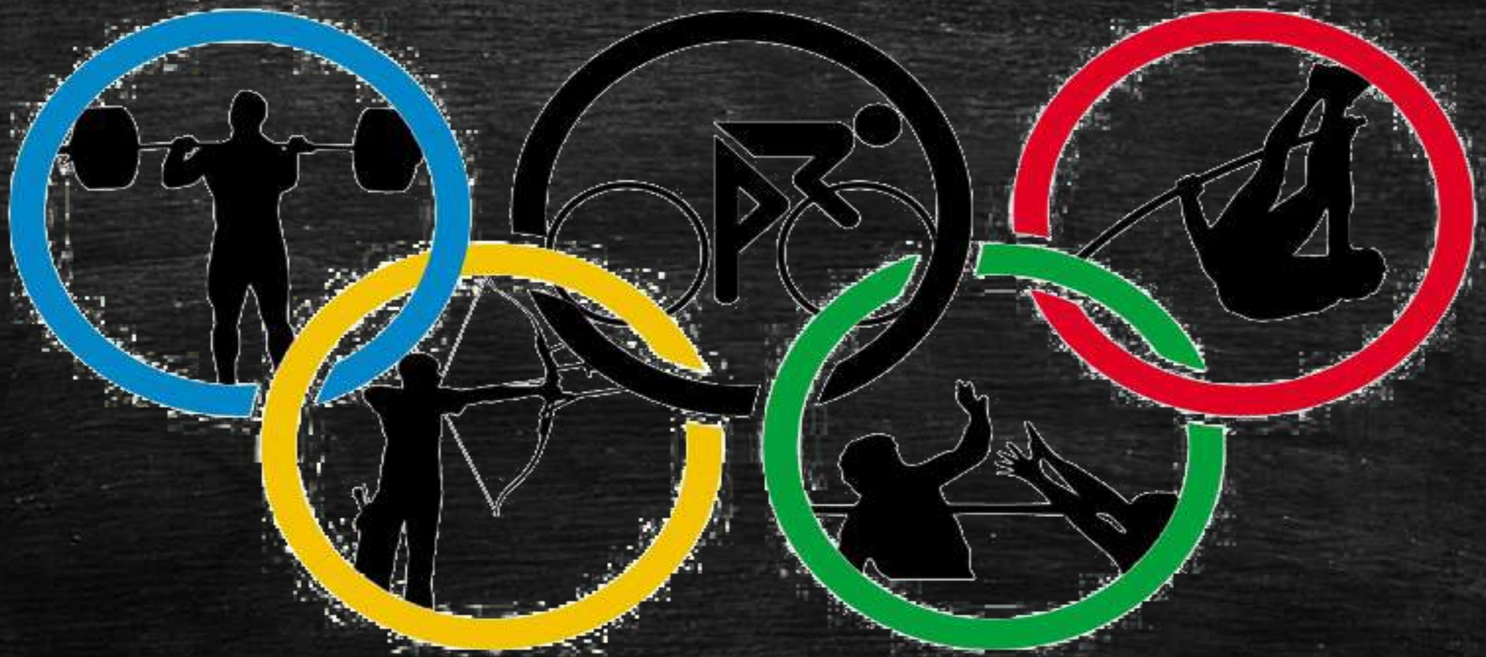
STATISTICS FOR DATA SCIENCE



ATHLETE DATASET.

CONTENT OUTLINE
Topics

- INTRODUCTION
- LIBRARIES USED
- DATA CLEANING
- DATA VISUALISATION
- NORMALISATION
- STANDARDISATION
- HYPOTHESIS TESTING
- CORRELATION
- STUDENT PROFILE

INTRODUCTION

The dataset chosen by the team is about the Olympic history from 1896 to 2016. It is a basic bio data on athletes and the medal result from Athens 1896 to Rio 2016 of all the athletes participated in summer and winter Olympics. Each row corresponds to an individual athlete competing in an individual Olympic event (athlete-events).

We tried to make simple observations and draw conclusions from the given dataset. Observations regarding medals won by athletes and countries as a whole and gender ration od winners etc. We used visualising techniques and hypothesis and testing to help us do what we aimed to complete.

BRIEF OVERVIEW

COLUMN NAME	EXPLANATION
Column ID:	stands for a unique number for each athlete.
Name:	stands for the athlete's name.
Sex	M or F
Age:	stands for the athlete's age in integer.
Height:	stands for the athlete's height in cm unit.
Weight:	stands for the athlete's weight in kg unit.
Team	stands for the athlete's nationality which he/she represent.
NOC	stands for National Olympic Committee which is a 3 letter code.
Game	stands for year and season which the athlete participated at.
Year	stands for the year of the event the athlete participated at.
Season	Winter or Summer
City	stands for the hoist city the event was conducted at.
Sport	stands for sport athlete participated at.
Event	stands for which category of event the athlete participated
Medal	stands for the medal the athlete received if any.

IMPORTING LIBRARIES

- PANDAS

It provides highly optimized performance with back-end source code is purely written in C or Python

- NUMPY

It extends python into a high-level language for manipulating numerical data, similar to MATLAB.

- SEABORN

It is a library for making statistical graphics in Python.

MATPLOTLIB

Pyplot is a collection of functions in the popular visualization package Matplotlib. Its functions manipulate elements of a figure, such as creating a figure, creating a plotting area, plotting lines, adding plot labels, etc.

- SCIPY

It is used to solve scientific and mathematical problems

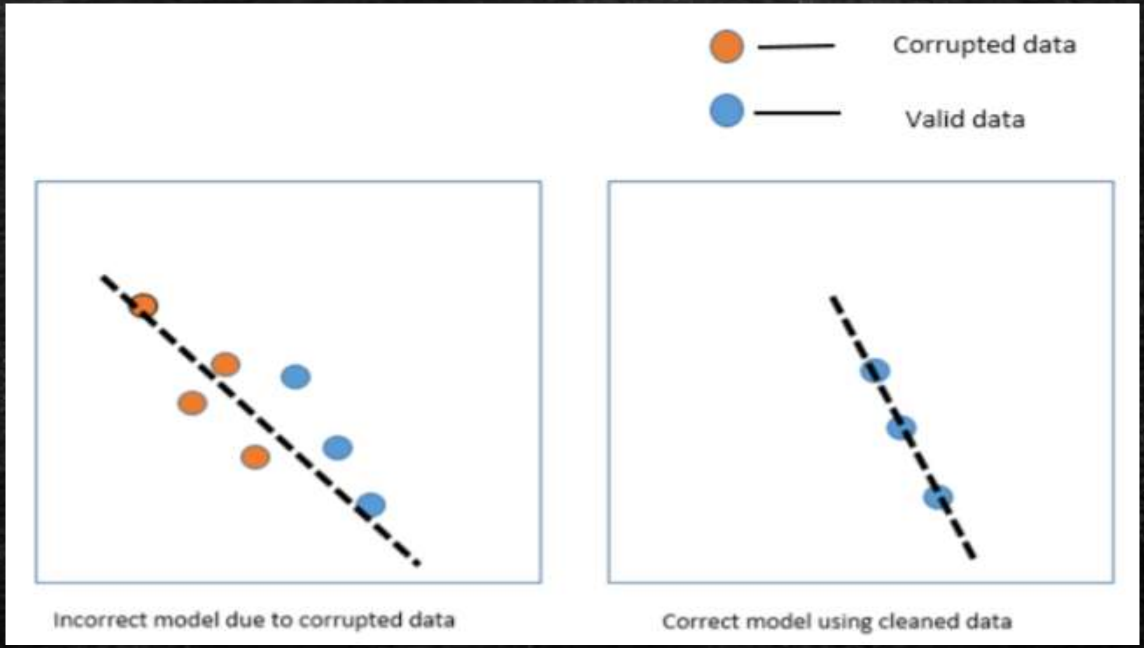
IMPORTING LIBRARIES

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

DATA CLEANING AND PREPROCESSING

Data Cleaning means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. Data is the most valuable thing for Analytics. When it comes to the real world data, it is not improbable that data may contain incomplete, inconsistent or missing values. If the data is corrupted then it may hinder the process or provide inaccurate results.

For example:



DATA CLEANING

1. First check for any missing data

```
In [4]: any(data.isnull())

Out[4]: True

In [5]: data.isnull().sum()

Out[5]: ID              0
        Name            0
        Sex             0
        Age          9474
        Height      60171
        Weight      62875
        Team            0
        NOC             0
        Games           0
        Year            0
        Season          0
        City            0
        Sport           0
        Event           0
        Medal      231333
        dtype: int64
```

Column ID, Name and Sex similarily have got no missing values.

```
In [8]: any(data['Age'].isnull())
```

```
Out[8]: True
```

```
In [9]: data['Age'].isnull().sum()
```

```
Out[9]: 9474
```

Column Age has got 9474 missing values. To deal with them, we shall replace the missing NaN values by the mode.

```
In [10]: median=data['Age'].median()
         data['Age'].fillna(median, inplace=True)
```

```
In [11]: any(data['Age'].isnull())
```

```
Out[11]: False
```

```
In [12]: data['Age'].isnull().sum()
```

```
Out[12]: 0
```

Column Age is replaced successfully, now similarily replace the NaN values from column Height and Weight with mean as its represents the entire dataset.

```
In [13]: mean=data['Height'].mean()
         data['Height'].fillna(mean,inplace=True)
         any(data['Height'].isnull())
```

```
Out[13]: False
```

```
In [14]: mean=data['Weight'].mean()
         data['Weight'].fillna(mean,inplace=True)
         data['Weight']
         any(data['Weight'].isnull())
```

```
Out[14]: False
```

DATA CLEANING

Since we choose a dataset on the modern olympics and particularily want to deal with the atheletes who have bagged a medal. We shall drop all the NaN values from column medal.

In [15]:
```
data=data.dropna()
```

In [16]:
```
data.isnull().sum()
```

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

DATA VISUALIZATION

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

DATA VISUALIZATION

Libraries needed for data visualization:

```
import matplotlib.pyplot as plt
import numpy as np
```

Various data visualization techniques shown in this project:

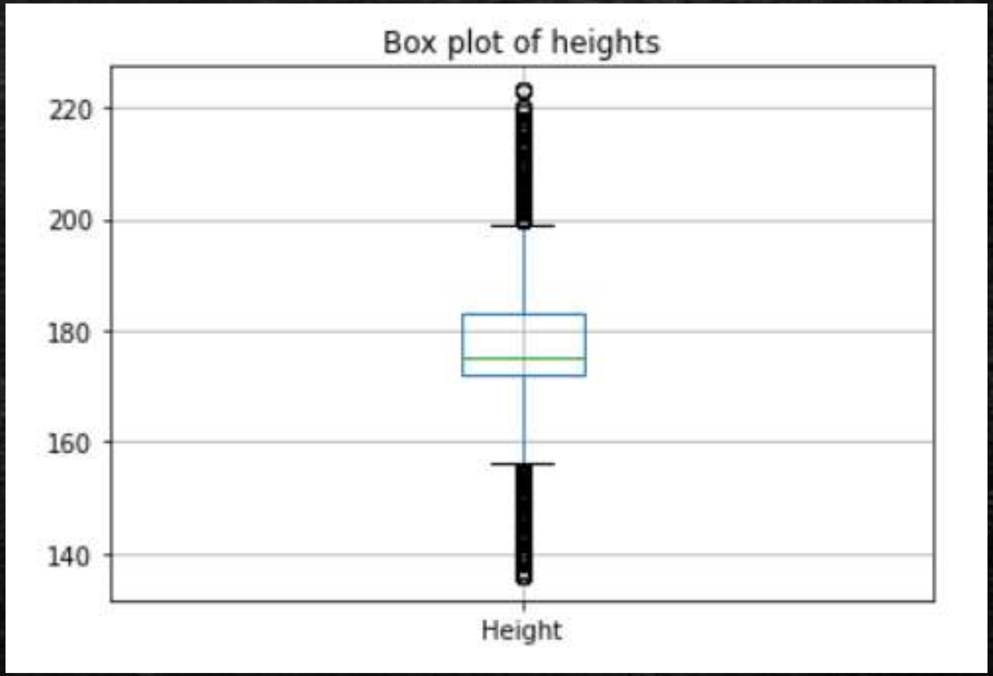
- Box plots
- Pie charts
- Stacked Bar charts
- Multi line graphs

<u>BOX PLOTS</u>- usually used to determine outliers

Here we used a boxplot to see if there are any outliers in the heights of the athletes:

```
Highest height: 223
Lowest height: 136
Mean height: 176.99492245431466
```

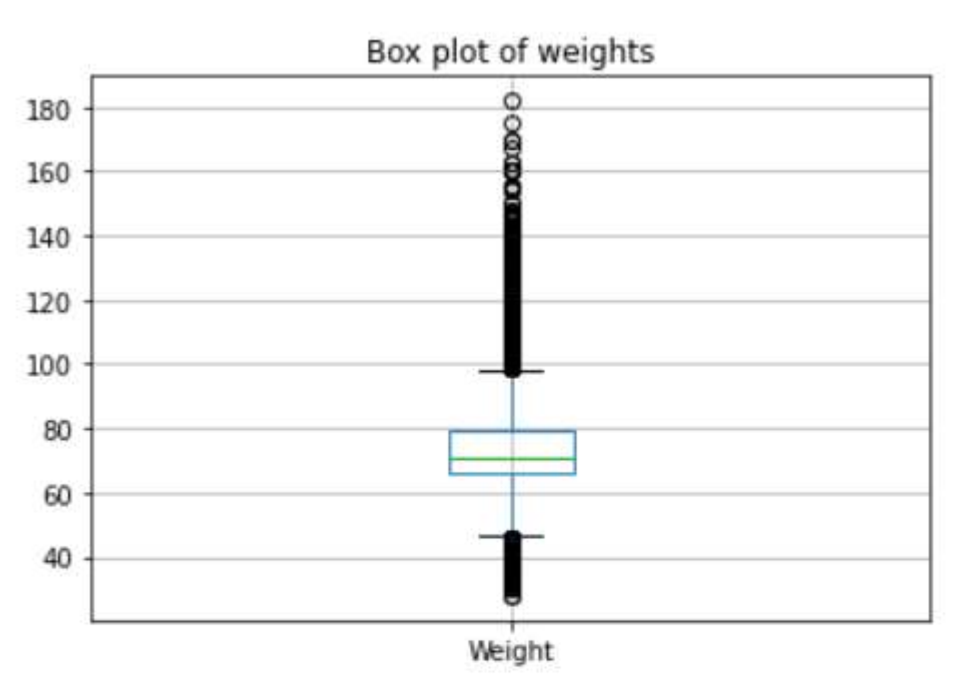
Here we see that there are a variety of heights among the athletes. The highest and lowest heights being 223 cm and 136 cm. The box plot proves the same and shows that there are quite a few outliers due to the variation in athlete heights



<u>BOX PLOTS</u>- usually used to determine outliers

Here we used a boxplot to see if there are any outliers in the weights of the athletes:

```
Highest weight: 182
Lowest weight: 28
Mean weight: 72.88402081291004
```



Here we observe how there are many outliers in the weight of the athletes with the highest and least weights among the athletes being 182 kg and 28 kg. While the mean weight is around 70 kg from the box plot and found to be 72.88 kg based on calculations.

PIE CHARTS – to determine percentage of medals from gold, silver and bronze categories

To find the top 10 countries with the highest aggregate of medals

```
data['Team'].value_counts().head(10)
```

```
United States     5219
Soviet Union      2451
Germany           1984
Great Britain     1673
France            1550
Italy             1527
Sweden            1434
Australia         1306
Canada            1243
Hungary           1127
Name: Team, dtype: int64
```

```
print("Total medals of top 3 countries")
data['Team'].value_counts().head(3)
```

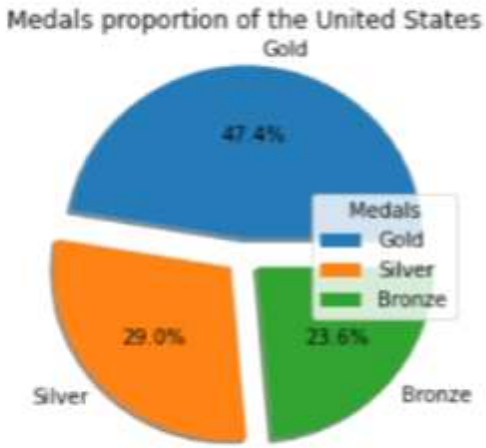
```
Total medals of top 3 countries

United States     5219
Soviet Union      2451
Germany           1984
Name: Team, dtype: int64
```

We find out the top 3 countries with the highest total medal aggregate. The medal percentages shall be analyzed for these 3 countries

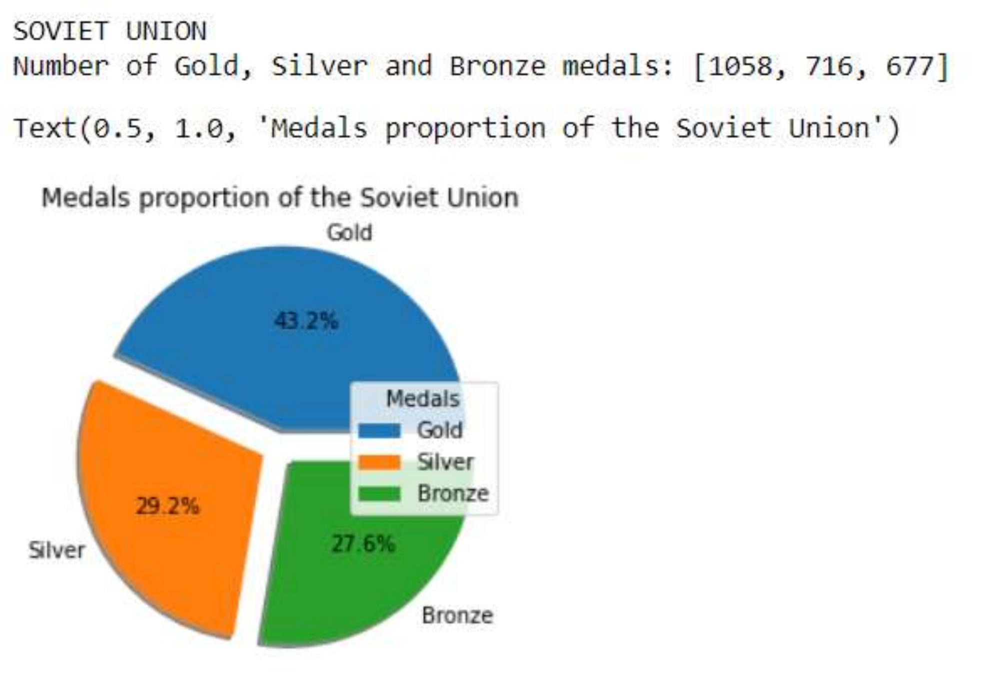
PIE CHARTS – continued..

```python
medals = [0,0,0]
print("UNITED STATES")
for i in data.index:
    if data['Team'][i] == "United States":
        if data['Medal'][i] == "Gold":
            medals[0] = medals[0]+1
        elif data['Medal'][i] == "Silver":
            medals[1] = medals[1]+1
        elif data['Medal'][i] == "Bronze":
            medals[2] = medals[2]+1
print ("Number of Gold, Silver and Bronze medals:",medals)

medals = np.array(medals)
label = ["Gold", "Silver", "Bronze"]
myexplode = [0.1, 0.1, 0.1]
plt.pie(medals, explode = myexplode, labels = label, shadow = True, autopct='%1.1f%%')
plt.legend(title = "Medals", loc = 5)
plt.title("Medals proportion of the United States")
```

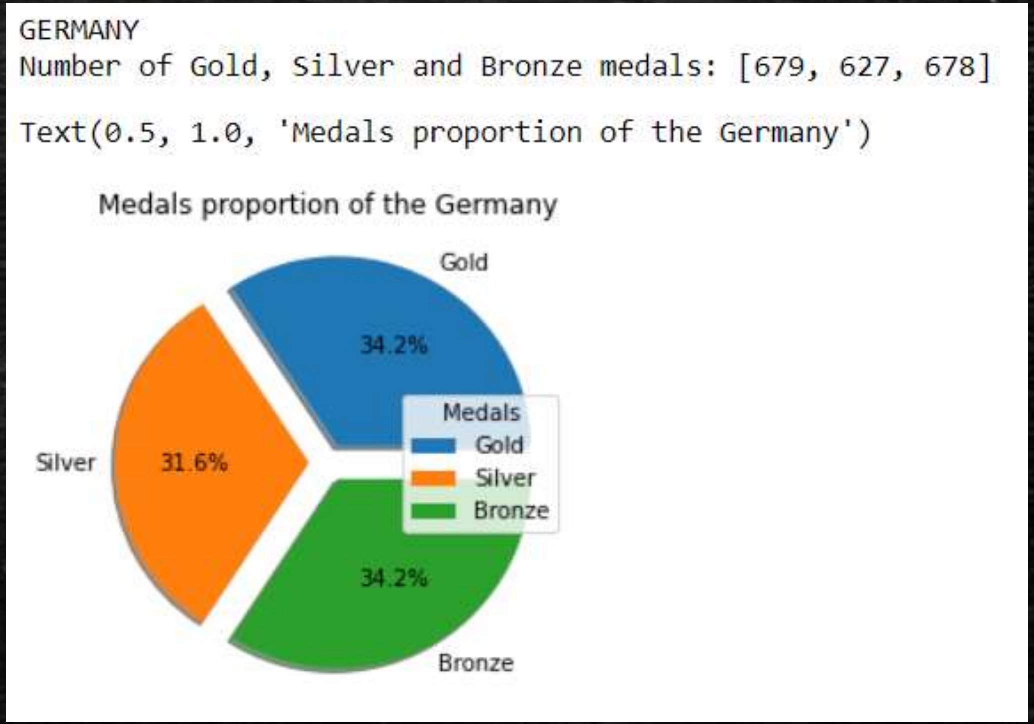
```
UNITED STATES
Number of Gold, Silver and Bronze medals: [2474, 1512, 1233]

Text(0.5, 1.0, 'Medals proportion of the United States')
```



PIE CHARTS – continued..



Similarly, we plot the graphs for the 2nd and 3rd countries.



The graphs are quite self- explanatory. We see that these countries have a more or less similar ratio of the 3 kinds of medals.

STACKED BAR GRAPH – to determine male-female winners ratio in top 10 countries

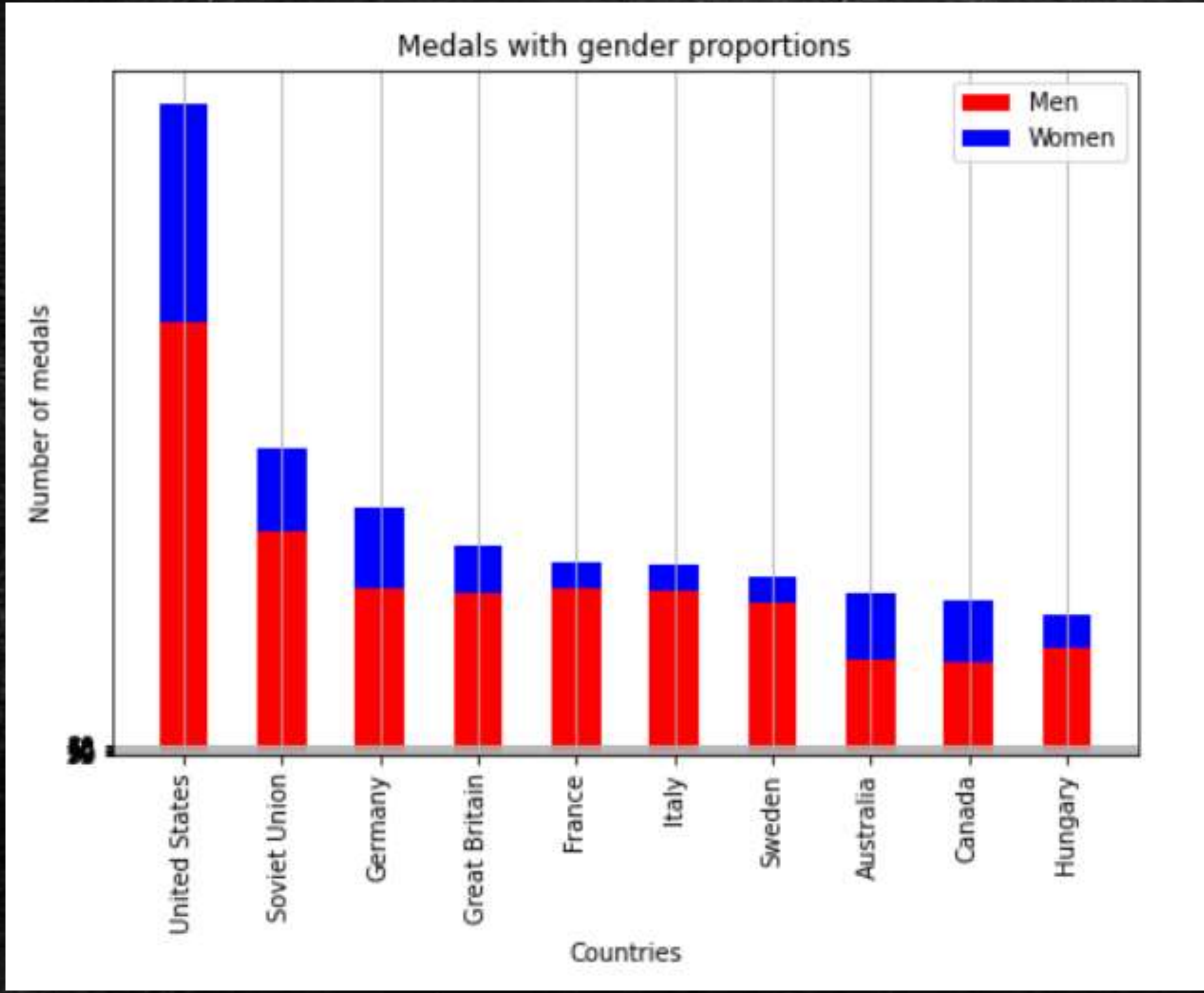
```
data['Team'].value_counts().head(10)
```

To get the top 10 countries

To plot the bar chart by creating a graph object ax

```
ind = np.arange(10) # the x locations for the groups
width = 0.5
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(ind, men, width, color='r')
ax.bar(ind, women, width,bottom=men, color='b')
ax.set_ylabel('Number of medals')
ax.set_xlabel('Countries')
ax.set_title('Medals with gender proportions')
plt.xticks(ind, ('United States','Soviet Union','Germany','Great Britain',
'France','Italy','Sweden','Australia','Canada','Hungary'),rotation = 90)
ax.grid()
ax.set_yticks(np.arange(0, 81, 10))
ax.legend(labels=['Men', 'Women'])
plt.show()
```

STACKED BAR GRAPH – continued..



Stacked bar graph of the top 10 countries

We see that:

- The number of medals won by the US males alone is higher than the number of total medals won by any other country
- The number of women medal winners is very less as compared to men in almost every country(especially Italy, France and Sweden)

LINE GRAPHS- to find the trend in medals received by any athlete specifically

```
data['Name'].value_counts().head(10)
```

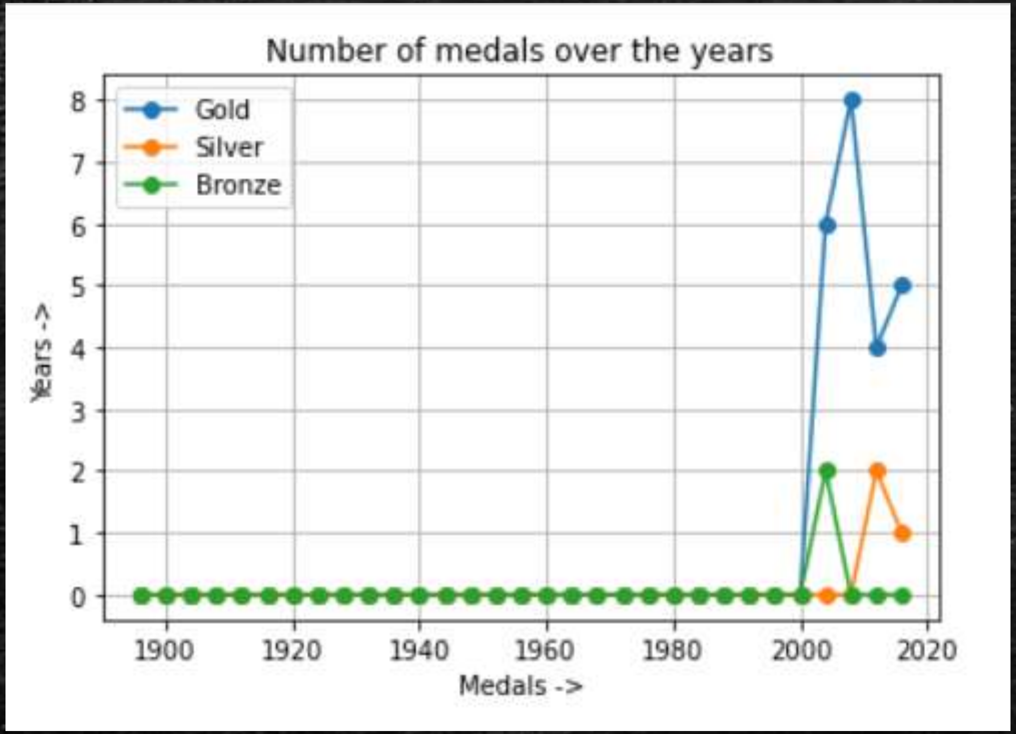
```
Michael Fred Phelps, II                   28
Larysa Semenivna Latynina (Diriy-)        18
Nikolay Yefimovich Andrianov              15
Edoardo Mangiarotti                       13
Ole Einar Bjrndalen                       13
Borys Anfiyanovych Shakhlin               13
Takashi Ono                               13
Paavo Johannes Nurmi                      12
Dara Grace Torres (-Hoffman, -Minas)      12
Natalie Anne Coughlin (-Hall)             12
Name: Name, dtype: int64
```

The top 10 athletes from the list of total medals won individually

To plot the line graph using the matplotlib library

```
plt.plot(years,g, label='Gold', marker = 'o')
plt.plot(years,s, label='Silver', marker = 'o')
plt.plot(years,b, label='Bronze', marker = 'o')
plt.grid()
plt.title("Number of medals over the years")
plt.xlabel("Medals ->")
plt.ylabel("Years ->")
plt.legend()
plt.show()
```

LINE GRAPHS – continued..



From the plot for Michael Phelps, we see that

- He started winning medals from the year 2000, which according to sources in the year he started taking part in the Olympics
- He won the maximum number of medals in the year 2008 and it was 8 gold medals.

DATA VISUALIZATION- conclusion

As we see, data visualization tools help us simply draw conclusions from a dataset containing SO MANY data values. They help in pictorially visualizing information which is any day simpler than having to go through all the data in a dataset. For example, we could see the trends in countries winning medals and the plot the male-female ratios of the top 10 countries etc., which wouldn't have been as simple without those graphical representations.

DATA NORMALISATION

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

The equation is shown below:

$$Z= x-min(x)/max(x) – min(x)$$

The goal of **normalization** is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

DATA NORMALISATION

In [31]:
```
df=data[data.Year>=1995]
s=len(df)
df.index=range(0,s,1)
df.columns
```

Out[31]:
```
Index(['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight', 'Team', 'NOC', 'Games',
       'Year', 'Season', 'City', 'Sport', 'Event', 'Medal'],
      dtype='object')
```

In [32]:
```
data.describe()
```

Out[32]:

	ID	Age	Height	Weight	Year
count	39783.000000	39783.000000	39783.000000	39783.000000	39783.000000
mean	69407.051806	25.918399	177.069144	73.051330	1973.943845
std	38849.980737	5.859573	9.670924	13.202504	33.822857
min	4.000000	10.000000	136.000000	28.000000	1896.000000
25%	36494.000000	22.000000	172.000000	66.000000	1952.000000
50%	68990.000000	25.000000	175.338970	70.702393	1984.000000
75%	103461.500000	29.000000	183.000000	79.000000	2002.000000
max	135563.000000	73.000000	223.000000	182.000000	2016.000000

DATA NORMALISATION

Getting the information of the data

In [33]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 39783 entries, 3 to 271103
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      39783 non-null  int64
 1   Name    39783 non-null  object
 2   Sex     39783 non-null  object
 3   Age     39783 non-null  float64
 4   Height  39783 non-null  float64
 5   Weight  39783 non-null  float64
 6   Team    39783 non-null  object
 7   NOC     39783 non-null  object
 8   Games   39783 non-null  object
 9   Year    39783 non-null  int64
 10  Season  39783 non-null  object
 11  City    39783 non-null  object
 12  Sport   39783 non-null  object
 13  Event   39783 non-null  object
 14  Medal   39783 non-null  object
dtypes: float64(3), int64(2), object(10)
memory usage: 6.1+ MB
```

In [34]:
```
data.var(axis=0)
```

Out[34]:
```
ID        1.509321e+09
Age       3.433459e+01
Height    9.352676e+01
Weight    1.743061e+02
Year      1.143986e+03
dtype: float64
```

DATA NORMALISATION

Getting mean and standard deviation of data.

```
In [35]: data.mean()

Out[35]: ID          69407.051806
         Age            25.918399
         Height        177.069144
         Weight         73.051330
         Year         1973.943845
         dtype: float64

In [36]: np.std(data)

Out[36]: ID          38849.492460
         Age             5.859499
         Height          9.670802
         Weight         13.202338
         Year           33.822432
         dtype: float64
```

DATA NORMALISATION

Here we are normalising the data

In [37]:

```
x = df[['Age','Height','Weight','Year']].values.astype(float)
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df_normalized = pd.DataFrame(x_scaled)
df_normalized.columns =['Age','Height','Weight','Year']
df_normalized["Name"]=df.Name
df_normalized["Event"]=df.Event
df_normalized["Sex"]=df.Sex
df_normalized["Team"]=df.Team
df_normalized["NOC"]=df.NOC
df_normalized["Games"]=df.Games
df_normalized["Season"]=df.Season
df_normalized["City"]=df.City
df_normalized["Sport"]=df.Sport
df_normalized["Medal"]=df.Medal
df_normalized = df_normalized[['Name','Sex','Age','Height','Weight','Team','NOC','Games','Year','Season','City','Sport','Event','
df_normalized
```

Out[37]:

	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	Juhamatti Tapio Aaltonen	M	0.312500	0.546512	0.387755	Finland	FIN	2014 Winter	0.9	Winter	Sochi	Ice Hockey	Ice Hockey Men's Ice Hockey	Bronze
1	Kjetil Andr Aamodt	M	0.354167	0.453488	0.387755	Norway	NOR	2002 Winter	0.3	Winter	Salt Lake City	Alpine Skiing	Alpine Skiing Men's Super G	Gold
2	Kjetil Andr Aamodt	M	0.354167	0.453488	0.387755	Norway	NOR	2002 Winter	0.3	Winter	Salt Lake City	Alpine Skiing	Alpine Skiing Men's Combined	Gold
3	Kjetil Andr Aamodt	M	0.437500	0.453488	0.387755	Norway	NOR	2006 Winter	0.5	Winter	Torino	Alpine Skiing	Alpine Skiing Men's Super G	Gold
4	Ragnhild Margrethe Aamodt	F	0.291667	0.302326	0.290492	Norway	NOR	2008 Summer	0.6	Summer	Beijing	Handball	Handball Women's Handball	Gold
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2016	[illegible]	[illegible]	Rio de	[illegible]	[illegible]	[illegible]

DATA NORMALISATION

The data we get after normalising.

```
In [38]: df_normalized.var()

Out[38]: Age       0.011069
         Height    0.017440
         Weight    0.011413
         Year      0.108615
         dtype: float64

In [39]: df_normalized.mean()

Out[39]: Age       0.274188
         Height    0.473379
         Weight    0.310885
         Year      0.510229
         dtype: float64
```

DATA STANDARDISATION

The result of standardization (or Z-score normalization) is that the features will be rescaled to ensure the mean and the standard deviation to be 0 and 1, respectively. The equation is shown below:

X(stand)=x-mean(x)/standard deviation(x)

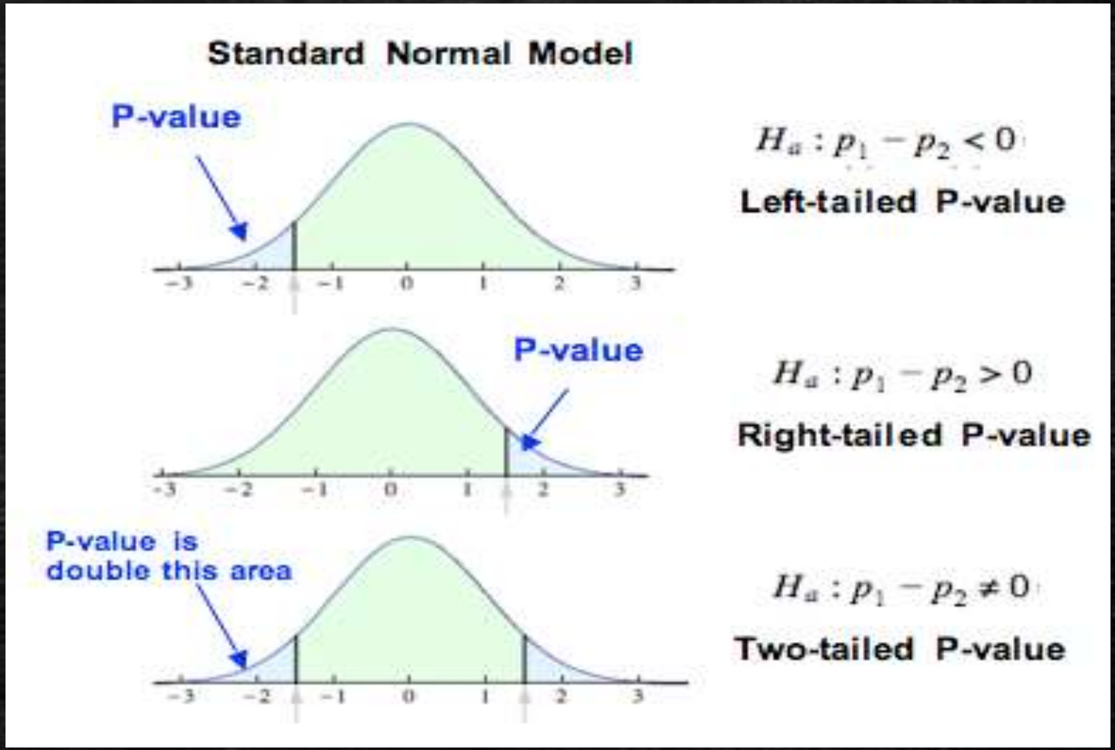
DATA STANDARDISATION

In [40]:
```
scaler = preprocessing.StandardScaler()  # create the scaler object
scaleddata = scaler.fit_transform(df.select_dtypes(include=['float']))  # fit data on scaler object
scaleddata = pd.DataFrame(scaleddata)  # standardizes the data
print(scaleddata)
```

```
              0         1         2
0      0.364165  0.553799  0.719582
1      0.760218 -0.150622  0.719582
2      0.760218 -0.150622  0.719582
3      1.552323 -0.150622  0.719582
4      0.166139 -1.295306 -0.190890
...         ...       ...       ...
14415  0.562191 -0.238675 -0.617698
14416 -0.823992  0.289641  0.273822
14417  0.958244  1.698482  1.229022
14418 -1.418071 -0.590885 -0.617698
14419 -0.625966 -0.590885 -0.617698

[14420 rows x 3 columns]
```

HYPOTHESIS TESTING

Hypothesis testing is a statistical method that is used in making statistical decisions using experimental data. Hypothesis Testing is basically an assumption that we make about the population parameter.



HYPOTHESIS TESTING

This is our hypothesis testing

Usually, statistical significance is associated with an alpha level of α = 0.05.

```
In [37]: def z_test(data, tail, null_hypothesis_mean):
             from scipy.stats import norm
             x_bar= data.mean()
             sigma = data.std(ddof=0)
             mu=null_hypothesis_mean
             N = 40
             SE = sigma/np.sqrt(N)
             z_val = (x_bar - mu)/SE
             p_val=norm.cdf(z_val)
             print(z_val,p_val)
             if tail ==0:
                 if p_val >= 0.05:
                     return True
                 else:
                     return False
             elif tail == 1:
                 if (1-p_val) >= 0.05:
                     return True
                 else:
                     return False
             elif tail == 2:
                 if (2*p_val) >= 0.05:
                     return True
                 else:
                     return False
```

HYPOTHESIS TESTING

```
In [41]: x_bar= data.mean()
         x_bar

Out[41]: ID         69407.051806
         Age           25.889752
         Height       177.069144
         Weight        73.051330
         Year        1973.943845
         dtype: float64

In [42]: sigma = data.std(ddof=0)
         sigma

Out[42]: ID         38849.492460
         Age            5.865000
         Height         9.670802
         Weight        13.202338
         Year          33.822432
         dtype: float64
```

Here we are finding the x_bar and sigma values of the columns like age, weight and height to calculate the z_value and compute the p_value.

HYPOTHESIS TESTING

```
In [38]: z_test(data["Weight"],1,68)

         2.419830115090449 0.9922361202386878

Out[38]: False
```

Ho: The mean weight of athletes is greater than or equal to average weight of human being(68kg) $\mu >= 68$;
H1: The mean weight of athletes is lesser than average weight of human being(68kg) $\mu < 68$
We fail to reject Ho

Ho: The mean height of athletes is greater than or equal to average height of human being(170cm) $\mu >= 170$;
H1: The mean height of athletes is lesser than average height of human being(170cm) $\mu < 170$
We fail to reject Ho

```
In [39]: z_test(data["Height"],1,170)

         4.62311131507761 0.9999981098663565

Out[39]: False
```

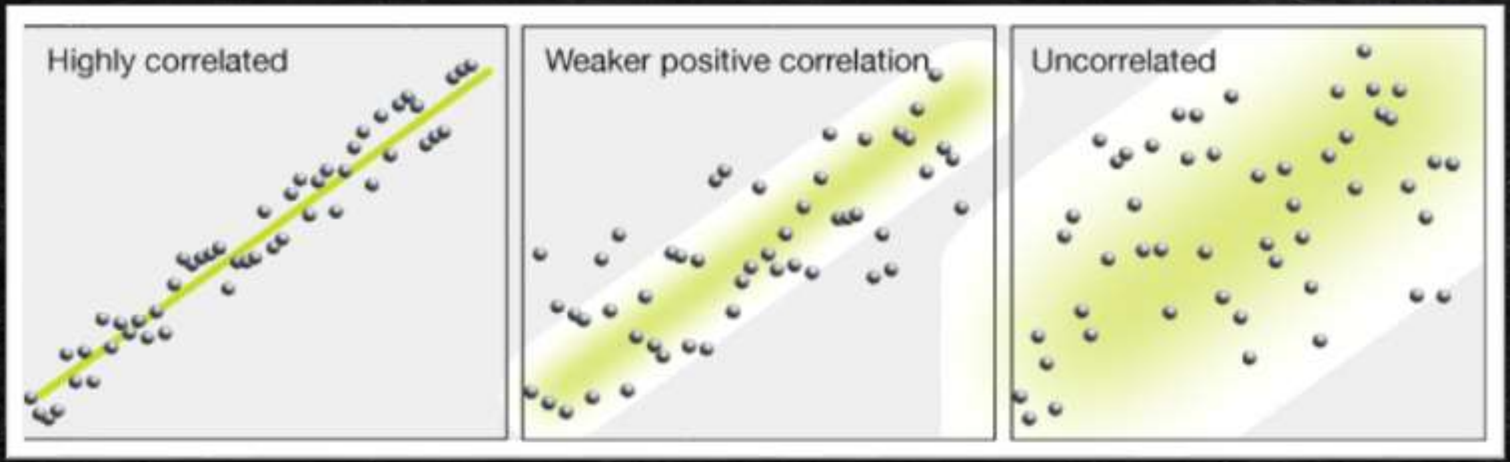
```
In [40]: z_test(data["Age"],1,27)

         -1.1674458795276428 0.12151517613965507

Out[40]: True
```

Ho: The mean age of athletes is greater than or equal to 27years: $\mu >= 27$
H1: The mean age of athletes is lesser than 27years: $\mu < 27$
We reject Ho

CORRELATION

Correlation is a statistical measure which determines co-relationship or association of two variables. It represents linear relationship between two variables. Correlation can have a value ranging from:

1) 1 is perfect positive correlation
2) -0 implies that there is no correlation
3) -1 is perfect negative correlation



CORRELATION

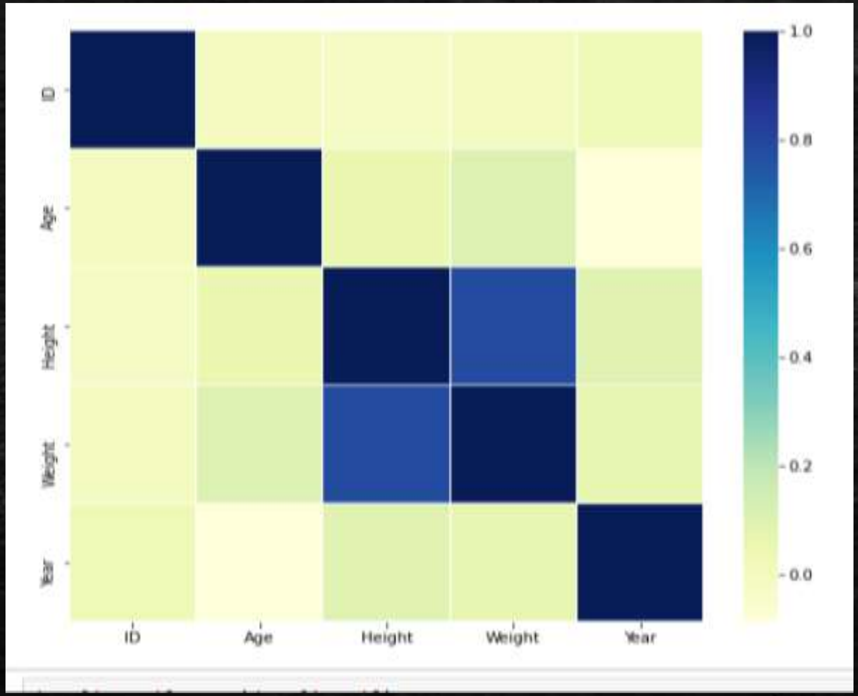
Here we are defining relationship between the variables.

```
In [41]: from scipy.stats import pearsonr

         # Convert dataframe into series
         list1 = df['Height']
         list2 = df['Weight']

         # Apply the pearsonr()
         corr, _ = pearsonr(list1, list2)
         print('Pearsons correlation: %.3f' % corr)

         Pearsons correlation: 0.805

In [42]:
         # Convert dastaframe into series
         list1 = df['Age']
         list2 = df['Weight']

         # Apply the pearsonr()
         corr, _ = pearsonr(list1, list2)
         print('Pearsons correlation: %.3f' % corr)

         Pearsons correlation: 0.175

In [43]: data.corr()
Out[43]:
```

	ID	Age	Height	Weight	Year
ID	1.000000	0.000099	-0.011929	-0.005445	0.032561
Age	0.000099	1.000000	0.056997	0.108108	-0.086881
Height	-0.011929	0.056997	1.000000	0.791071	0.092871
Weight	-0.005445	0.108108	0.791071	1.000000	0.074167
Year	0.032561	-0.086881	0.092871	0.074167	1.000000

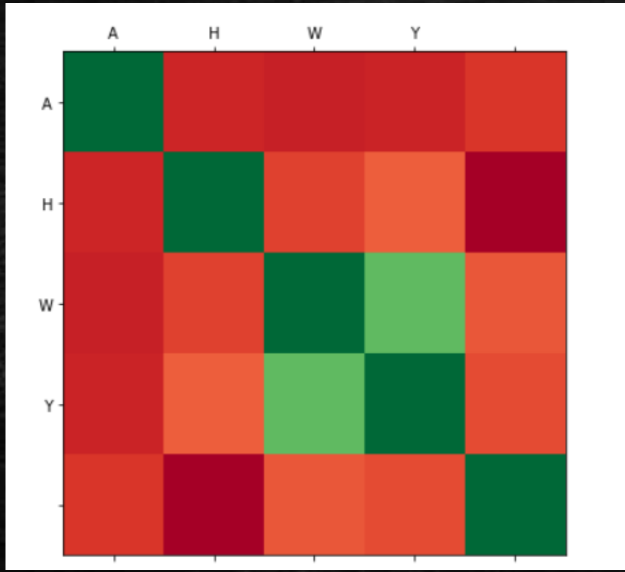
CORRELATION

```
In [44]: cormat = data.corr()

         f, ax = plt.subplots(figsize =(9, 8))
         sns.heatmap(cormat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)
```



CORRELATION

In [57]:
```
cormat = data.corr()
label=[i[:1] for i in cormat.columns]

fig=plt.figure(figsize=(6,12))
ax=fig.add_subplot(111)
ax.set_yticklabels(label)
ax.set_xticklabels(label)
ax.matshow(cormat,cmap=plt.cm.RdYlGn)
```



CONCLUSION

It's great pleasure for us to undertake this project as while doing this project we learned a lot about the athlete and the events happening in it.

We chose dataset on "athlete events" i.e. "120 years of Olympic History". We decided on this as the Olympics as this year has been postponed to next year due to the COVID-19 pandemic. This dataset was abundant in information about all the athletes who took part in the Olympics all through those 120 years. We could have a closer look at all the minute details about these respective Olympic winners including details like height, weight, sport they participated in, subcategory in the sport they won a medal in, the country they represented, etc. Although the data was large, it served well for us to apply our knowledge on data science and try and improve the quality of information contained in it. We learnt how to extract, deal with missing values .Used visualization techniques to the understood dataset which just plain numbers can't and also learnt about correlation which deals with the linear relationship.
India has a long way to go, to be successful in Olympics like USA but we Believe that the next Olympics in Tokyo, India will do better.

STUDENT PROFILE

STUDENT NAME	SRN	SECTION
BERU NEHA	PES2UG19CS084	B
D. LASYA PRIYA	PES2UG19CS111	B
DEEPALI SURAJ ATTAVAR	PES2UG19CS106	B
BHAVANA R	PES2UG19CS 089	B

THANK YOU.