olympics-analysis

July 6, 2024

1 Olympic Analysis: 120 Years of Sports Statistics

1.1 Introduction

The project focused on Olympics game Analysis, incorporating techniques in technical analysis, data visualization using data sourced from Kaggle. The analysis primarily concentrated on 120 years of Olympic history and results based on that. The project involved leveraging the Pandas library to retrieve and process Olympic information, enabling the visualization of various key indicators and trends. For plotting, Seaborn, Matplotlib, and Plotly libraries were used.

1.2 Questions

In this analysis, I want to explore the following questions:

1. How has the number of participating countries in the Olympics changed over time?
2. What trends can be observed in the number of events in the Olympics from 1896 to the present?
3. How has the number of athletes participating in the Olympics evolved since the first modern games?
4. How has the number of events in Athletics changed over the years?
5. Who is the most successful athlete, having won the maximum number of medals? (Top 15)
6. What are the country-wise tally, heatmap, and list of successful athletes?
7. What is the relationship between age and medal count?
8. What is the relationship between age and participation in sports?
9. What is the relationship between height, weight, and male-female athletes' medal wins across gold, silver, and bronze categories?
10. What was the distribution of male and female participants over the years?

1.3 Data Sources

- Data sourced from Kaggle: https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-athletes-and-results

1.4 Analysis

- For detailed analysis and code, please refer to the Jupyter Notebook file.

1.5 Libraries Used

- Pandas

- Seaborn
- Matplotlib
- Plotly

[405]:
```
pip install numpy panda matplotlib plotly seaborn
```

```
Requirement already satisfied: numpy in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (2.0.0)
Requirement already satisfied: panda in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (0.3.1)
Requirement already satisfied: matplotlib in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (3.9.1)
Requirement already satisfied: plotly in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages
(5.22.0)
Requirement already satisfied: seaborn in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages
(0.13.2)
Requirement already satisfied: setuptools in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
panda) (70.2.0)
Requirement already satisfied: requests in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
panda) (2.32.3)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (2.9.0.post0)
```

```
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
plotly) (8.4.2)
Requirement already satisfied: pandas>=1.2 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
requests->panda) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
requests->panda) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
requests->panda) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\madhu\appdata\local\programs\python\python312\lib\site-packages (from
requests->panda) (2024.6.2)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[406]:
```
import pandas as pd
import numpy as np
```

[407]:
```
df = pd.read_csv('athlete_events.csv')
region_df = pd.read_csv('noc_regions.csv')
```

[408]:
```
df.tail()
```

[408]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	\
271111	135569	Andrzej ya	M	29.0	179.0	89.0	Poland-1	POL	
271112	135570	Piotr ya	M	27.0	176.0	59.0	Poland	POL	
271113	135570	Piotr ya	M	27.0	176.0	59.0	Poland	POL	
271114	135571	Tomasz Ireneusz ya	M	30.0	185.0	96.0	Poland	POL	
271115	135571	Tomasz Ireneusz ya	M	34.0	185.0	96.0	Poland	POL	

```
                Games  Year  Season            City        Sport  \
271111  1976 Winter  1976  Winter       Innsbruck         Luge
271112  2014 Winter  2014  Winter           Sochi  Ski Jumping
271113  2014 Winter  2014  Winter           Sochi  Ski Jumping
271114  1998 Winter  1998  Winter          Nagano    Bobsleigh
271115  2002 Winter  2002  Winter  Salt Lake City    Bobsleigh

                                          Event Medal
271111                Luge Mixed (Men)'s Doubles   NaN
271112  Ski Jumping Men's Large Hill, Individual   NaN
271113       Ski Jumping Men's Large Hill, Team   NaN
271114                     Bobsleigh Men's Four   NaN
271115                     Bobsleigh Men's Four   NaN
```

[409]:
```
#How many datas are there
df.shape
```

[409]:
```
(271116, 15)
```

[410]:
```
#Filtering only the data from Summer
df = df[df['Season']=='Summer']
```

[411]:
```
df.shape
```

[411]:
```
(222552, 15)
```

[412]:
```
df.tail()
```

[412]:
```
            ID                          Name Sex   Age  Height  Weight  \
271106  135565      Fernando scar Zylberberg   M  27.0   168.0    76.0
271107  135566    James Francis "Jim" Zylker   M  21.0   175.0    75.0
271108  135567  Aleksandr Viktorovich Zyuzin   M  24.0   183.0    72.0
271109  135567  Aleksandr Viktorovich Zyuzin   M  28.0   183.0    72.0
271110  135568        Olga Igorevna Zyuzkova   F  33.0   171.0    69.0

                 Team  NOC        Games  Year  Season            City  \
271106      Argentina  ARG  2004 Summer  2004  Summer          Athina
271107  United States  USA  1972 Summer  1972  Summer          Munich
271108         Russia  RUS  2000 Summer  2000  Summer          Sydney
271109         Russia  RUS  2004 Summer  2004  Summer          Athina
271110        Belarus  BLR  2016 Summer  2016  Summer  Rio de Janeiro

             Sport                                      Event Medal
271106      Hockey                   Hockey Men's Hockey   NaN
271107    Football                 Football Men's Football   NaN
271108      Rowing  Rowing Men's Lightweight Coxless Fours   NaN
271109      Rowing  Rowing Men's Lightweight Coxless Fours   NaN
```

```
271110  Basketball           Basketball Women's Basketball   NaN
```

[413]:
```
#Finding the country name from NOC
df = df.merge(region_df,on='NOC',how='left')
```

[414]:
```
df.tail()
```

[414]:
```
            ID                          Name Sex   Age  Height  Weight  \
222547  135565      Fernando scar Zylberberg   M  27.0   168.0    76.0
222548  135566    James Francis "Jim" Zylker   M  21.0   175.0    75.0
222549  135567  Aleksandr Viktorovich Zyuzin   M  24.0   183.0    72.0
222550  135567  Aleksandr Viktorovich Zyuzin   M  28.0   183.0    72.0
222551  135568        Olga Igorevna Zyuzkova   F  33.0   171.0    69.0

                Team  NOC        Games  Year  Season            City  \
222547     Argentina  ARG  2004 Summer  2004  Summer          Athina
222548  United States  USA  1972 Summer  1972  Summer          Munich
222549        Russia  RUS  2000 Summer  2000  Summer          Sydney
222550        Russia  RUS  2004 Summer  2004  Summer          Athina
222551       Belarus  BLR  2016 Summer  2016  Summer  Rio de Janeiro

            Sport                                     Event Medal     region  \
222547      Hockey                       Hockey Men's Hockey   NaN  Argentina
222548    Football                    Football Men's Football   NaN        USA
222549      Rowing  Rowing Men's Lightweight Coxless Fours   NaN     Russia
222550      Rowing  Rowing Men's Lightweight Coxless Fours   NaN     Russia
222551  Basketball           Basketball Women's Basketball   NaN    Belarus

       notes
222547   NaN
222548   NaN
222549   NaN
222550   NaN
222551   NaN
```

[415]:
```
#Name of all the regions
df['region'].unique()
```

[415]:
```
array(['China', 'Denmark', 'Netherlands', 'Finland', 'Norway', 'Romania',
       'Estonia', 'France', 'Morocco', 'Spain', 'Egypt', 'Iran',
       'Bulgaria', 'Italy', 'Chad', 'Azerbaijan', 'Sudan', 'Russia',
       'Argentina', 'Cuba', 'Belarus', 'Greece', 'Cameroon', 'Turkey',
       'Chile', 'Mexico', 'USA', 'Nicaragua', 'Hungary', 'Nigeria',
       'Algeria', 'Kuwait', 'Bahrain', 'Pakistan', 'Iraq', 'Syria',
       'Lebanon', 'Qatar', 'Malaysia', 'Germany', 'Canada', 'Ireland',
       'Australia', 'South Africa', 'Eritrea', 'Tanzania', 'Jordan',
       'Tunisia', 'Libya', 'Belgium', 'Djibouti', 'Palestine', 'Comoros',
```

```
'Kazakhstan', 'Brunei', 'India', 'Saudi Arabia', 'Maldives',
'Ethiopia', 'United Arab Emirates', 'Yemen', 'Indonesia',
'Philippines', nan, 'Uzbekistan', 'Kyrgyzstan', 'Tajikistan',
'Japan', 'Republic of Congo', 'Switzerland', 'Brazil', 'Monaco',
'Israel', 'Uruguay', 'Sweden', 'Sri Lanka', 'Armenia',
'Ivory Coast', 'Kenya', 'Benin', 'UK', 'Ghana', 'Somalia', 'Niger',
'Mali', 'Afghanistan', 'Poland', 'Costa Rica', 'Panama', 'Georgia',
'Slovenia', 'Guyana', 'New Zealand', 'Portugal', 'Paraguay',
'Angola', 'Venezuela', 'Colombia', 'Bangladesh', 'Peru',
'El Salvador', 'Puerto Rico', 'Uganda', 'Honduras', 'Ecuador',
'Turkmenistan', 'Mauritius', 'Seychelles', 'Czech Republic',
'Luxembourg', 'Mauritania', 'Saint Kitts', 'Trinidad',
'Dominican Republic', 'Saint Vincent', 'Jamaica', 'Liberia',
'Suriname', 'Nepal', 'Mongolia', 'Austria', 'Palau', 'Lithuania',
'Togo', 'Namibia', 'Curacao', 'Ukraine', 'Iceland',
'American Samoa', 'Samoa', 'Rwanda', 'Croatia', 'Dominica',
'Haiti', 'Malta', 'Cyprus', 'Guinea', 'Belize', 'Thailand',
'Bermuda', 'Serbia', 'Sierra Leone', 'Papua New Guinea',
'Individual Olympic Athletes', 'Oman', 'Fiji', 'Vanuatu',
'Moldova', 'Bahamas', 'Guatemala', 'Latvia',
'Virgin Islands, British', 'Mozambique', 'Virgin Islands, US',
'Central African Republic', 'Madagascar', 'Bosnia and Herzegovina',
'Guam', 'Cayman Islands', 'Slovakia', 'Barbados', 'Guinea-Bissau',
'Timor-Leste', 'Democratic Republic of the Congo', 'Gabon',
'San Marino', 'Laos', 'Botswana', 'South Korea', 'Cambodia',
'North Korea', 'Solomon Islands', 'Senegal', 'Cape Verde',
'Equatorial Guinea', 'Boliva', 'Antigua', 'Andorra', 'Zimbabwe',
'Grenada', 'Saint Lucia', 'Micronesia', 'Myanmar', 'Malawi',
'Zambia', 'Taiwan', 'Sao Tome and Principe', 'Macedonia',
'Liechtenstein', 'Montenegro', 'Gambia', 'Cook Islands', 'Albania',
'Swaziland', 'Burkina Faso', 'Burundi', 'Aruba', 'Nauru',
'Vietnam', 'Bhutan', 'Marshall Islands', 'Kiribati', 'Tonga',
'Kosovo', 'South Sudan', 'Lesotho'], dtype=object)
```

[416]:
```
#How many missing values are there
#Is there any null value
df.isnull().sum()
```

[416]:
```
ID            0
Name          0
Sex           0
Age        9189
Height    51857
Weight    53854
Team          0
NOC           0
Games         0
```

```
Year           0
Season         0
City           0
Sport          0
Event          0
Medal     188464
region       370
notes     218151
dtype: int64
```

[417]:
```
#is there any duplicate value
df.duplicated().sum() #Afterr removing duplicate
```

[417]:
```
np.int64(1385)
```

[418]:
```
#Remove duplicates
df.drop_duplicates(inplace=True)
```

[419]:
```
#Counting the number of medals
df['Medal'].value_counts()
```

[419]:
```
Medal
Gold      11456
Bronze    11409
Silver    11212
Name: count, dtype: int64
```

[420]:
```
#Making different columns for different medals
pd.get_dummies(df['Medal']).astype(int)
```

[420]:

	Bronze	Gold	Silver
0	0	0	0
1	0	0	0
2	0	0	0
3	0	1	0
4	0	0	0
…	…	…	…
222547	0	0	0
222548	0	0	0
222549	0	0	0
222550	0	0	0
222551	0	0	0

[221167 rows x 3 columns]

[421]:
```
df.shape
```

[421]: (221167, 17)

[422]:
```
#concatinating with original data frame
df = pd.concat([df,pd.get_dummies(df['Medal']).astype(int)],axis=1)
```

[423]:
```
df.shape
```

[423]: (221167, 20)

[424]:
```
df.tail()
```

[424]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal	region	notes	Bronze	Gold	Silver
222547	135565	Fernando scar Zylberberg	M	27.0	168.0	76.0	Argentina	ARG	2004 Summer	2004	Summer	Athina	Hockey	Hockey Men's Hockey	NaN	Argentina	NaN	0	0	0
222548	135566	James Francis "Jim" Zylker	M	21.0	175.0	75.0	United States	USA	1972 Summer	1972	Summer	Munich	Football	Football Men's Football	NaN	USA	NaN	0	0	0
222549	135567	Aleksandr Viktorovich Zyuzin	M	24.0	183.0	72.0	Russia	RUS	2000 Summer	2000	Summer	Sydney	Rowing	Rowing Men's Lightweight Coxless Fours	NaN	Russia	NaN	0	0	0
222550	135567	Aleksandr Viktorovich Zyuzin	M	28.0	183.0	72.0	Russia	RUS	2004 Summer	2004	Summer	Athina	Rowing	Rowing Men's Lightweight Coxless Fours	NaN	Russia	NaN	0	0	0
222551	135568	Olga Igorevna Zyuzkova	F	33.0	171.0	69.0	Belarus	BLR	2016 Summer	2016	Summer	Rio de Janeiro	Basketball	Basketball Women's Basketball	NaN	Belarus	NaN	0	0	0

[425]:
```
print(df.groupby('NOC').sum().columns)
```

```
Index(['ID', 'Name', 'Sex', 'Age', 'Height', 'Weight', 'Team', 'Games', 'Year',
       'Season', 'City', 'Sport', 'Event', 'Medal', 'region', 'notes',
       'Bronze', 'Gold', 'Silver'],
      dtype='object')
```

[426]:
```
#Groupin by on NOC and summing all the columns
df.groupby('NOC').sum()[['Gold','Silver','Bronze']].
 ↪sort_values('Gold',ascending=False).reset_index().head(25)
```

[426]:

	NOC	Gold	Silver	Bronze
0	USA	2472	1333	1197
1	URS	832	635	596
2	GBR	635	729	620
3	GER	592	538	649
4	ITA	518	474	454
5	FRA	463	567	587
6	HUN	432	328	363
7	SWE	354	396	358
8	AUS	342	452	510
9	GDR	339	277	227
10	CHN	334	317	258
11	RUS	296	278	331
12	NED	245	302	371
13	JPN	230	287	333
14	NOR	227	196	167
15	DEN	179	236	177
16	KOR	171	206	175
17	CUB	164	129	116
18	ROU	161	200	290
19	CAN	158	239	344
20	FRG	144	172	188
21	FIN	132	125	217
22	IND	131	19	40
23	YUG	130	161	92
24	POL	111	185	242

[427]:
```
#Removing duplicate rows on the basis of Games, NOC, Cities, Sports, Events,␣
 ↪Medal
medal_tally=df.
 ↪drop_duplicates(subset=['Team','NOC','Games','Year','City','Sport','Event','Medal'])
```

[428]:
```
medal_tally = medal_tally.groupby('NOC').sum()[['Gold','Silver','Bronze']].
 ↪sort_values('Gold',ascending=False).reset_index()
```

[429]:
```
medal_tally['Total'] = medal_tally['Gold'] + medal_tally['Silver'] +␣
 ↪medal_tally['Bronze']
```

[430]:
```
medal_tally
```

[430]:

	NOC	Gold	Silver	Bronze	Total
0	USA	1035	802	708	2545
1	URS	394	317	294	1005

```
2    GBR   278     317     300    895
3    GER   235     261     283    779
4    FRA   234     256     287    777
..   …   …     …     …    …
225  WIF     0       0       2      2
226  YEM     0       0       0      0
227  YAR     0       0       0      0
228  YMD     0       0       0      0
229  ZAM     0       1       1      2

[230 rows x 5 columns]
```

[431]:
```
years = df['Year'].unique().tolist()
```

[432]:
```
years.sort()
```

[433]:
```
years.insert(0,'Overall')
```

[434]:
```
years
```

[434]:
```
['Overall',
 1896,
 1900,
 1904,
 1906,
 1908,
 1912,
 1920,
 1924,
 1928,
 1932,
 1936,
 1948,
 1952,
 1956,
 1960,
 1964,
 1968,
 1972,
 1976,
 1980,
 1984,
 1988,
 1992,
 1996,
 2000,
 2004,
```

```
        2008,
        2012,
        2016]
```

[435]:
```
country = np.unique(df['region'].dropna().values).tolist()
```

[436]:
```
country.sort()
```

[437]:
```
country
```

[437]:
```
['Afghanistan',
 'Albania',
 'Algeria',
 'American Samoa',
 'Andorra',
 'Angola',
 'Antigua',
 'Argentina',
 'Armenia',
 'Aruba',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas',
 'Bahrain',
 'Bangladesh',
 'Barbados',
 'Belarus',
 'Belgium',
 'Belize',
 'Benin',
 'Bermuda',
 'Bhutan',
 'Boliva',
 'Bosnia and Herzegovina',
 'Botswana',
 'Brazil',
 'Brunei',
 'Bulgaria',
 'Burkina Faso',
 'Burundi',
 'Cambodia',
 'Cameroon',
 'Canada',
 'Cape Verde',
 'Cayman Islands',
 'Central African Republic',
```

```
'Chad',
'Chile',
'China',
'Colombia',
'Comoros',
'Cook Islands',
'Costa Rica',
'Croatia',
'Cuba',
'Curacao',
'Cyprus',
'Czech Republic',
'Democratic Republic of the Congo',
'Denmark',
'Djibouti',
'Dominica',
'Dominican Republic',
'Ecuador',
'Egypt',
'El Salvador',
'Equatorial Guinea',
'Eritrea',
'Estonia',
'Ethiopia',
'Fiji',
'Finland',
'France',
'Gabon',
'Gambia',
'Georgia',
'Germany',
'Ghana',
'Greece',
'Grenada',
'Guam',
'Guatemala',
'Guinea',
'Guinea-Bissau',
'Guyana',
'Haiti',
'Honduras',
'Hungary',
'Iceland',
'India',
'Individual Olympic Athletes',
'Indonesia',
'Iran',
```

```
'Iraq',
'Ireland',
'Israel',
'Italy',
'Ivory Coast',
'Jamaica',
'Japan',
'Jordan',
'Kazakhstan',
'Kenya',
'Kiribati',
'Kosovo',
'Kuwait',
'Kyrgyzstan',
'Laos',
'Latvia',
'Lebanon',
'Lesotho',
'Liberia',
'Libya',
'Liechtenstein',
'Lithuania',
'Luxembourg',
'Macedonia',
'Madagascar',
'Malawi',
'Malaysia',
'Maldives',
'Mali',
'Malta',
'Marshall Islands',
'Mauritania',
'Mauritius',
'Mexico',
'Micronesia',
'Moldova',
'Monaco',
'Mongolia',
'Montenegro',
'Morocco',
'Mozambique',
'Myanmar',
'Namibia',
'Nauru',
'Nepal',
'Netherlands',
'New Zealand',
```

```
'Nicaragua',
'Niger',
'Nigeria',
'North Korea',
'Norway',
'Oman',
'Pakistan',
'Palau',
'Palestine',
'Panama',
'Papua New Guinea',
'Paraguay',
'Peru',
'Philippines',
'Poland',
'Portugal',
'Puerto Rico',
'Qatar',
'Republic of Congo',
'Romania',
'Russia',
'Rwanda',
'Saint Kitts',
'Saint Lucia',
'Saint Vincent',
'Samoa',
'San Marino',
'Sao Tome and Principe',
'Saudi Arabia',
'Senegal',
'Serbia',
'Seychelles',
'Sierra Leone',
'Slovakia',
'Slovenia',
'Solomon Islands',
'Somalia',
'South Africa',
'South Korea',
'South Sudan',
'Spain',
'Sri Lanka',
'Sudan',
'Suriname',
'Swaziland',
'Sweden',
'Switzerland',
```

```
 'Syria',
 'Taiwan',
 'Tajikistan',
 'Tanzania',
 'Thailand',
 'Timor-Leste',
 'Togo',
 'Tonga',
 'Trinidad',
 'Tunisia',
 'Turkey',
 'Turkmenistan',
 'UK',
 'USA',
 'Uganda',
 'Ukraine',
 'United Arab Emirates',
 'Uruguay',
 'Uzbekistan',
 'Vanuatu',
 'Venezuela',
 'Vietnam',
 'Virgin Islands, British',
 'Virgin Islands, US',
 'Yemen',
 'Zambia',
 'Zimbabwe']
```

[438]:
```
country.insert(0,'Overall')
```

[439]:
```
country
```

[439]:
```
['Overall',
 'Afghanistan',
 'Albania',
 'Algeria',
 'American Samoa',
 'Andorra',
 'Angola',
 'Antigua',
 'Argentina',
 'Armenia',
 'Aruba',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas',
```

```
'Bahrain',
'Bangladesh',
'Barbados',
'Belarus',
'Belgium',
'Belize',
'Benin',
'Bermuda',
'Bhutan',
'Boliva',
'Bosnia and Herzegovina',
'Botswana',
'Brazil',
'Brunei',
'Bulgaria',
'Burkina Faso',
'Burundi',
'Cambodia',
'Cameroon',
'Canada',
'Cape Verde',
'Cayman Islands',
'Central African Republic',
'Chad',
'Chile',
'China',
'Colombia',
'Comoros',
'Cook Islands',
'Costa Rica',
'Croatia',
'Cuba',
'Curacao',
'Cyprus',
'Czech Republic',
'Democratic Republic of the Congo',
'Denmark',
'Djibouti',
'Dominica',
'Dominican Republic',
'Ecuador',
'Egypt',
'El Salvador',
'Equatorial Guinea',
'Eritrea',
'Estonia',
'Ethiopia',
```

```
'Fiji',
'Finland',
'France',
'Gabon',
'Gambia',
'Georgia',
'Germany',
'Ghana',
'Greece',
'Grenada',
'Guam',
'Guatemala',
'Guinea',
'Guinea-Bissau',
'Guyana',
'Haiti',
'Honduras',
'Hungary',
'Iceland',
'India',
'Individual Olympic Athletes',
'Indonesia',
'Iran',
'Iraq',
'Ireland',
'Israel',
'Italy',
'Ivory Coast',
'Jamaica',
'Japan',
'Jordan',
'Kazakhstan',
'Kenya',
'Kiribati',
'Kosovo',
'Kuwait',
'Kyrgyzstan',
'Laos',
'Latvia',
'Lebanon',
'Lesotho',
'Liberia',
'Libya',
'Liechtenstein',
'Lithuania',
'Luxembourg',
'Macedonia',
```

```
'Madagascar',
'Malawi',
'Malaysia',
'Maldives',
'Mali',
'Malta',
'Marshall Islands',
'Mauritania',
'Mauritius',
'Mexico',
'Micronesia',
'Moldova',
'Monaco',
'Mongolia',
'Montenegro',
'Morocco',
'Mozambique',
'Myanmar',
'Namibia',
'Nauru',
'Nepal',
'Netherlands',
'New Zealand',
'Nicaragua',
'Niger',
'Nigeria',
'North Korea',
'Norway',
'Oman',
'Pakistan',
'Palau',
'Palestine',
'Panama',
'Papua New Guinea',
'Paraguay',
'Peru',
'Philippines',
'Poland',
'Portugal',
'Puerto Rico',
'Qatar',
'Republic of Congo',
'Romania',
'Russia',
'Rwanda',
'Saint Kitts',
'Saint Lucia',
```

```
'Saint Vincent',
'Samoa',
'San Marino',
'Sao Tome and Principe',
'Saudi Arabia',
'Senegal',
'Serbia',
'Seychelles',
'Sierra Leone',
'Slovakia',
'Slovenia',
'Solomon Islands',
'Somalia',
'South Africa',
'South Korea',
'South Sudan',
'Spain',
'Sri Lanka',
'Sudan',
'Suriname',
'Swaziland',
'Sweden',
'Switzerland',
'Syria',
'Taiwan',
'Tajikistan',
'Tanzania',
'Thailand',
'Timor-Leste',
'Togo',
'Tonga',
'Trinidad',
'Tunisia',
'Turkey',
'Turkmenistan',
'UK',
'USA',
'Uganda',
'Ukraine',
'United Arab Emirates',
'Uruguay',
'Uzbekistan',
'Vanuatu',
'Venezuela',
'Vietnam',
'Virgin Islands, British',
'Virgin Islands, US',
```

```
 'Yemen',
 'Zambia',
 'Zimbabwe']
```

[440]:
```
def fetch_medal_tally(df,year, country):
    medal_df=df.
 ↪drop_duplicates(subset=['Team','NOC','Games','Year','City','Sport','Event','Medal'])
    flag = 0
    if year == 'Overall' and country == 'Overall':
        temp_df = medal_df
    if year == 'Overall' and country != 'Overall':
        flag = 1
        temp_df = medal_df[medal_df['region'] == country]
    if year != 'Overall' and country == 'Overall':
        temp_df = medal_df[medal_df['Year'] == int(year)]
    if year != 'Overall' and country != 'Overall':
        temp_df = medal_df[(medal_df['Year'] == int(year)) &␣
 ↪(medal_df['region'] == country)]

    if flag == 1 :
        x = temp_df.groupby('Year').sum()[['Gold', 'Silver', 'Bronze']].
 ↪sort_values('Year').reset_index()
    else:
        x = temp_df.groupby('region').sum()[['Gold', 'Silver', 'Bronze']].
 ↪sort_values('Gold', ascending=False).reset_index()  # Return an empty␣
 ↪DataFrame if no condition is met

    x['Total'] = x['Gold'] + x['Silver'] + x['Bronze']

    print(x)
```

[441]:
```
#fetch_medal_tally(medal_df, year='1900', country='India')
```

[442]:
```
medal_df = df.
 ↪drop_duplicates(subset=['Team','NOC','Games','Year','City','Sport','Event','Medal'])
```

[443]:
```
medal_df[(medal_df['Year'] == 2016) & (medal_df['region'] == 'India')]
```

[443]:

	ID	Name	Sex	Age	Height	Weight	Team	\
1015	663	Sharath Kamal Achanta	M	34.0	186.0	85.0	India	
7065	4523	Seema Antil	F	33.0	182.0	92.0	India	
8713	5562	Aditi Ashok	F	18.0	173.0	57.0	India	
9202	5868	Manu Attri	M	23.0	172.0	73.0	India	
10070	6427	Lalita Shivaji Babar	F	27.0	166.0	50.0	India	
…	…	…	..	…	…	…	…	
182260	111467	Sathish Kumar Sivalingam	M	24.0	175.0	77.0	India	
195568	119515	Shiva Thapa	M	22.0	169.0	56.0	India	

```
198042  120871             Sandeep Tomar   M  25.0   168.0    61.0  India
216942  132143       Vikas Krishan Yadav   M  24.0   177.0    69.0  India
216985  132177      Mohammad Anas Yahiya   M  21.0   177.0    69.0  India

        NOC        Games  Year  Season            City          Sport  \
1015    IND  2016 Summer  2016  Summer  Rio de Janeiro   Table Tennis
7065    IND  2016 Summer  2016  Summer  Rio de Janeiro      Athletics
8713    IND  2016 Summer  2016  Summer  Rio de Janeiro           Golf
9202    IND  2016 Summer  2016  Summer  Rio de Janeiro      Badminton
10070   IND  2016 Summer  2016  Summer  Rio de Janeiro      Athletics
…     …          …   …     …             …             …
182260  IND  2016 Summer  2016  Summer  Rio de Janeiro  Weightlifting
195568  IND  2016 Summer  2016  Summer  Rio de Janeiro         Boxing
198042  IND  2016 Summer  2016  Summer  Rio de Janeiro      Wrestling
216942  IND  2016 Summer  2016  Summer  Rio de Janeiro         Boxing
216985  IND  2016 Summer  2016  Summer  Rio de Janeiro      Athletics

                                              Event Medal region notes  \
1015                    Table Tennis Men's Singles   NaN  India   NaN
7065                  Athletics Women's Discus Throw   NaN  India   NaN
8713                         Golf Women's Individual   NaN  India   NaN
9202                         Badminton Men's Doubles   NaN  India   NaN
10070   Athletics Women's 3,000 metres Steeplechase   NaN  India   NaN
…                                                 …   …    …   …
182260            Weightlifting Men's Middleweight   NaN  India   NaN
195568                  Boxing Men's Bantamweight   NaN  India   NaN
198042    Wrestling Men's Featherweight, Freestyle   NaN  India   NaN
216942                   Boxing Men's Middleweight   NaN  India   NaN
216985                  Athletics Men's 400 metres   NaN  India   NaN

        Bronze  Gold  Silver
1015         0     0       0
7065         0     0       0
8713         0     0       0
9202         0     0       0
10070        0     0       0
…          …   …     …
182260       0     0       0
195568       0     0       0
198042       0     0       0
216942       0     0       0
216985       0     0       0

[68 rows x 20 columns]
```

[444]:
```
#No. of Edition
#No. of cities
```

```
#No. of events/sports
#No. of athletes
#Participating Nation
```

[445]:
```
df.head(1)
```

[445]:
```
   ID       Name Sex   Age  Height  Weight   Team  NOC        Games  Year  \
0   1  A Dijiang   M  24.0   180.0    80.0  China  CHN  1992 Summer  1992

   Season       City       Sport                        Event Medal region  \
0  Summer  Barcelona  Basketball  Basketball Men's Basketball   NaN  China

  notes  Bronze  Gold  Silver
0   NaN       0     0       0
```

[446]:
```
#No. of Edition
df['Year'].unique().shape[0]-1 #As the Olympic played in 1906 was not considered
```

[446]: 28

[447]:
```
#No. of cities
df['City'].unique().shape[0]
```

[447]: 23

[448]:
```
#No. of sports
df['Sport'].unique().shape[0]
```

[448]: 52

[449]:
```
#No. of events
df['Event'].unique().shape[0]
```

[449]: 651

[450]:
```
#No. of athletes
df['Name'].unique().shape[0]
```

[450]: 116122

[451]:
```
#Participating Nation
df['region'].unique().shape[0]
```

[451]: 206

[452]:
```
#Over the years how many countries has participated Olympic
df.head()
```

[452]:
```
   ID                                Name Sex   Age  Height  Weight  \
0   1                           A Dijiang   M  24.0   180.0    80.0
1   2                            A Lamusi   M  23.0   170.0    60.0
2   3                 Gunnar Nielsen Aaby   M  24.0     NaN     NaN
3   4                Edgar Lindenau Aabye   M  34.0     NaN     NaN
4   8  Cornelia "Cor" Aalten (-Strannood)   F  18.0   168.0     NaN

              Team  NOC        Games  Year  Season         City       Sport  \
0            China  CHN  1992 Summer  1992  Summer    Barcelona  Basketball
1            China  CHN  2012 Summer  2012  Summer       London        Judo
2          Denmark  DEN  1920 Summer  1920  Summer    Antwerpen    Football
3   Denmark/Sweden  DEN  1900 Summer  1900  Summer        Paris  Tug-Of-War
4      Netherlands  NED  1932 Summer  1932  Summer  Los Angeles   Athletics

                          Event Medal       region notes  Bronze  Gold  Silver
0   Basketball Men's Basketball   NaN        China   NaN       0     0       0
1  Judo Men's Extra-Lightweight   NaN        China   NaN       0     0       0
2       Football Men's Football   NaN      Denmark   NaN       0     0       0
3   Tug-Of-War Men's Tug-Of-War  Gold      Denmark   NaN       0     1       0
4  Athletics Women's 100 metres   NaN  Netherlands   NaN       0     0       0
```

[453]:
```
nations_over_time = df.drop_duplicates(['Year', 'region'])['Year'].
↪value_counts().reset_index().rename(columns={'index': 'Year'}).
↪sort_values('Year')
```

[454]:
```
nations_over_time.columns = ['Edition', 'No. of countries']
```

[455]:
```
nations_over_time
```

[455]:

	Edition	No. of countries
28	1896	12
22	1900	31
27	1904	14
26	1906	20
25	1908	22
23	1912	29
24	1920	29
21	1924	45
20	1928	46
19	1932	47
18	1936	49
17	1948	59
16	1952	67
15	1956	71
13	1960	83
11	1964	93
10	1968	111

```
9       1972                120
12      1976                 91
14      1980                 80
8       1984                139
7       1988                156
6       1992                168
5       1996                196
4       2000                199
3       2004                200
2       2008                202
1       2012                203
0       2016                204
```

[456]:
```
#Number of events over the years
Events_over_time = df.drop_duplicates(['Year', 'Event'])['Year'].value_counts().
 ↪reset_index().rename(columns={'index': 'Year'}).sort_values('Year')
```

[457]:
```
Events_over_time.columns = ['Edition', 'Events']
```

[458]:
```
Events_over_time
```

[458]:
```
    Edition  Events
28      1896      43
26      1900      90
25      1904      95
27      1906      74
23      1908     109
24      1912     107
14      1920     158
20      1924     131
22      1928     122
21      1932     131
18      1936     150
15      1948     153
19      1952     149
16      1956     151
17      1960     150
13      1964     163
12      1968     172
11      1972     193
10      1976     198
9       1980     203
8       1984     221
7       1988     237
6       1992     257
5       1996     271
4       2000     300
```

```
3       2004     301
1       2008     302
2       2012     302
0       2016     306
```

[459]:
```
#Number of athletes over the years
Athletes_over_time = df.drop_duplicates(['Year', 'Name'])['Year'].
 ↪value_counts().reset_index().rename(columns={'index': 'Year'}).
 ↪sort_values('Year')
```

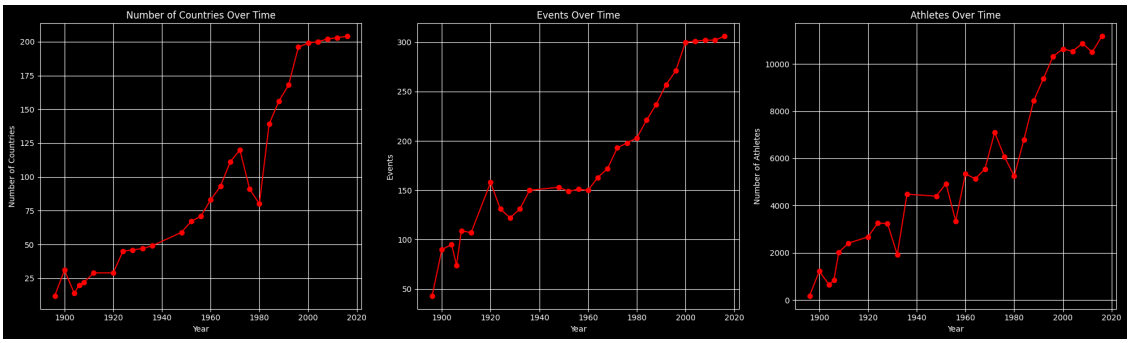
[460]:
```
Athletes_over_time.columns = ['Edition', 'Name']
```

[461]:
```
Athletes_over_time
```

[461]:
```
    Edition   Name
28     1896    176
25     1900   1220
27     1904    650
26     1906    841
23     1908   2024
22     1912   2409
21     1920   2675
19     1924   3256
20     1928   3246
24     1932   1922
16     1936   4482
17     1948   4402
15     1952   4931
18     1956   3346
12     1960   5348
14     1964   5134
11     1968   5552
8      1972   7105
10     1976   6070
13     1980   5252
9      1984   6791
7      1988   8443
6      1992   9380
5      1996  10324
2      2000  10639
3      2004  10537
1      2008  10880
4      2012  10502
0      2016  11174
```

1.6 Historical Trends in Olympic Participation

[462]:
```
import matplotlib.pyplot as plt

# Create subplots with 1 row and 3 columns
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20, 6))
plt.style.use('dark_background')

# Plot 1: Number of Countries Over Time
ax1.plot(nations_over_time['Edition'], nations_over_time['No. of countries'],␣
 ↪marker='o', linestyle='-', color='red')
ax1.set_title('Number of Countries Over Time')
ax1.set_xlabel('Year')
ax1.set_ylabel('Number of Countries')
ax1.grid(True)

# Plot 2: Events Over Time
ax2.plot(Events_over_time['Edition'], Events_over_time['Events'], marker='o',␣
 ↪linestyle='-', color='red')
ax2.set_title('Events Over Time')
ax2.set_xlabel('Year')
ax2.set_ylabel('Events')
ax2.grid(True)

# Plot 3: Athletes Over Time
ax3.plot(Athletes_over_time['Edition'], Athletes_over_time['Name'], marker='o',␣
 ↪linestyle='-', color='red')
ax3.set_title('Athletes Over Time')
ax3.set_xlabel('Year')
ax3.set_ylabel('Number of Athletes')
ax3.grid(True)

# Adjust layout to prevent overlap
plt.tight_layout()

# Display the plots
plt.show()
```



Here line graphs were utilized to visualize historical trends in Olympic participation. The increasing number of participating countries, events, and athletes over the years reflects the growing global enthusiasm and inclusivity of the Olympic Games. These graphs not only illustrate statistical trends but also highlight the expanding diversity and scale of the Olympics as a significant sporting and cultural event on a global scale.

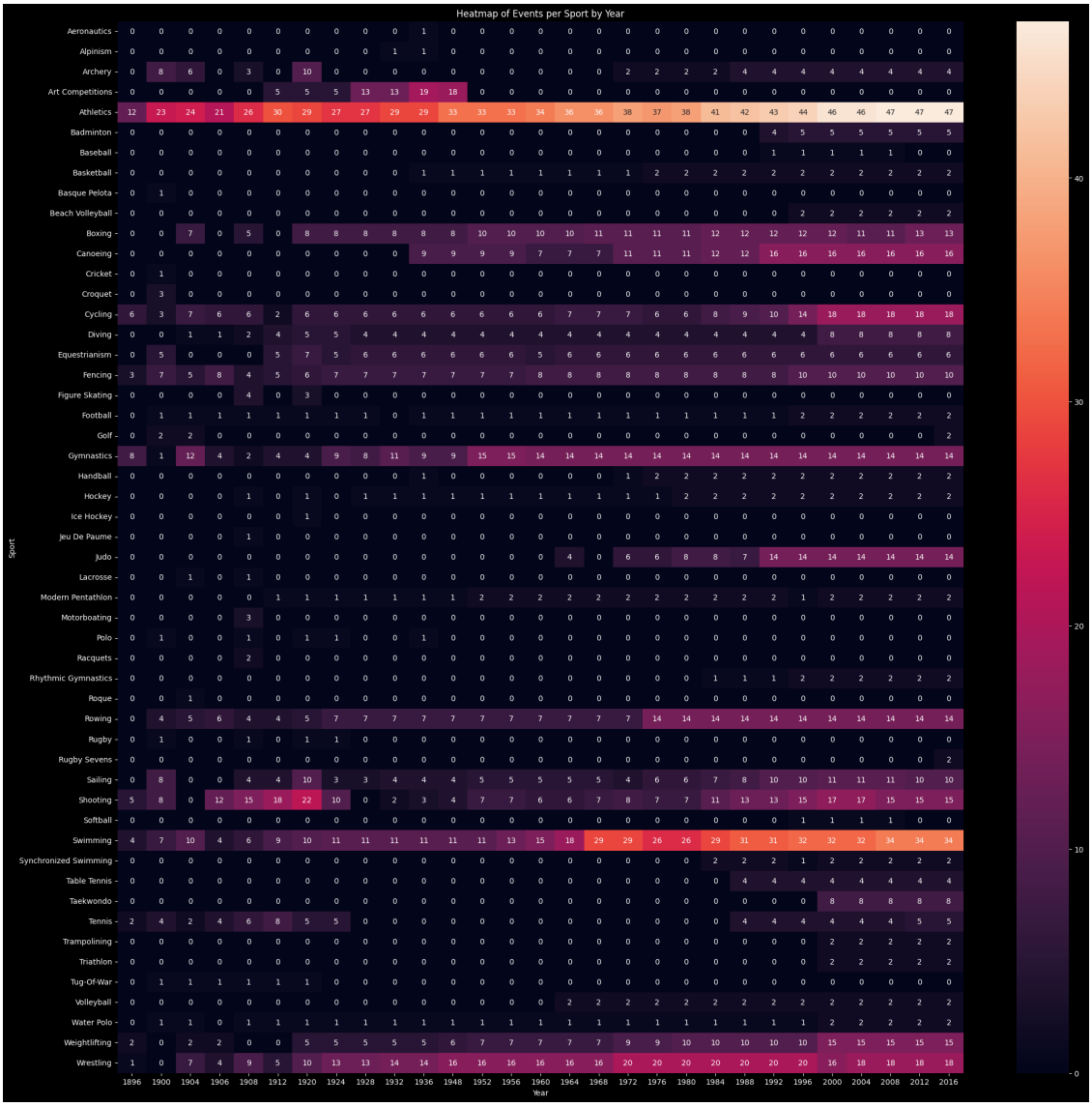
1.7 Evolution of Athletics Events in the Olympics

[463]:
```
#Over the years how many events hold in each sports i.e Number of Events overr␣
↪timme (Every Sport)
import seaborn as sns
```

[464]:
```
x = df.drop_duplicates(subset=['Year', 'Sport', 'Event'])
```

[465]:
```
plt.figure(figsize=(25,25))
sns.heatmap(x.pivot_table(index='Sport', columns='Year', values='Event',␣
↪aggfunc='count').fillna(0).astype(int), annot=True).set_title('Heatmap of␣
↪Events per Sport by Year')
```

[465]:
```
Text(0.5, 1.0, 'Heatmap of Events per Sport by Year')
```



For this question, a heatmap was used to visualize the evolution of Athletics events in the Olympics over the years. The heatmap provides a clear visual representation of how the number and types of Athletics events have changed since the inception of the modern Olympic Games in 1896. This graphical representation allows for easy identification of trends and shifts in focus within the Athletics category throughout Olympic history.

1.7.1 Top Athletes by Medal Count

[466]:
```
#Finding the most successful athlete who has won the maximum number of medal

def most_successful(df, sport):
    # Drop rows where 'Medal' is NaN
    temp_df = df.dropna(subset=['Medal'])
```

```
    # If a specific sport is provided, filter the DataFrame by that sport
    if sport != 'Overall':
        temp_df = temp_df[temp_df['Sport'] == sport]

    # Count the number of medals each athlete has won
    athlete_medal_counts = temp_df['Name'].value_counts().reset_index().head(15)
    athlete_medal_counts.columns = ['Name', 'Medal Count']

    # Merge with the original DataFrame to get additional athlete information
    result_df = athlete_medal_counts.merge(df, left_on='Name', right_on='Name',␣
 ↪how='left').drop_duplicates('Name')

    return result_df[['Name', 'Medal Count', 'Sport', 'region']]

# Example usage
most_successful(df, 'Overall')
```

[466]:

	Name	Medal Count	Sport	region
0	Michael Fred Phelps, II	28	Swimming	USA
30	Larysa Semenivna Latynina (Diriy-)	18	Gymnastics	Russia
49	Nikolay Yefimovich Andrianov	15	Gymnastics	Russia
73	Borys Anfiyanovych Shakhlin	13	Gymnastics	Russia
97	Takashi Ono	13	Gymnastics	Japan
130	Edoardo Mangiarotti	13	Fencing	Italy
144	Dara Grace Torres (-Hoffman, -Minas)	12	Swimming	USA
157	Birgit Fischer-Schmidt	12	Canoeing	Germany
170	Jennifer Elisabeth "Jenny" Thompson (-Cumpelik)	12	Swimming	USA
187	Ryan Steven Lochte	12	Swimming	USA
201	Paavo Johannes Nurmi	12	Athletics	Finland
213	Aleksey Yuryevich Nemov	12	Gymnastics	Russia
234	Sawao Kato	12	Gymnastics	Japan
258	Natalie Anne Coughlin (-Hall)	12	Swimming	USA
270	Vra slavsk (-Odloilov)	11	Gymnastics	Czech Republic

```
213          Russia
234           Japan
258             USA
270  Czech Republic
```

To identify the most successful athletes in Olympic history, a chart was created to visualize the top 15 athletes based on the number of medals won across all Olympic Games. This representation highlights the outstanding achievements of these athletes, showcasing their dominance and longevity in their respective sports over the years from all the countries.

2 Country Wise

-

2.1 Country wise medal tally per year (Line Plot)

-

2.2 Which countries are good at heatmap

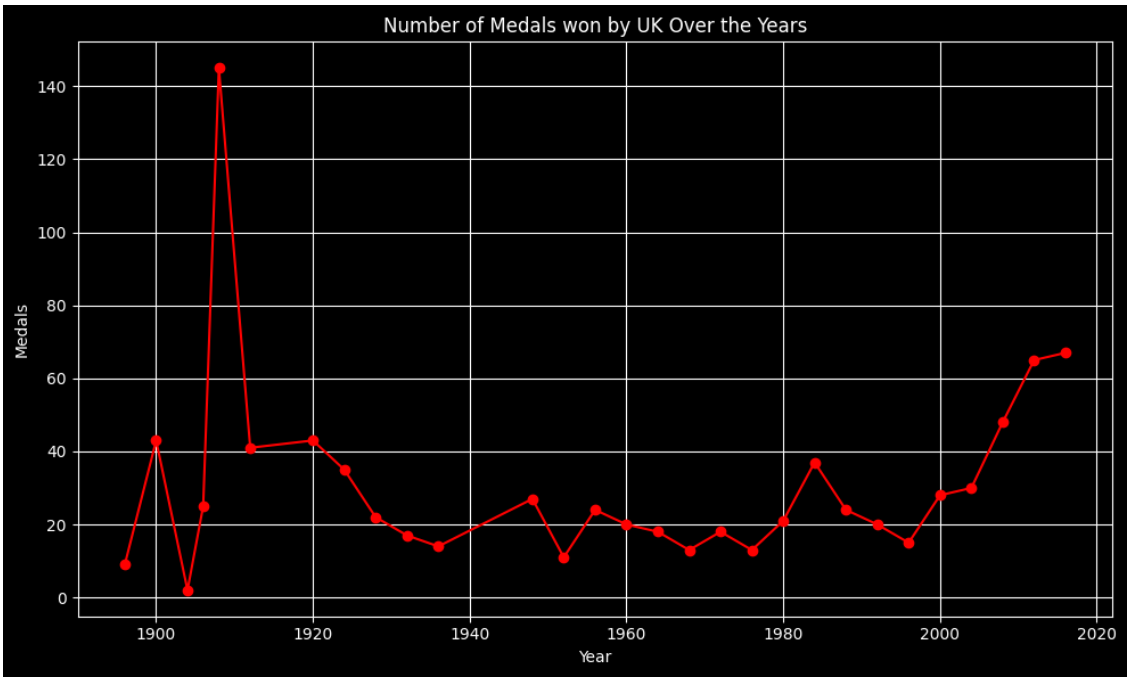
-

2.3 Most successful athlete (Top 10) Most successful athlete (Top 10)

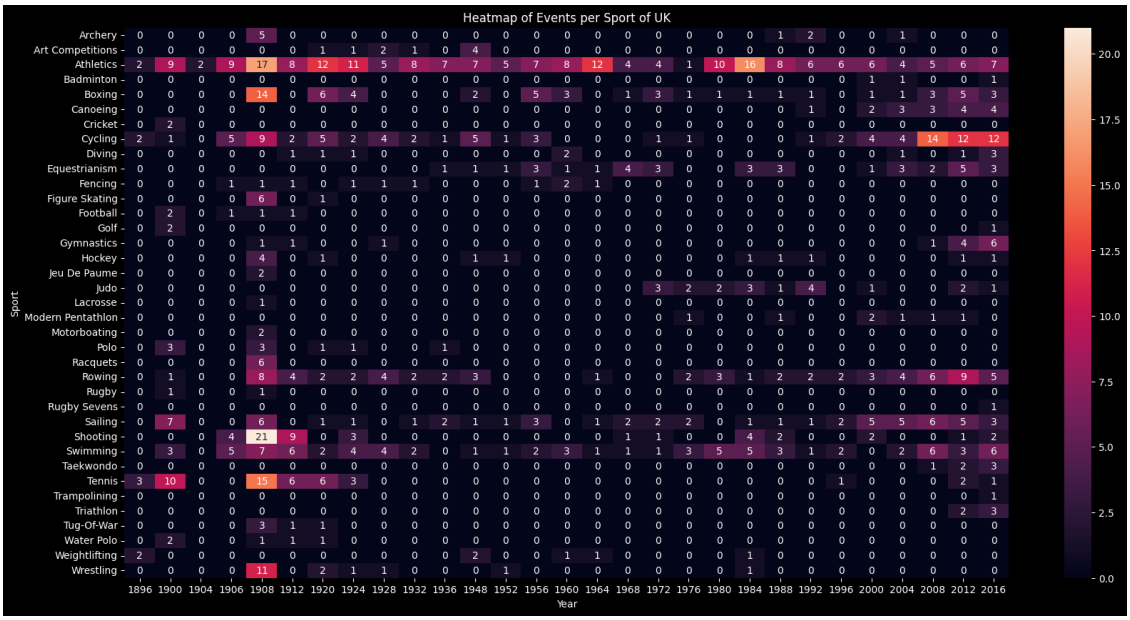
[467]:
```
temp_df = df.dropna(subset=['Medal'])
temp_df.
 ↪drop_duplicates(subset=['Team','NOC','Games','Year','City','Sport','Event','Medal'],inplace
```

[468]:
```
new_df = temp_df[temp_df['region']== 'UK'] #Taking the country UK
final_df = new_df.groupby('Year').count()['Medal'].reset_index()
```

[469]:
```
plt.style.use('dark_background')

# Plot: Number of Medals won by UK over the Years
plt.figure(figsize=(10, 6))
plt.plot(final_df['Year'], final_df['Medal'], marker='o', linestyle='-',␣
 ↪color='red')
plt.title('Number of Medals won by UK Over the Years')
plt.xlabel('Year')
plt.ylabel('Medals')
plt.grid(True)
plt.tight_layout()
plt.show()
```



[470]:
```
new_df = temp_df[temp_df['region']== 'UK'] #Taking the country UK
plt.figure(figsize=(20, 10))
heatmap_data = new_df.pivot_table(index='Sport', columns='Year',␣
 ↪values='Medal', aggfunc='count').fillna(0).astype(int)
sns.heatmap(heatmap_data, annot=True).set_title('Heatmap of Events per Sport of␣
 ↪UK')

plt.show()
```



[471]:
```
#Finding the most successful athlete who has won the maximum number of medal␣
 ↪from a particular country(Here we are taking India)

def most_successful(df, country):
    # Drop rows where 'Medal' is NaN
    temp_df = df.dropna(subset=['Medal'])

    # If a specific sport is provided, filter the DataFrame by that country
    if country != 'Overall':
        temp_df = temp_df[temp_df['region'] == country]

    # Count the number of medals each athlete has won
    athlete_medal_counts = temp_df['Name'].value_counts().reset_index().head(15)
    athlete_medal_counts.columns = ['Name', 'Medal Count']

    # Merge with the original DataFrame to get additional athlete information
    result_df = athlete_medal_counts.merge(df, left_on='Name', right_on='Name',␣
 ↪how='left').drop_duplicates('Name')

    return result_df[['Name', 'Medal Count', 'Sport']]

# Example usage
most_successful (df,'India')
```

[471]:

	Name	Medal Count	Sport
0	Leslie Walter Claudius	4	Hockey
4	Udham Singh Kular	4	Hockey
8	Victor John "V. J." Peter	3	Hockey
11	Dhyan Chand Bais	3	Hockey
14	Richard James Allen	3	Hockey
17	Shankar Pillay Laxman	3	Hockey
20	Balbir Singh	3	Hockey
23	Harbinder Singh Chimni	3	Hockey
26	Prithipal Singh	3	Hockey
29	Ranganathan Francis	3	Hockey
32	Balbir Singh Dosanjh, Sr.	3	Hockey
35	Randhir Singh Gentle	3	Hockey
38	Sushil Kumar Solanki	2	Wrestling
41	Jagjit Singh Kular	2	Hockey
43	Ajitpal Singh Kular	2	Hockey

In this section, the analysis explores the country-wise distribution of medals using data specific to the United Kingdom and India. Visualizations including a heatmap and lists of successful athletes illustrate the performance of these countries across various Olympic Games, showcasing their historical dominance and emerging trends in global sports competition.

2.3.1 Relationship Between Age and Medal Count

[472]:
```
import plotly.figure_factory as ff

athlete_df = df.drop_duplicates(subset=['Name','region'])
x_1 = athlete_df['Age'].dropna()
x_2 = athlete_df[athlete_df['Medal'] == 'Gold']['Age'].dropna()
x_3 = athlete_df[athlete_df['Medal'] == 'Silver']['Age'].dropna()
x_4 = athlete_df[athlete_df['Medal'] == 'Bronze']['Age'].dropna()

fig = ff.create_distplot(
    [x_1, x_2, x_3, x_4],
    ['Overall Age', 'Gold medalist', 'Silver Medalist', 'Bronze Medalist'],
    show_hist=False,  # Disable histogram
    show_rug=False  # Disable rug plot
)

fig.update_layout(
    title={
        'text': '<b>Age vs Medal</b><br><sub>Distribution of Medal Type␣
 ↪according to Age</sub>',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```

To understand the correlation between age and Olympic success, scatter plots and regression analysis were used to visualize how athletes' ages correlate with their medal counts. This analysis provides insights into whether there is an optimal age range for achieving athletic success at the Olympics and how age may influence medal-winning performances over time.

3 Distribution of Age according to every sport

[473]:
```
famous_sports = ['Basketball', 'Judo', 'Football', 'Tug-Of-War', 'Athletics',
'Swimming', 'Badminton', 'Sailing', 'Gymnastics',
'Art Competitions', 'Handball', 'Weightlifting', 'Wrestling',
'Water Polo', 'Hockey', 'Rowing', 'Fencing',
'Shooting', 'Boxing', 'Taekwondo', 'Cycling', 'Diving', 'Canoeing', 'Tennis',␣
 ↪'Golf', 'Softball', 'Archery',
'Volleyball', 'Synchronized Swimming', 'Table Tennis', 'Baseball',
'Rhythmic Gymnastics', 'Rugby Sevens',
'Beach Volleyball', 'Triathlon', 'Rugby', 'Polo', 'Ice Hockey']
```

[474]:
```
x = []
name = []
for sport in famous_sports :
    temp_df = athlete_df[athlete_df['Sport']== sport]
    x.append(temp_df[temp_df['Medal'] == 'Gold']['Age'].dropna())
    name.append(sport)
```

[475]:
```
fig = ff.create_distplot(x,name,show_hist=False,show_rug=False)
fig.update_layout(
    title={
        'text': '<b>Age vs Sport</b>',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```

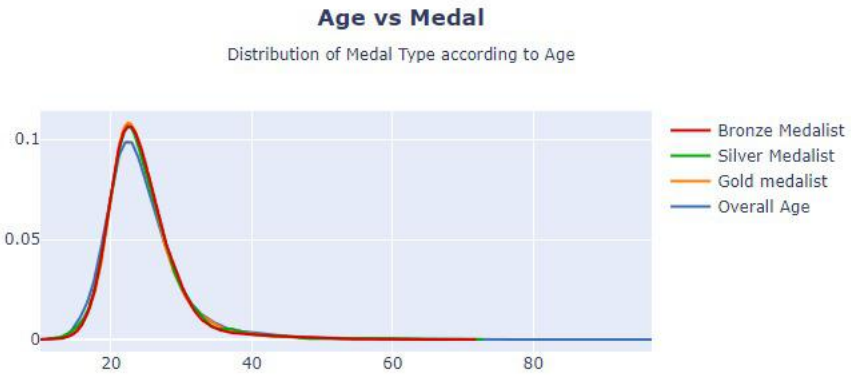
In this section, the analysis explores how age correlates with athletes' participation in various Olympic sports events. Histograms and density plots are employed to visualize age distributions across different sports, providing insights into the demographics of Olympic athletes. Sharp peaks in the curves indicate that certain ages may have a higher probability of athletes participating in sports events, suggesting optimal age ranges for peak performance. In contrast, broader or flatter curves suggest that the probability of participation remains relatively consistent across different age groups, indicating more uniform age distribution in those sports.

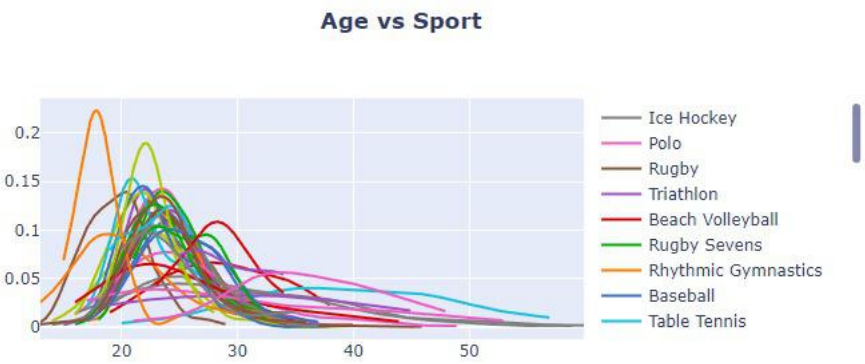
3.1 Relationship Between Height, Weight, and Athletes' Medal Wins

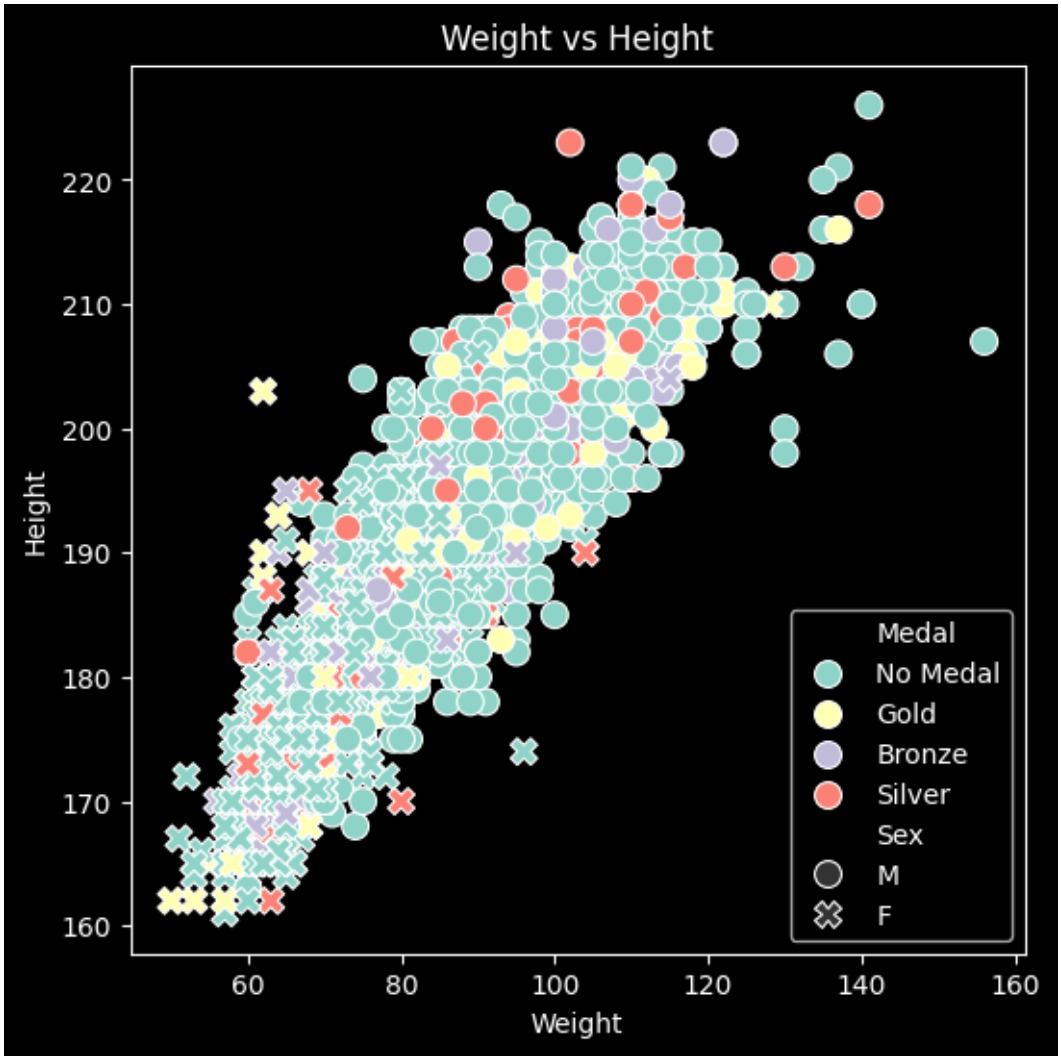
[476]:
```
athlete_df['Medal'].fillna('No Medal',inplace=True)
```

```
C:\Users\madhu\AppData\Local\Temp\ipykernel_1524\4160890072.py:1: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

[477]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Assuming athlete_df is already defined and loaded

plt.figure(figsize=(6, 6))

# Filter the DataFrame for the specific sport
temp_df = athlete_df[athlete_df['Sport'] == 'Basketball']

# Create scatter plot
sns.scatterplot(data=temp_df, x='Weight', y='Height', hue='Medal', style='Sex',␣
↪s=100)

# Set plot title and labels
plt.title('Weight vs Height')
plt.xlabel('Weight')
plt.ylabel('Height')

# Display the plot
plt.show()
```







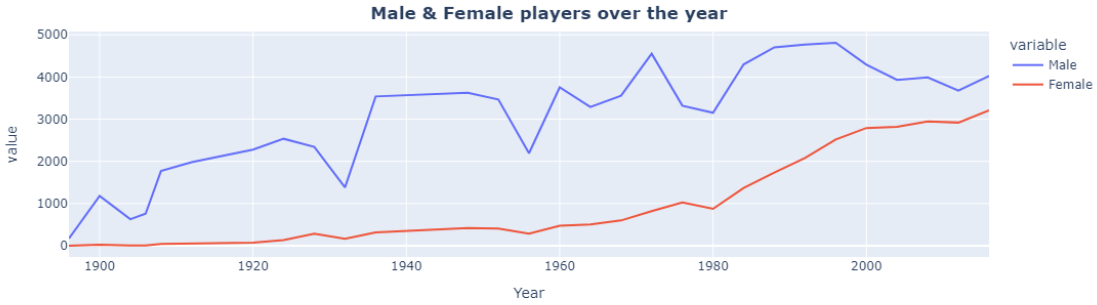
This section examines the relationship between athletes' height, weight, and their success in winning medals at the Olympics. Visualizations such as scatter plots and box plots are used to explore how the distribution of height and weight varies among male and female athletes across different medal categories (gold, silver, bronze). These visual analyses provide insights into any correlations between physical attributes and medal-winning performances, highlighting potential trends or differences between male and female athletes in achieving Olympic success.

3.2 Distribution of Male and Female Participants Over Time

[478]:
```
#In each year how many male participant were there
men = athlete_df[athlete_df['Sex'] == 'M'].groupby('Year').count()['Name'].
 ↪reset_index()
#In each year how many female participant were there
women = athlete_df[athlete_df['Sex'] == 'F'].groupby('Year').count()['Name'].
 ↪reset_index()
```

```
#merging both
final = men.merge(women , on = 'Year', how = 'left')
final.rename (columns = {'Name_x' : 'Male' , 'Name_y' : 'Female'},inplace=True)
final.fillna(0, inplace = True)
```

[479]:
```
import plotly.express as px
fig = px.line(final, x='Year', y=['Male' , 'Female'])
fig.update_layout(
    title={
        'text': '<b>Male & Female players over the year</b>',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```



In this section, the analysis explores the historical distribution of male and female participants in the Olympic Games over the years. Line charts and bar graphs are utilized to visualize how the representation of male and female athletes has evolved since the inception of the modern Olympic Games in 1896. These visualizations provide insights into the changing demographics and trends in gender representation within the Olympic movement, reflecting advancements in gender equality and inclusivity in sports over the past century.

3.3 Conclusion

The Olympic Analysis project delved into 120 years of Olympic history, using technical analysis and data visualization to explore participation trends, athlete achievements, and demographic shifts. It highlighted the global growth of the Games, evolution of sports events, standout athlete performances, and advancements in gender representation. This project not only celebrates Olympic achievements but also sets a foundation for future research in sports analytics and Olympic studies, emphasizing the Games' enduring impact on global sports culture and unity.

[ ]: