SportsStats Olympics 120 Years

May 22, 2025

1 Project Proposal

1.1 Step 1: Preparing for Your Proposal

Document your preparation in developing the project proposal. This includes:

1.1.1 1. Which client/dataset did you select and why?

Client 3: SportsStats (Olympics - 120 Years of Data)

"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."

I chose this dataset because I find the social/health/sports spheres engaging, and looking at 120 years of data could offer interesting insights.

1.1.2 2. Describe the steps you took to import and clean the data.

Steps taken to import and clean the data included:

1. Imported pandas and used the read function to import the datasets.
2. Ran EDA functions like head(), info(), and describe() to check for errors or missing data.
3. Updated one record with a missing value in regions.

[1]:
```
# Import pandas library

import pandas as pd
```

[2]:
```
# Import the datasets

events = pd.read_csv('athlete_events.csv')
regions = pd.read_csv('noc_regions.csv')
```

[15]:
```
events.head()
```

[15]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

[5]:
```
events.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      271116 non-null  int64
 1   Name    271116 non-null  object
 2   Sex     271116 non-null  object
 3   Age     261642 non-null  float64
 4   Height  210945 non-null  float64
 5   Weight  208241 non-null  float64
 6   Team    271116 non-null  object
 7   NOC     271116 non-null  object
 8   Games   271116 non-null  object
 9   Year    271116 non-null  int64
 10  Season  271116 non-null  object
 11  City    271116 non-null  object
 12  Sport   271116 non-null  object
 13  Event   271116 non-null  object
 14  Medal   39783 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
```

The data types look acceptable, however Age, Height, and Weight have some missing values. I would go back to the client and ask if additional records exist. It remains to be seen how the

columns with missing values effect calculations.

The medal colums also has far fewer records than the other columns, but that is to be expected. It is unknown at this point if any of those are missing values or simply null values where an athlete did not medal in an event.

[3]:
```
events.describe()
```

[3]:
```
                ID            Age         Height         Weight  \
count  271116.000000  261642.000000  210945.000000  208241.000000
mean    68248.954396      25.556898     175.338970      70.702393
std     39022.286345       6.393561      10.518462      14.348020
min         1.000000      10.000000     127.000000      25.000000
25%     34643.000000      21.000000     168.000000      60.000000
50%     68205.000000      24.000000     175.000000      70.000000
75%    102097.250000      28.000000     183.000000      79.000000
max    135571.000000      97.000000     226.000000     214.000000

                Year
count  271116.000000
mean     1978.378480
std        29.877632
min      1896.000000
25%      1960.000000
50%      1988.000000
75%      2002.000000
max      2016.000000
```

Checking the describe() output for value ranges and outliers. Ages range from 10 to 97, both seem odd for an Olympic athlete. Height and Weight fall into acceptable ranges. Year spans 1896 - 2016.

[16]:
```
regions.head()
```

[16]:
```
   NOC       region                notes
0  AFG  Afghanistan                  NaN
1  AHO      Curacao  Netherlands Antilles
2  ALB      Albania                  NaN
3  ALG      Algeria                  NaN
4  AND      Andorra                  NaN
```

[6]:
```
regions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230 entries, 0 to 229
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   NOC     230 non-null    object
 1   region  227 non-null    object
```

```
 2   notes   21 non-null     object
dtypes: object(3)
memory usage: 5.5+ KB
```

The data types all look correct. Region has 3 missing or null values. NOC appears to be a 3 letter country code, and region is the full name of the country, not actually a region. Notes seems to indicate whether a country is a terrirtory of another.

[4]:
```
regions.describe()
```

[4]:
```
         NOC   region                 notes
count    230      227                    21
unique   230      206                    21
top      SYR  Germany  Netherlands Antilles
freq       1        4                     1
```

[3]:
```
# Which records have missing values?

def missing_region(df, column_name):
  if column_name not in df.columns:
    print(f"Error: Column '{column_name}' not found in the DataFrame.")
    return df
  return df[df[column_name].isnull()]

# Apply the function to your 'regions' DataFrame and the 'region' column
missing_records = missing_region(regions, 'region')

# Print the resulting DataFrame containing the full records with missing␣
 ↪'region' values
print(missing_records)
```

```
     NOC region                 notes
168  ROT    NaN  Refugee Olympic Team
208  TUV    NaN                Tuvalu
213  UNK    NaN               Unknown
```

After further examination, the record of TUV appears to refer to the country Tuvalu, located in Oceania.

The Refugee Olympic Team (ROT), is "a team of independent Olympic participants who are refugees, established by the International Olympic Committee (IOC) and the Olympic Refuge Foundation (ORF). The team was created to give forcibly displaced athletes the opportunity to showcase their talents on the highest sporting stage."

Unknown (UNK) is also not associated with a specific region.

[23]:
```
# Identify the record where NOC is 'TUV' and region is missing
condition = (regions['NOC'] == 'TUV') & (regions['region'].isnull())

# Use .loc to select that specific record and update the 'region' column
```

```
regions.loc[condition, 'region'] = 'Tuvalu'

# Verify the change
print(regions[regions['NOC'] == 'TUV'])
```

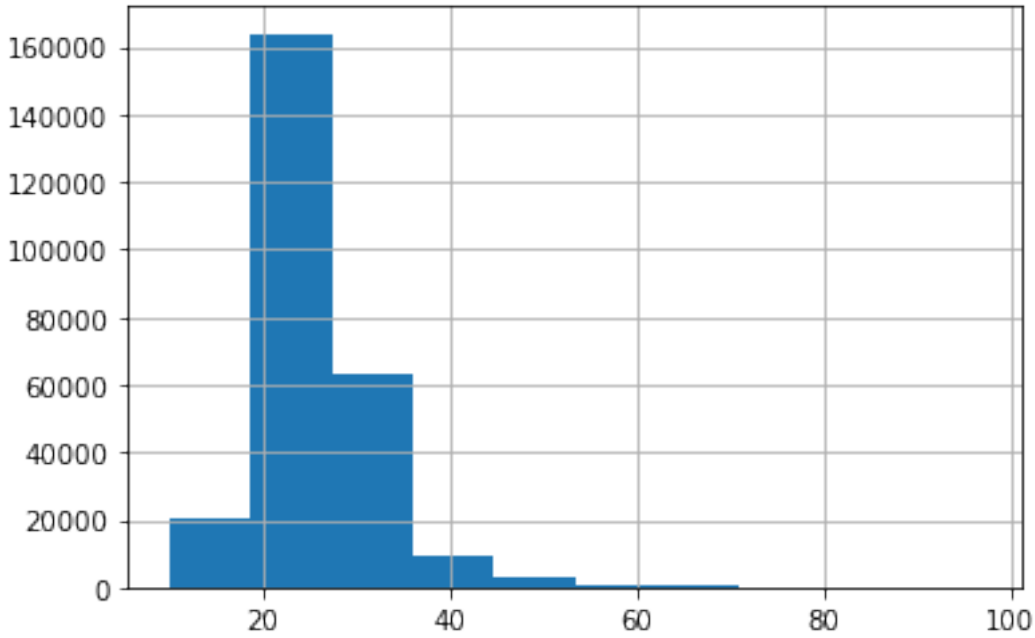
```
     NOC  region   notes
208  TUV  Tuvalu  Tuvalu
```

1.1.3 3. Perform initial exploration of data and provide some screenshots or display some stats of the data you are looking at.

[31]:
```
# Create histogram for each numerical variable from the events table to show␣
↪the distribution of values and check for outliers.

events['Age'].hist()
```

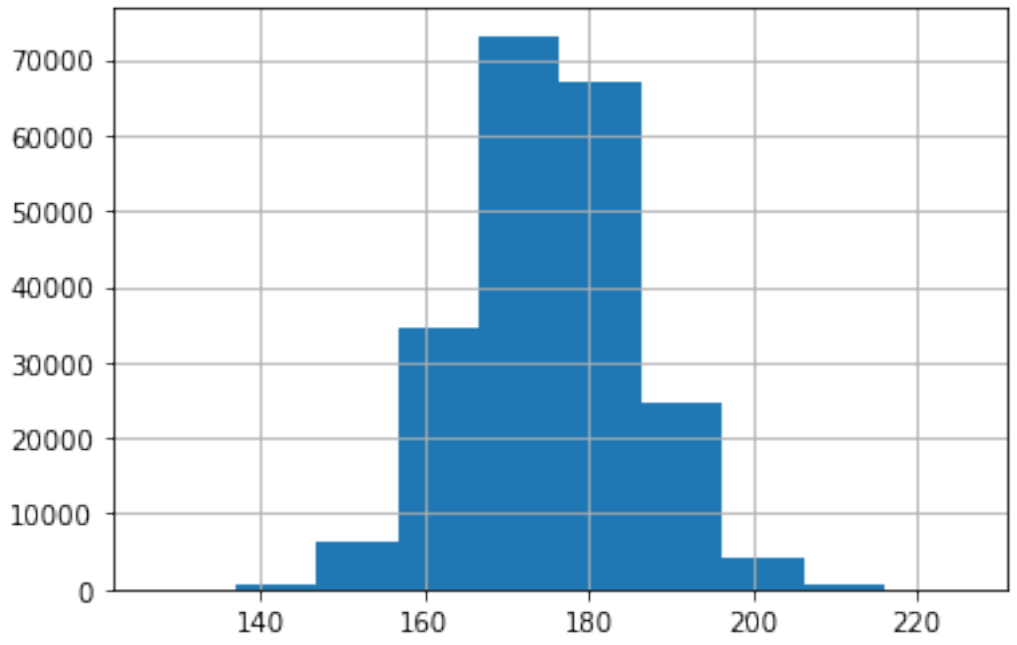
[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x722544558a50>
```



The Age historgram shows most athletes are in their 20s.

[14]:
```
events['Height'].hist()
```

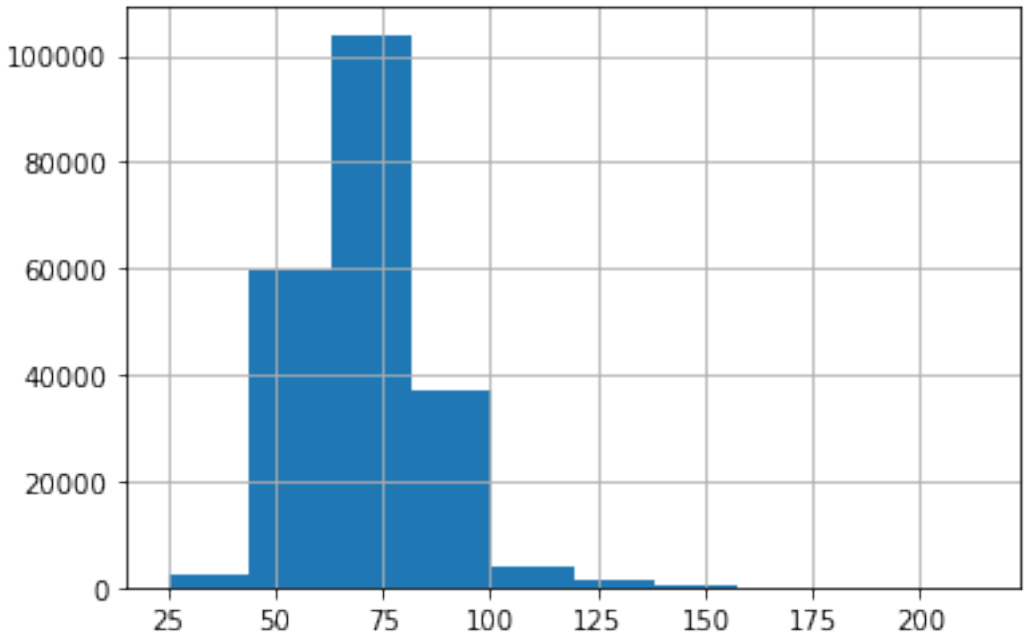
[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7225066a0610>
```



The Height histogram shows most athletes are 170 - 190 cms tall.

[15]:
```
events['Weight'].hist()
```

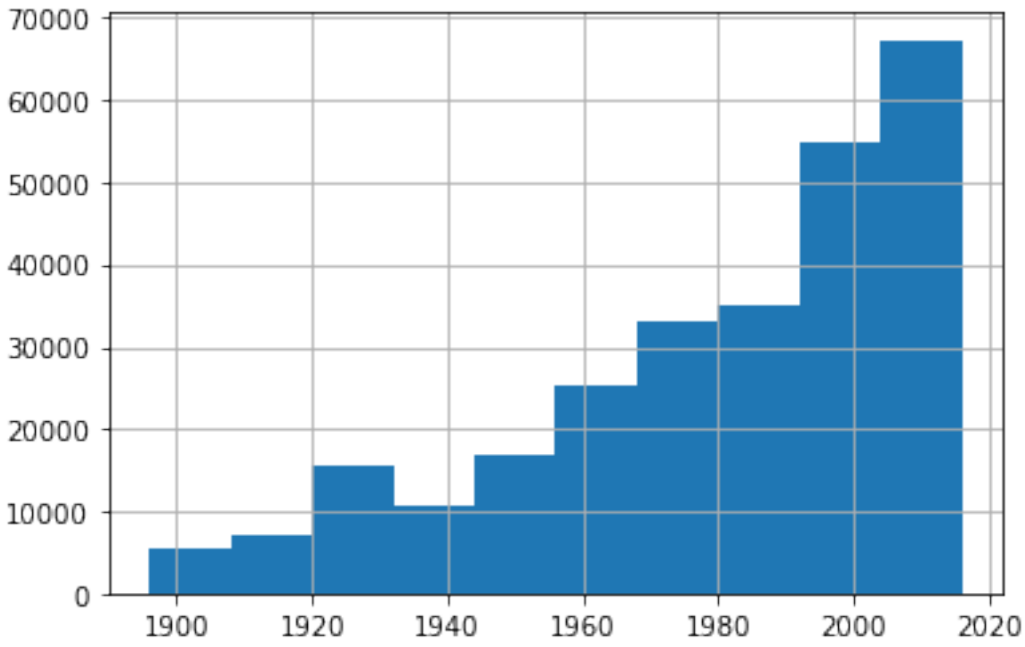
[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x722504c88ad0>
```



The Weight histogram shows most athletes weight 50 - 100 kgs.

[16]:
```
events['Year'].hist()
```

[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x722504b78c50>
```



The Year histogram reveals that the number of athletes competing in events steadily increased over the time span we are looking at, with a big jump around 1990.

In addition to Python functions, we can also use SQL queries to examine the string and object variables. Below are SQL versions of what could be returned using the Python value_counts() function.

[21]:
```
# Import the SQL library

from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

[32]:
```
# Count by Sex

pysqldf('SELECT Sex, COUNT(Sex) AS count FROM events GROUP BY Sex')
```

[32]:

	Sex	count
0	F	74522
1	M	196594

[39]:
```
# Count by NOC and Team, ordered largest to smallest

pysqldf('SELECT NOC, Team, COUNT(Team) AS count FROM events GROUP BY NOC, Team␣
↪ORDER BY count DESC')
```

[39]:

	NOC	Team	count
0	USA	United States	17847
1	FRA	France	11988
2	GBR	Great Britain	11404
3	ITA	Italy	10260
4	GER	Germany	9326
…	…	…	…
1226	USA	Mythilus	1
1227	USA	Ravel	1
1228	USA	Union des Socits Franais de Sports Athletiques	1
1229	USA	United States/France	1
1230	YUG	Konstanz	1

[1231 rows x 3 columns]

[43]:
```
# Count by Olympic Games and City

pysqldf('SELECT Games, City, COUNT(Games) AS count FROM events GROUP BY Games,␣
↪City')
```

[43]:

	Games	City	count
0	1896 Summer	Athina	380
1	1900 Summer	Paris	1936
2	1904 Summer	St. Louis	1301
3	1906 Summer	Athina	1733
4	1908 Summer	London	3101
5	1912 Summer	Stockholm	4040
6	1920 Summer	Antwerpen	4292
7	1924 Summer	Paris	5233
8	1924 Winter	Chamonix	460
9	1928 Summer	Amsterdam	4992
10	1928 Winter	Sankt Moritz	582
11	1932 Summer	Los Angeles	2969
12	1932 Winter	Lake Placid	352
13	1936 Summer	Berlin	6506
14	1936 Winter	Garmisch-Partenkirchen	895
15	1948 Summer	London	6405
16	1948 Winter	Sankt Moritz	1075
17	1952 Summer	Helsinki	8270
18	1952 Winter	Oslo	1088
19	1956 Summer	Melbourne	4829
20	1956 Summer	Stockholm	298

```
21  1956 Winter       Cortina d'Ampezzo   1307
22  1960 Summer                   Roma   8119
23  1960 Winter            Squaw Valley   1116
24  1964 Summer                  Tokyo   7702
25  1964 Winter              Innsbruck   1778
26  1968 Summer            Mexico City   8588
27  1968 Winter               Grenoble   1891
28  1972 Summer                 Munich  10304
29  1972 Winter                Sapporo   1655
30  1976 Summer               Montreal   8641
31  1976 Winter              Innsbruck   1861
32  1980 Summer                 Moskva   7191
33  1980 Winter            Lake Placid   1746
34  1984 Summer            Los Angeles   9454
35  1984 Winter               Sarajevo   2134
36  1988 Summer                  Seoul  12037
37  1988 Winter                Calgary   2639
38  1992 Summer              Barcelona  12977
39  1992 Winter            Albertville   3436
40  1994 Winter            Lillehammer   3160
41  1996 Summer                Atlanta  13780
42  1998 Winter                 Nagano   3605
43  2000 Summer                 Sydney  13821
44  2002 Winter          Salt Lake City   4109
45  2004 Summer                 Athina  13443
46  2006 Winter                 Torino   4382
47  2008 Summer                Beijing  13602
48  2010 Winter              Vancouver   4402
49  2012 Summer                 London  12920
50  2014 Winter                  Sochi   4891
51  2016 Summer          Rio de Janeiro  13688
```

[41]:
```
# Count by Season

pysqldf('SELECT Season, COUNT(Season) AS count FROM events GROUP BY Season')
```

[41]:
```
   Season   count
0  Summer  222552
1  Winter   48564
```

[47]:
```
# Count by Event and Sport, ordered most to least

pysqldf('SELECT Event, Sport, COUNT(Sport) AS count FROM events GROUP BY Event,␣
 ↪Sport ORDER BY count DESC')
```

[47]:
```
                                        Event          Sport  count
0                            Football Men's Football       Football   5733
```

```
1                          Ice Hockey Men's Ice Hockey     Ice Hockey   4762
2                                  Hockey Men's Hockey         Hockey   3958
3                          Water Polo Men's Water Polo     Water Polo   3358
4                          Basketball Men's Basketball     Basketball   3280
..                                                   …            …    …
760  Archery Men's Target Archery, 50 metres, Indiv…        Archery      2
761       Basque Pelota Men's Two-Man Teams With Cesta  Basque Pelota      2
762                              Croquet Mixed Doubles        Croquet      2
763                              Sailing Mixed 18 foot        Sailing      2
764                        Aeronautics Mixed Aeronautics    Aeronautics      1

[765 rows x 3 columns]
```

[48]:
```
# Count Medal

pysqldf('SELECT Medal, COUNT(Medal) AS count FROM events GROUP BY Medal')
```

[48]:
```
     Medal  count
0     None      0
1   Bronze  13295
2     Gold  13372
3   Silver  13116
```

[52]:
```
# Count Medal by NOC, ordered most to least

pysqldf('SELECT NOC, Medal, COUNT(Medal) AS count FROM events GROUP BY NOC,␣
 ↪Medal ORDER BY count DESC')
```

[52]:
```
      NOC   Medal  count
0     USA    Gold   2638
1     USA  Silver   1641
2     USA  Bronze   1358
3     URS    Gold   1082
4     GER  Bronze    746
..    …      …     …
587   YEM    None      0
588   YMD    None      0
589   YUG    None      0
590   ZAM    None      0
591   ZIM    None      0

[592 rows x 3 columns]
```

1.1.4 4. Create an ERD or proposed ERD to show the relationships of the data you are exploring.

With only 2 tables, an ERD is very straightforward. Both the events and regions table have the NOC variable in common, so that is our primary key. The values in the NOC column each appear once in the regions table, whereas each NOC value can appear multiple times in the events table.

Therefore, there is a one-to-many (1:M) relationship between the two tables. Using Crow's Foot Notation, this is indicated by the 'train tracks' at the regions table end and the 'crow's foot' at the events table end.

1.2 Step 2: Develop Project Proposal

In this step, you will need to include the following:

1.2.1 Description

The SportStats Olympic Games project aims to explore the history of the Olympics through athlete data. With 120 years of Games to draw from, many aspects of the data can be analyzed over a lonfg timeframe. Demographics, performance, medals, events, and sports can all be examined from a country/regional level down to individual athletes. Not only would an average Olympic sports fan be interested in the findings of this project, but athletic trainers and Olympic teams would find the analysis useful. The target audience could be anyone from a sports trainer to an Olympic team coach.

1.2.2 Questions

1. What are the demographic trends of Olympic athletes over time? Is the average height, weight or age changing?
2. What does country participation look like?
3. What are the medal count trends?
4. What events are most popular (have the most athletes competing)?

1.2.3 Hypothesis

1. Is there an advantage for athletes being from the host country? Do they win more?
2. Does it help or hurt for an athlete to compete in multiple events?
3. Is there a correlation between physical attributes and winning medals?

1.2.4 Approach

In order to prove or disprove my hypotheses, most of the features of the dataset will be examined. 1. To determine whether an athlete from a host country medals more often, the Team, NOC, Games, Year, Season, City, and Medal features will be examined. 2. To determine whether competing in multiple events at the same Games helps or hurts an athlete, the Name, Games, Year, Season,

Sport, Event, and Medal features will be examined. 3. To determine whether there is a correlation between physical attributes and winning medals, the Age, Height, Weight, Sport, Event, and Medal features will be examined.

Metrics necessary for this analysis include aggregating counts, sums, and averages. These include how many medals an athlete or country won and how many events a single athlete competes in. Searching for correlations is also key, like whether the medals leader for a specific Games is also the host country.