SportsStats Data Analysis Case Study

Exploratory Data Analysis
Created by: Malcomb Brown
Updated: 2023-01-18

Overview

SportsStats is a sports analysis firm partnering with local news and elite personal trainers to provide “interesting” insights to help their partners. Insights could be patterns/trends highlighting certain groups/events/countries, etc. for the purpose of developing a news story or discovering key health insights.

Dataset consists of Olympic data from 1896 - 2016 for both Winter and Summer games and can be found here.

Questions

1. Which countries do better in the Winter Olympics?

- Where are they located?
- This will help understand whether or not there is a relationship between a country's location and its performance.
- We can compare how these countries do in the Summer Games.

2. Which countries do better in the Summer Olympics?

- Where are they located?
- This will help understand whether or not there is a relationship between a country's location and its performance.
- We can compare how these countries do in the Winter Games.

3. Which countries do better based on gender?

- This will show if there is any correlation to the countries overall medal count and how well a gender performs.
- We may discover if there is a pattern between where a country is located and the medal count by gender.

Hypotheses

1. Countries with colder climates will have a higher medal count during the Winter Olympics.

- These places should excel in sports like skiing, hockey, curling, etc.
- The people from these climates are naturally better prepared and acclimated to the weather and events

2. Countries with warmer climates will win more medals during the Summer Olympics.

- These places should excel in track & field, beach volleyball, soccer, baseball, etc.
- The people from these climates are naturally better prepared and acclimated to the weather and events

3. The US will lead all countries in total medal count.

- The lead built during the Summer Games will be too much for anyone to overcome.
- They should be in the top 10 during the Winter Olympics

Import libraries

In [1]:
```
import os                                       # for interacting with the operating syste
import pandas as pd                             # for manipulating data
from sqlalchemy import create_engine            # for creating the connection engine to th
from database import mysql_cnxn                 # database credentials
import plotly.express as px                     # for interactive plotting
from plotly.offline import init_notebook_mode # Plotly notebook mode

init_notebook_mode(connected = True)            # Plotly graphs will persist

px.defaults.template = "presentation"
pd.set_option("min_rows", 20)                   # Sets the minimum rows returned from a qu
```

In [2]:
```
class ProjectSetup():

    """
        Class to setup my Data Analysis Projects.  Automates setting up the project
    directory subfolders and connect to the MySQL database the dataset will be loaded
    to.  Loads the SQL Magic Ipython-sql extension to allow database queries to be done
    in SQL.  Instead of loading the entire table into a dataframe with Pandas and then
    filter, I can filter in the database, saving storage resources. \n

    Database must already exist on the RDBMS Server and the name updated in the
    'database.py' file.

    Parameters
    ============
        database: str
                  Name of the database that will be queried
                  Default: "airbnb"

          create: bool
                  Boolean value that when 'True' will create project subdirectories.
                  Default: False

            conn: str
                  Database connection string.  Currently
                  Default: MySQL
    """

    # Class Variables
    paths = {"raw": "\\Original\\", "prepared": "\\Prepared\\",
            "uploaded": "\\Uploaded\\", "errors": "\\Errors\\",
            "archive": "\\Archive\\"}

    def __init__(self, database: str, create: bool = False):
        self.database = database
        self.conn = mysql_cnxn + database
        self.create = create

        if self.create:
            self.create_paths()
        else:
            self.mount_paths()

        self.db_connection()
```

```
    def __repr__(self):
        return f"{self.database.capitalize()} Data Analysis Case Study Setup Script."

    def mount_paths(self):
        """ Sets up file paths to the project's subfolders."""

        print("="*75)
        print("Getting project directories.....")
        # Get the current path
        self.base_path = os.getcwd()

        # Create a path to the directory for the original csv files.
        self.raw_data_path = f"{self.base_path}{self.paths['raw']}"

        # Create a path to the directory for cleaned datasets
        self.prepared_data_path = f"{self.base_path}{self.paths['prepared']}"

        # Create a path to the directory for files to be loaded in the database
        self.uploaded_data_path = f"{self.base_path}{self.paths['uploaded']}"

        # Create path to the directory to save removed records
        self.errors_data_path = f"{self.base_path}{self.paths['errors']}"

        # Create path to the archive directory
        self.archive_path = f"{self.base_path}{self.paths['archive']}"
        print("All directory paths saved.")
        print("="*75)

    def create_paths(self):
        """ Creates project subdirectories and mounts the paths."""

        self.mount_paths()
        print("="*75)
        print("Creating project folders......")
        dirs = [self.raw_data_path, self.prepared_data_path, self.uploaded_data_path,
                self.errors_data_path, self.archive_path]

        for d in dirs:
            try:
                os.mkdir(d)
            except OSError as error:
                print(error)
                print(f"Project directory not created: {d}")
            print(f"Project directory created: {d}")

        print("Project subfolders setup.")
        print("="*75)

    def db_connection(self):
        """
            Establishes a connection, via SQLAlchemy's 'create_engine' method to the
            database.  Setup notebook to run SQL inline with the '%'.  Currently will
            only work for MySQL and SQLite.

            Adding PosrgreSQL and MS SQL Server....
        """

        print("="*75)
        print("Loading Ipython-sql.....")

        # Using SQL Magic to interact with the MySQL database
        %load_ext sql

        print("Connecting and configuring to the MySQL database.....")

        # Establish the connection to the MySQL database
```

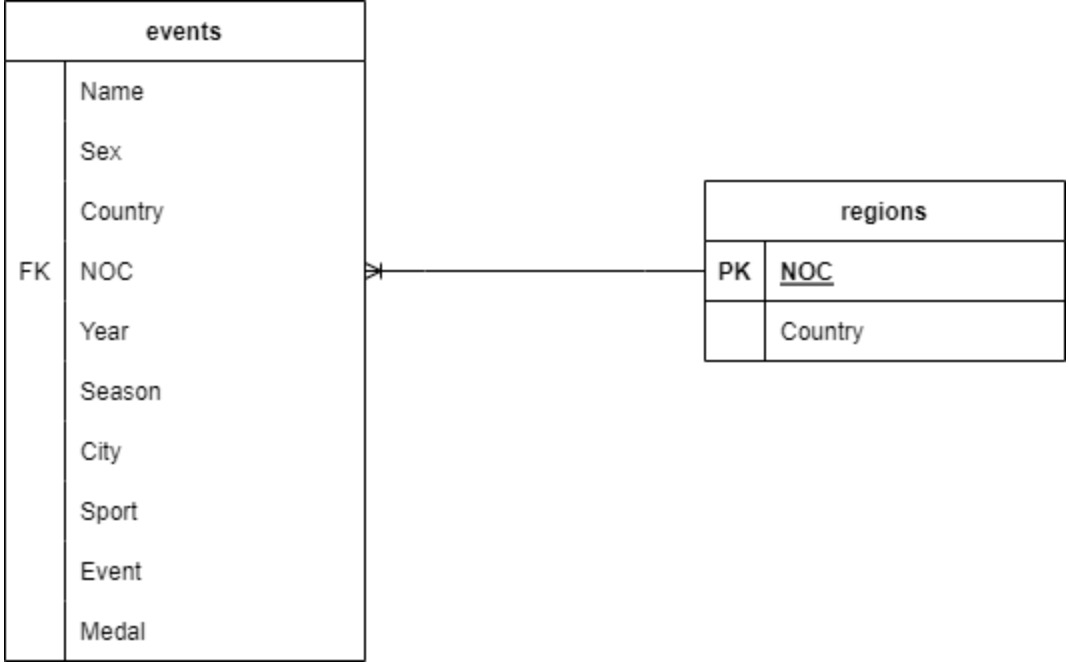
```
        %sql $self.conn

        # Configure output to be returned as a Pandas dataframe.
        %config SqlMagic.autopandas = True

        self.eng = create_engine(self.conn)     # Create the engone to connect to the My
        print("Connection complete!")
        print("="*75)

    def extract_dataset(self, nfile: str, out_file: str = None, save: bool = False):
        """
            Extracts data from a single csv file.  If the file is not in the same direct
            <class_name>, the file path needs to be included.  File can also be extracte
            Prints the metadata of the dataframe and saves to project subfolder.

        Parameters
        ============
               nfile: str
                      Name, path, or URL of the csv file to extract

            out_file: str
                      Name of the file that the extracted data will be stored in the 'Or
                      project folder
                      Default: None
                save: bool
                      Boolean value to denote whether the dataset is to be saved or not
                      Default: False

        """

        print("="*75)
        print("Extracting csv file.....")
        self.raw = pd.read_csv(nfile)

        if save == True:
            if out_file is not None:
                print("Saving original dataset.....")
                self.raw.to_csv(f"{self.raw_data_path}{out_file}.csv")
            else:
                print("No output file name provided.")

        print("Printing metadata......\n")
        print("-"*65)
        print(self.raw.info())
        print("="*75)
        return self.raw
```

In [3]:

```
# Setup the project directories and connect to the MySQL database
project =  ProjectSetup(database="sportsstats")
```

```
===========================================================================
Getting project directories.....
All directory paths saved.
===========================================================================
===========================================================================
Loading Ipython-sql.....
Connecting and configuring to the MySQL database.....
Connection complete!
===========================================================================
```

ERD



Explore

In [4]:

```
%%sql games_df <<
SELECT
    e.Name,
    e.Sex,
    r.Country,
    e.NOC,
    e.Year,
    e.Season,
    e.City,
    e.Sport,
    e.Event,
    e.Medal
FROM
    sportsstats.events e
        LEFT JOIN
    sportsstats.regions r ON e.NOC = r.NOC
WHERE
    r.Country IS NOT NULL;
```

```
 * mysql+pymysql://root:***@localhost/sportsstats
39759 rows affected.
Returning data to local variable games_df
```

In [5]:

```
# Check the metadata
games_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39759 entries, 0 to 39758
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Name     39759 non-null  object
 1   Sex      39759 non-null  object
 2   Country  39759 non-null  object
 3   NOC      39759 non-null  object
 4   Year     39759 non-null  object
 5   Season   39759 non-null  object
 6   City     39759 non-null  object
 7   Sport    39759 non-null  object
```

```
 8   Event     39759 non-null  object
 9   Medal     39759 non-null  object
dtypes: object(10)
memory usage: 3.0+ MB
```

In [6]:
```
# Get Summary Statistics
games_df.describe()
```

Out[6]:

	Name	Sex	Country	NOC	Year	Season	City	Sport	Event	Medal
count	39759	39759	39759	39759	39759	39759	39759	39759	39759	39759
unique	28197	2	136	148	35	2	42	66	756	3
top	Michael Fred Phelps, II	M	USA	USA	2008	Summer	London	Athletics	Football Men's Football	Gold
freq	28	28515	5637	5637	2045	34066	3620	3969	1269	13368

What years does the data cover?

In [7]:
```
print(f"Years range from {games_df.Year.min()} to {games_df.Year.max()}.")
```

```
Years range from 1896 to 2016.
```

In [8]:
```
# Save joined table to the database for analysis
games_df.to_sql("olympics", con=project.eng, if_exists="replace", index=False)
```

Out[8]:
```
39759
```

- Dataset verified to have no null values.
- Covers Olympic Games results from 1896 to 2016.

What countries have won the most medals?

In [9]:
```
%%sql medals_df <<
SELECT
    Country, COUNT(Medal) AS Medals
FROM
    sportsstats.olympics
GROUP BY Country
ORDER BY Medals DESC;
```

```
 * mysql+pymysql://root:***@localhost/sportsstats
136 rows affected.
Returning data to local variable medals_df
```

In [10]:
```
medals_df.head(20)
```

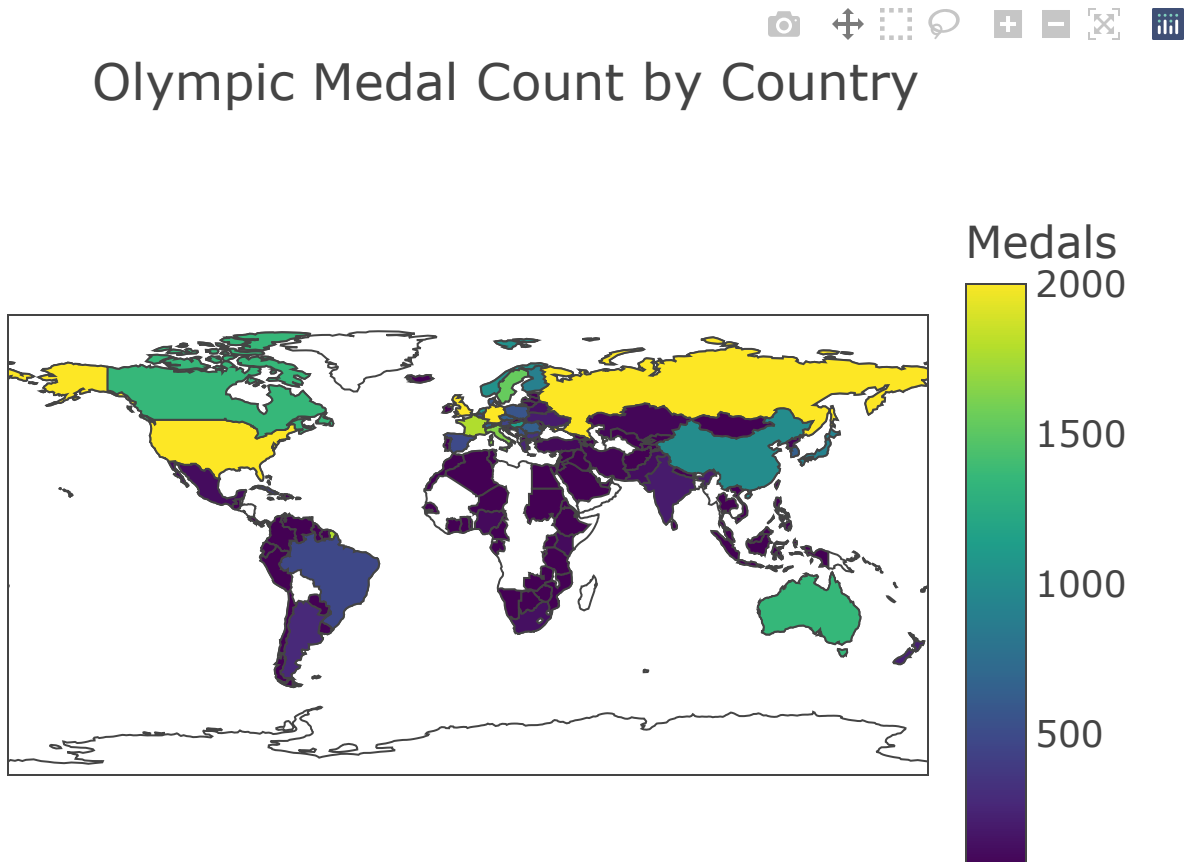
Out[10]:

	Country	Medals
0	USA	5637
1	Russia	3947
2	Germany	3756
3	UK	2067
4	France	1767

5	Italy	1637
6	Sweden	1536
7	Canada	1352
8	Australia	1349
9	Hungary	1135
10	Netherlands	1040
11	Norway	1033
12	China	991
13	Japan	913
14	Finland	900
15	Switzerland	691
16	Romania	653
17	Czech Republic	644
18	South Korea	638
19	Denmark	597

In [11]:

```
# Visualize medal counts by country
fig = px.choropleth(data_frame=medals_df,
                    locations="Country",
                    color="Medals",
                    locationmode="country names",
                    title="Olympic Medal Count by Country",
                    range_color=[50, 2000]
                   )
fig.show(renderer='notebook')
```



- As expected, the United States has won the most medals.
- Russia has won the second most medals followed by Germany

What are the Top 10 Countries by medal count?

In [12]:
```
%%sql top10_df <<
SELECT
    Country, COUNT(Medal) AS Medals
FROM
    sportsstats.olympics
GROUP BY Country
ORDER BY Medals DESC
LIMIT 10;
```

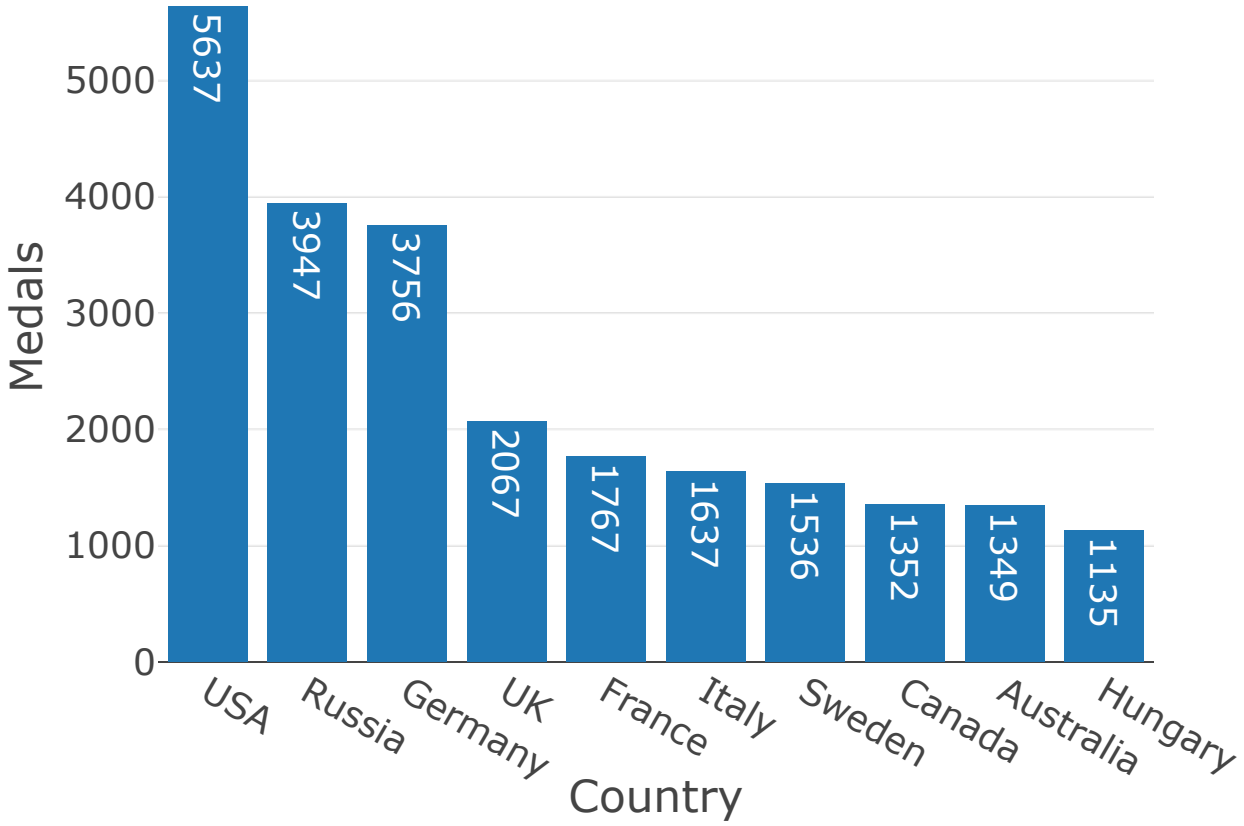
```
 * mysql+pymysql://root:***@localhost/sportsstats
10 rows affected.
Returning data to local variable top10_df
```

In [13]:
```
top10_df
```

Out[13]:

	Country	Medals
0	USA	5637
1	Russia	3947
2	Germany	3756
3	UK	2067
4	France	1767
5	Italy	1637
6	Sweden	1536
7	Canada	1352
8	Australia	1349
9	Hungary	1135

In [14]:
```
# Plot the data using a bar chart
fig = px.bar(data_frame=top10_df,
             x="Country",
             y="Medals",
             title="Top 10 Countries by Medals",
             text="Medals"
            )
fig.show(renderer='notebook')
```



What are the countries' medal counts per season(edition)?

In [15]:
```
%%sql games_df <<
SELECT
    Country,
    Season,
    COUNT(Medal) AS Medals
FROM
    sportsstats.olympics
WHERE
    Country IN (SELECT
                    Country
                FROM
                    (SELECT
                        Country,
                        COUNT(Medal) AS Medals
                    FROM
                        sportsstats.olympics
                    GROUP BY Country
                    ORDER BY Medals DESC
                    LIMIT 10
                    ) as sub_medals)
GROUP BY Country, Season
ORDER BY Medals DESC;
```

```
 * mysql+pymysql://root:***@localhost/sportsstats
20 rows affected.
Returning data to local variable games_df
```

In [16]:
```
games_df.head()
```

Out[16]:

	Country	Season	Medals
0	USA	Summer	5002

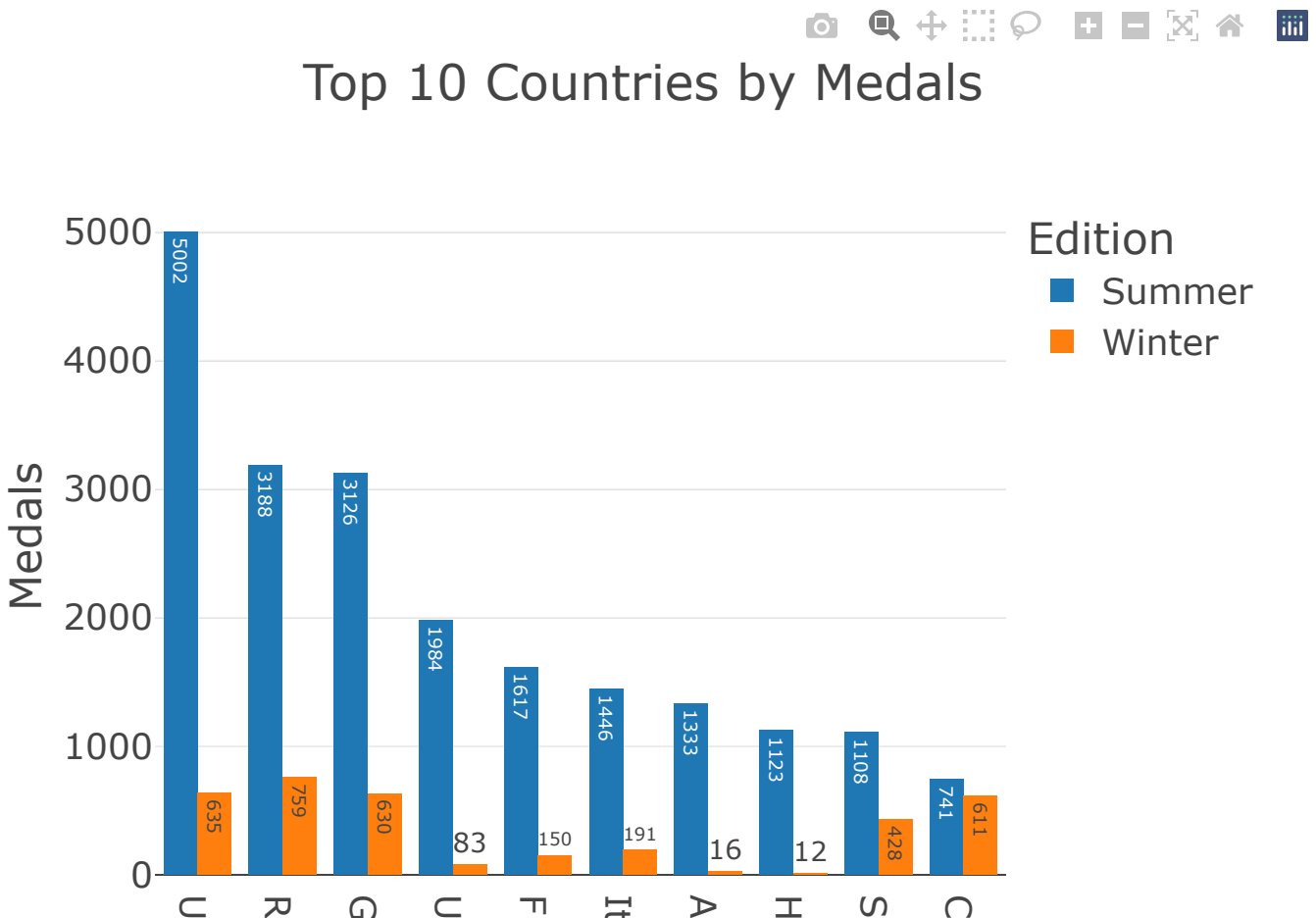
1	Russia	Summer	3188
2	Germany	Summer	3126
3	UK	Summer	1984
4	France	Summer	1617

In [17]:
```
games_df.tail()
```

Out[17]:

	Country	Season	Medals
15	Italy	Winter	191
16	France	Winter	150
17	UK	Winter	83
18	Australia	Winter	16
19	Hungary	Winter	12

In [18]:
```
# Visualize the medal count by country and season
fig = px.bar(data_frame=games_df,
             x="Country",
             y="Medals",
             color="Season",
             title="Top 10 Countries by Medals",
             labels={"Season": "Edition"},
             text="Medals",
             barmode="group",
            )
fig.show(renderer='notebook')
```



Get the medal count for each country for the Summer Olympics

In [19]:
```
%%sql summer_medals <<
SELECT
    Country, COUNT(Medal) AS Medals
FROM
    sportsstats.olympics
WHERE Season = 'Summer'
GROUP BY Country
ORDER BY Medals DESC;
```

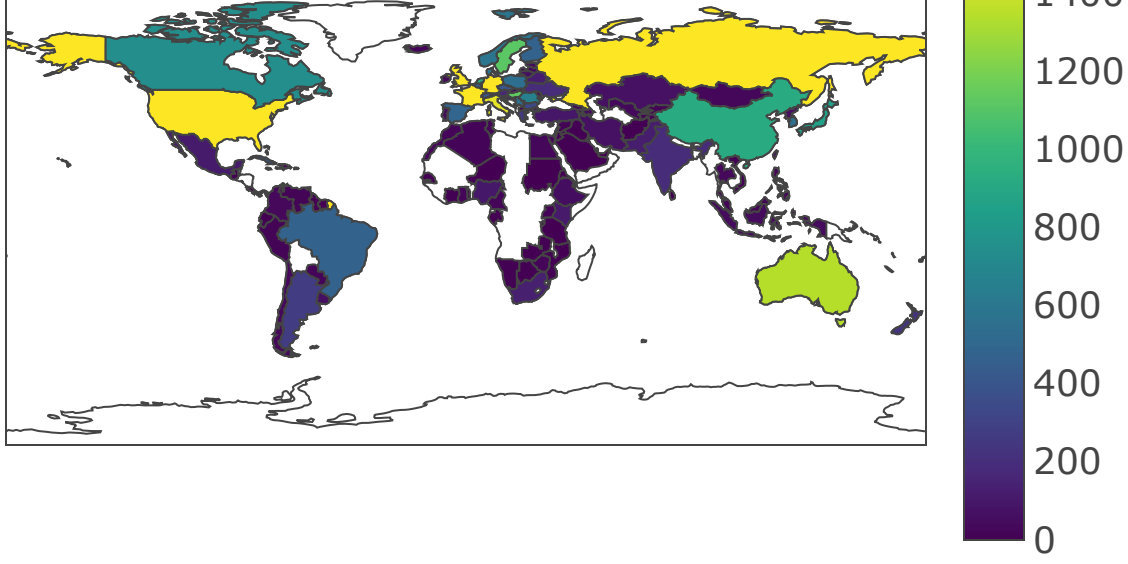
```
 * mysql+pymysql://root:***@localhost/sportsstats
134 rows affected.
Returning data to local variable summer_medals
```

In [20]:
```
summer_medals.head(10)
```

Out[20]:

	Country	Medals
0	USA	5002
1	Russia	3188
2	Germany	3126
3	UK	1984
4	France	1617
5	Italy	1446
6	Australia	1333
7	Hungary	1123
8	Sweden	1108
9	Netherlands	918

In [21]:
```
# Visualize Countries by Summer Olympics Medal Count
fig = px.choropleth(data_frame=summer_medals,
                    locations="Country",
                    color="Medals",
                    locationmode="country names",
                    title="Summer Olympics Medal Count by Country",
                    range_color=[0, 1500]
                   )
fig.show(renderer='notebook')
```



Get the medal count for the Winter Olympics by Country

In [22]:
```
%%sql winter_medals <<
SELECT
    Country, COUNT(Medal) AS Medals
FROM
    sportsstats.olympics
WHERE Season = 'Winter'
GROUP BY Country
ORDER BY Medals DESC;
```

```
 * mysql+pymysql://root:***@localhost/sportsstats
41 rows affected.
Returning data to local variable winter_medals
```

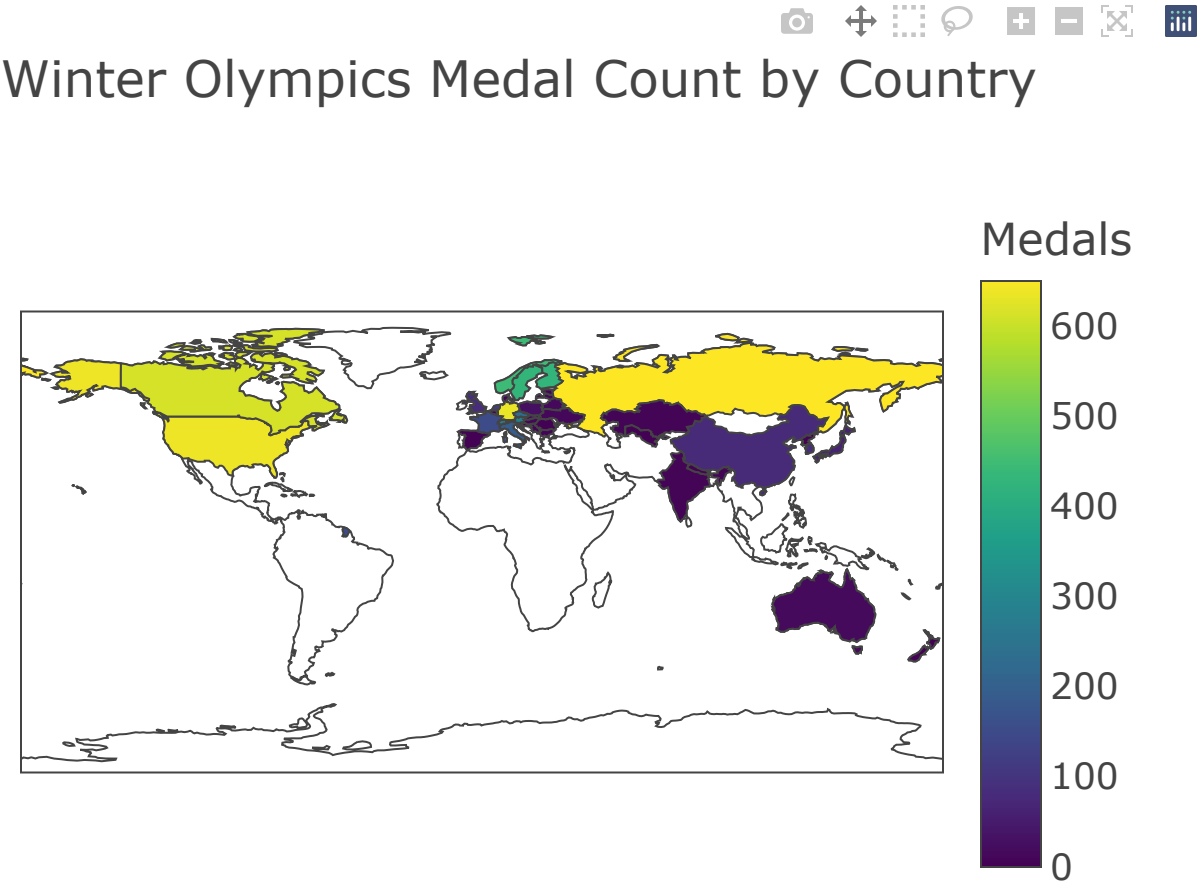
In [23]:
```
winter_medals.head(10)
```

Out[23]:

	Country	Medals
0	Russia	759
1	USA	635
2	Germany	630
3	Canada	611
4	Norway	443
5	Sweden	428
6	Finland	426
7	Austria	280
8	Switzerland	275
9	Czech Republic	231

In [24]:
```
# Visualize Countries by Winter Olympics  Medal Count
fig = px.choropleth(data_frame=winter_medals,
```

```
                    locations="Country",
                    color="Medals",
                    locationmode="country names",
                    title="Winter Olympics Medal Count by Country",
                    range_color=[0, 650]
                   )
fig.show(renderer='notebook')
```



In [25]:

```
# Delete dataframes no longer needed to save memory
del summer_medals
del winter_medals
del top10_df
del medals_df
del games_df
```

- Less countries have earned medals in the Winter Olympics compared to the Summer Olympics.
- Countries in South America and Africa have only won medals during the Summer Olympics.
- USA, Russia, and Germany have won the most medals in both editions.

Aggregate and Rank the Top 50 Countries by:

- Total medal count
- Gender
- Olympic edition

Aggregate Top 50 Countries

In [26]:

```
%%sql
SELECT
    Country,
    SUM(CASE WHEN Season = "Summer" THEN 1 ELSE 0 END) AS Summer,
    SUM(CASE WHEN Season = "Winter" THEN 1 ELSE 0 END) AS Winter,
    SUM(CASE WHEN Sex = "M" THEN 1 ELSE 0 END) AS Men,
    SuM(CASE WHEN Sex = "F" THEN 1 ELSE 0 END) AS Women,
    COUNT(Medal) AS Total
FROM
   sportsstats.olympics
GROUP BY Country
ORDER BY Total DESC
LIMIT 50;
```

```
 * mysql+pymysql://root:***@localhost/sportsstats
50 rows affected.
```

Out[26]:

	Country	Summer	Winter	Men	Women	Total
0	USA	5002	635	3832	1805	5637
1	Russia	3188	759	2590	1357	3947
2	Germany	3126	630	2510	1246	3756
3	UK	1984	83	1672	395	2067
4	France	1617	150	1539	228	1767
5	Italy	1446	191	1418	219	1637
6	Sweden	1108	428	1316	220	1536
7	Canada	741	611	832	520	1352
8	Australia	1333	16	800	549	1349
9	Hungary	1123	12	868	267	1135
10	Netherlands	918	122	608	432	1040
11	Norway	590	443	803	230	1033
12	China	913	78	337	654	991
13	Japan	850	63	612	301	913
14	Finland	474	426	781	119	900
15	Switzerland	416	275	589	102	691
16	Romania	651	2	298	355	653
17	Czech Republic	413	231	513	131	644
18	South Korea	552	86	343	295	638
19	Denmark	592	5	490	107	597
20	Poland	538	27	439	126	565
21	Serbia	532	7	447	92	539
22	Spain	487	2	356	133	489
23	Brazil	475	0	340	135	475
24	Belgium	455	13	445	23	468
25	Austria	170	280	353	97	450
26	Cuba	409	0	316	93	409

27	Bulgaria	336	6	214	128	342
28	Argentina	274	0	201	73	274
29	Greece	255	0	208	47	255
30	New Zealand	227	1	171	57	228
31	Ukraine	188	11	94	105	199
32	India	188	7	190	5	195
33	Jamaica	157	0	71	86	157
34	Croatia	138	11	134	15	149
35	Belarus	124	15	62	77	139
36	South Africa	131	0	107	24	131
37	Pakistan	121	0	121	0	121
38	Mexico	110	0	93	17	110
39	Kenya	106	0	85	21	106
40	Nigeria	99	0	80	19	99
41	Turkey	95	0	85	10	95
42	Kazakhstan	70	7	56	21	77
43	Iran	68	0	67	1	68
44	North Korea	65	2	33	34	67
45	Uruguay	63	0	63	0	63
46	Lithuania	61	0	51	10	61
47	Ethiopia	53	0	30	23	53
48	Estonia	43	7	46	4	50
49	Taiwan	49	0	30	19	49

Rank each feature

In [27]:

```
%%sql ranks_df <<
WITH top50 AS (
SELECT
    Country,
    SUM(CASE WHEN Season = "Summer" THEN 1 ELSE 0 END) AS Summer,
    SUM(CASE WHEN Season = "Winter" THEN 1 ELSE 0 END) AS Winter,
    SUM(CASE WHEN Sex = "M" THEN 1 ELSE 0 END) AS Men,
    SuM(CASE WHEN Sex = "F" THEN 1 ELSE 0 END) AS Women,
    COUNT(Medal) AS Total
FROM
   sportsstats.olympics
GROUP BY Country
ORDER BY Total DESC
LIMIT 50)
SELECT
    Country,
    RANK() OVER (ORDER BY Summer DESC) AS Summer,
    RANK() OVER (ORDER BY Winter DESC) AS Winter,
    RANK() OVER (ORDER BY Men DESC) AS Men,
    RANK() OVER (ORDER BY Women DESC) AS Women,
    RANK() OVER (ORDER BY Total DESC) AS Total
```

```
FROM
    top50;
```

```
 * mysql+pymysql://root:***@localhost/sportsstats
50 rows affected.
Returning data to local variable ranks_df
```

In [28]:
```
ranks_df
```

Out[28]:

	Country	Summer	Winter	Men	Women	Total
0	USA	1	2	1	1	1
1	Russia	2	1	2	2	2
2	Germany	3	3	3	3	3
3	UK	4	15	4	8	4
4	France	5	12	5	14	5
5	Italy	6	11	6	16	6
6	Sweden	9	6	7	15	7
7	Canada	13	4	9	6	8
8	Australia	7	19	11	5	9
9	Hungary	8	22	8	12	10
10	Netherlands	10	13	14	7	11
11	Norway	16	5	10	13	12
12	China	11	16	25	4	13
13	Japan	12	17	13	10	14
14	Finland	22	7	12	22	15
15	Switzerland	24	9	15	25	16
16	Romania	14	31	27	9	17
17	Czech Republic	25	10	16	19	18
18	South Korea	17	14	23	11	19
19	Denmark	15	30	17	23	20
20	Poland	18	18	20	21	21
21	Serbia	19	25	18	28	22
22	Spain	20	31	21	18	23
23	Brazil	21	35	24	17	24
24	Belgium	23	21	19	36	25
25	Austria	33	8	22	26	26
26	Cuba	26	35	26	27	27
27	Bulgaria	27	29	28	20	28
28	Argentina	28	35	30	31	29
29	Greece	29	35	29	33	30
30	New Zealand	30	34	32	32	31

31	Ukraine	31	23	36	24	32
32	India	31	25	31	46	33
33	Jamaica	34	35	41	29	34
34	Croatia	35	23	33	43	35
35	Belarus	37	20	44	30	36
36	South Africa	36	35	35	35	37
37	Pakistan	38	35	34	49	38
38	Mexico	39	35	37	42	39
39	Kenya	40	35	38	38	40
40	Nigeria	41	35	40	40	41
41	Turkey	42	35	38	44	42
42	Kazakhstan	43	25	45	38	43
43	Iran	44	35	42	48	44
44	North Korea	45	31	48	34	45
45	Uruguay	46	35	43	49	46
46	Lithuania	47	35	46	44	47
47	Ethiopia	48	35	49	36	48
48	Estonia	50	25	47	47	49
49	Taiwan	49	35	49	40	50

In [29]:
```
# Set 'Country' as the index
ranks_df.set_index("Country", drop=True, inplace=True)
```

In [30]:
```
ranks_df.head()
```

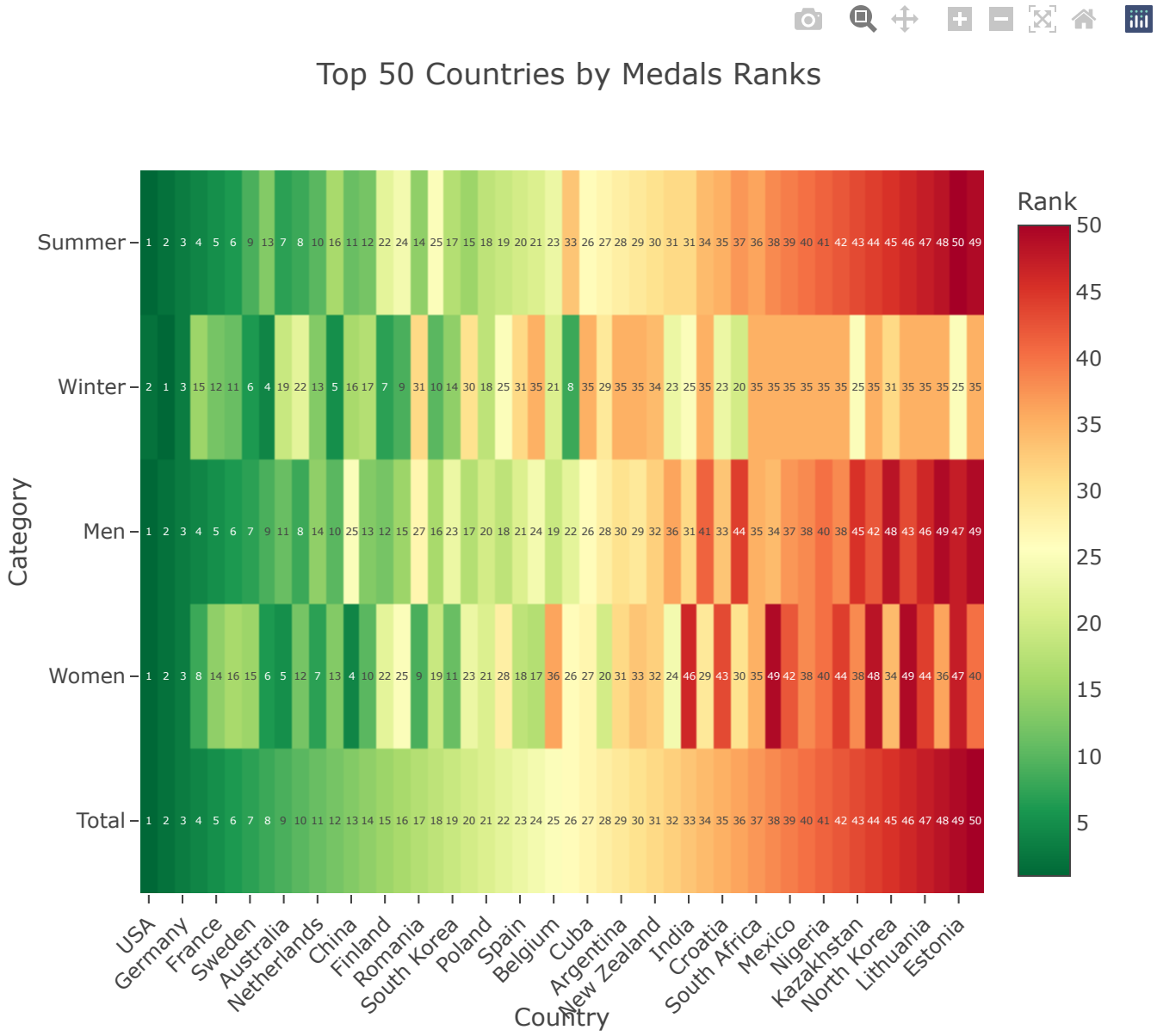
Out[30]:

Country	Summer	Winter	Men	Women	Total
USA	1	2	1	1	1
Russia	2	1	2	2	2
Germany	3	3	3	3	3
UK	4	15	4	8	4
France	5	12	5	14	5

Visualize the relationship between the ranks

In [31]:
```
# Plot using a heatmap
fig = px.imshow(ranks_df.T,
                title="Top 50 Countries by Medals Ranks",
                text_auto=True,
                labels={"x": "Country", "y": "Category", "color": "Rank"},
                aspect="auto",
                color_continuous_scale="RdYlGn_r",
```

```
                  height=600)
fig.update_layout(font_size=12)
fig.update_xaxes(tickangle=315)
fig.show(renderer='notebook')
```



In [32]:

```
del ranks_df
```