Preliminary data analysis

December 11, 2024

1 File structure

This project assumes the following initial file structure:

```
.\main
|   Preliminary data analysis.ipynb
|   ...
|
\---data
    \---olympic_sw_1896_2022
            olympic_athletes.csv
            olympic_hosts.csv
            olympic_medals.csv
            olympic_results.csv
            olympic_results.pkl
            ...
```

2 Modules

2.1 Imports

[ ]:
```
import os
import types
from functools import partial

import geopandas as gpd
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pkg_resources
import pycountry
from geopy.geocoders import Nominatim
from tqdm.notebook import tqdm
```

[ ]:
```
tqdm.pandas()
```

2.2 Versions

The current module versions in use are as follows:

[ ]:
```
def get_imports():
    for name, val in globals().items():
        if isinstance(val, types.ModuleType):
            name = val.__name__.split(".")[0]

        elif isinstance(val, type):
            name = val.__module__.split(".")[0]

        poorly_named_packages = {"PIL": "pillow", "sklearn": "scikit-learn"}
        if name in poorly_named_packages.keys():
            name = poorly_named_packages[name]

        yield name


imports = list(set(get_imports()))

requirements = []
for m in pkg_resources.working_set:
    if m.project_name in imports and m.project_name != "pip":
        requirements.append((m.project_name, m.version))

pd.DataFrame(requirements, columns=["Module", "Version"])
```

[ ]:

	Module	Version
0	geopandas	0.14.2
1	geopy	2.4.1
2	matplotlib	3.9.2
3	numpy	1.26.4
4	pandas	2.2.2
5	pycountry	24.6.1
6	tqdm	4.66.5

3 Global variables

The following global variables are used throughout the entirety of the methodology section.

[ ]:
```
DATA_PATH = "./data"
OLYMPIC_DATA_PATH = os.path.join(DATA_PATH, "olympic_sw_1896_2022")
CLEAN_DATA_PATH = os.path.join(DATA_PATH, "clean")
```

4 Preliminary data analysis

In this section, we embark on a foundational exploration of our dataset to glean essential insights that underpin our research objectives. We outline the dataset's key attributes, including its size, composition, and structure, while examining descriptive statistics to uncover central tendencies and distributions.

4.1 Data import

[ ]:
```
athletes_data = pd.read_csv(os.path.join(OLYMPIC_DATA_PATH, "olympic_athletes.
↪csv"))
medals_data = pd.read_csv(os.path.join(OLYMPIC_DATA_PATH, "olympic_medals.csv"))
hosts_data = pd.read_csv(os.path.join(OLYMPIC_DATA_PATH, "olympic_hosts.csv"))
results_data = pd.read_csv(os.path.join(OLYMPIC_DATA_PATH, "olympic_results.
↪csv"))
```

4.2 Athletes

[ ]:
```
DROPPED_ATHLETES = 0
```

[ ]:
```
athletes_data.sample(10)
```

[ ]:
```
                                          athlete_url  \
27190  https://olympics.com/en/athletes/antonios-boug…
52111  https://olympics.com/en/athletes/giancarlo-mor…
65585  https://olympics.com/en/athletes/ernesto-ambro…
51734  https://olympics.com/en/athletes/renate-garisc…
48099      https://olympics.com/en/athletes/tore-milsett
34177  https://olympics.com/en/athletes/reuven-hadinatov
62678     https://olympics.com/en/athletes/tadao-okayama
18894     https://olympics.com/en/athletes/simon-munyutu
28050   https://olympics.com/en/athletes/jorge-gutierrez
42911         https://olympics.com/en/athletes/odd-sorli

                  athlete_full_name  games_participations  \
27190            Antonios BOUGIOURIS                     1
52111              Giancarlo MORRESI                     1
65585              Ernesto AMBROSINI                     2
51734  Renate GARISCH-CULMBERGER-BOY                     3
48099                   Tore MILSETT                     2
34177               Reuven HADINATOV                     1
62678                  Tadao OKAYAMA                     1
18894                  Simon MUNYUTU                     1
28050                Jorge GUTIERREZ                     1
42911                     Odd SÖRLI                     3

                     first_game  athlete_year_birth    athlete_medals  \
27190                Sydney 2000              1974.0               NaN
```

```
52111             Mexico City 1968              1944.0               NaN
65585                 Antwerp 1920              1894.0  \n\n\n1\n\nB\n\n
51734                    Rome 1960              1939.0  \n\n\n1\n\nS\n\n
48099             Mexico City 1968              1944.0               NaN
34177               Barcelona 1992              1969.0               NaN
62678  Garmisch-Partenkirchen 1936              1913.0               NaN
18894                 Beijing 2008              1977.0               NaN
28050                  Sydney 2000              1975.0  \n\n\n1\n\nG\n\n
42911                Innsbruck 1976              1954.0               NaN

                                                    bio
27190                                               NaN
52111                                               NaN
65585  \n\n\nErnesto Abrosini started running in 1912…
51734  \n\n\nRenate Boy was born in East Prussia as G…
48099                                               NaN
34177                                               NaN
62678                                               NaN
18894  \n\n\nPersonal Best: Mar - 2-09:24 (2008).\n\n…
28050                                               NaN
42911                                               NaN
```

[ ]:
```
athletes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75904 entries, 0 to 75903
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   athlete_url           75904 non-null  object
 1   athlete_full_name     75904 non-null  object
 2   games_participations  75904 non-null  int64
 3   first_game            75882 non-null  object
 4   athlete_year_birth    73448 non-null  float64
 5   athlete_medals        15352 non-null  object
 6   bio                   22842 non-null  object
dtypes: float64(1), int64(1), object(5)
memory usage: 4.1+ MB
```

[ ]:
```
athletes_data.describe(include=np.number)
```

[ ]:
```
       games_participations  athlete_year_birth
count          75904.000000        73448.000000
mean               1.535874         1961.619377
std                0.854563           28.129576
min                0.000000         1836.000000
25%                1.000000         1946.000000
50%                1.000000         1968.000000
```

```
75%                2.000000         1983.000000
max               10.000000         2009.000000
```

[ ]:
```
athletes_data.describe(include=object)
```

[ ]:
```
                                          athlete_url athlete_full_name  \
count                                           75904             75904
unique                                          75900             75480
top     https://olympics.com/en/athletes/mariana-pajon       Ivan IVANOV
freq                                                2                 4

       first_game    athlete_medals  \
count       75882             15352
unique         53               170
top      Rio 2016  \n\n\n1\n\nB\n\n
freq         4111              4209

                                                  bio
count                                           22842
unique                                          22530
top     \n\n\nPersonal Best: Mar - unknown.\n\n\n\n\n\n
freq                                              110
```

The following code converts the `athlete_year_birth` column to the `Int64` data type, which is a pandas-specific integer type that supports nullable values. This ensures that the column contains integer values while allowing for `NaN` entries, providing better consistency for numerical operations involving birth years.

[ ]:
```
athletes_data["athlete_year_birth"] = athletes_data["athlete_year_birth"].
 ↪astype("Int64")
```

The following code extracts information from the `first_game` column into two new columns, `first_game_host_city` and `first_game_year`, using a regular expression. The `.str.extract()` method is used with the regex pattern `r"^(.+?)\s(\d+?)$"` to capture the host city and the year separately. The pattern `^(.+?)\s(\d+?)$` matches the city name (which can contain spaces) followed by a space and a four-digit year, ensuring that both parts are properly captured. This allows us to split the `first_game` data into distinct columns for easier analysis.

[ ]:
```
athletes_data[["first_game_city", "first_game_year"]] = athletes_data[
    "first_game"
].str.extract(r"^(.+?)\s(\d+?)$")

athletes_data["first_game_year"] = athletes_data["first_game_year"].
 ↪astype("Int64")
```

The following code extracts medal information from the `athlete_medals` column, converts it into structured medal counts, and stores these in new columns. First, the regex `.str.findall()` is used to extract medal pairs (number and type) from each string into the `medal_tuples` column. Next, a helper function `count_medals()` is defined to count specific medal types (`G`, `S`, `B`) in each

list of tuples. This function is applied to create `bronze_count`, `silver_count`, and `gold_count` columns, representing the count of each medal type. Finally, a `total_medals` column is created by calculating the sum of the three newly created columns.

[ ]:
```
medal_tuples = athletes_data["athlete_medals"].str.
↪findall(r"\n*(\d+)\n*([GSB])\n*")
```

[ ]:
```
def count_medals(medal_list, medal_type):
    if not isinstance(medal_list, list):
        return 0
    return sum(int(count) for count, medal in medal_list if medal == medal_type)
```

[ ]:
```
athletes_data["bronze_count"] = medal_tuples.apply(lambda x: count_medals(x,␣
↪"B"))
athletes_data["silver_count"] = medal_tuples.apply(lambda x: count_medals(x,␣
↪"S"))
athletes_data["gold_count"] = medal_tuples.apply(lambda x: count_medals(x, "G"))

athletes_data["total_medals"] = athletes_data[
    ["bronze_count", "silver_count", "gold_count"]
].sum(axis=1)
```

Here, we remove information that is redundant, or not pertinent for the present analysis.

[ ]:
```
athletes_data = athletes_data.drop(["first_game", "bio", "athlete_medals"],␣
↪axis=1)
```

We add some extra columns based on the location.

[ ]:
```
geolocator = Nominatim(user_agent="city_to_country_mapper")
geocode = partial(geolocator.geocode, language="en")

city_country_cache = {}
country_code_cache_a2 = {}
country_code_cache_a3 = {}

country_code_cache_a3["Turkey"] = "TUR"
country_code_cache_a3["Chinese Taipei"] = "TWN"

def get_country(city):
    if pd.isna(city):
        return np.nan
    if city in city_country_cache:
        return city_country_cache[city]
    try:
        location = geocode(city)
        if location:
            country = location.address.split(",")[-1].strip()
```

```
            city_country_cache[city] = country
            return country
    except Exception as e:
        print(f"Error geocoding city '{city}': {e}")
    city_country_cache[city] = np.nan
    return np.nan


def get_country_code(country_name, alpha=2):
    if pd.isna(country_name):
        return np.nan
    cache = country_code_cache_a2 if alpha == 2 else country_code_cache_a3
    if country_name in cache:
        return cache[country_name]
    try:
        code = pycountry.countries.search_fuzzy(country_name)
        if code:
            iso_code = code[0].alpha_2 if alpha == 2 else code[0].alpha_3
            cache[country_name] = iso_code
            return iso_code
    except Exception as e:
        print(f"Error fetching country code for '{country_name}': {e}")
    cache[country_name] = np.nan
    return np.nan
```

[ ]:
```
cities = pd.Series(athletes_data["first_game_city"].unique())
countries = cities.progress_apply(get_country)
country_codes_a2 = countries.map(lambda x: get_country_code(x, alpha=2))
country_codes_a3 = countries.map(lambda x: get_country_code(x, alpha=3))

city_to_country = dict(zip(cities, countries))
city_to_country_code_a2 = dict(zip(cities, country_codes_a2))
city_to_country_code_a3 = dict(zip(cities, country_codes_a3))

athletes_data.insert(
    athletes_data.columns.get_loc("first_game_city") + 1,
    "first_game_country",
    athletes_data["first_game_city"].map(city_to_country),
)
athletes_data.insert(
    athletes_data.columns.get_loc("first_game_city") + 2,
    "country_code",
    athletes_data["first_game_city"].map(city_to_country_code_a2),
)
athletes_data.insert(
    athletes_data.columns.get_loc("first_game_city") + 3,
    "country_3_letter_code",
```

```
    athletes_data["first_game_city"].map(city_to_country_code_a3),
)
```

```
  0%|          | 0/44 [00:00<?, ?it/s]
```

Finally, we visualize again, some preliminary informations of the entries of the resulting data frame.

[ ]:
```
athletes_data.sample(10)
```

[ ]:
```
                                          athlete_url  \
10372    https://olympics.com/en/athletes/walid-mohamed
60128   https://olympics.com/en/athletes/hermann-baumann
59763   https://olympics.com/en/athletes/stefan-kovalcik
3238     https://olympics.com/en/athletes/jonathan-groth
2403        https://olympics.com/en/athletes/emily-lewis
39767     https://olympics.com/en/athletes/rick-mewborn
877                 https://olympics.com/en/athletes/gow
15527    https://olympics.com/en/athletes/merab-turkadze
12721       https://olympics.com/en/athletes/qiuhong-liu
34760  https://olympics.com/en/athletes/lisa-boscarin…

          athlete_full_name  games_participations  athlete_year_birth  \
10372         Walid MOHAMED                     0                1961
60128       Hermann BAUMANN                     1                1921
59763       Stefan KOVALCIK                     2                1921
3238         Jonathan GROTH                     2                1992
2403            Emily LEWIS                     1                1993
39767          Rick MEWBORN                     1                1965
877            Christian GOW                     2                1993
15527        Merab TURKADZE                     1                1988
12721           Qiuhong LIU                     1                1988
34760  Lisa BOSCARINO PAGAN                     2                1961

      first_game_city first_game_country country_code country_3_letter_code  \
10372             NaN                NaN          NaN                   NaN
60128          London     United Kingdom           GB                   GBR
59763      St. Moritz        Switzerland           CH                   CHE
3238              Rio             Brazil           BR                   BRA
2403            Tokyo              Japan           JP                   JPN
39767         Calgary             Canada           CA                   CAN
877       PyeongChang        South Korea           KR                   KOR
15527          London     United Kingdom           GB                   GBR
12721           Sochi             Russia           RU                   RUS
34760           Seoul        South Korea           KR                   KOR

       first_game_year  bronze_count  silver_count  gold_count  total_medals
10372             <NA>             0             0           0             0
60128             1948             1             0           0             1
```

59763	1948	0	0	0	0
3238	2016	0	0	0	0
2403	2020	0	0	0	0
39767	1988	0	0	0	0
877	2018	0	0	0	0
15527	2012	0	0	0	0
12721	2014	0	0	0	0
34760	1988	0	0	0	0

[ ]:
```
athletes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75904 entries, 0 to 75903
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   athlete_url            75904 non-null  object
 1   athlete_full_name      75904 non-null  object
 2   games_participations   75904 non-null  int64
 3   athlete_year_birth     73448 non-null  Int64
 4   first_game_city        75882 non-null  object
 5   first_game_country     75882 non-null  object
 6   country_code           75882 non-null  object
 7   country_3_letter_code  75882 non-null  object
 8   first_game_year        75882 non-null  Int64
 9   bronze_count           75904 non-null  int64
 10  silver_count           75904 non-null  int64
 11  gold_count             75904 non-null  int64
 12  total_medals           75904 non-null  int64
dtypes: Int64(2), int64(5), object(6)
memory usage: 7.7+ MB
```

[ ]:
```
athletes_data.describe(include=np.number)
```

[ ]:

	games_participations	athlete_year_birth	first_game_year	\
count	75904.000000	73448.0	75882.0	
mean	1.535874	1961.619377	1984.765742	
std	0.854563	28.129576	28.683834	
min	0.000000	1836.0	1896.0	
25%	1.000000	1946.0	1968.0	
50%	1.000000	1968.0	1992.0	
75%	2.000000	1983.0	2008.0	
max	10.000000	2009.0	2022.0	

	bronze_count	silver_count	gold_count	total_medals
count	75904.000000	75904.000000	75904.000000	75904.000000
mean	0.109889	0.105844	0.109994	0.325727
std	0.369809	0.380472	0.458560	0.851044

```
min        0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000
max        6.000000      6.000000     23.000000     28.000000
```

[ ]:
```
athletes_data.describe(include=object)
```

[ ]:
```
                                          athlete_url athlete_full_name  \
count                                           75904             75904
unique                                          75900             75480
top     https://olympics.com/en/athletes/mariana-pajon       Ivan IVANOV
freq                                                2                 4

       first_game_city first_game_country country_code country_3_letter_code
count            75882              75882        75882                  75882
unique              43                 23           23                     23
top             London      United States           US                    USA
freq              6113               9812         9812                   9812
```

Inconsistencies The following code drops rows with athletes who have no games_participations.

[ ]:
```
DROPPED_ATHLETES += np.sum(athletes_data["games_participations"] == 0)

print(f"Total athletes dropped: {DROPPED_ATHLETES}")
```

```
Total athletes dropped: 22
```

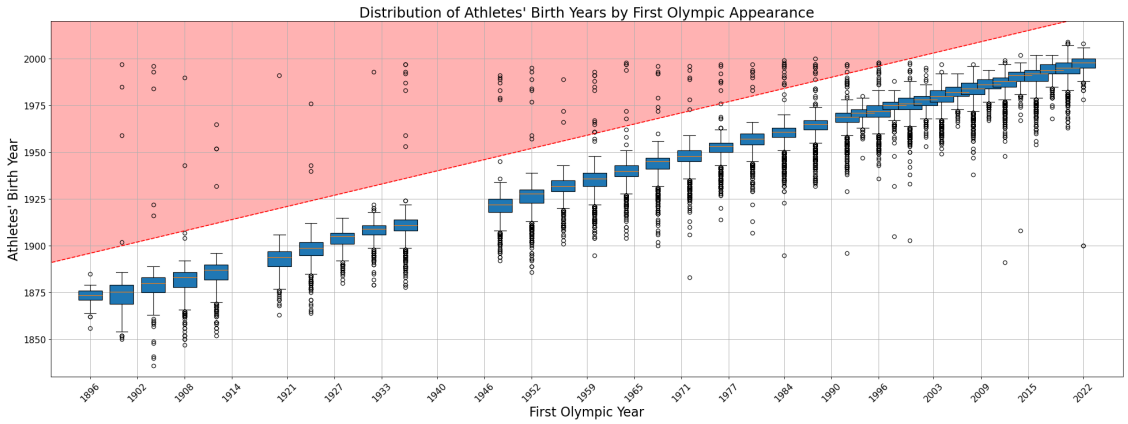
[ ]:
```
athletes_data = athletes_data[athletes_data["games_participations"] != 0]
```

Now, let's first take a look at the time stamps of the olympics as compared to athletes' ages.

[ ]:
```
game_years = pd.Series(
    pd.unique(athletes_data["first_game_year"].dropna())
).sort_values(ascending=True)

box_data_birth = []
box_data_age = []
for game_year in game_years:
    year_births = athletes_data[athletes_data["first_game_year"] == game_year][
        "athlete_year_birth"
    ].dropna()

    box_data_birth.append(year_births)
    box_data_age.append(game_year - year_births)
```

[ ]:
```
plt.figure(figsize=(30, 10), dpi=80)
```

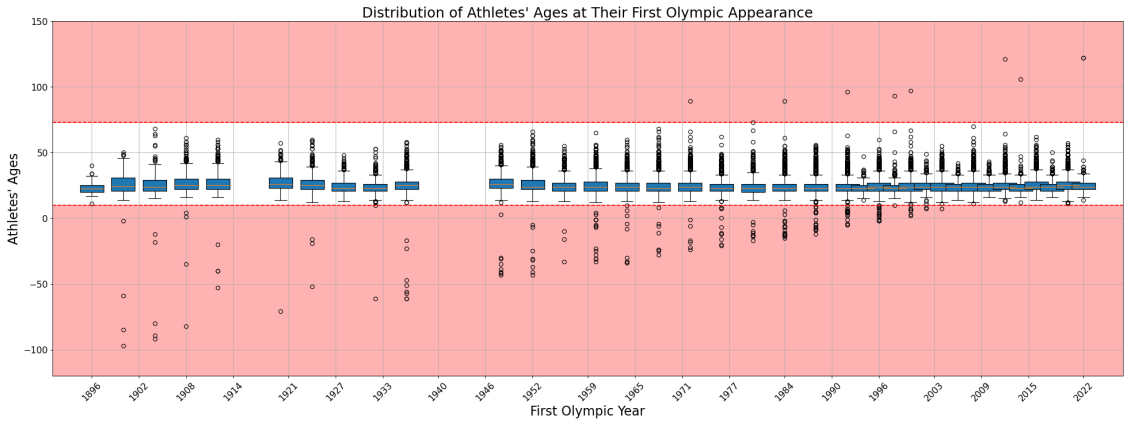
```
plt.boxplot(
    box_data_birth,
    positions=game_years,
    widths=[3] * len(game_years),
    patch_artist=True,
    manage_ticks=False,
)

line_data = np.arange(min(game_years) / 1.1, 1.1 * max(game_years))
plt.plot(line_data, line_data, linestyle="--", c="red")
plt.fill_between(x=line_data, y1=line_data, y2=2040, alpha=0.3, color="red")

# Decoration
plt.title(
    "Distribution of Athletes' Birth Years by First Olympic Appearance",␣
 ↪fontsize=22
)
plt.xlabel("First Olympic Year", fontsize=20)
plt.xticks(
    ticks=np.linspace(min(game_years), max(game_years), num=21, dtype=int),
    rotation=45,
    fontsize=14,
)
plt.xlim(min(game_years) - 5, max(game_years) + 5)
plt.yticks(fontsize=14)
plt.ylabel("Athletes' Birth Year", fontsize=20)
plt.ylim(1830, 2020)
plt.grid()
plt.show()
```



[ ]:
```
# References:
```

```
# https://eu.usatoday.com/story/sports/olympics/2022/11/24/
 ↪who-is-youngest-olympian/10380713002/
# https://www.oldest.org/sports/olympians/
youngestOlympian = 10
oldestOlympian = 73
```

[ ]:
```
plt.figure(figsize=(30, 10), dpi=80)


plt.boxplot(
    box_data_age,
    positions=game_years,
    widths=[3] * len(game_years),
    patch_artist=True,
    manage_ticks=False,
)

line_data = np.arange(min(game_years) / 2, 2 * max(game_years))
plt.axhline(youngestOlympian, linestyle="--", c="red")
plt.fill_between(x=line_data, y1=youngestOlympian, y2=-120, alpha=0.3,␣
 ↪color="red")

plt.axhline(oldestOlympian, linestyle="--", c="red")
plt.fill_between(x=line_data, y1=oldestOlympian, y2=150, alpha=0.3, color="red")

# Decoration
plt.title("Distribution of Athletes' Ages at Their First Olympic Appearance",␣
 ↪fontsize=22)
plt.xlabel("First Olympic Year", fontsize=20)
plt.xticks(
    ticks=np.linspace(min(game_years), max(game_years), num=21, dtype=int),
    rotation=45,
    fontsize=14,
)
plt.xlim(min(game_years) - 5, max(game_years) + 5)
plt.yticks(fontsize=14)
plt.ylabel("Athletes' Ages", fontsize=20)
plt.ylim(-120, 150)
plt.grid()
plt.show()
```



[ ]:
```
ageAtFirstOlympiad = (
    athletes_data["first_game_year"] - athletes_data["athlete_year_birth"]
)
tooYoungOldOlympic = (ageAtFirstOlympiad < youngestOlympian) | (
    oldestOlympian < ageAtFirstOlympiad
)
```

[ ]:
```
print(f"We discovered at least {np.sum(tooYoungOldOlympic)} inconsistencies!")
```

```
We discovered at least 186 inconsistencies!
```

We remove the inconsistent data:

[ ]:
```
DROPPED_ATHLETES += np.sum(tooYoungOldOlympic)

print(f"Total athletes dropped: {DROPPED_ATHLETES}")
```

```
Total athletes dropped: 208
```

[ ]:
```
athletes_data = athletes_data.drop(athletes_data[tooYoungOldOlympic].index)
```

Let's now look at the athletes' names.

[ ]:
```
athletes_data["athlete_full_name"].sort_values()[:10]
```

[ ]:
```
9840                  DENI DENI
5312                      . DENI
6257                   . PRIYANKA
6502                     . RAHUL
31310              A Baser WASIQI
1265                    A J HURT
31995        A-Aziz Hassan JALOOF
67749                  A. DARNIS
64387         A. Germaine GOLDING
```

```
69797    A. Linger ANDREAS LINGER
Name: athlete_full_name, dtype: object
```

In our data, there are names which were abbreviated and names which start with uncommon characters such as spaces. We, thus, clean our data by: removing leading/trailing spaces, removing rows starting with a period

[ ]:
```
athletes_data["athlete_full_name"] = (
    athletes_data["athlete_full_name"]
    .str.replace(r"\s*\.\s*", "", regex=True)
    .str.strip()
    .str.upper()
)
```

[ ]:
```
countsDict = athletes_data["athlete_full_name"].value_counts().to_dict()

athletes_data["name_frequency"] = athletes_data.apply(lambda x:␣
 ↪countsDict[x["athlete_full_name"]], axis=1)

athletes_data[athletes_data["name_frequency"] > 1].sort_values(
    by=["name_frequency", "athlete_full_name"], ascending=[False, True]
).head(10)
```

[ ]:
```
                                        athlete_url  athlete_full_name  \
28156  https://olympics.com/en/athletes/francisco-san…  FRANCISCO SANCHEZ
44334  https://olympics.com/en/athletes/francisco-san…  FRANCISCO SANCHEZ
47128  https://olympics.com/en/athletes/francisco-san…  FRANCISCO SANCHEZ
72475  https://olympics.com/en/athletes/francisco-san…  FRANCISCO SANCHEZ
15863       https://olympics.com/en/athletes/hao-wang-5           HAO WANG
18819       https://olympics.com/en/athletes/hao-wang-4           HAO WANG
40755       https://olympics.com/en/athletes/hao-wang-3           HAO WANG
69809       https://olympics.com/en/athletes/hao-wang-2           HAO WANG
12242    https://olympics.com/en/athletes/ivan-ivanov-8        IVAN IVANOV
18714      https://olympics.com/en/athletes/ivan-ivanov        IVAN IVANOV

       games_participations  athlete_year_birth first_game_city  \
28156                     2                1976         Atlanta
44334                     1                1958          Moscow
47128                     1                1956        Montreal
72475                     3                1965           Seoul
15863                     3                1983          Athens
18819                     1                1989         Beijing
40755                     1                1962     Los Angeles
69809                     1                1992          London
12242                     1                1989             Rio
18714                     1                1986         Beijing

      first_game_country country_code country_3_letter_code  first_game_year  \
```

28156	United States	US	USA	1996
44334	Russia	RU	RUS	1980
47128	Canada	CA	CAN	1976
72475	South Korea	KR	KOR	1988
15863	Greece	GR	GRC	2004
18819	China	CN	CHN	2008
40755	United States	US	USA	1984
69809	United Kingdom	GB	GBR	2012
12242	Brazil	BR	BRA	2016
18714	China	CN	CHN	2008

	bronze_count	silver_count	gold_count	total_medals	name_frequency
28156	0	0	0	0	4
44334	0	0	0	0	4
47128	0	0	0	0	4
72475	0	0	1	1	4
15863	0	3	2	5	4
18819	0	0	0	0	4
40755	0	0	0	0	4
69809	0	0	1	1	4
12242	0	0	0	0	4
18714	0	0	0	0	4

[ ]:
```
print(
    f"We discovered {len(athletes_data[athletes_data["name_frequency"] >␣
 ↪1]["athlete_full_name"].unique())} with repeated rows"
)
print(f"We discovered at least␣
 ↪{len(athletes_data[athletes_data["name_frequency"] > 1])} inconsistencies!")
```

```
We discovered 374 with repeated rows
We discovered at least 795 inconsistencies!
```

We drop the rows with duplicate names, retaining only the first occurrence of each unique name.

[ ]:
```
DROPPED_ATHLETES += len(athletes_data[athletes_data["name_frequency"] > 1]) -␣
 ↪len(
    athletes_data[athletes_data["name_frequency"] > 1]["athlete_full_name"].
 ↪unique()
)

print(f"Total athletes dropped: {DROPPED_ATHLETES}")
```

```
Total athletes dropped: 629
```

[ ]:
```
athletes_data = athletes_data.drop_duplicates(
    subset=["athlete_full_name"], keep="first"
)
```

```
athletes_data = athletes_data.drop("name_frequency", axis=1)
```

Let's finally take a look at athlete's URL's.

[ ]:
```
print(f"Total number of entries: {len(athletes_data)}")
print(f"Number of unique names: {len(athletes_data["athlete_full_name"].
 ↪unique())}")
print(f"Number of unique URLs: {len(athletes_data["athlete_url"].unique())}")
print(
    f"Difference: {len(athletes_data["athlete_full_name"].unique()) -␣
 ↪len(athletes_data["athlete_url"].unique())}"
)
```

```
Total number of entries: 75275
Number of unique names: 75275
Number of unique URLs: 75272
Difference: 3
```

We drop the rows with duplicate athlete URL, retaining only the first occurrence of each unique URL.

[ ]:
```
DROPPED_ATHLETES += len(athletes_data["athlete_full_name"].unique()) - len(
    athletes_data["athlete_url"].unique()
)

print(f"Total athletes dropped: {DROPPED_ATHLETES}")
```

```
Total athletes dropped: 632
```

[ ]:
```
athletes_data = athletes_data.drop_duplicates(subset=["athlete_url"],␣
 ↪keep="first")
```

Finally, we save our cleaned data.

[ ]:
```
athletes_data = athletes_data.reset_index(drop=True)
```

[ ]:
```
athletes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75272 entries, 0 to 75271
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   athlete_url           75272 non-null  object
 1   athlete_full_name     75272 non-null  object
 2   games_participations  75272 non-null  int64
 3   athlete_year_birth    72821 non-null  Int64
 4   first_game_city       75272 non-null  object
 5   first_game_country    75272 non-null  object
```

```
 6   country_code           75272 non-null  object
 7   country_3_letter_code  75272 non-null  object
 8   first_game_year        75272 non-null  Int64
 9   bronze_count           75272 non-null  int64
 10  silver_count           75272 non-null  int64
 11  gold_count             75272 non-null  int64
 12  total_medals           75272 non-null  int64
dtypes: Int64(2), int64(5), object(6)
memory usage: 7.6+ MB
```

[ ]:
```
athletes_data.to_csv(os.path.join(CLEAN_DATA_PATH, "olympic_athletes.csv"),␣
↪index=False)
```

4.3 Medals

[ ]:
```
medals_data.sample(10)
```

[ ]:
```
      discipline_title         slug_game  \
4624           Archery      beijing-2008
7604            Diving       sydney-2000
1932           Fencing          rio-2016
17771     Canoe Sprint       london-1948
6384     Weightlifting       athens-2004
14093    Cycling Track       munich-1972
16637           Boxing    melbourne-1956
14961         Swimming  mexico-city-1968
14054          Sailing       munich-1972
8190       Ski Jumping       nagano-1998

                                      event_title event_gender medal_type  \
4624     individual FITA Olympic round  70m women        Women     BRONZE
7604       synchronized diving 10m platform women        Women     SILVER
1932                               foil team men          Men     SILVER
17771                K1 500m kayak single women        Women     BRONZE
6384                                  69kg women        Women       GOLD
14093                      Sprint individual men          Men     BRONZE
16637               75-81kg lightheavyweight men          Men     BRONZE
14961                       100m butterfly women        Women       GOLD
14054  Finn - One Person Dinghy (Heavyweight) men         Open     BRONZE
8190                  Normal Hill Individual men          Men     BRONZE

      participant_type participant_title  \
4624           Athlete               NaN
7604          GameTeam       Canada team
1932          GameTeam       France team
17771          Athlete               NaN
6384           Athlete               NaN
14093          Athlete               NaN
```

```
16637          Athlete               NaN
14961          Athlete               NaN
14054          Athlete               NaN
8190           Athlete               NaN

                                              athlete_url    athlete_full_name  \
4624         https://olympics.com/en/athletes/ok-hee-yun           Ok-Hee YUN
7604     https://olympics.com/en/athletes/emilie-heymans       Emilie HEYMANS
1932                                                 NaN                  NaN
17771  https://olympics.com/en/athletes/friederike-sc…  Friederike SCHWINGL
6384       https://olympics.com/en/athletes/chunhong-liu         Chunhong LIU
14093    https://olympics.com/en/athletes/omar-pkhakadze       Omar PKHAKADZE
16637      https://olympics.com/en/athletes/carlos-lucas         Carlos LUCAS
14961  https://olympics.com/en/athletes/lynette-mccle…   Lynette MCCLEMENTS
14054    https://olympics.com/en/athletes/viktor-potapov       Viktor POTAPOV
8190   https://olympics.com/en/athletes/andreas-widho…    Andreas WIDHOELZL

                    country_name country_code country_3_letter_code
4624             Republic of Korea           KR                   KOR
7604                        Canada           CA                   CAN
1932                        France           FR                   FRA
17771                      Austria           AT                   AUT
6384   People's Republic of China           CN                   CHN
14093                Soviet Union          NaN                   URS
16637                       Chile           CL                   CHI
14961                   Australia           AU                   AUS
14054                Soviet Union          NaN                   URS
8190                        Austria           AT                   AUT
```

[ ]:
```
medals_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21697 entries, 0 to 21696
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   discipline_title      21697 non-null  object
 1   slug_game             21697 non-null  object
 2   event_title           21697 non-null  object
 3   event_gender          21697 non-null  object
 4   medal_type            21697 non-null  object
 5   participant_type      21697 non-null  object
 6   participant_title     6584 non-null   object
 7   athlete_url           17027 non-null  object
 8   athlete_full_name     18073 non-null  object
 9   country_name          21697 non-null  object
 10  country_code          20195 non-null  object
 11  country_3_letter_code 21697 non-null  object
```

```
dtypes: object(12)
memory usage: 2.0+ MB
```

[ ]:
```
medals_data.describe(include=object)
```

[ ]:
```
       discipline_title   slug_game     event_title event_gender medal_type  \
count             21697       21697           21697        21697      21697
unique               86          53            1436            4          3
top            Athletics  tokyo-2020  Individual men          Men     BRONZE
freq               3080        1188             215        13932       7529

       participant_type   participant_title  \
count             21697                6584
unique                2                 493
top              Athlete  United States team
freq              15113                 523

                                                athlete_url athlete_full_name  \
count                                                 17027              18073
unique                                                12116              12895
top     https://olympics.com/en/athletes/michael-phelp…    Michael PHELPS
freq                                                     16                 16

                     country_name country_code country_3_letter_code
count                       21697        20195                 21697
unique                        154          143                   154
top     United States of America           US                   USA
freq                         3094         3094                  3094
```

We noticed that some athletes in this dataset lack corresponding metadata in `athletes_data`.

[ ]:
```
athlete_urls_set = set(athletes_data["athlete_url"].dropna())
medals_urls_set = set(medals_data["athlete_url"].dropna())

missing_urls = list(medals_urls_set - athlete_urls_set)

print(f"There are {len(missing_urls)} medalists without metadata")
```

```
There are 115 medalists without metadata
```

We noticed that not all athletes have an associated URL.

[ ]:
```
np.sum(~medals_data["athlete_full_name"].isna()) == np.
  ↪sum(~medals_data["athlete_url"].isna())
```

[ ]:
```
False
```

[ ]:
```
print(
```

```
    f"Athletes without URL: {np.sum(~medals_data["athlete_full_name"].isna() &␣
  ↪medals_data["athlete_url"].isna())}"
)
```

```
Athletes without URL: 1046
```

We observed that individual competitions do not have a group title, which is logical given their nature, whereas group competitions are appropriately assigned a group title.

[ ]:
```
len(medals_data[medals_data["participant_type"] == "Athlete"]) == np.
  ↪sum(medals_data["participant_title"].isna())
```

[ ]:
```
True
```

[ ]:
```
len(medals_data[medals_data["participant_type"] == "GameTeam"]) == np.
  ↪sum(~medals_data["participant_title"].isna())
```

[ ]:
```
True
```

However, we discovered that some group competitions included an athlete's full name, presumably of an individual who participated in the competition. Since no documentation regarding these entries was found, this interpretation remains speculative.

[ ]:
```
len(medals_data[medals_data["participant_type"] == "GameTeam"]) == np.
  ↪sum(medals_data["athlete_full_name"].isna())
```

[ ]:
```
False
```

[ ]:
```
group_competitions = medals_data[medals_data["participant_type"] == "GameTeam"]
group_with_full_name = group_competitions[
    ~group_competitions["athlete_full_name"].isna()
]

group_with_full_name.head()
```

[ ]:
```
  discipline_title     slug_game    event_title event_gender medal_type  \
0          Curling  beijing-2022  Mixed Doubles        Mixed       GOLD
1          Curling  beijing-2022  Mixed Doubles        Mixed       GOLD
2          Curling  beijing-2022  Mixed Doubles        Mixed     SILVER
3          Curling  beijing-2022  Mixed Doubles        Mixed     SILVER
4          Curling  beijing-2022  Mixed Doubles        Mixed     BRONZE

  participant_type participant_title  \
0         GameTeam             Italy
1         GameTeam             Italy
2         GameTeam            Norway
3         GameTeam            Norway
4         GameTeam            Sweden
```

```
                                          athlete_url     athlete_full_name  \
0  https://olympics.com/en/athletes/stefania-cons…  Stefania CONSTANTINI
1      https://olympics.com/en/athletes/amos-mosaner           Amos MOSANER
2  https://olympics.com/en/athletes/kristin-skaslien      Kristin SKASLIEN
3  https://olympics.com/en/athletes/magnus-nedreg…    Magnus NEDREGOTTEN
4     https://olympics.com/en/athletes/almida-de-val         Almida DE VAL

  country_name country_code country_3_letter_code
0        Italy           IT                   ITA
1        Italy           IT                   ITA
2       Norway           NO                   NOR
3       Norway           NO                   NOR
4       Sweden           SE                   SWE
```

[ ]:
```
print(f"Number of group competitions: {len(group_competitions)}")
print(f"Number of group competitions with full name:␣
 ↪{len(group_with_full_name)}")
print(f"Number of group competitions without full name:␣
 ↪{len(group_competitions) - len(group_with_full_name)}")
```

```
Number of group competitions: 6584
Number of group competitions with full name: 2960
Number of group competitions without full name: 3624
```

Up until now, despite encountering some unusual occurrences, we have chosen not to remove the rows containing them.

We noticed some countries missing their countries codes:

[ ]:
```
print(medals_data[medals_data["country_code"].isna()]["country_name"].unique())
```

```
['Namibia' 'Olympic Athletes from Russia' 'Trinidad and Tobago'
 'Unified Team' 'Soviet Union' 'Lebanon' 'West Indies Federation'
 'United Arab Republic' 'Australasia' 'Bohemia' 'MIX']
```

[ ]:
```
medals_data["country_code"] = medals_data.apply(
    lambda row: (
        get_country_code(row["country_name"])
        if pd.isna(row["country_code"])
        and row["country_name"] in ["Namibia", "Trinidad and Tobago", "Lebanon"]
        else row["country_code"]
    ),
    axis=1,
)
```

We have noticed that some of the rows of `country_3_letter_code` did not correspond to the actual country code. Thus, we repopulate this column:

[58]:
```
medals_data["country_3_letter_code"] = medals_data["country_name"].map(lambda x:
 ↪ get_country_code(x, alpha=3))
```

```
Error fetching country code for 'Hong Kong, China': hong kong, china
Error fetching country code for 'Olympic Athletes from Russia': olympic athletes
from russia
Error fetching country code for 'Independent Olympic Athletes': independent
olympic athletes
Error fetching country code for 'Serbia and Montenegro': serbia and montenegro
Error fetching country code for 'Unified Team': unified team
Error fetching country code for 'Czechoslovakia': czechoslovakia
Error fetching country code for 'Soviet Union': soviet union
Error fetching country code for 'German Democratic Republic (Germany)': german
democratic republic (germany)
Error fetching country code for 'Yugoslavia': yugoslavia
Error fetching country code for 'Netherlands Antilles': netherlands antilles
Error fetching country code for 'Virgin Islands, US': virgin islands, us
Error fetching country code for 'West Indies Federation': west indies federation
Error fetching country code for 'United Arab Republic': united arab republic
Error fetching country code for 'Australasia': australasia
Error fetching country code for 'Bohemia': bohemia
Error fetching country code for 'MIX': mix
```

Finally, we save our cleaned data.

[ ]:
```
medals_data = medals_data.reset_index(drop=True)
```

[ ]:
```
athletes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75272 entries, 0 to 75271
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   athlete_url            75272 non-null  object
 1   athlete_full_name      75272 non-null  object
 2   games_participations   75272 non-null  int64
 3   athlete_year_birth     72821 non-null  Int64
 4   first_game_city        75272 non-null  object
 5   first_game_country     75272 non-null  object
 6   country_code           75272 non-null  object
 7   country_3_letter_code  75272 non-null  object
 8   first_game_year        75272 non-null  Int64
 9   bronze_count           75272 non-null  int64
 10  silver_count           75272 non-null  int64
 11  gold_count             75272 non-null  int64
 12  total_medals           75272 non-null  int64
dtypes: Int64(2), int64(5), object(6)
memory usage: 7.6+ MB
```

[ ]:
```
medals_data.to_csv(os.path.join(CLEAN_DATA_PATH, "olympic_medals.csv"),␣
↪index=False)
```

4.4 Hosts

[ ]:
```
hosts_data.sample(10)
```

[ ]:

	game_slug	game_end_date	game_start_date	\
19	los-angeles-1984	1984-08-12T19:00:00Z	1984-07-28T15:00:00Z	
46	chamonix-1924	1924-02-05T20:00:00Z	1924-01-25T08:00:00Z	
2	pyeongchang-2018	2018-02-25T08:00:00Z	2018-02-08T23:00:00Z	
21	moscow-1980	1980-08-03T18:00:00Z	1980-07-19T05:00:00Z	
16	albertville-1992	1992-02-23T19:00:00Z	1992-02-08T07:00:00Z	
42	lake-placid-1932	1932-02-15T18:00:00Z	1932-02-04T13:00:00Z	
29	tokyo-1964	1964-10-24T11:00:00Z	1964-10-09T23:00:00Z	
32	squaw-valley-1960	1960-02-28T04:00:00Z	1960-02-18T16:00:00Z	
39	berlin-1936	1936-08-16T19:00:00Z	1936-08-01T07:00:00Z	
15	barcelona-1992	1992-08-09T18:00:00Z	1992-07-25T06:00:00Z	

	game_location	game_name	game_season	game_year
19	United States	Los Angeles 1984	Summer	1984
46	France	Chamonix 1924	Winter	1924
2	Republic of Korea	PyeongChang 2018	Winter	2018
21	USSR	Moscow 1980	Summer	1980
16	France	Albertville 1992	Winter	1992
42	United States	Lake Placid 1932	Winter	1932
29	Japan	Tokyo 1964	Summer	1964
32	United States	Squaw Valley 1960	Winter	1960
39	Germany	Berlin 1936	Summer	1936
15	Spain	Barcelona 1992	Summer	1992

[ ]:
```
hosts_data["game_start_date"] = pd.to_datetime(
    hosts_data["game_start_date"], errors="coerce"
)
hosts_data["game_end_date"] = pd.to_datetime(
    hosts_data["game_end_date"], errors="coerce"
)
```

[ ]:
```
hosts_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   game_slug        53 non-null     object
 1   game_end_date    53 non-null     datetime64[ns, UTC]
 2   game_start_date  53 non-null     datetime64[ns, UTC]
```

```
 3   game_location    53 non-null     object
 4   game_name        53 non-null     object
 5   game_season      53 non-null     object
 6   game_year        53 non-null     int64
dtypes: datetime64[ns, UTC](2), int64(1), object(4)
memory usage: 3.0+ KB
```

[ ]:
```
hosts_data.describe(include=np.number)
```

[ ]:
```
          game_year
count     53.000000
mean    1967.547170
std       35.201926
min     1896.000000
25%     1936.000000
50%     1972.000000
75%     1996.000000
max     2022.000000
```

[ ]:
```
hosts_data.describe(include="datetime64[ns, UTC]")
```

[ ]:
```
                           game_end_date  \
count                                 53
mean   1967-12-29 22:12:03.735849056+00:00
min              1896-04-15 11:39:39+00:00
25%              1936-08-16 19:00:00+00:00
50%              1972-02-13 11:00:00+00:00
75%              1996-08-05 21:00:00+00:00
max              2022-02-20 12:00:00+00:00

                         game_start_date
count                                 53
mean   1967-11-30 15:08:28.641509432+00:00
min              1896-04-06 11:38:39+00:00
25%              1936-08-01 07:00:00+00:00
50%              1972-02-02 23:00:00+00:00
75%              1996-07-19 12:00:00+00:00
max              2022-02-04 15:00:00+00:00
```

[ ]:
```
hosts_data.describe(include=object)
```

[ ]:
```
           game_slug  game_location     game_name game_season
count             53             53            53          53
unique            53             26            53           2
top     beijing-2022  United States  Beijing 2022      Summer
freq               1              8             1          29
```

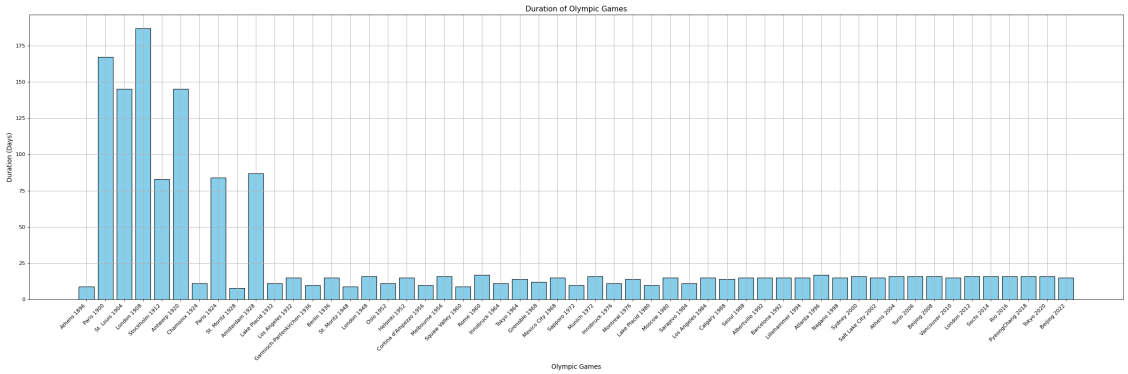
[ ]:
```
hosts_data["country_code"] = hosts_data["game_location"].apply(
    lambda x: get_country_code(x, alpha=2)
)
hosts_data["country_3_letter_code"] = hosts_data["game_location"].apply(
    lambda x: get_country_code(x, alpha=3)
)
```

```
Error fetching country code for 'Yugoslavia': yugoslavia
Error fetching country code for 'USSR': ussr
Error fetching country code for 'Australia, Sweden': australia, sweden
Error fetching country code for 'USSR': ussr
Error fetching country code for 'Australia, Sweden': australia, sweden
```

[ ]:
```
hosts_data = hosts_data.reset_index(drop=True)
```

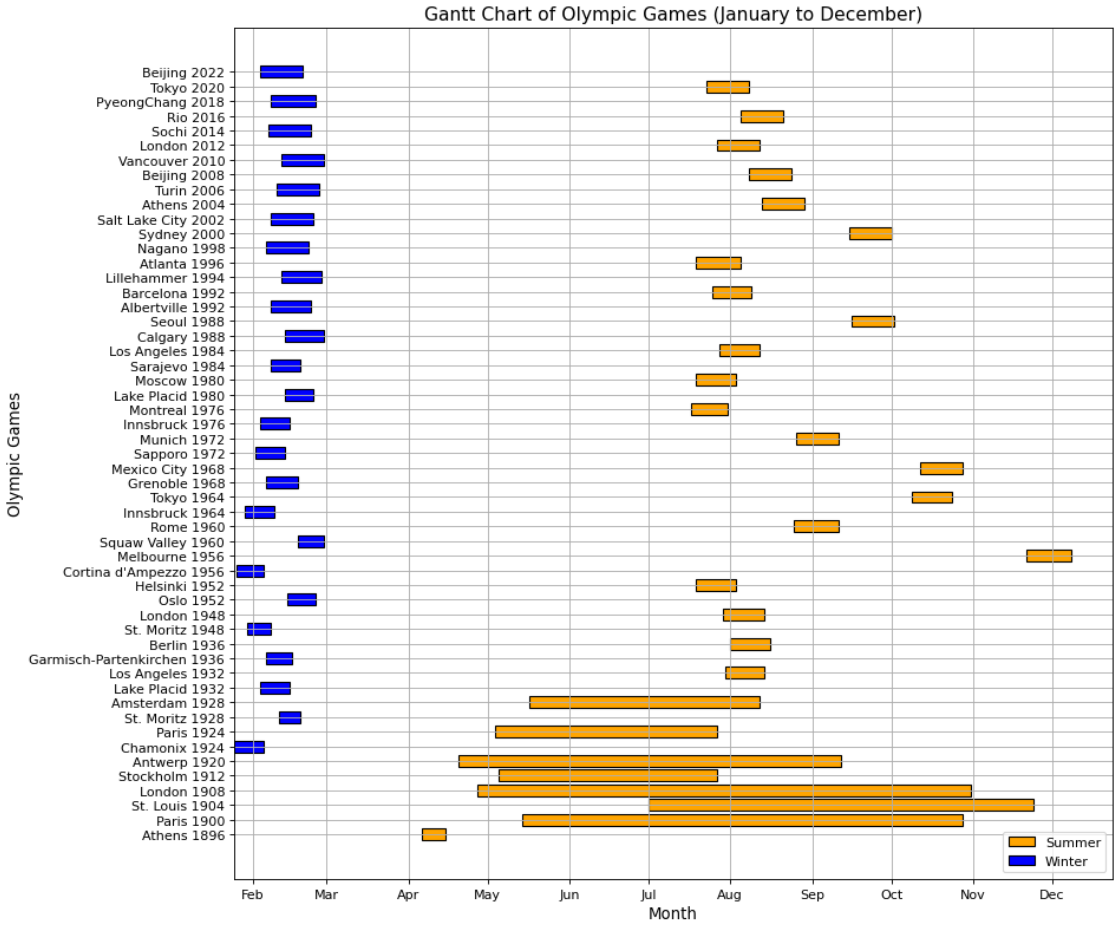
[ ]:
```
hosts_data.to_csv(os.path.join(CLEAN_DATA_PATH, "olympic_hosts.csv"),␣
↪index=False)
```

[ ]:
```
hosts_data["game_duration_days"] = (
    hosts_data["game_end_date"] - hosts_data["game_start_date"]
).dt.days

hosts_data = hosts_data.sort_values("game_start_date")

plt.figure(figsize=(30, 10), dpi=80)
plt.bar(
    hosts_data["game_name"],
    hosts_data["game_duration_days"],
    color="skyblue",
    edgecolor="black",
)
plt.xlabel("Olympic Games", fontsize=12)
plt.ylabel("Duration (Days)", fontsize=12)
plt.title("Duration of Olympic Games", fontsize=14)
plt.xticks(rotation=45, ha="right")

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



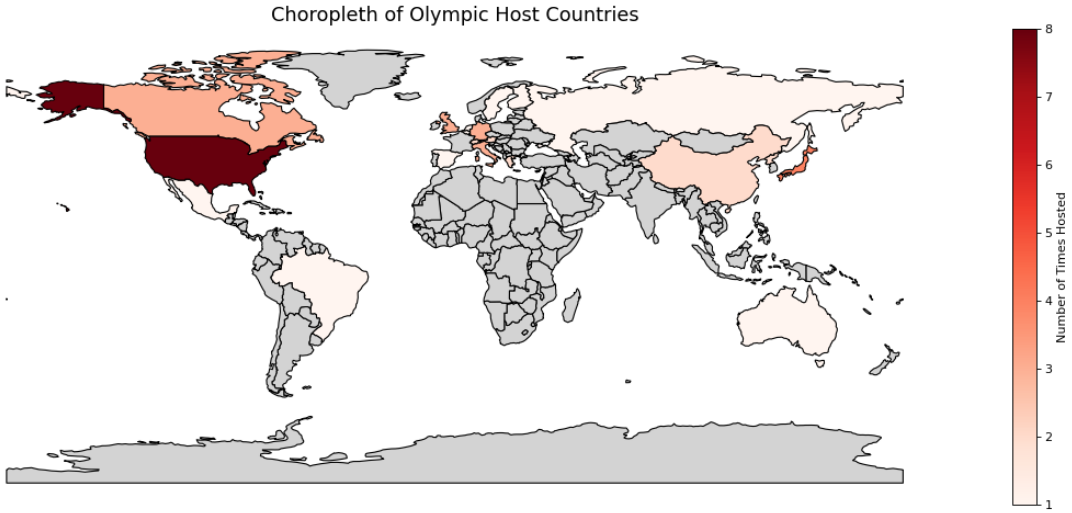
[ ]:
```
hosts_data["game_season_color"] = hosts_data["game_season"].map(
    {"Winter": "blue", "Summer": "orange"}
)

hosts_data["start_month_day"] = hosts_data["game_start_date"].dt.
 ↪strftime("%m-%d")
hosts_data["end_month_day"] = hosts_data["game_end_date"].dt.strftime("%m-%d")

hosts_data["start_date"] = pd.to_datetime(hosts_data["start_month_day"],␣
 ↪format="%m-%d")
hosts_data["end_date"] = pd.to_datetime(hosts_data["end_month_day"],␣
 ↪format="%m-%d")

hosts_data["duration"] = (hosts_data["end_date"] - hosts_data["start_date"]).dt.
 ↪days

plt.figure(figsize=(12, 10), dpi=80)

bars = []
labels = []
for index, row in hosts_data.iterrows():
    bar = plt.barh(
        y=row["game_name"],
        left=row["start_date"],
        width=row["duration"],
        color=row["game_season_color"],
        edgecolor="black",
    )
    if row["game_season"] not in labels:
        bars.append(bar[0])
        labels.append(row["game_season"])

# Format the x-axis for months
```

```
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%b"))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.xlabel("Month", fontsize=12)
plt.ylabel("Olympic Games", fontsize=12)
plt.title("Gantt Chart of Olympic Games (January to December)", fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.legend(bars, labels, loc="lower right")

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



[ ]:
```
world = gpd.read_file(os.path.join(DATA_PATH, "map", "ne_110m_admin_0_countries.
↪shp"))
```

```
host_counts = hosts_data.groupby("country_code")["game_name"].count().
 ↪reset_index()
host_counts.columns = ["country_code", "host_count"]

world = world.merge(host_counts, how="left", left_on="ISO_A2",␣
 ↪right_on="country_code")

fig, ax = plt.subplots(figsize=(15, 6), dpi=80)

world.plot(
    column="host_count",
    cmap="Reds",
    legend=True,
    legend_kwds={"label": "Number of Times Hosted"},
    edgecolor="black",
    missing_kwds={"color": "lightgrey", "label": "No Data"},
    ax=ax
)

plt.title("Choropleth of Olympic Host Countries", fontsize=16)
plt.axis("off")
plt.tight_layout()
plt.show()
```



4.5 Results

[ ]:
```
results_data.sample(10)
```

[ ]:

	discipline_title	event_title
54977	Figure skating	Individual women
34869	Short Track Speed Skating	500m men
89219	Boxing	48-51kg flyweight men
20026	Judo	+ 100kg (heavyweight) men
51860	Shooting	10m air pistol 60 shots men
129812	Modern Pentathlon	Individual competition men
202	Freestyle Skiing	Mixed Team Aerials
81938	Swimming	100m butterfly men
21779	Equestrian Dressage	Dressage Individual Grand Prix mixed
99950	Alpine Skiing	giant slalom men

	slug_game	participant_type	medal_type	athletes	rank_equal
54977	salt-lake-city-2002	Athlete	SILVER	NaN	NaN
34869	vancouver-2010	Athlete	NaN	NaN	NaN
89219	seoul-1988	Athlete	NaN	NaN	True
20026	rio-2016	Athlete	NaN	NaN	True
51860	athens-2004	Athlete	NaN	NaN	True
129812	melbourne-1956	Athlete	NaN	NaN	NaN
202	beijing-2022	GameTeam	GOLD	NaN	False
81938	barcelona-1992	Athlete	NaN	NaN	NaN
21779	rio-2016	Athlete	NaN	NaN	NaN
99950	sarajevo-1984	Athlete	NaN	NaN	NaN

	rank_position	country_name	country_code
54977	2	Russian Federation	RU
34869	4	Republic of Korea	KR
89219	33	Denmark	DK
20026	17	Austria	AT
51860	17	Italy	IT
129812	12	Mexico	MX
202	1	United States of America	US
81938	59	Bahamas	BS
21779	33	Canada	CA
99950	56	People's Republic of China	CN

	country_3_letter_code
54977	RUS
34869	KOR
89219	DEN
20026	AUT
51860	ITA
129812	MEX
202	USA
81938	BAH
21779	CAN
99950	CHN

```
                                                athlete_url  \
54977    https://olympics.com/en/athletes/irina-slutskaya
34869                                                  NaN
89219   https://olympics.com/en/athletes/johnny-bredah…
20026   https://olympics.com/en/athletes/daniel-allers…
51860   https://olympics.com/en/athletes/francesco-bru…
129812   https://olympics.com/en/athletes/jose-perez-mier
202                                                    NaN
81938   https://olympics.com/en/athletes/timothy-alexa…
21779         https://olympics.com/en/athletes/megan-lane
99950   https://olympics.com/en/athletes/chang-cheng-l…

              athlete_full_name value_unit value_type
54977           Irina SLUTSKAYA        2.0       RANK
34869               Yun-Gi Gwak        NaN        NaN
89219   Johnny Bredahl JOHANSEN        NaN        NaN
20026       Daniel ALLERSTORFER        NaN        NaN
51860           Francesco BRUNO        NaN        NaN
129812          Jose PEREZ MIER    4,093.5     POINTS
202                         NaN        NaN       CODE
81938   Timothy Alexander ENEAS        NaN        NaN
21779                Megan LANE        NaN        NaN
99950           Chang-Cheng LIU     208510       TIME
```

[ ]:
```
results_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162804 entries, 0 to 162803
Data columns (total 15 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   discipline_title       162804 non-null  object
 1   event_title            162804 non-null  object
 2   slug_game              162804 non-null  object
 3   participant_type       162804 non-null  object
 4   medal_type             20206 non-null   object
 5   athletes               7976 non-null    object
 6   rank_equal             32526 non-null   object
 7   rank_position          158926 non-null  object
 8   country_name           162804 non-null  object
 9   country_code           157768 non-null  object
 10  country_3_letter_code  162804 non-null  object
 11  athlete_url            129991 non-null  object
 12  athlete_full_name      141646 non-null  object
 13  value_unit             78646 non-null   object
 14  value_type             90049 non-null   object
dtypes: object(15)
```

```
memory usage: 18.6+ MB
```

We noticed that some athletes in this dataset lack corresponding metadata in `athletes_data`.

[ ]:
```
athlete_urls_set = set(athletes_data["athlete_url"].dropna())
results_urls_set = set(results_data["athlete_url"].dropna())

missing_urls = list(results_urls_set - athlete_urls_set)

print(f"There are {len(missing_urls)} medalists without metadata")
```

```
There are 559 medalists without metadata
```

We noticed that not all athletes have an associated URL.

[ ]:
```
np.sum(~results_data["athlete_full_name"].isna()) == np.
↪sum(~results_data["athlete_url"].isna())
```

[ ]:
```
False
```

[ ]:
```
print(
    f"Athletes without URL: {np.sum(~results_data["athlete_full_name"].isna() &␣
↪results_data["athlete_url"].isna())}"
)
```

```
Athletes without URL: 11655
```

[ ]:
```
results_data = results_data.reset_index(drop=True)
```

[ ]:
```
results_data.to_csv(os.path.join(CLEAN_DATA_PATH, "olympic_results.csv"),␣
↪index=False)
```