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

In [2]:
```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Teams.xlsx to Teams.xlsx
Saving Medals.xlsx to Medals.xlsx
Saving events.csv to events.csv
Saving EntriesGender.xlsx to EntriesGender.xlsx
Saving Athletes.xlsx to Athletes.xlsx
```

In [44]:
```
read_file = pd.read_excel ("Athletes.xlsx")
read_sfile = pd.read_excel ("Medals.xlsx")
read_gfile = pd.read_excel ("EntriesGender.xlsx")
read_tfile = pd.read_excel ("Teams.xlsx")
read_file.to_csv ("Athletes.csv",
                  index = None,
                  header=True)
read_sfile.to_csv ("Medals.csv",
                   index = None,
                  header=True )
read_gfile.to_csv ("EntriesGender.csv",
                  index = None,
                  header=True)
read_tfile.to_csv ("Teams.csv",
                  index = None,
                  header=True)


athletes = pd.read_csv('Athletes.csv')
medals = pd.read_csv('Medals.csv')
gender = pd.read_csv('EntriesGender.csv')
teams = pd.read_csv('Teams.csv')
```

```
/usr/local/lib/python3.10/dist-packages/openpyxl/styles/stylesheet.py:237: UserWarning: Workbook contains no default st
yle, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

In [37]:
```
athletes_df = pd.DataFrame(athletes)
athletes_df.head()
```

Out[37]:

	Name	NOC	Discipline
0	AALERUD Katrine	Norway	Cycling Road
1	ABAD Nestor	Spain	Artistic Gymnastics
2	ABAGNALE Giovanni	Italy	Rowing
3	ABALDE Alberto	Spain	Basketball
4	ABALDE Tamara	Spain	Basketball

In [38]:
```
athletes_df.rename(columns = {'Discipline':'Sport'}, inplace = True)
```

In [40]:
```
athletes_df.head()
```

Out[40]:

	Name	NOC	Sport
0	AALERUD Katrine	Norway	Cycling Road
1	ABAD Nestor	Spain	Artistic Gymnastics
2	ABAGNALE Giovanni	Italy	Rowing
3	ABALDE Alberto	Spain	Basketball
4	ABALDE Tamara	Spain	Basketball

In [41]:
```
athletes_df.describe()
```

Out[41]:

	Name	NOC	Sport
count	11085	11085	11085
unique	11062	206	46
top	CHEN Yang	United States of America	Athletics
freq	2	615	2068

In [45]:
```
medals_df = pd.DataFrame(medals)
gender_df = pd.DataFrame(gender)
teams_df = pd.DataFrame(teams)
```

In [47]:
```
medals_df.rename(columns = {'Team/NOC':'NOC'}, inplace = True)
```

In [48]:
```
medals_df.head()
```

Out[48]:

	Rank	NOC	Gold	Silver	Bronze	Total	Rank by Total
0	1	United States of America	39	41	33	113	1
1	2	People's Republic of China	38	32	18	88	2
2	3	Japan	27	14	17	58	5
3	4	Great Britain	22	21	22	65	4
4	5	ROC	20	28	23	71	3

In [49]:
```
aathletes = athletes_df.merge(medals_df, how='left', on='NOC')
aathletes.head()
```

Out[49]:

	Name	NOC	Sport	Rank	Gold	Silver	Bronze	Total	Rank by Total
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	29.0
1	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	17.0
2	ABAGNALE Giovanni	Italy	Rowing	10.0	10.0	10.0	20.0	40.0	7.0
3	ABALDE Alberto	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	17.0
4	ABALDE Tamara	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	17.0

In [92]:
```
aathletes.rename(columns = {'Total':'TotalMedals'}, inplace = True)
aathletes.drop(columns=['Rank by Total'], inplace = True)
aathletes.head()
```

Out[92]:

	Name	NOC	Sport	Rank	Gold	Silver	Bronze	TotalMedals
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0
1	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0
2	ABAGNALE Giovanni	Italy	Rowing	10.0	10.0	10.0	20.0	40.0
3	ABALDE Alberto	Spain	Basketball	22.0	3.0	8.0	6.0	17.0
4	ABALDE Tamara	Spain	Basketball	22.0	3.0	8.0	6.0	17.0

In [93]:
```
gender_df.head()
```

Out[93]:

	Sport	Female	Male	Total
0	3x3 Basketball	32	32	64
1	Archery	64	64	128
2	Artistic Gymnastics	98	98	196
3	Artistic Swimming	105	0	105
4	Athletics	969	1072	2041

In [95]:
```
gender_df.rename(columns = {'Discipline':'Sport', 'Total' : 'TotalPlayers'}, inplace = True)
```

In [96]:
```
gender_df.head()
```

Out[96]:

	Sport	Female	Male	TotalPlayers
0	3x3 Basketball	32	32	64
1	Archery	64	64	128
2	Artistic Gymnastics	98	98	196
3	Artistic Swimming	105	0	105
4	Athletics	969	1072	2041

In [97]:
```
aathletes_gender = aathletes.merge(gender_df, how='left', on='Sport')
aathletes_gender.head()
```

Out[97]:

	Name	NOC	Sport	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201
1	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	98	98	196
2	ABAGNALE Giovanni	Italy	Rowing	10.0	10.0	10.0	20.0	40.0	257	265	522
3	ABALDE Alberto	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288
4	ABALDE Tamara	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288

In [118…
```
bathletes_df = pd.DataFrame(aathletes_gender)
bathletes_df.head()
```

Out[118]:

	Name	NOC	Sport	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201
1	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	98	98	196
2	ABAGNALE Giovanni	Italy	Rowing	10.0	10.0	10.0	20.0	40.0	257	265	522
3	ABALDE Alberto	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288
4	ABALDE Tamara	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288

In [119…
```
bathletes_df.head()
```

Out[119]:

	Name	NOC	Sport	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201
1	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	98	98	196
2	ABAGNALE Giovanni	Italy	Rowing	10.0	10.0	10.0	20.0	40.0	257	265	522
3	ABALDE Alberto	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288
4	ABALDE Tamara	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288

In [120…
```
teams_df = pd.DataFrame(teams)
teams_df.head()
```

Out[120]:

	Name	Discipline	NOC	Event
0	Belgium	3x3 Basketball	Belgium	Men
1	China	3x3 Basketball	People's Republic of China	Men
2	China	3x3 Basketball	People's Republic of China	Women
3	France	3x3 Basketball	France	Women
4	Italy	3x3 Basketball	Italy	Women

In [122…
```
ateams_df = pd.DataFrame(teams_df)
ateams_df.head()
ateams_df.rename(columns = {'Name' : 'Country', 'Discipline' : 'Sport'}, inplace = True)
ateams_df.head()
```

Out[122]:

	Country	Sport	NOC	Event
0	Belgium	3x3 Basketball	Belgium	Men
1	China	3x3 Basketball	People's Republic of China	Men
2	China	3x3 Basketball	People's Republic of China	Women
3	France	3x3 Basketball	France	Women
4	Italy	3x3 Basketball	Italy	Women

In [138…

```
olympics_df.drop_duplicates(subset=['Name'], keep='first', inplace=True)
olympics_df.head()
```

Out[138]:

	Name	NOC	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers	Country	Event	Sports
0	AALERUD Katrine	Norway	20.0	4.0	2.0	2.0	8.0	70	131	201	Mol A./Sorum C.	Men	Cycling Road\nBeach Volleyball
3	ABAD Nestor	Spain	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Mixed Team	Artistic Gymnastics\nArchery
23	ABAGNALE Giovanni	Italy	10.0	10.0	10.0	20.0	40.0	257	265	522	Italy	Women	Rowing\n3x3 Basketball
60	ABALDE Alberto	Spain	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Mixed Team	Basketball\nArchery
80	ABALDE Tamara	Spain	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Mixed Team	Basketball\nArchery

In [139…

```
olympics_df = pd.DataFrame(bathletes_df.merge(ateams_df, how='left', on='NOC'))
olympics_df.head()
```

Out[139]:

	Name	NOC	Sport_x	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers	Country	Sport_y	Event
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201	Mol A./Sorum C.	Beach Volleyball	Men
1	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201	Norway	Handball	Men
2	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201	Norway	Handball	Women
3	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Archery	Mixed Team
4	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Artistic Gymnastics	Men's Team

In [140…

```
olympics_df.drop_duplicates(subset=['Name'], keep='first', inplace=True)
olympics_df.head()
```

Out[140]:

	Name	NOC	Sport_x	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers	Country	Sport_y	Event
0	AALERUD Katrine	Norway	Cycling Road	20.0	4.0	2.0	2.0	8.0	70	131	201	Mol A./Sorum C.	Beach Volleyball	Men
3	ABAD Nestor	Spain	Artistic Gymnastics	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Archery	Mixed Team
23	ABAGNALE Giovanni	Italy	Rowing	10.0	10.0	10.0	20.0	40.0	257	265	522	Italy	3x3 Basketball	Women
60	ABALDE Alberto	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Archery	Mixed Team
80	ABALDE Tamara	Spain	Basketball	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Archery	Mixed Team

In [162…

```
#pd.get_dummies(olympics_df['Sport_x'])
df = pd.DataFrame(olympics_df['Sport_x'])
df.head()
#pd.get_dummies(olympics_df['Sport_y'])
```

```
df1 = pd.DataFrame(olympics_df['Sport_y'])
df1.head()
dolympics_df= pd.concat([df, df1], axis=0, ignore_index=True)
dolympics_df['Sports'] = pd.concat([olympics_df['Sport_x'], olympics_df['Sport_y']], ignore_index=True)
dolympics_df.head()
```

Out[162]:

	Sport_x	Sport_y	Sports
0	Cycling Road	NaN	Cycling Road
1	Artistic Gymnastics	NaN	Artistic Gymnastics
2	Rowing	NaN	Rowing
3	Basketball	NaN	Basketball
4	Basketball	NaN	Basketball

In [167…

```
z = olympics_df.merge(dolympics_df, left_index=True, right_index=True)
z.head()
```

Out[167]:

	Name	NOC	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers	Country	Event	Sport_x	Sport_y	Spor
0	AALERUD Katrine	Norway	20.0	4.0	2.0	2.0	8.0	70	131	201	Mol A./Sorum C.	Men	Cycling Road	NaN	Cyclir Roa
3	ABAD Nestor	Spain	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Mixed Team	Basketball	NaN	Basketb
23	ABAGNALE Giovanni	Italy	10.0	10.0	10.0	20.0	40.0	257	265	522	Italy	Women	Wrestling	NaN	Wrestlir
60	ABALDE Alberto	Spain	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Mixed Team	Athletics	NaN	Athleti
80	ABALDE Tamara	Spain	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Mixed Team	Triathlon	NaN	Triathlc

In [168…

```
z.drop(columns=['Sport_x', 'Sport_y'], inplace = True)
z.head()
```

Out[168]:

	Name	NOC	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers	Country	Event	Sports
0	AALERUD Katrine	Norway	20.0	4.0	2.0	2.0	8.0	70	131	201	Mol A./Sorum C.	Men	Cycling Road
3	ABAD Nestor	Spain	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Mixed Team	Basketball
23	ABAGNALE Giovanni	Italy	10.0	10.0	10.0	20.0	40.0	257	265	522	Italy	Women	Wrestling
60	ABALDE Alberto	Spain	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Mixed Team	Athletics
80	ABALDE Tamara	Spain	22.0	3.0	8.0	6.0	17.0	144	144	288	Spain	Mixed Team	Triathlon

In [169…

```
olympics_df.isnull().sum()
```

Out[169]:

```
Name              0
NOC               0
Rank            866
Gold            866
Silver          866
Bronze          866
TotalMedals     866
Female            0
Male              0
TotalPlayers      0
Country         938
Event           938
dtype: int64
```

In [170…

```
z.drop_duplicates(subset=['NOC'], keep='first', inplace=True)
z.head()
```

Out[170]:

	Name	NOC	Rank	Gold	Silver	Bronze	TotalMedals	Female	Male	TotalPlayers	Country	Event	Sports
0	AALERUD Katrine	Norway	20.0	4.0	2.0	2.0	8.0	70	131	201	Mol A./Sorum C.	Men	Cycling Road
3	ABAD Nestor	Spain	22.0	3.0	8.0	6.0	17.0	98	98	196	Spain	Mixed Team	Basketball
23	ABAGNALE Giovanni	Italy	10.0	10.0	10.0	20.0	40.0	257	265	522	Italy	Women	Wrestling
100	ABALO Luc	France	8.0	10.0	12.0	11.0	33.0	168	168	336	France	Women	Wrestling
133	ABAROA Cesar	Chile	NaN	NaN	NaN	NaN	NaN	257	265	522	Grimalt M./Grimalt E.	Men	Athletics

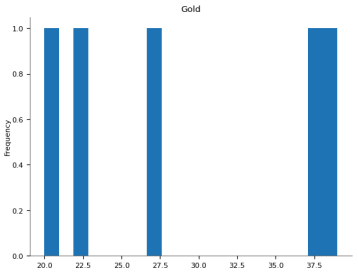
In [177…

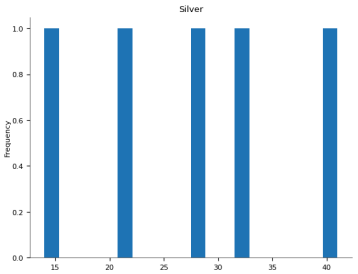
```
olympics1_df = pd.DataFrame(z)
grouped_df = olympics1_df.groupby('NOC').sum()
sorted_df = grouped_df[['Gold', 'Silver', 'Bronze','TotalMedals']].sort_values('Gold', ascending=False).reset_index()
sorted_df.head()
```

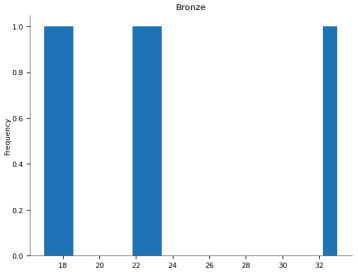
Out[177]:

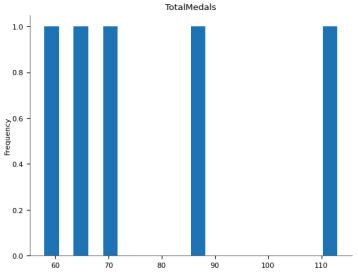
	NOC	Gold	Silver	Bronze	TotalMedals
0	United States of America	39.0	41.0	33.0	113.0
1	People's Republic of China	38.0	32.0	18.0	88.0
2	Japan	27.0	14.0	17.0	58.0
3	Great Britain	22.0	21.0	22.0	65.0
4	ROC	20.0	28.0	23.0	71.0

Distributions

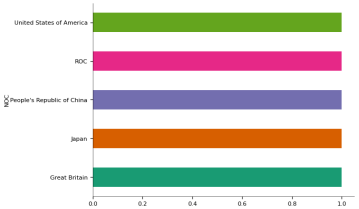




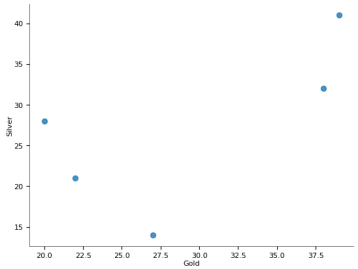


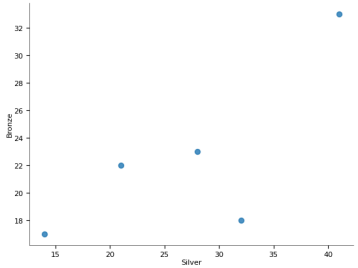


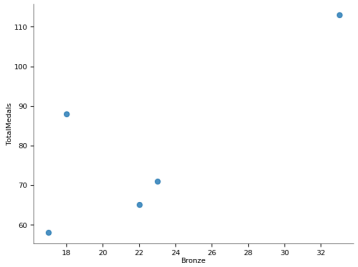
Categorical distributions



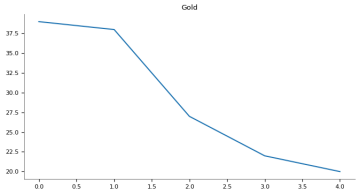
2-d distributions

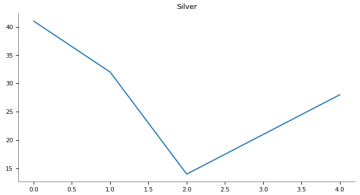


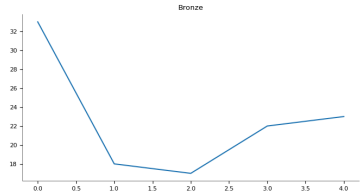


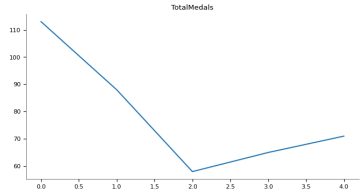


Values



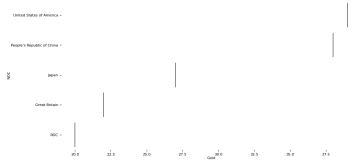




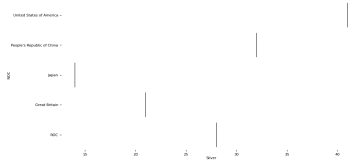


Faceted distributions

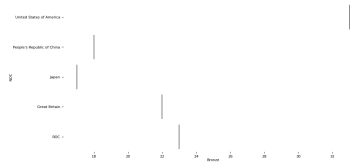
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue
` and set `legend=False` for the same effect.
```



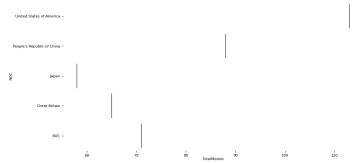

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue
` and set `legend=False` for the same effect.
```




```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue
` and set `legend=False` for the same effect.
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue
` and set `legend=False` for the same effect.
```



In [192…

```
goldcountries = sorted_df[sorted_df['Gold'] > 0]
goldcountries
```

Out[192]:

	NOC	Gold	Silver	Bronze	TotalMedals
0	United States of America	39.0	41.0	33.0	113.0
1	People's Republic of China	38.0	32.0	18.0	88.0
2	Japan	27.0	14.0	17.0	58.0
3	Great Britain	22.0	21.0	22.0	65.0
4	ROC	20.0	28.0	23.0	71.0
...	...	...	...	...	...
57	Slovakia	1.0	2.0	1.0	4.0
58	Fiji	1.0	0.0	1.0	2.0
59	Venezuela	1.0	3.0	0.0	4.0
60	Morocco	1.0	0.0	0.0	1.0
61	South Africa	1.0	2.0	0.0	3.0

62 rows × 5 columns

In [193…

```
country_shortforms = {
    'United States of America': 'USA',
    "People's Republic of China": 'CHN',
    'Japan': 'JPN',
    'Great Britain': 'GBR',
    'ROC': 'ROC',  # Russian Olympic Committee
    'Australia': 'AUS',
    'Netherlands': 'NED',
    'France': 'FRA',
    'Germany': 'GER',
    'Italy': 'ITA',
    'Canada': 'CAN',
    'Brazil': 'BRA',
    'New Zealand': 'NZL',
    'Cuba': 'CUB',
    'Hungary': 'HUN',
    'Republic of Korea': 'KOR',
    'Poland': 'POL',
    'Czech Republic': 'CZE',
    'Kenya': 'KEN',
    'Norway': 'NOR',
    'Jamaica': 'JAM',
    'Spain': 'ESP',
    'Sweden': 'SWE',
    'Switzerland': 'SUI',
    'Denmark': 'DEN',
    'Croatia': 'CRO',
    'Islamic Republic of Iran': 'IRI',
    'Serbia': 'SRB',
    'Belgium': 'BEL',
    'Bulgaria': 'BUL',
    'Slovenia': 'SLO',
    'Uzbekistan': 'UZB',
    'Georgia': 'GEO',
    'Chinese Taipei': 'TPE',
    'Turkey': 'TUR',
    'Greece': 'GRE',
    'Uganda': 'UGA',
    'Ecuador': 'ECU',
    'Ireland': 'IRL',
    'Israel': 'ISR',
    'Qatar': 'QAT',
    'Bahamas': 'BAH',
    'Kosovo': 'KOS',
```

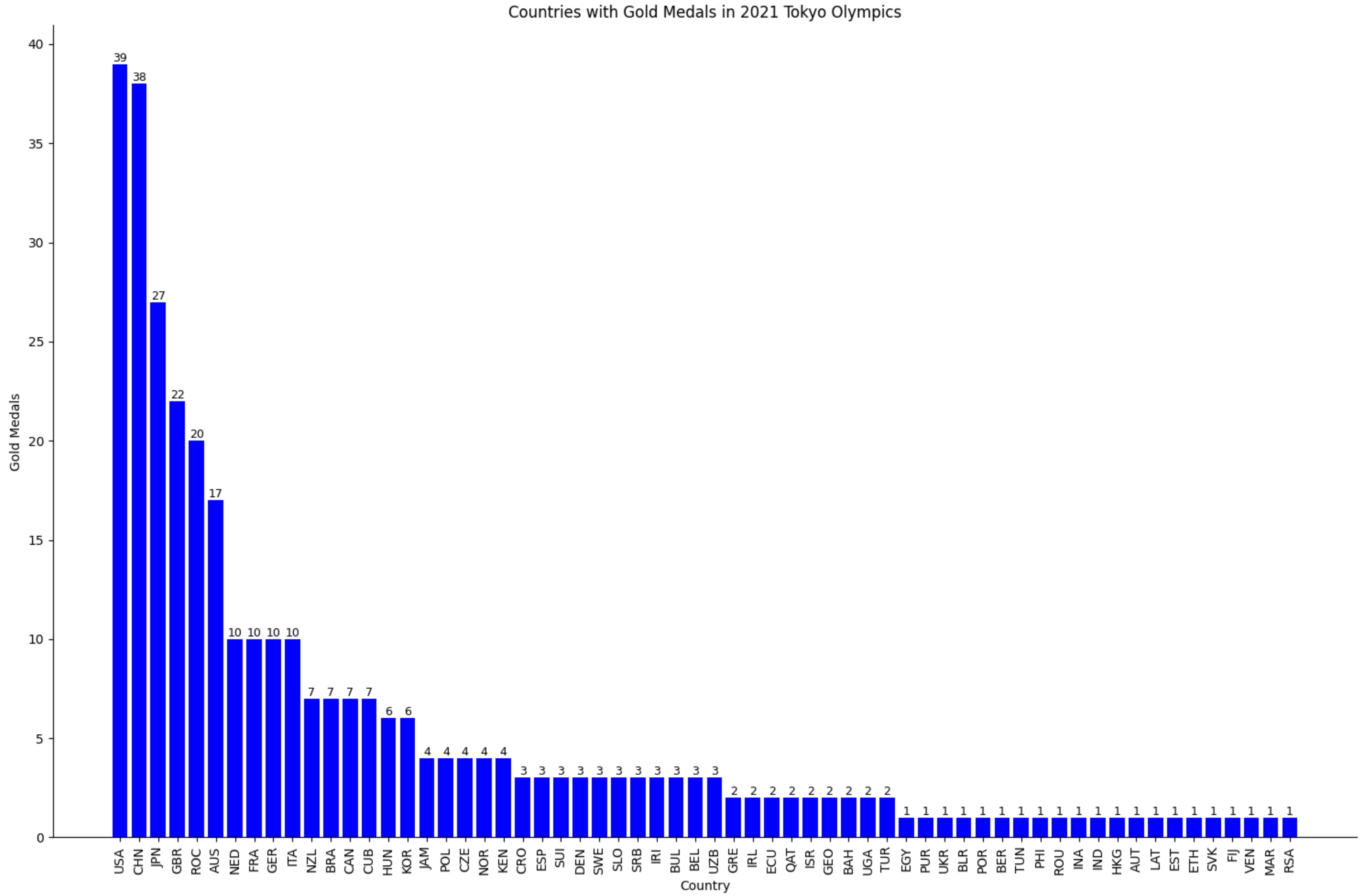
```
    'Ukraine': 'UKR',
    'Belarus': 'BLR',
    'Romania': 'ROU',
    'Venezuela': 'VEN',
    'India': 'IND',
    'Hong Kong, China': 'HKG',
    'Philippines': 'PHI',
    'Slovakia': 'SVK',
    'South Africa': 'RSA',
    'Austria': 'AUT',
    'Egypt': 'EGY',
    'Indonesia': 'INA',
    'Ethiopia': 'ETH',
    'Portugal': 'POR',
    'Tunisia': 'TUN',
    'Estonia': 'EST',
    'Fiji': 'FIJ',
    'Latvia': 'LAT',
    'Thailand': 'THA',
    'Bermuda': 'BER',
    'Morocco': 'MAR',
    'Puerto Rico': 'PUR'
}
```

In [197…

```
import matplotlib.pyplot as plt

# Countries with zero gold medals
sorted_df = sorted_df[sorted_df['Gold'] > 0]

# Plotting with improved readability
plt.figure(figsize=(15, 10))

# Create bar plot with filtered country names on x-axis and gold medal counts on y-axis
bars = plt.bar(sorted_df['NOC'], sorted_df['Gold'], color=['blue'])

country_shortforms = {
    'United States of America': 'USA',
    "People's Republic of China": 'CHN',
    'Japan': 'JPN',
    'Great Britain': 'GBR',
    'ROC': 'ROC',
    'Australia': 'AUS',
    'Netherlands': 'NED',
    'France': 'FRA',
    'Germany': 'GER',
```

```
    'Italy': 'ITA',
    'Canada': 'CAN',
    'Brazil': 'BRA',
    'New Zealand': 'NZL',
    'Cuba': 'CUB',
    'Hungary': 'HUN',
    'Republic of Korea': 'KOR',
    'Poland': 'POL',
    'Czech Republic': 'CZE',
    'Kenya': 'KEN',
    'Norway': 'NOR',
    'Jamaica': 'JAM',
    'Spain': 'ESP',
    'Sweden': 'SWE',
    'Switzerland': 'SUI',
    'Denmark': 'DEN',
    'Croatia': 'CRO',
    'Islamic Republic of Iran': 'IRI',
    'Serbia': 'SRB',
    'Belgium': 'BEL',
    'Bulgaria': 'BUL',
    'Slovenia': 'SLO',
    'Uzbekistan': 'UZB',
    'Georgia': 'GEO',
    'Chinese Taipei': 'TPE',
    'Turkey': 'TUR',
    'Greece': 'GRE',
    'Uganda': 'UGA',
    'Ecuador': 'ECU',
    'Ireland': 'IRL',
    'Israel': 'ISR',
    'Qatar': 'QAT',
    'Bahamas': 'BAH',
    'Kosovo': 'KOS',
    'Ukraine': 'UKR',
    'Belarus': 'BLR',
    'Romania': 'ROU',
    'Venezuela': 'VEN',
    'India': 'IND',
    'Hong Kong, China': 'HKG',
    'Philippines': 'PHI',
    'Slovakia': 'SVK',
    'South Africa': 'RSA',
    'Austria': 'AUT',
    'Egypt': 'EGY',
```

```
    'Indonesia': 'INA',
    'Ethiopia': 'ETH',
    'Portugal': 'POR',
    'Tunisia': 'TUN',
    'Estonia': 'EST',
    'Fiji': 'FIJ',
    'Latvia': 'LAT',
    'Thailand': 'THA',
    'Bermuda': 'BER',
    'Morocco': 'MAR',
    'Puerto Rico': 'PUR'
}


# Set plot title and labels
plt.title('Countries with Gold Medals in 2021 Tokyo Olympics')
plt.xlabel('Country')
plt.ylabel('Gold Medals')

# Rotate x-axis labels vertically
plt.xticks(sorted_df['NOC'], [country_shortforms[c] for c in sorted_df['NOC']], rotation='vertical', fontsize=10)

# Hide top and right spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Add text annotations on top of each bar
for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height(), str(int(bar.get_height())), ha='center', va='bottom', f

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



In [208…

```
countries_with_silver_medals = sorted_df[sorted_df['Silver'] > 0]
countries_with_silver_medals
```

Out[208]:

	NOC	Gold	Silver	Bronze	TotalMedals
0	United States of America	39.0	41.0	33.0	113.0
1	People's Republic of China	38.0	32.0	18.0	88.0
2	Japan	27.0	14.0	17.0	58.0
3	Great Britain	22.0	21.0	22.0	65.0
4	ROC	20.0	28.0	23.0	71.0
5	Australia	17.0	7.0	22.0	46.0
6	Netherlands	10.0	12.0	14.0	36.0
7	France	10.0	12.0	11.0	33.0
8	Germany	10.0	11.0	16.0	37.0
9	Italy	10.0	10.0	20.0	40.0
10	New Zealand	7.0	6.0	7.0	20.0
11	Brazil	7.0	6.0	8.0	21.0
12	Canada	7.0	6.0	11.0	24.0
13	Cuba	7.0	3.0	5.0	15.0
14	Hungary	6.0	7.0	7.0	20.0
15	Republic of Korea	6.0	4.0	10.0	20.0
16	Jamaica	4.0	1.0	4.0	9.0
17	Poland	4.0	5.0	5.0	14.0
18	Czech Republic	4.0	4.0	3.0	11.0
19	Norway	4.0	2.0	2.0	8.0
20	Kenya	4.0	4.0	2.0	10.0
21	Croatia	3.0	3.0	2.0	8.0
22	Spain	3.0	8.0	6.0	17.0
23	Switzerland	3.0	4.0	6.0	13.0
24	Denmark	3.0	4.0	4.0	11.0

	NOC	Gold	Silver	Bronze	TotalMedals
25	Sweden	3.0	6.0	0.0	9.0
26	Slovenia	3.0	1.0	1.0	5.0
27	Serbia	3.0	1.0	5.0	9.0
28	Islamic Republic of Iran	3.0	2.0	2.0	7.0
29	Bulgaria	3.0	1.0	2.0	6.0
30	Belgium	3.0	1.0	3.0	7.0
32	Greece	2.0	1.0	1.0	4.0
34	Ecuador	2.0	1.0	0.0	3.0
37	Georgia	2.0	5.0	1.0	8.0
39	Uganda	2.0	1.0	1.0	4.0
40	Turkey	2.0	2.0	9.0	13.0
41	Egypt	1.0	1.0	4.0	6.0
43	Ukraine	1.0	6.0	12.0	19.0
44	Belarus	1.0	3.0	3.0	7.0
45	Portugal	1.0	1.0	2.0	4.0
47	Tunisia	1.0	1.0	0.0	2.0
48	Philippines	1.0	2.0	1.0	4.0
49	Romania	1.0	3.0	0.0	4.0
50	Indonesia	1.0	1.0	3.0	5.0
51	India	1.0	2.0	4.0	7.0
52	Hong Kong, China	1.0	2.0	3.0	6.0
53	Austria	1.0	1.0	5.0	7.0
56	Ethiopia	1.0	1.0	2.0	4.0
57	Slovakia	1.0	2.0	1.0	4.0
59	Venezuela	1.0	3.0	0.0	4.0

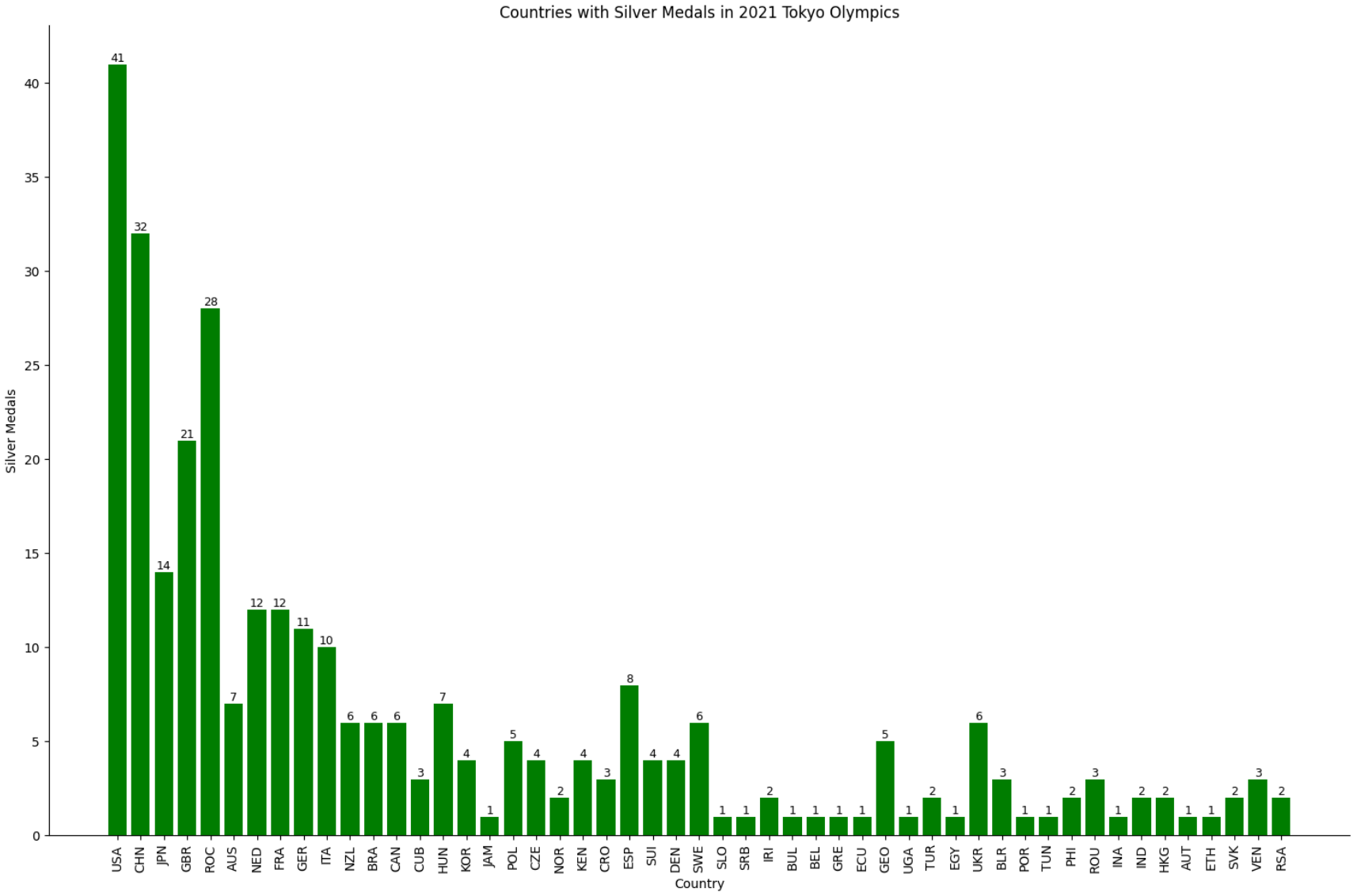
	NOC	Gold	Silver	Bronze	TotalMedals
61	South Africa	1.0	2.0	0.0	3.0

In [211…

```
import matplotlib.pyplot as plt

# Countries with silver medals
sorted_df = sorted_df[sorted_df['Silver'] > 0]

# Plotting with improved readability
plt.figure(figsize=(15, 10))

#Bar plot with filtered country names on x-axis and silver medal counts on y-axis
bars = plt.bar(sorted_df['NOC'], sorted_df['Silver'], color=['green'])

# Short forms of countries
country_short_forms = {
    'United States of America': 'USA',
    "People's Republic of China": 'CHN',
    'Japan': 'JPN',
    'Great Britain': 'GBR',
    'ROC': 'ROC',
    'Australia': 'AUS',
    'Netherlands': 'NED',
    'France': 'FRA',
    'Germany': 'GER',
    'Italy': 'ITA',
    'New Zealand': 'NZL',
    'Brazil': 'BRA',
    'Canada': 'CAN',
    'Cuba': 'CUB',
    'Hungary': 'HUN',
    'Republic of Korea': 'KOR',
    'Jamaica': 'JAM',
    'Poland': 'POL',
    'Czech Republic': 'CZE',
    'Norway': 'NOR',
    'Kenya': 'KEN',
    'Croatia': 'CRO',
    'Spain': 'ESP',
    'Switzerland': 'SUI',
    'Denmark': 'DEN',
    'Sweden': 'SWE',
```

```
    'Slovenia': 'SLO',
    'Serbia': 'SRB',
    'Islamic Republic of Iran': 'IRI',
    'Bulgaria': 'BUL',
    'Belgium': 'BEL',
    'Greece': 'GRE',
    'Ecuador': 'ECU',
    'Georgia': 'GEO',
    'Uganda': 'UGA',
    'Turkey': 'TUR',
    'Egypt': 'EGY',
    'Ukraine': 'UKR',
    'Belarus': 'BLR',
    'Portugal': 'POR',
    'Tunisia': 'TUN',
    'Philippines': 'PHI',
    'Romania': 'ROU',
    'Indonesia': 'INA',
    'India': 'IND',
    'Hong Kong, China': 'HKG',
    'Austria': 'AUT',
    'Ethiopia': 'ETH',
    'Slovakia': 'SVK',
    'Venezuela': 'VEN',
    'South Africa': 'RSA'
}

# Set plot title and labels
plt.title('Countries with Silver Medals in 2021 Tokyo Olympics')
plt.xlabel('Country')
plt.ylabel('Silver Medals')

plt.xticks(sorted_df['NOC'], [country_shortforms[c] for c in sorted_df['NOC']], rotation='vertical', fontsize=10)


# Hide top and right spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Add text annotations on top of each bar
for i, bar in enumerate(bars):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height(), str(int(bar.get_height())), ha='center', va='bottom', f

# Show the plot
```

```
plt.tight_layout()
plt.show()
```



In [212…

```
countries_with_bronze = sorted_df[sorted_df['Bronze'] > 0]
countries_with_bronze
```

Out[212]:

	NOC	Gold	Silver	Bronze	TotalMedals
0	United States of America	39.0	41.0	33.0	113.0
1	People's Republic of China	38.0	32.0	18.0	88.0
2	Japan	27.0	14.0	17.0	58.0
3	Great Britain	22.0	21.0	22.0	65.0
4	ROC	20.0	28.0	23.0	71.0
5	Australia	17.0	7.0	22.0	46.0
6	Netherlands	10.0	12.0	14.0	36.0
7	France	10.0	12.0	11.0	33.0
8	Germany	10.0	11.0	16.0	37.0
9	Italy	10.0	10.0	20.0	40.0
10	New Zealand	7.0	6.0	7.0	20.0
11	Brazil	7.0	6.0	8.0	21.0
12	Canada	7.0	6.0	11.0	24.0
13	Cuba	7.0	3.0	5.0	15.0
14	Hungary	6.0	7.0	7.0	20.0
15	Republic of Korea	6.0	4.0	10.0	20.0
16	Jamaica	4.0	1.0	4.0	9.0
17	Poland	4.0	5.0	5.0	14.0
18	Czech Republic	4.0	4.0	3.0	11.0
19	Norway	4.0	2.0	2.0	8.0
20	Kenya	4.0	4.0	2.0	10.0
21	Croatia	3.0	3.0	2.0	8.0
22	Spain	3.0	8.0	6.0	17.0
23	Switzerland	3.0	4.0	6.0	13.0
24	Denmark	3.0	4.0	4.0	11.0

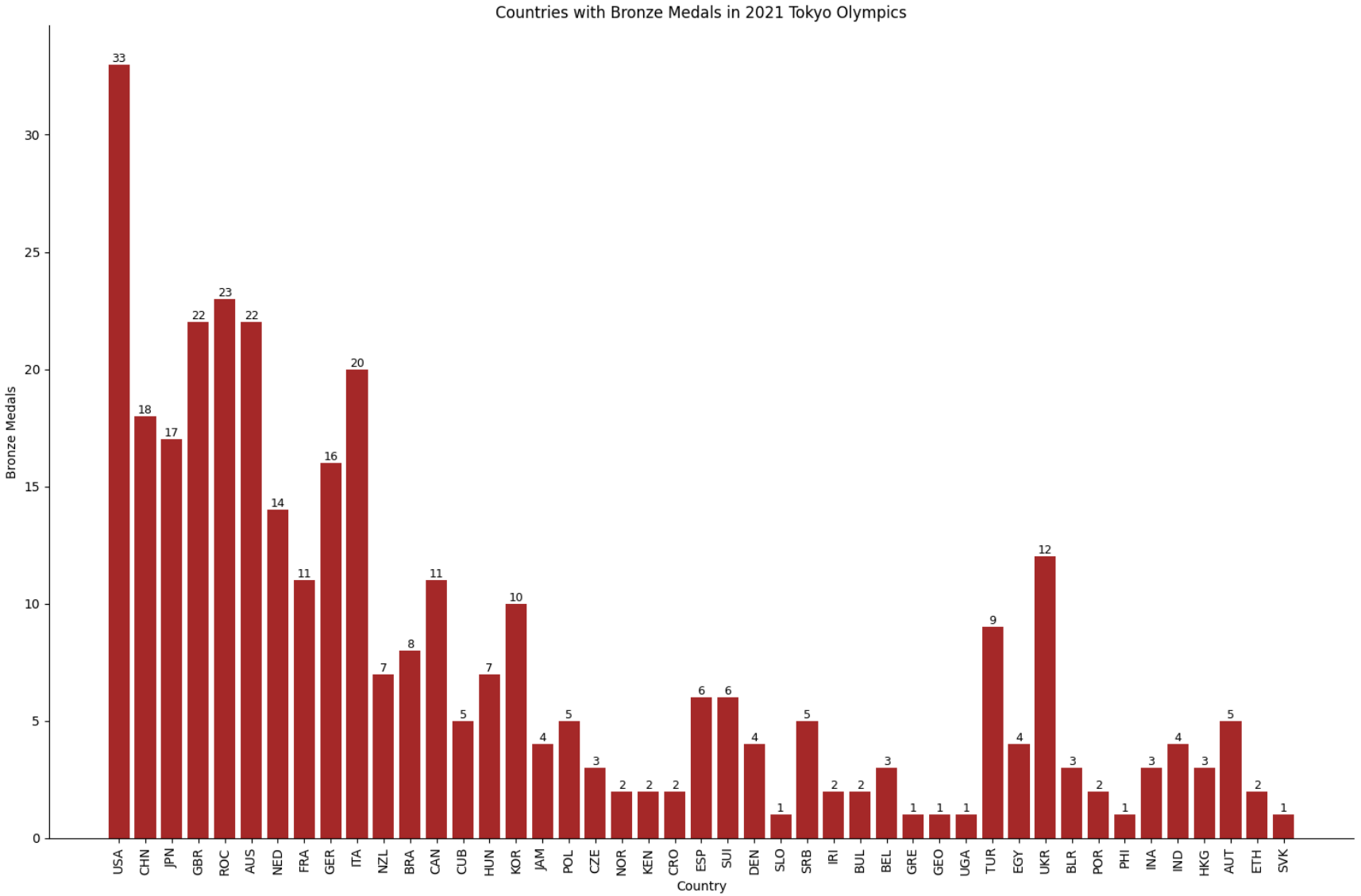
	NOC	Gold	Silver	Bronze	TotalMedals
26	Slovenia	3.0	1.0	1.0	5.0
27	Serbia	3.0	1.0	5.0	9.0
28	Islamic Republic of Iran	3.0	2.0	2.0	7.0
29	Bulgaria	3.0	1.0	2.0	6.0
30	Belgium	3.0	1.0	3.0	7.0
32	Greece	2.0	1.0	1.0	4.0
37	Georgia	2.0	5.0	1.0	8.0
39	Uganda	2.0	1.0	1.0	4.0
40	Turkey	2.0	2.0	9.0	13.0
41	Egypt	1.0	1.0	4.0	6.0
43	Ukraine	1.0	6.0	12.0	19.0
44	Belarus	1.0	3.0	3.0	7.0
45	Portugal	1.0	1.0	2.0	4.0
48	Philippines	1.0	2.0	1.0	4.0
50	Indonesia	1.0	1.0	3.0	5.0
51	India	1.0	2.0	4.0	7.0
52	Hong Kong, China	1.0	2.0	3.0	6.0
53	Austria	1.0	1.0	5.0	7.0
56	Ethiopia	1.0	1.0	2.0	4.0
57	Slovakia	1.0	2.0	1.0	4.0

In [214…

```
import matplotlib.pyplot as plt

# Plotting with improved readability
plt.figure(figsize=(15, 10))

sorted_df = sorted_df[sorted_df['Bronze'] > 0]
```

```python
# Bar plot with filtered country names on x-axis and bronze medal counts on y-axis
bars = plt.bar(sorted_df['NOC'], sorted_df['Bronze'], color='brown')

# Short forms of countries
country_short_forms = {
    'United States of America': 'USA',
    "People's Republic of China": 'CHN',
    'Japan': 'JPN',
    'Great Britain': 'GBR',
    'ROC': 'ROC',
    'Australia': 'AUS',
    'Netherlands': 'NED',
    'France': 'FRA',
    'Germany': 'GER',
    'Italy': 'ITA',
    'New Zealand': 'NZL',
    'Brazil': 'BRA',
    'Canada': 'CAN',
    'Cuba': 'CUB',
    'Hungary': 'HUN',
    'Republic of Korea': 'KOR',
    'Jamaica': 'JAM',
    'Poland': 'POL',
    'Czech Republic': 'CZE',
    'Norway': 'NOR',
    'Kenya': 'KEN',
    'Croatia': 'CRO',
    'Spain': 'ESP',
    'Switzerland': 'SUI',
    'Denmark': 'DEN',
    'Slovenia': 'SLO',
    'Serbia': 'SRB',
    'Islamic Republic of Iran': 'IRI',
    'Bulgaria': 'BUL',
    'Belgium': 'BEL',
    'Greece': 'GRE',
    'Georgia': 'GEO',
    'Uganda': 'UGA',
    'Turkey': 'TUR',
    'Egypt': 'EGY',
    'Ukraine': 'UKR',
    'Belarus': 'BLR',
    'Portugal': 'POR',
    'Philippines': 'PHI',
    'Indonesia': 'INA',
```

```
    'India': 'IND',
    'Hong Kong, China': 'HKG',
    'Austria': 'AUT',
    'Ethiopia': 'ETH',
    'Slovakia': 'SVK'
}

# Set plot title and labels
plt.title('Countries with Bronze Medals in 2021 Tokyo Olympics')
plt.xlabel('Country')
plt.ylabel('Bronze Medals')

plt.xticks(sorted_df['NOC'], [country_short_forms[c] for c in sorted_df['NOC']], rotation='vertical', fontsize=10)

# Hide top and right spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Add text annotations on top of each bar
for i, bar in enumerate(bars):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height(), str(int(bar.get_height())), ha='center', va='bottom', f

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



In [217…

```
totalmedals= sorted_df[sorted_df['TotalMedals'] > 0]
totalmedals
```

Out[217]:

	NOC	Gold	Silver	Bronze	TotalMedals
0	United States of America	39.0	41.0	33.0	113.0
1	People's Republic of China	38.0	32.0	18.0	88.0
2	Japan	27.0	14.0	17.0	58.0
3	Great Britain	22.0	21.0	22.0	65.0
4	ROC	20.0	28.0	23.0	71.0
5	Australia	17.0	7.0	22.0	46.0
6	Netherlands	10.0	12.0	14.0	36.0
7	France	10.0	12.0	11.0	33.0
8	Germany	10.0	11.0	16.0	37.0
9	Italy	10.0	10.0	20.0	40.0
10	New Zealand	7.0	6.0	7.0	20.0
11	Brazil	7.0	6.0	8.0	21.0
12	Canada	7.0	6.0	11.0	24.0
13	Cuba	7.0	3.0	5.0	15.0
14	Hungary	6.0	7.0	7.0	20.0
15	Republic of Korea	6.0	4.0	10.0	20.0
16	Jamaica	4.0	1.0	4.0	9.0
17	Poland	4.0	5.0	5.0	14.0
18	Czech Republic	4.0	4.0	3.0	11.0
19	Norway	4.0	2.0	2.0	8.0
20	Kenya	4.0	4.0	2.0	10.0
21	Croatia	3.0	3.0	2.0	8.0
22	Spain	3.0	8.0	6.0	17.0
23	Switzerland	3.0	4.0	6.0	13.0
24	Denmark	3.0	4.0	4.0	11.0

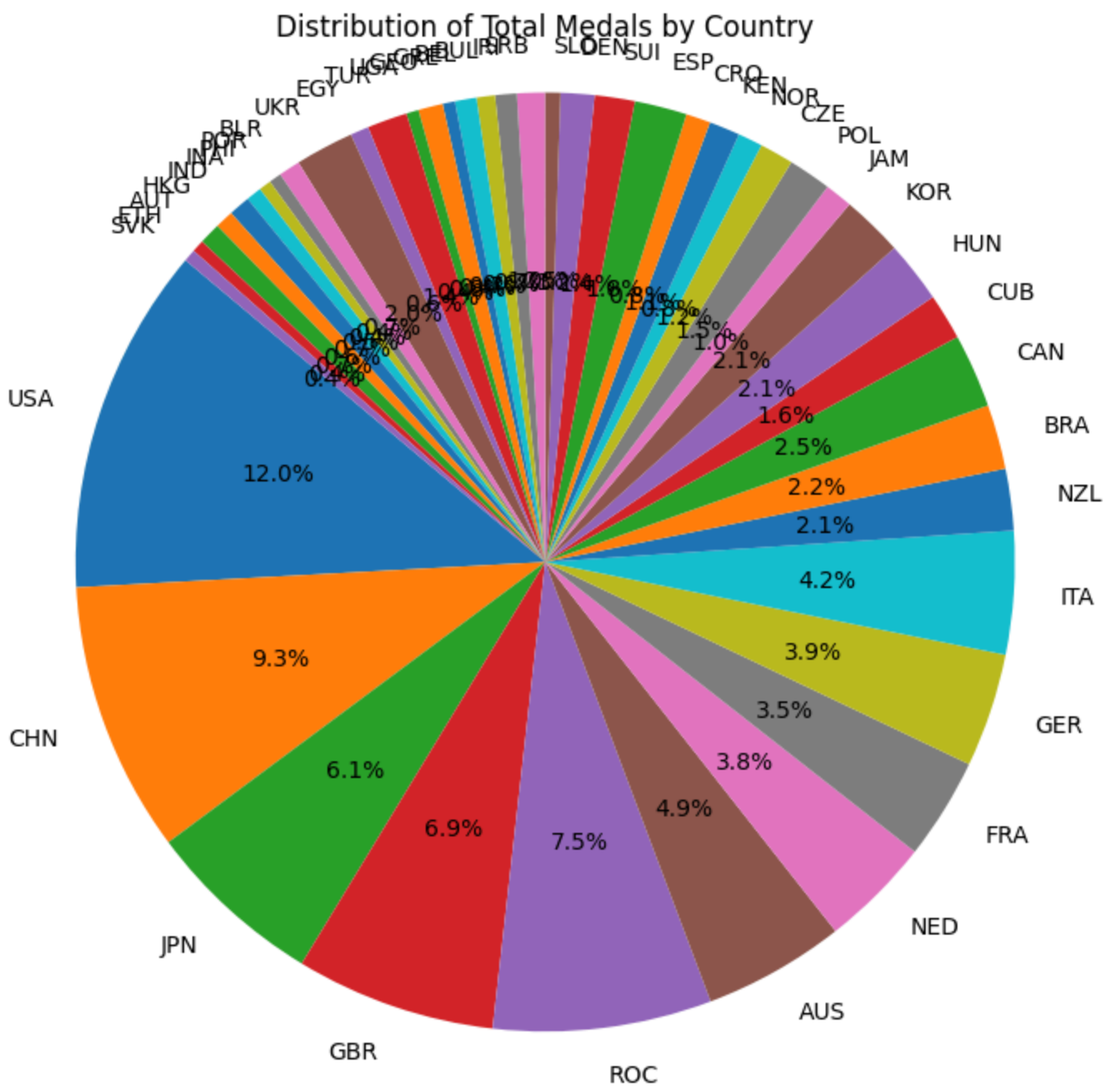
	NOC	Gold	Silver	Bronze	TotalMedals
26	Slovenia	3.0	1.0	1.0	5.0
27	Serbia	3.0	1.0	5.0	9.0
28	Islamic Republic of Iran	3.0	2.0	2.0	7.0
29	Bulgaria	3.0	1.0	2.0	6.0
30	Belgium	3.0	1.0	3.0	7.0
32	Greece	2.0	1.0	1.0	4.0
37	Georgia	2.0	5.0	1.0	8.0
39	Uganda	2.0	1.0	1.0	4.0
40	Turkey	2.0	2.0	9.0	13.0
41	Egypt	1.0	1.0	4.0	6.0
43	Ukraine	1.0	6.0	12.0	19.0
44	Belarus	1.0	3.0	3.0	7.0
45	Portugal	1.0	1.0	2.0	4.0
48	Philippines	1.0	2.0	1.0	4.0
50	Indonesia	1.0	1.0	3.0	5.0
51	India	1.0	2.0	4.0	7.0
52	Hong Kong, China	1.0	2.0	3.0	6.0
53	Austria	1.0	1.0	5.0	7.0
56	Ethiopia	1.0	1.0	2.0	4.0
57	Slovakia	1.0	2.0	1.0	4.0

In [304…

```
#pie chart of total medals
country_short_forms = {
    'United States of America': 'USA',
    "People's Republic of China": 'CHN',
    'Japan': 'JPN',
    'Great Britain': 'GBR',
    'ROC': 'ROC',
```

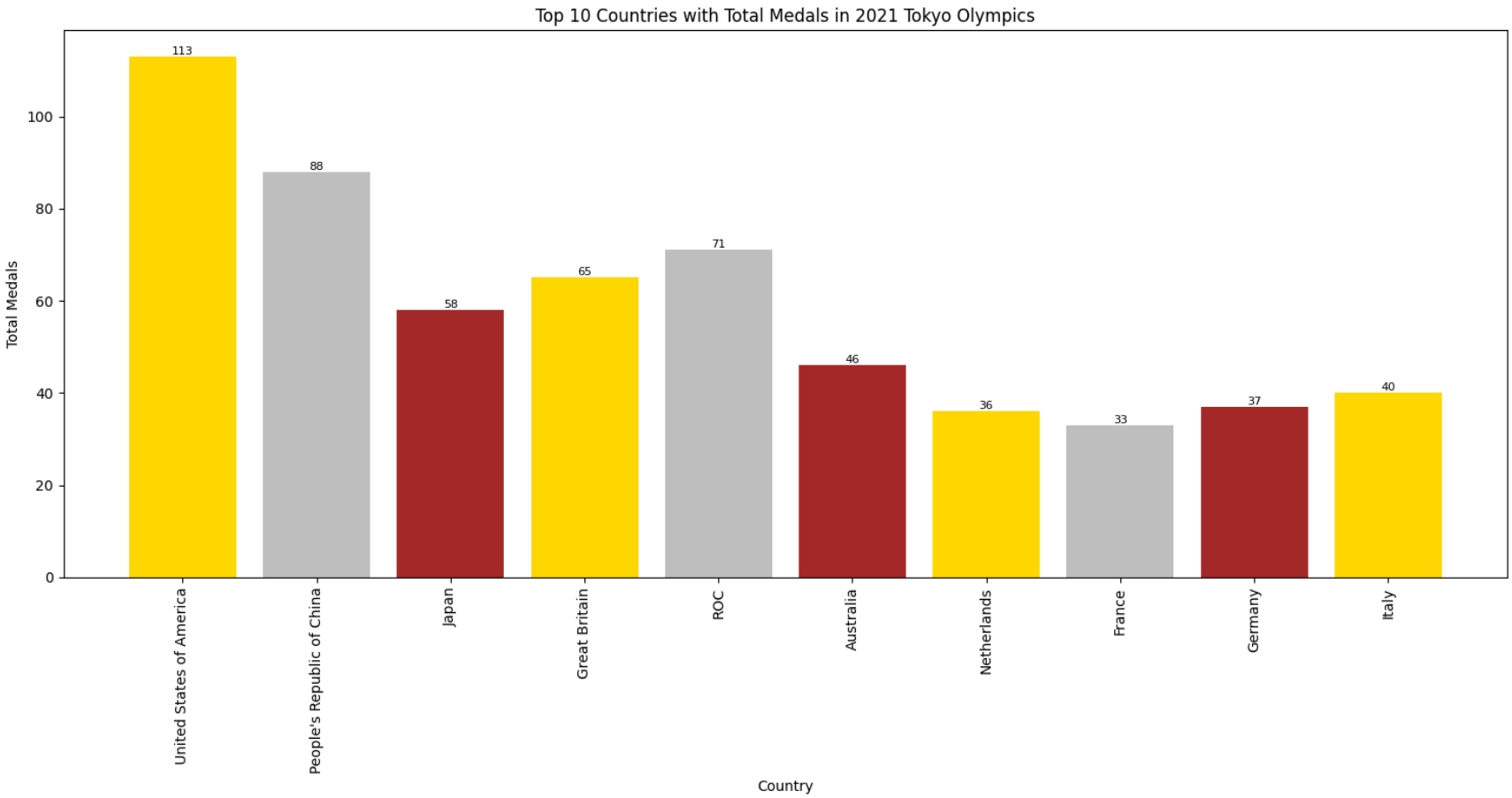
```
    'Australia': 'AUS',
    'Netherlands': 'NED',
    'France': 'FRA',
    'Germany': 'GER',
    'Italy': 'ITA',
    'New Zealand': 'NZL',
    'Brazil': 'BRA',
    'Canada': 'CAN',
    'Cuba': 'CUB',
    'Hungary': 'HUN',
    'Republic of Korea': 'KOR',
    'Jamaica': 'JAM',
    'Poland': 'POL',
    'Czech Republic': 'CZE',
    'Norway': 'NOR',
    'Kenya': 'KEN',
    'Croatia': 'CRO',
    'Spain': 'ESP',
    'Switzerland': 'SUI',
    'Denmark': 'DEN',
    'Slovenia': 'SLO',
    'Serbia': 'SRB',
    'Islamic Republic of Iran': 'IRI',
    'Bulgaria': 'BUL',
    'Belgium': 'BEL',
    'Greece': 'GRE',
    'Georgia': 'GEO',
    'Uganda': 'UGA',
    'Turkey': 'TUR',
    'Egypt': 'EGY',
    'Ukraine': 'UKR',
    'Belarus': 'BLR',
    'Portugal': 'POR',
    'Philippines': 'PHI',
    'Indonesia': 'INA',
    'India': 'IND',
    'Hong Kong, China': 'HKG',
    'Austria': 'AUT',
    'Ethiopia': 'ETH',
    'Slovakia': 'SVK'
}

# Create a pie chart for total medals won by each country
plt.figure(figsize=(8, 8))
plt.pie(sorted_df['TotalMedals'], labels=[country_short_forms[c] for c in sorted_df['NOC']], autopct='%1.1f%%', startan
```

```
plt.title('Distribution of Total Medals by Country')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```



In [223…

```
# Selecting top 10 countries
top_10 = sorted_df.head(10)

# Create a bar chart for top 10 countries with total medals
plt.figure(figsize=(15, 8))
bars = plt.bar(top_10['NOC'], top_10['TotalMedals'], color=['gold', 'silver', 'brown'])
plt.xlabel('Country')
plt.ylabel('Total Medals')
plt.title('Top 10 Countries with Total Medals in 2021 Tokyo Olympics')
plt.xticks(rotation='vertical')
plt.tight_layout()

# Add text annotations on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), va='bottom', ha='center', fontsize=8)

plt.show()
```



In [237…

```
from sklearn.linear_model import LinearRegression
import numpy as np
```

In [241…

```
df = sorted_df
df.shape
df.insert(0, 'Year', [2021]*45)
df.head()
```

Out[241]:

	Year	NOC	Gold	Silver	Bronze	TotalMedals
0	2021	United States of America	39.0	41.0	33.0	113.0
1	2021	People's Republic of China	38.0	32.0	18.0	88.0
2	2021	Japan	27.0	14.0	17.0	58.0
3	2021	Great Britain	22.0	21.0	22.0	65.0
4	2021	ROC	20.0	28.0	23.0	71.0

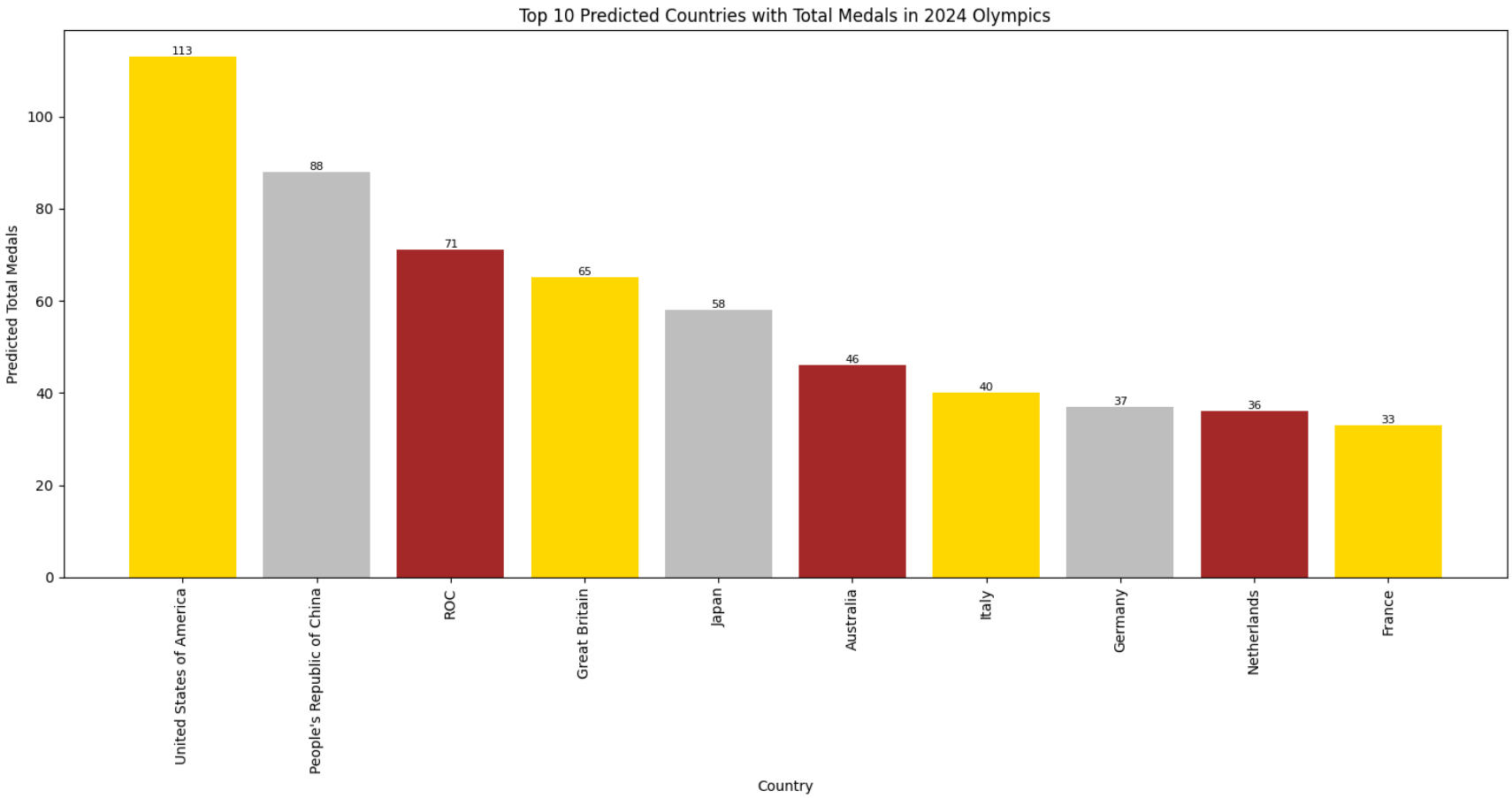
In [248…

```python
# Assume a simple linear increase
years = np.array([2021]).reshape(-1, 1)  # Current year
future_years = np.array([2024]).reshape(-1, 1)  # Year to predict

# Prepare the linear regression model
model = LinearRegression()

# Predict each medal type
predictions = {}
for medal_type in ['Gold', 'Silver', 'Bronze', 'TotalMedals']:
    X = df['Year'].values.reshape(-1, 1)
    y = df[medal_type]

    # Fit the model
    model.fit(X, y)

    # Predict for 2024
    predictions[medal_type] = model.predict(future_years)[0]

# Display the predictions
for medal_type, prediction in predictions.items():
    print(f"Predicted {medal_type} in 2024: {prediction:.2f}")

# Add predicted medals to the DataFrame for 2024
df_pred = df.copy()
df_pred['Gold'] = df_pred['Gold'] + (predictions['Gold'] - df_pred['Gold'].mean())
df_pred['Silver'] = df_pred['Silver'] + (predictions['Silver'] - df_pred['Silver'].mean())
df_pred['Bronze'] = df_pred['Bronze'] + (predictions['Bronze'] - df_pred['Bronze'].mean())
df_pred['TotalMedals'] = df_pred['Gold'] + df_pred['Silver'] + df_pred['Bronze']

# Sort DataFrame by predicted total medals
df_pred = df_pred.sort_values(by='TotalMedals', ascending=False)
```

```
Predicted Gold in 2024: 6.87
Predicted Silver in 2024: 6.49
Predicted Bronze in 2024: 7.62
Predicted TotalMedals in 2024: 20.98
```

In [249…

```python
#Top 10 predicted countries with total medals
top_10_pred = df_pred.head(10)

plt.figure(figsize=(15, 8))
bars = plt.bar(top_10_pred['NOC'], top_10_pred['TotalMedals'], color=['gold', 'silver', 'brown'])
plt.xlabel('Country')
plt.ylabel('Predicted Total Medals')
plt.title('Top 10 Predicted Countries with Total Medals in 2024 Olympics')
plt.xticks(rotation='vertical')
plt.tight_layout()

# Add text annotations on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), va='bottom', ha='center', fontsize=8)

plt.show()
```



In [272…

```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving athlete_events.csv to athlete_events.csv
```

In [303…

```
# Load the data
pred = pd.read_csv('athlete_events.csv')
pred.head()

# Create a DataFrame with counts of medals for each year
preddf = pred.groupby(['NOC','Year', 'Medal']).size().reset_index(name='Count')
```

```
# Display the first few rows of the new DataFrame
preddf.head()
```

Out[303]:

	NOC	Year	Medal	Count
0	AFG	2008	Bronze	1
1	AFG	2012	Bronze	1
2	AHO	1988	Silver	1
3	ALG	1984	Bronze	2
4	ALG	1992	Bronze	1

In [282…

```
preddf.drop_duplicates(inplace=True)
preddf.head()
```

Out[282]:

	Year	Medal	Count
0	1896	Bronze	38
1	1896	Gold	62
2	1896	Silver	43
3	1900	Bronze	175
4	1900	Gold	201

In [305…

```
new = df
new = pd.melt(df, id_vars=['Year', 'NOC', 'TotalMedals'], value_vars=['Gold', 'Silver', 'Bronze'],
                  var_name='Medal', value_name='Count')
new.drop(columns=['TotalMedals'], inplace=True)
country_short_forms = {
    'United States of America': 'USA',
    "People's Republic of China": 'CHN',
    'Japan': 'JPN',
    'Great Britain': 'GBR',
    'ROC': 'ROC',
    'Australia': 'AUS',
    'Netherlands': 'NED',
    'France': 'FRA',
    'Germany': 'GER',
    'Italy': 'ITA',
```

```
    'New Zealand': 'NZL',
    'Brazil': 'BRA',
    'Canada': 'CAN',
    'Cuba': 'CUB',
    'Hungary': 'HUN',
    'Republic of Korea': 'KOR',
    'Jamaica': 'JAM',
    'Poland': 'POL',
    'Czech Republic': 'CZE',
    'Norway': 'NOR',
    'Kenya': 'KEN',
    'Croatia': 'CRO',
    'Spain': 'ESP',
    'Switzerland': 'SUI',
    'Denmark': 'DEN',
    'Slovenia': 'SLO',
    'Serbia': 'SRB',
    'Islamic Republic of Iran': 'IRI',
    'Bulgaria': 'BUL',
    'Belgium': 'BEL',
    'Greece': 'GRE',
    'Georgia': 'GEO',
    'Uganda': 'UGA',
    'Turkey': 'TUR',
    'Egypt': 'EGY',
    'Ukraine': 'UKR',
    'Belarus': 'BLR',
    'Portugal': 'POR',
    'Philippines': 'PHI',
    'Indonesia': 'INA',
    'India': 'IND',
    'Hong Kong, China': 'HKG',
    'Austria': 'AUT',
    'Ethiopia': 'ETH',
    'Slovakia': 'SVK'
}
new['NOC'] = new['NOC'].map(country_short_forms)
new.head()
```

Out[305]:

	Year	NOC	Medal	Count
0	2021	USA	Gold	39.0
1	2021	CHN	Gold	38.0
2	2021	JPN	Gold	27.0
3	2021	GBR	Gold	22.0
4	2021	ROC	Gold	20.0

In [306…

```
new.drop_duplicates(inplace=True)
new.head()
```

Out[306]:

	Year	NOC	Medal	Count
0	2021	USA	Gold	39.0
1	2021	CHN	Gold	38.0
2	2021	JPN	Gold	27.0
3	2021	GBR	Gold	22.0
4	2021	ROC	Gold	20.0

In [313…

```
preddf = pd.merge(preddf, new, on=['NOC','Year', 'Medal','Count'], how='left')
preddf.head()
preddf
```

Out[313]:

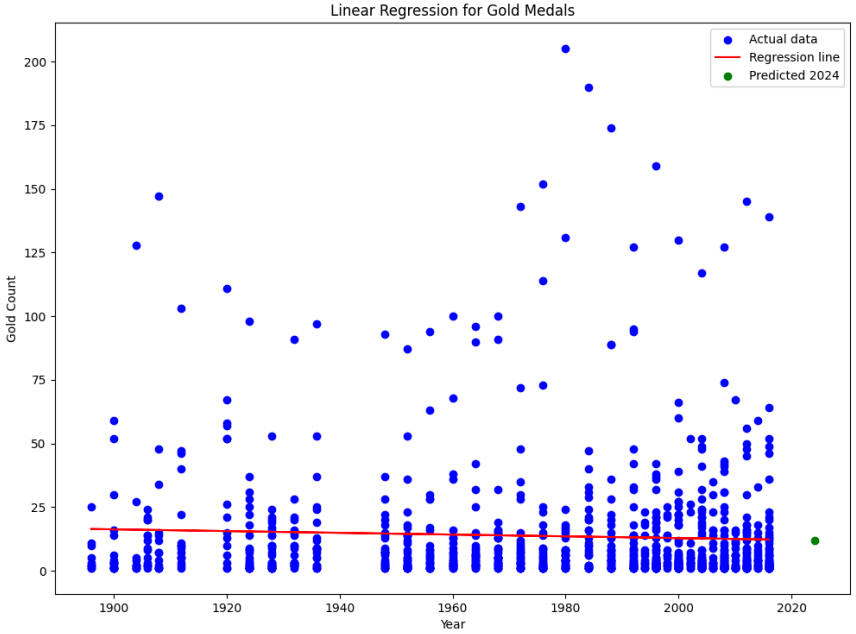
	NOC	Year	Medal	Count
0	AFG	2008	Bronze	1
1	AFG	2012	Bronze	1
2	AHO	1988	Silver	1
3	ALG	1984	Bronze	2
4	ALG	1992	Bronze	1
...	...	...	...	...
3234	ZIM	2004	Bronze	1
3235	ZIM	2004	Gold	1
3236	ZIM	2004	Silver	1
3237	ZIM	2008	Gold	1
3238	ZIM	2008	Silver	3

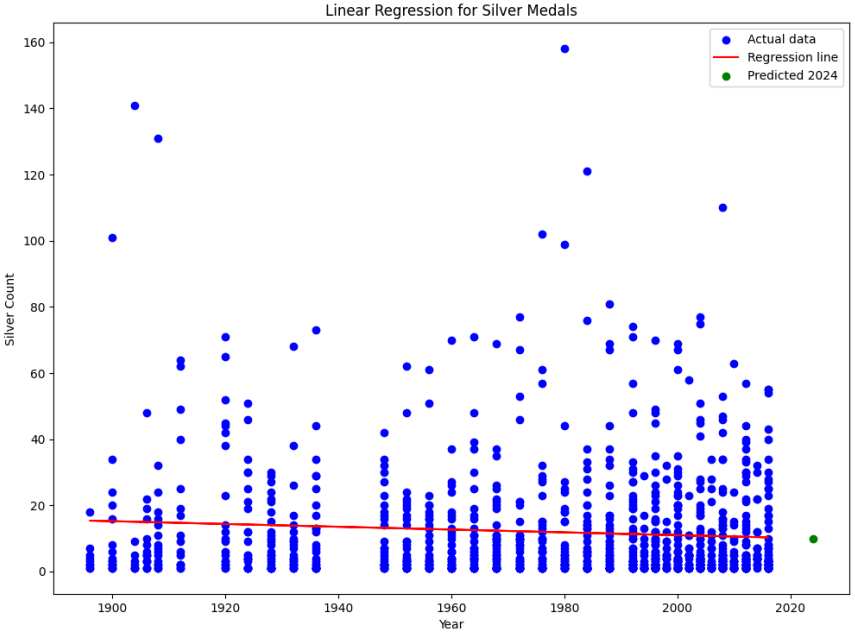
3239 rows × 4 columns

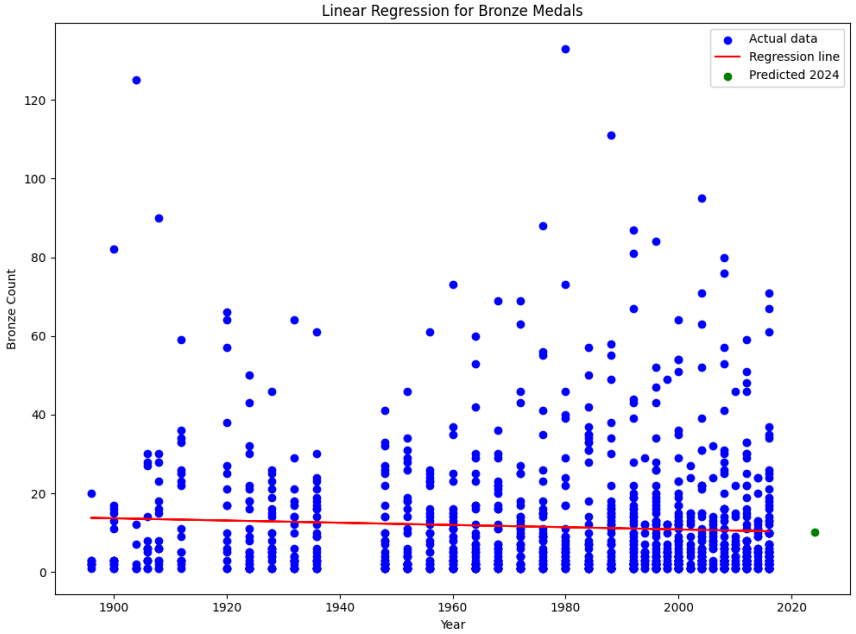
In [308…

```
model = LinearRegression()

# Assume a simple linear increase
future_years = np.array([2024]).reshape(-1, 1)  # Year to predict

# Container for predictions
predictions = {}

# Define medal types
medal_types = ['Gold', 'Silver', 'Bronze']

# Create a linear regression graph for each medal type
plt.figure(figsize=(20, 15))

for i, medal_type in enumerate(medal_types, start=1):
    plt.subplot(2, 2, i)

    # Filter data for the current medal type
    medal_data = preddf[preddf['Medal'] == medal_type]
    X = medal_data['Year'].values.reshape(-1, 1)
```

```
    y = medal_data['Count']

    # Fit the model
    model.fit(X, y)

    # Predict for 2024
    predictions[medal_type] = model.predict(future_years)[0]

    # Plot the actual data points
    plt.scatter(X, y, color='blue', label='Actual data')

    # Plot the regression line
    plt.plot(X, model.predict(X), color='red', label='Regression line')

    # Plot the prediction for 2024
    plt.scatter(future_years, predictions[medal_type], color='green', label='Predicted 2024')

    plt.xlabel('Year')
    plt.ylabel(f'{medal_type} Count')
    plt.title(f'Linear Regression for {medal_type} Medals')
    plt.legend()

plt.tight_layout()
plt.show()

# Display the predictions
for medal_type, prediction in predictions.items():
    print(f"Predicted {medal_type} count in 2024: {prediction:.2f}")
```







Predicted Gold count in 2024: 12.02
Predicted Silver count in 2024: 9.98
Predicted Bronze count in 2024: 10.13

In [317…

```
predictions = {
    'Gold': {},
    'Silver': {},
    'Bronze': {}
}

# Define medal types
medal_types = ['Gold', 'Silver', 'Bronze']

# Loop through each medal type and perform linear regression
for medal_type in medal_types:
    # Filter data for the current medal type
    medal_data = preddf[preddf['Medal'] == medal_type]

    # Prepare X (Years) and y (Counts)
    X = medal_data[['Year']]
    y = medal_data['Count']

    # Initialize the model
    model = LinearRegression()

    # Fit the model
    model.fit(X, y)

    # Predict for 2024
    future_years = np.array([[2024]])
    predictions[medal_type] = model.predict(future_years)[0]

# Aggregate total predicted medals per country
predicted_totals = {}
for country in df['NOC'].unique():
    total_medals = sum(predictions[medal_type] for medal_type in medal_types)
    predicted_totals[country] = total_medals

# Sort countries by predicted total medals in descending order
sorted_countries = sorted(predicted_totals.items(), key=lambda x: x[1], reverse=True)

# Display the sorted list of countries with predicted total medals
print("Predicted Medal Totals for 2024 Olympics:")
for country, total_medals in sorted_countries:
    print(f"{country}: {total_medals:.2f} medals")

# The country with the highest predicted total medals
```

```
predicted_winner = sorted_countries[0][0]
print(f"\nPredicted winner of the 2024 Olympics: {predicted_winner}")
```

Predicted Medal Totals for 2024 Olympics:
United States of America: 32.13 medals
People's Republic of China: 32.13 medals
Japan: 32.13 medals
Great Britain: 32.13 medals
ROC: 32.13 medals
Australia: 32.13 medals
Netherlands: 32.13 medals
France: 32.13 medals
Germany: 32.13 medals
Italy: 32.13 medals
New Zealand: 32.13 medals
Brazil: 32.13 medals
Canada: 32.13 medals
Cuba: 32.13 medals
Hungary: 32.13 medals
Republic of Korea: 32.13 medals
Jamaica: 32.13 medals
Poland: 32.13 medals
Czech Republic: 32.13 medals
Norway: 32.13 medals
Kenya: 32.13 medals
Croatia: 32.13 medals
Spain: 32.13 medals
Switzerland: 32.13 medals
Denmark: 32.13 medals
Slovenia: 32.13 medals
Serbia: 32.13 medals
Islamic Republic of Iran: 32.13 medals
Bulgaria: 32.13 medals
Belgium: 32.13 medals
Greece: 32.13 medals
Georgia: 32.13 medals
Uganda: 32.13 medals
Turkey: 32.13 medals
Egypt: 32.13 medals
Ukraine: 32.13 medals
Belarus: 32.13 medals
Portugal: 32.13 medals
Philippines: 32.13 medals
Indonesia: 32.13 medals
India: 32.13 medals
Hong Kong, China: 32.13 medals
Austria: 32.13 medals
Ethiopia: 32.13 medals

```
Slovakia: 32.13 medals

Predicted winner of the 2024 Olympics: United States of America
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Line
arRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Line
arRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Line
arRegression was fitted with feature names
  warnings.warn(
```

In [319…

```
!jupyter nbconvert --to html
```

```
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the error message in the cell outp
ut (the default behaviour is to abort conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_di
rectory=]
```

```
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_di
rectory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True --TemplateExpor
ter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides',
'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
```

```
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook. To recover
                                    previous default behaviour (outputting to the current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
```

```
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'sc
ript', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve
```

```
                Multiple notebooks can be given at the command line in a couple of
                different ways:

                > jupyter nbconvert notebook*.ipynb
                > jupyter nbconvert notebook1.ipynb notebook2.ipynb

                or you can specify the notebooks list in a config file, containing::

                    c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

                > jupyter nbconvert --config mycfg.py

        To see all available configurables, use `--help-all`.
```

In [ ]:

In [ ]:

In [ ]: