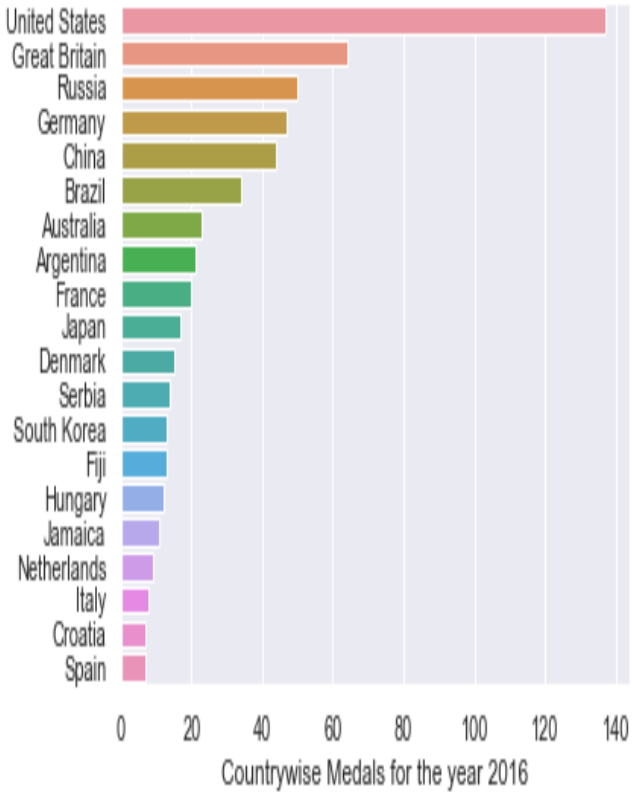


In [29]:
```
totalGoldMedals = goldMedals.region.value_counts().reset_index(name='Medal').head(6)
g = sns.catplot (x="index", y="Medal", data=totalGoldMedals,height=5, kind="bar", palette="rocket")
g.despine (left=True)
g.set_xlabels("Top 5 countries")
g.set_ylabels("Number of Medals")
plt.title('Gold Medals per Country')
```

Out[29]: Text(0.5, 1.0, 'Gold Medals per Country')

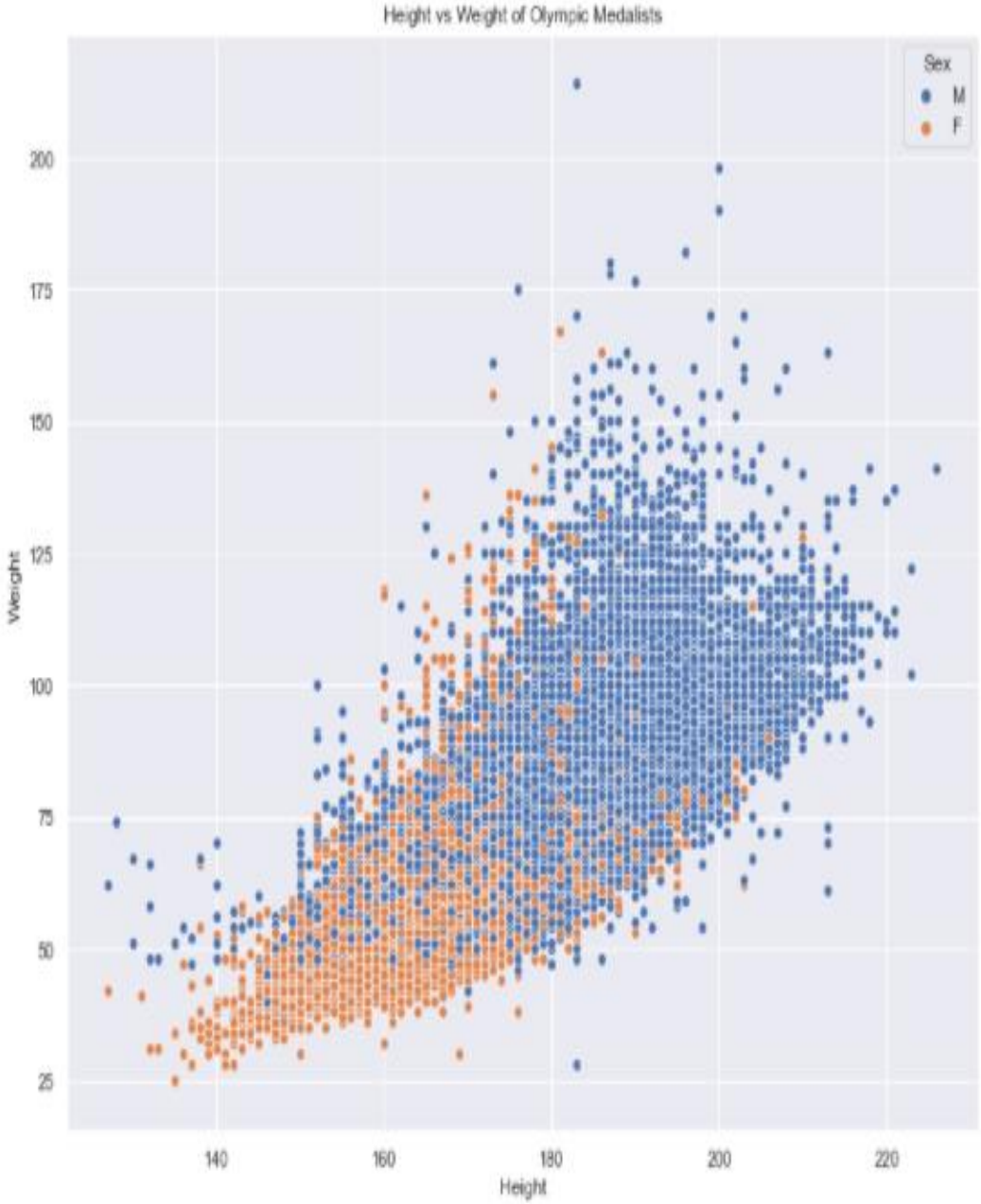


In [32]:
```
sns.barplot(x=team_names.value_counts().head(20),y=team_names.value_counts().head(20).index)
plt.ylabel (None);
plt.xlabel('Countrywise Medals for the year 2016');
```



In [33]:
```
not_null_medals=athletes_df[(athletes_df[ 'Height'].notnull())&(athletes_df['Weight'].notnull())];
plt.figure(figsize =(12, 10))
axis = sns.scatterplot (x="Height", y="Weight", data=not_null_medals, hue="Sex")
plt.title('Height vs Weight of Olympic Medalists')
```

Out[33]: Text(0.5, 1.0, 'Height vs Weight of Olympic Medalists')



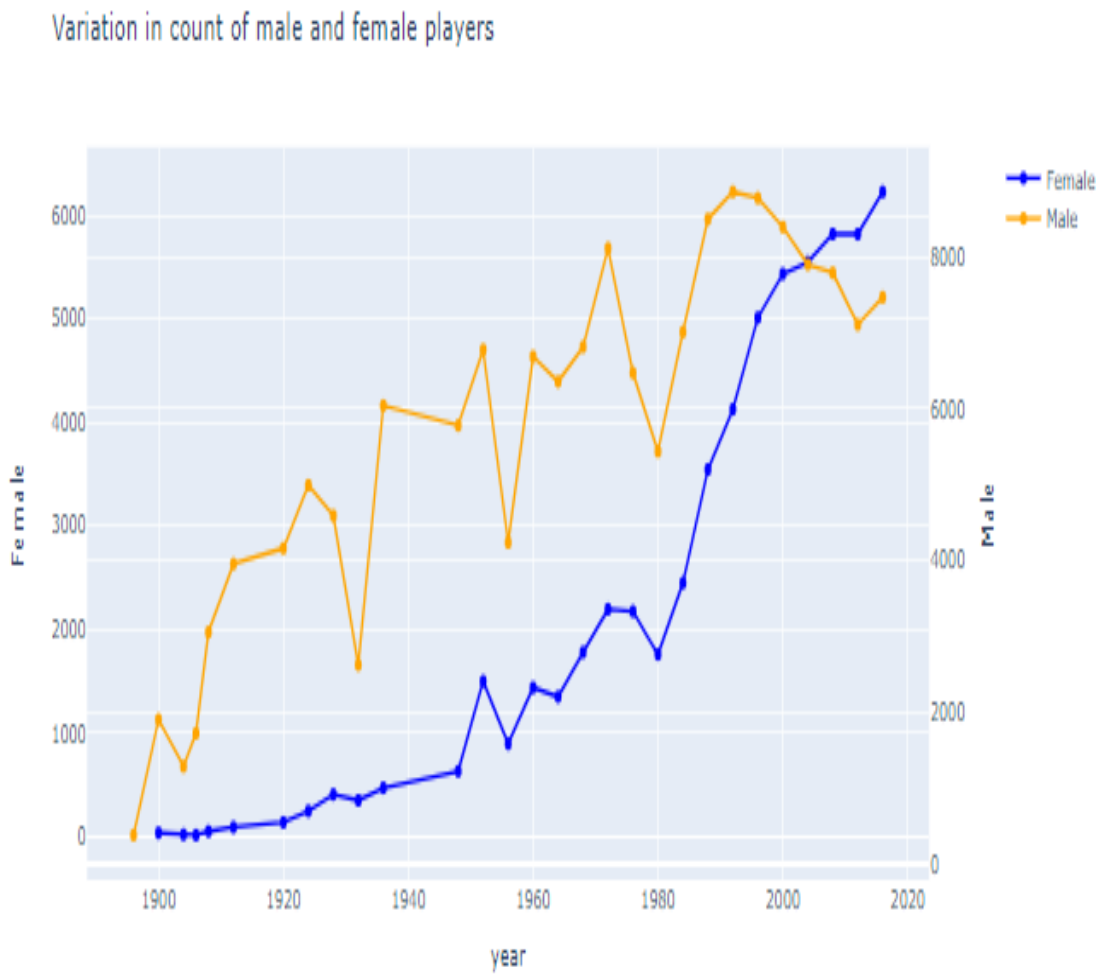
In [36]:
```
x, y1, y2 = Trend['Year'], Trend['F'], Trend['M']

fig = make_subplots(specs=[[{'secondary_y':True}]])
fig.add_trace(go.Scatter(x=x, y=y1, mode='lines+markers', name='Female',
                         line=dict(color='Blue', width=2)), secondary_y=False,)
fig.add_trace(go.Scatter(x=x, y=y2, mode='lines+markers', name='Male',
                         line=dict(color='Orange', width=2)), secondary_y=True,)

fig.update_layout(
title_text='Number of mean and women athelete over time')

fig.update_layout(title='Variation in count of male and female players',
                  xaxis_title = 'year')

fig.update_yaxes(title_text='Female', secondary_y=False)
fig.update_yaxes(title_text='Male', secondary_y=True)
fig.show()
```

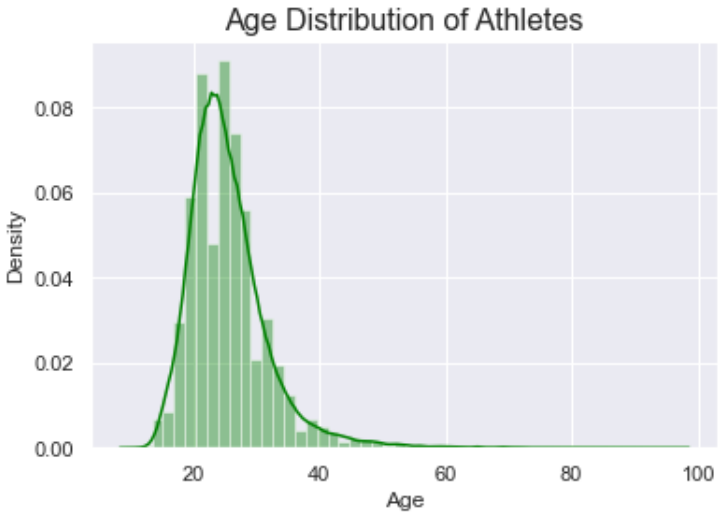


In [37]:
```
x = sns.distplot(athletes_df['Age'].dropna(), color='Green')
x.set_title('Age Distribution of Athletes', fontsize=16, fontweight=200)
```

```
D:\CSE 3rd year\anaconda\lib\site-packages\seaborn\distributions.py:2557:

`distplot` is a deprecated function and will be removed in a future versi
figure-level function with similar flexibility) or `histplot` (an axes-le
```

Out[37]: Text(0.5, 1.0, 'Age Distribution of Athletes')

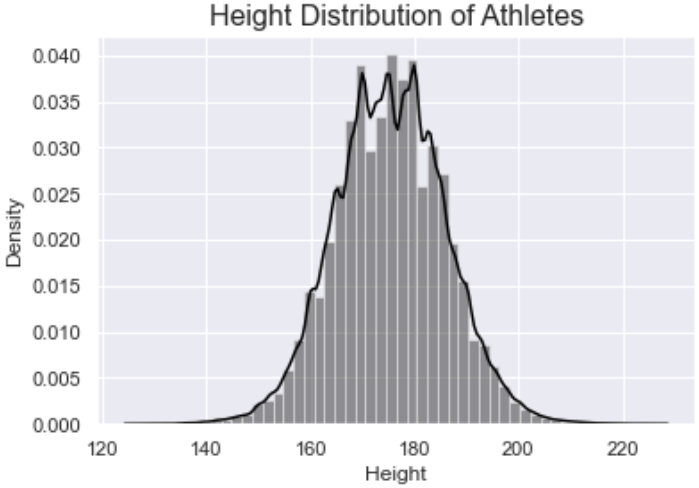


In [38]:
```
h = sns.distplot(athletes_df['Height'].dropna(), color='Black')
h.set_title('Height Distribution of Athletes', fontsize=16, fontweight=200)
```

```
D:\CSE 3rd year\anaconda\lib\site-packages\seaborn\distributions.py:2557: F

`distplot` is a deprecated function and will be removed in a future version
figure-level function with similar flexibility) or `histplot` (an axes-leve
```

Out[38]: Text(0.5, 1.0, 'Height Distribution of Athletes')

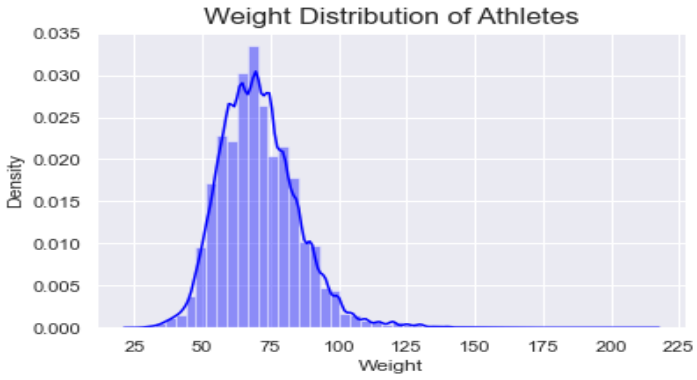


In [39]:
```
w = sns.distplot(athletes_df['Weight'].dropna(), color='Blue')
w.set_title('Weight Distribution of Athletes', fontsize=16, fontweight=200)
```

```
D:\CSE 3rd year\anaconda\lib\site-packages\seaborn\distributions.py:2557: F

`distplot` is a deprecated function and will be removed in a future version
figure-level function with similar flexibility) or `histplot` (an axes-leve
```

Out[39]:
```
Text(0.5, 1.0, 'Weight Distribution of Athletes')
```

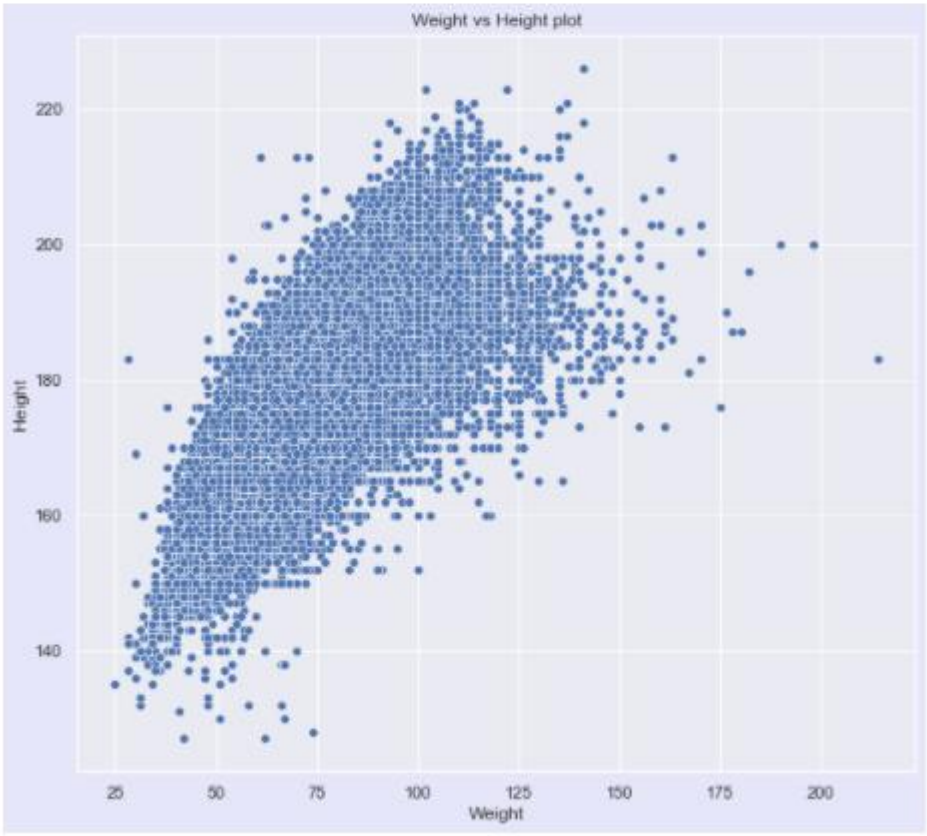


In [40]:
```
plt.figure(figsize=(10,9), facecolor='lavender')
sns.scatterplot(athletes_df['Weight'], athletes_df['Height'])
plt.title('Weight vs Height plot')
```

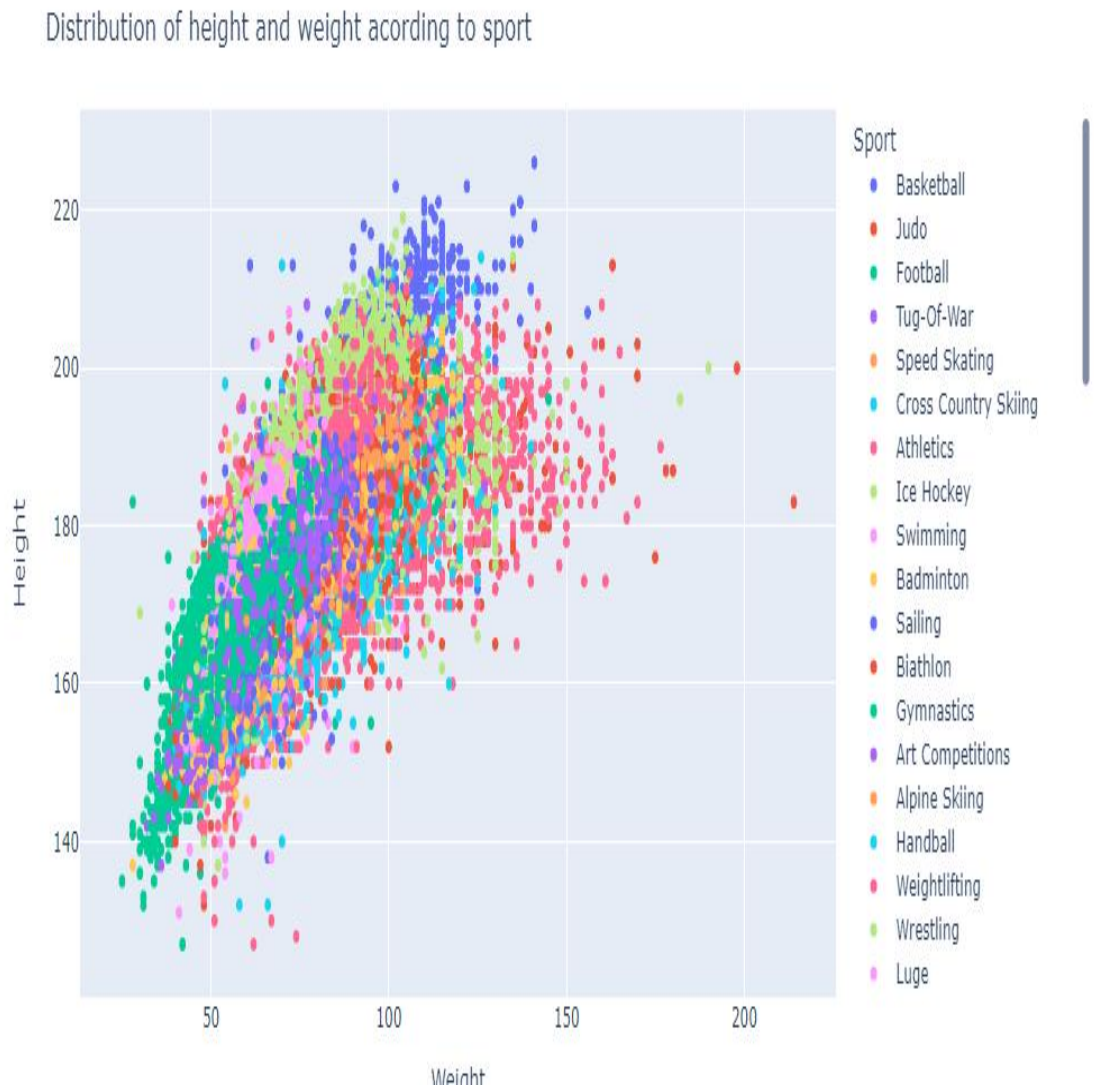
```
D:\CSE 3rd year\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarni

Pass the following variables as keyword args: x, y. From version 0.12, the only
assing other arguments without an explicit keyword will result in an error or mis
```

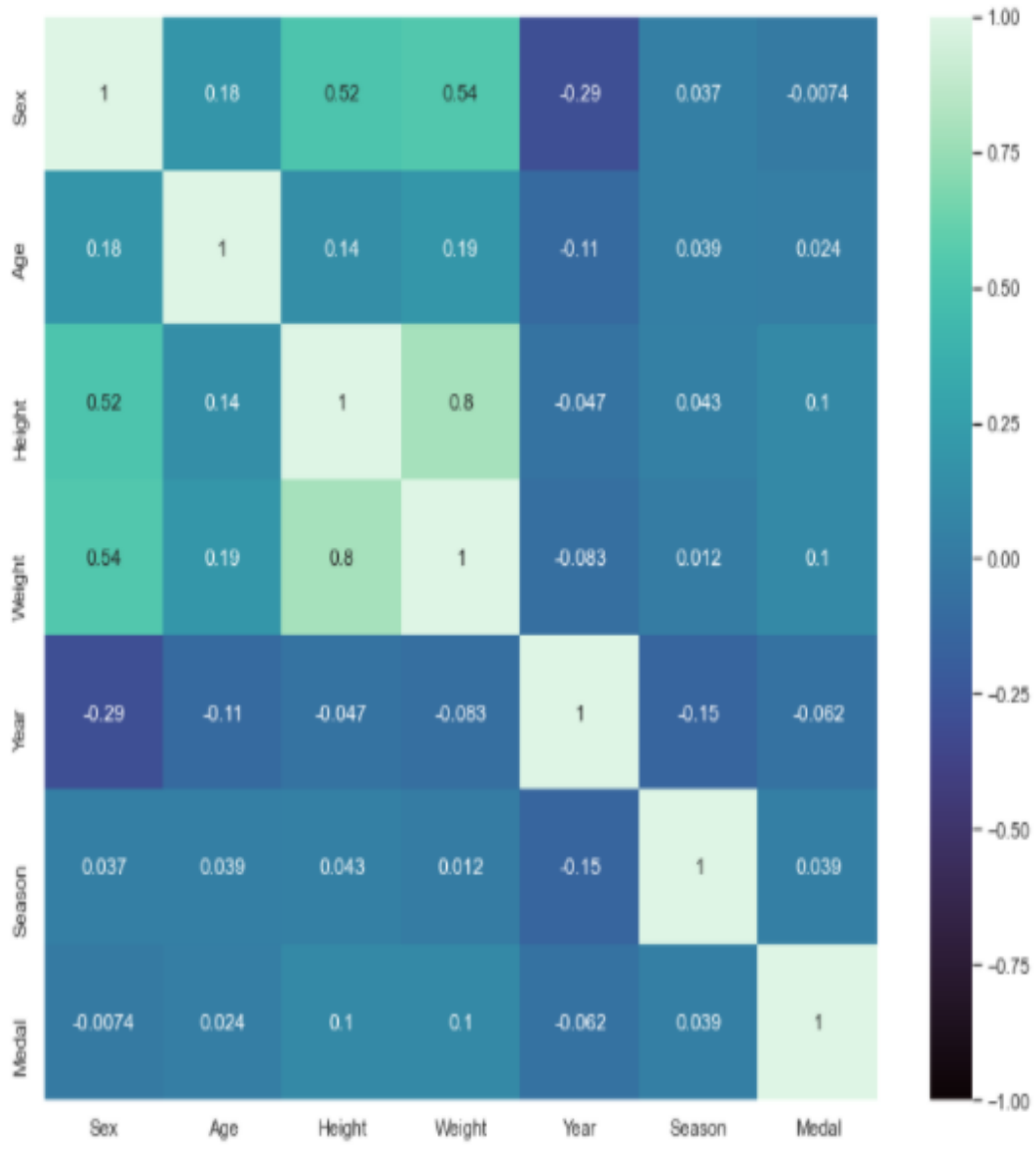
Out[40]:
```
Text(0.5, 1.0, 'Weight vs Height plot')
```



In [41]:
```
fig = px.scatter(athletes_df, x='Weight', y='Height', color='Sport')
fig.update_layout(title='Distribution of height and weight acording to sport')
fig.show()
```

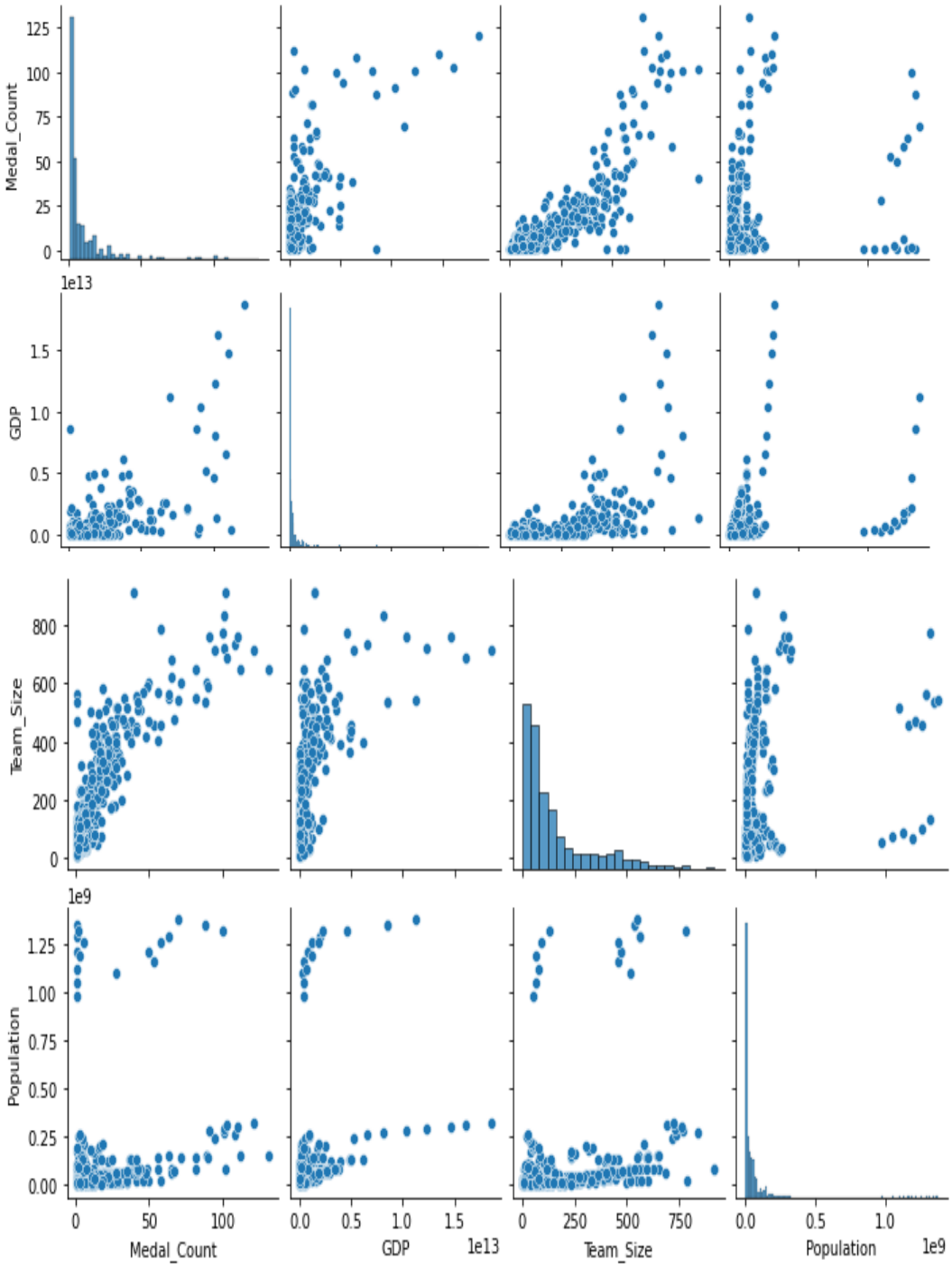


In [49]:
```
corr = data.loc[:, :'Medal'].corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr, annot=True, vmin=-1.0, cmap='mako')
plt.show()
```



```
In [27]: sns.pairplot(train,vars=['Medal_Count','GDP','Team_Size','Population'])
```

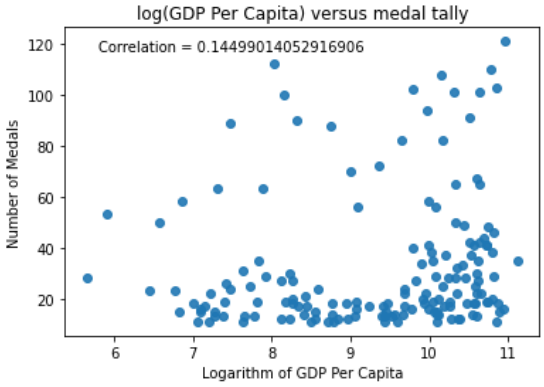
```
Out[27]: <seaborn.axisgrid.PairGrid at 0x2adf6f1adc0>
```



In [31]:
```
corr = train.loc[train['Medal_Count']>10.0,['Log_GDP_PC', 'Medal_Count']].corr()['Medal_Count'][0]

plt.plot(train.loc[train['Medal_Count']>10.0,  'Log_GDP_PC'],
     train.loc[train['Medal_Count']>10.0,  'Medal_Count'] ,
     linestyle = 'none',
     marker = 'o',
    alpha = 0.9)
plt.xlabel('Logarithm of GDP Per Capita')
plt.ylabel('Number of Medals')
plt.title('log(GDP Per Capita) versus medal tally')
plt.text(5.8,
     117,
     "Correlation = " + str(corr))
```

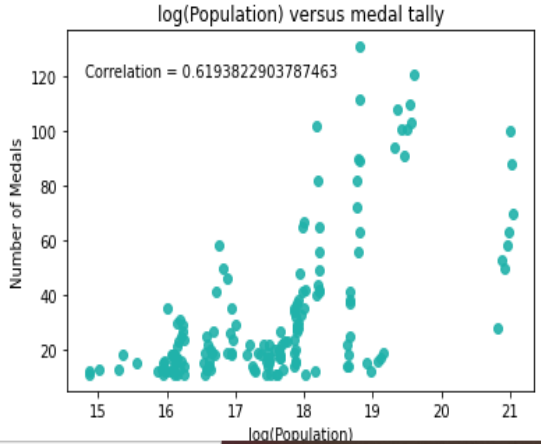
Out[31]:
```
Text(5.8, 117, 'Correlation = 0.14499014052916906')
```



In [30]:
```
corr = train.loc[train['Medal_Count']>10.0,['Log_Population', 'Medal_Count']].corr()['Medal_Count'][0]

plt.plot(train.loc[train['Medal_Count']>10.0,  'Log_Population'],
     train.loc[train['Medal_Count']>10.0,  'Medal_Count'] ,
     linestyle = 'none',
     marker = 'o',
    color = 'lightseagreen',
    alpha = 0.9)
plt.xlabel('log(Population)')
plt.ylabel('Number of Medals')
plt.title('log(Population) versus medal tally')
plt.text(14.8,
     120,
     "Correlation = " + str(corr))
```

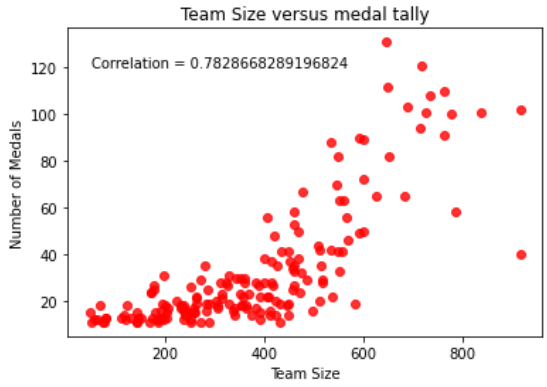
Out[30]:
```
Text(14.8, 120, 'Correlation = 0.6193822903787463')
```



In [32]:
```
corr = train.loc[train['Medal_Count']>10.0,['Team_Size', 'Medal_Count']].corr()['Medal_Count'][0]

plt.plot(train.loc[train['Medal_Count']>10.0,  'Team_Size'],
     train.loc[train['Medal_Count']>10.0,  'Medal_Count'] ,
     linestyle = 'none',
     marker = 'o',
         color='r',
    alpha = 0.8)
plt.xlabel('Team Size')
plt.ylabel('Number of Medals')
plt.title('Team Size versus medal tally')
plt.text(50,120,
     "Correlation = " + str(corr))
```

Out[32]:
```
Text(50, 120, 'Correlation = 0.7828668289196824')
```



In [10]:
```
summer['Host_Country']=summer['City'].map(country_dict)
summer.head()
```

Out[10]:

	ID	Name	Sex	Age	Height	Weight	NOC	Games	Year	Season	City	Sport	Event	Medal	Country	Host_Country
0	1	A Dijiang	M	24.0	180.0	80.0	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	DNW	China	Spain
1	2	A Lamusi	M	23.0	170.0	60.0	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	DNW	China	UK
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	DNW	Denmark	Belgium
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold	Denmark	France
26	8	Cornelia "Cor" Aalten (-Strannood)	F	18.0	168.0	NaN	NED	1932 Summer	1932	Summer	Los Angeles	Athletics	Athletics Women's 100 metres	DNW	Netherlands	USA

In [11]:
```
medals = summer.loc[summer['Medal']!='DNW']
medals.head()
```

Out[11]:

	ID	Name	Sex	Age	Height	Weight	NOC	Games	Year	Season	City	Sport	Event	Medal	Country	Host_Country
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold	Denmark	France
37	15	Arvo Ossian Aaltonen	M	30.0	NaN	NaN	FIN	1920 Summer	1920	Summer	Antwerpen	Swimming	Swimming Men's 200 metres Breaststroke	Bronze	Finland	Belgium
38	15	Arvo Ossian Aaltonen	M	30.0	NaN	NaN	FIN	1920 Summer	1920	Summer	Antwerpen	Swimming	Swimming Men's 400 metres Breaststroke	Bronze	Finland	Belgium
41	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Individual All-Around	Bronze	Finland	UK
42	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Team All-Around	Gold	Finland	UK

In [19]:
```
medals_by_country = medals_tally.loc[medals_tally['Year']>1984].groupby(['Year','NOC','Country'])['Medal_Count'].sum().reset_inde
medals_by_country.head()
```

Out[19]:

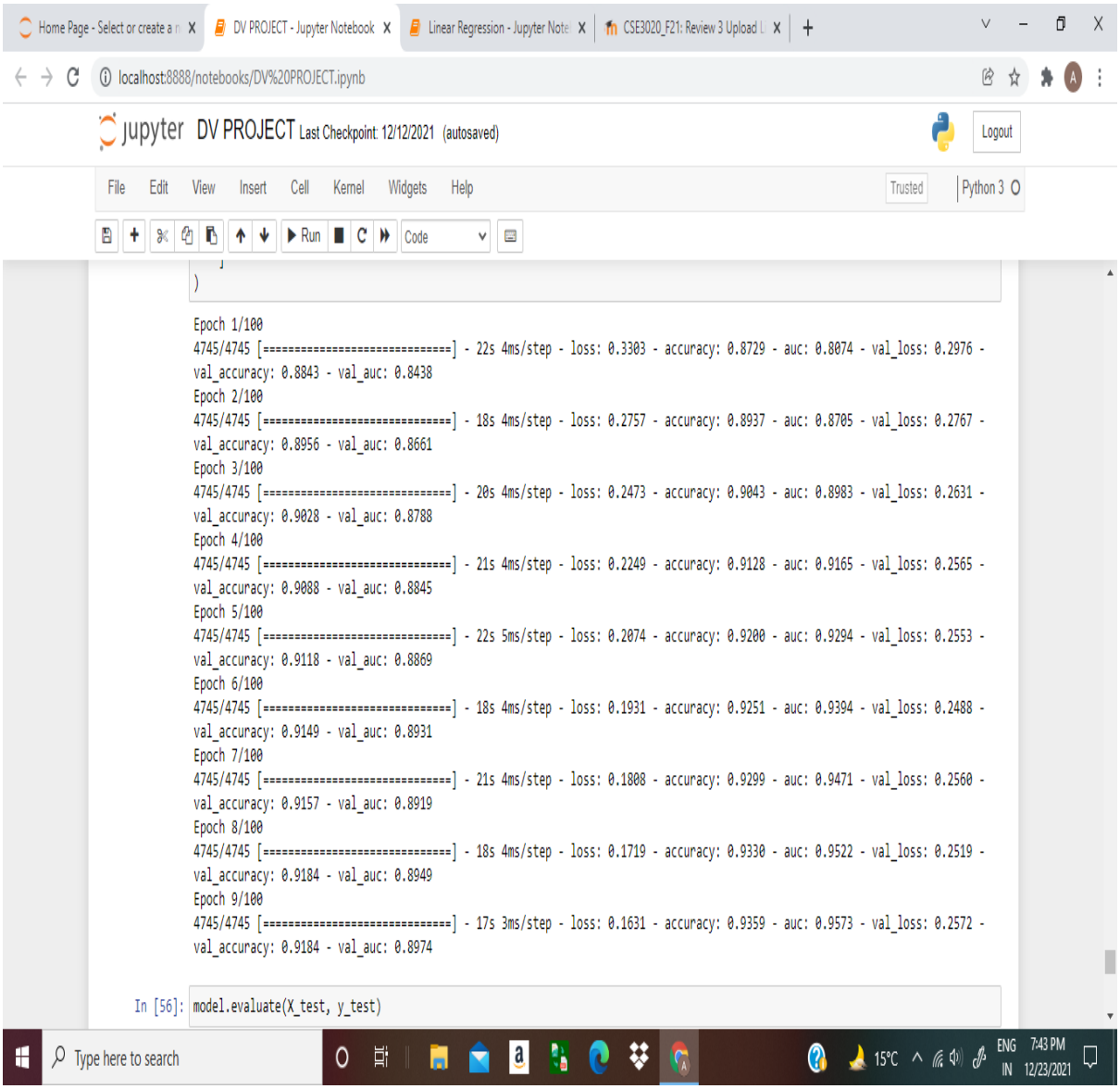
	Year	NOC	Country	Medal_Count
0	1988	AHO	Curacao	1.0
1	1988	ARG	Argentina	2.0
2	1988	AUS	Australia	14.0
3	1988	AUT	Austria	1.0
4	1988	BEL	Belgium	2.0

In [20]:
```
gdp['Year'] = gdp['Year'].astype(int)
set(medals_tally['Country']) - set(gdp['Country'])
```

Out[20]:
```
{'Individual Olympic Athletes', 'Macedonia', 'Taiwan'}
```

In [21]:
```
medals_tally_gdp = medals_by_country.merge(gdp,
                                left_on = ['Year', 'Country'],
                                right_on = ['Year', 'Country'],
                                how = 'left')
```

OUTPUT


Home Page - Select or create a
DV PROJECT - Jupyter Notebook
Linear Regression - Jupyter Note
CSE3020_F21: Review 3 Upload L
localhost:8888/notebooks/DV%20PROJECT.ipynb
jupyter DV PROJECT Last Checkpoint: 12/12/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
)
Epoch 1/100
4745/4745 [==========] - 22s 4ms/step - loss: 0.3303 - accuracy: 0.8729 - auc: 0.8074 - val_loss: 0.2976 - val_accuracy: 0.8843 - val_auc: 0.8438
Epoch 2/100
4745/4745 [==========] - 18s 4ms/step - loss: 0.2757 - accuracy: 0.8937 - auc: 0.8705 - val_loss: 0.2767 - val_accuracy: 0.8956 - val_auc: 0.8661
Epoch 3/100
4745/4745 [==========] - 20s 4ms/step - loss: 0.2473 - accuracy: 0.9043 - auc: 0.8983 - val_loss: 0.2631 - val_accuracy: 0.9028 - val_auc: 0.8788
Epoch 4/100
4745/4745 [==========] - 21s 4ms/step - loss: 0.2249 - accuracy: 0.9128 - auc: 0.9165 - val_loss: 0.2565 - val_accuracy: 0.9088 - val_auc: 0.8845
Epoch 5/100
4745/4745 [==========] - 22s 5ms/step - loss: 0.2074 - accuracy: 0.9200 - auc: 0.9294 - val_loss: 0.2553 - val_accuracy: 0.9118 - val_auc: 0.8869
Epoch 6/100
4745/4745 [==========] - 18s 4ms/step - loss: 0.1931 - accuracy: 0.9251 - auc: 0.9394 - val_loss: 0.2488 - val_accuracy: 0.9149 - val_auc: 0.8931
Epoch 7/100
4745/4745 [==========] - 21s 4ms/step - loss: 0.1808 - accuracy: 0.9299 - auc: 0.9471 - val_loss: 0.2560 - val_accuracy: 0.9157 - val_auc: 0.8919
Epoch 8/100
4745/4745 [==========] - 18s 4ms/step - loss: 0.1719 - accuracy: 0.9330 - auc: 0.9522 - val_loss: 0.2519 - val_accuracy: 0.9184 - val_auc: 0.8949
Epoch 9/100
4745/4745 [==========] - 17s 3ms/step - loss: 0.1631 - accuracy: 0.9359 - auc: 0.9573 - val_loss: 0.2572 - val_accuracy: 0.9184 - val_auc: 0.8974
In [56]:
model.evaluate(X_test, y_test)
Type here to search
15°C
ENG
IN
7:43 PM
12/23/2021

In [33]:
```
#train models on data upto 2012
X_tr = train.loc[train.Year != 2016].dropna()[['Team_Size','Log_GDP','Log_Population']]
y_tr = train.loc[train.Year != 2016].dropna()['Medal_Count']
#predict on 2016
X_tst = train.loc[train.Year == 2016].dropna()[['Team_Size','Log_GDP','Log_Population']]
y_tst = train.loc[train.Year == 2016].dropna()['Medal_Count']
```

In [34]:
```
lr = LinearRegression()
lr.fit(X_tr,y_tr)
y_pred = lr.predict(X_tst)

lr_score = lr.score(X_tst,y_tst) #this gives the R^2 score
lr_err = np.sqrt(mse(y_tst,y_pred)) #this gives the rms error

print('Linear Regression R^2: {}, Linear Regression RMSE: {}'.format(lr_score,lr_err))
```

```
Linear Regression R^2: 0.688994121700861 3, Linear Regression RMSE: 10.809899949900524
```

In [35]:
```
OLS = smf.ols('Medal_Count ~ Team_Size + Log_GDP + Log_Population', data=train.loc[train.Year!=2016]).fit()

y_ols = OLS.predict(X_tst)
ols_score = OLS.rsquared #R^2
ols_err = np.sqrt(mse(y_tst, y_ols)) #rms error
print('Statsmodels OLS R^2: {}, Statsmodels OLS RMSE: {}'.format(ols_score,ols_err))
```

```
Statsmodels OLS R^2: 0.7365664804407976, Statsmodels OLS RMSE: 10.809899949900528
```

In [58]:
```
print("Classification Report:\n\n", classification_report(y_true, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.98      0.95     69458
           1       0.82      0.53      0.64     11877

    accuracy                           0.91     81335
   macro avg       0.87      0.76      0.80     81335
weighted avg       0.91      0.91      0.91     81335
```

In [59]:
```
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
```

```
Confusion Matrix:
 [[68049  1409]
 [ 5577  6300]]
```

Result and Discussion

The number of athletes, events, and nations has grown dramatically since 1896, but growth leveled off around 2000 for the Summer Games. The Art Competitions were included from 1912 to 1948, and were dominated by Germany, France, and Italy. Nazi Germany was especially dominant in the 1936 Games. Geographic representation in the Games has grown since 1896, although Africa, Southeast Asia, the Middle East, and South America are still very under-represented. Female participation increased dramatically, and this trend started during the Cold War. Nazi women dominated the medals in 1936, East German and Soviet women dominated in 1976, and American women dominated in 2016. The size of Olympians has become more extreme over time. In most sports this means taller and heavier, but in a few sports such as gymnastics, athletes have become smaller.

CONCLUSION AND FUTURE WORK

It is clear that the host countries have always a better chance of winning medals in the Olympics; they can win at least 10–20 percent more medals. Looking at the economic effect, even though country's population and per capita GDP affected the number of medals won in the past, the total GDP of the country is more significant to determine the winnings in the recent years. With the age factor, the age range of players winning medals has decreased over the years, and an optimal age for each sport can be identified in the recent years. Thus, there is a high chance for an athlete from a host country with high GDP, whose age range falls in the optimum age range for the sport to win a medal in the Olympics

REFERENCES

Datasets https://www.kaggle.com/heesoo37/120-years-of-olympic-hist ory-athletes-and-results

- https://bmjopen.bmj.com/content/3/1/e002058.full
- https://www.sciencedirect.com/science/article/abs/pii/S1469029212000544