MN5813: 120 years of Olympic history: athletes and results

Introduction: The Olympics have always been a fascinating lens through which we can explore global sports, culture, and history. This report dives into the data behind the games, uncovering patterns and insights about participation, medal achievements, and gender representation. Along the way, we tackled challenges like missing information, unusual data values, and the impact of historical events on medal counts. By carefully cleaning and analysing the dataset, we were able to paint a clearer picture of how the Olympics have evolved over time and what they reveal about the world of sports.

Aim: To uncover meaningful insights from Olympic data by analysing trends, medal distributions, and gender representation while addressing inconsistencies and historical nuances.

Objectives:

1. Clean and Prepare the Data: Handle missing values, inconsistent entries, and unusual records like negative ages to ensure the dataset is ready for analysis.
2. Understand Historical Context: Acknowledge the impact of geopolitical changes, such as the dissolution of the Soviet Union, on medal counts and representation.
3. Highlight Gender Representation: Compare male and female participation in both the Summer and Winter Olympics to identify trends over time.
4. Track Participation Trends: Explore how the number of participating countries has grown over the years for each Olympic season.
5. Analyse Medal Performances: Identify the top-performing countries and sports for both seasons to showcase global dominance and competitiveness.
6. Create Engaging Visuals: Use charts and graphs to make the findings easy to understand and visually appealing.

Link to GitHub:
https://github.com/riyashelwante/MN5813_Group

In [1]:
```
import pandas as pd

#read athlete_events.csv file
df = pd.read_csv('athlete_events.csv')

#print first 10 records
df.head(10)
```

Out[1]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Se
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Su
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Su
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Su
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Su
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
5	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
6	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
7	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
8	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V
9	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V

In [2]:

```
#convert the .csv file to .json file
df.to_json("athlete_events1.json", orient="records")
df_json1 = pd.read_json("athlete_events1.json")

#print first 10 records
df_json1.head(10)
```

Out[2]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Se
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Su
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Su
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Su
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Su
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
5	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
6	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
7	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
8	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V
9	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V

In [3]:

```
#DATA CLEANING

#find sum of duplicate records in the dataset
print(df_json1.duplicated().sum())
```

1385

In [4]:
```
#delete the duplicates
df_json1 = df_json1.drop_duplicates()
print(df_json1.duplicated().sum())
```

```
0
```

In [5]:
```
#printing the info inorder to know the datatype of each column
print(df_json1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 269731 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   ID      269731 non-null  int64  
 1   Name    269731 non-null  object 
 2   Sex     269731 non-null  object 
 3   Age     260416 non-null  float64
 4   Height  210917 non-null  float64
 5   Weight  208204 non-null  float64
 6   Team    269731 non-null  object 
 7   NOC     269731 non-null  object 
 8   Games   269731 non-null  object 
 9   Year    269731 non-null  int64  
 10  Season  269731 non-null  object 
 11  City    269731 non-null  object 
 12  Sport   269731 non-null  object 
 13  Event   269731 non-null  object 
 14  Medal   39772 non-null   object 
dtypes: float64(3), int64(2), object(10)
memory usage: 32.9+ MB
None
```

In [6]:
```
#converting the "Year" column to datetime datatype
#df_json1['Year']= pd.to_datetime(df_json1['Year'])
```

In [7]:
```
#printing the 'now correct' datatypes
print(df_json1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 269731 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   ID      269731 non-null  int64  
 1   Name    269731 non-null  object 
 2   Sex     269731 non-null  object 
 3   Age     260416 non-null  float64
 4   Height  210917 non-null  float64
 5   Weight  208204 non-null  float64
 6   Team    269731 non-null  object 
 7   NOC     269731 non-null  object 
 8   Games   269731 non-null  object 
 9   Year    269731 non-null  int64  
 10  Season  269731 non-null  object 
 11  City    269731 non-null  object 
 12  Sport   269731 non-null  object 
 13  Event   269731 non-null  object 
 14  Medal   39772 non-null   object 
dtypes: float64(3), int64(2), object(10)
memory usage: 32.9+ MB
None
```

In [8]:
```
#checking if dataset contains any null values and printing the sum of them
print(df_json1.isnull().sum())
```

```
ID              0
Name            0
Sex             0
Age          9315
Height      58814
Weight      61527
Team            0
NOC             0
Games           0
Year            0
Season          0
City            0
Sport           0
Event           0
Medal      229959
dtype: int64
```

In [9]:
```
#Handle missing age
#group the cloumns based on Name
group_columns = ['Name']

#iterate through each group based on Name
for group_keys, group_indices in df_json1.groupby(group_columns).groups.items():
    #sort by year
    group = df_json1.loc[group_indices].sort_values('Year')

    #iterate to find if the age is missing
    for i in range (len(group)):
        if pd.isnull(group.iloc[i]['Age']):
            #if age is known for previous row
            if i >0 and not pd.isnull(group.iloc[i-1]['Age']):
                year_diff = group.iloc[i]['Year'] - group.iloc[i-1]['Year']
                #year difference is then ADDED with previous given age
```

```
                df_json1.loc[group_indices[i],'Age']= group.iloc[i-1]['Age'] + y

            #if age is known for next row
            elif i<len(group) - 1 and not pd.isnull(group.iloc[i+1]['Age']):
                year_diff = group.iloc[i+1]['Year'] - group.iloc[i]['Year']
                #year difference is then SUBTRACTED with previous given age
                df_json1.loc[group_indices[i],'Age']= group.iloc[i+1]['Age'] - y

#using forward-fill and backward-fill to ensure no missing value remain
res = df_json1.groupby(group_columns)['Age'].apply(lambda x: x.ffill().bfill())
#
df_json1['Age'] = res.droplevel(0)
print('Missing Age Handled!')
df_json1
```

```
Missing Age Handled!
```

Out[9]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter
...	...	...	...	...	...	...	...	...	...
271111	135569	Andrzej ya	M	29.0	179.0	89.0	Poland-1	POL	1976 Winter
271112	135570	Piotr ya	M	27.0	176.0	59.0	Poland	POL	2014 Winter
271113	135570	Piotr ya	M	27.0	176.0	59.0	Poland	POL	2014 Winter
271114	135571	Tomasz Ireneusz ya	M	30.0	185.0	96.0	Poland	POL	1998 Winter
271115	135571	Tomasz Ireneusz ya	M	34.0	185.0	96.0	Poland	POL	2002 Winter

269731 rows × 15 columns

We noticed that the dataset had missing values in the **"Age"** column, which needed to be addressed for accurate analysis. To resolve this, we grouped the data by **"Name"**, allowing us to work on records associated with each individual. Within each group, we sorted the data by **"Year"**, making it easier to track the chronological progression of ages. For rows with missing ages, we applied a logical approach to estimate their values. If the previous record in the group had a known age, we calculated the difference in years and added it to the previous age. If the previous record wasn't available or didn't have a known age, we used the next record instead, subtracting the year difference to estimate the missing age. After filling as many gaps as possible, we applied forward-fill and backward-fill within each group to ensure no missing values remained. Finally, we updated the dataset with the completed "Age" values, leaving it fully prepared for further analysis.

In [10]:
```
#print to check how many missing age values are handled
print(df_json1.isnull().sum())
```

```
ID             0
Name           0
Sex            0
Age         9233
Height     58814
Weight     61527
Team           0
NOC            0
Games          0
Year           0
Season         0
City           0
Sport          0
Event          0
Medal     229959
dtype: int64
```

In [11]:
```
#copying the same height and weight as found for the person having 'notnull' rec
#assuming height and weight does not change as athletes prefer playing in their
df_json1['Height'] = df_json1.groupby('Name')['Height'].transform(lambda y : y.f
df_json1['Weight'] = df_json1.groupby('Name')['Weight'].transform(lambda z : z.f
```

In [12]:
```
#print to check how many missing height and weight values are handled
print(df_json1.isnull().sum())
```

```
ID             0
Name           0
Sex            0
Age         9233
Height     58426
Weight     61138
Team           0
NOC            0
Games          0
Year           0
Season         0
City           0
Sport          0
Event          0
Medal     229959
dtype: int64
```

In [13]:
```
#creating a new column 'Age_Group' to divide them into categories of 4
df_json1["Age_Group"] = pd.cut(df_json1["Age"], bins=[1,18,25,35,100], labels=["
df_json1.head(10)
```

Out[13]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Se
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Su
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Su
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Su
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Su
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
5	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
6	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
7	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
8	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V
9	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V

In [14]:

```
#creating a new column 'Century'
#subtract 1 from year so year like 2000 will be in 20th century
df_json1['Century'] = (df_json1['Year'] - 1) // 100 + 1
def add_suffix(century):
    #check the last 2 digits in order to add suffix, for eg: 11th, 12th, 13th
    if 11 <= century % 100 <= 13:
        return f"{century}th"
```

```
    last_digit = century % 10
    if last_digit == 1:
        return f"{century}st"
    elif last_digit == 2:
        return f"{century}nd"
    elif last_digit == 3:
        return f"{century}rd"
    else:
        return f"{century}th"

#append the values to existing 'Century' column
df_json1['Century'] = df_json1['Century'].apply(add_suffix)
df_json1.head(10)
```

Out[14]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Se
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Su
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Su
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Su
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Su
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
5	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	V
6	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
7	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	V
8	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V
9	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	V

We created a new column called **"Century"** to categorize the years in the dataset by their respective centuries. To ensure that years like 2000 were correctly assigned to the 20th century, we subtracted 1 from the year before performing integer division by 100 and adding 1. This calculated the numerical century for each year. Next, we defined a function called `add_suffix` to append the appropriate suffix (e.g., "st," "nd," "rd," or "th") to each century value. The function checked the last two digits of the century to handle special cases like 11th, 12th, and 13th correctly. For other values, it determined the suffix based on the last digit (e.g., 1st, 2nd, 3rd). Finally, we applied this function to the **"Century"** column to append the suffixes to the numerical values, resulting in a properly formatted and meaningful **"Century"** column. The updated dataset was displayed using `head(10)` to verify the results.

In [15]:
```
#Identify records with negative Age

import numpy as np

negative_age_records = df_json1[df_json1['Age'] < 0]

#Replace negative Age values with NaN
df_json1.loc[df_json1['Age'] < 0, 'Age'] = np.nan

#sorting the values by age and pushing all the NaN values to the last of dataset
df_json1.sort_values(by='Age', na_position='last').head(50)
```

Out[15]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Gam
45429	23439	Eugne Coulon	M	0.0	NaN	NaN	Pupilles de Neptune de Lille #2-1	FRA	19 Summ
81773	41516	Francisco Gonzlez Suaste	M	1.0	185.0	71.0	Mexico	MEX	19 Summ
102622	51917	Viktor Ilyin	M	4.0	160.0	60.0	Soviet Union	URS	19 Win
154785	77702	James McKenzie	M	4.0	NaN	NaN	Great Britain	GBR	19 Summ
154784	77701	James McKenzie	M	4.0	NaN	NaN	Great Britain	GBR	19 Summ
156926	78823	Pedro Mercado	M	5.0	NaN	NaN	Argentina	ARG	19 Summ
101943	51561	Mohamed Ibrahim	M	6.0	NaN	NaN	Egypt	EGY	19 Summ
256154	128257	Hugo Walser	M	8.0	185.0	67.0	Liechtenstein	LIE	19 Summ
135346	68101	Lee Sang-Hun	M	10.0	169.0	54.0	South Korea	KOR	19 Summ
142882	71691	Dimitrios Loundras	M	10.0	NaN	NaN	Ethnikos Gymnastikos Syllogos	GRE	18 Summ
73461	37333	Carlos Bienvenido Front Barrera	M	11.0	NaN	NaN	Spain	ESP	19 Summ
237141	118925	Megan Olwen Devenish Taylor (-Mandeville-Ellis)	F	11.0	157.0	NaN	Great Britain	GBR	19 Win
94058	47618	Sonja Henie (-Topping, -Gardiner, -Onstad)	F	11.0	155.0	45.0	Norway	NOR	19 Win

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Gam
102916	52070	Etsuko Inada	F	11.0	NaN	NaN	Japan	JPN	19 Win
252230	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	19 Summ
140650	70616	Liu Luyang	F	11.0	NaN	NaN	China	CHN	19 Win
252232	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	19 Summ
252233	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	19 Summ
101378	51268	Beatrice Hutiu	F	11.0	151.0	38.0	Romania	ROU	19 Win
152798	76675	Marcelle Matthews	F	11.0	NaN	NaN	South Africa	RSA	19 Win
79024	40129	Luigina Giavotti	F	11.0	NaN	NaN	Italy	ITA	19 Summ
252231	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	19 Summ
43468	22411	Magdalena Cecilia Colledge	F	11.0	152.0	NaN	Great Britain	GBR	19 Win
251495	125944	Ines Vercesi	F	12.0	NaN	NaN	Italy	ITA	19 Summ
97086	49142	Jan Hoffmann	M	12.0	178.0	73.0	East Germany	GDR	19 Win

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Gam
236977	118832	Chin Say "Molly" Tay	F	12.0	145.0	41.0	Malaysia	MAS	19 Summ
118048	59727	Marika Kilius (-Zahn, -Schfer)	F	12.0	168.0	51.0	Germany	GER	19 Win
96236	48728	Margery Hinton	F	12.0	168.0	NaN	Great Britain	GBR	19 Summ
60911	31203	Patricia Anne "Pat" Eastwood	F	12.0	NaN	NaN	South Africa	RSA	19 Win
79352	40296	Alain C. Giletti	M	12.0	NaN	NaN	France	FRA	19 Win
85840	43528	Antoinette Joyce Guedia Mouafo	F	12.0	170.0	55.0	Cameroon	CMR	20 Summ
226019	113580	Inge Srensen (-Tabur)	F	12.0	NaN	NaN	Denmark	DEN	19 Summ
126542	63816	Carolyn Patricia Krau	F	12.0	137.0	36.0	Great Britain-2	GBR	19 Win
93850	47506	Hem Reaksmey	F	12.0	175.0	65.0	Cambodia	CAM	19 Summ
197894	99362	Antonia Real Horrach	F	12.0	160.0	50.0	Spain	ESP	19 Summ
91913	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	19 Summ
91912	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	19 Summ
91911	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	19 Summ

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Gam
91910	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	19 Summ
148942	74712	Carla Marangoni	F	12.0	NaN	NaN	Italy	ITA	19 Summ
145150	72854	Licia Macchini	F	12.0	NaN	NaN	Italy	ITA	19 Summ
120234	60854	Judit Kiss	F	12.0	171.0	57.0	Hungary	HUN	19 Summ
120233	60854	Judit Kiss	F	12.0	171.0	57.0	Hungary	HUN	19 Summ
168941	84913	Clstine N'Drin	F	12.0	169.0	56.0	Cote d'Ivoire	CIV	19 Summ
168942	84913	Clstine N'Drin	F	12.0	169.0	56.0	Cote d'Ivoire	CIV	19 Summ
149033	74755	Luciana Marcellini Hercolani Gaddi	F	12.0	163.0	58.0	Italy	ITA	19 Summ
50294	25877	Olga Lucia de Angulo Irragorri	F	12.0	160.0	60.0	Colombia	COL	19 Summ
253662	127018	Yelena Germanovna Vodorezova (-Buyanova)	F	12.0	161.0	47.0	Soviet Union	URS	19 Win
96665	48939	Ho Gang	M	12.0	NaN	NaN	North Korea	PRK	19 Win
50293	25877	Olga Lucia de Angulo Irragorri	F	12.0	160.0	60.0	Colombia	COL	19 Summ

To address inconsistencies in the **"Age"** column, we identified records where the age had a negative value. Using a filter, we created a subset of the dataset containing only the rows with **negative Age** values. Recognizing that negative ages were invalid, we replaced these values with **NaN** (Not a Number) to signify missing or undefined data. After making these adjustments, we sorted the dataset by **"Age"**, ensuring that all rows with NaN values were pushed to the end of the dataset. Finally, we used `head(50)` to display the first 50 records, verifying that the data was cleaned and properly sorted.

In [16]:
```
#Some of the ages of records dont make sense, for eg: age cannot be 0 and 1 year
#So we are taking one minimum age to clean the data
#If you disagree feel free to change the minimum age

min_age = 11
df_json1.loc[df_json1['Age'] < min_age, 'Age'] = np.nan
```

In [17]:
```
df_json1.sort_values(by='Age', na_position='last').head(50)
```

Out[17]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Y
237141	118925	Megan Olwen Devenish Taylor (-Mandeville-Ellis)	F	11.0	157.0	NaN	Great Britain	GBR	1932 Winter	19
152798	76675	Marcelle Matthews	F	11.0	NaN	NaN	South Africa	RSA	1960 Winter	19
73461	37333	Carlos Bienvenido Front Barrera	M	11.0	NaN	NaN	Spain	ESP	1992 Summer	19
94058	47618	Sonja Henie (-Topping, -Gardiner, -Onstad)	F	11.0	155.0	45.0	Norway	NOR	1924 Winter	19
252230	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	1968 Summer	19
252231	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	1968 Summer	19
102916	52070	Etsuko Inada	F	11.0	NaN	NaN	Japan	JPN	1936 Winter	19
101378	51268	Beatrice Hutiu	F	11.0	151.0	38.0	Romania	ROU	1968 Winter	19
140650	70616	Liu Luyang	F	11.0	NaN	NaN	China	CHN	1988 Winter	19
43468	22411	Magdalena Cecilia Colledge	F	11.0	152.0	NaN	Great Britain	GBR	1932 Winter	19
252232	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	1968 Summer	19
252233	126307	Liana Vicens	F	11.0	158.0	50.0	Puerto Rico	PUR	1968 Summer	19

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Ye
79024	40129	Luigina Giavotti	F	11.0	NaN	NaN	Italy	ITA	1928 Summer	19
91911	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	1976 Summer	19
253662	127018	Yelena Germanovna Vodorezova (-Buyanova)	F	12.0	161.0	47.0	Soviet Union	URS	1976 Winter	19
85840	43528	Antoinette Joyce Guedia Mouafo	F	12.0	170.0	55.0	Cameroon	CMR	2008 Summer	20
93850	47506	Hem Reaksmey	F	12.0	175.0	65.0	Cambodia	CAM	1996 Summer	19
226019	113580	Inge Srensen (-Tabur)	F	12.0	NaN	NaN	Denmark	DEN	1936 Summer	19
50294	25877	Olga Lucia de Angulo Irragorri	F	12.0	160.0	60.0	Colombia	COL	1968 Summer	19
50293	25877	Olga Lucia de Angulo Irragorri	F	12.0	160.0	60.0	Colombia	COL	1968 Summer	19
50291	25877	Olga Lucia de Angulo Irragorri	F	12.0	160.0	60.0	Colombia	COL	1968 Summer	19
79352	40296	Alain C. Giletti	M	12.0	NaN	NaN	France	FRA	1952 Winter	19
168941	84913	Clstine N'Drin	F	12.0	169.0	56.0	Cote d'Ivoire	CIV	1976 Summer	19

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Ye
168942	84913	Clstine N'Drin	F	12.0	169.0	56.0	Cote d'Ivoire	CIV	1976 Summer	19
251495	125944	Ines Vercesi	F	12.0	NaN	NaN	Italy	ITA	1928 Summer	19
96665	48939	Ho Gang	M	12.0	NaN	NaN	North Korea	PRK	1988 Winter	19
197895	99362	Antonia Real Horrach	F	12.0	160.0	50.0	Spain	ESP	1976 Summer	19
60911	31203	Patricia Anne "Pat" Eastwood	F	12.0	NaN	NaN	South Africa	RSA	1960 Winter	19
149033	74755	Luciana Marcellini Hercolani Gaddi	F	12.0	163.0	58.0	Italy	ITA	1960 Summer	19
108031	54620	Belita Gladys Lyne Jepson Turner (-Riordan, -K...	F	12.0	166.0	NaN	Great Britain	GBR	1936 Winter	19
145150	72854	Licia Macchini	F	12.0	NaN	NaN	Italy	ITA	1948 Summer	19
197894	99362	Antonia Real Horrach	F	12.0	160.0	50.0	Spain	ESP	1976 Summer	19
249804	125092	tienne Nol Henri Vandernotte	M	12.0	NaN	37.0	France	FRA	1936 Summer	19
249803	125092	tienne Nol Henri Vandernotte	M	12.0	NaN	37.0	France	FRA	1936 Summer	19
84361	42835	Werner Grieshofer	M	12.0	170.0	48.0	Austria	AUT	1972 Summer	19

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Ye
96236	48728	Margery Hinton	F	12.0	168.0	NaN	Great Britain	GBR	1928 Summer	19
148942	74712	Carla Marangoni	F	12.0	NaN	NaN	Italy	ITA	1928 Summer	19
91912	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	1976 Summer	19
50292	25877	Olga Lucia de Angulo Irragorri	F	12.0	160.0	60.0	Colombia	COL	1968 Summer	19
46955	24191	Philippe Cuelenaere	M	12.0	170.0	50.0	Belgium	BEL	1984 Summer	19
192507	96664	Dorothy Poynton-Hill (-Teuber)	F	12.0	NaN	NaN	United States	USA	1928 Summer	19
118048	59727	Marika Kilius (-Zahn, -Schfer)	F	12.0	168.0	51.0	Germany	GER	1956 Winter	19
236977	118832	Chin Say "Molly" Tay	F	12.0	145.0	41.0	Malaysia	MAS	1964 Summer	19
126542	63816	Carolyn Patricia Krau	F	12.0	137.0	36.0	Great Britain-2	GBR	1956 Winter	19
9650	5291	Marcia Arriaga Larrinua	F	12.0	168.0	58.0	Mexico	MEX	1968 Summer	19
97086	49142	Jan Hoffmann	M	12.0	178.0	73.0	East Germany	GDR	1968 Winter	19
91913	46578	Diana Hatler	F	12.0	164.0	46.0	Puerto Rico	PUR	1976 Summer	19

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Ye
9649	5291	Marcia Arriaga Larrinua	F	12.0	168.0	58.0	Mexico	MEX	1968 Summer	19
9648	5291	Marcia Arriaga Larrinua	F	12.0	168.0	58.0	Mexico	MEX	1968 Summer	19
120233	60854	Judit Kiss	F	12.0	171.0	57.0	Hungary	HUN	1992 Summer	19

In [18]:
```
# Calculate average age for each athlete for each event
avg_age = df_json1.groupby("Event")["Age"].mean().reset_index(name="Average Age"
#round the avg to zero decimal places
avg_age["Average Age"] = avg_age["Average Age"].round(0)

print("\nAverage Age of Athletes for Each Event:")
avg_age
```

```
Average Age of Athletes for Each Event:
```

Out[18]:

	Event	Average Age
0	Aeronautics Mixed Aeronautics	26.0
1	Alpine Skiing Men's Combined	24.0
2	Alpine Skiing Men's Downhill	24.0
3	Alpine Skiing Men's Giant Slalom	23.0
4	Alpine Skiing Men's Slalom	24.0
...	...	...
760	Wrestling Women's Flyweight, Freestyle	25.0
761	Wrestling Women's Heavyweight, Freestyle	26.0
762	Wrestling Women's Light-Heavyweight, Freestyle	25.0
763	Wrestling Women's Lightweight, Freestyle	25.0
764	Wrestling Women's Middleweight, Freestyle	25.0

765 rows × 2 columns

In [19]:
```
#count gold, silver and bronze medals, country wise
medal_summary = (
    #to summarize data in tabular format
    df_json1.pivot_table(
        index="Team", #unique value to identify
```

```
        columns="Medal",
        aggfunc="size",
        fill_value=0  #Replace missing values with 0
    )
    .reset_index() #resetting the Team column from index to regular column
)

# Rename columns for clarity
medal_summary.columns.name = None  #Remove the pivot column name
medal_summary.rename(columns={"Gold": "Gold", "Silver": "Silver", "Bronze": "Brc

print("\nMedal Summary by Country:")
print(medal_summary)
```

```
Medal Summary by Country:
                         Team  Bronze  Gold  Silver
0         A North American Team       4     0       0
1                   Afghanistan       2     0       0
2                       Algeria       8     5       4
3                   Ali-Baba II       5     0       0
4    Amateur Athletic Association       0     5       0
..                          ...     ...   ...     ...
493          Winnipeg Shamrocks-1       0    12       0
494                    Yugoslavia      93   130     167
495                        Zambia       1     0       1
496                      Zimbabwe       1    17       4
497                           Zut       0     0       3

[498 rows x 4 columns]
```

In [20]:

```
#count gold medals won by each country and display top 10
gold_medals = df_json1[df_json1['Medal']=='Gold']
gold_medal_count = gold_medals['Team'].value_counts()

top_ten_countries = gold_medal_count.head(10)
top_ten_countries
```

Out[20]:

```
Team
United States    2474
Soviet Union     1058
Germany           679
Italy             535
Great Britain     519
France            455
Sweden            451
Hungary           432
Canada            422
East Germany      369
Name: count, dtype: int64
```

This code and its associated insights focus on analyzing the distribution of gold medals across different countries:

Code Functionality:

- The code filters the dataset to identify records where the medal type is 'Gold'.
- A **count of gold medals** won by each country is calculated.
- The **top 10 countries** with the most gold medals are extracted and displayed.

Considerations:

- During the analysis, we encountered a historical nuance related to countries like the **Soviet Union**. The Soviet Union, which competed as a single entity during certain Olympic Games (e.g., 1980), has since been divided into multiple independent nations such as **Russia**, **Ukraine**, and others.
  - A key question is whether to treat the **Soviet Union and its successor states as separate entities** or aggregate their medal counts under one umbrella.
  - If treated separately, nations like Russia may appear to have fewer total medals compared to their historical performance as part of the Soviet Union.
  - Conversely, countries like the **USA**, which have remained consistent in their representation over the years, might seem more dominant due to the lack of such complexities.

Acknowledgment:

- This analysis acknowledges this issue but does not combine medal counts for countries that underwent geopolitical changes. For example, the Soviet Union and Russia are treated as distinct entities in this dataset. This approach avoids subjective decisions about how to merge historical records but may impact comparative dominance in the medal standings.
- These considerations are important to keep in mind when interpreting the **top 10 gold medal-winning countries**, as historical context plays a role in medal distribution.

Insights:

- The top-performing countries by gold medal count can be influenced by how historical entities like the Soviet Union are handled in the dataset. Further analysis or adjustments might be necessary for a more balanced comparison across nations.

In [21]:
```
#most medals won in each sport and by which player
medal_count = df_json1.groupby(['Sport', 'Name']).size().reset_index(name='Medal
#identify the row where medal count is high for a given sport
top_athlete_medals = medal_count.loc[medal_count.groupby('Sport')['Medal Count']
top_athlete_medals
```

Out[21]:

	Sport	Name	Medal Count
0	Aeronautics	Hermann Schreiber	1
1463	Alpine Skiing	Kjetil Andr Aamodt	20
2735	Alpinism	Antarge Sherpa	1
3067	Archery	Gerard Theodor Hubert Van Innis	11
4769	Art Competitions	Jean Lucien Nicolas Jacoby	9
...	...	...	...
123503	Tug-Of-War	Edwin Archer Mills	3
125643	Volleyball	Sergey Yuryevich Tetyukhin	6
127677	Water Polo	Manuel Estiarte Duocastella	6
129778	Weightlifting	Imre Fldi	5
136317	Wrestling	Wilfried Dietrich	8

66 rows × 3 columns

In [22]:

```
#Medal count of countires within specified year range

# Define the year range as variables
# feel free to experiment with years
start_year = 1920
end_year = 1940

# Filter data based on the year range
filtered_df = df_json1[(df_json1["Year"] >= start_year) & (df_json1["Year"] <= e

# Group by Country and count medals within the specified range
medal_count = (
    filtered_df.groupby("Team")["Medal"]
    .count() #count non-Nan values, returns total no. of medals won by each team
    .reset_index(name="Medal Count")
    .sort_values("Medal Count", ascending=False)
)

print(f"\nMedal Count from {start_year} to {end_year}:")
medal_count
```

```
Medal Count from 1920 to 1940:
```

Out[22]:

	Team	Medal Count
225	United States	770
82	France	376
126	Italy	354
91	Germany	350
210	Sweden	338
...	...	...
105	Hatuey	0
106	Heidelberg	0
109	Holland	0
111	Hungaria	0
237	Zimbabwe	0

238 rows × 2 columns

In [23]:

```
#Medal count of players along with country they represent within specified year

# Define the year range as variables
# feel free to experiment
start_year = 1980
end_year = 2000

# Filter data based on the year range
filtered_df = df_json1[(df_json1["Year"] >= start_year) & (df_json1["Year"] <= e

# Group by Country and count medals within the specified range
medal_count = (
    filtered_df.groupby(["Name","Team"])["Medal"]
    .count() #count non-Nan values, returns total no. of medals won by each team
    .reset_index(name="Medal Count")
    .sort_values("Medal Count", ascending=False)
)

print(f"\nMedal Count from {start_year} to {end_year}:")
medal_count
```

```
Medal Count from 1980 to 2000:
```

Out[23]:

	Name	Team	Medal Count
1406	Aleksey Yuryevich Nemov	Russia	12
28646	Matthew Nicholas "Matt" Biondi	United States	11
18940	Jennifer Elisabeth "Jenny" Thompson (-Cumpelik)	United States	10
12942	Frederick Carlton "Carl" Lewis	United States	10
28994	Merlene Joyce Ottey-Page	Jamaica	9
...	...	...	...
17224	Irina Olegovna Bazilevskaya	Belarus	0
17226	Irina Olegovna Shilova	Belarus	0
17228	Irina Olegovna Shilova	Unified Team	0
17229	Irina Petcule	Romania	0
46886	zdemir Akbal	Turkey	0

46887 rows × 3 columns

In [24]:
```
#Medal count based on sport and which country has won maximum medals

medal_count = df_json1.groupby(['Sport', 'Team']).size().reset_index(name='Medal
top_athlete_medals = medal_count.loc[medal_count.groupby('Sport')['Medal Count']
top_athlete_medals
```

Out[24]:

	Sport	Team	Medal Count
0	Aeronautics	Switzerland	1
7	Alpine Skiing	Austria	556
104	Alpinism	Great Britain	12
204	Archery	United States	155
256	Art Competitions	United States	314
...	...	...	...
4811	Tug-Of-War	Sweden	24
4817	Volleyball	Brazil	285
4881	Water Polo	Hungary	304
5061	Weightlifting	United States	153
5193	Wrestling	United States	383

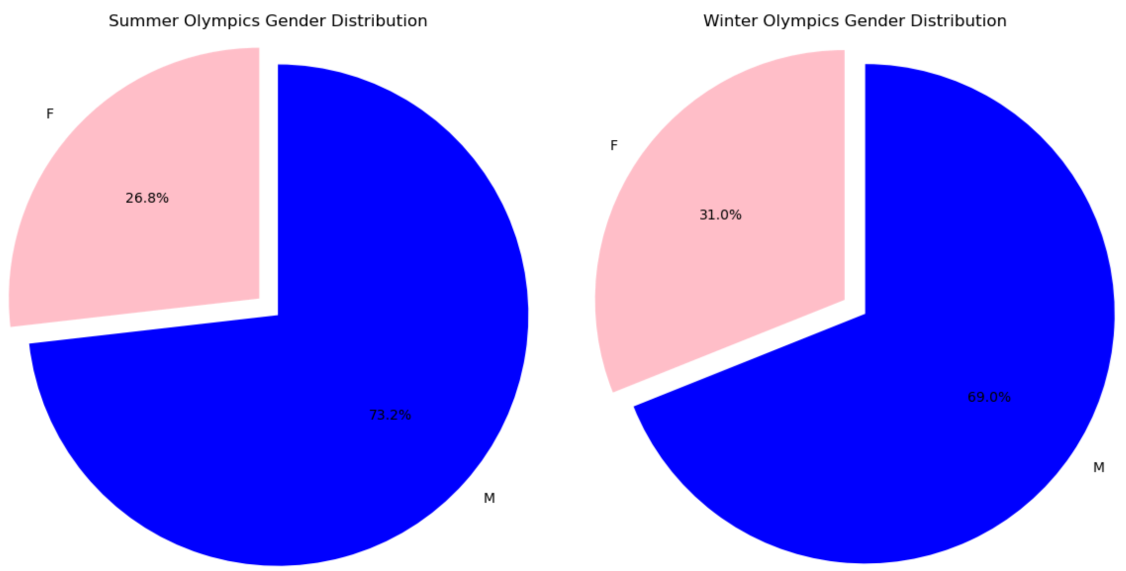
66 rows × 3 columns

In [25]:
```
print(df_json1['Season'].unique())
print(df_json1['Sex'].unique())
```

```
['Summer' 'Winter']
['M' 'F']
```

In [26]:

```
import pandas as pd
import matplotlib.pyplot as plt

gender_counts = df_json1.groupby(['Season', 'Sex']).size().reset_index(name='Cou

# Separate data for Summer and Winter
summer = gender_counts[gender_counts['Season'] == 'Summer']
winter = gender_counts[gender_counts['Season'] == 'Winter']

# Function to plot a pie chart
def plot_pie_chart(data, title):
    labels = data['Sex']
    sizes = data['Counts']
    colors = ['pink', 'blue']  # Adjust colors as needed
    explode = (0.1, 0)  # Highlight the first slice if desired

    plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, explode=expl
    plt.title(title)
    plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.

# Create subplots for side-by-side pie charts
plt.figure(figsize=(12, 6))

# Summer Olympics
plt.subplot(1, 2, 1)
plot_pie_chart(summer, 'Summer Olympics Gender Distribution')

# Winter Olympics
plt.subplot(1, 2, 2)
plot_pie_chart(winter, 'Winter Olympics Gender Distribution')

# Show the plot
plt.tight_layout()
plt.show()
```



This code and visualization provide a breakdown of gender distribution in the **Summer Olympics** and **Winter Olympics** using pie charts. Here's what it shows:

1. **Data Aggregation**:
   The dataset was grouped by **Season** (Summer or Winter) and **Sex** (Male or Female) to count the number of participants in each category. This grouped data was separated into two subsets: one for the Summer Olympics and another for the Winter Olympics.

2. **Pie Chart Visualization**:
   The **plot_pie_chart** function was used to create pie charts displaying the proportions of male and female participants for each Olympic season. The function highlights one segment (female) using an "explode" effect, making the charts visually distinct.

3. **Comparison of Gender Distribution**:

   - In the **Summer Olympics**, male participants accounted for **73.2%**, while female participants made up **26.8%**.
   - In the **Winter Olympics**, male participants represented **69.0%**, and females accounted for **31.0%**.

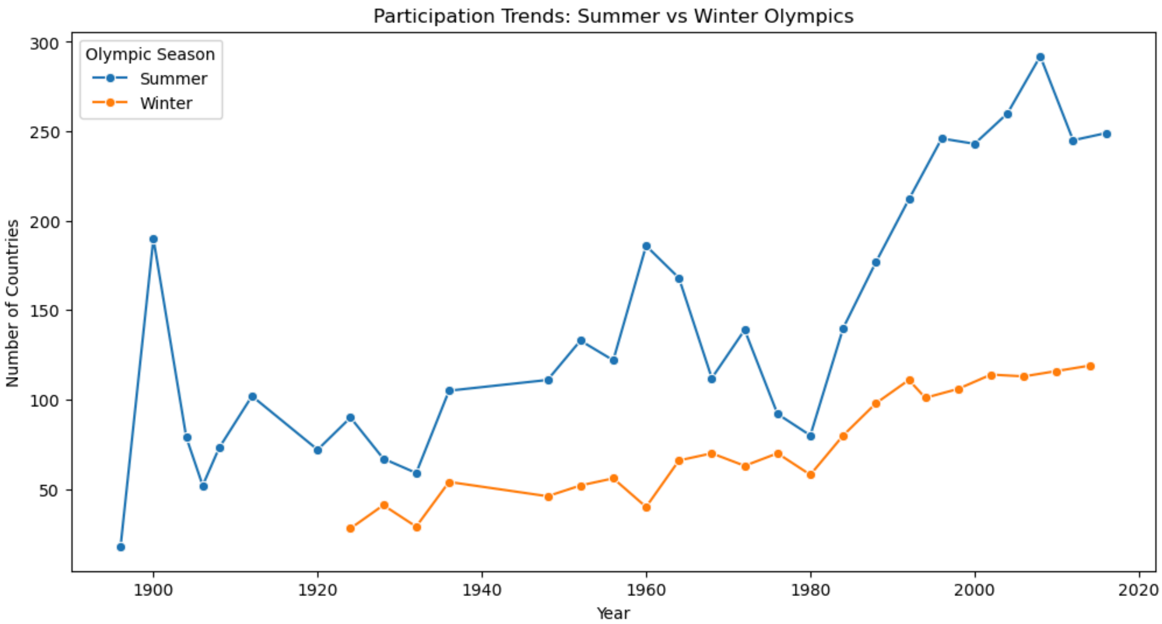
   The gender gap is slightly narrower in the Winter Olympics compared to the Summer Olympics.

4. **Layout**:
   The charts are displayed side-by-side for easy comparison, with clear titles indicating the season and labels showing the percentage contribution of each gender.

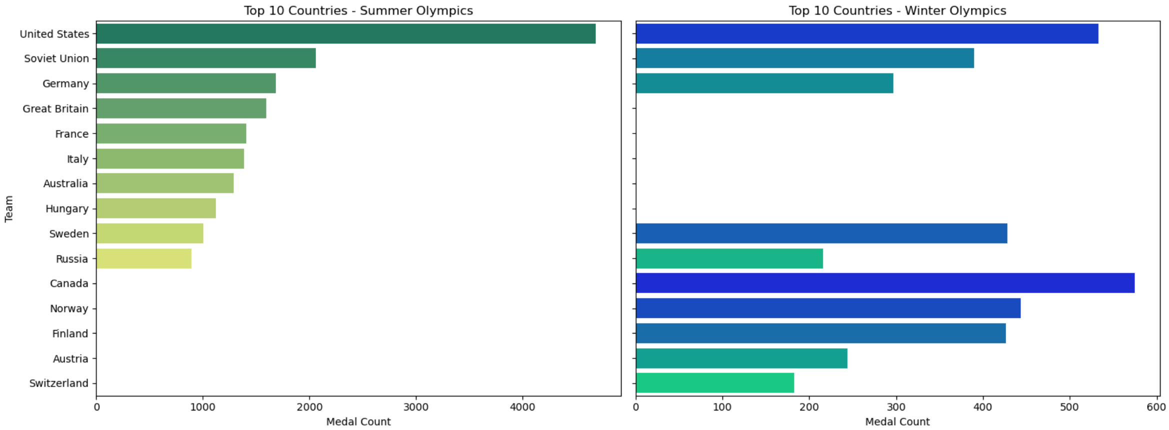
This visualization effectively highlights the gender distribution across the two Olympic seasons, making it easy to observe the differences in participation between males and females.

In [27]:
```
import seaborn as sns
import matplotlib.pyplot as plt


# Ensure necessary columns are present
required_columns = {'Season', 'Year', 'Team', 'Sport', 'Medal'}
if not required_columns.issubset(df_json1.columns):
    raise ValueError(f"Dataset must contain the following columns: {required_col

# Filter for Summer and Winter Olympics data
summer_df = df_json1[df_json1['Season'] == 'Summer']
winter_df = df_json1[df_json1['Season'] == 'Winter']

### 1. Participation Trends ###
# Group by Year and Season, then count the number of unique participants or coun
participation_trends = df_json1.groupby(['Year', 'Season'])['Team'].nunique().re

# Plot Participation Trends
plt.figure(figsize=(12, 6))
sns.lineplot(data=participation_trends, x='Year', y='Unique Countries', hue='Sea
plt.title('Participation Trends: Summer vs Winter Olympics')
plt.xlabel('Year')
plt.ylabel('Number of Countries')
plt.legend(title='Olympic Season')
```

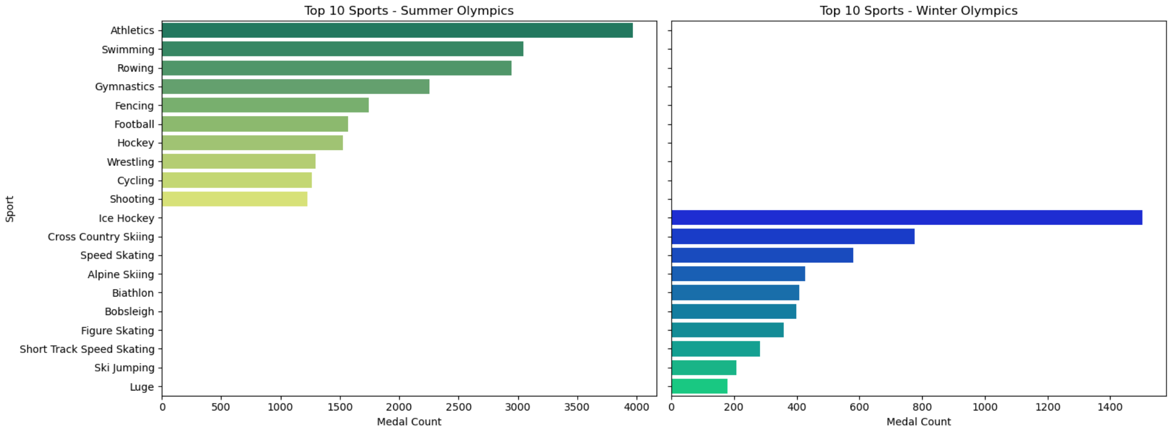
```
plt.show()

### 2. Medal Counts by Country ###
# Count medals by Country and Season
medals_by_country = df_json1[df_json1['Medal'].notnull()].groupby(['Team', 'Seas

# Get top 10 countries by medal count in each season
top_summer_countries = medals_by_country[medals_by_country['Season'] == 'Summer'
top_winter_countries = medals_by_country[medals_by_country['Season'] == 'Winter'

# Plot Top Countries Medal Counts
fig, axes = plt.subplots(1, 2, figsize=(16, 6), sharey=True)
sns.barplot(data=top_summer_countries, x='Medal Count', y='Team', ax=axes[0], pa
axes[0].set_title('Top 10 Countries - Summer Olympics')
sns.barplot(data=top_winter_countries, x='Medal Count', y='Team', ax=axes[1], pa
axes[1].set_title('Top 10 Countries - Winter Olympics')
plt.tight_layout()
plt.show()

### 3. Medal Counts by Sport ###
# Count medals by Sport and Season
medals_by_sport = df_json1[df_json1['Medal'].notnull()].groupby(['Sport', 'Seaso

# Get top sports by medal count in each season
top_summer_sports = medals_by_sport[medals_by_sport['Season'] == 'Summer'].nlarg
top_winter_sports = medals_by_sport[medals_by_sport['Season'] == 'Winter'].nlarg

# Plot Top Sports Medal Counts
fig, axes = plt.subplots(1, 2, figsize=(16, 6), sharey=True)
sns.barplot(data=top_summer_sports, x='Medal Count', y='Sport', ax=axes[0], pale
axes[0].set_title('Top 10 Sports - Summer Olympics')
sns.barplot(data=top_winter_sports, x='Medal Count', y='Sport', ax=axes[1], pale
axes[1].set_title('Top 10 Sports - Winter Olympics')
plt.tight_layout()
plt.show()
```



```
C:\Users\Riddhi Shelwante\AppData\Local\Temp\ipykernel_28552\2388618874.py:37: Fu
tureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=top_summer_countries, x='Medal Count', y='Team', ax=axes[0], p
alette='summer')
C:\Users\Riddhi Shelwante\AppData\Local\Temp\ipykernel_28552\2388618874.py:39: Fu
tureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=top_winter_countries, x='Medal Count', y='Team', ax=axes[1], p
alette='winter')
```




```
C:\Users\Riddhi Shelwante\AppData\Local\Temp\ipykernel_28552\2388618874.py:54: Fu
tureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=top_summer_sports, x='Medal Count', y='Sport', ax=axes[0], pal
ette='summer')
C:\Users\Riddhi Shelwante\AppData\Local\Temp\ipykernel_28552\2388618874.py:56: Fu
tureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=top_winter_sports, x='Medal Count', y='Sport', ax=axes[1], pal
ette='winter')
```



This code and visualization analyze participation and medal trends in the Summer and Winter Olympics across several dimensions:

1. Participation Trends:

- The number of unique countries participating in each Olympic season over the years was calculated.
- A **line plot** was created to compare participation trends for the Summer and Winter Olympics.
  - **Insights**: The Summer Olympics consistently have higher participation, but both seasons show a growing trend in the number of participating countries over time.

2. Top Medal-Winning Countries:

- Medal counts for each country were calculated and grouped by season.
- The **top 10 medal-winning countries** for both the Summer and Winter Olympics were identified.
- Two **bar charts** were created to visualize these rankings:
  - The **left chart** shows the top-performing countries in the Summer Olympics.
  - The **right chart** shows the top-performing countries in the Winter Olympics.
  - **Insights**: These visualizations highlight which nations dominate in terms of medal achievements in both seasons.

3. Top Medal-Winning Sports:

- Medal counts for each sport were calculated and grouped by season.
- The **top 10 sports with the highest medal counts** for both the Summer and Winter Olympics were identified.
- Two **bar charts** were generated:
  - The **left chart** displays the top sports in the Summer Olympics.
  - The **right chart** displays the top sports in the Winter Olympics.
  - **Insights**: These plots reveal the most competitive sports in terms of medal distribution for both seasons.

Overall, this analysis provides an in-depth look into Olympic participation and performance trends, offering valuable insights into how different countries and sports have performed across both Summer and Winter Olympic games.

Conclusion:

The Olympics dataset offa of insi,ghts once cleaned and analysed. By resolving missing values and addressing historical nuances, we uncovered meaningful trends in participation, medal achievements, and gender representation. The analysis revealed that while the Summer Olympics consistently attract more participants, the Winter Olympics are closing the gender gap slightly faster. Dominant medal performances from certain nations and sports also stood out, with geopolitical shifts adding layers of complexity to the data. Overall, this study emphasizes the importance of thorough data cleaning and thoughtful analysis to tell a meaningful story about the world's most celebrated sporting evnt.

Team Members:

1. Riya Shelwante 2502398
2. Vaishnavi Jadhav 2500370
3. Gabriel Rajan 2504957
4. Lemar Vega 2501567