# Project :120 years of Olympic history :

Submitted by
SUMAN Mani (EBEON0722613186)

# Abstract

# Abstract :

Olympic games are an event where athletes from all over the world participate to compete with each other. In this paper, we have tried to study the data of Olympic games from the year 1896 – 2016. To study the dataset and derive conclusions we have used different python libraries which are used for Data Analysis. Libraries such as ‘numpy’, ‘pandas’, matplotlib , seaborn are used to study the dataset. The purpose of this paper is to analyse the country wise participation, participation of female athletes, participation of female athletes in Summer and Winter Olympics, age distribution of participants and performance analysis in Olympics from 1896 to 2016.

# CONTENTS

*Table of Content*

# Chapter – 1

# Chapter 1

# INTRODUCTION

The modern Olympic Games or Olympics are leading international sporting events featuring summer and winter sports competitions in which thousands of athletes from around the world participate in a variety of competitions. The Olympic Games are considered the world's foremost sports competition with more than 200 nations participating. The Olympic Games are held every four years, with the Summer and Winter Games alternating by occurring every four years but two years apart.

The evolution of the Olympic Movement during the 20th and 21st centuries has resulted in several changes to the Olympic Games. Some of these adjustments include the creation of the Winter Olympic Games for snow and ice sports, the Paralympic Games for athletes with a disability, the Youth Olympic Games for athletes aged 14 to 18, the five Continental games (Pan American, African, Asian, European, and Pacific), and the World Games for sports that are not contested in the Olympic Games. The Deaflympics and Special Olympics are also endorsed by the IOC. The IOC has had to adapt to a variety of economic, political, and technological advancements.

As a result, the Olympics has shifted away from pure amateurism, as envisioned by Coubertin, to allowing participation of professional athletes. The growing importance of mass media created the issue of corporate sponsorship and commercialisation of the Games. World wars led to the cancellation of the 1916, 1940, and 1944 Games. Large boycotts during the Cold War limited participation in the 1980 and 1984 Games. The latter, however, attracted 140 National Olympic Committees, which was a record at the time.

# Acknowledgements :

_The Olympic data on www.kaggle.com is the result of an incredible amount of research by a group of Olympic history enthusiasts and self-proclaimed 'statistorians'. Check out their blog for more information. All I did was consolidated their decades of work into a convenient format for data analysis.


# Tool Used :

- Python jupyter notebook.
- Machine learning algorithms
- Pandas library
- Numpy library
- Seaborn Library
- Matplotlib.pyplotlibrary
- Train Test split
- Confusion matrix

# CHAPTER - 2

# Content :

ID - Unique number of each athlete

Name - Athlete’s name

Sex - M or F

Age - Integer

Height - In centimeters

Weight - In kilograms

Team - Team name

NOC - National Olympic Committee 3-letter code

Games - Year and season

Year - Integer

Season - Summer or Winter

City - Host city

Sport - Sport

Medal - Gold, Silver, Bronze, or NA

# Chapter – 3

# Chapter – 3

## Methodology

### Data Cleaning and Preprocessing :

The datasets which were collected from UCI machine learning repository and Kaggle website contain unfiltered data which must be filtered before the final data set can be used to train the model. Also, data has some categorical variables which must be modified into numerical values for which we used Pandas library of Python. In data cleaning step, first we checked whether there are any missing or junk values in the data set for which we used the is null() function. Then for handling categorical variables we converted them into numerical variables.

### Machine Learning Algorithms :

### a) Random Forest :

Random Forest is the most famous and it is considered as the best algorithm for machine learning. It is a supervised learning algorithm. To achieve more accurate and consistent prediction, random forest creates several decision trees and combines them together. The major benefit of using it is its ability to solve both regression and classification issues.

When building each individual tree, it employs bagging and feature randomness in order to produce an uncorrelated tree forest whose collective forecast has much better accuracy than any individual tree's prediction. Baggingenhances accuracy of machine learning methods by grouping them together. In this algorithm, during the splitting of nodes it takes only random subset of nodes into an account. When splitting a node, it looks for the best feature from a random group of features rather than the most significant feature. This results into getting better accuracy. It efficiently deals with the huge data sets. It also solves the issue of overfitting in datasets. It works as follows: First, it'll select random samples from the provided dataset. Next, for every selected sample it'll create a decision tree and it'll receive a forecasted result from every created decision tree. Then foreach result which was predicted, it'll perform voting and through voting it will select the best predicted result.

## b) Logistic Regression :

Logistic regression is often used a lot of times in machine learning for predicting the likelihood of response attributes when a set of explanatory independent attributes are given. It is used when the target attribute is also known as a dependent variable having categorical values like yes/no or true/false, etc. It’s widely used for solving classification problems. It falls under the category of supervised machine learning. It efficiently solves linear and binary classification problems. It is one of the most commonly used and easy to implement algorithms. It’s a statistical technique to predict classes which are binary. When the target variable has two possible classes in that case it predicts the likelihood of occurrence of the event. In our dataset the target variable is categorical as it has only two classes-yes/no.

## c) Naive Bayes :

It is a probabilistic machine learning algorithm which is mainly used in
classification problems. 11 | Page It's based on Bayes theorem. It is simple and easy to build. It deals with huge datasets efficiently. It can solve complicated classification problems. The existence of a specific feature in a class is assumed to be independent of the presence of any other feature according to naïve bayes theorem. It's formula is as follows :$P(S|T) = P(T|S) * P(S) / P(T)$ Here, T is the event to be predicted, S is the class value for an event. This equation. will find out the class in which the expected feature for classification.

## d) K Nearest Neighbor (KNN) :

KNN is a supervised machine learning algorithm. It assumes similar objects are nearer to one another. When the parameters are continuous in that case knn is preferred. In this algorithm it classifies objects by predicting their nearest neighbor. It's simple and easy to implement and also has high speed because of which it is preferred over the other algorithms when it comes to solving classification problems. The algorithm classifies whether or not the patient has disease by taking the heart disease dataset as an input. It takes input parameters like latitude, longitude ,cld, etc and classify person with lumpy skin disease.

## d) Support Vector Machine :

It is a powerful machine learning algorithm that falls under the category of supervised learning. Many people use SVM to solve both regression and classification problems. The primary role of SVM algorithm is that it separates two classes by creating a line of hyperplanes. Data points which are closest to the hyperplane or points of the data set that , if deleted, would change the position of dividing the hyperplane are known as support vectors. As a result, they might be regarded as essential components of the data set. The margin is the distance between hyperplane and nearest data point from either collection . The goal is to select the hyperplane with the maximum possible margin between it and any point in the training set increasing the likelihood of a new data being properly classified . SVM's main objective is to find a hyperplane in N-dimensional space which will classify all the data points. The dimension of a hyperplane is actually dependent on the quantity of input features. If input has two features in that case the hyperplane will be a line and two dimensional plane.

Algorithm takes following steps :-

Step 1: Select the value for K.
Step 2 : Find the Euclidean distance of K no. of neighbors.
Step 3 : Based on calculated distance, select the K nearest neighbors in the training data which are nearest to unknown data points.

Step 4 : Calculate no. of data points in each category among these K neighbors.
Step 5 : Assign new data points to the category which has the maximum no. of
neighbors.
Step 6 : Stop.

# Import Libraries

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
```

# Reading Data

```
In [2]: ath = pd.read_csv('120 years of Olympic history athletes and results.csv')
        ath
```

Out[2]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | A Dijiang | M | 24.0 | 180.0 | 80.0 | China | CHN | 1992 Summer | 1992 | Summer | Barcelona | Basketball | Basketball Men's Basketball | No Medal |
| **1** | 2 | A Lamusi | M | 23.0 | 170.0 | 60.0 | China | CHN | 2012 Summer | 2012 | Summer | London | Judo | Judo Men's Extra-Lightweight | No Medal |
| **2** | 3 | Gunnar Nielsen Aaby | M | 24.0 | NaN | NaN | Denmark | DEN | 1920 Summer | 1920 | Summer | Antwerpen | Football | Football Men's Football | No Medal |
| **3** | 4 | Edgar Lindenau Aabye | M | 34.0 | NaN | NaN | Denmark/Sweden | DEN | 1900 Summer | 1900 | Summer | Paris | Tug-Of-War | Tug-Of-War Men's Tug-Of-War | Gold |
| **4** | 5 | Christine Jacoba Aaftink | F | 21.0 | 185.0 | 82.0 | Netherlands | NED | 1988 Winter | 1988 | Winter | Calgary | Speed Skating | Speed Skating Women's 500 metres | No Medal |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **65530** | 33537 | Nelson vora | M | 24.0 | 183.0 | 76.0 | Portugal | POR | 2008 Summer | 2008 | Summer | Beijing | Athletics | Athletics Men's Triple Jump | Gold |
| **65531** | 33537 | Nelson vora | M | 32.0 | 183.0 | 76.0 | Portugal | POR | 2016 Summer | 2016 | Summer | Rio de Janeiro | Athletics | Athletics Men's Triple Jump | No Medal |
| **65532** | 33538 | Joseph Evouna | M | 19.0 | 172.0 | 69.0 | Cameroon | CMR | 1972 Summer | 1972 | Summer | Munich | Cycling | Cycling Men's Road Race, Individual | No Medal |
| **65533** | 33538 | Joseph Evouna | M | 19.0 | 172.0 | 69.0 | Cameroon | CMR | 1972 Summer | 1972 | Summer | Munich | Cycling | Cycling Men's 100 kilometres Team Time Trial | No Medal |
| | | Joseph | | | | | | | 1980 | | | | | Cycling Men's Road | No |

## Reading Second Data

In [3]:
```
reg = pd.read_csv('noc_regions.csv')
reg
```

Out[3]:

| | NOC | region | notes |
|---|---|---|---|
| 0 | AFG | Afghanistan | NaN |
| 1 | AHO | Curacao | Netherlands Antilles |
| 2 | ALB | Albania | NaN |
| 3 | ALG | Algeria | NaN |
| 4 | AND | Andorra | NaN |
| ... | ... | ... | ... |
| 225 | YEM | Yemen | NaN |
| 226 | YMD | Yemen | South Yemen |
| 227 | YUG | Serbia | Yugoslavia |
| 228 | ZAM | Zambia | NaN |
| 229 | ZIM | Zimbabwe | NaN |

230 rows × 3 columns

## Merging Both Data

In [4]:
```
ath_df = ath.merge(reg,on = 'NOC')

# on is used for common column that is NOC
# how is used to that where we have to merge on left or right
```

In [5]:
```
ath_df
```

Out[5]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | region | notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | A Dijiang | M | 24.0 | 180.0 | 80.0 | China | CHN | 1992 Summer | 1992 | Summer | Barcelona | Basketball | Basketball Men's Basketball | No Medal | China | NaN |
| 1 | 2 | A Lamusi | M | 23.0 | 170.0 | 60.0 | China | CHN | 2012 Summer | 2012 | Summer | London | Judo | Judo Men's Extra-Lightweight | No Medal | China | NaN |
| 2 | 602 | Abudoureheman | M | 22.0 | 182.0 | 75.0 | China | CHN | 2000 Summer | 2000 | Summer | Sydney | Boxing | Boxing Men's Middleweight | No Medal | China | NaN |
| 3 | 1463 | Ai Linuer | M | 25.0 | 160.0 | 62.0 | China | CHN | 2004 Summer | 2004 | Summer | Athina | Wrestling | Wrestling Men's Lightweight, Greco-Roman | No Medal | China | NaN |
| 4 | 1464 | Ai Yanhan | F | 14.0 | 168.0 | 54.0 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Swimming | Swimming Women's 200 metres Freestyle | No Medal | China | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 65482 | 23772 | Mariana Cress | F | 17.0 | 159.0 | 52.0 | Marshall | MHL | 2016 | 2016 | Summer | Rio de | Athletics | Athletics Women's | No | Marshall | NaN |

# Data Cleaning

## Column Names

In [6]:
```
ath_df.columns
```

Out[6]:
```
Index(['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight', 'Team', 'NOC', 'Games',
       'Year', 'Season', 'City', 'Sport', 'Event', 'Medal', 'region', 'notes'],
      dtype='object')
```

## Checking Unique Values

In [7]:
```
ath_df.nunique()
```

Out[7]:
```
ID        33510
Name      33417
Sex           2
Age          68
Height       85
Weight      171
Team        795
NOC         225
Games        51
Year         35
Season        2
City         42
Sport        65
Event       742
Medal         4
region      202
notes        20
dtype: int64
```

## Top 5 Rows

In [8]:
```
ath_df.head(5)
```

Out[8]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | region | notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | A Dijiang | M | 24.0 | 180.0 | 80.0 | China | CHN | 1992 Summer | 1992 | Summer | Barcelona | Basketball | Basketball Men's Basketball | No Medal | China | NaN |
| **1** | 2 | A Lamusi | M | 23.0 | 170.0 | 60.0 | China | CHN | 2012 Summer | 2012 | Summer | London | Judo | Judo Men's Extra-Lightweight | No Medal | China | NaN |
| **2** | 602 | Abudoureheman | M | 22.0 | 182.0 | 75.0 | China | CHN | 2000 Summer | 2000 | Summer | Sydney | Boxing | Boxing Men's Middleweight | No Medal | China | NaN |
| **3** | 1463 | Ai Linuer | M | 25.0 | 160.0 | 62.0 | China | CHN | 2004 Summer | 2004 | Summer | Athina | Wrestling | Wrestling Men's Lightweight, Greco-Roman | No Medal | China | NaN |
| **4** | 1464 | Ai Yanhan | F | 14.0 | 168.0 | 54.0 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Swimming | Swimming Women's 200 metres Freestyle | No Medal | China | NaN |

## Bottom 5 Rows

In [9]:
```
ath_df.tail(5)
```

Out[9]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | region | notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **65482** | 23772 | Mariana Cress | F | 17.0 | 159.0 | 52.0 | Marshall Islands | MHL | 2016 Summer | 2016 | Summer | Rio de Janeiro | Athletics | Athletics Women's 100 metres | No Medal | Marshall Islands | NaN |
| **65483** | 23773 | Roman William Cress | M | 31.0 | NaN | NaN | Marshall Islands | MHL | 2008 Summer | 2008 | Summer | Beijing | Athletics | Athletics Men's 100 metres | No Medal | Marshall Islands | NaN |
| **65484** | 25568 | Kaingaue David | F | 17.0 | 167.0 | 67.0 | Kiribati | KIR | 2012 Summer | 2012 | Summer | London | Athletics | Athletics Women's 100 metres | No Medal | Kiribati | NaN |
| **65485** | 31292 | Fritz Eccard | M | NaN | NaN | NaN | Unknown | UNK | 1912 Summer | 1912 | Summer | Stockholm | Art Competitions | Art Competitions Mixed Architecture | No Medal | NaN | Unknown |
| **65486** | 33094 | Logona Esau | M | 21.0 | 163.0 | 69.0 | Tuvalu | TUV | 2008 Summer | 2008 | Summer | Beijing | Weightlifting | Weightlifting Men's Lightweight | No Medal | NaN | Tuvalu |

## Shape of Dataset

In [10]:
```
ath_df.shape
```

Out[10]:
```
(65487, 17)
```

## Rename columns

In [11]:
```
ath_df.rename(columns={'region':'Region' , 'notes':'Notes'},inplace=True)
```

## Information about data

In [12]:
```
ath_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 65487 entries, 0 to 65486
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      65487 non-null  int64  
 1   Name    65487 non-null  object 
 2   Sex     65487 non-null  object 
 3   Age     62941 non-null  float64
 4   Height  50212 non-null  float64
 5   Weight  49421 non-null  float64
 6   Team    65487 non-null  object 
 7   NOC     65487 non-null  object 
 8   Games   65487 non-null  object 
 9   Year    65487 non-null  int64  
 10  Season  65487 non-null  object 
 11  City    65487 non-null  object 
 12  Sport   65487 non-null  object 
 13  Event   65487 non-null  object 
 14  Medal   65487 non-null  object 
 15  Region  65479 non-null  object 
 16  Notes   1191 non-null   object 
dtypes: float64(3), int64(2), object(12)
memory usage: 9.0+ MB
```

## Statistical Information

In [13]:
```
ath_df.describe()
```

Out[13]:

| | ID | Age | Height | Weight | Year |
|---|---|---|---|---|---|
| **count** | 65487.000000 | 62941.000000 | 50212.000000 | 49421.000000 | 65487.000000 |
| **mean** | 16955.900209 | 25.644206 | 175.512387 | 70.914136 | 1977.715791 |
| **std** | 9591.980854 | 6.487744 | 10.381617 | 14.235204 | 30.153737 |
| **min** | 1.000000 | 11.000000 | 127.000000 | 25.000000 | 1896.000000 |
| **25%** | 8797.500000 | 21.000000 | 168.000000 | 61.000000 | 1960.000000 |

## Datatypes

In [14]:
```
ath_df.dtypes
```

Out[14]:
```
ID          int64
Name       object
Sex        object
Age       float64
Height    float64
Weight    float64
Team       object
NOC        object
Games      object
Year        int64
Season     object
City       object
Sport      object
Event      object
Medal      object
Region     object
Notes      object
dtype: object
```

## Droping Column

In [15]:
```
ath_df.drop(['Notes'],axis=1,inplace=True)
```

## Checking null values

In [16]:
```
ath_df.isna().sum()
```

Out[16]:
```
ID            0
Name          0
Sex           0
Age        2546
Height    15275
Weight    16066
Team          0
NOC           0
Games         0
Year          0
Season        0
City          0
Sport         0
Event         0
Medal         0
Region        8
dtype: int64
```

# Exploratory Data Analysis

# athlets participated from India in olympics

In [19]:
```
ath_df.query('Team == "India"')
```

Out[19]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **41167** | 281 | S. Abdul Hamid | M | 0 | 0 | 0 | India | IND | 1928 Summer | 1928 | Summer | Amsterdam | Athletics | Athletics Men's 110 metres Hurdles | No Medal | India |
| **41168** | 281 | S. Abdul Hamid | M | 0 | 0 | 0 | India | IND | 1928 Summer | 1928 | Summer | Amsterdam | Athletics | Athletics Men's 400 metres Hurdles | No Medal | India |
| **41169** | 512 | Shiny Kurisingal Abraham-Wilson | F | 19.0 | 167.0 | 53.0 | India | IND | 1984 Summer | 1984 | Summer | Los Angeles | Athletics | Athletics Women's 800 metres | No Medal | India |
| **41170** | 512 | Shiny Kurisingal Abraham-Wilson | F | 19.0 | 167.0 | 53.0 | India | IND | 1984 Summer | 1984 | Summer | Los Angeles | Athletics | Athletics Women's 4 x 400 metres Relay | No Medal | India |
| **41171** | 512 | Shiny Kurisingal Abraham-Wilson | F | 23.0 | 167.0 | 53.0 | India | IND | 1988 Summer | 1988 | Summer | Seoul | Athletics | Athletics Women's 800 metres | No Medal | India |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **41453** | 31804 | Karunagaran Ekambaram | M | 26.0 | 164.0 | 52.0 | India | IND | 1980 Summer | 1980 | Summer | Moskva | Weightlifting | Weightlifting Men's Flyweight | No Medal | India |
| **41454** | 31835 | Deep Grace Ekka | F | 22.0 | 158.0 | 63.0 | India | IND | 2016 Summer | 2016 | Summer | Rio de Janeiro | Hockey | Hockey Women's Hockey | No Medal | India |
| **41455** | 32514 | Lionel Charles Renwick Emmett | M | 23.0 | 0 | 0 | India | IND | 1936 Summer | 1936 | Summer | Berlin | Hockey | Hockey Men's Hockey | Gold | India |
| **41456** | 33340 | Kamineni Eswara | M | 33.0 | 0 | 88.5 | India | IND | 1952 | 1952 | Summer | Helsinki | Weightlifting | Weightlifting Men's | No | India |

# athlets participated from Japan in olympics

In [20]:
```
ath_df.query('Team == "Japan"')
```

Out[20]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **42628** | 362 | Isao Ko Abe | M | 24.0 | 177.0 | 75.0 | Japan | JPN | 1936 Summer | 1936 | Summer | Berlin | Athletics | Athletics Men's Hammer Throw | No Medal | Japan |
| **42632** | 363 | Kazumi Abe | M | 28.0 | 178.0 | 67.0 | Japan | JPN | 1976 Winter | 1976 | Winter | Innsbruck | Bobsleigh | Bobsleigh Men's Four | No Medal | Japan |
| **42633** | 364 | Kazuo Abe | M | 25.0 | 166.0 | 69.0 | Japan | JPN | 1960 Summer | 1960 | Summer | Roma | Wrestling | Wrestling Men's Lightweight, Freestyle | No Medal | Japan |
| **42634** | 365 | Kinya Abe | M | 23.0 | 168.0 | 68.0 | Japan | JPN | 1992 Summer | 1992 | Summer | Barcelona | Fencing | Fencing Men's Foil, Individual | No Medal | Japan |
| **42635** | 366 | Kiyoshi Abe | M | 25.0 | 167.0 | 62.0 | Japan | JPN | 1972 Summer | 1972 | Summer | Munich | Wrestling | Wrestling Men's Featherweight, Freestyle | No Medal | Japan |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **43145** | 33377 | Takashi Eto | M | 25.0 | 183.0 | 67.0 | Japan | JPN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Athletics | Athletics Men's High Jump | No Medal | Japan |
| **43146** | 33378 | Yosuke Eto | M | 25.0 | 162.0 | 60.0 | Japan | JPN | 1960 Winter | 1960 | Winter | Squaw Valley | Ski Jumping | Ski Jumping Men's Normal Hill, Individual | No Medal | Japan |
| **43147** | 33378 | Yosuke Eto | M | 25.0 | 162.0 | 60.0 | Japan | JPN | 1960 Winter | 1960 | Winter | Squaw Valley | Nordic Combined | Nordic Combined Men's Individual | No Medal | Japan |
| **43148** | 33378 | Yosuke Eto | M | 29.0 | 162.0 | 60.0 | Japan | JPN | 1964 Winter | 1964 | Winter | Innsbruck | Ski Jumping | Ski Jumping Men's Normal Hill, Individual | No Medal | Japan |
| **43149** | 33378 | Yosuke Eto | M | 29.0 | 162.0 | 60.0 | Japan | JPN | 1964 Winter | 1964 | Winter | Innsbruck | Ski Jumping | Ski Jumping Men's Large Hill, Individual | No Medal | Japan |

# Top Countries Participating

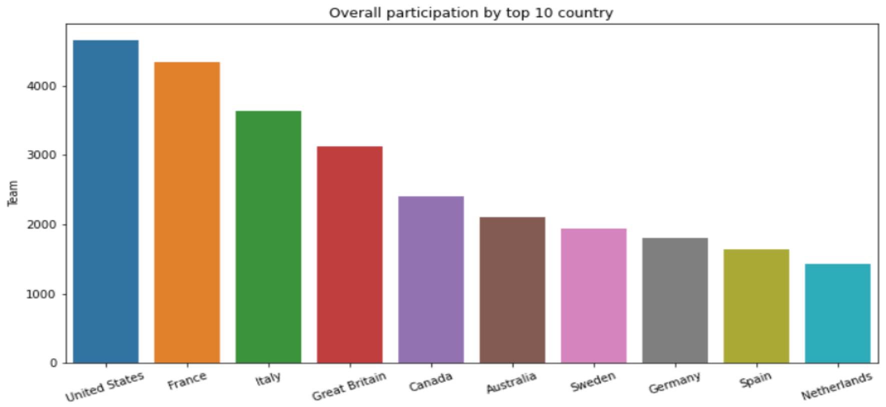
In [21]:
```
top_10_countries = ath_df.Team.value_counts().sort_values(ascending=False).head(10)
```

In [22]:
```
top_10_countries
```

Out[22]:
```
United States    4659
France           4345
Italy            3634
Great Britain    3120
Canada           2405
Australia        2108
Sweden           1942
Germany          1808
Spain            1641
Netherlands      1434
Name: Team, dtype: int64
```

## Plot graph for top 10 countries with index

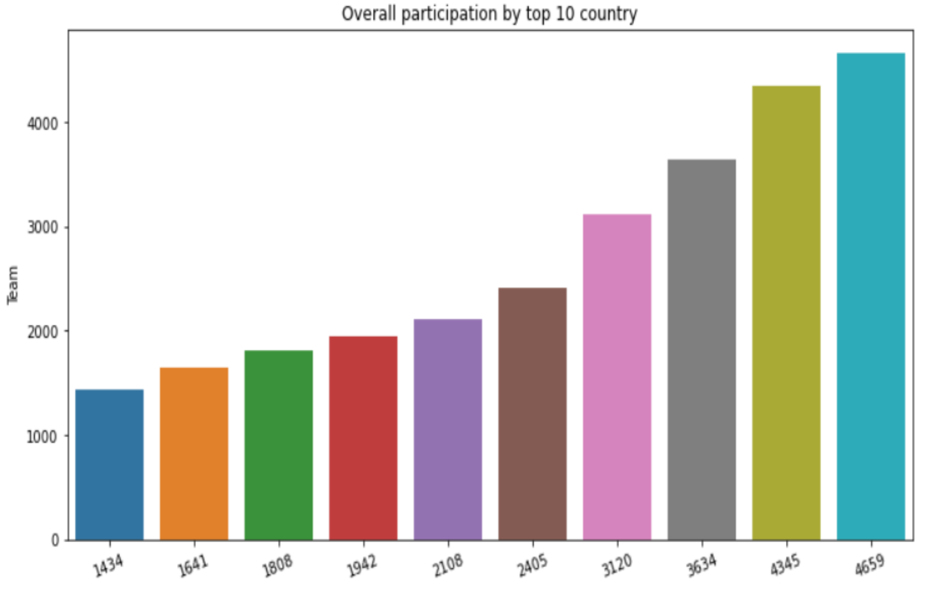
In [23]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Overall participation by top 10 country')
sns.barplot(x=top_10_countries.index,y=top_10_countries)
plt.show()
```




From USA most of the participant participated in olympics

## Plot graph for top 10 countries without index,with exact number

In [24]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Overall participation by top 10 country')
sns.barplot(x=top_10_countries,y=top_10_countries)
plt.show()
```



# Last of bottom 20 countries

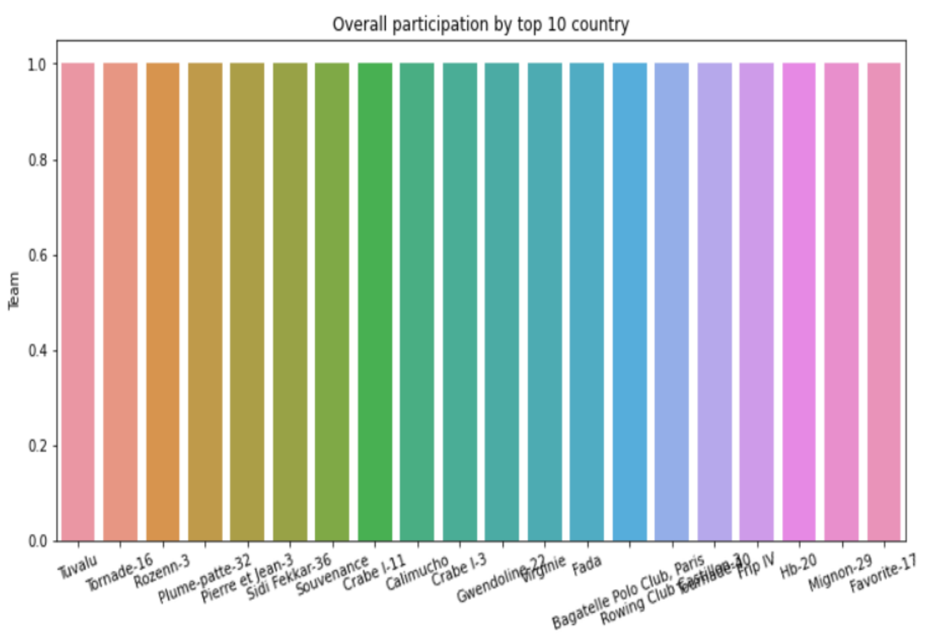
In [25]:
```
bottom_20_countries = ath_df.Team.value_counts().sort_values(ascending=True).head(20)
```

In [26]:
```
bottom_20_countries
```

Out[26]:
```
Tuvalu                       1
Tornade-16                   1
Rozenn-3                     1
Plume-patte-32               1
Pierre et Jean-3             1
Sidi Fekkar-36               1
Souvenance                   1
Crabe I-11                   1
Calimucho                    1
Crabe I-3                    1
Gwendoline-22                1
Virginie                     1
Fada                         1
Bagatelle Polo Club, Paris   1
Rowing Club Castillon-3      1
Tournade-40                  1
Frip IV                      1
Hb-20                        1
Mignon-29                    1
Favorite-17                  1
Name: Team, dtype: int64
```

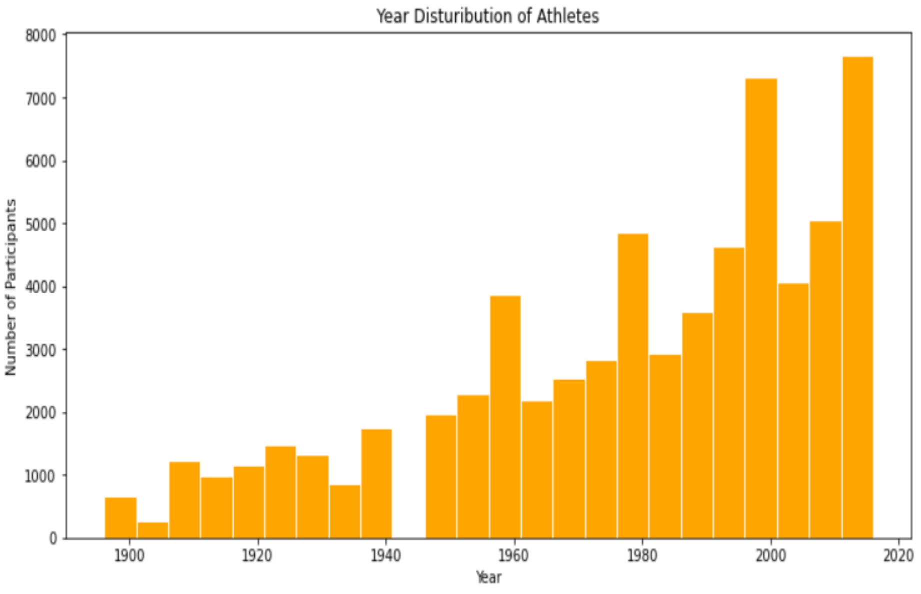
# plot graph for bottom 20 countries

In [27]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Overall participation by top 10 country')
sns.barplot(x=bottom_20_countries.index,y=bottom_20_countries)
plt.show()
```



## Year wise Distribution of the Participants

In [28]:
```
plt.figure(figsize=(12,6))
plt.title('Year Disturibution of Athletes')
plt.xlabel('Year')
plt.ylabel('Number of Participants')
plt.hist(ath_df.Year,color='orange',bins=np.arange(1896,2020,5),edgecolor='white')
plt.show()
```

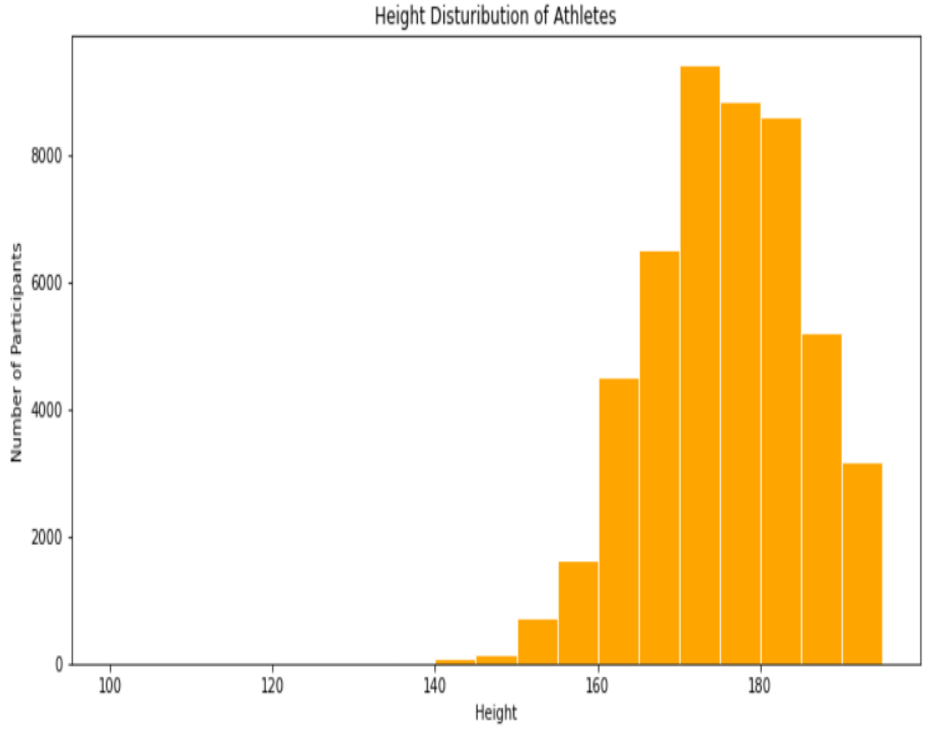



In year 2015 and in year 2000 most participant participate in olympics

# Height wise distribution

In [29]:
```
#converting height column from float to integer
ath_df['Height']=ath_df['Height'].astype(int)
```

In [30]:
```
plt.figure(figsize=(12,6))
plt.title('Height Disturibution of Athletes')
plt.xlabel('Height')
plt.ylabel('Number of Participants')
plt.hist(ath_df.Height,color='orange',bins=np.arange(100,200,5),edgecolor='white')
plt.show()
```

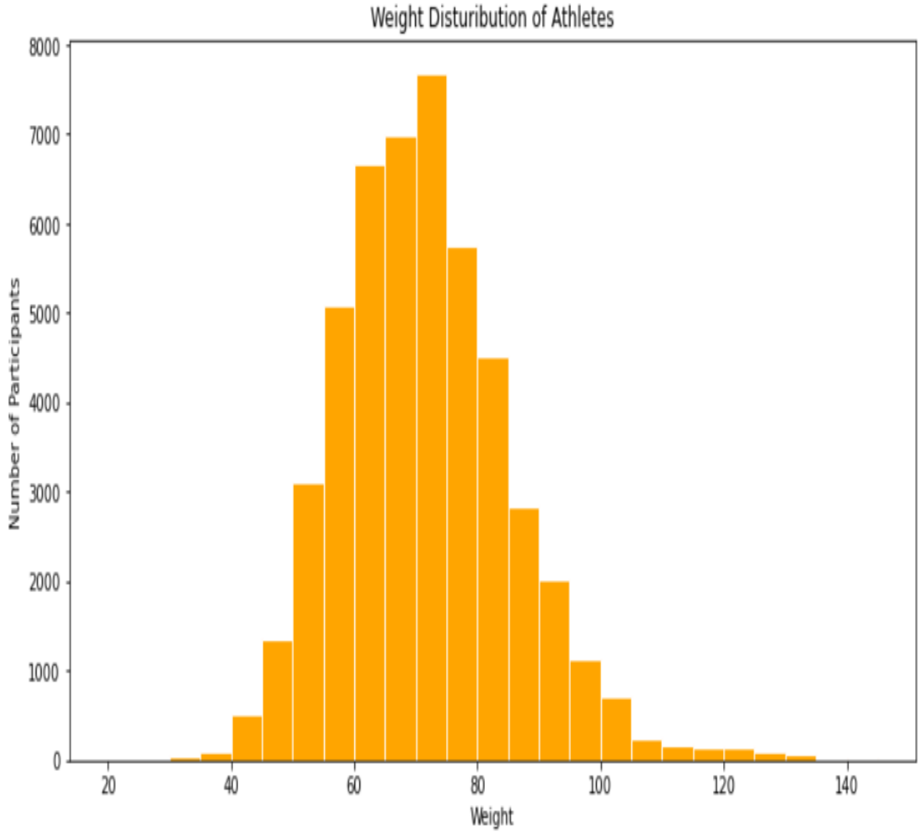



Paricipants whose height from 170cm to 190cm has participated most in olympics

# Weight Wise Distribution

In [31]:
```
#converting Weight column from float to integer
ath_df['Weight']=ath_df['Weight'].astype(int)
```

In [32]:
```
plt.figure(figsize=(12,6))
plt.title('Weight Disturibution of Athletes')
plt.xlabel('Weight')
plt.ylabel('Number of Participants')
plt.hist(ath_df.Weight,color='orange',bins=np.arange(20,150,5),edgecolor='white')
plt.show()
```

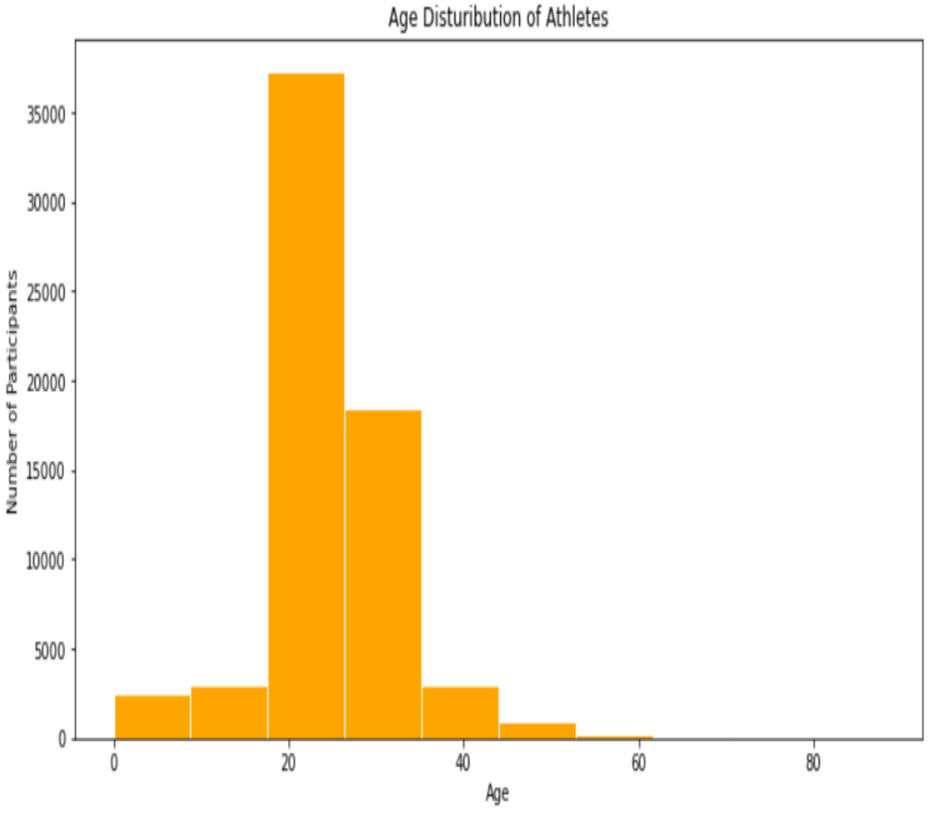



athletes whose weight is between 60kg to 80kg participated most in olympic

# Age wise Distribution

In [33]:
```
#converting age column from float to integer
ath_df['Age']=ath_df['Age'].astype(int)
```

In [34]:
```
plt.figure(figsize=(12,6))
plt.title('Age Disturibution of Athletes')
plt.xlabel('Age')
plt.ylabel('Number of Participants')
plt.hist(ath_df.Age,color='orange',edgecolor='white')
plt.show()
```

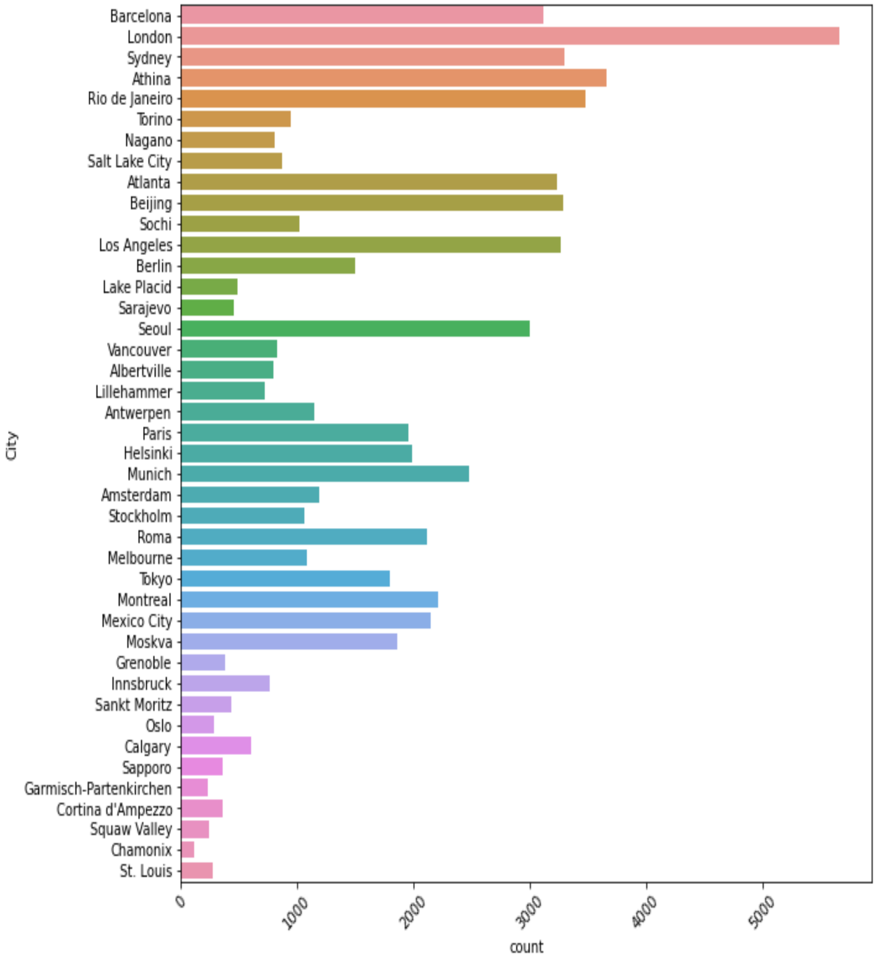



athletes whose age is between 20 to 30 participated most in olympic

## City wise Participants

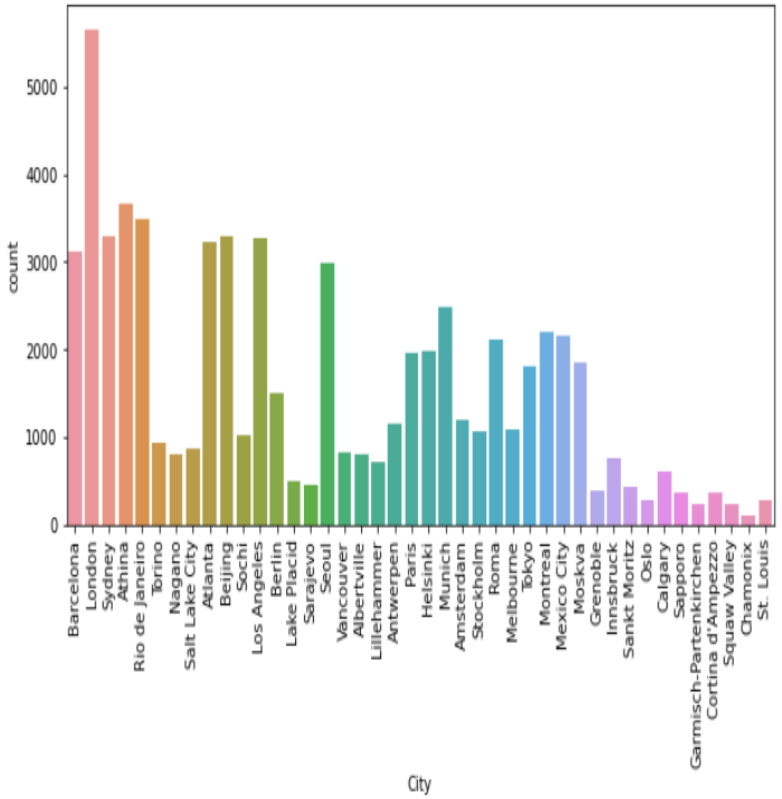
In [35]:
```
#with 45 degree rotation

plt.figure(figsize=(10,10))
sns.countplot(y=ath_df['City'])
plt.xticks(rotation=45)
plt.show()
```



In [36]:
```
#with 90 degree rotation

plt.figure(figsize=(10,5))
sns.countplot(x=ath_df['City'])
plt.xticks(rotation=90)
plt.show()
```




athletes from London Participated most in olympics

## Checking unique values in Medal

In [37]:
```
ath_df['Medal'].unique()
```

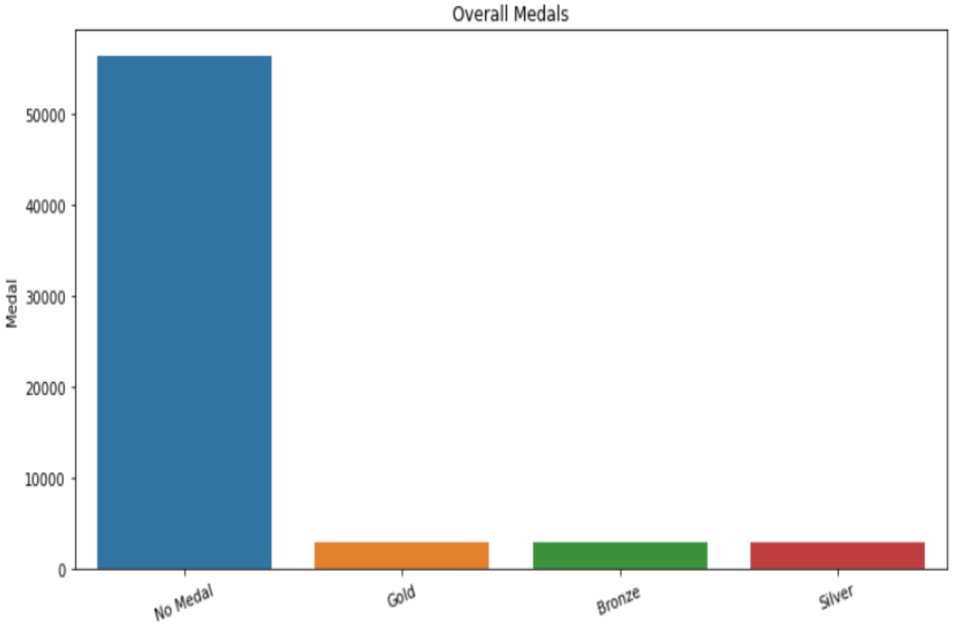
Out[37]:
```
array(['No Medal', 'Silver', 'Bronze', 'Gold'], dtype=object)
```

In [38]:
```
medal = ath_df.Medal.value_counts()
medal
```

Out[38]:
```
No Medal    56419
Gold         3051
Bronze       3023
Silver       2994
Name: Medal, dtype: int64
```

## Ploting Graph for Medal

In [39]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Overall Medals')
sns.barplot(x=medal.index,y=medal)
plt.show()
```




From Graph it is clearly seen that count of no medal is too high than Gold,Silver,Bronze

## Name of sports played in winter

In [40]:
```
winter_sports = ath_df[ath_df.Season == 'Winter'].Sport.unique()
winter_sports
```

Out[40]:
```
array(['Speed Skating', 'Short Track Speed Skating', 'Curling',
       'Figure Skating', 'Snowboarding', 'Cross Country Skiing',
       'Ice Hockey', 'Freestyle Skiing', 'Alpine Skiing', 'Bobsleigh',
       'Nordic Combined', 'Biathlon', 'Ski Jumping', 'Skeleton', 'Luge',
       'Military Ski Patrol', 'Alpinism'], dtype=object)
```

## Name of sports played in Summer ¶

In [41]:
```
summer_sports = ath_df[ath_df.Season == 'Summer'].Sport.unique()
summer_sports
```

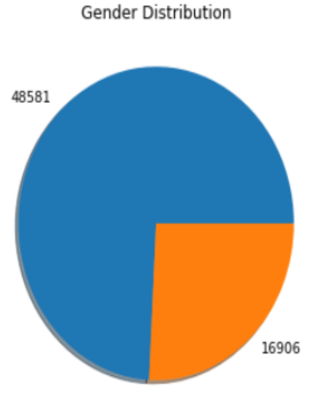
Out[41]:
```
array(['Basketball', 'Judo', 'Boxing', 'Wrestling', 'Swimming',
       'Softball', 'Hockey', 'Archery', 'Triathlon', 'Football',
       'Rhythmic Gymnastics', 'Athletics', 'Badminton', 'Fencing',
       'Gymnastics', 'Volleyball', 'Baseball', 'Water Polo', 'Shooting',
       'Weightlifting', 'Cycling', 'Rowing', 'Sailing', 'Diving',
       'Modern Pentathlon', 'Art Competitions', 'Synchronized Swimming',
       'Handball', 'Canoeing', 'Table Tennis', 'Tennis', 'Taekwondo',
       'Beach Volleyball', 'Trampolining', 'Tug-Of-War', 'Equestrianism',
       'Golf', 'Polo', 'Rugby Sevens', 'Ice Hockey', 'Figure Skating',
       'Roque', 'Rugby', 'Lacrosse', 'Cricket', 'Croquet',
       'Basque Pelota', 'Alpinism', 'Racquets', 'Motorboating',
       'Jeu De Paume'], dtype=object)
```

## Count of Male and Female Participant

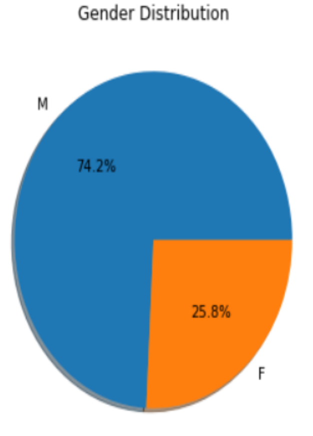
In [42]:
```
gender_counts=ath_df['Sex'].value_counts()
gender_counts
```

Out[42]:
```
M    48581
F    16906
Name: Sex, dtype: int64
```

In [43]:
```
plt.figure(figsize=(10,5))
plt.title('Gender Distribution')
plt.pie(gender_counts,labels=gender_counts,shadow=True)
plt.show()
```




In [44]:
```
plt.figure(figsize=(10,5))
plt.title('Gender Distribution')
plt.pie(gender_counts,labels=gender_counts.index,shadow=True,autopct='%1.1f%%')
plt.show()
```



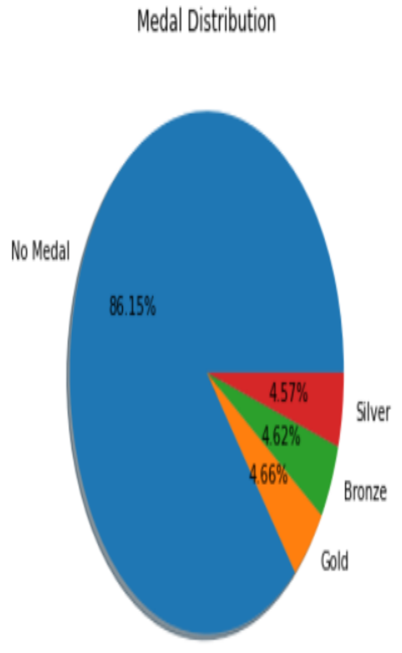

from piechart it is clearly seen that Male participant are more than female i.e- 72.5%

## total medal won by athletes

In [45]:
```
medal_counts = ath_df.Medal.value_counts()
medal_counts
```

Out[45]:
```
No Medal     56419
Gold          3051
Bronze        3023
Silver        2994
Name: Medal, dtype: int64
```

In [46]:
```
plt.figure(figsize=(10,5))
plt.title('Medal Distribution')
plt.pie(medal_counts,labels=medal_counts.index,shadow=True,autopct='%.2f%%')
plt.show()
```




4.84% won silver,4.90% won bronze and 4.94% won gold

## Total number of female athletes in each olympics

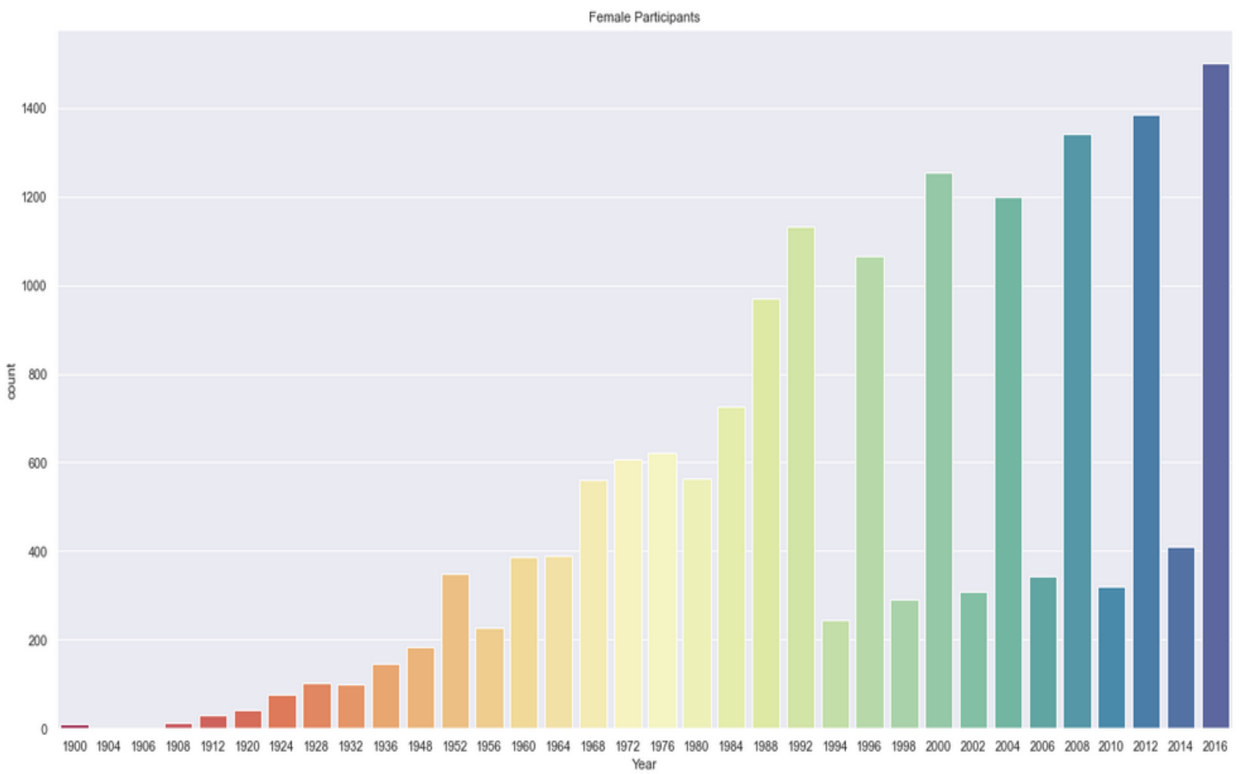
In [47]:
```
female_participants = ath_df[(ath_df.Sex=='F')][['Sex','Year']]
female_participants1 = female_participants.groupby('Year').count().reset_index()
```

In [48]:
```
female_participants1.tail(10)   ## result for bottom 10 years
```

Out[48]:

| | Year | Sex |
|---|---|---|
| 24 | 1998 | 290 |
| 25 | 2000 | 1253 |
| 26 | 2002 | 308 |
| 27 | 2004 | 1198 |
| 28 | 2006 | 342 |
| 29 | 2008 | 1341 |
| 30 | 2010 | 319 |

In [49]:
```
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
sns.countplot(x='Year',data=female_participants,palette='Spectral')
plt.title('Female Participants')
plt.show()
```




In year 2016 most females participated in olympics then in year 2012 to 2008 count is nearly same.

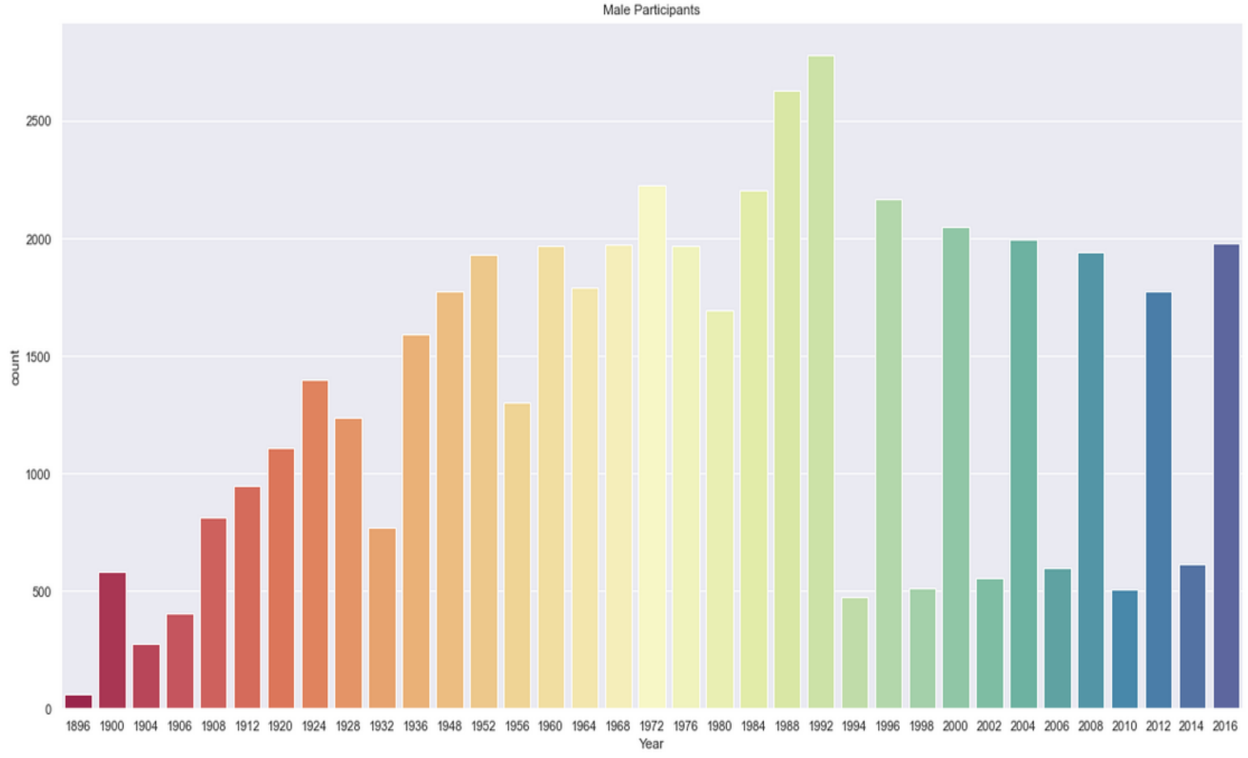
# Total number of male athletes in each olympics

In [50]:
```
male_participants = ath_df[(ath_df.Sex=='M')][['Sex','Year']]
male_participants1 = male_participants.groupby('Year').count().reset_index()
male_participants1.tail(10)    ## result for bottom 10 years
```

Out[50]:

| | Year | Sex |
|---|---|---|
| 25 | 1998 | 511 |
| 26 | 2000 | 2046 |
| 27 | 2002 | 557 |
| 28 | 2004 | 1994 |
| 29 | 2006 | 599 |
| 30 | 2008 | 1943 |
| 31 | 2010 | 509 |
| 32 | 2012 | 1772 |
| 33 | 2014 | 615 |
| 34 | 2016 | 1979 |

In [51]:
```
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
sns.countplot(x='Year',data=male_participants,palette='Spectral')
plt.title('Male Participants')
plt.show()
```




In year 1992 most male athletes participated in olympic

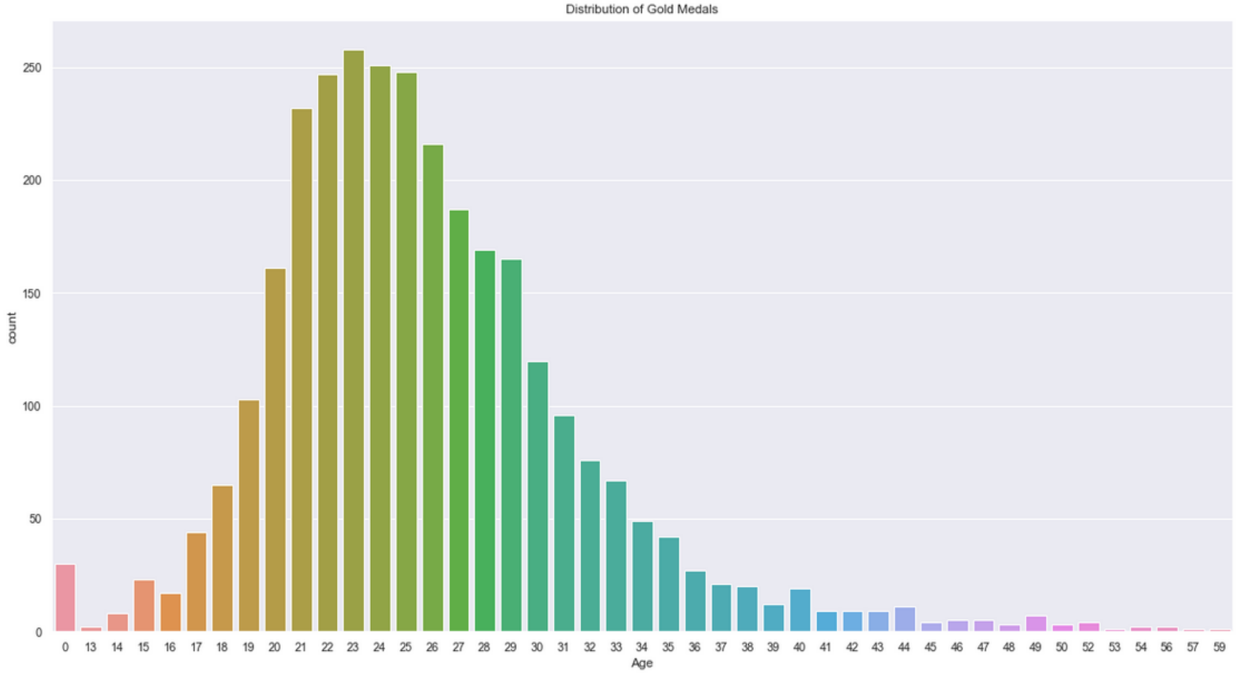
## Gold medal athletes

In [52]:
```
gold_medals = ath_df[(ath_df.Medal == 'Gold')]
gold_medals.head(10)
```

Out[52]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **68** | 17294 | Cai Yalin | M | 23 | 174 | 60 | China | CHN | 2000 Summer | 2000 | Summer | Sydney | Shooting | Shooting Men's Air Rifle, 10 metres | Gold | China |
| **77** | 17299 | Cai Yun | M | 32 | 181 | 68 | China-1 | CHN | 2012 Summer | 2012 | Summer | London | Badminton | Badminton Men's Doubles | Gold | China |
| **87** | 17995 | Cao Lei | F | 24 | 168 | 75 | China | CHN | 2008 Summer | 2008 | Summer | Beijing | Weightlifting | Weightlifting Women's Heavyweight | Gold | China |
| **104** | 18005 | Cao Yuan | M | 17 | 160 | 42 | China | CHN | 2012 Summer | 2012 | Summer | London | Diving | Diving Men's Synchronized Platform | Gold | China |
| **105** | 18005 | Cao Yuan | M | 21 | 160 | 42 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Diving | Diving Men's Springboard | Gold | China |
| **125** | 20150 | Chen Aisen | M | 20 | 168 | 60 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Diving | Diving Men's Platform | Gold | China |
| **126** | 20150 | Chen Aisen | M | 20 | 168 | 60 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Diving | Diving Men's Synchronized Platform | Gold | China |
| **141** | 20182 | Chen Ding | M | 19 | 175 | 62 | China | CHN | 2012 Summer | 2012 | Summer | London | Athletics | Athletics Men's 20 kilometres Walk | Gold | China |
| **180** | 20217 | Chen Jing | F | 19 | 170 | 60 | China | CHN | 1988 Summer | 1988 | Summer | Seoul | Table Tennis | Table Tennis Women's Singles | Gold | China |
| **186** | 20220 | Chen Jing | F | 28 | 182 | 75 | China | CHN | 2004 Summer | 2004 | Summer | Athina | Volleyball | Volleyball Women's Volleyball | Gold | China |

In [53]:
```
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
sns.countplot(gold_medals['Age'])
plt.title('Distribution of Gold Medals')
plt.show()
```




athletes whose age are 23 and 24 won most gold medals

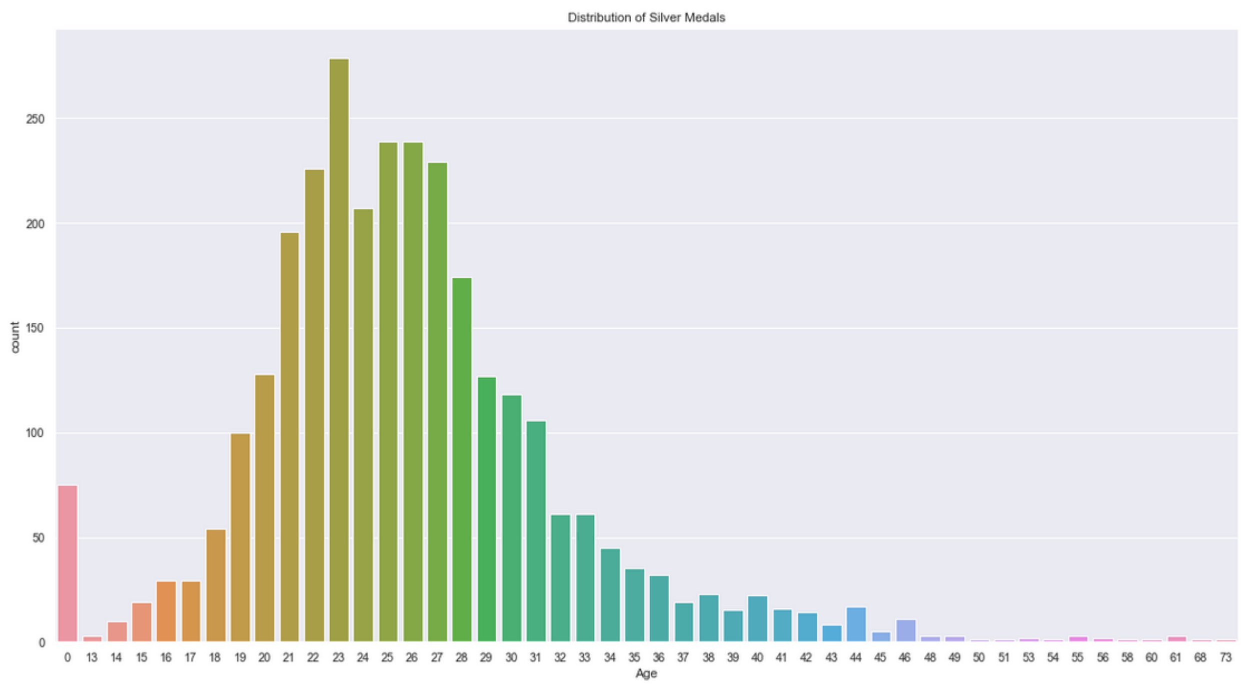
## Silver medal athletes

In [54]:
```python
silver_medals = ath_df[(ath_df.Medal == 'Silver')]
silver_medals.head(10)
```

Out[54]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **8** | 3610 | An Yulong | M | 19 | 173 | 70 | China | CHN | 1998 Winter | 1998 | Winter | Nagano | Short Track Speed Skating | Short Track Speed Skating Men's 500 metres | Silver | China |
| **12** | 3611 | An Zhongxin | F | 23 | 170 | 65 | China | CHN | 1996 Summer | 1996 | Summer | Atlanta | Softball | Softball Women's Softball | Silver | China |
| **33** | 7597 | Bao Yingying | F | 24 | 172 | 67 | China | CHN | 2008 Summer | 2008 | Summer | Beijing | Fencing | Fencing Women's Sabre, Team | Silver | China |
| **41** | 11223 | Bi Wenjing | F | 14 | 142 | 35 | China | CHN | 1996 Summer | 1996 | Summer | Atlanta | Gymnastics | Gymnastics Women's Uneven Bars | Silver | China |
| **63** | 17289 | Cai Tongtong | F | 18 | 168 | 48 | China | CHN | 2008 Summer | 2008 | Summer | Beijing | Rhythmic Gymnastics | Rhythmic Gymnastics Women's Group | Silver | China |
| **76** | 17299 | Cai Yun | M | 28 | 181 | 68 | China-1 | CHN | 2008 Summer | 2008 | Summer | Beijing | Badminton | Badminton Men's Doubles | Silver | China |
| **79** | 17300 | Cai Zelin | M | 25 | 175 | 55 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Athletics | Athletics Men's 20 kilometres Walk | Silver | China |
| **91** | 17996 | Cao Mianying | F | 29 | 176 | 71 | China | CHN | 1996 Summer | 1996 | Summer | Atlanta | Rowing | Rowing Women's Double Sculls | Silver | China |
| **109** | 18007 | Cao Zhongrong | M | 30 | 180 | 73 | China | CHN | 2012 Summer | 2012 | Summer | London | Modern Pentathlon | Modern Pentathlon Men's Individual | Silver | China |
| **118** | 19779 | Chang Si | F | 25 | 170 | 56 | China | CHN | 2012 Summer | 2012 | Summer | London | Synchronized Swimming | Synchronized Swimming Women's Team | Silver | China |

In [55]:
```python
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
sns.countplot(silver_medals['Age'])
plt.title('Distribution of Silver Medals')
plt.show()
```




athletes whose age are 23 most silver medals and after that whose age are 24 and 25 won silver medals

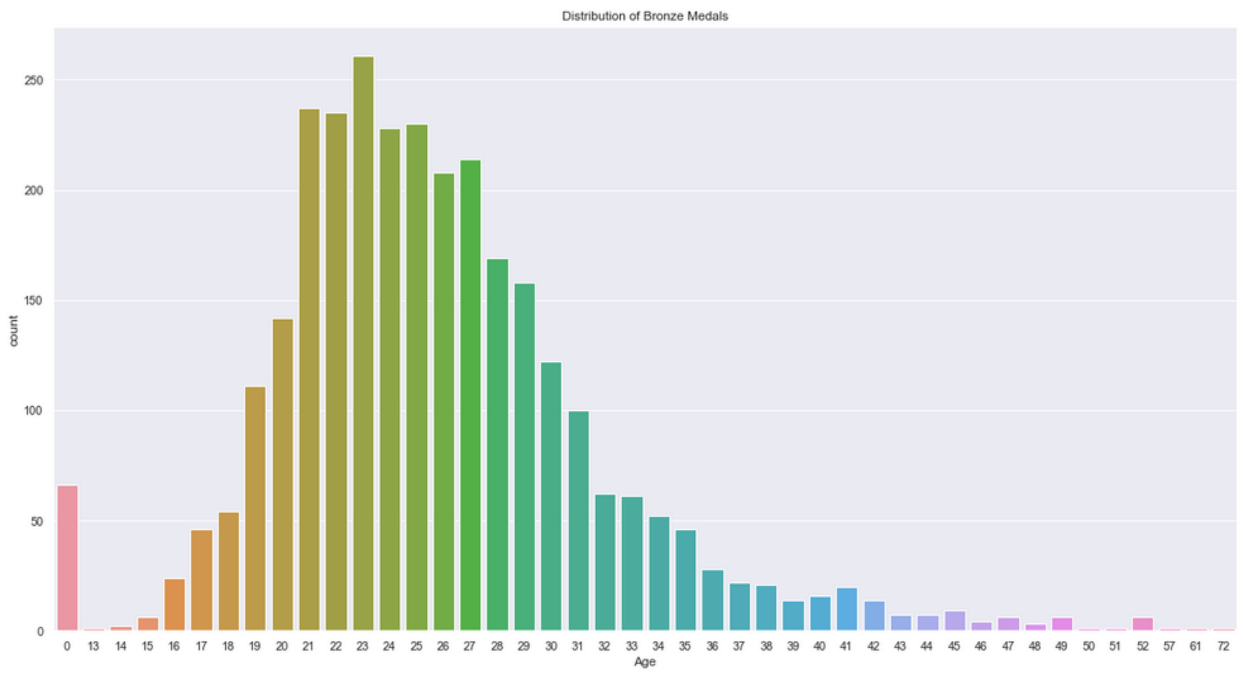
## Bronze Medal athletes

In [56]:
```
bronze_medals = ath_df[(ath_df.Medal == 'Bronze')]
bronze_medals.head(10)
```

Out[56]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 3610 | An Yulong | M | 19 | 173 | 70 | China | CHN | 1998 Winter | 1998 | Winter | Nagano | Short Track Speed Skating | Short Track Speed Skating Men's 5,000 metres R... | Bronze | China |
| **11** | 3610 | An Yulong | M | 23 | 173 | 70 | China | CHN | 2002 Winter | 2002 | Winter | Salt Lake City | Short Track Speed Skating | Short Track Speed Skating Men's 5,000 metres R... | Bronze | China |
| **17** | 6381 | Ba Yan | F | 21 | 183 | 78 | China | CHN | 1984 Summer | 1984 | Summer | Los Angeles | Basketball | Basketball Women's Basketball | Bronze | China |
| **53** | 17282 | Cai Huijue | F | 16 | 174 | 63 | China | CHN | 1996 Summer | 1996 | Summer | Atlanta | Swimming | Swimming Women's 4 x 100 metres Medley Relay | Bronze | China |
| **106** | 18005 | Cao Yuan | M | 21 | 160 | 42 | China | CHN | 2016 Summer | 2016 | Summer | Rio de Janeiro | Diving | Diving Men's Synchronized Springboard | Bronze | China |
| **140** | 20181 | Chen Dequan | M | 18 | 176 | 66 | China | CHN | 2014 Winter | 2014 | Winter | Sochi | Short Track Speed Skating | Short Track Speed Skating Men's 5,000 metres R... | Bronze | China |
| **176** | 20215 | Chen Jin | M | 22 | 181 | 73 | China | CHN | 2008 Summer | 2008 | Summer | Beijing | Badminton | Badminton Men's Singles | Bronze | China |
| **203** | 20238 | Chen Long | M | 23 | 188 | 81 | China | CHN | 2012 Summer | 2012 | Summer | London | Badminton | Badminton Men's Singles | Bronze | China |
| **208** | 20240 | Chen Lu | F | 17 | 162 | 52 | China | CHN | 1994 Winter | 1994 | Winter | Lillehammer | Figure Skating | Figure Skating Women's Singles | Bronze | China |
| **209** | 20240 | Chen Lu | F | 21 | 162 | 52 | China | CHN | 1998 Winter | 1998 | Winter | Nagano | Figure Skating | Figure Skating Women's Singles | Bronze | China |

In [57]:
```
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
sns.countplot(bronze_medals['Age'])
plt.title('Distribution of Bronze Medals')
plt.show()
```




athletes whose age between 22 to 25 won most bronze medals

## Top 10 country who won gold medal

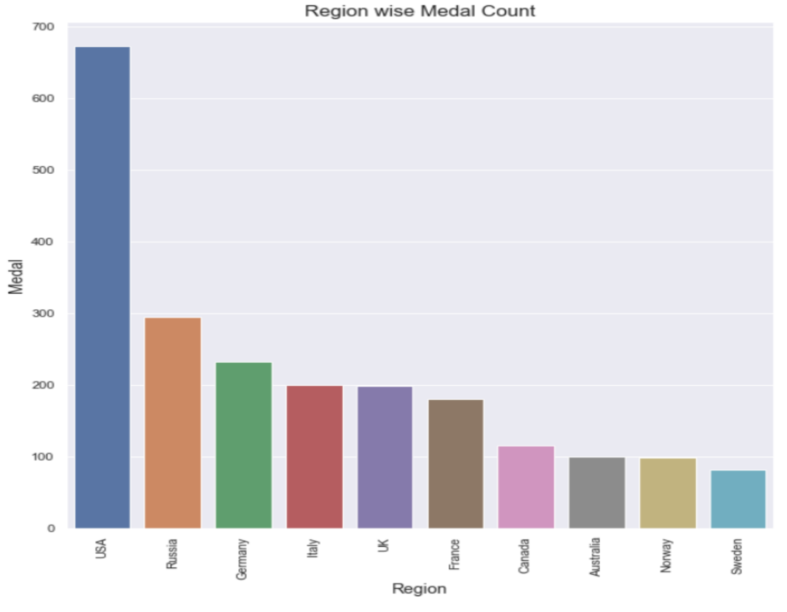
In [58]:
```
gold_country = ath_df[ath_df.Medal == 'Gold'].groupby(['Region']).Medal.size()
gold_top_10 = gold_country.sort_values(ascending= False)[:10]
gold_top_10 = gold_top_10.reset_index()
gold_top_10
```

Out[58]:

| | Region | Medal |
|---|---|---|
| 0 | USA | 673 |
| 1 | Russia | 295 |
| 2 | Germany | 232 |
| 3 | Italy | 200 |
| 4 | UK | 199 |
| 5 | France | 181 |
| 6 | Canada | 115 |
| 7 | Australia | 100 |
| 8 | Norway | 99 |
| 9 | Sweden | 82 |

In [59]:
```
#barplot
plt.figure(figsize=(10,10))
sns.barplot(x=gold_top_10['Region'],y=gold_top_10['Medal'])
plt.title('Region wise Medal Count',size=16)
plt.xlabel('Region',size=14)
plt.ylabel('Medal',size=14)
plt.xticks(rotation=90)

plt.show()
```




USA won most gold medal in all olympic season i.e - 2638

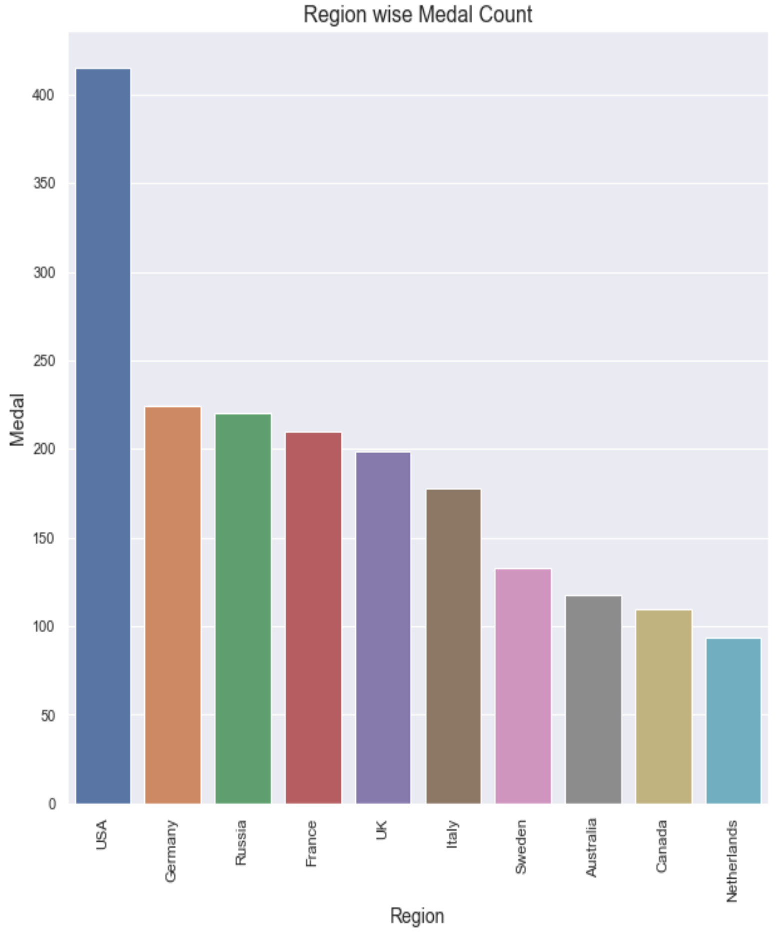
## Top 10 country who won silver medal

In [60]:
```
silver_country = ath_df[ath_df.Medal == 'Silver'].groupby(['Region']).Medal.size()
silver_top_10 = silver_country.sort_values(ascending= False)[:10]
silver_top_10 = silver_top_10.reset_index()
silver_top_10
```

Out[60]:

| | Region | Medal |
|---|---|---|
| **0** | USA | 415 |
| **1** | Germany | 224 |
| **2** | Russia | 220 |
| **3** | France | 210 |
| **4** | UK | 199 |
| **5** | Italy | 178 |
| **6** | Sweden | 133 |
| **7** | Australia | 118 |
| **8** | Canada | 110 |
| **9** | Netherlands | 94 |

In [61]:
```
plt.figure(figsize=(10,10))
sns.barplot(x=silver_top_10['Region'],y=silver_top_10['Medal'])
plt.title('Region wise Medal Count',size=16)
```




USA won most silver medal in all season of olympics i.e-1641

## Top 10 country who won Bronze medal

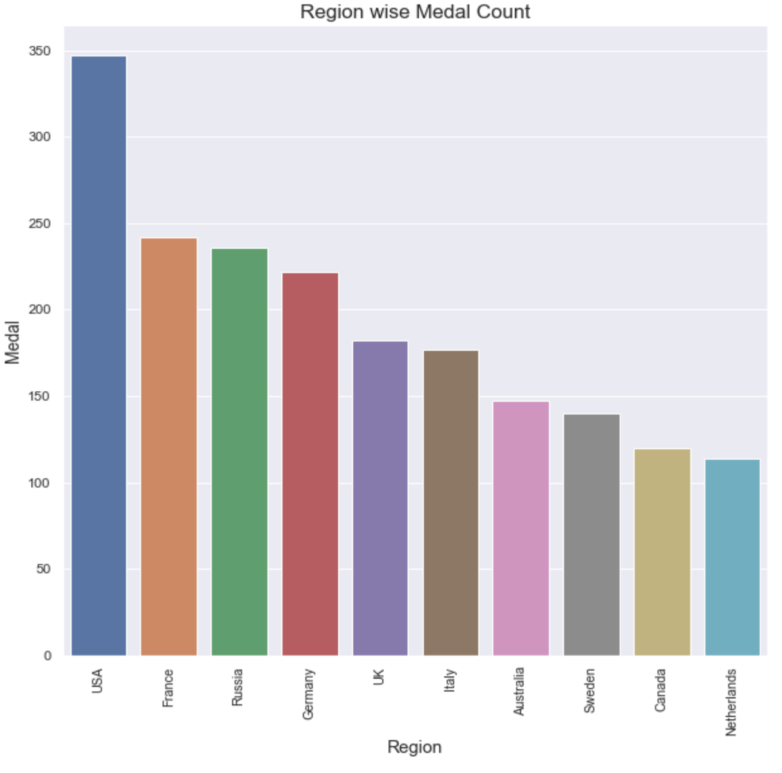
In [62]:
```
bronze_country = ath_df[ath_df.Medal == 'Bronze'].groupby(['Region']).Medal.size()
bronze_top_10 = bronze_country.sort_values(ascending= False)[:10]
bronze_top_10 = bronze_top_10.reset_index()
bronze_top_10
```

Out[62]:

| | Region | Medal |
|---|---|---|
| 0 | USA | 347 |
| 1 | France | 242 |
| 2 | Russia | 236 |
| 3 | Germany | 222 |
| 4 | UK | 182 |
| 5 | Italy | 177 |
| 6 | Australia | 147 |
| 7 | Sweden | 140 |
| 8 | Canada | 120 |
| 9 | Netherlands | 114 |

In [63]:
```
plt.figure(figsize=(10,10))
sns.barplot(x=bronze_top_10['Region'],y=bronze_top_10['Medal'])
plt.title('Region wise Medal Count',size=16)
plt.xlabel('Region',size=14)
plt.ylabel('Medal',size=14)
plt.xticks(rotation=90)

plt.show()
```




USA won most bronze medal in all olympics season i.e-1358

## Indian Gold Medalist list

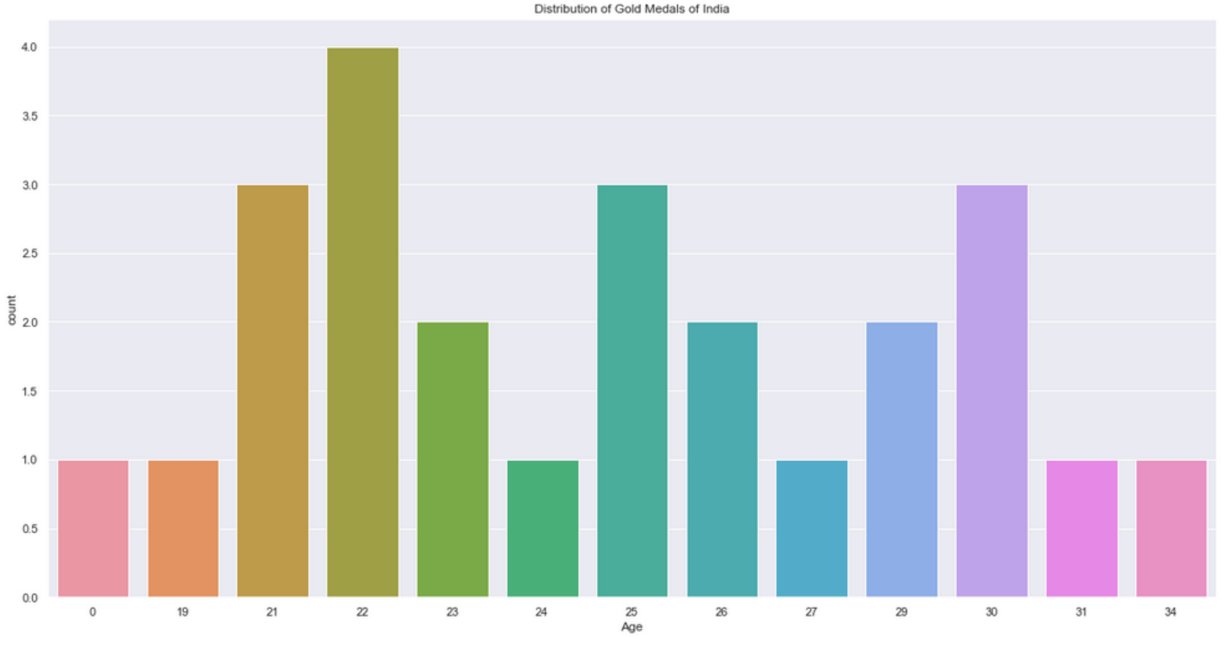
In [64]:
```
g_medal_India = ath_df[(ath_df.Medal == 'Gold') & (ath_df.Team == 'India')]
g_medal_India
```

Out[64]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **41191** | 2699 | Shaukat Ali | M | 30 | 0 | 0 | India | IND | 1928 Summer | 1928 | Summer | Amsterdam | Hockey | Hockey Men's Hockey | Gold | India |
| **41193** | 2703 | Syed Mushtaq Ali | M | 22 | 165 | 61 | India | IND | 1964 Summer | 1964 | Summer | Tokyo | Hockey | Hockey Men's Hockey | Gold | India |
| **41195** | 2864 | Richard James Allen | M | 25 | 172 | 0 | India | IND | 1928 Summer | 1928 | Summer | Amsterdam | Hockey | Hockey Men's Hockey | Gold | India |
| **41196** | 2864 | Richard James Allen | M | 30 | 172 | 0 | India | IND | 1932 Summer | 1932 | Summer | Los Angeles | Hockey | Hockey Men's Hockey | Gold | India |
| **41197** | 2864 | Richard James Allen | M | 34 | 172 | 0 | India | IND | 1936 Summer | 1936 | Summer | Berlin | Hockey | Hockey Men's Hockey | Gold | India |
| **41217** | 5618 | Sardar Mohammad Aslam | M | 0 | 0 | 0 | India | IND | 1932 Summer | 1932 | Summer | Los Angeles | Hockey | Hockey Men's Hockey | Gold | India |
| **41266** | 11197 | Vasudevan Bhaskaran | M | 29 | 174 | 68 | India | IND | 1980 Summer | 1980 | Summer | Moskva | Hockey | Hockey Men's Hockey | Gold | India |
| **41294** | 11601 | Abhinav Bindra | M | 25 | 173 | 70 | India | IND | 2008 Summer | 2008 | Summer | Beijing | Shooting | Shooting Men's Air Rifle, 10 metres | Gold | India |
| **41309** | 12911 | Lal Shah S. Bokhari | M | 23 | 173 | 0 | India | IND | 1932 Summer | 1932 | Summer | Los Angeles | Hockey | Hockey Men's Hockey | Gold | India |
| **41316** | 15011 | Frank Gerald Singlehurst Brewin | M | 22 | 0 | 0 | India | IND | 1932 Summer | 1932 | Summer | Los Angeles | Hockey | Hockey Men's Hockey | Gold | India |
| **41329** | 18475 | Richard John "Dickie" Carr | M | 21 | 180 | 0 | India | IND | 1932 Summer | 1932 | Summer | Los Angeles | Hockey | Hockey Men's Hockey | Gold | India |
| **41338** | 19716 | Dhyan Chand Bais | M | 22 | 169 | 0 | India | IND | 1928 Summer | 1928 | Summer | Amsterdam | Hockey | Hockey Men's Hockey | Gold | India |
| **41339** | 19716 | Dhyan Chand Bais | M | 26 | 169 | 0 | India | IND | 1932 Summer | 1932 | Summer | Los Angeles | Hockey | Hockey Men's Hockey | Gold | India |
| **41340** | 19716 | Dhyan Chand Bais | M | 30 | 169 | 0 | India | IND | 1936 Summer | 1936 | Summer | Berlin | Hockey | Hockey Men's Hockey | Gold | India |
| **41373** | 20565 | Bir Bahadur Chettri | M | 24 | 165 | 68 | India | IND | 1980 Summer | 1980 | Summer | Moskva | Hockey | Hockey Men's Hockey | Gold | India |

## Age wise Distribution of Indian Gold medalist

In [65]:
```
plt.figure(figsize=(20,10))
sns.countplot(g_medal_India['Age'])
plt.title('Distribution of Gold Medals of India')
plt.show()
```



## Name sport in which India won Gold Medal

In [66]:
```
sporting_event = g_medal_India['Sport']
sporting_event
```

Out[66]:
```
41191      Hockey
41193      Hockey
41195      Hockey
41196      Hockey
41197      Hockey
41217      Hockey
41266      Hockey
41294    Shooting
41309      Hockey
41316      Hockey
41329      Hockey
41338      Hockey
41339      Hockey
41340      Hockey
41373      Hockey
41381      Hockey
41382      Hockey
41383      Hockey
41389      Hockey
41392      Hockey
41404      Hockey
41405      Hockey
41439      Hockey
41440      Hockey
41455      Hockey
Name: Sport, dtype: object
```

# Indian Silver Medalist list

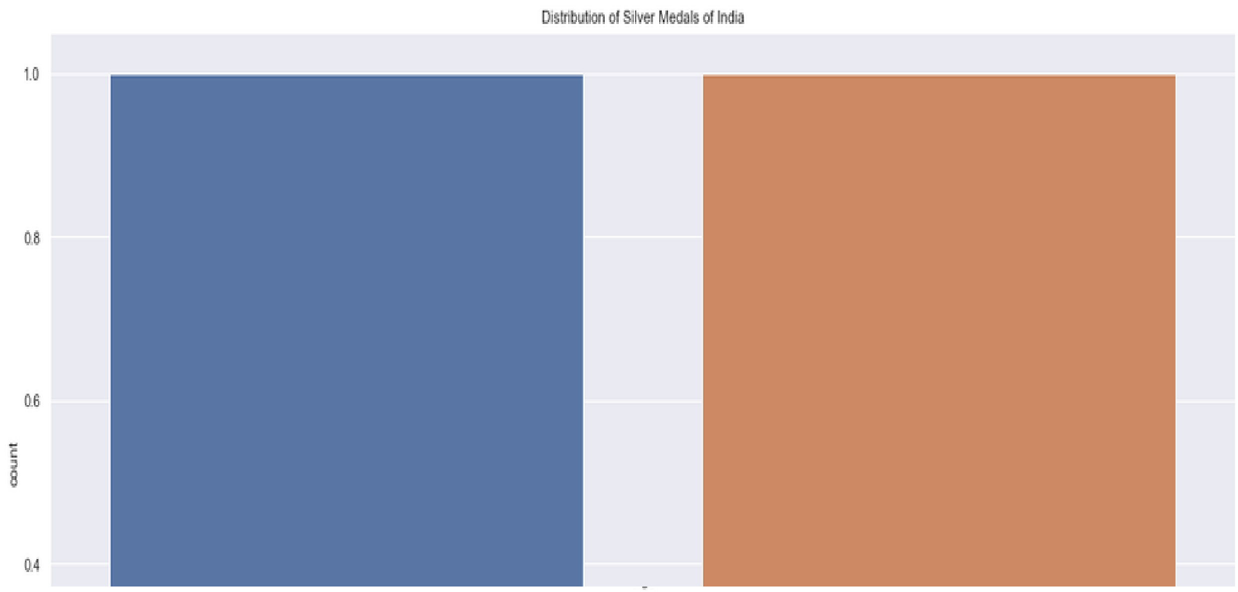
In [67]:
```
s_medal_India = ath_df[(ath_df.Medal == 'Silver') & (ath_df.Team == 'India')]
s_medal_India
```

Out[67]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **41209** | 4518 | Joseph Anthony "Joe" Antic | M | 29 | 168 | 59 | India | IND | 1960 Summer | 1960 | Summer | Roma | Hockey | Hockey Men's Hockey | Silver | India |
| **41384** | 21912 | Leslie Walter Claudius | M | 33 | 162 | 53 | India | IND | 1960 Summer | 1960 | Summer | Roma | Hockey | Hockey Men's Hockey | Silver | India |

# Age wise Distribution of Indian Silver medalist

In [68]:
```
plt.figure(figsize=(20,10))
sns.countplot(s_medal_India['Age'])
plt.title('Distribution of Silver Medals of India')
plt.show()
```




# Name sport in which India won Silver Medal

In [69]:
```
sporting_event = s_medal_India['Sport']
sporting_event
```

Out[69]:
```
41209    Hockey
41384    Hockey
Name: Sport, dtype: object
```

# Indian Bronze Medalist list

In [70]:
```
b_medal_India = ath_df[(ath_df.Medal == 'Bronze') & (ath_df.Team == 'India')]
b_medal_India
```

Out[70]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **41289** | 11520 | Govinda Billimogaputtaswamy | M | 20 | 171 | 60 | India | IND | 1972 Summer | 1972 | Summer | Munich | Hockey | Hockey Men's Hockey | Bronze | India |
| **41378** | 21339 | Rajendra Absolem Christy | M | 30 | 165 | 58 | India | IND | 1968 Summer | 1968 | Summer | Mexico City | Hockey | Hockey Men's Hockey | Bronze | India |
| **41385** | 23098 | Charles Cornelius | M | 26 | 170 | 65 | India | IND | 1972 Summer | 1972 | Summer | Munich | Hockey | Hockey Men's Hockey | Bronze | India |
| **41444** | 30913 | Yogeshwar Dutt | M | 29 | 168 | 65 | India | IND | 2012 Summer | 2012 | Summer | London | Wrestling | Wrestling Men's Lightweight, Freestyle | Bronze | India |

# Age wise Distribution of Indian Bronze medalist

In [71]:
```
plt.figure(figsize=(20,10))
sns.countplot(b_medal_India['Age'])
plt.title('Distribution of Bronze Medals of India')
plt.show()
```


# Name sport in which India won Bronze Medal

In [72]:
```
sporting_event = b_medal_India['Sport']
sporting_event
```

Out[72]:
```
41289       Hockey
41378       Hockey
41385       Hockey
41444    Wrestling
Name: Sport, dtype: object
```

## Prediction

In [73]:
```
ath
```

Out[73]:

| | ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Season | City | Sport | Event | Medal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | A Dijiang | M | 24.0 | 180.0 | 80.0 | China | CHN | 1992 Summer | 1992 | Summer | Barcelona | Basketball | Basketball Men's Basketball | No Medal |
| **1** | 2 | A Lamusi | M | 23.0 | 170.0 | 60.0 | China | CHN | 2012 Summer | 2012 | Summer | London | Judo | Judo Men's Extra-Lightweight | No Medal |
| **2** | 3 | Gunnar Nielsen Aaby | M | 24.0 | NaN | NaN | Denmark | DEN | 1920 Summer | 1920 | Summer | Antwerpen | Football | Football Men's Football | No Medal |
| **3** | 4 | Edgar Lindenau Aabye | M | 34.0 | NaN | NaN | Denmark/Sweden | DEN | 1900 Summer | 1900 | Summer | Paris | Tug-Of-War | Tug-Of-War Men's Tug-Of-War | Gold |
| **4** | 5 | Christine Jacoba Aaftink | F | 21.0 | 185.0 | 82.0 | Netherlands | NED | 1988 Winter | 1988 | Winter | Calgary | Speed Skating | Speed Skating Women's 500 metres | No Medal |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **65530** | 33537 | Nelson vora | M | 24.0 | 183.0 | 76.0 | Portugal | POR | 2008 Summer | 2008 | Summer | Beijing | Athletics | Athletics Men's Triple Jump | Gold |
| **65531** | 33537 | Nelson vora | M | 32.0 | 183.0 | 76.0 | Portugal | POR | 2016 Summer | 2016 | Summer | Rio de Janeiro | Athletics | Athletics Men's Triple Jump | No Medal |
| **65532** | 33538 | Joseph Evouna | M | 19.0 | 172.0 | 69.0 | Cameroon | CMR | 1972 Summer | 1972 | Summer | Munich | Cycling | Cycling Men's Road Race, Individual | No Medal |
| **65533** | 33538 | Joseph Evouna | M | 19.0 | 172.0 | 69.0 | Cameroon | CMR | 1972 Summer | 1972 | Summer | Munich | Cycling | Cycling Men's 100 kilometres Team Time Trial | No Medal |
| **65534** | 33538 | Joseph Evouna | M | 27.0 | 172.0 | 69.0 | Cameroon | CMR | 1980 Summer | 1980 | Summer | Moskva | Cycling | Cycling Men's Road Race, Individual | No Medal |

65535 rows × 15 columns

In [74]:
```
new = ath[['Sex','Age','Height','Weight','Season','Medal']]
new
```

Out[74]:

| | Sex | Age | Height | Weight | Season | Medal |
|---|---|---|---|---|---|---|
| **0** | M | 24.0 | 180.0 | 80.0 | Summer | No Medal |
| **1** | M | 23.0 | 170.0 | 60.0 | Summer | No Medal |
| **2** | M | 24.0 | NaN | NaN | Summer | No Medal |
| **3** | M | 34.0 | NaN | NaN | Summer | Gold |
| **4** | F | 21.0 | 185.0 | 82.0 | Winter | No Medal |
| **...** | ... | ... | ... | ... | ... | ... |
| **65530** | M | 24.0 | 183.0 | 76.0 | Summer | Gold |
| **65531** | M | 32.0 | 183.0 | 76.0 | Summer | No Medal |
| **65532** | M | 19.0 | 172.0 | 69.0 | Summer | No Medal |
| **65533** | M | 19.0 | 172.0 | 69.0 | Summer | No Medal |
| **65534** | M | 27.0 | 172.0 | 69.0 | Summer | No Medal |

65535 rows × 6 columns

# Checking Null Values

In [75]:
```
new.isna().sum()
```

Out[75]:
```
Sex           0
Age        2548
Height    15289
Weight    16079
Season        0
Medal         0
dtype: int64
```

# Replace Values

In [76]:
```
new.Medal=new.Medal.replace({'No Medal':1,'Gold':2,'Silver':3,'Bronze':4})
new.Season=new.Season.replace({'Summer':1,'Winter':0})
new.Sex=new.Sex.replace({'M':7,'F':9})
```

# Checking datatypes

In [77]:
```
new.dtypes
```

Out[77]:
```
Sex         int64
Age       float64
Height    float64
Weight    float64
Season      int64
Medal       int64
dtype: object
```

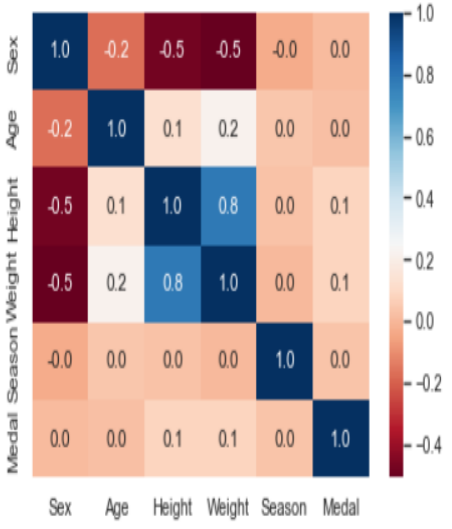
# Heatmap

In [78]:
```
new.corr()
```

Out[78]:

| | Sex | Age | Height | Weight | Season | Medal |
|---|---|---|---|---|---|---|
| **Sex** | 1.000000 | -0.174068 | -0.481459 | -0.507355 | -0.034339 | 0.005489 |
| **Age** | -0.174068 | 1.000000 | 0.126696 | 0.211782 | 0.036862 | 0.022433 |
| **Height** | -0.481459 | 0.126696 | 1.000000 | 0.782915 | 0.027195 | 0.080316 |
| **Weight** | -0.507355 | 0.211782 | 0.782915 | 1.000000 | 0.000170 | 0.079173 |
| **Season** | -0.034339 | 0.036862 | 0.027195 | 0.000170 | 1.000000 | 0.030842 |
| **Medal** | 0.005489 | 0.022433 | 0.080316 | 0.079173 | 0.030842 | 1.000000 |

In [79]:
```
sns.heatmap(new.corr(),cmap='RdBu',annot=True,fmt='.1f')
plt.show()
```




We can see there is a positive correlation between Sex(Predictor) & Medal This makes sense since, The greater Sex results in a greater chance of medal.

In addition, we see a negative correlation between season & our predictor i.e-sex.

## Seprating Dependent and Indeoendent Variable

In [80]:
```
new
```

Out[80]:

| | Sex | Age | Height | Weight | Season | Medal |
|---|---|---|---|---|---|---|
| **0** | 7 | 24.0 | 180.0 | 80.0 | 1 | 1 |
| **1** | 7 | 23.0 | 170.0 | 60.0 | 1 | 1 |
| **2** | 7 | 24.0 | NaN | NaN | 1 | 1 |
| **3** | 7 | 34.0 | NaN | NaN | 1 | 2 |
| **4** | 9 | 21.0 | 185.0 | 82.0 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... |
| **65530** | 7 | 24.0 | 183.0 | 76.0 | 1 | 2 |
| **65531** | 7 | 32.0 | 183.0 | 76.0 | 1 | 1 |
| **65532** | 7 | 19.0 | 172.0 | 69.0 | 1 | 1 |
| **65533** | 7 | 19.0 | 172.0 | 69.0 | 1 | 1 |
| **65534** | 7 | 27.0 | 172.0 | 69.0 | 1 | 1 |

65535 rows × 6 columns

In [81]:
```
new.isna().sum()
```

Out[81]:
```
Sex           0
Age        2548
Height    15289
Weight    16079
Season        0
Medal         0
dtype: int64
```

In [82]:
```
new.dropna(inplace=True)
```

In [83]:
```
new.isna().sum()
```

Out[83]:
```
Sex       0
Age       0
Height    0
Weight    0
Season    0
Medal     0
dtype: int64
```

In [84]:
```
x = new.drop(['Sex'],axis=1)
y = new.Sex
```

In [85]:
```
x
```

Out[85]:

| | Age | Height | Weight | Season | Medal |
|---|---|---|---|---|---|
| **0** | 24.0 | 180.0 | 80.0 | 1 | 1 |
| **1** | 23.0 | 170.0 | 60.0 | 1 | 1 |
| **4** | 21.0 | 185.0 | 82.0 | 0 | 1 |
| **5** | 21.0 | 185.0 | 82.0 | 0 | 1 |
| **6** | 25.0 | 185.0 | 82.0 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... |
| **65530** | 24.0 | 183.0 | 76.0 | 1 | 2 |
| **65531** | 32.0 | 183.0 | 76.0 | 1 | 1 |
| **65532** | 19.0 | 172.0 | 69.0 | 1 | 1 |
| **65533** | 19.0 | 172.0 | 69.0 | 1 | 1 |
| **65534** | 27.0 | 172.0 | 69.0 | 1 | 1 |

48894 rows × 5 columns

In [86]:
```
y
```

Out[86]:
```
0        7
1        7
4        9
5        9
6        9
        ..
65530    7
65531    7
65532    7
65533    7
65534    7
Name: Sex, Length: 48894, dtype: int64
```

In [87]:
```
y.value_counts()
```

Out[87]:
```
7    33855
9    15039
Name: Sex, dtype: int64
```

# Import train-test Split

In [88]:
```
from sklearn.model_selection import train_test_split
```

In [89]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [90]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(39115, 5)
(9779, 5)
(39115,)
(9779,)
```

# 1 - Logistic Regression

```
In [91]: from sklearn.linear_model import LogisticRegression

In [92]: model = LogisticRegression()

In [93]: model

Out[93]: LogisticRegression()

In [94]: model.fit(x_train,y_train)

Out[94]: LogisticRegression()

In [95]: model.score(x_train,y_train)*100

Out[95]: 81.04052153905151

In [96]: model.score(x_test,y_test)*100

Out[96]: 79.84456488393496

In [97]: y_predict = model.predict(x_test)

In [98]: y_predict

Out[98]: array([7, 7, 7, ..., 7, 7, 7], dtype=int64)

In [99]: new = pd.DataFrame({'Actual':y_test,'Predicted':y_predict})

In [100]: new

Out[100]:
```

| | Actual | Predicted |
|---|---|---|
| **53264** | 7 | 7 |
| **2773** | 7 | 7 |
| **22381** | 9 | 7 |
| **31584** | 7 | 9 |
| **46483** | 9 | 9 |
| **...** | ... | ... |
| **7536** | 9 | 9 |
| **29145** | 7 | 7 |
| **1748** | 7 | 7 |
| **42555** | 7 | 7 |
| **11467** | 7 | 7 |

9779 rows × 2 columns

```
In [101]: # checking accuracy
          from sklearn.metrics import accuracy_score

In [102]: test_acc = accuracy_score(y_test,y_predict)*100

In [103]: test_acc

Out[103]: 79.84456488393496
```

# Importing confusion matrix

In [104]:
```
from sklearn.metrics import confusion_matrix
```

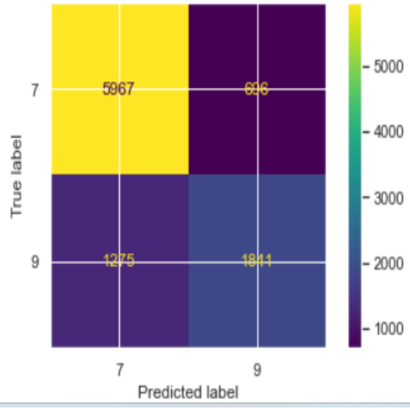
In [105]:
```
performance = confusion_matrix(y_test,y_predict)
```

In [106]:
```
performance
```

Out[106]:
```
array([[5967,  696],
       [1275, 1841]], dtype=int64)
```

In [107]:
```
from sklearn.metrics import plot_confusion_matrix
```

In [108]:
```
plot_confusion_matrix(model,x_test,y_test)
plt.show()
```




6146 is the amount of True Positives in our data, while 1773 is the amount of True Negatives.

662 & 1198 are the number of errors.

There are 662 type 1 error (False Positives)- You predicted positive and it's false.

There are 1198 type 2 error (False Negatives)- You predicted negative and it's false.

Hence if we calculate the accuracy its # Correct Predicted/ # Total. In other words, where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives.

(TP + TN)/(TP + TN + FP + FN).(6146+1773)/(6146+1773+662+1198)=0.809796 = 80.97% accuracy

Accuracy = 80.97%

# Classification report

In [109]:
```
from sklearn.metrics import classification_report
```

In [110]:
```
performance = classification_report(y_test,y_predict)
```

In [111]:
```
print(performance)
```

```
              precision    recall  f1-score   support

           7       0.82      0.90      0.86      6663
           9       0.73      0.59      0.65      3116

    accuracy                           0.80      9779
   macro avg       0.77      0.74      0.75      9779
weighted avg       0.79      0.80      0.79      9779
```

# 2 - Decision Tree

```
In [112]: from sklearn.tree import DecisionTreeClassifier

In [113]: model_dt = DecisionTreeClassifier()

In [114]: model_dt.fit(x_train,y_train)

Out[114]: DecisionTreeClassifier()

In [115]: model_dt.score(x_train,y_train)*100

Out[115]: 93.33503770931868

In [116]: model_dt.score(x_test,y_test)*100

Out[116]: 81.65456590653442

In [117]: from sklearn.metrics import accuracy_score

In [118]: y_predict = model_dt.predict(x_test)

In [119]: y_predict

Out[119]: array([7, 7, 7, ..., 7, 7, 7], dtype=int64)

In [120]: test_acc = accuracy_score(y_test,y_predict)*100

In [121]: test_acc

Out[121]: 81.65456590653442
```

# 3 - Random Forest

In [122]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [123]:
```
model_rf = RandomForestClassifier(n_estimators=100)
```

In [124]:
```
model_rf.fit(x_train,y_train)
```

Out[124]:
```
RandomForestClassifier()
```

In [125]:
```
model_rf.score(x_train,y_train)*100
```

Out[125]:
```
93.33248114534067
```

In [126]:
```
model_rf.score(x_test,y_test)*100
```

Out[126]:
```
82.99417118314756
```

In [127]:
```
from sklearn.metrics import accuracy_score
```

In [128]:
```
y_predict = model_rf.predict(x_test)
```

In [129]:
```
y_predict
```

Out[129]:
```
array([7, 7, 7, ..., 7, 7, 7], dtype=int64)
```

In [130]:
```
test_acc = accuracy_score(y_test,y_predict)*100
```

In [131]:
```
test_acc
```

Out[131]:
```
82.99417118314756
```

# 4- KNeighborsClassifier

In [132]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [133]:
```
model_knn = KNeighborsClassifier(n_neighbors=10)
```

In [134]:
```
model_knn.fit(x_train,y_train)
```

Out[134]: KNeighborsClassifier(n_neighbors=10)

In [135]:
```
model_knn.score(x_train,y_train)*100
```

Out[135]: 84.42796880991948

In [136]:
```
model_knn.score(x_test,y_test)*100
```

Out[136]: 80.7444523979957

In [137]:
```
from sklearn.metrics import accuracy_score
```

In [138]:
```
y_predict=model_knn.predict(x_test)
```

In [139]:
```
y_predict
```

Out[139]: array([7, 7, 7, ..., 7, 7, 7], dtype=int64)

In [140]:
```
test_acc = accuracy_score(y_test,y_predict)*100
```

In [141]:
```
test_acc
```

Out[141]: 80.7444523979957

# 5 - Naive bayes

```
In [142]: from sklearn.naive_bayes import MultinomialNB
```

```
In [143]: model_nb = MultinomialNB()
```

```
In [144]: model_nb.fit(x_train,y_train)

Out[144]: MultinomialNB()
```

```
In [145]: model_nb.score(x_train,y_train)*100

Out[145]: 78.47884443308195
```

```
In [146]: model_nb.score(x_test,y_test)*100

Out[146]: 77.57439410982718
```

```
In [147]: from sklearn.metrics import accuracy_score
```

```
In [148]: y_predict = model_nb.predict(x_test)
```

```
In [149]: y_predict

Out[149]: array([7, 7, 7, ..., 7, 7, 7], dtype=int64)
```

```
In [150]: test_acc = accuracy_score(y_test,y_predict)*100
```

```
In [151]: test_acc

Out[151]: 77.57439410982718
```

# Chapter – 4

# Analysis of result :

We used precision, F1-score, recall and accuracy evaluation metrics for evaluating our models.

False Positive(FP) is when a model incorrectly predicts a positive outcome.

False Negative(FN) is when a model incorrectly predicts the negative outcome.

True Positive(TP) is when model correctly predicts a positive outcome.

True Negative(TN) is when a model correctly predicts a negative outcome.

Precision=TP/(TP+FP) =

Recall = TP / (TP + FN)

F1 score = 2 * precision * Recall / (precision + Recall)

Machine Learning Models - Accuracy

1 - Logistic Regression – 81.03

2 - Decision Tree – 82.15

3 - Random Forest – 83.14

4 – Kneighbors Classifier – 81.80

5 - Naive bayes – 78.43

# Chapter – 5

## Conclusion :

**Our Random Forest algorithm yields the highest accuracy, 83.14 %. Any accuracy above 80% is considered good. Thus, 82.98 % is the ideal accuracy! and Random forest model has Highest testing score i.e-83.147% and score of train data is 88.5457%.**

**Out of the 17 features we examined the 5 features i. e- Age , Height, Weight ,Season and Medal features that helped us classify between a Male & Female (Sex).**

**As we are getting high accuracy in training as well as testing data set , it is example of Right fit.**

# Reference :


[1] 120 years of Olympic Dataset, Available:
https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-athletes-and-results

[2] Wikipedia:
https://en.wikipedia.org/wiki/Olympic_Games

[3] Video reference : 1:
https://www.youtube.com/watch?v=q1FttL_G1G4&ab_channel=Simplilearn

2:
https://www.youtube.com/watch?v=JmBRfApfnz8&ab_channel=BoardInfinity