Project

July 11, 2024

0.0.1 Importing necessary libraries

[2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

0.0.2 To read csv dataset

We will use pandas to read csv file and to store it in a dataframe as follws:

[4]:
```
df = pd.read_csv(r'data.csv')   #to read the csv file
df
```

[4]:

	name	age	country	year	Date_given	sports	gold_medal	silver_medal	bronze_medal	total_medal
0	Michael Phelps	23	United States	2008	24-08-2008	Swimming	8	0	0	8
1	Michael Phelps	19	United States	2004	29-08-2004	Swimming	6	0	2	8
2	Michael Phelps	27	United States	2012	12-08-2012	Swimming	4	2	0	6
3	Natalie Coughlin	25	United States	2008	24-08-2008	Swimming	1	2	3	6
4	Aleksey Nemov	24	Russia	2000	01-10-2000	Gymnastics	2	1	3	6
…	…	…	…	…	…	…	…	…	…	…
8613	Olena Sadovnycha	32	Ukraine	2000	01-10-2000	Archery	0	1	0	1
8614	Kateryna Serdiuk	17	Ukraine	2000	01-10-2000	Archery	0	1	0	1
8615	Wietse van Alten	21	Netherlands	2000	01-10-2000	Archery	0	0	1	1
8616	Sandra Wagner-Sachse	31	Germany	2000	01-10-2000	Archery	0	0	1	1
8617	Rod White	23	United States	2000	01-10-2000	Archery				

```
8617           0              0              1              1

[8618 rows x 10 columns]
```

0.0.3 Basic Exploration of the dataset

As we have successfully imported the csv file into a dataFrame, we will perform basic exploration of the dataset to understand the given data.

[6]:
```
#To know the size of database using shape function
df.shape
```

[6]:
```
(8618, 10)
```

We can see that there are total 8618 number of rows and 10 number of columns are present in our given dataset.

[8]:
```
#To check for all column present in the data
df.columns
```

[8]:
```
Index(['name', 'age', 'country', 'year', 'Date_given', 'sports', 'gold_medal',
       'silver_medal', 'bronze_medal', 'total_medal'],
      dtype='object')
```

[9]:
```
# Exploring first 5 rows of the dataset using head function
df.head(5)
```

[9]:

	name	age	country	year	Date_given	sports	gold_medal	silver_medal	bronze_medal	total_medal
0	Michael Phelps	23	United States	2008	24-08-2008	Swimming	8	0	0	8
1	Michael Phelps	19	United States	2004	29-08-2004	Swimming	6	0	2	8
2	Michael Phelps	27	United States	2012	12-08-2012	Swimming	4	2	0	6
3	Natalie Coughlin	25	United States	2008	24-08-2008	Swimming	1	2	3	6
4	Aleksey Nemov	24	Russia	2000	01-10-2000	Gymnastics	2	1	3	6

[10]:
```
# Exploring last 5 rows of dataset using tail function.
df.tail(5)
```

[10]:

	name	age	country	year	Date_given	sports	gold_medal	silver_medal	bronze_medal	total_medal
8613	Olena Sadovnycha	32	Ukraine	2000	01-10-2000	Archery	0	0	1	1
8614	Kateryna Serdiuk	17	Ukraine	2000	01-10-2000	Archery	0	0	1	1
8615	Wietse van Alten	21	Netherlands	2000	01-10-2000	Archery	0	0	1	1

```
8616  Sandra Wagner-Sachse   31        Germany  2000  01-10-2000  Archery
8617             Rod White   23  United States  2000  01-10-2000  Archery

      gold_medal  silver_medal  bronze_medal  total_medal
8613           0             1             0            1
8614           0             1             0            1
8615           0             0             1            1
8616           0             0             1            1
8617           0             0             1            1
```

[11]:
```
#To check for basic information of the dataset such as null elements and␣
 ↪datatypes of column, memory usage by data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8618 entries, 0 to 8617
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          8613 non-null   object
 1   age           8618 non-null   int64
 2   country       8618 non-null   object
 3   year          8618 non-null   int64
 4   Date_given    8618 non-null   object
 5   sports        8618 non-null   object
 6   gold_medal    8618 non-null   int64
 7   silver_medal  8618 non-null   int64
 8   bronze_medal  8618 non-null   int64
 9   total_medal   8618 non-null   int64
dtypes: int64(6), object(4)
memory usage: 673.4+ KB
```

Using info() function we get to know following information about the given dataset:

- There are total 8618 entries i.e. there are 8618 rows and 10 columns.
- For name and age columns Non-Null Count is less than total rows which suggest missing values are present.
- All column's datatypes are reading correctly.

0.0.4 Dealing with Missing values and Duplicate values using pandas

Handling missing values and duplicate values is the important step to perform before analysis of data. Presence of this kind of data can affect our analysis and create bias in the study of dataset. It can cause error while giving output. Therefore, data cleansing is important step in data analysis.

First, we will check for null values in the dataset after that, we will check duplicate values.

[14]:
```
#to check which column have missing values using isna()
df.isna().any()
```

[14]:
```
name             True
age             False
country         False
year            False
Date_given      False
sports          False
gold_medal      False
silver_medal    False
bronze_medal    False
total_medal     False
dtype: bool
```

The isna().any() returns output in the boolean form. From above we can see that for name and age column value is True, which means null values or NaN values are present in those two columns.

[16]:
```
#to check number of missing values
df.isna().sum()
```

[16]:
```
name            5
age             0
country         0
year            0
Date_given      0
sports          0
gold_medal      0
silver_medal    0
bronze_medal    0
total_medal     0
dtype: int64
```

There are 5 null values present. We will explore rows having null values as follows:

[18]:
```
df[df.isna().any(axis=1)]
```

[18]:

	name	age	country	year	Date_given	sports	gold_medal	silver_medal	bronze_medal	total_medal
1428	NaN	0	Brazil	2012	12-08-2012	Volleyball	0	1	0	1
1429	NaN	0	Brazil	2012	12-08-2012	Volleyball	0	1	0	1
1430	NaN	0	Brazil	2012	12-08-2012	Volleyball	0	1	0	1
4485	NaN	0	Argentina	2012	12-08-2012	Hockey	0			
4486	NaN	0	Argentina	2012	12-08-2012	Hockey	0			

```
4485              1              0              1
4486              1              0              1
```

We can see these 5 entries are having NaN values i.e. missing values.

Futher, we will check for duplicates present.

[20]:
```
#to check for count of duplicates
df.duplicated().sum()
```

[20]:
```
3
```

[21]:
```
#exploring duplicated rows present in df
df[df.duplicated()]
```

[21]:
```
      name  age    country  year  Date_given      sports  gold_medal  \
1429   NaN    0     Brazil  2012  12-08-2012  Volleyball           0
1430   NaN    0     Brazil  2012  12-08-2012  Volleyball           0
4486   NaN    0  Argentina  2012  12-08-2012      Hockey           0

      silver_medal  bronze_medal  total_medal
1429             1             0            1
1430             1             0            1
4486             1             0            1
```

From above we can explore 3 duplicate values which are present only for columns having NaN values.

As we can see there are only three duplicate rows in entire dataset. Our dataset and findings will not affect if we drop them. Hence we will drop duplicate rows.

[23]:
```
#dropping all duplicates from df
df.drop_duplicates(inplace=True)
```

[24]:
```
#to recheck if all duplicates are removed or not
df.duplicated().sum()
```

[24]:
```
0
```

All duplicates are successfully removed from our original dataset, as we have set inplace parameter true. Now our data set has 8615 rows.

Now, we will fill 'unknown' value for NaN values in name column using fillna(). After that we will cross check if values are filled successfully or not using index.

[26]:
```
df['name'].fillna('unknown', inplace =True)
df.iloc[[1428,4483]]
```

[26]:
```
          name  age    country  year  Date_given      sports  gold_medal  \
1428   unknown    0     Brazil  2012  12-08-2012  Volleyball           0
4485   unknown    0  Argentina  2012  12-08-2012      Hockey           0

      silver_medal  bronze_medal  total_medal
1428             1             0            1
4485             1             0            1
```

The NaN values in the name column has been successfully filled by 'unknown'.

For filling age column, we can use mean value of age. As there are only two countries having NaN values, we can also use mean value for each country separately. For that, first we have to subset dataset for column country having Brazil and Argentina. And later we can find mean age for each country as follows:

[28]:
```
df_brazil = df[df["country"]=="Brazil"]          #dataset having country as␣
 ↪Brazil only
df_brazil[['age']].mean()
```

[28]:
```
age    26.399083
dtype: float64
```

[29]:
```
df_argentina = df[df["country"]=="Argentina"]    #dataset having country as␣
 ↪Argentina only
df_argentina[["age"]].mean()
```

[29]:
```
age    26.157143
dtype: float64
```

[30]:
```
df[['age']].mean()    #mean age of whole dataset
```

[30]:
```
age    26.399304
dtype: float64
```

But after finding mean ages for Brazil, and Argentina, we can see that it is almost equal. Also they are equal to mean age for whole dataset. Therefore we will fill NaN values in age column with mean age which is 26.

[32]:
```
df['age'].fillna(26, inplace =True)
df.iloc[[1428,4483]]
```

[32]:
```
          name  age    country  year  Date_given      sports  gold_medal  \
1428   unknown    0     Brazil  2012  12-08-2012  Volleyball           0
4485   unknown    0  Argentina  2012  12-08-2012      Hockey           0

      silver_medal  bronze_medal  total_medal
1428             1             0            1
4485             1             0            1
```

The NaN values in the age column has been successfully filled by 26.

0.0.5 To save cleaned csv file to the system

Now we will save this cleaned dataset to our system using df.to_csv to the path we will provide as olympic_data.csv.

[35]:
```
df.to_csv(r'olympics_data.csv', index = False)
```

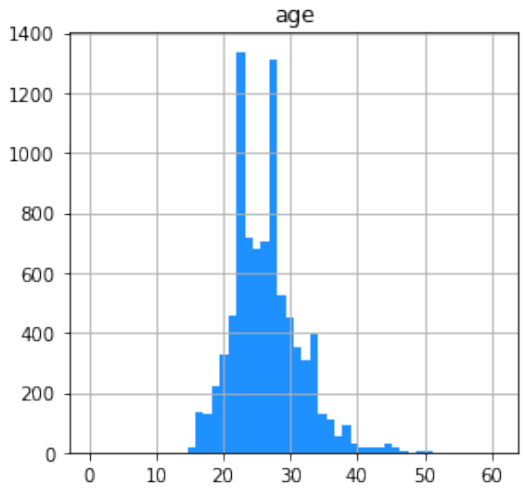
We have successfully stored csv file on given location

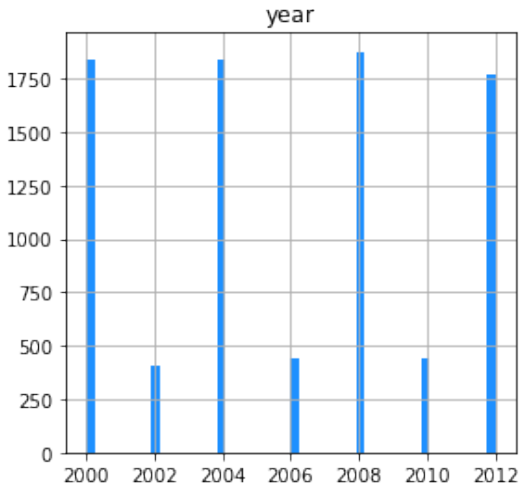
0.0.6 Data Visualisation

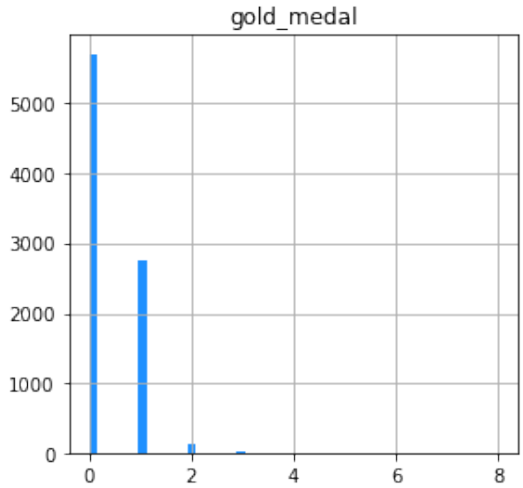
Data visualization is an important aspect, as we can gain key insights into our data through different graphical representations. It will help in making statistic analysis easy and comprehensive.

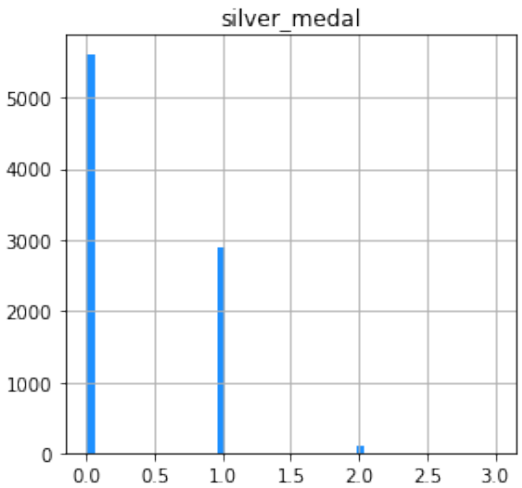
We will plot histplot maxtrix first, as it shows distribution of a numeric variable's values as a series of bars for all numeric variables present.

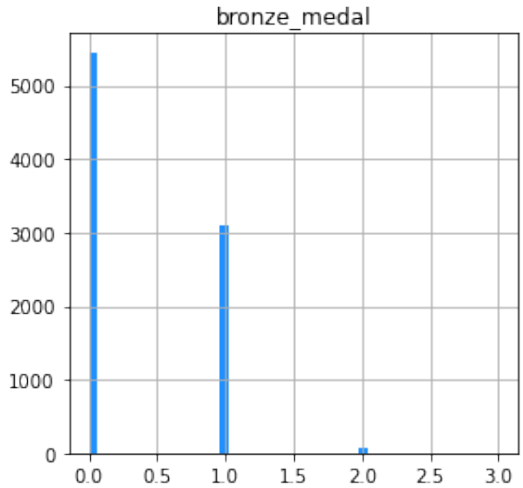
[38]:
```
%matplotlib inline
df.hist(bins=50, figsize=(10, 15),facecolor="dodgerblue")
plt.show()
```

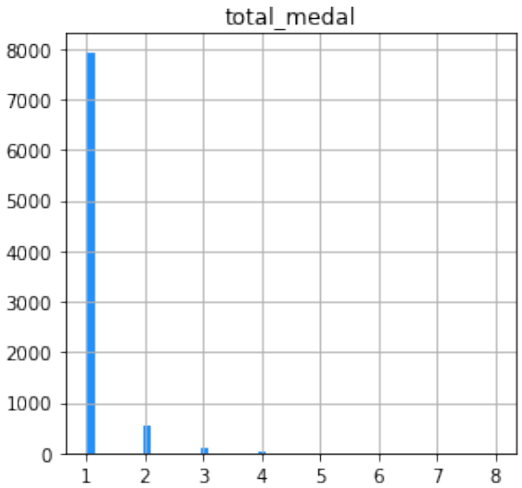

age
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50
60


year
1750
1500
1250
1000
750
500
250
0
2000
2002
2004
2006
2008
2010
2012


gold_medal
5000
4000
3000
2000
1000
0
0
2
4
6
8

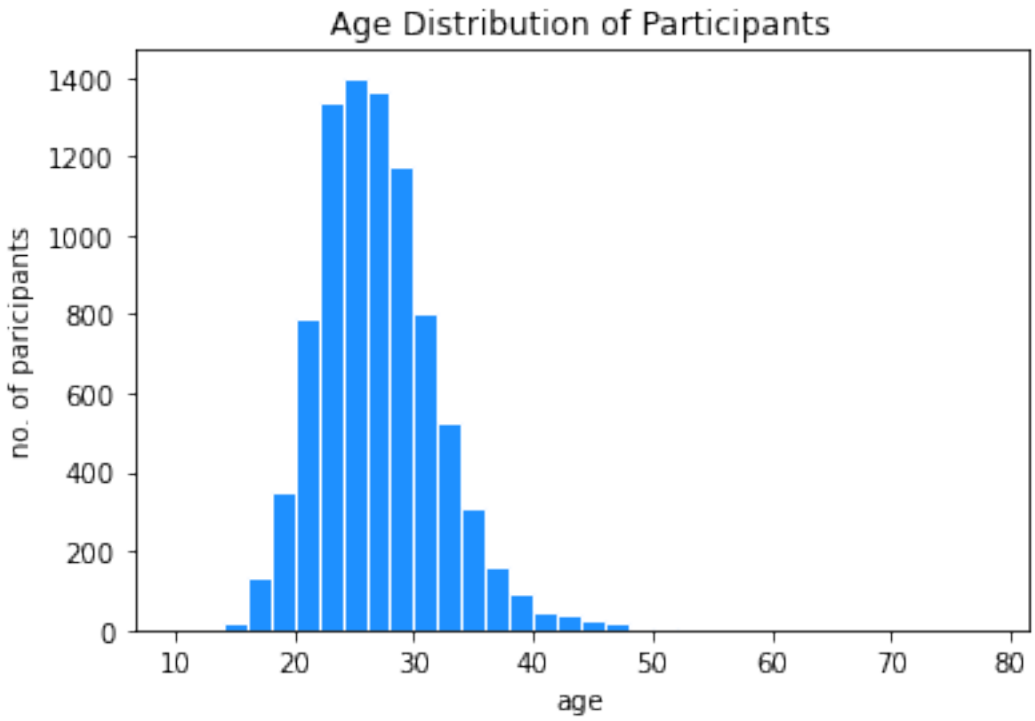

silver_medal
5000
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


bronze_medal
5000
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


total_medal
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8

0.0.7 To check 'age' distribution of Participants

[40]:
```
plt.hist(df.age,bins=np.arange(10,80,2),␣
  ↪edgecolor="white",facecolor="dodgerblue")
plt.xlabel("age")
plt.ylabel("no. of paricipants")
plt.title("Age Distribution of Participants")
plt.show()
```



0.0.8 Performing appropriate Numerical Measures on each column

The describe() function generates summary of descriptive statistics for all the columns. It can analyze both numeric as ell as object type data.

0.0.9 Applying describe() to numeric type data columns

For numeric data the, result's index will include count, mean, std, min, max, IQR i.e. 25, 50, and 75 percentiles.

[42]:
```
df.describe()
```

[42]:

	age	year	gold_medal	silver_medal	bronze_medal	\
count	8615.000000	8615.000000	8615.000000	8615.000000	8615.000000	
mean	26.399304	2005.976785	0.364829	0.363088	0.377829	

```
std       5.117364     4.289221     0.545401      0.511564      0.505044
min       0.000000  2000.000000     0.000000      0.000000      0.000000
25%      23.000000  2002.000000     0.000000      0.000000      0.000000
50%      26.000000  2006.000000     0.000000      0.000000      0.000000
75%      29.000000  2010.000000     1.000000      1.000000      1.000000
max      61.000000  2012.000000     8.000000      3.000000      3.000000

       total_medal
count  8615.000000
mean      1.105746
std       0.408958
min       1.000000
25%       1.000000
50%       1.000000
75%       1.000000
max       8.000000
```

Where,

count is the number of objects in the column.

mean is the mean value of the (numeric)column.

std is the Standard Deviation of values in the column.

min is the minimum value appearing in the column.

25% is the 25th percentile of values in the column.

50% is the 50th percentile of values in the column.

75% is the 75th percentile of values in the column.

max is the maximum value appearing in the column.

The result of df.describe() is self-explanatory. We can check various outcomes in the output.

0.0.10 Applying describe() to object type data columns

For object data the result's index will include count, unique, top, and freq.

[45]:
```
df.describe(include=object)
```

[45]:
```
                name        country  Date_given     sports
count           8615           8615        8615       8615
unique          6958            110           7         49
top     Matt Wells  United States  24-08-2008  Athletics
freq               4           1109        1872        687
```

Where,

- **count** is the number of objects in the column.
- **unique** is the number of distinct object in the column.

- **top** is the Most frequently occurring object in the column.
- **freq** is te number of times the top appearing object in the column.

The result of df.describe(include=object) is self-explanatory. We can check various outcomes in the output.

Additionally, we can explore correlation and variance for numeric type data column.

0.0.11 Exploring country column

We can check which country is having maximum participants in the over all years from 2000 to 2012.

[48]:
```
df.country.value_counts().sort_values(ascending=False).head(10)
```

[48]:
```
United States    1109
Russia            706
Germany           552
Australia         524
China             450
Canada            351
Italy             307
Great Britain     296
France            287
Netherlands       286
Name: country, dtype: int64
```

The United States is highly participating in the Olympic Games. Also we have the top ten countries having highest participants.

0.0.12 Exploring year column

We can find out years for which Olympic's data is given as bellow:

[50]:
```
df["year"].unique()
```

[50]:
```
array([2008, 2004, 2012, 2000, 2006, 2010, 2002])
```

0.0.13 Exploring country column

We will check sportsperson from India who is in our dataset.

[52]:
```
df_india = df.query('country == "India"')
df_india
```

[52]:

	name	age	country	year	Date_given	sports	\
702	Yogeshwar Dutt	29	India	2012	12-08-2012	Wrestling	
722	Sushil Kumar	29	India	2012	12-08-2012	Wrestling	
797	Sushil Kumar	25	India	2008	24-08-2008	Wrestling	
1094	Karnam Malleswari	25	India	2000	01-10-2000	Weightlifting	

```
2786           Vijay Kumar   26   India  2012  12-08-2012       Shooting
2791         Gagan Narang    29   India  2012  12-08-2012       Shooting
2807       Abhinav Bindra    25   India  2008  24-08-2008       Shooting
2877  Rajyavardhan Rathore   34   India  2004  29-08-2004       Shooting
7170       M. C. Mary Kom    29   India  2012  12-08-2012         Boxing
7235       Vijender Singh    22   India  2008  24-08-2008         Boxing
7806         Saina Nehwal    22   India  2012  12-08-2012      Badminton

      gold_medal  silver_medal  bronze_medal  total_medal
702            0             0             1            1
722            0             1             0            1
797            0             0             1            1
1094           0             0             1            1
2786           0             1             0            1
2791           0             0             1            1
2807           1             0             0            1
2877           0             1             0            1
7170           0             0             1            1
7235           0             0             1            1
7806           0             0             1            1
```

[53]:
```
df_india.name.count()
```

[53]:
```
11
```

[54]:
```
df_india["year"].unique()
```

[54]:
```
array([2012, 2008, 2000, 2004])
```

[55]:
```
df_india["sports"].unique()
```

[55]:
```
array(['Wrestling', 'Weightlifting', 'Shooting', 'Boxing', 'Badminton'],
      dtype=object)
```

There are 11 sportsperson from India who won medals in the Olympic Games in the years 2000,2004,2008 and 2012. They have participated in sports listed as 'Wrestling', 'Weightlifting', 'Shooting', 'Boxing', 'Badminton'

0.0.14 Exploring sports column

We know there are 49 types of different sports are involved in the Olympics from describe() function. Now we will find out names of different types of sports played in Olympics as follows:

[58]:
```
df["sports"].unique()
```

[58]:
```
array(['Swimming', 'Gymnastics', 'Speed Skating', 'Cross Country Skiing',
       'Short-Track Speed Skating', 'Diving', 'Cycling', 'Biathlon',
       'Alpine Skiing', 'Ski Jumping', 'Nordic Combined', 'Athletics',
```

```
       'Table Tennis', 'Tennis', 'Synchronized Swimming', 'Shooting',
       'Rowing', 'Fencing', 'Equestrian', 'Canoeing', 'Bobsleigh',
       'Badminton', 'Archery', 'Wrestling', 'Weightlifting', 'Waterpolo',
       'Volleyball', 'Triathlon', 'Trampoline', 'Taekwondo', 'Softball',
       'Snowboarding', 'Skeleton', 'Sailing', 'Rhythmic Gymnastics',
       'Modern Pentathlon', 'Luge', 'Judo', 'Ice Hockey', 'Hockey',
       'Handball', 'Football', 'Figure Skating', 'Freestyle Skiing',
       'Curling', 'Baseball', 'Boxing', 'Beach Volleyball', 'Basketball'],
      dtype=object)
```

[59]:
```
df.groupby("sports")["name"].unique().head(10)
```

[59]:
```
sports
Alpine Skiing       [Janica Kostelic, Bode Miller, Aksel Lund Svin…
Archery             [Ki Bo-Bae, Oh Jin-Hyek, Park Gyeong-Mo, Park …
Athletics           [Yohan Blake, Usain Bolt, Allyson Felix, Shell…
Badminton           [Zhao Yunlei, Lee Hyo-Jeong, Yu Yang, Gao Ling…
Baseball            [Brett Anderson, Jake Arrieta, Brian Barden, A…
Basketball          [Carmelo Anthony, Semyon Antonov, Seimone Augu…
Beach Volleyball    [Julius Brink, Alison Cerutti, Emanuel, Julian…
Biathlon            [Ole Einar BjÃ¸rndalen, Magdalena Neuner, Emil…
Bobsleigh           [Kevin Kuske, AndrÃ© Lange, Martin Annen, Beat…
Boxing              [Nicola Adams, Misha Aloyan, LÃ¡zaro Ã lvarez,…
Name: name, dtype: object
```

0.0.15 Top 10 countries with the Gold Medal

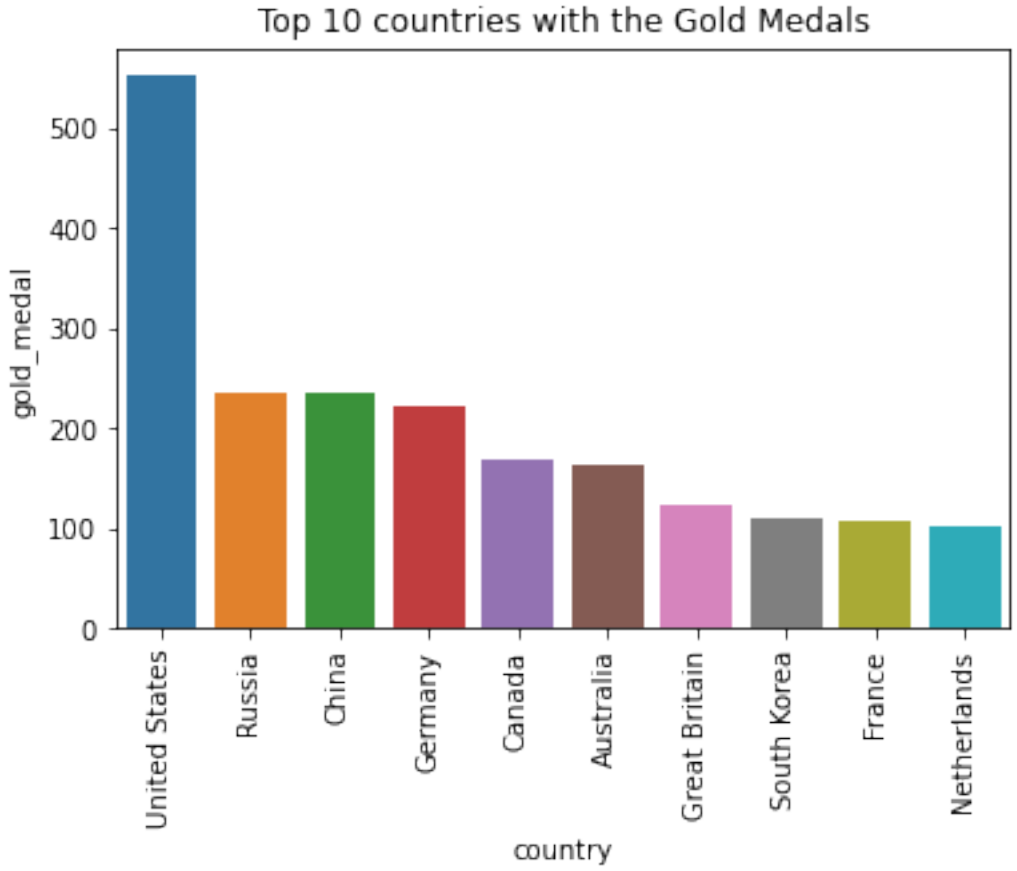
[61]:
```
df_medals = df.groupby('country').sum()
gold_medals = df_medals.sort_values(by=['gold_medal'],ascending=False,␣
 ↪inplace=False)
G = gold_medals[['gold_medal']].head(10)
G
```

[61]:

country	gold_medal
United States	552
Russia	234
China	234
Germany	223
Canada	168
Australia	163
Great Britain	124
South Korea	110
France	108
Netherlands	101

0.0.16 Visulization of the result using Barplot

[63]:
```
plt.title("Top 10 countries with the Gold Medals")
sns.barplot(x = G.index, y="gold_medal", data=G)
plt.xticks(rotation=90)
plt.show()
```



0.1 Top 10 countries with the Silver Medal

[65]:
```
silver_medals = df_medals.sort_values(by=['silver_medal'],ascending=False,␣
↪inplace=False)
S = silver_medals[['silver_medal']].head(10)
S
```

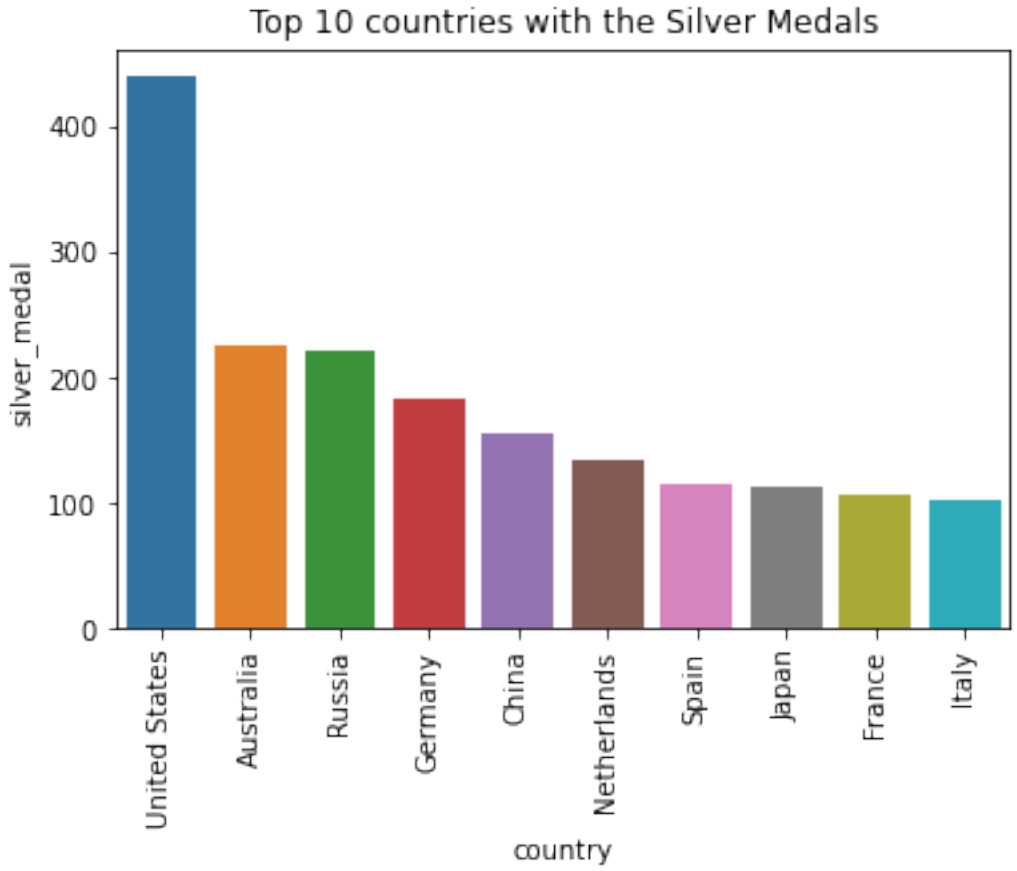
[65]:

country	silver_medal
United States	440
Australia	226
Russia	221

```
Germany                 183
China                   156
Netherlands             135
Spain                   116
Japan                   112
France                  107
Italy                   103
```

0.1.1 Visulization of the result using Barplot

[67]:
```
plt.title("Top 10 countries with the Silver Medals")
sns.barplot(x = S.index, y="silver_medal", data=S)
plt.xticks(rotation=90)
plt.show()
```



0.2 Top 10 countries with the Bronze Medal

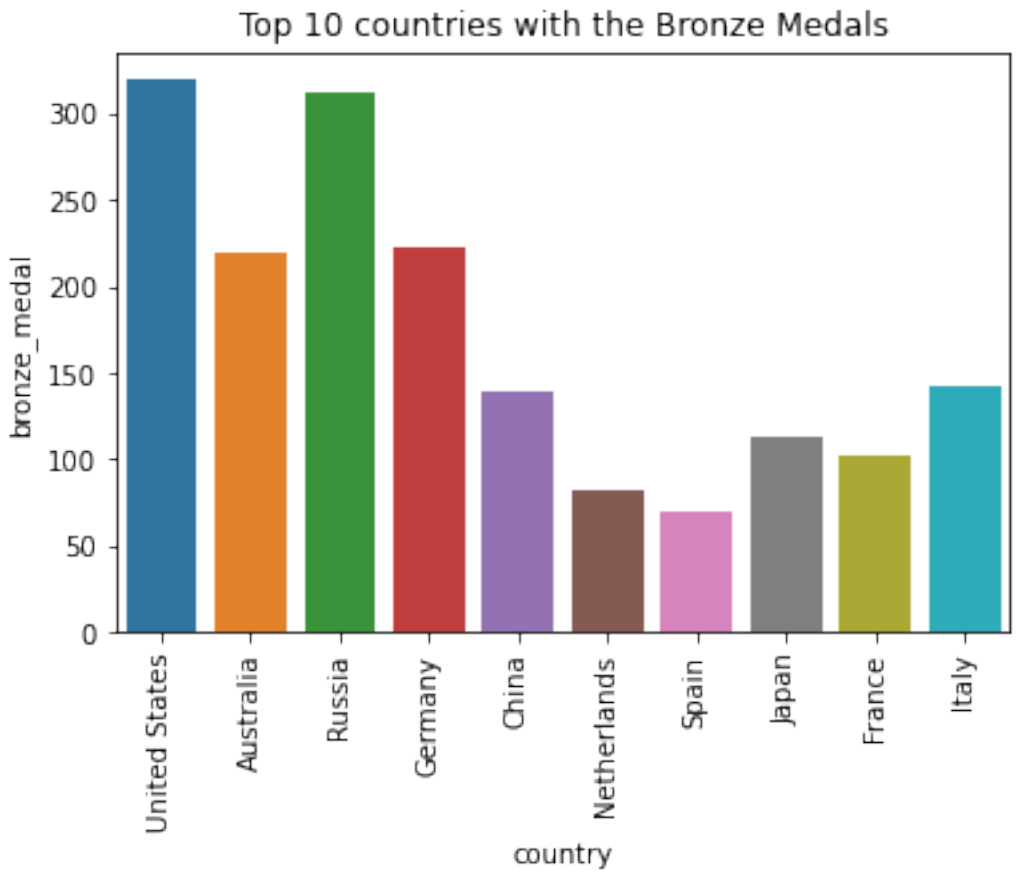
[69]:
```
bronze_medals = df_medals.sort_values(by=['silver_medal'],ascending=False,␣
↪inplace=False)
B = bronze_medals[['bronze_medal']].head(10)
B
```

[69]:

country	bronze_medal
United States	320
Australia	220
Russia	313
Germany	223
China	140
Netherlands	82
Spain	70
Japan	113
France	103
Italy	142

0.2.1 Visulization of the result using Barplot

[71]:
```
plt.title("Top 10 countries with the Bronze Medals")
sns.barplot(x = B.index, y="bronze_medal", data=B)
plt.xticks(rotation=90)
plt.show()
```



0.3 Top 10 countries with the Total Medals

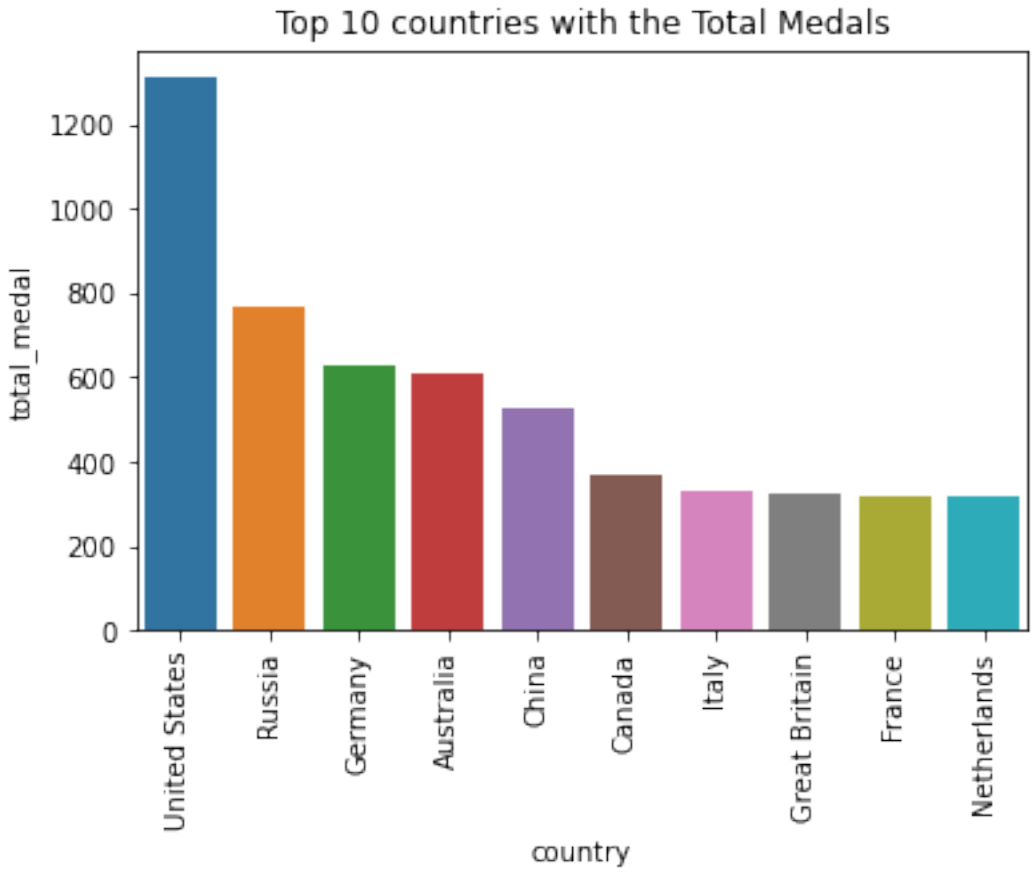
[73]:
```
total_medals = df_medals.sort_values(by=['total_medal'],ascending=False,␣
↪inplace=False)
T = total_medals[['total_medal']].head(10)
T
```

[73]:

country	total_medal
United States	1312
Russia	768
Germany	629
Australia	609
China	530
Canada	370
Italy	331
Great Britain	322
France	318
Netherlands	318

0.3.1 Visulization of the result using Barplot

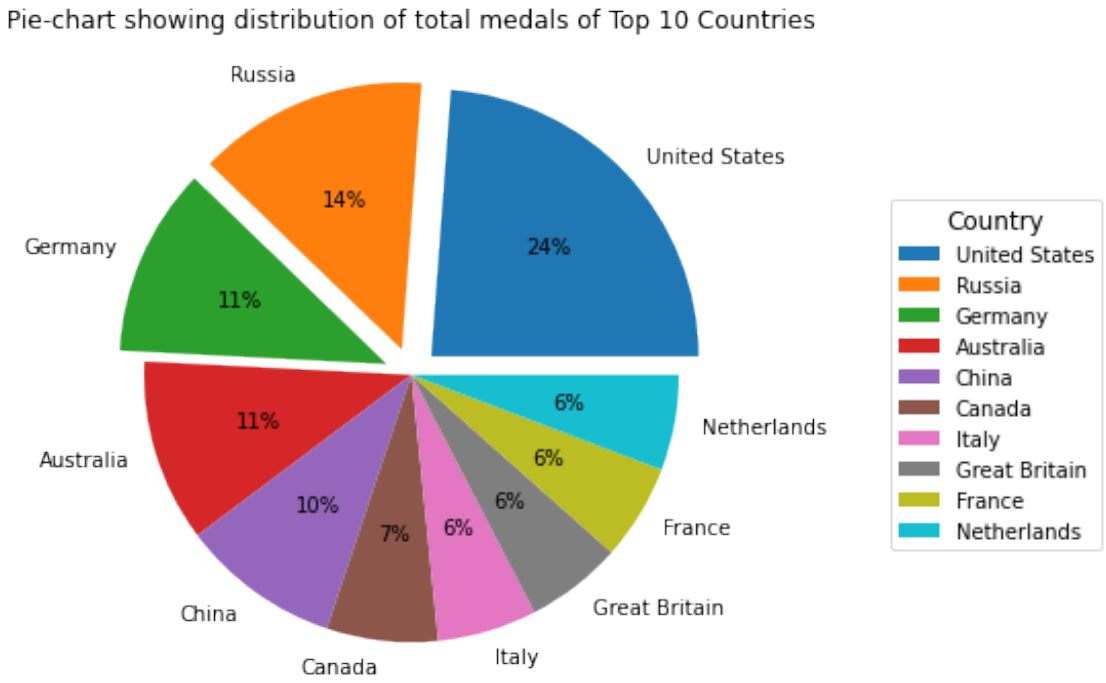
[75]:
```
plt.title("Top 10 countries with the Total Medals")
sns.barplot(x = T.index, y="total_medal", data=T)
plt.xticks(rotation=90)
plt.show()
```



0.4 Representing top 10 countries having maximum total medals using pie chart

For easy visalization, finding out contribution of top 10 countries in percentage having sum of three medals i.e. gold, silver and bronze from year 2000 to 2012.

[77]:
```
my_explode=(0.1,0.1,0.1,0,0,0,0,0,0,0)
T.total_medal.plot(kind='pie', autopct='%.0f%%',explode=my_explode,␣
 ↪label="",figsize=(6,6),legend=True)
plt.legend(title='Country',title_fontsize=12,loc='center left',␣
 ↪bbox_to_anchor=(1.2, 0.5))
plt.title('Pie-chart showing distribution of total medals of Top 10 Countries')
plt.show()
```



0.4.1 Conclusion

From the overall exploration of the data set, we can conclude as follows:

- Missing values and duplicate values didn't affect our exploration and findings, if handled properly.
- Average age of the athletes participated in the Olympics was 26 year.
- Total 49 sports were included for the year 2000-2012
- United States is having the highest number of total medals.
- Russia and Germany also showed good overall performance in the Olympics.
- India won 11 medals(1 Gold, 3 Silver and 7 Bronze) in the year 2000, 2004, 2008 and 2012.
- India won medals in 'Wrestling', 'Weightlifting', 'Shooting', 'Boxing', 'Badminton'
- Abhinav Bindra was the only Indian to win a Gold medal in the Olympic Games between 2000 and 2012.

##

SHUBH SHARMA 21013570096