PROJECT TITLE : Olympics Data Analysis (Sports Domain)

Libraries Import

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Display options

In [3]:
```
pd.set_option('display.max_columns', 50)
pd.set_option('display.max_rows', 100)
%matplotlib inline
plt.rcParams['figure.figsize'] = (10, 6)
```

Load dataset

In [9]:
```
df=pd.read_csv("C:\\Users\\hp5cd\\Downloads\\python study resocures\\olympics_datas
df
```

Out[9]:

	player_id	Name	Sex	Team	NOC	Year	Season	City	Sport	
0	0	A Dijiang	M	China	CHN	1992	Summer	Barcelona	Basketball	Basl Basl
1	1	A Lamusi	M	China	CHN	2012	Summer	London	Judo	Judo Lightv
2	2	Gunnar Aaby	M	Denmark	DEN	1920	Summer	Antwerpen	Football	Fc Fc
3	3	Edgar Aabye	M	Denmark/ Sweden	DEN	1900	Summer	Paris	Tug-Of-War	Tu War Tu
4	26	Cornelia (-strannood)	F	Netherlands	NED	1932	Summer	Los Angeles	Athletics	At Wc 100 r
...	...	...	...	...	...	...	...	...	...	
252560	4986655	Sefora Ada	F	Equatorial Guinea	GEQ	2024	Summer	Paris	Athletics	Wc
252561	9460001	Emanuela Liuzzi	F	Italy	ITA	2024	Summer	Paris	Wrestling	Wc Fre
252562	1972077	Isayah Boers	M	Netherlands	NED	2024	Summer	Paris	Athletics	4 x
252563	1899865	Kevin Staut	M	France	FRA	2024	Summer	Paris	Equestrian	Ju
252564	1924402	Charlie Carvell	M	Great Britain	GBR	2024	Summer	Paris	Athletics	Me 400m

252565 rows × 11 columns

first 5 rows

In [10]:
```
df.head()
```

Out[10]:

	player_id	Name	Sex	Team	NOC	Year	Season	City	Sport	Event
0	0	A Dijiang	M	China	CHN	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball
1	1	A Lamusi	M	China	CHN	2012	Summer	London	Judo	Judo Men's Extra-Lightweight
2	2	Gunnar Aaby	M	Denmark	DEN	1920	Summer	Antwerpen	Football	Football Men's Football
3	3	Edgar Aabye	M	Denmark/ Sweden	DEN	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War
4	26	Cornelia (-strannood)	F	Netherlands	NED	1932	Summer	Los Angeles	Athletics	Athletics Women's 100 metres

Last 5 rows

In [11]:
```
df.tail()
```

Out[11]:

	player_id	Name	Sex	Team	NOC	Year	Season	City	Sport	Event	M
252560	4986655	Sefora Ada	F	Equatorial Guinea	GEQ	2024	Summer	Paris	Athletics	Women's 100m	r
252561	9460001	Emanuela Liuzzi	F	Italy	ITA	2024	Summer	Paris	Wrestling	Women's Freestyle 50kg	r
252562	1972077	Isayah Boers	M	Netherlands	NED	2024	Summer	Paris	Athletics	4 x 400m Relay Mixed	
252563	1899865	Kevin Staut	M	France	FRA	2024	Summer	Paris	Equestrian	Jumping Team	B
252564	1924402	Charlie Carvell	M	Great Britain	GBR	2024	Summer	Paris	Athletics	Men's 4 x 400m Relay	B

INFORMATION

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252565 entries, 0 to 252564
Data columns (total 11 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   player_id  252565 non-null  int64
 1   Name       252565 non-null  object
 2   Sex        252565 non-null  object
 3   Team       252565 non-null  object
 4   NOC        252565 non-null  object
 5   Year       252565 non-null  int64
 6   Season     252565 non-null  object
 7   City       252565 non-null  object
 8   Sport      252565 non-null  object
 9   Event      252565 non-null  object
 10  Medal      252565 non-null  object
dtypes: int64(2), object(9)
memory usage: 21.2+ MB
```

Describe

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

Out[13]:

	player_id	Year
count	2.525650e+05	252565.000000
mean	2.305499e+05	1981.743908
std	4.289330e+05	32.596548
min	0.000000e+00	1896.000000
25%	5.713700e+04	1960.000000
50%	1.356110e+05	1988.000000
75%	2.118590e+05	2008.000000
max	9.460001e+06	2024.000000

Check missing values and duplicates

In [14]:
```
missing = df.isnull().sum()
duplicates = df.duplicated().sum()
print("Missing values per column:\n", missing)
print("\nTotal duplicate rows:", duplicates)
```

```
Missing values per column:
 player_id    0
Name          0
Sex           0
Team          0
NOC           0
Year          0
Season        0
City          0
Sport         0
Event         0
Medal         0
dtype: int64

Total duplicate rows: 0
```

Data type conversions and basic cleaning

In [15]:
```
# Standardize Medal column: replace empty or 'No medal' with NaN for analysis of me
df['Medal'] = df['Medal'].replace({'No medal': np.nan, '': np.nan})
# Ensure Year is integer
df['Year'] = pd.to_numeric(df['Year'], errors='coerce').astype('Int64')
# Sex as category
df['Sex'] = df['Sex'].astype('category')
# Create medal_won boolean
df['medal_won'] = ~df['Medal'].isna()
# Drop exact duplicate rows (if any)
df = df.drop_duplicates()
print("After cleaning shape:", df.shape)
```

```
After cleaning shape: (252565, 12)
```

cleaned sample

In [16]:
```
df.info()
df[['Name','Sex','Team','NOC','Year','Sport','Event','Medal']].head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252565 entries, 0 to 252564
Data columns (total 12 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   player_id  252565 non-null  int64
 1   Name       252565 non-null  object
 2   Sex        252565 non-null  category
 3   Team       252565 non-null  object
 4   NOC        252565 non-null  object
 5   Year       252565 non-null  Int64
 6   Season     252565 non-null  object
 7   City       252565 non-null  object
 8   Sport      252565 non-null  object
 9   Event      252565 non-null  object
 10  Medal      38818 non-null   object
 11  medal_won  252565 non-null  bool
dtypes: Int64(1), bool(1), category(1), int64(1), object(8)
memory usage: 20.0+ MB
```

Out[16]:

	Name	Sex	Team	NOC	Year	Sport	Event	Medal
0	A Dijiang	M	China	CHN	1992	Basketball	Basketball Men's Basketball	NaN
1	A Lamusi	M	China	CHN	2012	Judo	Judo Men's Extra-Lightweight	NaN
2	Gunnar Aaby	M	Denmark	DEN	1920	Football	Football Men's Football	NaN
3	Edgar Aabye	M	Denmark/ Sweden	DEN	1900	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	Cornelia (-strannood)	F	Netherlands	NED	1932	Athletics	Athletics Women's 100 metres	NaN

Exploratory Data Analysis (EDA)

In [17]:
```python
#  Overall descriptive statistics

total_participants = len(df)
total_medalists = df['medal_won'].sum()
unique_countries = df['Team'].nunique()
unique_sports = df['Sport'].nunique()

print(f"Total records: {total_participants}")
print(f"Total medal records: {total_medalists}")
print(f"Unique teams/countries: {unique_countries}")
print(f"Unique sports: {unique_sports}")
```

```
Total records: 252565
Total medal records: 38818
Unique teams/countries: 1193
Unique sports: 76
```

Medal counts by type

In [18]:
```
medal_counts = df['Medal'].value_counts(dropna=True)
medal_counts
```

Out[18]:
```
Medal
Bronze    13070
Gold      13002
Silver    12746
Name: count, dtype: int64
```

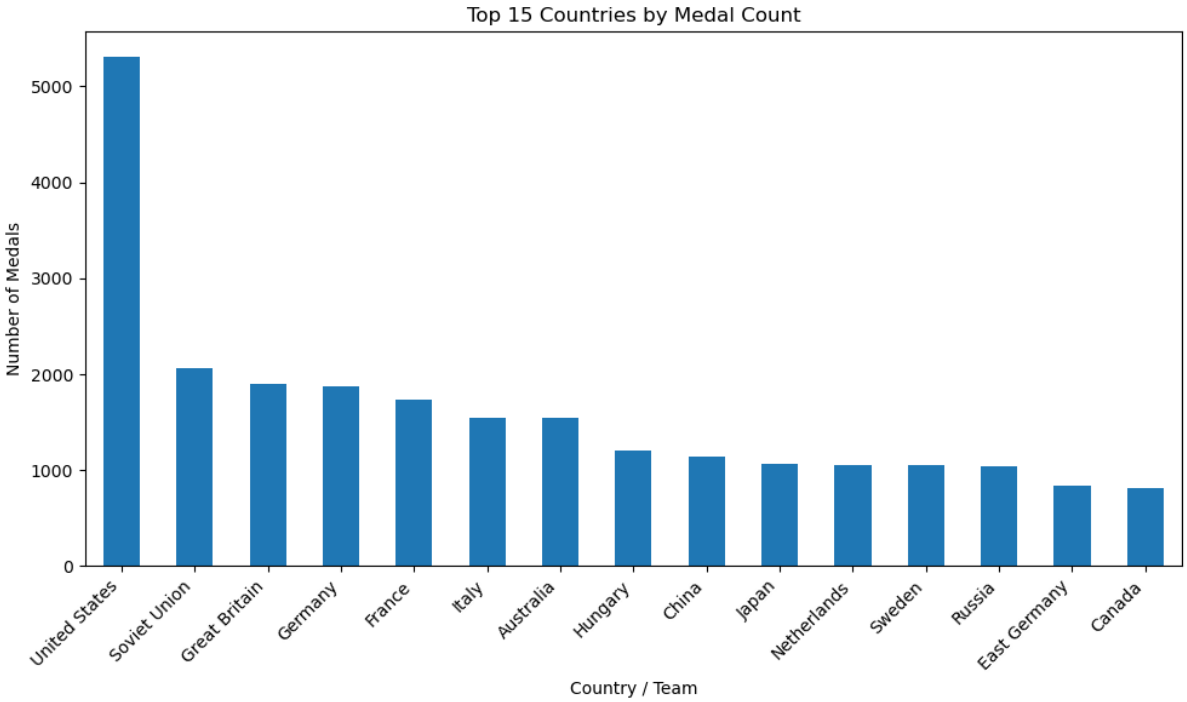
Top 15 countries by medal count (Team)

In [19]:
```
top_countries = (df[df['medal_won']]
                 .groupby('Team')['medal_won']
                 .sum()
                 .sort_values(ascending=False)
                 .head(15))
top_countries
```

Out[19]:
```
Team
United States    5305
Soviet Union     2061
Great Britain    1895
Germany          1876
France           1736
Italy            1542
Australia        1542
Hungary          1204
China            1140
Japan            1061
Netherlands      1056
Sweden           1056
Russia           1043
East Germany      841
Canada            810
Name: medal_won, dtype: int64
```

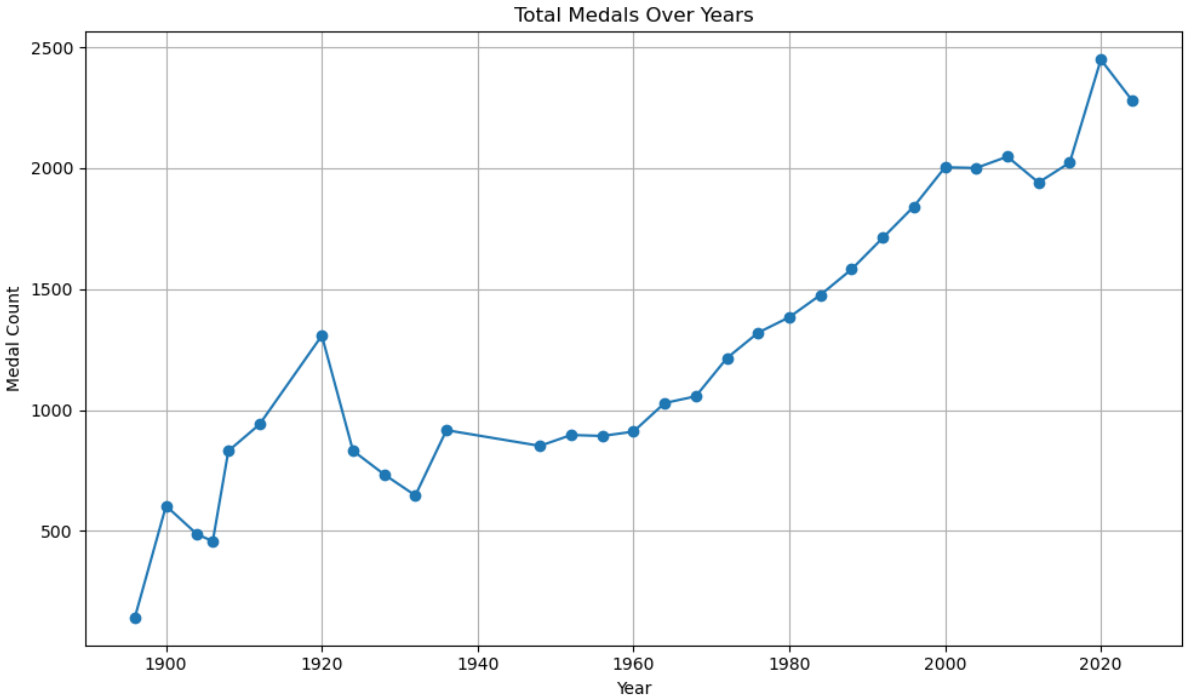
Plot top countries

In [20]:
```
top_countries.plot(kind='bar')
plt.title("Top 15 Countries by Medal Count")
plt.ylabel("Number of Medals")
plt.xlabel("Country / Team")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Medals over years (trend)

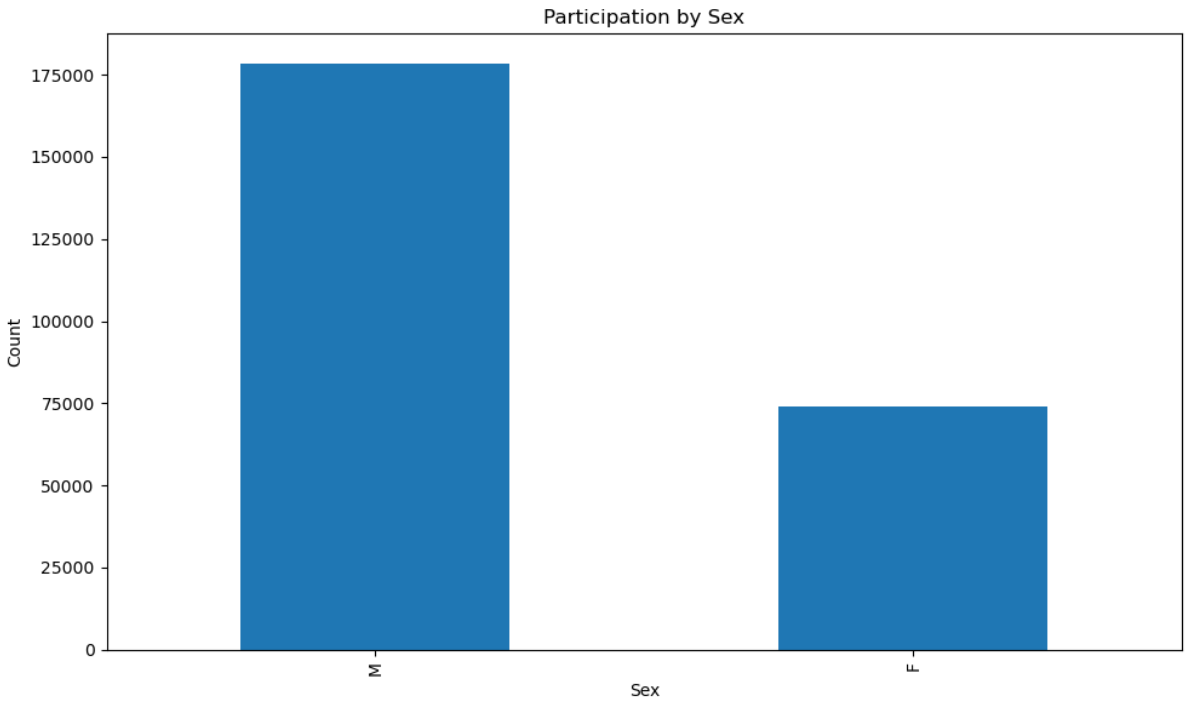
In [21]:
```
medals_by_year = (df[df['medal_won']]
                  .groupby('Year')['medal_won']
                  .sum()
                  .sort_index())
medals_by_year.plot(marker='o')
plt.title("Total Medals Over Years")
plt.ylabel("Medal Count")
plt.xlabel("Year")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Gender distribution (overall participation and medal share)

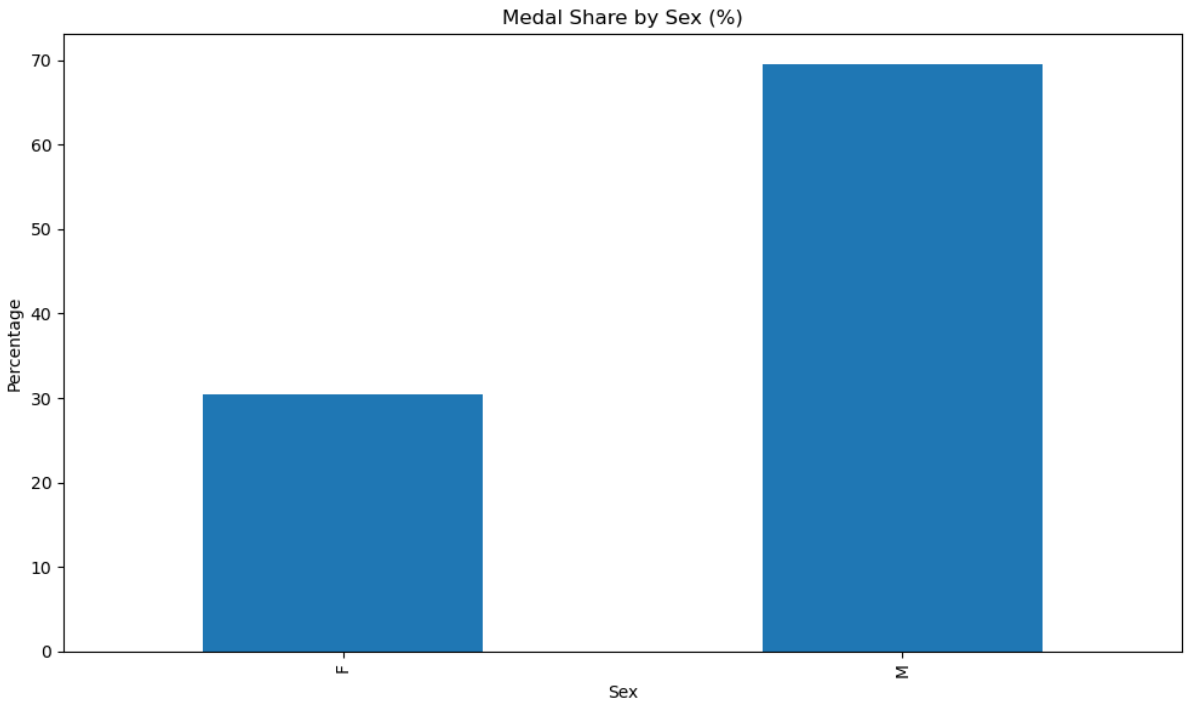
In [22]:
```
participation_by_sex = df['Sex'].value_counts()
medals_by_sex = df[df['medal_won']].groupby('Sex')['medal_won'].sum()
print("Participation by Sex:\n", participation_by_sex)
print("\nMedals by Sex:\n", medals_by_sex)

# Plot participation
participation_by_sex.plot(kind='bar')
plt.title("Participation by Sex")
plt.xlabel("Sex")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

```
Participation by Sex:
 Sex
M    178544
F     74021
Name: count, dtype: int64

Medals by Sex:
 Sex
F    11797
M    27021
Name: medal_won, dtype: int64
```



Medal share by sex (percentage)

In [23]:
```
medal_share_sex = (medals_by_sex / medals_by_sex.sum()) * 100
medal_share_sex.plot(kind='bar')
plt.title("Medal Share by Sex (%)")
plt.ylabel("Percentage")
plt.xlabel("Sex")
plt.tight_layout()
plt.show()
```



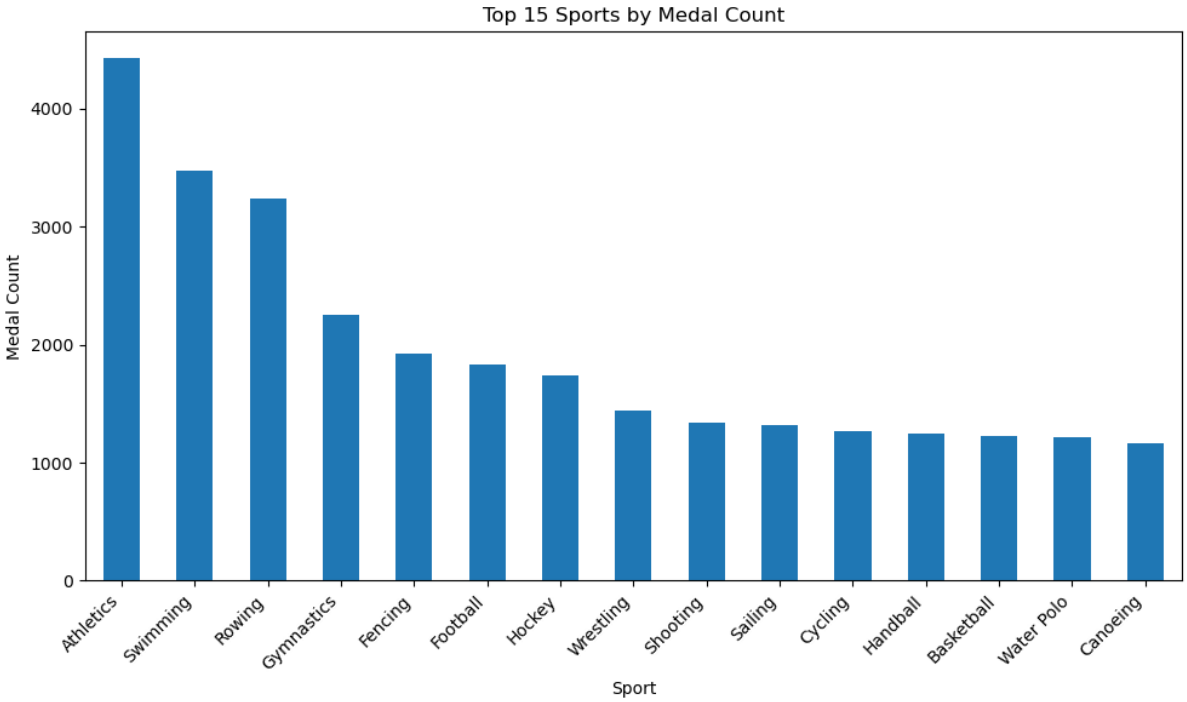
Top sports by medal count

In [24]:
```
top_sports = (df[df['medal_won']]
              .groupby('Sport')['medal_won']
              .sum()
              .sort_values(ascending=False)
              .head(15))
top_sports
```

Out[24]:
```
Sport
Athletics     4429
Swimming      3470
Rowing        3233
Gymnastics    2256
Fencing       1923
Football      1827
Hockey        1738
Wrestling     1440
Shooting      1336
Sailing       1319
Cycling       1263
Handball      1247
Basketball    1223
Water Polo    1213
Canoeing      1165
Name: medal_won, dtype: int64
```

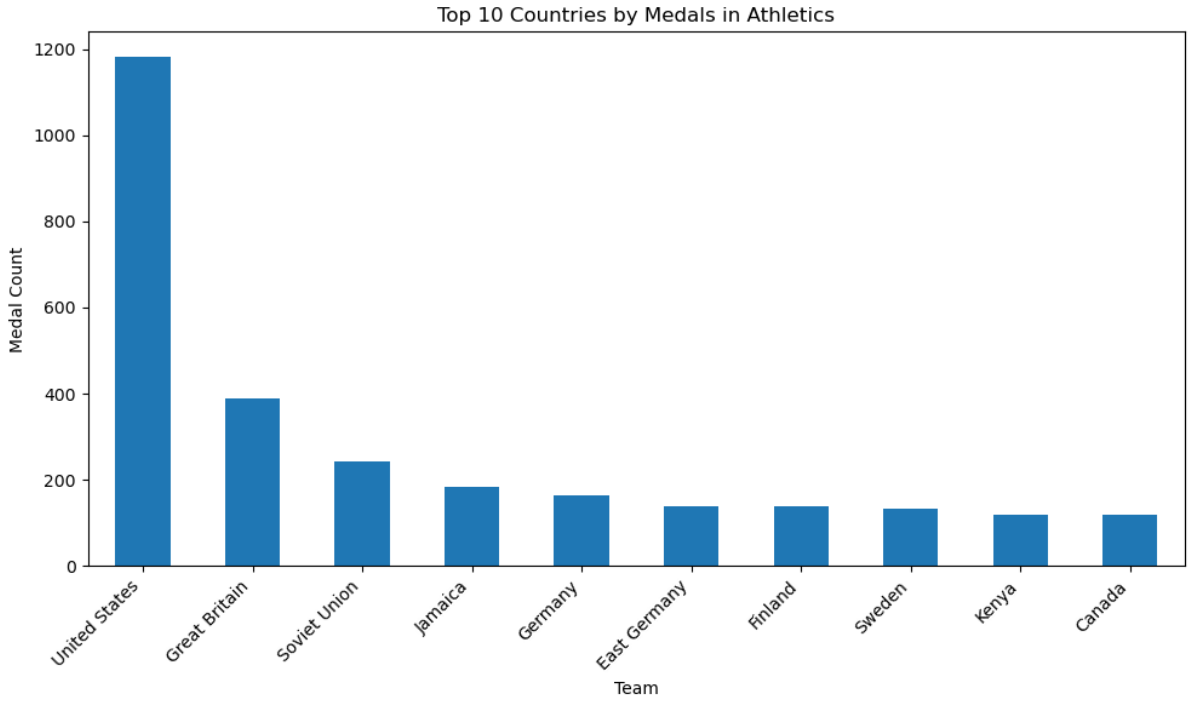
Plot top sports

In [25]:
```
top_sports.plot(kind='bar')
plt.title("Top 15 Sports by Medal Count")
plt.xlabel("Sport")
plt.ylabel("Medal Count")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Country dominance in a single sport (example: top country in Athletics)

In [26]:
```
sport = 'Athletics'
if sport in df['Sport'].unique():
    sport_country = (df[(df['Sport']==sport) & (df['medal_won'])]
                     .groupby('Team')['medal_won'].sum()
                     .sort_values(ascending=False).head(10))
    sport_country.plot(kind='bar')
    plt.title(f"Top 10 Countries by Medals in {sport}")
    plt.xlabel("Team")
    plt.ylabel("Medal Count")
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
else:
    print(f"Sport '{sport}' not present in dataset.")
```



Save cleaned dataset

In [28]:
```
cleaned_path = "C:\\Users\\hp5cd\\Downloads\\python study resocures\\olympics_datas
df.to_csv(cleaned_path, index=False)
print("Cleaned dataset saved to:", cleaned_path)
```

```
Cleaned dataset saved to: C:\Users\hp5cd\Downloads\python study resocures\olympics_
dataset.csv
```

Appendix sample: function to compute medal counts per NOC and year (pivot)

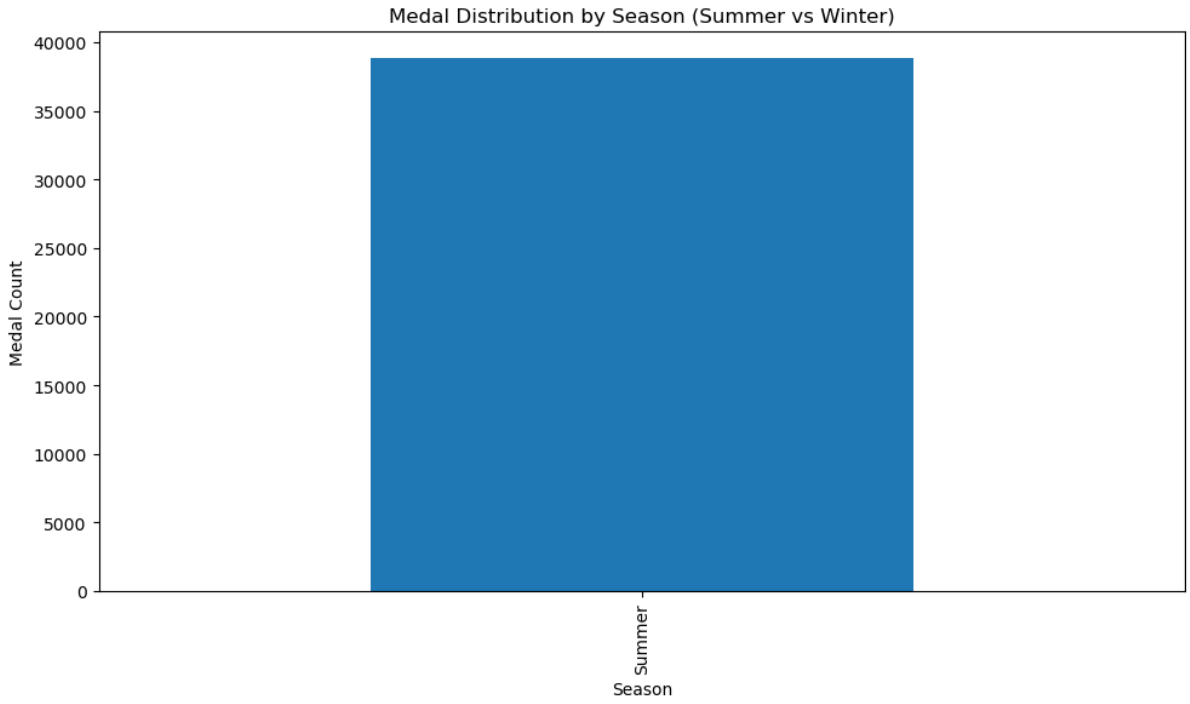
In [29]:
```
def medals_pivot(df):
    pivot = (df[df['medal_won']]
             .pivot_table(index='Year', columns='Team', values='medal_won', aggfunc
    return pivot

pivot_sample = medals_pivot(df)
pivot_sample.head()
```

Out[29]:

Team	A North American Team	AIN	Afghanistan	Albania	Algeria	Ali-Baba II	Amateur Athletic Association	Amstel Amsterdam	Ancora	Ang
Year										
1896	0	0	0	0	0	0	0	0	0	
1900	4	0	0	0	0	0	5	0	0	
1904	0	0	0	0	0	0	0	0	0	
1906	0	0	0	0	0	0	0	0	0	
1908	0	0	0	0	0	0	0	4	0	

5 rows × 499 columns

In [30]:
```
season_medals = df[df['medal_won']].groupby('Season')['medal_won'].sum()

season_medals.plot(kind='bar')
plt.title("Medal Distribution by Season (Summer vs Winter)")
plt.xlabel("Season")
plt.ylabel("Medal Count")
plt.tight_layout()
plt.show()
```



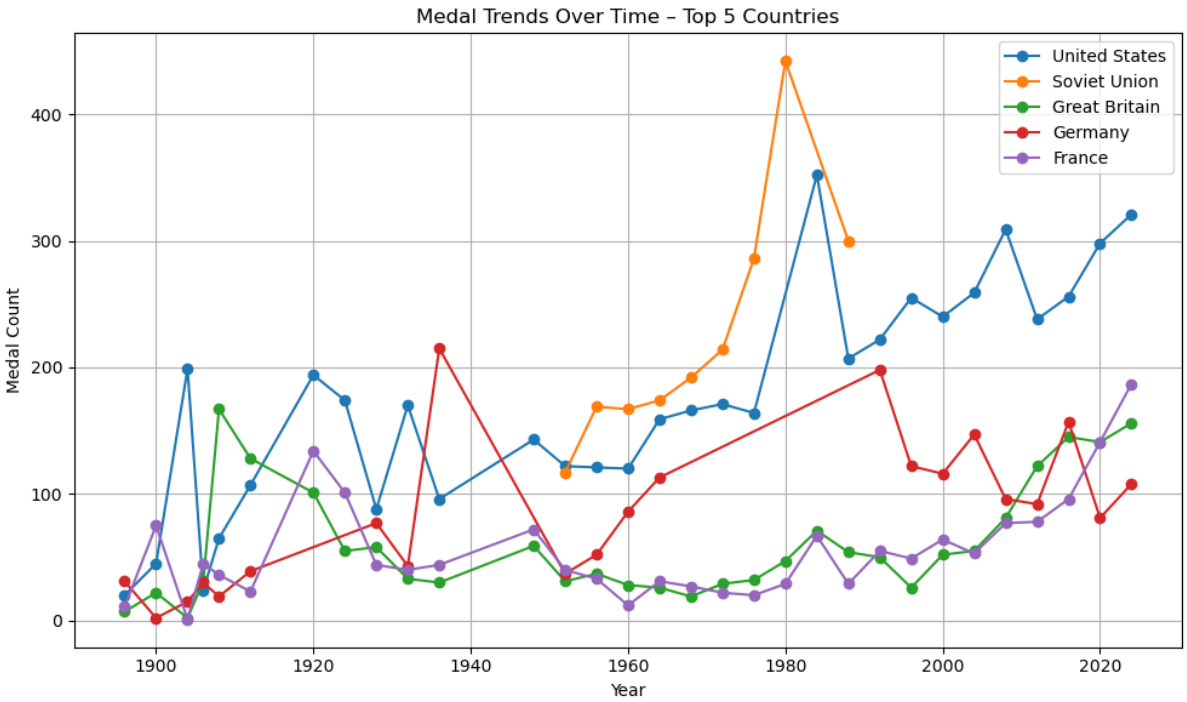
Age Distribution of Medal Winners (Boxplot)

In [32]:
```
if 'Age' in df.columns:
    df_age = df[df['medal_won'] & df['Age'].notna()]

    plt.boxplot(df_age['Age'])
    plt.title("Age Distribution of Medal Winners")
    plt.ylabel("Age")
    plt.tight_layout()
    plt.show()
else:
    print("Age column not found in dataset.")
```

```
Age column not found in dataset.
```

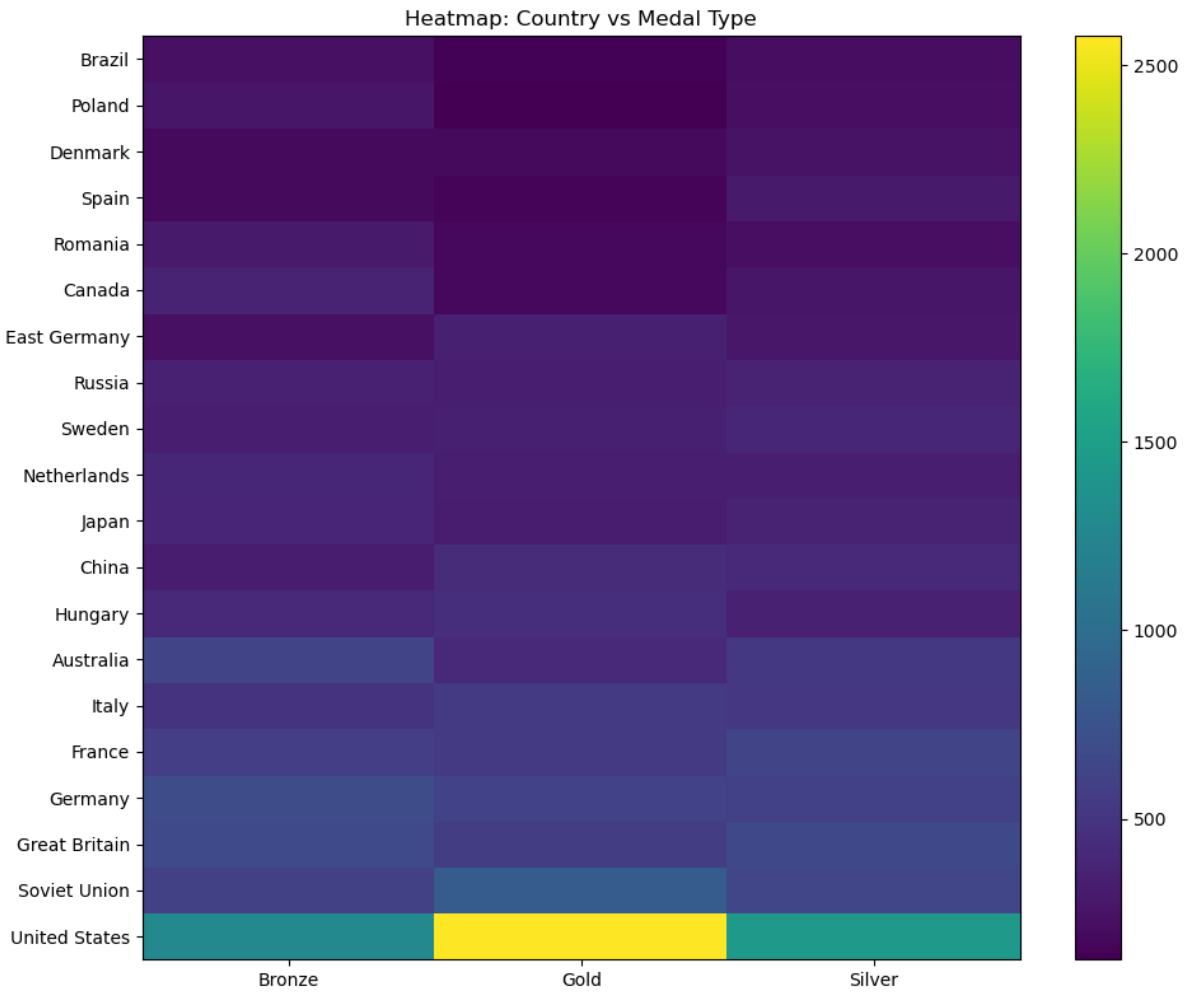
Medal Trend Over Time for Top 5 Countries

In [33]:
```
top5 = (df[df['medal_won']]
        .groupby('Team')['medal_won']
        .sum()
        .sort_values(ascending=False)
        .head(5)
        .index)

for team in top5:
    yearly = df[(df['Team']==team) & (df['medal_won'])].groupby('Year')['medal_won'
    plt.plot(yearly.index, yearly.values, marker='o', label=team)

plt.title("Medal Trends Over Time - Top 5 Countries")
plt.xlabel("Year")
plt.ylabel("Medal Count")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



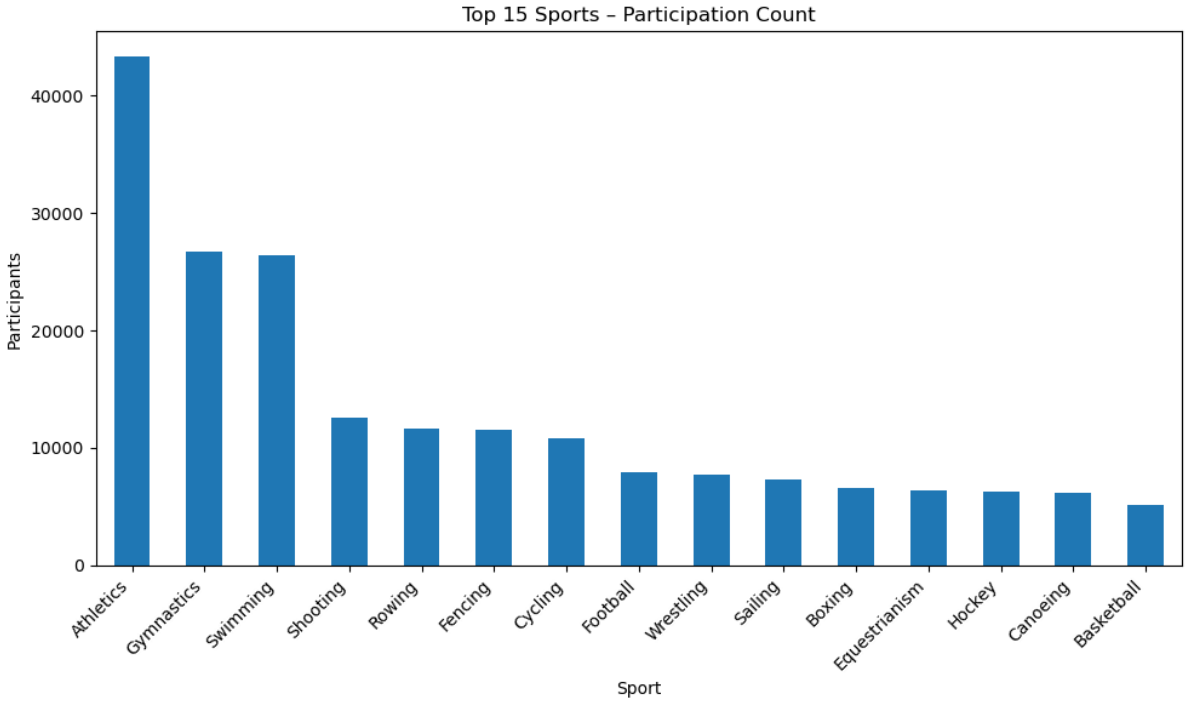
Heatmap: Country vs Medal Type (Gold/ Silver/Bronze)

In [34]:
```
pivot = (df[df['medal_won']]
         .pivot_table(index='Team', columns='Medal', values='medal_won',
                      aggfunc='sum', fill_value=0))

top20 = pivot.sum(axis=1).sort_values(ascending=False).head(20)
heat_data = pivot.loc[top20.index]

plt.figure(figsize=(10,8))
plt.pcolor(heat_data)
plt.xticks(np.arange(len(heat_data.columns))+0.5, heat_data.columns)
plt.yticks(np.arange(len(heat_data.index))+0.5, heat_data.index)
plt.title("Heatmap: Country vs Medal Type")
plt.colorbar()
plt.tight_layout()
plt.show()
```



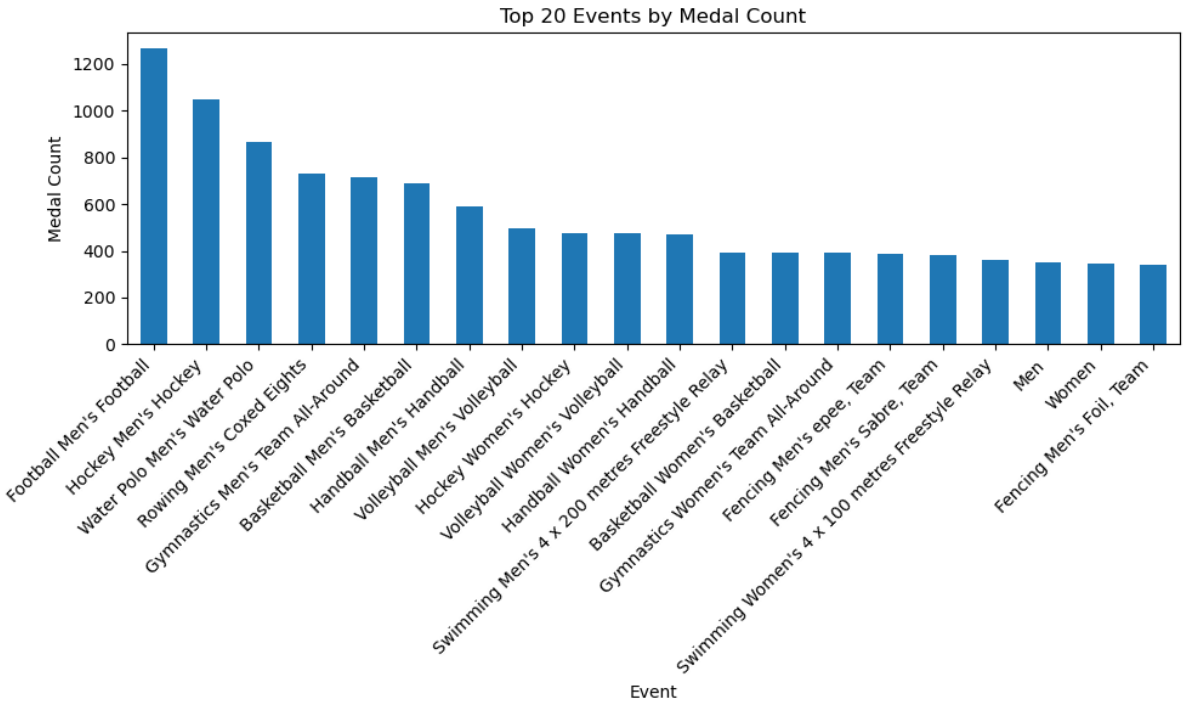
Sport-wise Participation Count

In [35]:
```
participation_sport = df['Sport'].value_counts().head(15)

participation_sport.plot(kind='bar')
plt.title("Top 15 Sports – Participation Count")
plt.xlabel("Sport")
plt.ylabel("Participants")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



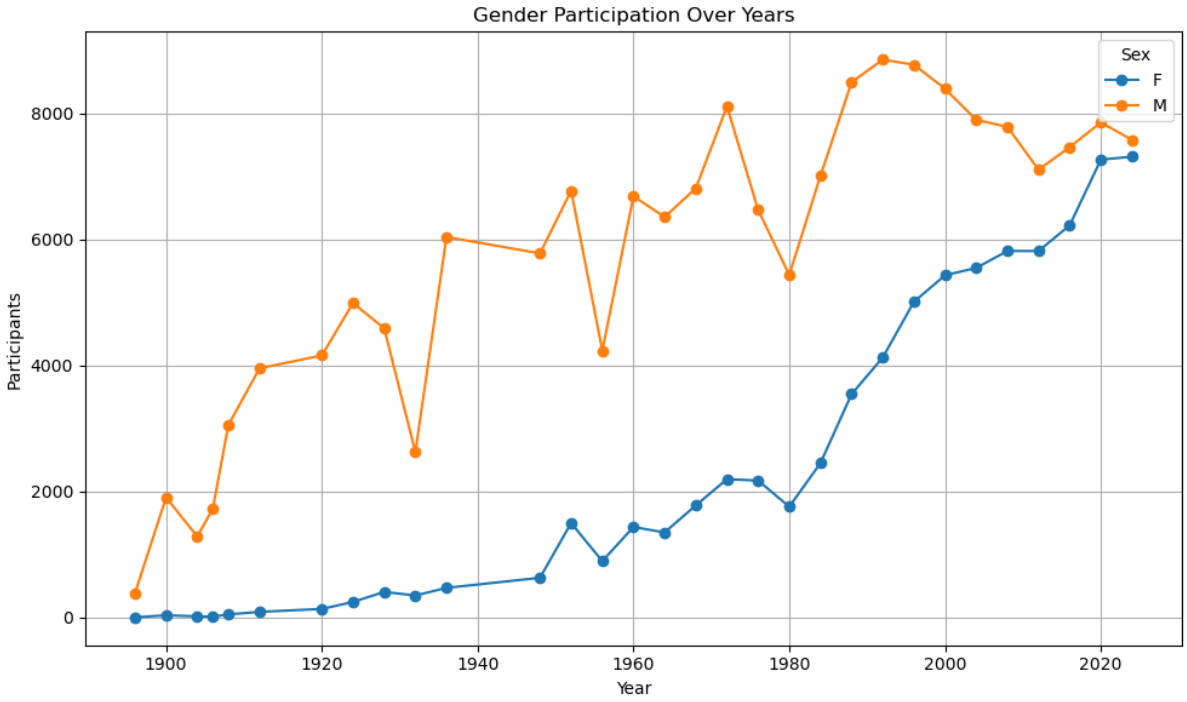
Event-wise Medal Distribution (Top 20 Events)

In [36]:
```
top20_events = (df[df['medal_won']]
                .groupby('Event')['medal_won']
                .sum()
                .sort_values(ascending=False)
                .head(20))

top20_events.plot(kind='bar')
plt.title("Top 20 Events by Medal Count")
plt.xlabel("Event")
plt.ylabel("Medal Count")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Gender Ratio Over Years (Participation Trend)

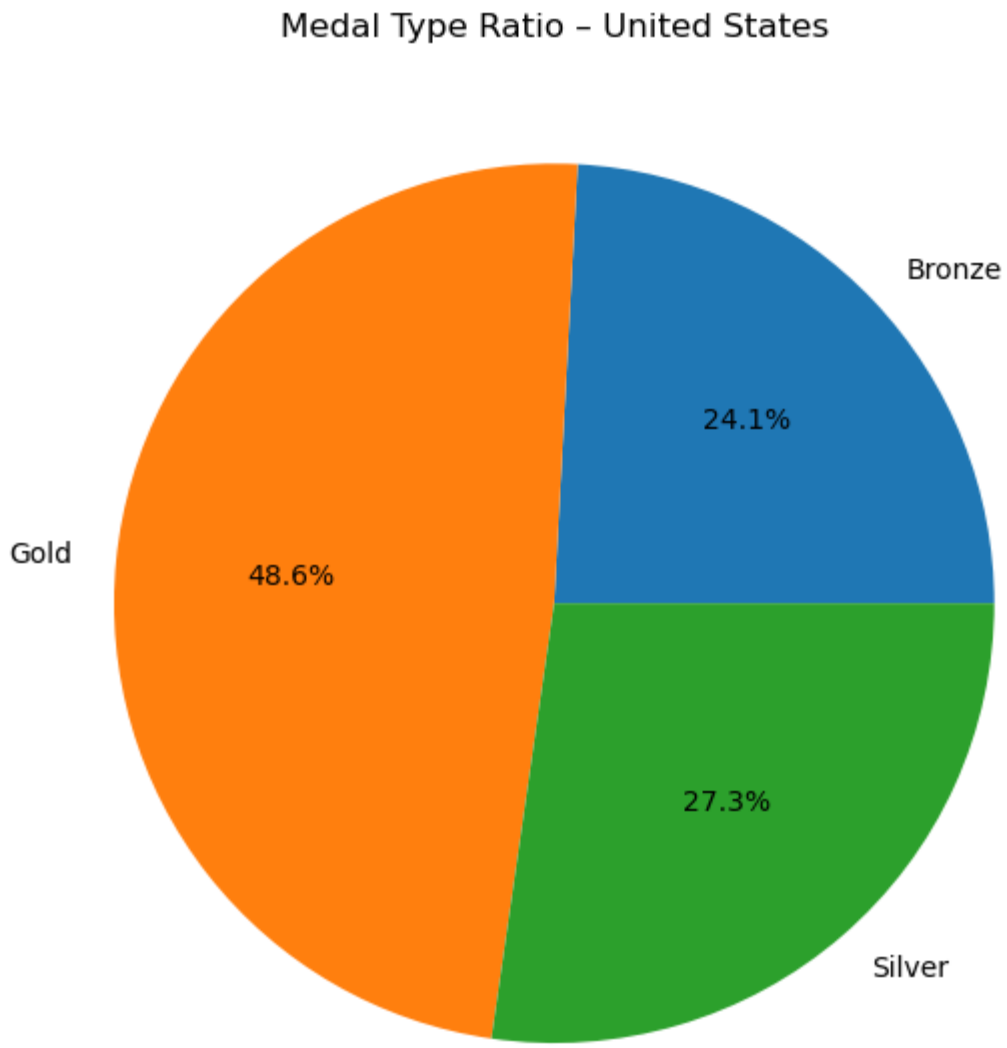
In [37]:
```
gender_year = df.groupby(['Year','Sex'])['Name'].count().unstack()

gender_year.plot(marker='o')
plt.title("Gender Participation Over Years")
plt.xlabel("Year")
plt.ylabel("Participants")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Medal Type Ratio (Pie Chart) – Select Country

(Example: USA)

In [38]:
```
country = "United States"

country_medals = (df[(df['Team']==country) & (df['medal_won'])]
                  .groupby('Medal')['medal_won']
                  .sum())

plt.pie(country_medals.values, labels=country_medals.index,
        autopct='%1.1f%%')

plt.title(f"Medal Type Ratio - {country}")
plt.tight_layout()
plt.show()
```



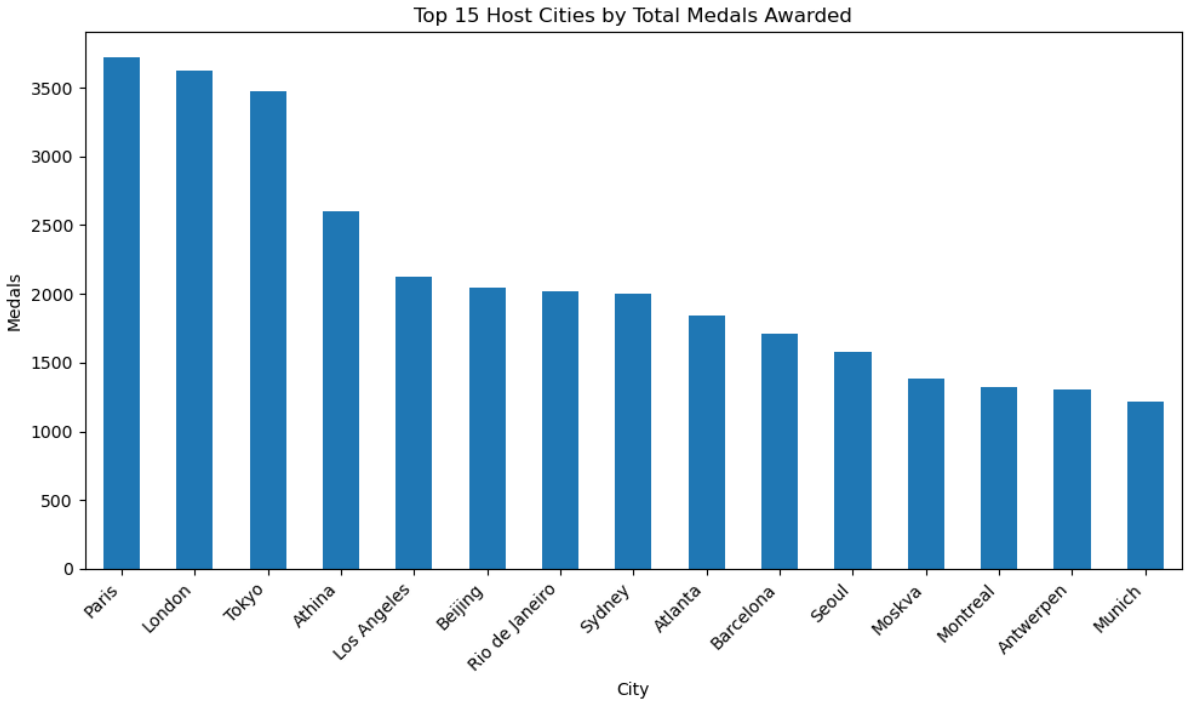
Height vs Weight Scatter Plot

In [39]:
```
if ('Height' in df.columns) and ('Weight' in df.columns):
    df_hw = df[df['Height'].notna() & df['Weight'].notna()]
    plt.scatter(df_hw['Height'], df_hw['Weight'])
    plt.title("Height vs Weight (All Athletes)")
    plt.xlabel("Height")
    plt.ylabel("Weight")
    plt.tight_layout()
    plt.show()
else:
    print("Height/Weight columns not found.")
```

```
Height/Weight columns not found.
```

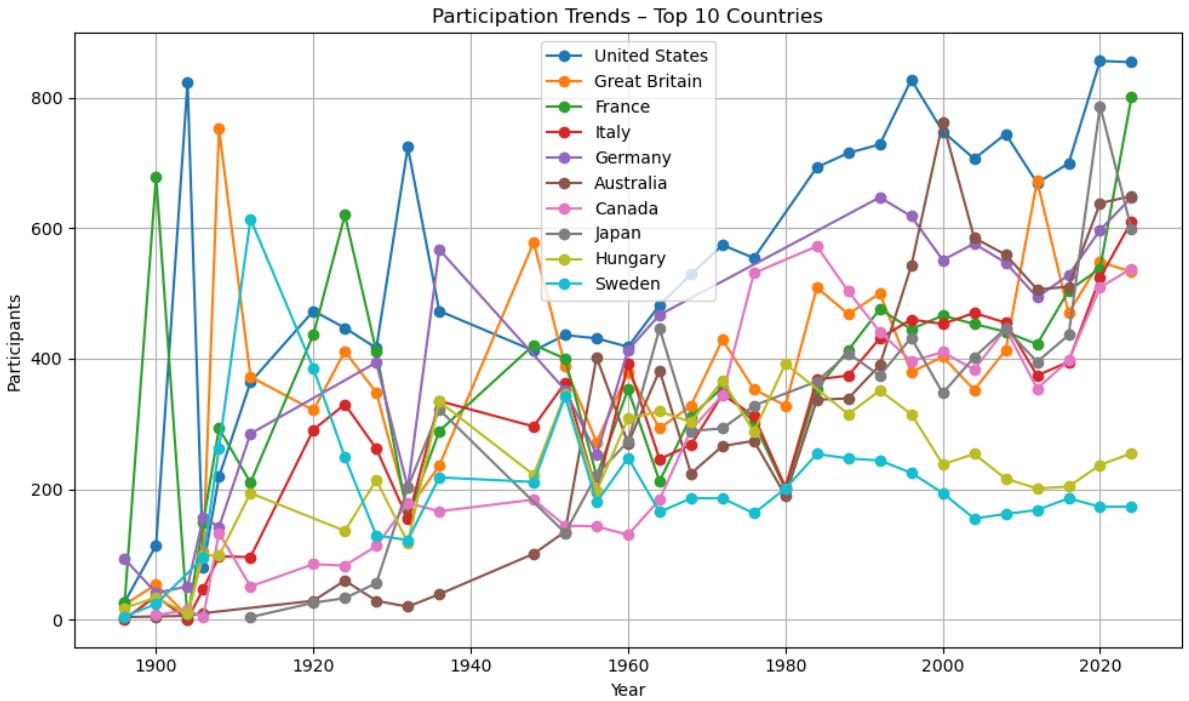
Medal Count by Host City

In [40]:
```
city_medals = (df[df['medal_won']]
               .groupby('City')['medal_won']
               .sum()
               .sort_values(ascending=False)
               .head(15))

city_medals.plot(kind='bar')
plt.title("Top 15 Host Cities by Total Medals Awarded")
plt.xlabel("City")
plt.ylabel("Medals")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Team-wise Participation Trend (Top 10 Countries)

In [41]:
```
top10_teams = (df.groupby('Team')['Name'].count()
               .sort_values(ascending=False)
               .head(10)
               .index)

for t in top10_teams:
    trend = df[df['Team']==t].groupby('Year')['Name'].count()
    plt.plot(trend.index, trend.values, marker='o', label=t)

plt.title("Participation Trends – Top 10 Countries")
plt.xlabel("Year")
plt.ylabel("Participants")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



DONE PROJECT

NAME : JYOTI KALAMBE