athlete dt rf loggggggggggg READY

August 6, 2023

[2]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
df=pd.read_csv('D:/DS/resume projects/athlete seaborn/athlete_events.csv')
```

[3]:
```
df.shape
```

[3]:
```
(271116, 15)
```

[4]:
```
df
```

[4]:
```
            ID                      Name Sex   Age  Height  Weight  \
0            1                 A Dijiang   M  24.0   180.0    80.0
1            2                  A Lamusi   M  23.0   170.0    60.0
2            3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN
3            4      Edgar Lindenau Aabye   M  34.0     NaN     NaN
4            5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
…        …                       …  ..   …     …     …
271111  135569                Andrzej ya   M  29.0   179.0    89.0
271112  135570                  Piotr ya   M  27.0   176.0    59.0
271113  135570                  Piotr ya   M  27.0   176.0    59.0
271114  135571        Tomasz Ireneusz ya   M  30.0   185.0    96.0
271115  135571        Tomasz Ireneusz ya   M  34.0   185.0    96.0

                  Team  NOC        Games  Year  Season            City  \
0                China  CHN  1992 Summer  1992  Summer       Barcelona
1                China  CHN  2012 Summer  2012  Summer          London
2              Denmark  DEN  1920 Summer  1920  Summer       Antwerpen
3       Denmark/Sweden  DEN  1900 Summer  1900  Summer           Paris
4          Netherlands  NED  1988 Winter  1988  Winter         Calgary
…                …  …          …   …     …             …
271111        Poland-1  POL  1976 Winter  1976  Winter       Innsbruck
271112          Poland  POL  2014 Winter  2014  Winter           Sochi
271113          Poland  POL  2014 Winter  2014  Winter           Sochi
271114          Poland  POL  1998 Winter  1998  Winter          Nagano
271115          Poland  POL  2002 Winter  2002  Winter  Salt Lake City
```

```
                Sport                                     Event Medal
0          Basketball               Basketball Men's Basketball   NaN
1                Judo              Judo Men's Extra-Lightweight   NaN
2            Football                   Football Men's Football   NaN
3          Tug-Of-War               Tug-Of-War Men's Tug-Of-War  Gold
4       Speed Skating          Speed Skating Women's 500 metres   NaN
…               …                                        …   …
271111           Luge                Luge Mixed (Men)'s Doubles   NaN
271112    Ski Jumping  Ski Jumping Men's Large Hill, Individual   NaN
271113    Ski Jumping        Ski Jumping Men's Large Hill, Team   NaN
271114      Bobsleigh                      Bobsleigh Men's Four   NaN
271115      Bobsleigh                      Bobsleigh Men's Four   NaN

[271116 rows x 15 columns]
```

[5]:
```
df.size
```

[5]:
```
4066740
```

[6]:
```
df
```

[6]:
```
            ID                      Name Sex   Age  Height  Weight  \
0            1                 A Dijiang   M  24.0   180.0    80.0
1            2                  A Lamusi   M  23.0   170.0    60.0
2            3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN
3            4      Edgar Lindenau Aabye   M  34.0     NaN     NaN
4            5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
…        …                       …  ..   …     …     …
271111  135569                Andrzej ya   M  29.0   179.0    89.0
271112  135570                  Piotr ya   M  27.0   176.0    59.0
271113  135570                  Piotr ya   M  27.0   176.0    59.0
271114  135571        Tomasz Ireneusz ya   M  30.0   185.0    96.0
271115  135571        Tomasz Ireneusz ya   M  34.0   185.0    96.0

                  Team  NOC        Games  Year  Season            City  \
0                China  CHN  1992 Summer  1992  Summer       Barcelona
1                China  CHN  2012 Summer  2012  Summer          London
2              Denmark  DEN  1920 Summer  1920  Summer       Antwerpen
3       Denmark/Sweden  DEN  1900 Summer  1900  Summer           Paris
4          Netherlands  NED  1988 Winter  1988  Winter         Calgary
…                …  …          …   …     …             …
271111        Poland-1  POL  1976 Winter  1976  Winter       Innsbruck
271112          Poland  POL  2014 Winter  2014  Winter           Sochi
271113          Poland  POL  2014 Winter  2014  Winter           Sochi
271114          Poland  POL  1998 Winter  1998  Winter          Nagano
271115          Poland  POL  2002 Winter  2002  Winter  Salt Lake City
```

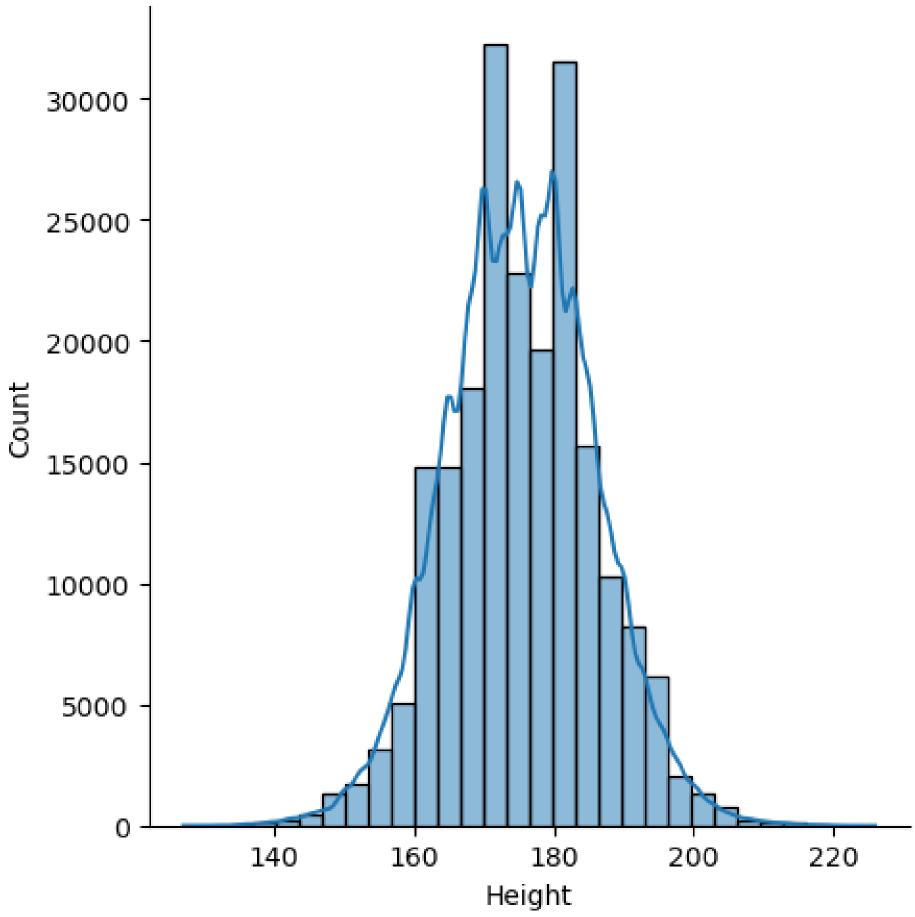
```
                  Sport                                       Event  Medal
0            Basketball             Basketball Men's Basketball    NaN
1                  Judo            Judo Men's Extra-Lightweight    NaN
2              Football                 Football Men's Football    NaN
3            Tug-Of-War             Tug-Of-War Men's Tug-Of-War   Gold
4         Speed Skating        Speed Skating Women's 500 metres    NaN
…                   …                                         …    …
271111             Luge                Luge Mixed (Men)'s Doubles    NaN
271112     Ski Jumping  Ski Jumping Men's Large Hill, Individual    NaN
271113     Ski Jumping        Ski Jumping Men's Large Hill, Team    NaN
271114       Bobsleigh                      Bobsleigh Men's Four    NaN
271115       Bobsleigh                      Bobsleigh Men's Four    NaN

[271116 rows x 15 columns]
```

```
[7]: # athelete who won gold from Team US
     c=len(df[(df['Team']=='United States') & (df['Medal']=='Gold')])
     c
```

```
[7]: 2474
```

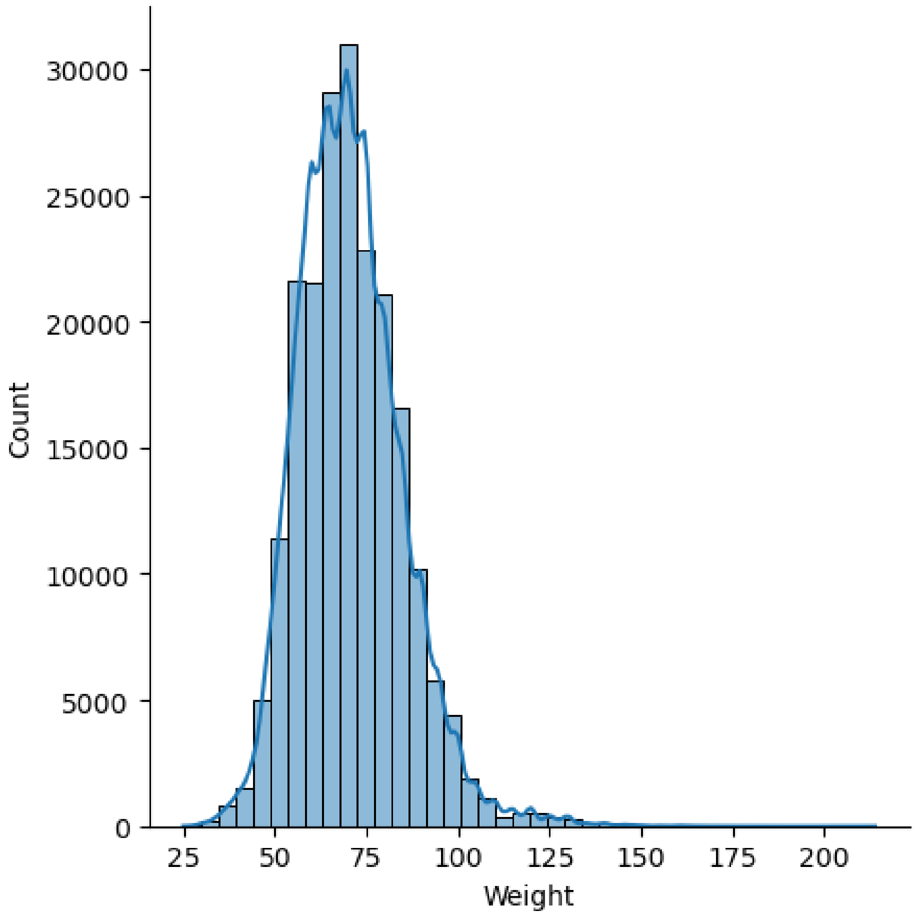
```
[8]: #Q1 analyse hight data by removing None
     sns.displot(x=df.Height.dropna(),bins=30,kde=True)
```

```
[8]: <seaborn.axisgrid.FacetGrid at 0x1e18d73c4c0>
```



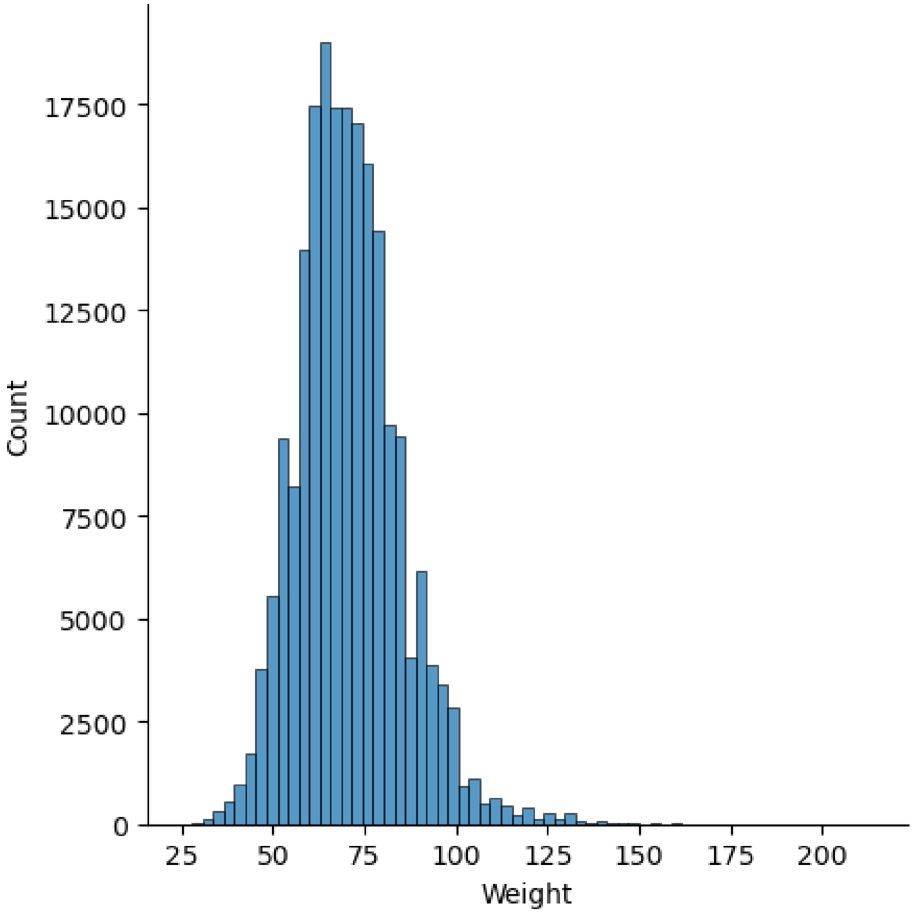
```
[9]: #Q1 analyse weight data by removing None
     sns.displot(x=df.Weight.dropna(),bins=40,kde=True)
```

```
[9]: <seaborn.axisgrid.FacetGrid at 0x1e18d6e4820>
```



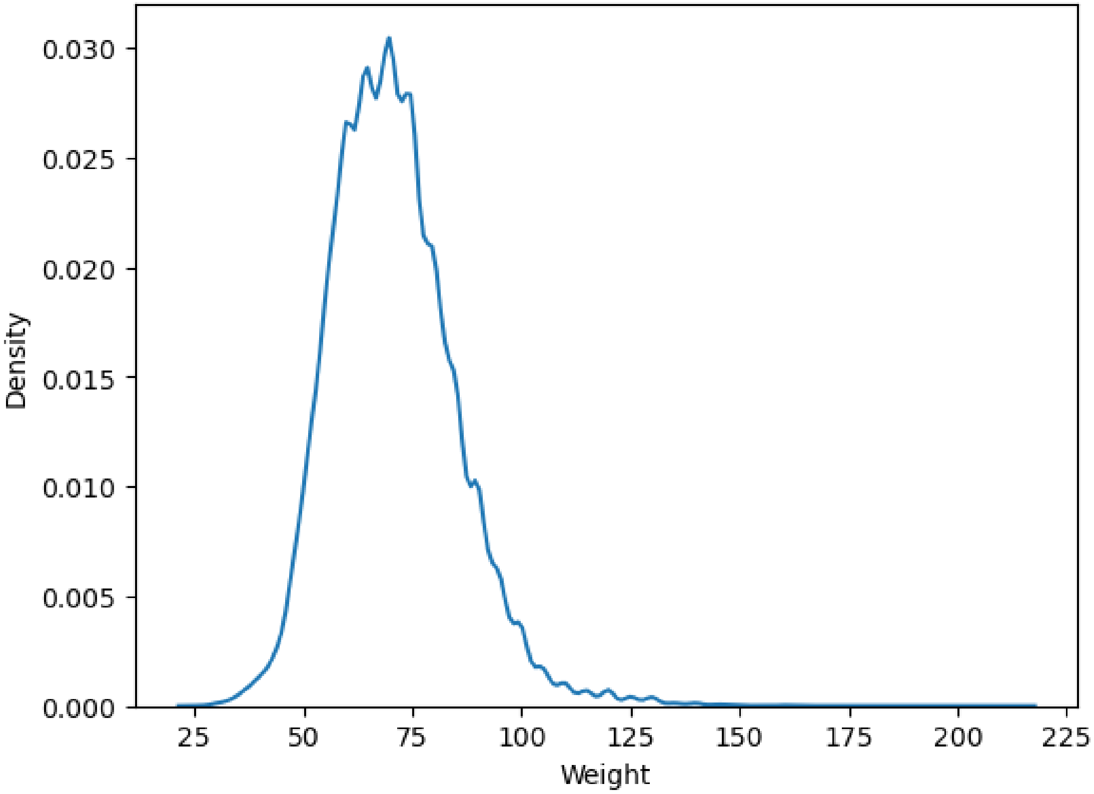
[10]:
```
plt.figure(figsize=(10,5))
sns.displot(x=df.Weight.dropna(),bins=65)
```

[10]:
```
<seaborn.axisgrid.FacetGrid at 0x1e193a6e9e0>

<Figure size 1000x500 with 0 Axes>
```



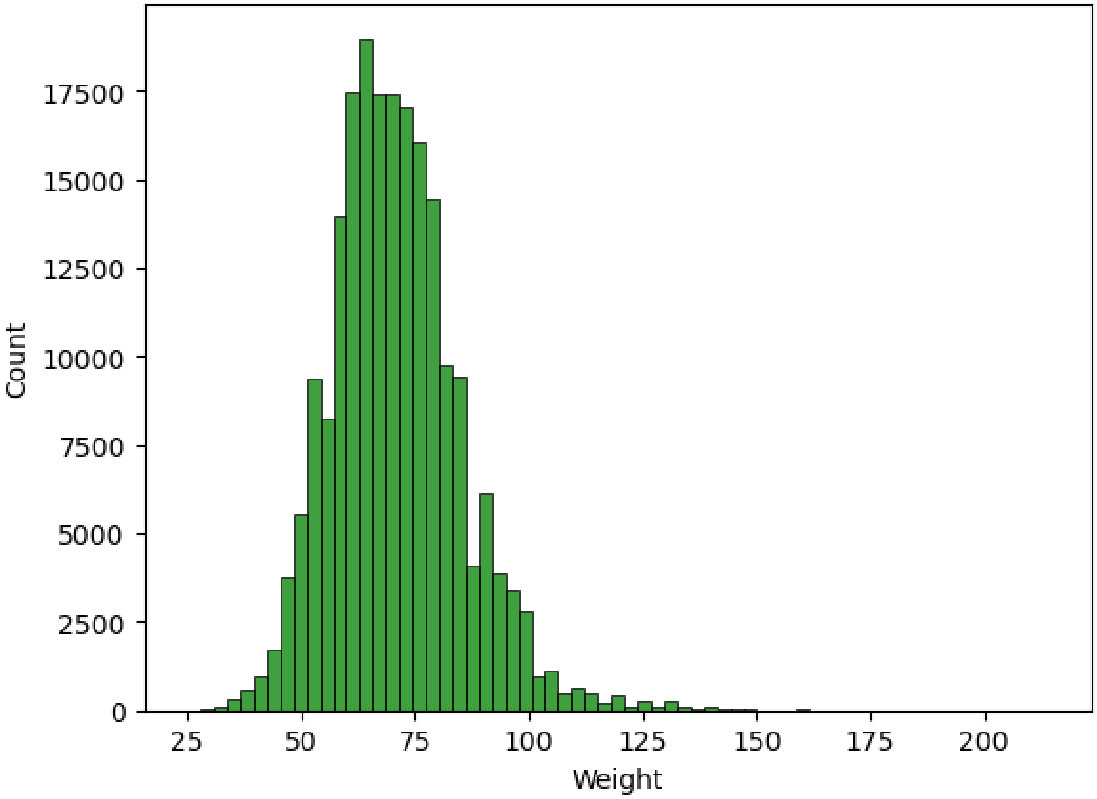
```
[11]: sns.kdeplot(x=df.Weight.dropna())
```

```
[11]: <Axes: xlabel='Weight', ylabel='Density'>
```



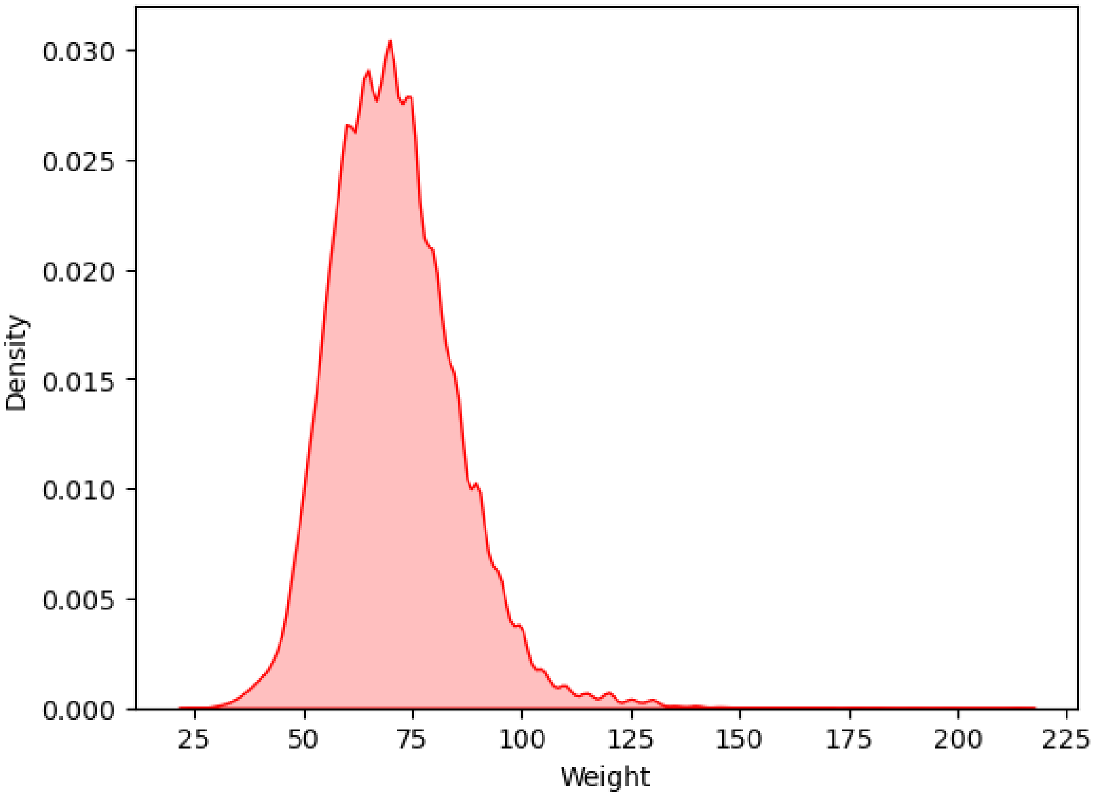
[12]:
```
sns.histplot(x=df.Weight.dropna(),bins=65,color='green')
```

[12]:
```
<Axes: xlabel='Weight', ylabel='Count'>
```



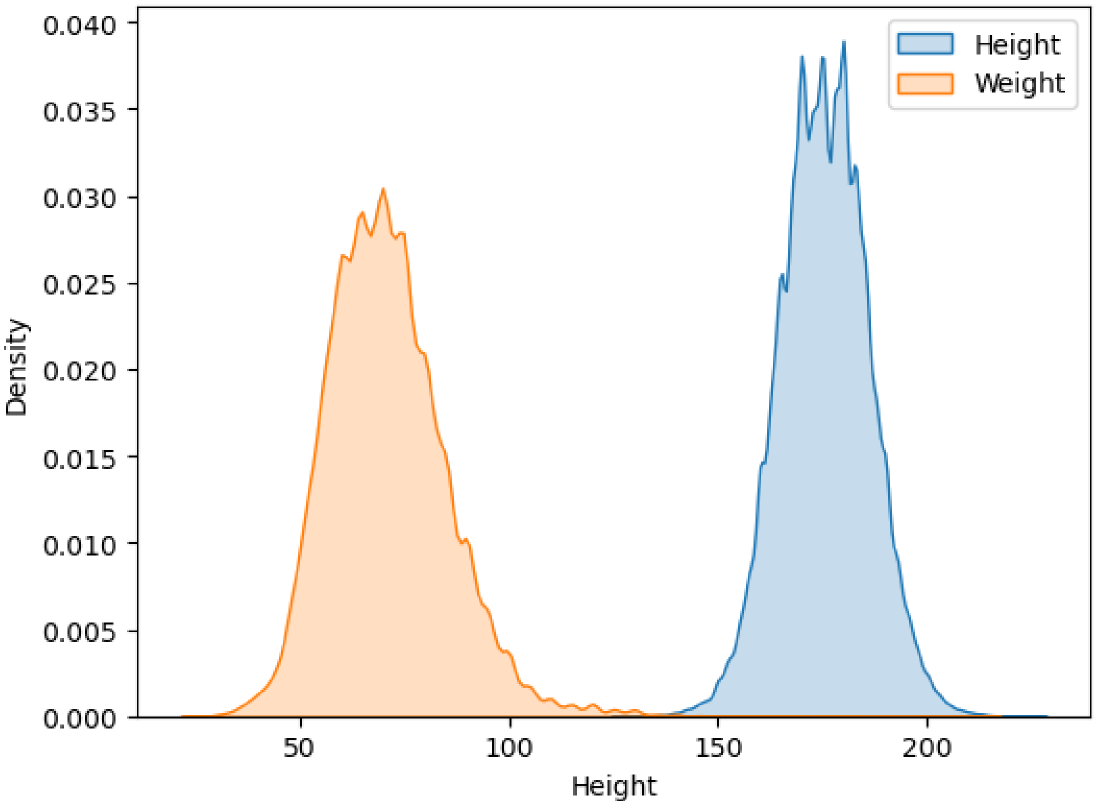
[13]:
```
sns.kdeplot(x=df.Weight,color='red',fill='red')
```

[13]:
```
<Axes: xlabel='Weight', ylabel='Density'>
```



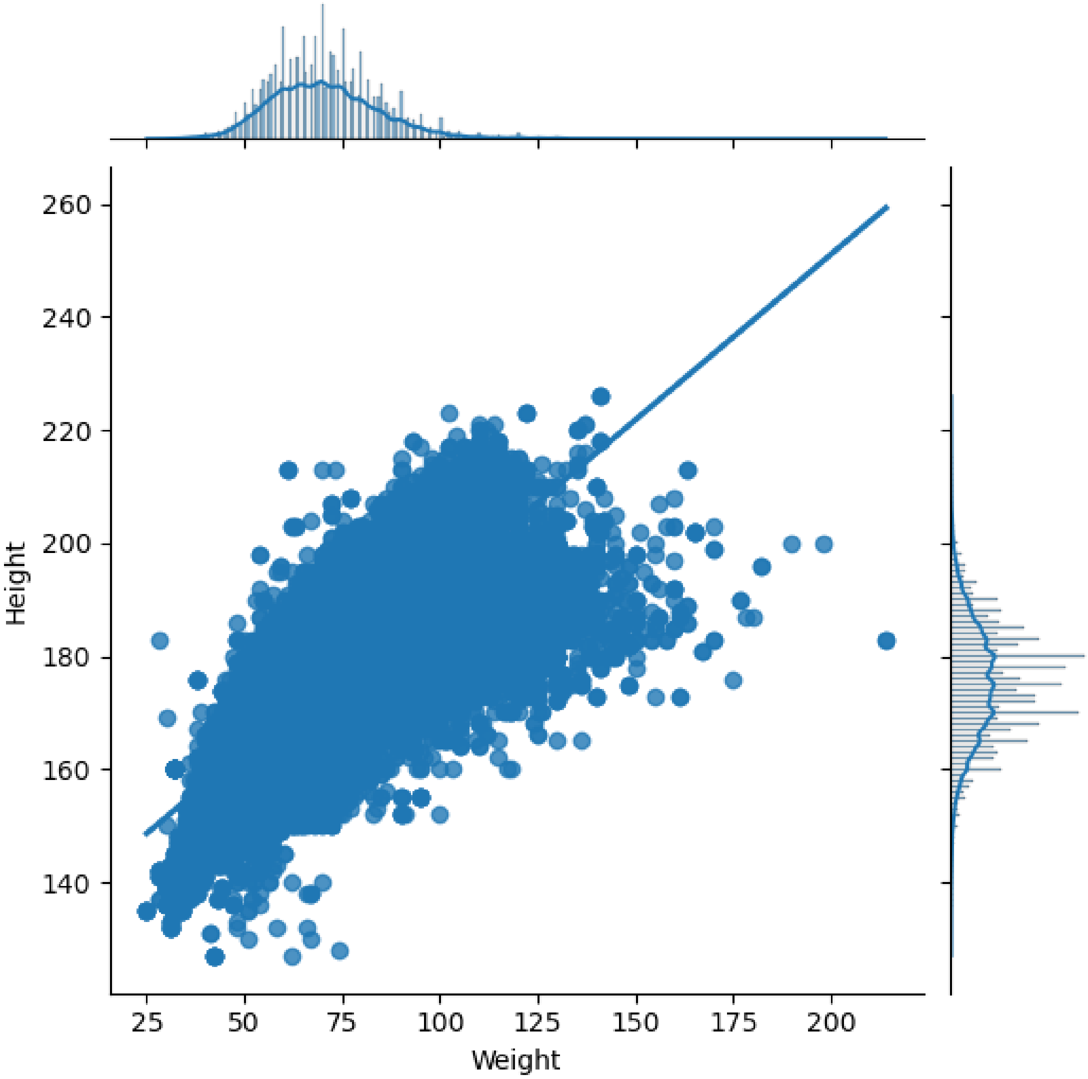
[14]:
```
sns.kdeplot(x=df.Height.dropna(),label='Height',fill='red')
sns.kdeplot(x=df.Weight.dropna(),label='Weight',fill='red')
plt.legend()
```

[14]:
```
<matplotlib.legend.Legend at 0x20cf87232b0>
```



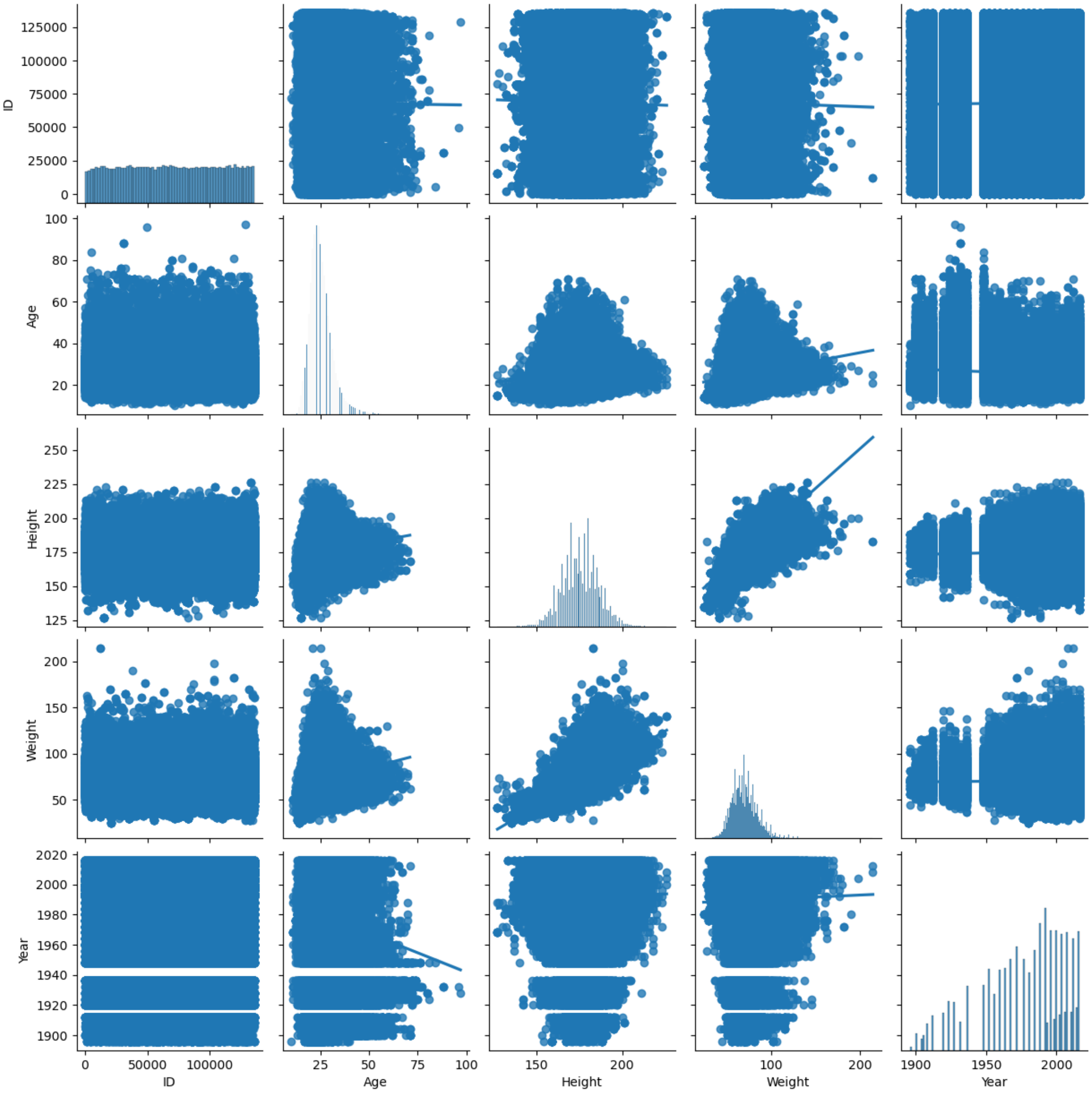
```
[15]: sns.jointplot(y=df.Height.dropna(), x=df.Weight.dropna(), kind='reg')
```

```
[15]: <seaborn.axisgrid.JointGrid at 0x20cf8723be0>
```



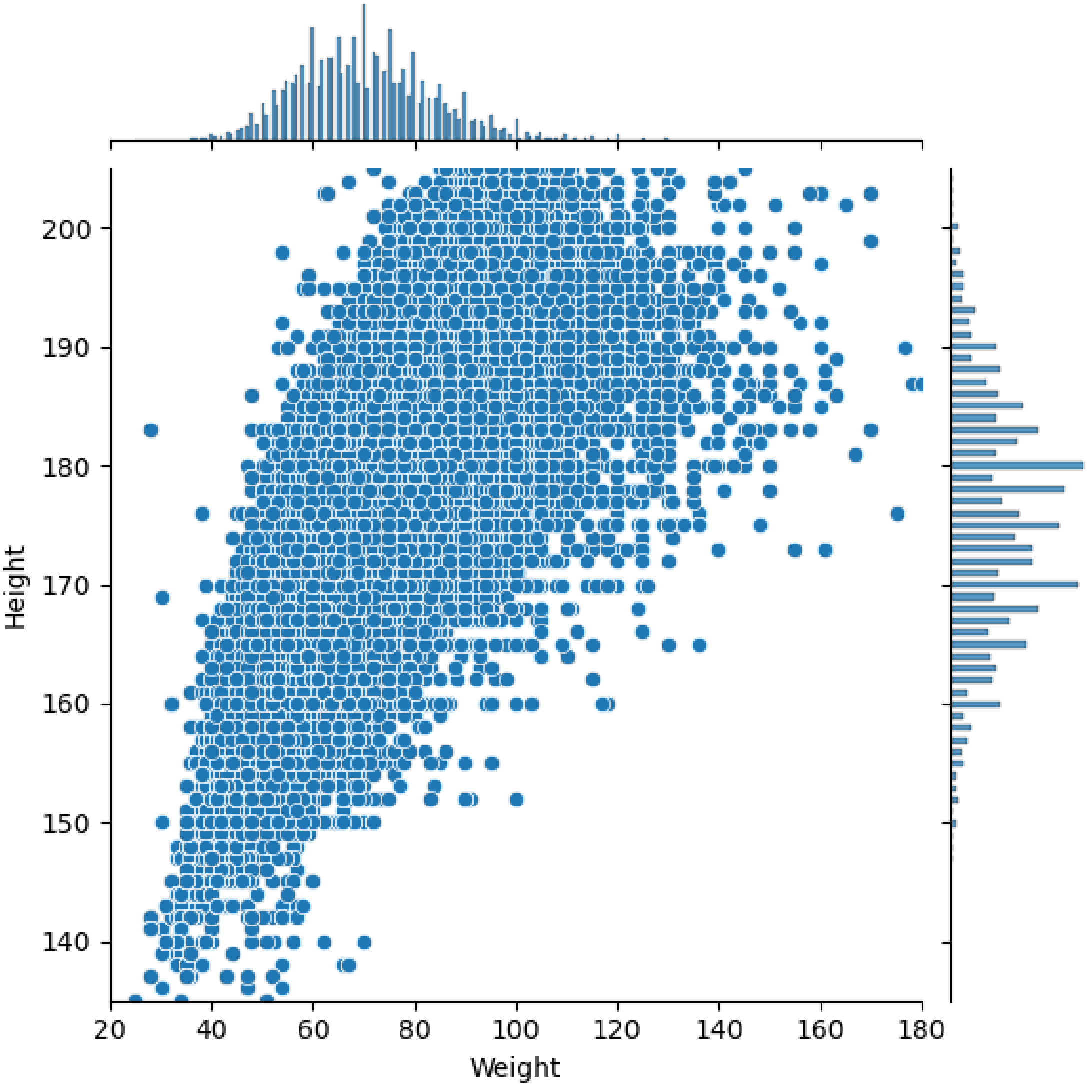
[16]:
```
sns.pairplot(df,kind='reg')
```

[16]:
```
<seaborn.axisgrid.PairGrid at 0x20cfce51db0>
```



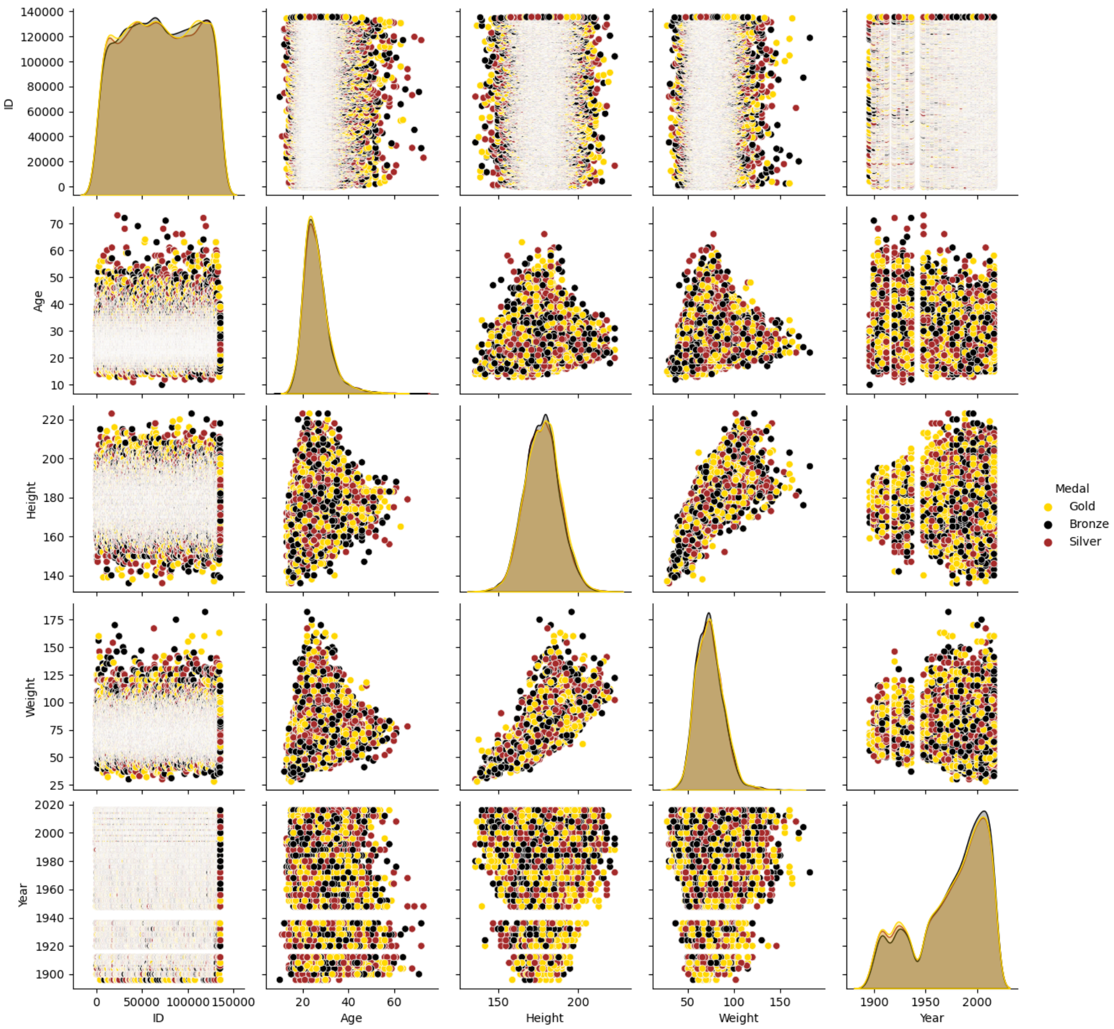
[17]:
```
sns.jointplot(x=df.Weight.dropna(),y=df.Height.
 ↪dropna(),xlim=(20,180),ylim=(135,205))
```

[17]:
```
<seaborn.axisgrid.JointGrid at 0x20c8301aec0>
```



[22]:
```
sns.pairplot(df,hue='Medal',palette=['gold','black','brown'])
```

[22]:
```
<seaborn.axisgrid.PairGrid at 0x26916042b20>
```



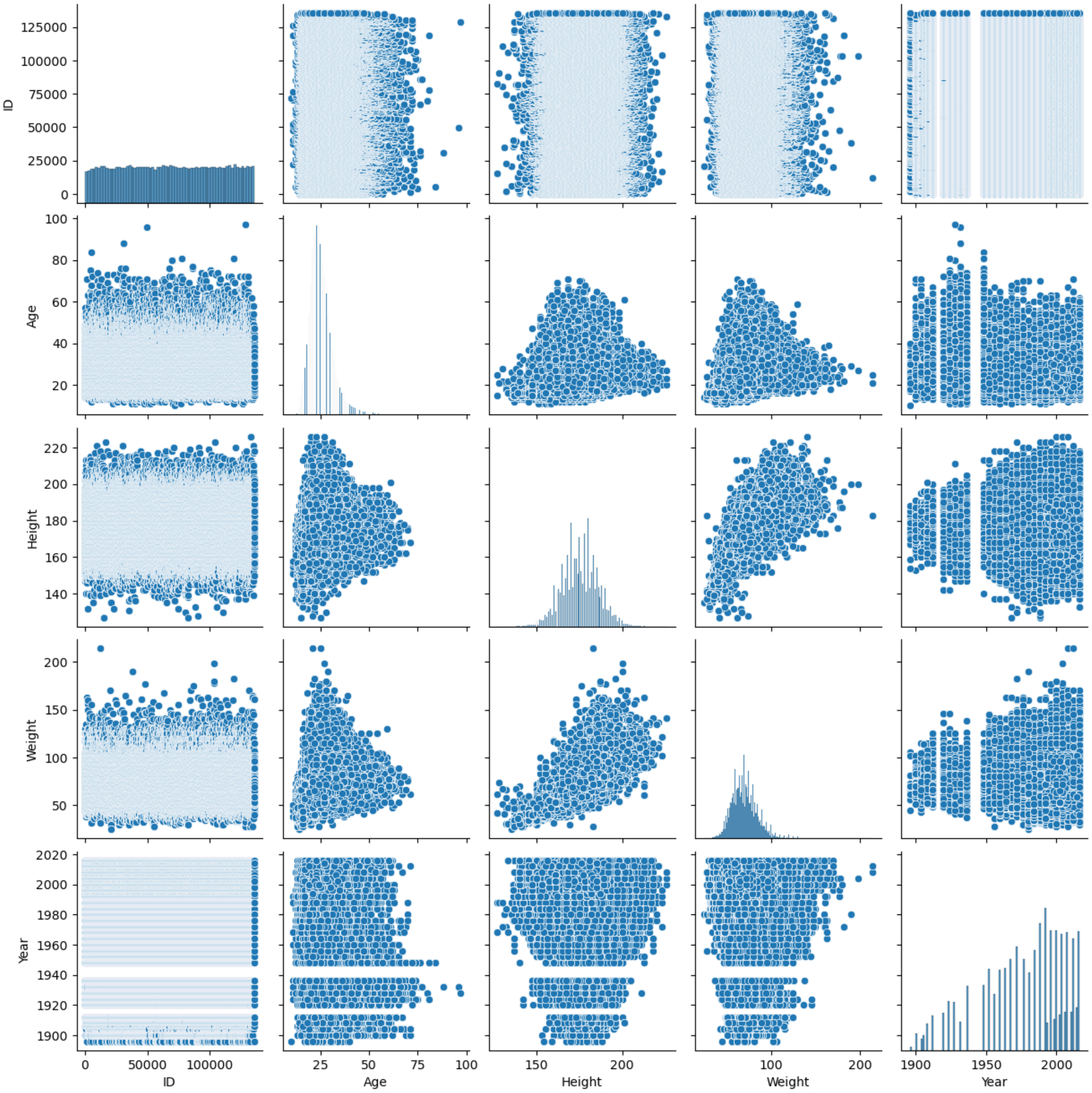
```
[16]: df
```

```
[16]:             ID                     Name Sex   Age  Height  Weight  \
      0            1                A Dijiang   M  24.0   180.0    80.0
      1            2                 A Lamusi   M  23.0   170.0    60.0
      2            3      Gunnar Nielsen Aaby   M  24.0     NaN     NaN
      3            4     Edgar Lindenau Aabye   M  34.0     NaN     NaN
      4            5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
      …            …                      …  ..    …      …
      271111  135569               Andrzej ya   M  29.0   179.0    89.0
      271112  135570                 Piotr ya   M  27.0   176.0    59.0
      271113  135570                 Piotr ya   M  27.0   176.0    59.0
      271114  135571       Tomasz Ireneusz ya   M  30.0   185.0    96.0
      271115  135571       Tomasz Ireneusz ya   M  34.0   185.0    96.0
```

```
                  Team  NOC        Games  Year  Season            City  \
0                China  CHN  1992 Summer  1992  Summer       Barcelona
1                China  CHN  2012 Summer  2012  Summer          London
2              Denmark  DEN  1920 Summer  1920  Summer       Antwerpen
3       Denmark/Sweden  DEN  1900 Summer  1900  Summer           Paris
4          Netherlands  NED  1988 Winter  1988  Winter         Calgary
…                …  …          …   …     …             …
271111        Poland-1  POL  1976 Winter  1976  Winter       Innsbruck
271112          Poland  POL  2014 Winter  2014  Winter           Sochi
271113          Poland  POL  2014 Winter  2014  Winter           Sochi
271114          Poland  POL  1998 Winter  1998  Winter          Nagano
271115          Poland  POL  2002 Winter  2002  Winter  Salt Lake City

                Sport                                     Event Medal
0          Basketball               Basketball Men's Basketball   NaN
1                Judo              Judo Men's Extra-Lightweight   NaN
2            Football                   Football Men's Football   NaN
3          Tug-Of-War               Tug-Of-War Men's Tug-Of-War  Gold
4       Speed Skating          Speed Skating Women's 500 metres   NaN
…               …                                       …   …
271111           Luge                Luge Mixed (Men)'s Doubles   NaN
271112    Ski Jumping  Ski Jumping Men's Large Hill, Individual   NaN
271113    Ski Jumping        Ski Jumping Men's Large Hill, Team   NaN
271114      Bobsleigh                      Bobsleigh Men's Four   NaN
271115      Bobsleigh                      Bobsleigh Men's Four   NaN

[271116 rows x 15 columns]
```

[18]:
```
sns.pairplot(df)
```

[18]:
```
<seaborn.axisgrid.PairGrid at 0x2690f15bf10>
```



[24]:

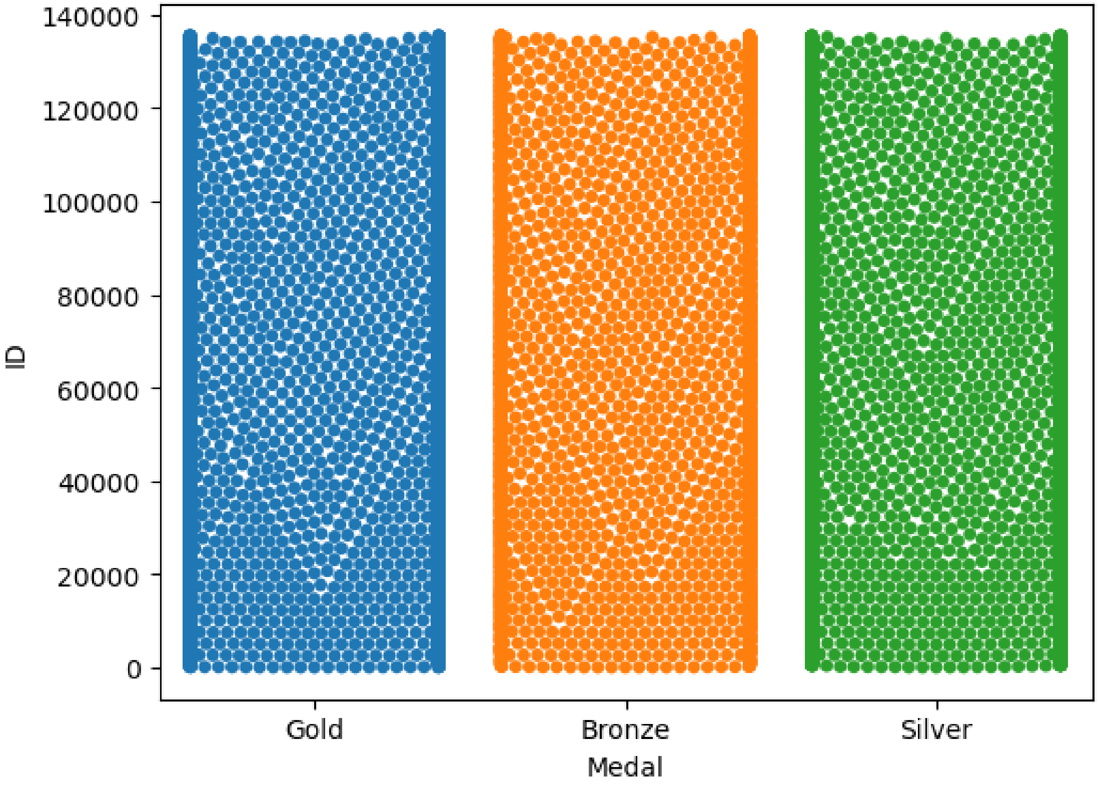
```
sns.swarmplot(x=df.Medal,y=df.ID)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 92.8% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 92.7% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 92.6% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
```

```
warnings.warn(msg, UserWarning)
```

[24]:

```
<AxesSubplot:xlabel='Medal', ylabel='ID'>
```

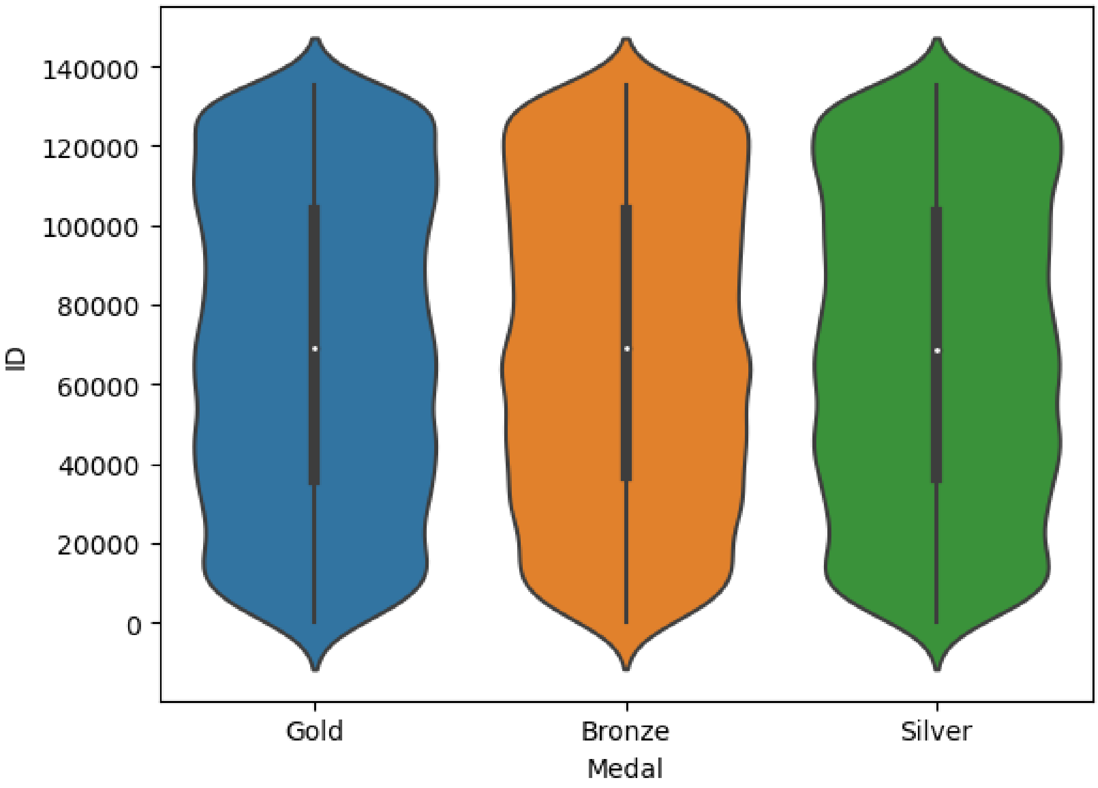


[26]:

```python
import seaborn as sns
import pandas as pd

df = pd.read_csv('D:/DS/resume projects/athlete seaborn/athlete_events.csv')

sns.violinplot(x='Medal', y='ID', data=df)
```

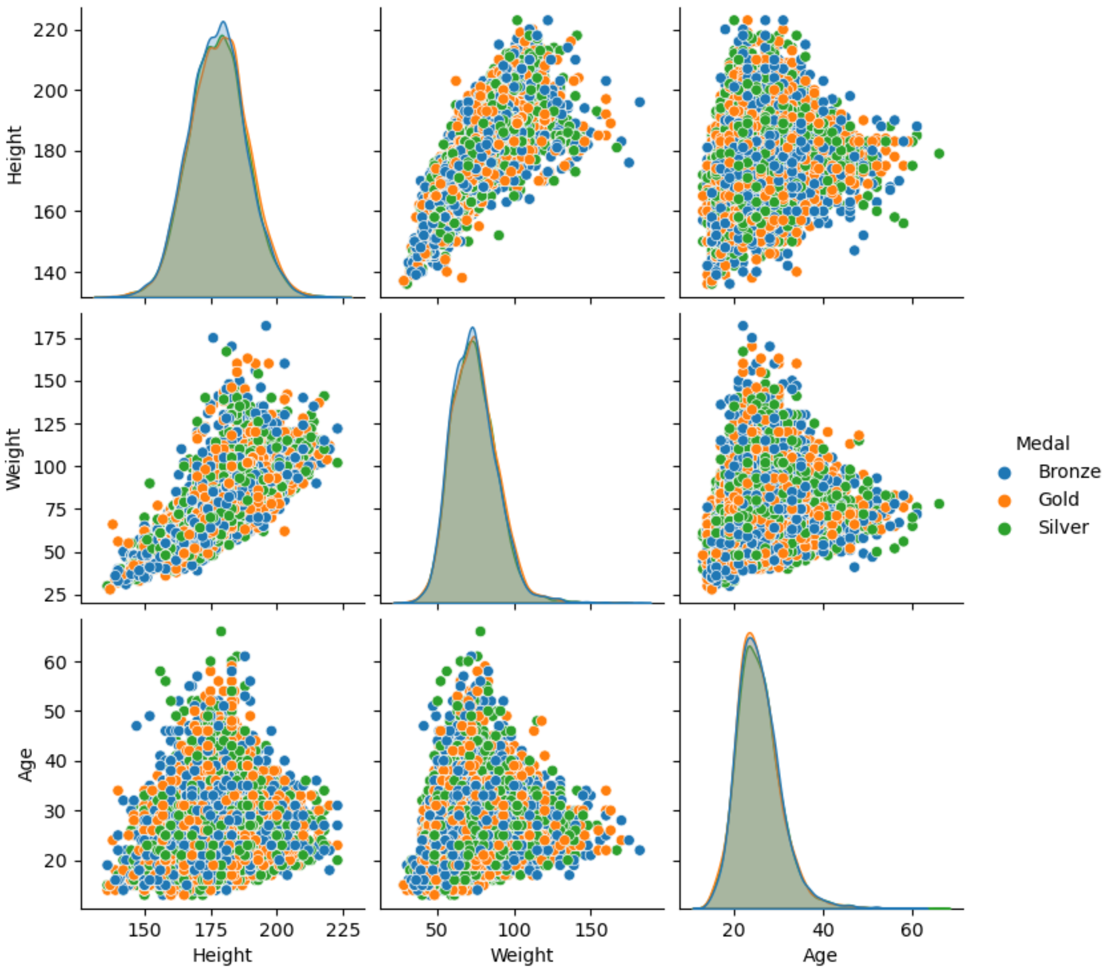
[26]:

```
<AxesSubplot:xlabel='Medal', ylabel='ID'>
```



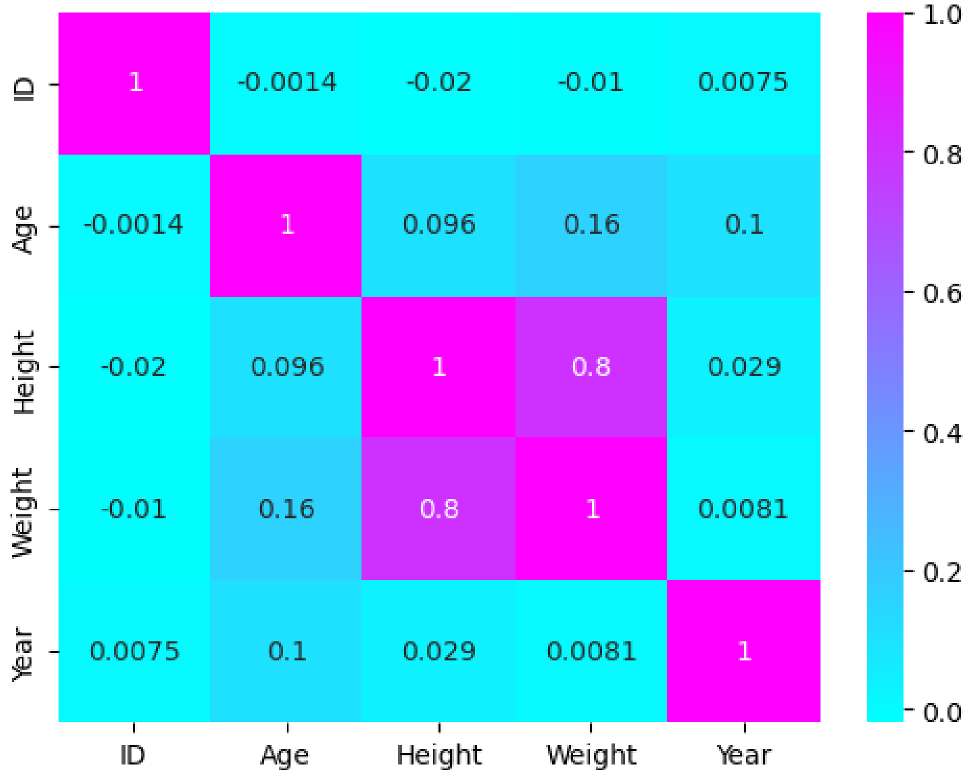
```
[28]: sns.pairplot(df[['Height','Weight','Age','Medal']].dropna(),hue='Medal')
```

```
[28]: <seaborn.axisgrid.PairGrid at 0x2691c787a00>
```



```
[35]: sns.heatmap(df.dropna().corr(),annot=True,cmap='cool')
```

```
[35]: <AxesSubplot:>
```



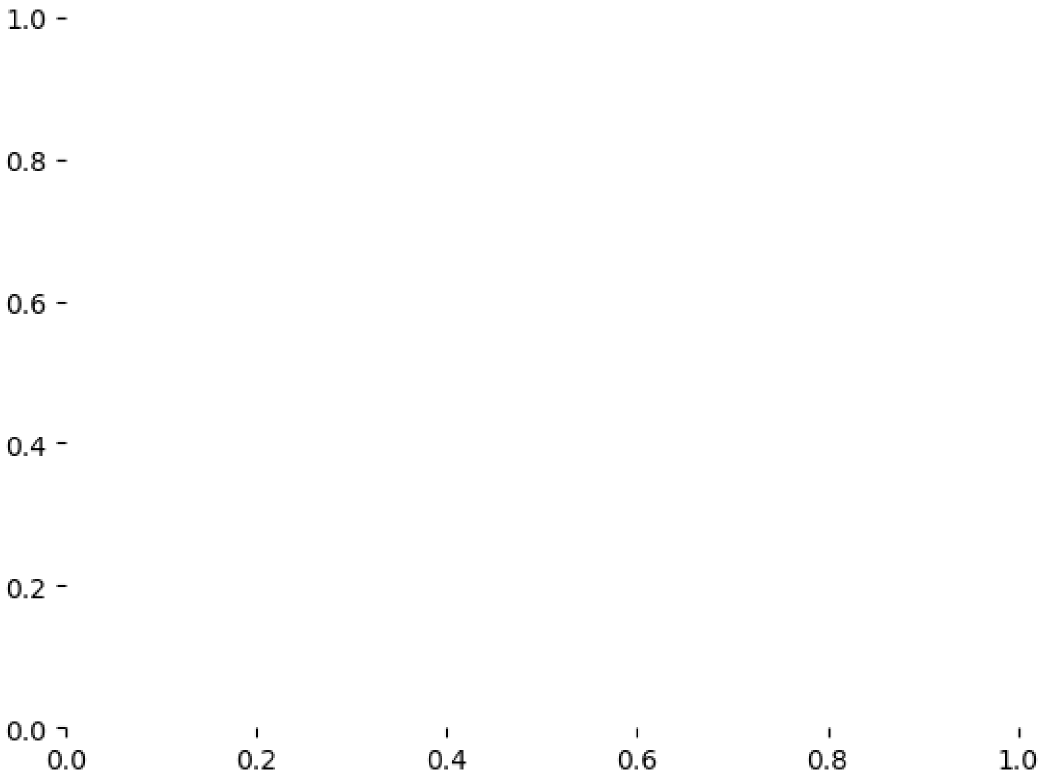
```
[18]: sns.heatmap(df.dropna().corr(),annot=True,cmap='cool',
                 ␣
       ↪x_var=['Age','Height','Weight','Year'],y_var=['Age','Height','Weight','Year'])
```

```
C:\Users\Kundan Mourya\AppData\Local\Temp\ipykernel_14416\1671216070.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.dropna().corr(),annot=True,cmap='cool',
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[18], line 1
----> 1 sns.heatmap(df.dropna().corr(),annot=True,cmap='cool',
      2␣
 ↪            x_var=['Age','Height','Weight','Year'],y_var=['Age','Height','Wei ht','Year'])

File ~\anaconda3\lib\site-packages\seaborn\matrix.py:459, in heatmap(data, vmin ␣
 ↪vmax, cmap, center, robust, annot, fmt, annot_kws, linewidths, linecolor,␣
 ↪cbar, cbar_kws, cbar_ax, square, xticklabels, yticklabels, mask, ax, **kwargs
    457 if square:
    458     ax.set_aspect("equal")
```

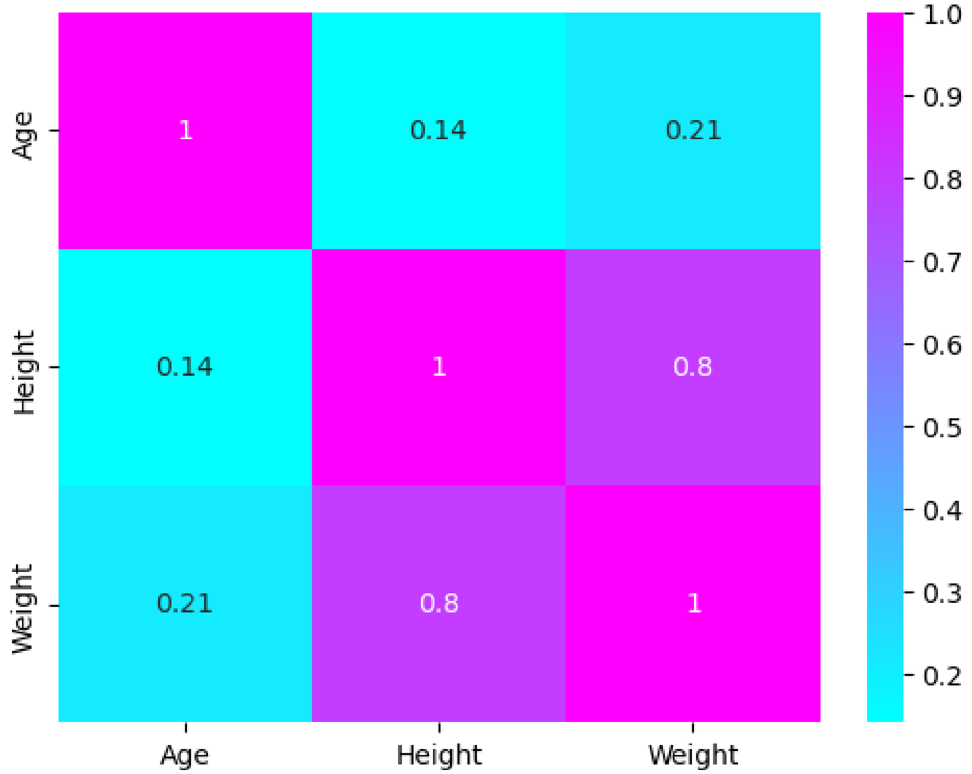
```
--> 459 plotter.plot(ax, cbar_ax, kwargs)
    460 return ax

File ~\anaconda3\lib\site-packages\seaborn\matrix.py:306, in _HeatMapper.
 ↪plot(self, ax, cax, kws)
    303     kws.setdefault("vmax", self.vmax)
    305 # Draw the heatmap
--> 306 mesh = ax.pcolormesh(self.plot_data, cmap=self.cmap, **kws)
    308 # Set the axis limits
    309 ax.set(xlim=(0, self.data.shape[1]), ylim=(0, self.data.shape[0]))

File ~\anaconda3\lib\site-packages\matplotlib\__init__.py:1442, in␣
 ↪_preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1439 @functools.wraps(func)
   1440 def inner(ax, *args, data=None, **kwargs):
   1441     if data is None:
-> 1442         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1444     bound = new_sig.bind(ax, *args, **kwargs)
   1445     auto_label = (bound.arguments.get(label_namer)
   1446                   or bound.kwargs.get(label_namer))

File ~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6229, in Axes.
 ↪pcolormesh(self, alpha, norm, cmap, vmin, vmax, shading, antialiased, *args,␣
 ↪**kwargs)
   6225     C = C.ravel()
   6227 kwargs.setdefault('snap', mpl.rcParams['pcolormesh.snap'])
-> 6229 collection = mcoll.QuadMesh(
   6230     coords, antialiased=antialiased, shading=shading,
   6231     array=C, cmap=cmap, norm=norm, alpha=alpha, **kwargs)
   6232 collection._scale_norm(norm, vmin, vmax)
   6234 coords = coords.reshape(-1, 2)  # flatten the grid structure; keep x, y

File ~\anaconda3\lib\site-packages\matplotlib\collections.py:1939, in QuadMesh.
 ↪__init__(self, coordinates, antialiased, shading, **kwargs)
   1936 self._bbox.update_from_data_xy(self._coordinates.reshape(-1, 2))
   1937 # super init delayed after own init because array kwarg requires
   1938 # self._coordinates and self._shading
-> 1939 super().__init__(**kwargs)
   1940 self.set_mouseover(False)

File ~\anaconda3\lib\site-packages\matplotlib\_api\deprecation.py:454, in␣
 ↪make_keyword_only.<locals>.wrapper(*args, **kwargs)
    448 if len(args) > name_idx:
    449     warn_deprecated(
    450         since, message="Passing the %(name)s %(obj_type)s "
    451         "positionally is deprecated since Matplotlib %(since)s; the "
    452         "parameter will become keyword-only %(removal)s.",
    453         name=name, obj_type=f"parameter of {func.__name__}()")
```

```
--> 454 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\collections.py:201, in Collection
  ↪__init__(self, edgecolors, facecolors, linewidths, linestyles, capstyle,␣
  ↪joinstyle, antialiaseds, offsets, offset_transform, norm, cmap, pickradius,␣
  ↪hatch, urls, zorder, **kwargs)
    198 self._offset_transform = offset_transform
    200 self._path_effects = None
--> 201 self._internal_update(kwargs)
    202 self._paths = None

File ~\anaconda3\lib\site-packages\matplotlib\artist.py:1223, in Artist.
  ↪_internal_update(self, kwargs)
   1216 def _internal_update(self, kwargs):
   1217     """
   1218     Update artist properties without prenormalizing them, but generatin
   1219     errors as if calling `set`.
   1220 
   1221     The lack of prenormalization is to maintain backcompatibility.
   1222     """
-> 1223     return self._update_props(
   1224␣
  ↪        kwargs, "{cls.__name__}.set() got an unexpected keyword argument "
   1225         "{prop_name!r}")

File ~\anaconda3\lib\site-packages\matplotlib\artist.py:1197, in Artist.
  ↪_update_props(self, props, errfmt)
   1195             func = getattr(self, f"set_{k}", None)
   1196             if not callable(func):
-> 1197                 raise AttributeError(
   1198                     errfmt.format(cls=type(self), prop_name=k))
   1199             ret.append(func(v))
   1200 if ret:

AttributeError: QuadMesh.set() got an unexpected keyword argument 'x_var'
```

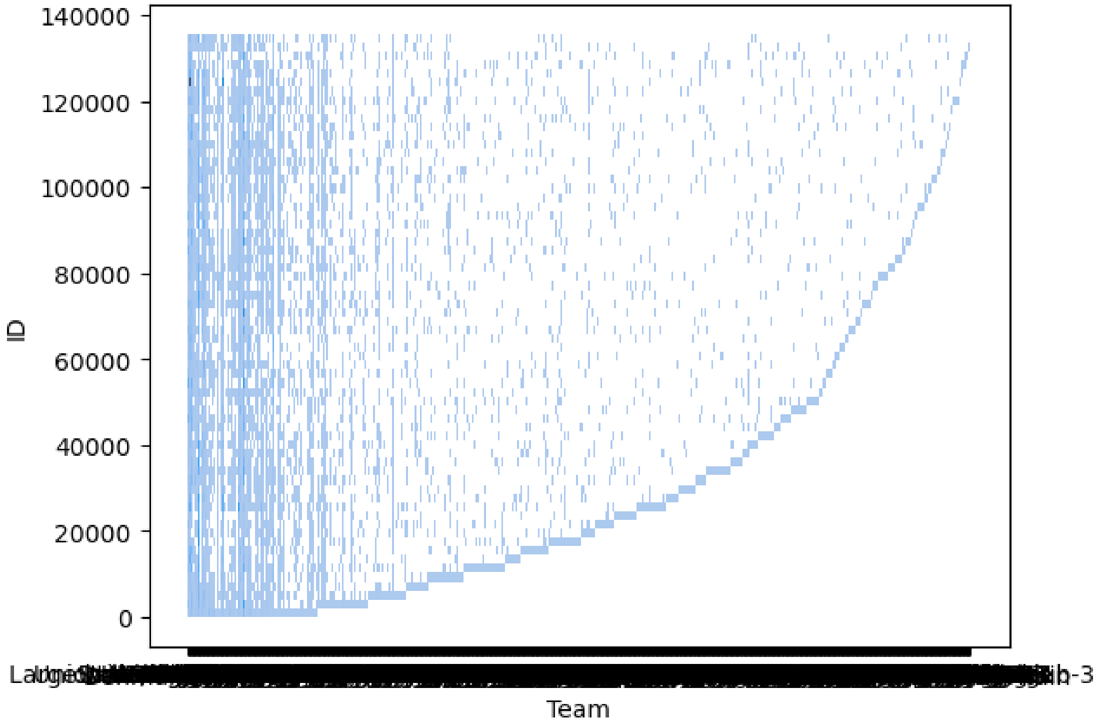


[19]:
```
sns.heatmap(df[['Age','Height','Weight']].dropna().
 ↪corr(),annot=True,cmap='cool')
```

[19]:
```
<Axes: >
```



[17]:
```
#medalistcount by top 20 country
sns.histplot(x=df.Team,y=df.ID)
plt.show()
```



```
[20]: df
```

```
[20]:             ID                        Name Sex   Age  Height  Weight  \
      0              1                   A Dijiang   M  24.0   180.0    80.0   
      1              2                    A Lamusi   M  23.0   170.0    60.0   
      2              3         Gunnar Nielsen Aaby   M  24.0     NaN     NaN   
      3              4        Edgar Lindenau Aabye   M  34.0     NaN     NaN   
      4              5    Christine Jacoba Aaftink   F  21.0   185.0    82.0   
      …        …                       …   ..   …     …       …             
      271111  135569                  Andrzej ya   M  29.0   179.0    89.0   
      271112  135570                    Piotr ya   M  27.0   176.0    59.0   
      271113  135570                    Piotr ya   M  27.0   176.0    59.0   
      271114  135571          Tomasz Ireneusz ya   M  30.0   185.0    96.0   
      271115  135571          Tomasz Ireneusz ya   M  34.0   185.0    96.0   

                        Team  NOC        Games  Year  Season          City  \
      0                China  CHN  1992 Summer  1992  Summer     Barcelona   
      1                China  CHN  2012 Summer  2012  Summer        London   
      2              Denmark  DEN  1920 Summer  1920  Summer     Antwerpen   
      3       Denmark/Sweden  DEN  1900 Summer  1900  Summer         Paris   
      4          Netherlands  NED  1988 Winter  1988  Winter       Calgary   
      …                 …  …          …   …     …             …             
      271111        Poland-1  POL  1976 Winter  1976  Winter     Innsbruck   
```

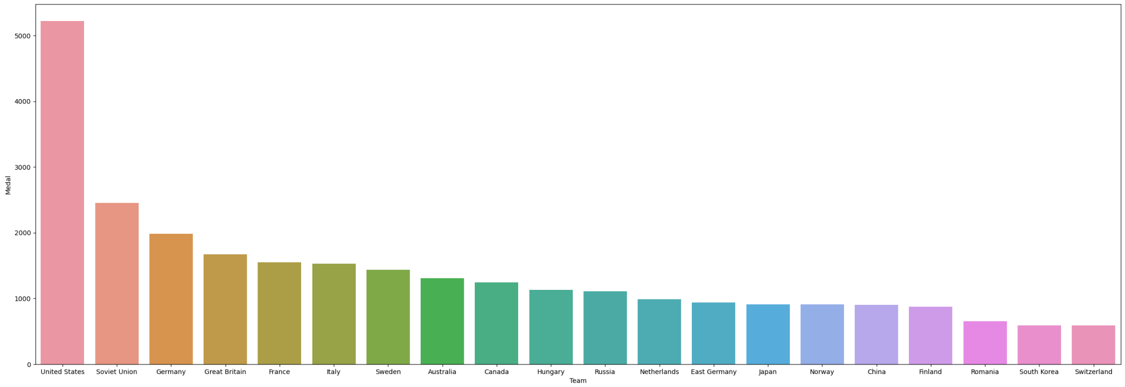
```
271112          Poland  POL  2014 Winter  2014  Winter          Sochi
271113          Poland  POL  2014 Winter  2014  Winter          Sochi
271114          Poland  POL  1998 Winter  1998  Winter         Nagano
271115          Poland  POL  2002 Winter  2002  Winter  Salt Lake City

                Sport                                     Event Medal
0          Basketball               Basketball Men's Basketball   NaN
1                Judo              Judo Men's Extra-Lightweight   NaN
2            Football                   Football Men's Football   NaN
3          Tug-Of-War               Tug-Of-War Men's Tug-Of-War  Gold
4       Speed Skating          Speed Skating Women's 500 metres   NaN
…                 …                                         …    …
271111           Luge                Luge Mixed (Men)'s Doubles   NaN
271112    Ski Jumping  Ski Jumping Men's Large Hill, Individual   NaN
271113    Ski Jumping        Ski Jumping Men's Large Hill, Team   NaN
271114      Bobsleigh                      Bobsleigh Men's Four   NaN
271115      Bobsleigh                      Bobsleigh Men's Four   NaN

[271116 rows x 15 columns]
```

[21]:

```
df1=df.groupby('Team').count()['Medal'].sort_values(ascending=False)
df1=df1[0:20]
df1=df1.reset_index()
```

[22]:

```
plt.figure(figsize=(30,10))
sns.barplot(x=df1.Team,y=df1.Medal)
plt.show()
```



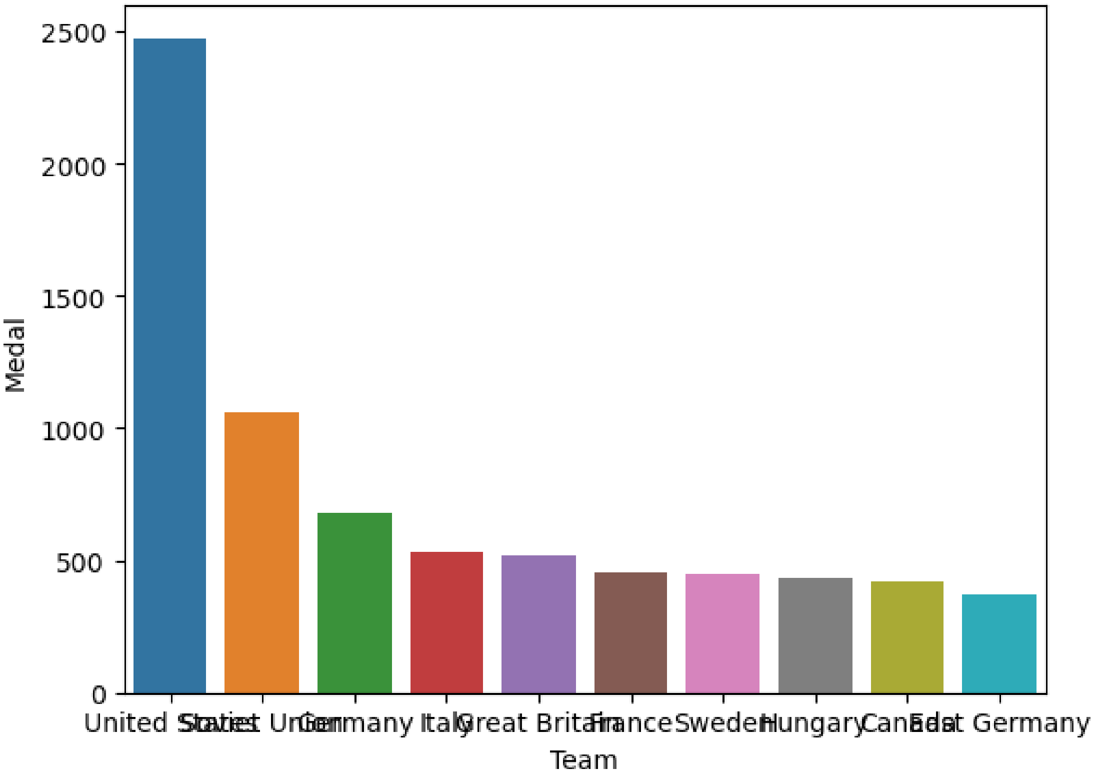
[23]:

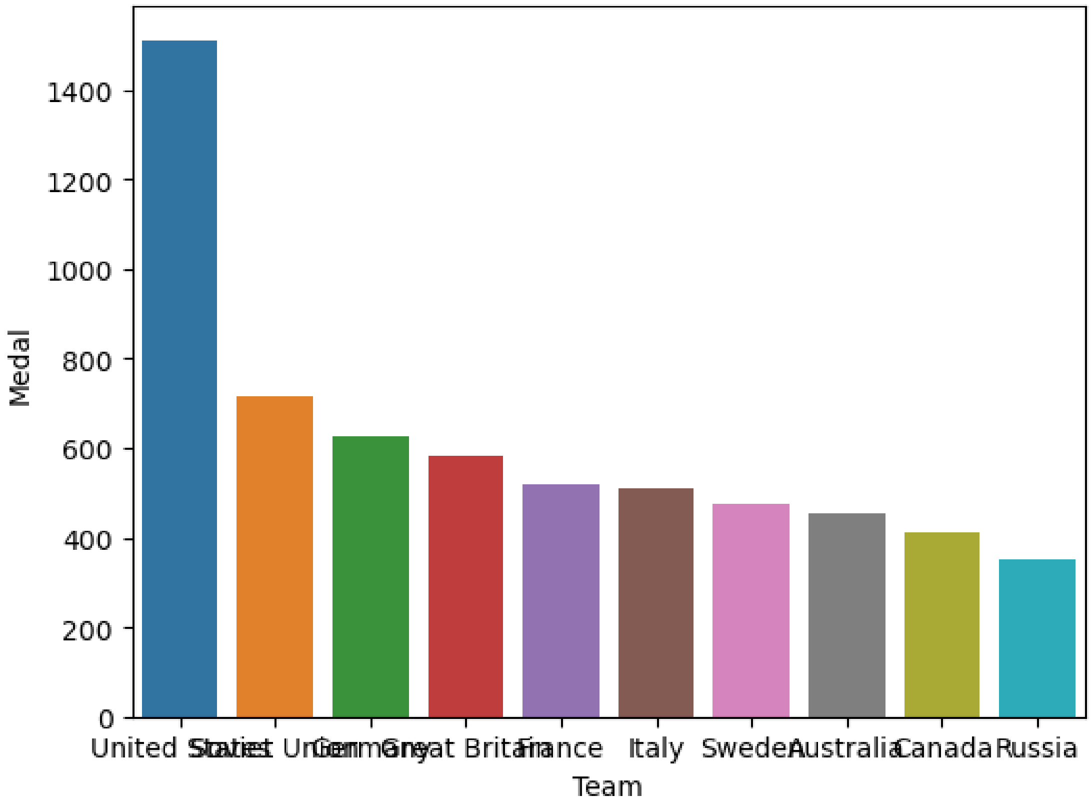
```
dfg=df[df['Medal']=='Gold'].groupby('Team').count()['Medal'].
 ↪sort_values(ascending=False).reset_index().head(10)
dfs=df[df['Medal']=='Silver'].groupby('Team').count()['Medal'].
 ↪sort_values(ascending=False).reset_index().head(10)
```

```
dfb=df[df['Medal']=='Bronze'].groupby('Team').count()['Medal'].
 ↪sort_values(ascending=False).reset_index().head(10)
```

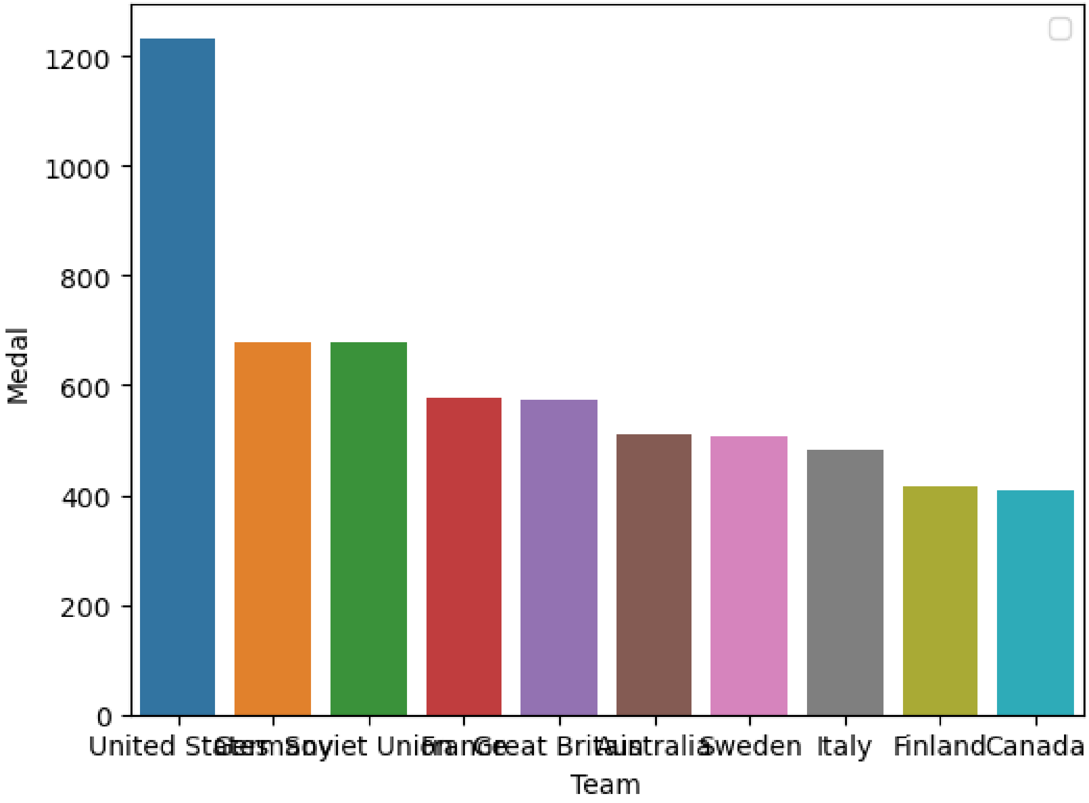
[24]:

```
sns.barplot(x=dfg.Team.dropna(),y=dfg.Medal.dropna())
plt.show()
sns.barplot(x=dfs.Team.dropna(),y=dfs.Medal.dropna())
plt.show()
sns.barplot(x=dfb.Team.dropna(),y=dfb.Medal.dropna())
plt.legend()
plt.show()
```





```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



[25]:

```
plt.bar(x=dfg.Team.dropna(),y=dfg.Medal.dropna(),width=0.5)
sns.barplot(x=dfs.Team.dropna(),y=dfs.Medal.dropna())
sns.barplot(x=dfb.Team.dropna(),y=dfb.Medal.dropna())
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 plt.bar(x=dfg.Team.dropna(),y=dfg.Medal.dropna(),width=0.5)
      2 sns.barplot(x=dfs.Team.dropna(),y=dfs.Medal.dropna())
      3 sns.barplot(x=dfb.Team.dropna(),y=dfb.Medal.dropna())

TypeError: bar() missing 1 required positional argument: 'height'
```

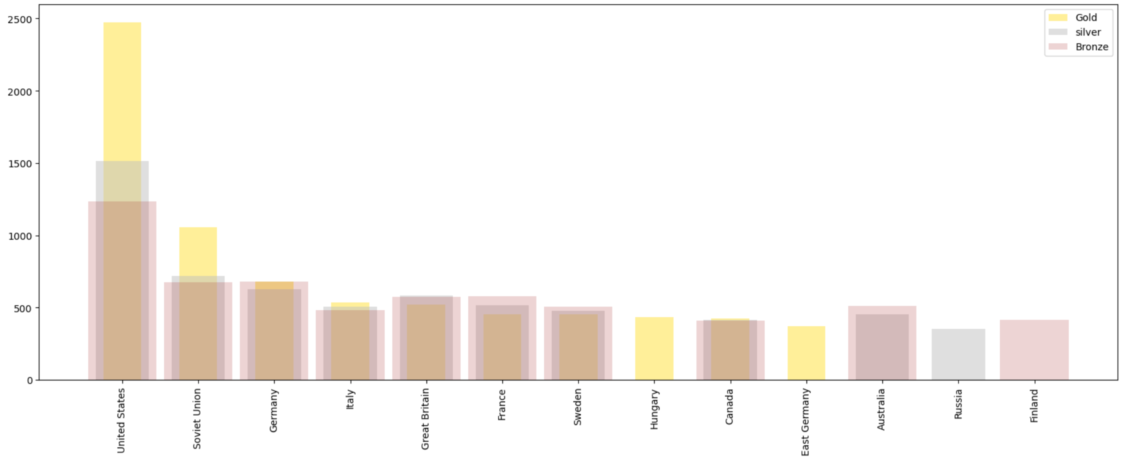
[173]:

```
plt.bar(x=dfg.Team.dropna(),y=dfg.Medal.dropna(),width=5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_18752\4132119427.py in <module>
----> 1 plt.bar(x=dfg.Team.dropna(),y=dfg.Medal.dropna(),width=5)
```

```
TypeError: bar() missing 1 required positional argument: 'height'
```

```
[26]: plt.figure(figsize=(20,7))
      plt.bar(x=dfg.Team, height=dfg.Medal, width=0.5,alpha=0.
       ↪4,color='Gold',label='Gold')
      plt.bar(x=dfs.Team,height=dfs.Medal,width=0.7,alpha=0.
       ↪5,color='Silver',label='silver')
      plt.bar(x=dfb.Team,height=dfb.Medal,width=0.9,alpha=0.
       ↪2,color='brown',label='Bronze')
      plt.legend()
      plt.xticks(rotation=90)
      plt.show()
```



```
[27]: plt.bar(x=dfg.Team.dropna(),y=dfg.Medal.dropna(),width=0.5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 plt.bar(x=dfg.Team.dropna(),y=dfg.Medal.dropna(),width=0.5)

TypeError: bar() missing 1 required positional argument: 'height'
```

```
[28]: g=np.arange(len(dfg))-.2
      s=np.arange(len(dfg))
      b=np.arange(len(dfg))+.2
```

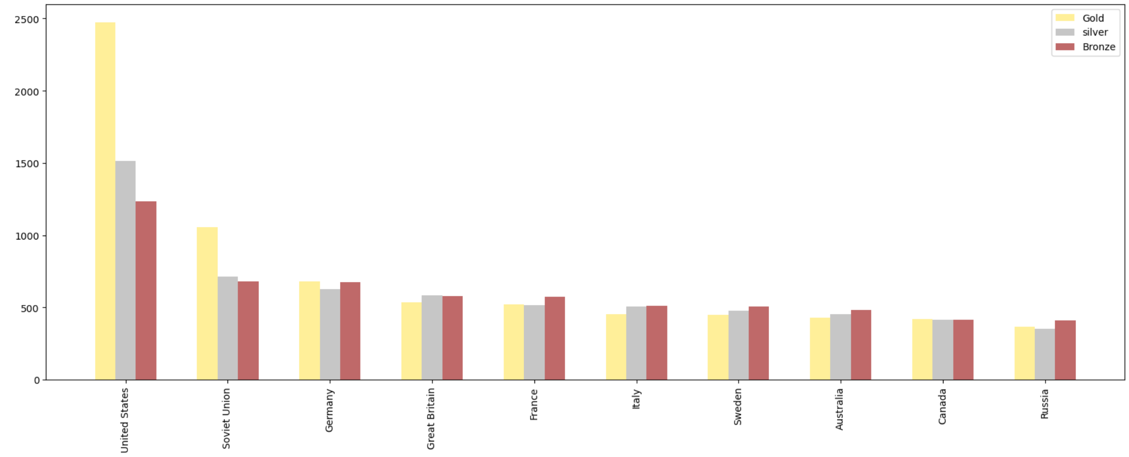
```
[29]: g
```

[29]:
```
array([-0.2,  0.8,  1.8,  2.8,  3.8,  4.8,  5.8,  6.8,  7.8,  8.8])
```

[30]:
```
b
```

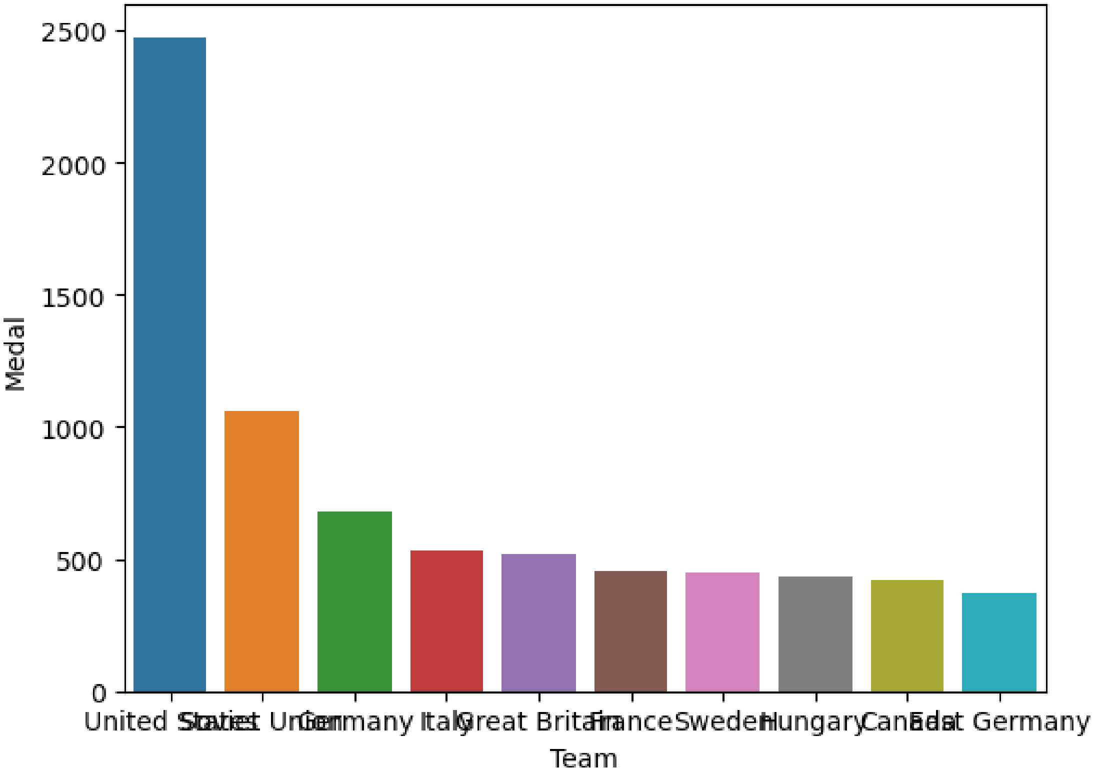
[30]:
```
array([0.2, 1.2, 2.2, 3.2, 4.2, 5.2, 6.2, 7.2, 8.2, 9.2])
```

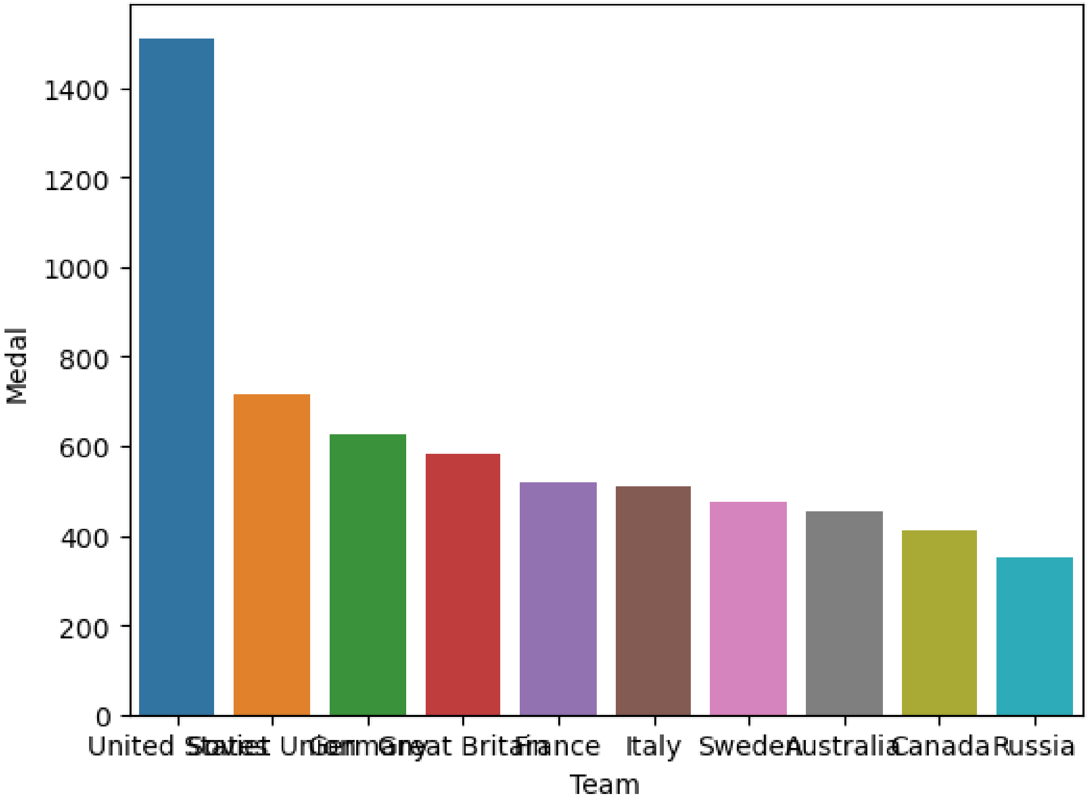
[31]:
```
plt.figure(figsize=(20,7))
plt.bar(x=g, height=dfg.Medal, width=0.2,alpha=0.4,color='Gold',label='Gold')
plt.bar(x=dfs.Team,height=dfs.Medal,width=0.2,alpha=0.
  ↪9,color='Silver',label='silver')
plt.bar(x=b,height=dfb.Medal,width=0.2,alpha=0.7,color='brown',label='Bronze')
plt.legend()
plt.xticks(dfs.Team,rotation=90)
plt.show()
```



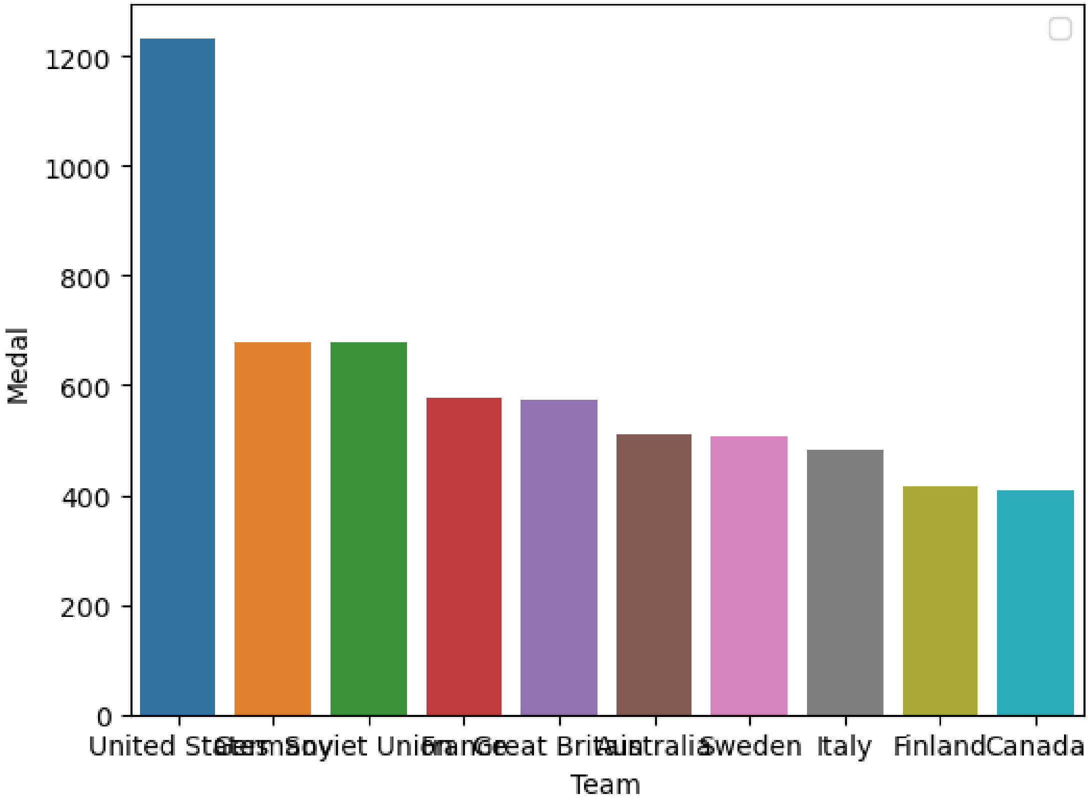
[ ]:
```

```

[32]:
```
sns.barplot(x=dfg.Team.dropna(),y=dfg.Medal.dropna())
plt.show()
sns.barplot(x=dfs.Team.dropna(),y=dfs.Medal.dropna())
plt.show()
sns.barplot(x=dfb.Team.dropna(),y=dfb.Medal.dropna())
plt.legend()
plt.show()
```


Medal
2500
2000
1500
1000
500
0
United States
Soviet Union
Germany
Italy
Great Britain
France
Sweden
Hungary
Canada
East Germany
Team



```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



```
[33]: s=np.arange(len(dfg))
      g=s-.2
      b=s+.2
```

```
[34]: s,g,b
```

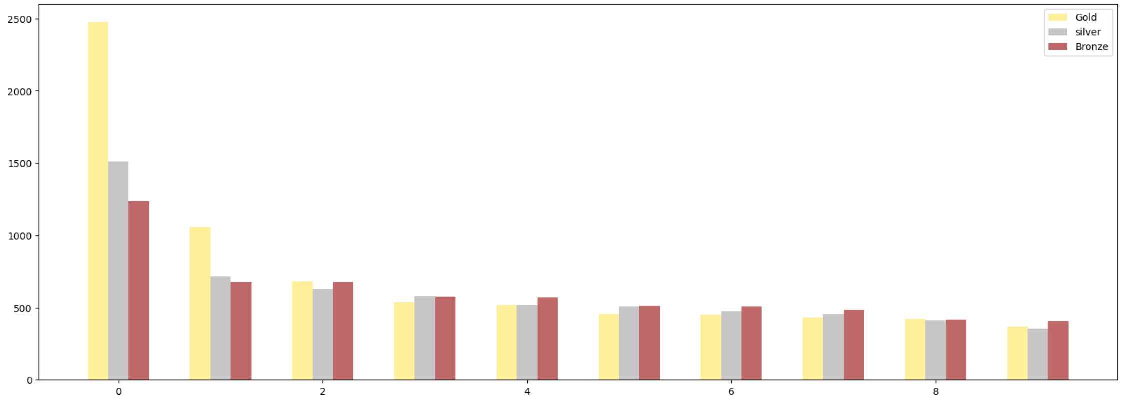
```
[34]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
       array([-0.2,  0.8,  1.8,  2.8,  3.8,  4.8,  5.8,  6.8,  7.8,  8.8]),
       array([0.2, 1.2, 2.2, 3.2, 4.2, 5.2, 6.2, 7.2, 8.2, 9.2]))
```

```
[35]: plt.figure(figsize=(20,7))
      a=np.arange(len(dfg))
      plt.bar(x=g, height=dfg.Medal, width=0.2,alpha=0.4,color='Gold',label='Gold')
      plt.bar(x=s,height=dfs.Medal,width=0.2,alpha=0.9,color='Silver',label='silver')
      plt.bar(x=b,height=dfb.Medal,width=0.2,alpha=0.7,color='brown',label='Bronze')
      plt.legend()
      plt.xticks(dfs.Team,rotation=90)
      plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

```
File ~\anaconda3\lib\site-packages\matplotlib\axis.py:1736, in Axis.
 ↪convert_units(self, x)
   1735 try:
-> 1736     ret = self.converter.convert(x, self.units, self)
   1737 except Exception as e:

File ~\anaconda3\lib\site-packages\matplotlib\category.py:49, in␣
 ↪StrCategoryConverter.convert(value, unit, axis)
     48 if unit is None:
---> 49     raise ValueError(
     50         'Missing category information for StrCategoryConverter; '
     51         'this might be caused by unintendedly mixing categorical and '
     52         'numeric data')
     53 StrCategoryConverter._validate_unit(unit)

ValueError: Missing category information for StrCategoryConverter; this might be␣
 ↪caused by unintendedly mixing categorical and numeric data

The above exception was the direct cause of the following exception:

ConversionError                           Traceback (most recent call last)
Cell In[35], line 7
      5 plt.bar(x=b,height=dfb.Medal,width=0.2,alpha=0.
 ↪7,color='brown',label='Bronze')
      6 plt.legend()
----> 7 plt.xticks(dfs.Team,rotation=90)
      8 plt.show()

File ~\anaconda3\lib\site-packages\matplotlib\pyplot.py:1859, in xticks(ticks,␣
 ↪labels, minor, **kwargs)
   1856         raise TypeError("xticks(): Parameter 'labels' can't be set "
   1857                         "without setting 'ticks'")
   1858 else:
-> 1859     locs = ax.set_xticks(ticks, minor=minor)
   1861 if labels is None:
   1862     labels = ax.get_xticklabels(minor=minor)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:74, in␣
 ↪_axis_method_wrapper.__set_name__.<locals>.wrapper(self, *args, **kwargs)
     73 def wrapper(self, *args, **kwargs):
---> 74     return get_method(self)(*args, **kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\axis.py:2078, in Axis.
 ↪set_ticks(self, ticks, labels, minor, **kwargs)
   2075 if labels is None and kwargs:
   2076     raise ValueError('labels argument cannot be None when '
   2077                      'kwargs are passed')
-> 2078 result = self._set_tick_locations(ticks, minor=minor)
```

```
   2079 if labels is not None:
   2080     self.set_ticklabels(labels, minor=minor, **kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\axis.py:2018, in Axis.
 ↪_set_tick_locations(self, ticks, minor)
   2014 def _set_tick_locations(self, ticks, *, minor=False):
   2015     # see docstring of set_ticks
   2016
   2017     # XXX if the user changes units, the information will be lost here
-> 2018     ticks = self.convert_units(ticks)
   2019     locator = mticker.FixedLocator(ticks)  # validate ticks early.
   2020     for name, axis in self.axes._axis_map.items():

File ~\anaconda3\lib\site-packages\matplotlib\axis.py:1738, in Axis.
 ↪convert_units(self, x)
   1736     ret = self.converter.convert(x, self.units, self)
   1737 except Exception as e:
-> 1738     raise munits.ConversionError('Failed to convert value(s) to axis '
   1739                                  f'units: {x!r}') from e
   1740 return ret

ConversionError: Failed to convert value(s) to axis units: 0    United States
1     Soviet Union
2          Germany
3    Great Britain
4           France
5            Italy
6           Sweden
7        Australia
8           Canada
9           Russia
Name: Team, dtype: object
```



[36]:
```
a+.2
```

[36]:
```
array([0.2, 1.2, 2.2, 3.2, 4.2, 5.2, 6.2, 7.2, 8.2, 9.2])
```

[37]:
```
dfgg=df[df['Medal']=='Gold'].groupby('Team').count()['Medal'].
  ↪sort_values(ascending=False).reset_index().head(20)
```

[38]:
```
dfgg
```

[38]:
```
              Team  Medal
0    United States   2474
1     Soviet Union   1058
2          Germany    679
3            Italy    535
4    Great Britain    519
5           France    455
6           Sweden    451
7          Hungary    432
8           Canada    422
9     East Germany    369
10          Russia    366
11       Australia    342
12           China    308
13          Norway    299
14     Netherlands    277
15           Japan    247
16     South Korea    211
17         Finland    198
18         Denmark    168
19            Cuba    164
```

[39]:
```
dfg=df[df['Medal']=='Gold'].groupby('Team').count()['Medal'].
  ↪sort_values(ascending=False).reset_index().head(25)
dfs=df[df['Medal']=='Silver'].groupby('Team').count()['Medal'].
  ↪sort_values(ascending=False).reset_index().head(25)
dfb=df[df['Medal']=='Bronze'].groupby('Team').count()['Medal'].
  ↪sort_values(ascending=False).reset_index().head(25)
```

[40]:
```
dff=pd.merge(dfg, dfs, how='left', on='Team')
```

[41]:
```
dff=pd.merge(dff,dfb,how='left')
```

[42]:
```
dff=dff.rename(columns={'Medal_x':'Gold','Medal_y':'Silver','Medal':'Bronze'})
```

[43]:
```
dff
```

[43]:

```
              Team  Gold  Silver  Bronze
0    United States  2474  1512.0  1233.0
1     Soviet Union  1058   716.0   677.0
2          Germany   679   627.0   678.0
3            Italy   535   508.0   484.0
4    Great Britain   519   582.0   572.0
5           France   455   518.0   577.0
6           Sweden   451   476.0   507.0
7          Hungary   432   330.0   365.0
8           Canada   422   413.0   408.0
9     East Germany   369   309.0   263.0
10          Russia   366   351.0   393.0
11       Australia   342   453.0   511.0
12           China   308   325.0   268.0
13          Norway   299   330.0   281.0
14     Netherlands   277   321.0   390.0
15           Japan   247   307.0   357.0
16     South Korea   211   222.0   159.0
17         Finland   198   263.0   415.0
18         Denmark   168   223.0   162.0
19            Cuba   164     NaN     NaN
20         Romania   161   200.0   290.0
21    West Germany   155   184.0   219.0
22     Switzerland   144   213.0   231.0
23           India   138     NaN     NaN
24      Yugoslavia   130     NaN     NaN
```

[44]:

```
dff.set_index('Team')
```

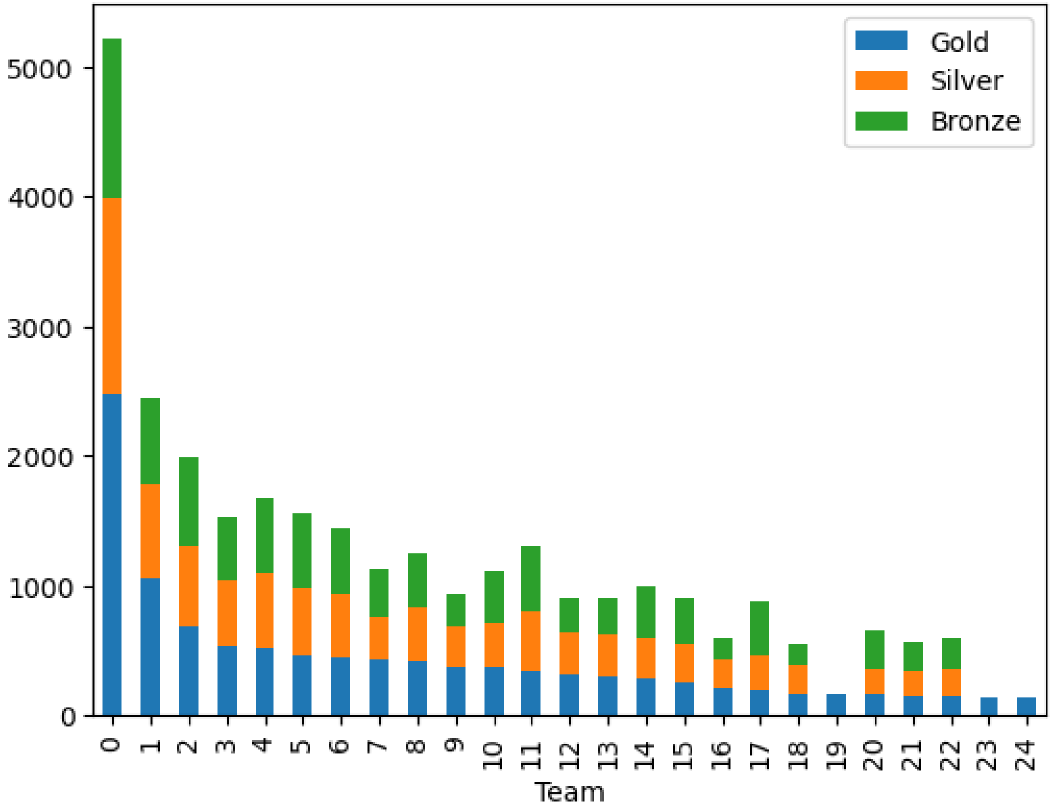
[44]:

```
               Gold  Silver  Bronze
Team
United States  2474  1512.0  1233.0
Soviet Union   1058   716.0   677.0
Germany         679   627.0   678.0
Italy           535   508.0   484.0
Great Britain   519   582.0   572.0
France          455   518.0   577.0
Sweden          451   476.0   507.0
Hungary         432   330.0   365.0
Canada          422   413.0   408.0
East Germany    369   309.0   263.0
Russia          366   351.0   393.0
Australia       342   453.0   511.0
China           308   325.0   268.0
Norway          299   330.0   281.0
Netherlands     277   321.0   390.0
Japan           247   307.0   357.0
```

```
South Korea     211   222.0   159.0
Finland         198   263.0   415.0
Denmark         168   223.0   162.0
Cuba            164     NaN     NaN
Romania         161   200.0   290.0
West Germany    155   184.0   219.0
Switzerland     144   213.0   231.0
India           138     NaN     NaN
Yugoslavia      130     NaN     NaN
```

[45]:

```
dff.plot(kind='bar',stacked=True)
plt.xlabel('Team')
plt.show()
```



[46]:

```
pd.pivot_table(df,index='Team',aggfunc={df.Medal=='Gold':'count',df.
  ↪Medal=='Silver':'count'})
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[46], line 1
```

```
----> 1 pd.pivot_table(df,index='Team',aggfunc={df.Medal=='Gold':'count',df.
  ↪Medal=='Silver':'count'})

TypeError: unhashable type: 'Series'
```

[47]:
```
k=pd.pivot_table(df, index='Team', aggfunc={'Medal': [['Gold', lambda x: sum(x␣
  ↪== 'Gold')],['Silver', lambda x: sum(x == 'Silver')],['Bronze', lambda x:␣
  ↪sum(x == 'Bronze')]]})#.sort_values(by=('Medal'),ascending=False)
k=k.sort_values(by=('Medal','Gold'),ascending=False)
k
```

[47]:
```
                Medal
               Bronze  Gold Silver
Team
United States    1233  2474   1512
Soviet Union      677  1058    716
Germany           678   679    627
Italy             484   535    508
Great Britain     572   519    582
…                  …     …     …
Hakahana            0     0      0
Hamburg             0     0      0
Hannover            0     0      0
Harmony             0     0      0
rn-2                0     0      0

[1184 rows x 3 columns]
```

[48]:
```
pivot_table = pd.pivot_table(df, index='Team', aggfunc={'Medal': ['count',␣
  ↪lambda x: sum(x == 'Gold'), lambda x: sum(x == 'Silver'), lambda x: sum(x ==␣
  ↪'Bronze')]})
sorted_pivot_table = pivot_table.sort_values(by=('Medal', 'count'),␣
  ↪ascending=False)
sorted_pivot_table
```

[48]:
```
                     Medal
              <lambda_0> <lambda_1> <lambda_2> count
Team
United States       2474       1512       1233  5219
Soviet Union        1058        716        677  2451
Germany              679        627        678  1984
Great Britain        519        582        572  1673
France               455        518        577  1550
…                    …         …         …    …
Ireland-1              0          0          0     0
Israel-1               0          0          0     0
```

```
Israel-2                0          0          0     0
Italy-3                 0          0          0     0
rn-2                    0          0          0     0

[1184 rows x 4 columns]
```

[49]:
```
df=pd.read_csv('D:/DS/resume projects/athlete seaborn/athlete_events.csv')
```

[50]:
```
df1=df.groupby('Team').count().sort_values('Medal',ascending=False).head(20)
```

[51]:
```
df1=df1.sort_values('Medal',ascending=False).reset_index()
```

[52]:
```
df1['Team']
```

[52]:
```
0      United States
1       Soviet Union
2            Germany
3      Great Britain
4             France
5              Italy
6             Sweden
7          Australia
8             Canada
9            Hungary
10            Russia
11       Netherlands
12      East Germany
13             Japan
14            Norway
15             China
16           Finland
17           Romania
18       South Korea
19       Switzerland
Name: Team, dtype: object
```

[53]:
```
dfb=df[(df['Medal']=='Bronze') & (df['Team'].isin(df1['Team']))].
 ↪sort_values('Medal',ascending=False)
dfg=df[(df['Medal']=='Gold') & (df['Team'].isin(df1['Team']))].
 ↪sort_values('Medal',ascending=False)
dfs=df[(df['Medal']=='Silver') & (df['Team'].isin(df1['Team']))].
 ↪sort_values('Medal',ascending=False)
```

[54]:
```
dfg
```

[54]:
```
            ID                        Name Sex   Age  Height  Weight  \
42          17            Paavo Johannes Aaltonen   M  28.0   175.0    64.0
```

```
183584   92274                         Carlo Pavesi   M  33.0    NaN    NaN
183423   92193                          Ilse Paulis   F  23.0  174.0   57.0
183488   92229         Maartje Yvonne Helene Paumen   F  22.0  176.0   66.0
183489   92229         Maartje Yvonne Helene Paumen   F  26.0  176.0   66.0
…        …                                 …   . .   …    …      …
93061    47137                        Jayna Hefford   F  28.0  163.0   63.0
93062    47137                        Jayna Hefford   F  32.0  163.0   63.0
93063    47137                        Jayna Hefford   F  36.0  163.0   63.0
93087    47150                         Csaba Hegeds   M  23.0  187.0   82.0
271076  135553  Galina Ivanovna Zybina (-Fyodorova)   F  21.0  168.0   80.0

                 Team  NOC        Games  Year  Season            City  \
42            Finland  FIN  1948 Summer  1948  Summer          London
183584          Italy  ITA  1956 Summer  1956  Summer       Melbourne
183423    Netherlands  NED  2016 Summer  2016  Summer  Rio de Janeiro
183488    Netherlands  NED  2008 Summer  2008  Summer         Beijing
183489    Netherlands  NED  2012 Summer  2012  Summer          London
…                 …   …           …    …      …               …
93061          Canada  CAN  2006 Winter  2006  Winter          Torino
93062          Canada  CAN  2010 Winter  2010  Winter       Vancouver
93063          Canada  CAN  2014 Winter  2014  Winter           Sochi
93087         Hungary  HUN  1972 Summer  1972  Summer          Munich
271076   Soviet Union  URS  1952 Summer  1952  Summer        Helsinki

              Sport                                    Event Medal
42       Gymnastics           Gymnastics Men's Team All-Around  Gold
183584      Fencing            Fencing Men's epee, Individual  Gold
183423       Rowing   Rowing Women's Lightweight Double Sculls  Gold
183488       Hockey                     Hockey Women's Hockey  Gold
183489       Hockey                     Hockey Women's Hockey  Gold
…                 …                                         …    …
93061    Ice Hockey             Ice Hockey Women's Ice Hockey  Gold
93062    Ice Hockey             Ice Hockey Women's Ice Hockey  Gold
93063    Ice Hockey             Ice Hockey Women's Ice Hockey  Gold
93087     Wrestling  Wrestling Men's Middleweight, Greco-Roman  Gold
271076    Athletics                Athletics Women's Shot Put  Gold

[9947 rows x 15 columns]
```

[55]:

```
dfg=dfg.groupby('Team').count()['Medal'].reset_index()
dfs=dfs.groupby('Team').count()['Medal'].reset_index()
dfb=dfb.groupby('Team').count()['Medal'].reset_index()
```

[56]:

```
dfg
```

[56]:

```
             Team  Medal
0       Australia    342
```

```
1          Canada   422
2           China   308
3    East Germany   369
4         Finland   198
5          France   455
6         Germany   679
7   Great Britain   519
8         Hungary   432
9           Italy   535
10          Japan   247
11    Netherlands   277
12         Norway   299
13        Romania   161
14         Russia   366
15    South Korea   211
16   Soviet Union  1058
17         Sweden   451
18    Switzerland   144
19  United States  2474
```

[57]:
```
dfg=dfg.sort_values('Medal',ascending=False).reset_index()
dfs=dfs.sort_values('Medal',ascending=False).reset_index()
dfb=dfb.sort_values('Medal',ascending=False).reset_index()
```

[58]:
```
dfs
```

[58]:
```
    index           Team  Medal
0      19  United States   1512
1      16   Soviet Union    716
2       6        Germany    627
3       7  Great Britain    582
4       5         France    518
5       9          Italy    508
6      17         Sweden    476
7       0      Australia    453
8       1         Canada    413
9      14         Russia    351
10      8        Hungary    330
11     12         Norway    330
12      2          China    325
13     11    Netherlands    321
14      3   East Germany    309
15     10          Japan    307
16      4        Finland    263
17     15    South Korea    222
18     18    Switzerland    213
19     13        Romania    200
```

[59]:
```
dfb
```

[59]:
```
    index           Team  Medal
0      19  United States   1233
1       6        Germany    678
2      16   Soviet Union    677
3       5         France    577
4       7  Great Britain    572
5       0      Australia    511
6      17         Sweden    507
7       9          Italy    484
8       4        Finland    415
9       1         Canada    408
10     14         Russia    393
11     11    Netherlands    390
12      8        Hungary    365
13     10          Japan    357
14     13        Romania    290
15     12         Norway    281
16      2          China    268
17      3   East Germany    263
18     18    Switzerland    231
19     15    South Korea    159
```

[60]:
```
dfg
```

[60]:
```
    index           Team  Medal
0      19  United States   2474
1      16   Soviet Union   1058
2       6        Germany    679
3       9          Italy    535
4       7  Great Britain    519
5       5         France    455
6      17         Sweden    451
7       8        Hungary    432
8       1         Canada    422
9       3   East Germany    369
10     14         Russia    366
11      0      Australia    342
12      2          China    308
13     12         Norway    299
14     11    Netherlands    277
15     10          Japan    247
16     15    South Korea    211
17      4        Finland    198
18     13        Romania    161
19     18    Switzerland    144
```

```
[61]: dt=pd.DataFrame({'Team':df1['Team'],'Gold':dfg['Medal'],'Silver':
       ↪dfs['Medal'],'Bronze':dfb['Medal']})
```

```
[62]: dt
```

```
[62]:              Team  Gold  Silver  Bronze
      0   United States  2474    1512    1233
      1    Soviet Union  1058     716     678
      2         Germany   679     627     677
      3   Great Britain   535     582     577
      4          France   519     518     572
      5           Italy   455     508     511
      6          Sweden   451     476     507
      7       Australia   432     453     484
      8          Canada   422     413     415
      9         Hungary   369     351     408
      10         Russia   366     330     393
      11    Netherlands   342     330     390
      12   East Germany   308     325     365
      13          Japan   299     321     357
      14         Norway   277     309     290
      15          China   247     307     281
      16        Finland   211     263     268
      17        Romania   198     222     263
      18    South Korea   161     213     231
      19    Switzerland   144     200     159
```

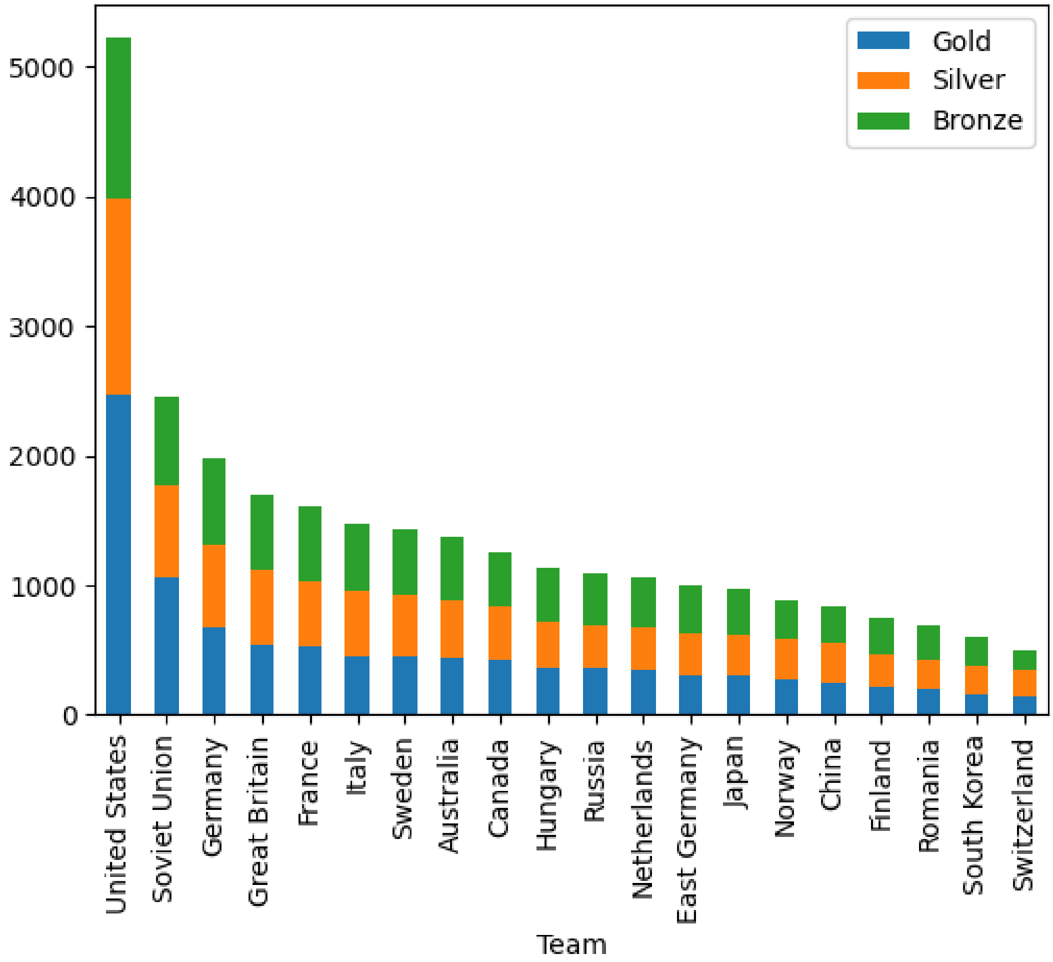
```
[63]: dt.sort_values(by='Gold',ascending=False)
```

```
[63]:              Team  Gold  Silver  Bronze
      0   United States  2474    1512    1233
      1    Soviet Union  1058     716     678
      2         Germany   679     627     677
      3   Great Britain   535     582     577
      4          France   519     518     572
      5           Italy   455     508     511
      6          Sweden   451     476     507
      7       Australia   432     453     484
      8          Canada   422     413     415
      9         Hungary   369     351     408
      10         Russia   366     330     393
      11    Netherlands   342     330     390
      12   East Germany   308     325     365
      13          Japan   299     321     357
      14         Norway   277     309     290
      15          China   247     307     281
      16        Finland   211     263     268
```

```
17       Romania   198     222     263
18   South Korea   161     213     231
19   Switzerland   144     200     159
```

[64]:

```
dt=dt.set_index('Team')
dt.plot(kind='bar',stacked=True)
plt.show()
```



[65]:

```
dt
```

[65]:

```
               Gold  Silver  Bronze
Team
United States  2474    1512    1233
Soviet Union   1058     716     678
Germany         679     627     677
Great Britain   535     582     577
```

```
France          519     518     572
Italy           455     508     511
Sweden          451     476     507
Australia       432     453     484
Canada          422     413     415
Hungary         369     351     408
Russia          366     330     393
Netherlands     342     330     390
East Germany    308     325     365
Japan           299     321     357
Norway          277     309     290
China           247     307     281
Finland         211     263     268
Romania         198     222     263
South Korea     161     213     231
Switzerland     144     200     159
```

[66]:

```
l=df.groupby('Team').count()['Medal'].sort_values(ascending=False).head(5).
 ↪reset_index()
```

[67]:

```
l1=df[(df['Team'].isin(l.Team)) & (df.Season=='Summer')]
```

[68]:

```
l1=l1.groupby(['Team','Year']).count()['Medal'].sort_values(ascending=True).
 ↪reset_index()
```

[69]:

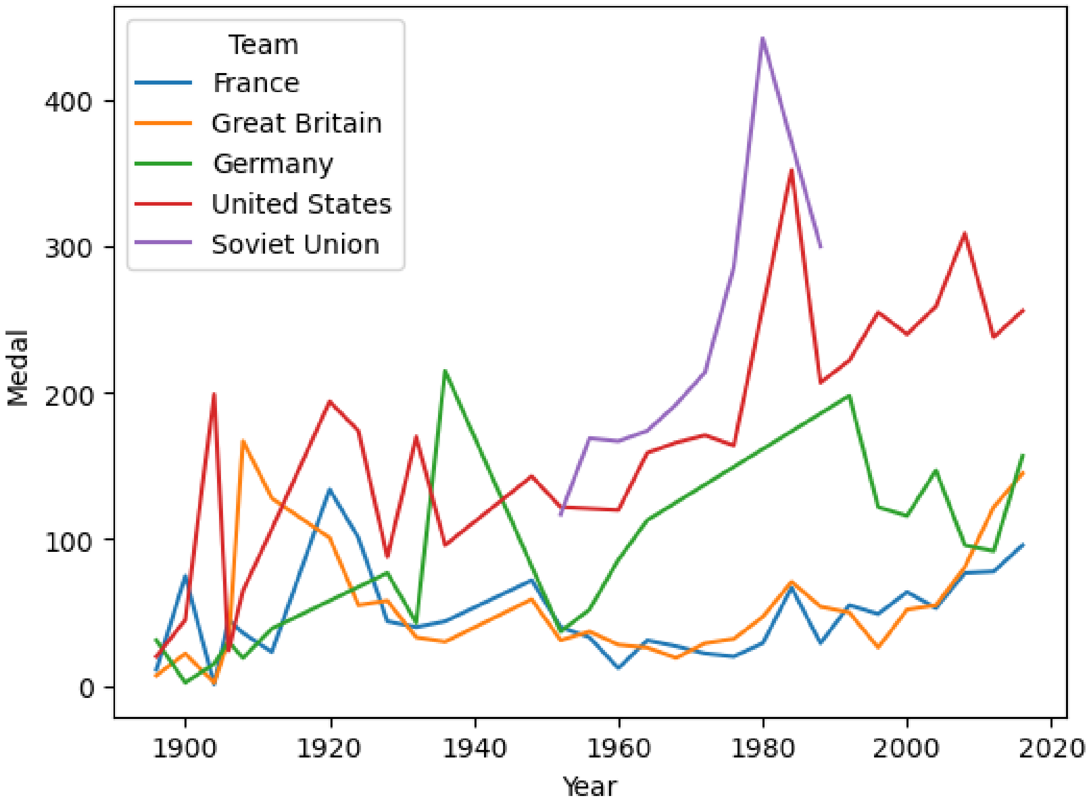
```
l1
```

[69]:

```
              Team  Year  Medal
0           France  1904      1
1    Great Britain  1904      2
2          Germany  1900      2
3    Great Britain  1896      7
4           France  1896     11
..             ...   ...    ...
110   Soviet Union  1976    286
111   Soviet Union  1988    300
112  United States  2008    309
113  United States  1984    352
114   Soviet Union  1980    442

[115 rows x 3 columns]
```

[70]:

```
sns.lineplot(x=l1.Year,y=l1.Medal,hue=l1.Team)
plt.show()
```



```
[107]: df
```

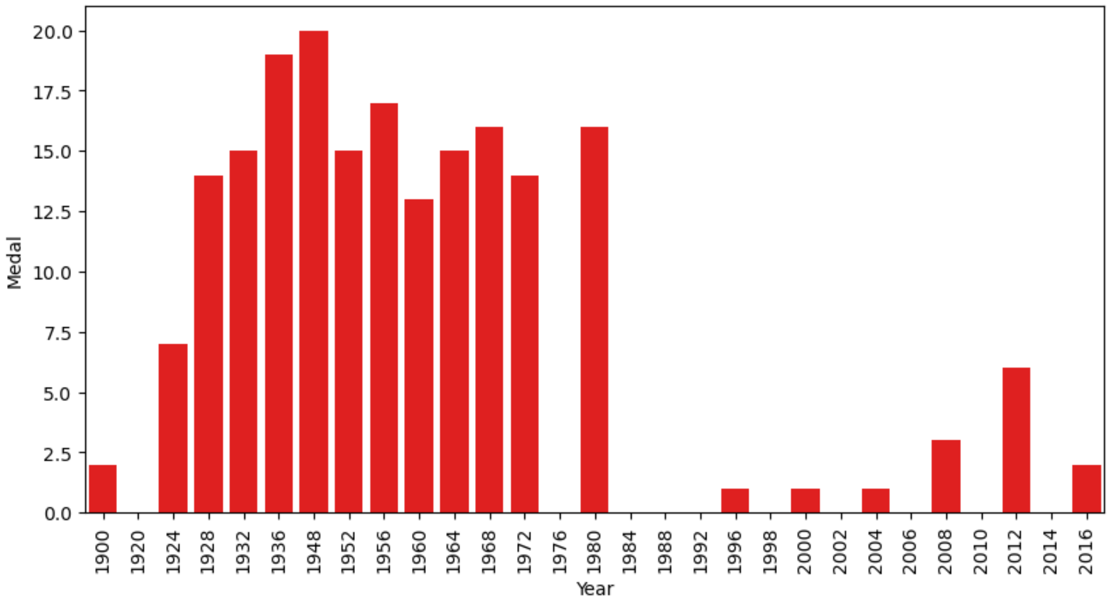
```
[107]:             ID                      Name Sex   Age  Height  Weight  \
       0            1                 A Dijiang   M  24.0   180.0    80.0
       1            2                  A Lamusi   M  23.0   170.0    60.0
       2            3       Gunnar Nielsen Aaby   M  24.0   180.0    70.0
       3            4      Edgar Lindenau Aabye   M  34.0   180.0    70.0
       4            5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
       …        …                       …   ..   …     …     …
       271111  135569                Andrzej ya   M  29.0   179.0    89.0
       271112  135570                  Piotr ya   M  27.0   176.0    59.0
       271113  135570                  Piotr ya   M  27.0   176.0    59.0
       271114  135571        Tomasz Ireneusz ya   M  30.0   185.0    96.0
       271115  135571        Tomasz Ireneusz ya   M  34.0   185.0    96.0

                         Team  NOC       Games  Year  Season            City  \
       0                China  CHN  1992 Summer  1992  Summer       Barcelona
       1                China  CHN  2012 Summer  2012  Summer          London
       2              Denmark  DEN  1920 Summer  1920  Summer       Antwerpen
       3       Denmark/Sweden  DEN  1900 Summer  1900  Summer           Paris
       4          Netherlands  NED  1988 Winter  1988  Winter         Calgary
```

```
…                …  …          …   …     …             …
271111        Poland-1  POL  1976 Winter  1976  Winter       Innsbruck
271112          Poland  POL  2014 Winter  2014  Winter           Sochi
271113          Poland  POL  2014 Winter  2014  Winter           Sochi
271114          Poland  POL  1998 Winter  1998  Winter          Nagano
271115          Poland  POL  2002 Winter  2002  Winter  Salt Lake City

                Sport                                     Event Medal
0          Basketball               Basketball Men's Basketball     0
1                Judo              Judo Men's Extra-Lightweight     0
2            Football                   Football Men's Football     0
3          Tug-Of-War               Tug-Of-War Men's Tug-Of-War  Gold
4       Speed Skating          Speed Skating Women's 500 metres     0
…               …                                       …   …
271111           Luge                Luge Mixed (Men)'s Doubles     0
271112    Ski Jumping  Ski Jumping Men's Large Hill, Individual     0
271113    Ski Jumping        Ski Jumping Men's Large Hill, Team     0
271114      Bobsleigh                      Bobsleigh Men's Four     0
271115      Bobsleigh                      Bobsleigh Men's Four     0

[271116 rows x 15 columns]
```

```
[72]: dfi=df[df['Team']=='India']
```

```
[73]: dfi=dfi.groupby(['Year']).count()['Medal'].sort_values(ascending=True).
        ↪reset_index()
```

```
[74]: plt.figure(figsize=(10,5))
      sns.barplot(x=dfi.Year,y=dfi.Medal,color='red')
      plt.xticks(rotation=90)
      plt.show()
```



[4]:
```
df=pd.read_csv('D:/DS/resume projects/athlete seaborn/athlete_events.csv')
```

[5]:
```
df2=df
```

[6]:
```
df2.isna().sum().sum()
```

[6]:
```
363853
```

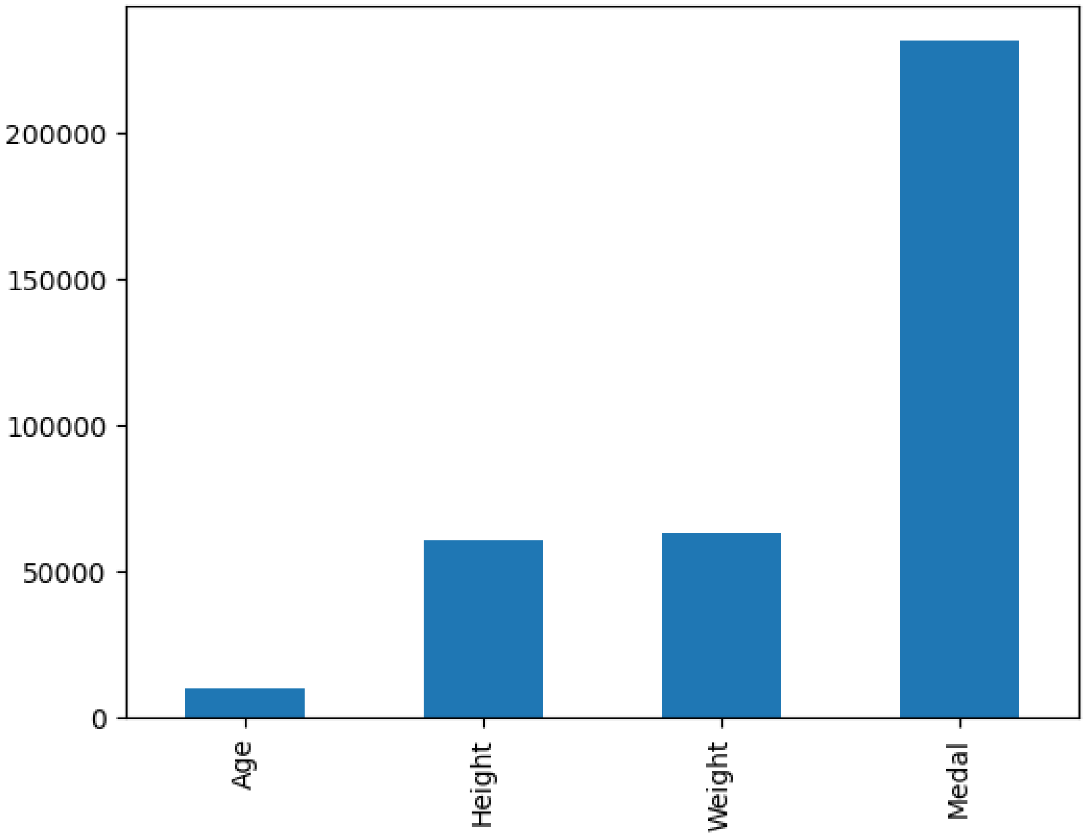
[7]:
```
l=[]
for i in df2.columns:
    if df2[i].isna().sum()>0:
        l.append(i)
```

[8]:
```
l
```

[8]:
```
['Age', 'Height', 'Weight', 'Medal']
```

[9]:
```
df2[l].isna().sum().plot(kind='bar')
```

[9]:
```
<Axes: >
```



```
[10]: df['Medal'] = df['Medal'].replace({'Gold': 1, 'Silver': 1, 'Bronze': 1})
      df['Medal'].fillna(0, inplace=True)
```

```
[11]: df['Medal'].value_counts()
```

```
[11]: 0.0    231333
      1.0     39783
      Name: Medal, dtype: int64
```

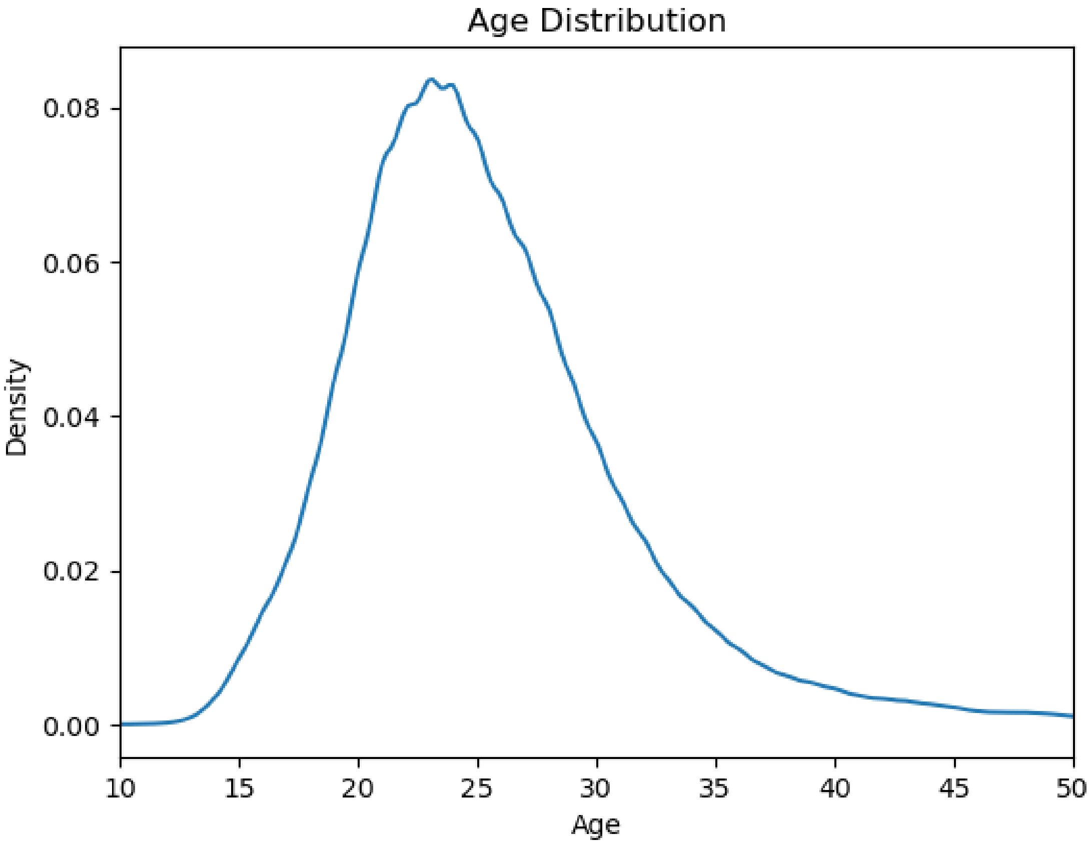
```
[55]: df
```

```
[55]:             ID                      Name Sex   Age  Height  Weight  \
      0            1                 A Dijiang   M  24.0   180.0    80.0
      1            2                  A Lamusi   M  23.0   170.0    60.0
      2            3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN
      3            4      Edgar Lindenau Aabye   M  34.0     NaN     NaN
      4            5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
      …         …                       …   ..  …     …     …
      271111  135569                Andrzej ya   M  29.0   179.0    89.0
      271112  135570                  Piotr ya   M  27.0   176.0    59.0
```

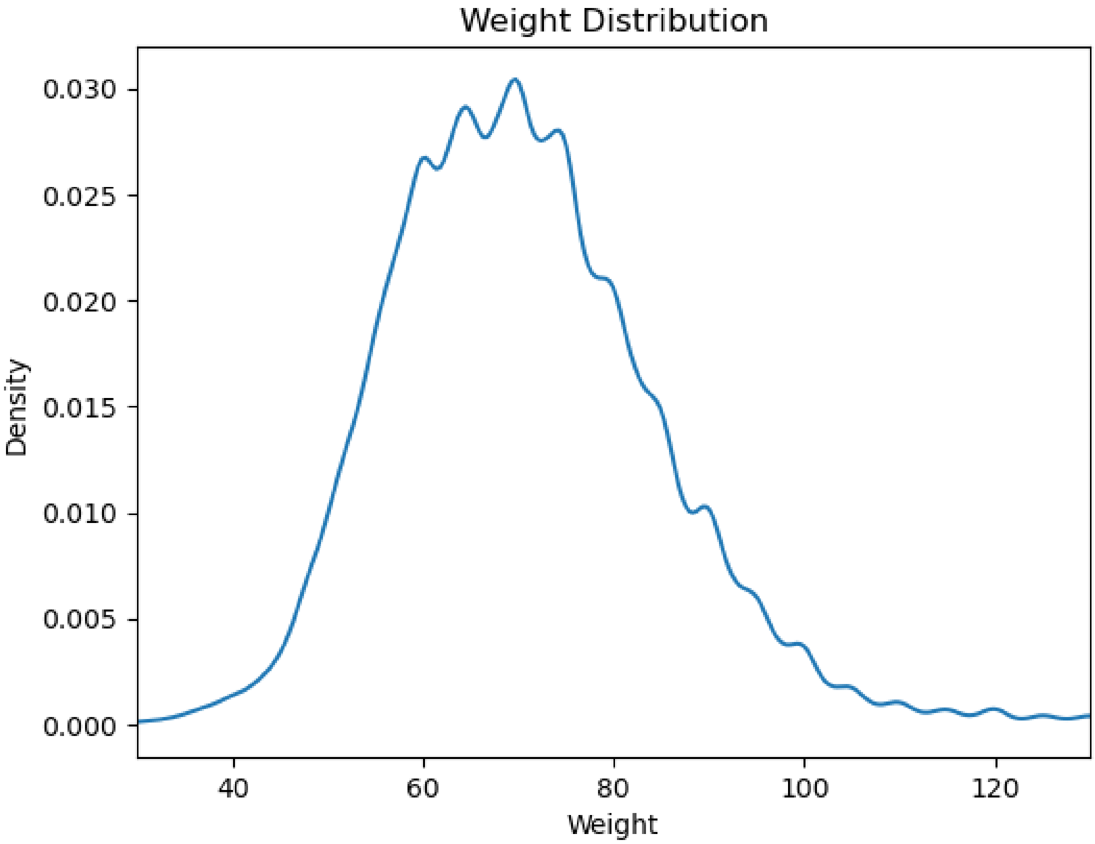
```
271113  135570                  Piotr ya   M  27.0   176.0    59.0
271114  135571        Tomasz Ireneusz ya   M  30.0   185.0    96.0
271115  135571        Tomasz Ireneusz ya   M  34.0   185.0    96.0

                  Team  NOC        Games  Year  Season            City  \
0                China  CHN  1992 Summer  1992  Summer       Barcelona
1                China  CHN  2012 Summer  2012  Summer          London
2              Denmark  DEN  1920 Summer  1920  Summer       Antwerpen
3       Denmark/Sweden  DEN  1900 Summer  1900  Summer           Paris
4          Netherlands  NED  1988 Winter  1988  Winter         Calgary
…                …  …          …   …     …             …
271111        Poland-1  POL  1976 Winter  1976  Winter       Innsbruck
271112          Poland  POL  2014 Winter  2014  Winter           Sochi
271113          Poland  POL  2014 Winter  2014  Winter           Sochi
271114          Poland  POL  1998 Winter  1998  Winter          Nagano
271115          Poland  POL  2002 Winter  2002  Winter  Salt Lake City

                Sport                                     Event  Medal
0          Basketball               Basketball Men's Basketball    0.0
1                Judo              Judo Men's Extra-Lightweight    0.0
2            Football                   Football Men's Football    0.0
3          Tug-Of-War               Tug-Of-War Men's Tug-Of-War    1.0
4       Speed Skating          Speed Skating Women's 500 metres    0.0
…               …                                       …    …
271111           Luge                Luge Mixed (Men)'s Doubles    0.0
271112    Ski Jumping  Ski Jumping Men's Large Hill, Individual    0.0
271113    Ski Jumping        Ski Jumping Men's Large Hill, Team    0.0
271114      Bobsleigh                      Bobsleigh Men's Four    0.0
271115      Bobsleigh                      Bobsleigh Men's Four    0.0

[271116 rows x 15 columns]
```

[12]:

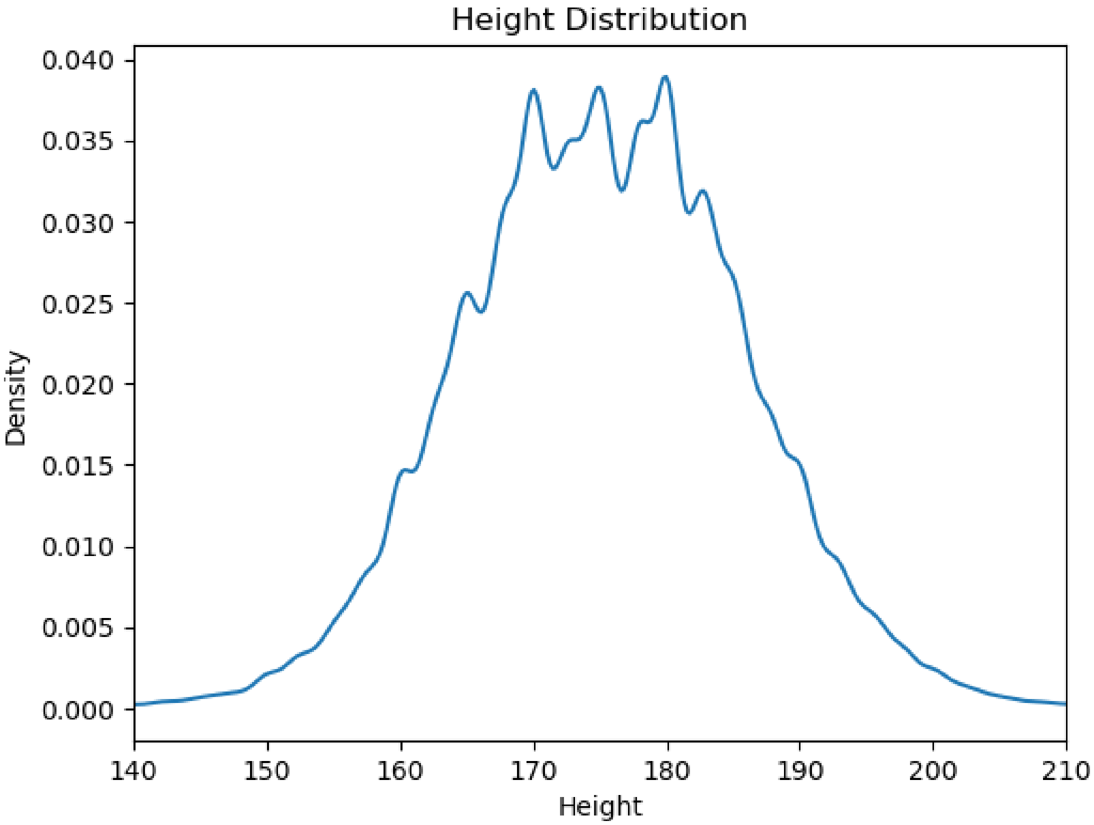
```
df2['Age'].plot(kind='kde')
plt.xlim(10, 50)
plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Age Distribution')
plt.show()
```



```
[139]: df2['Weight'].plot(kind='kde')
       plt.xlim(30, 130)
       plt.xlabel('Weight')
       plt.ylabel('Density')
       plt.title('Weight Distribution')
       plt.show()
```



[140]:
```
df2['Height'].plot(kind='kde')
plt.xlim(140, 210)
plt.xlabel('Height')
plt.ylabel('Density')
plt.title('Height Distribution')
plt.show()
```



[13]:
```
for i in l:
    df2[i] = df2[i].fillna(df[i].mode().iloc[0])
```

[14]:
```
df2.isna().sum().sum()
```

[14]: 0

[15]:
```
df2
```

[15]:
```
              ID                    Name Sex   Age  Height  Weight  \
0              1               A Dijiang   M  24.0   180.0    80.0
1              2                A Lamusi   M  23.0   170.0    60.0
2              3     Gunnar Nielsen Aaby   M  24.0   180.0    70.0
3              4    Edgar Lindenau Aabye   M  34.0   180.0    70.0
4              5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
…        …                         …   ..   …     …     …
271111  135569              Andrzej ya   M  29.0   179.0    89.0
271112  135570                Piotr ya   M  27.0   176.0    59.0
271113  135570                Piotr ya   M  27.0   176.0    59.0
271114  135571      Tomasz Ireneusz ya   M  30.0   185.0    96.0
```

```
271115  135571        Tomasz Ireneusz ya   M  34.0   185.0    96.0

                 Team  NOC        Games  Year  Season           City  \
0               China  CHN  1992 Summer  1992  Summer      Barcelona
1               China  CHN  2012 Summer  2012  Summer         London
2             Denmark  DEN  1920 Summer  1920  Summer      Antwerpen
3      Denmark/Sweden  DEN  1900 Summer  1900  Summer          Paris
4         Netherlands  NED  1988 Winter  1988  Winter        Calgary
…              …   …          …    …      …            …
271111       Poland-1  POL  1976 Winter  1976  Winter      Innsbruck
271112         Poland  POL  2014 Winter  2014  Winter          Sochi
271113         Poland  POL  2014 Winter  2014  Winter          Sochi
271114         Poland  POL  1998 Winter  1998  Winter         Nagano
271115         Poland  POL  2002 Winter  2002  Winter  Salt Lake City

                Sport                                    Event  Medal
0          Basketball              Basketball Men's Basketball    0.0
1                Judo             Judo Men's Extra-Lightweight    0.0
2            Football                  Football Men's Football    0.0
3          Tug-Of-War              Tug-Of-War Men's Tug-Of-War    1.0
4       Speed Skating         Speed Skating Women's 500 metres    0.0
…               …                                    …    …
271111           Luge               Luge Mixed (Men)'s Doubles    0.0
271112    Ski Jumping  Ski Jumping Men's Large Hill, Individual    0.0
271113    Ski Jumping        Ski Jumping Men's Large Hill, Team    0.0
271114      Bobsleigh                      Bobsleigh Men's Four    0.0
271115      Bobsleigh                      Bobsleigh Men's Four    0.0

[271116 rows x 15 columns]
```

[16]:
```
df=df2
```

[17]:
```
X = df.drop(['Medal','ID','Name','City','Games','Year'], axis=1)
y = df['Medal']
```

1 Label Encoding

[18]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X['Team'] = le.fit_transform(X['Team'])
X['Sport'] = le.fit_transform(X['Sport'])
X['Event'] = le.fit_transform(X['Event'])
X['NOC'] = le.fit_transform(X['NOC'])
```

2 One Hot Encoding

[19]:
```
X = pd.get_dummies(X, drop_first=True, sparse=True)
```

3 Train Test Split

[20]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[21]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
import numpy as np

# Create the DecisionTreeClassifier instance with balanced class weights
dt = DecisionTreeClassifier(class_weight='balanced')

# Define the parameter grid with reduced ranges
param_dist = {
    'max_depth': [5, 10, 15, None],  # A smaller range for max_depth
    'min_samples_split': np.arange(2, 8, 2),  # Fewer values for␣
 ↪min_samples_split
    'min_samples_leaf': np.arange(2, 5),  # Fewer values for min_samples_leaf
    'max_features': [0.2, 0.3, 0.5, 0.7],
}

# Create the RandomizedSearchCV instance
random_search = RandomizedSearchCV(dt, param_distributions=param_dist, cv=5,␣
 ↪n_iter=20, n_jobs=-1)

# Perform the random search on your dataset
random_search.fit(X_train, y_train)

# Print the best hyperparameters and corresponding score
print("Best Hyperparameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

```
Best Hyperparameters: {'min_samples_split': 2, 'min_samples_leaf': 2,
'max_features': 0.7, 'max_depth': None}
Best Score: 0.8155303097799423
```

4 1.Hyper Parameter Tunning

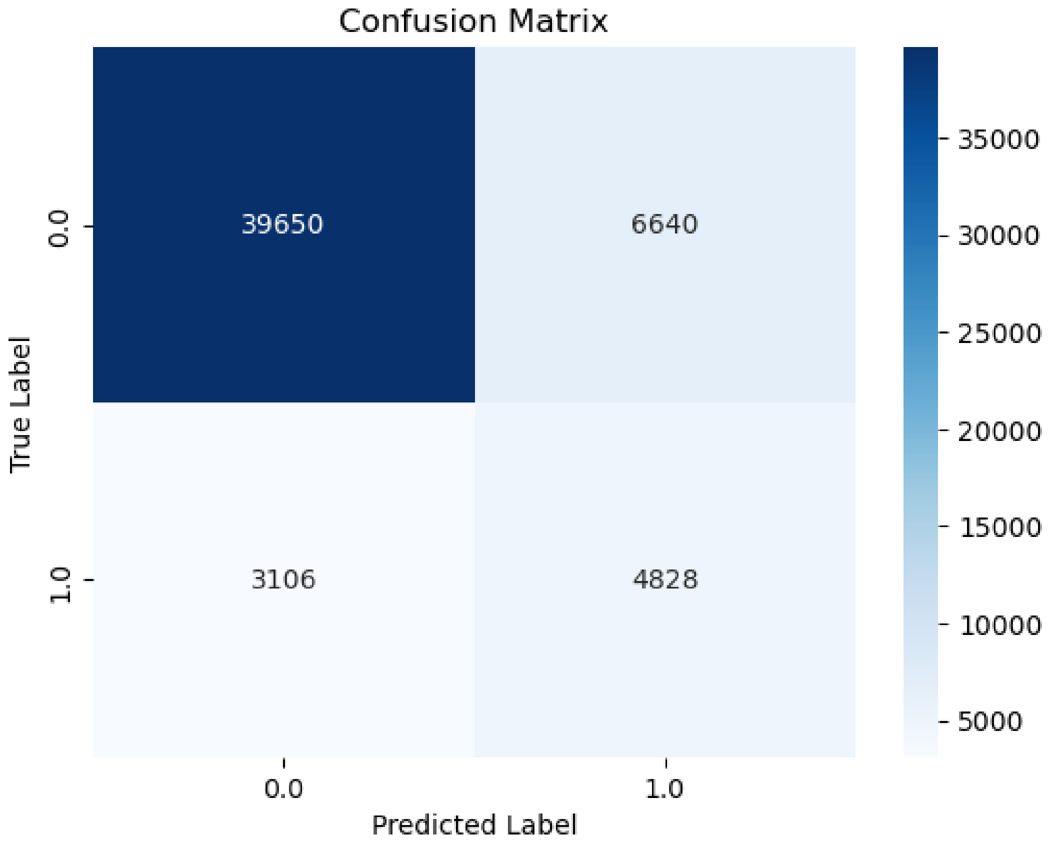
[23]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Define the updated parameter grid with more options
param_dist = {
    'max_depth': [None, 5, 10, 15, 20],  # More options for max_depth
    'min_samples_split': np.arange(2, 10, 2),  # Expanded range for␣
 ↪min_samples_split
    'min_samples_leaf': np.arange(2, 6),  # Expanded range for min_samples_leaf
    'max_features': [0.2, 0.3, 0.5, 0.7, 0.9],  # More options for max_features
}

# Create the DecisionTreeClassifier instance with balanced class weights
dt = DecisionTreeClassifier(class_weight='balanced')

# Create the RandomizedSearchCV instance with more iterations
random_search = RandomizedSearchCV(dt, param_distributions=param_dist, cv=5,␣
 ↪n_iter=50, n_jobs=-1)

# Perform the random search on your dataset
random_search.fit(X_train, y_train)

# Print the best hyperparameters and corresponding score
print("Best Hyperparameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)

# Get the best model from the random search
best_model = random_search.best_estimator_

# Predict on the test set (assuming you have a separate test set)
y_pred = best_model.predict(X_test)

# Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix heatmap
labels = np.unique(y_test)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

```
Best Hyperparameters: {'min_samples_split': 4, 'min_samples_leaf': 2,
'max_features': 0.9, 'max_depth': None}
Best Score: 0.8191588506847454
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```



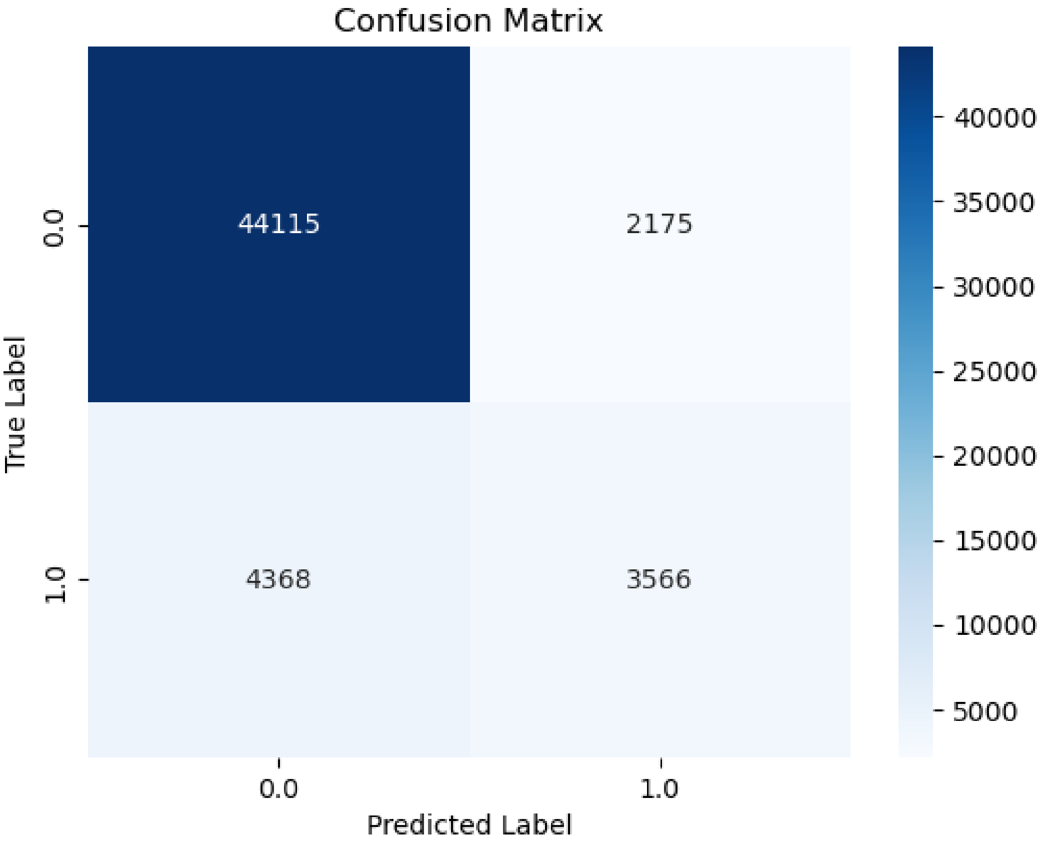
5 2.Hyperparameter tuning(Computationally expensive)

[24]:
```
from sklearn.ensemble import RandomForestClassifier

# Create the RandomForestClassifier instance with balanced class weights
rf = RandomForestClassifier(class_weight='balanced', n_jobs=-1)
```

```python
# Define a smaller parameter grid with reduced options
param_dist = {
    #'n_estimators': [20, 25],
    'max_depth': [20, 50, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 3],
    'max_features': [0.5, 0.7],
}

# Create the RandomizedSearchCV instance with fewer iterations
random_search = RandomizedSearchCV(rf, param_distributions=param_dist, cv=5,␣
 ↪n_iter=10, n_jobs=-1)

# Perform the random search on a subset of your dataset
subset_size = 5000  # Adjust this size as needed
random_search.fit(X_train[:subset_size], y_train[:subset_size])

# Print the best hyperparameters and corresponding score
print("Best Hyperparameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)

# Get the best model from the random search
best_model = random_search.best_estimator_

# Now fit the best model on the entire training set (optional)
best_model.fit(X_train, y_train)

# Predict on the test set (assuming you have a separate test set)
y_pred = best_model.predict(X_test)

# Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix heatmap
labels = np.unique(y_test)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
Best Hyperparameters: {'min_samples_split': 2, 'min_samples_leaf': 1,
'max_features': 0.5, 'max_depth': None}
Best Score: 0.8549999999999999
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```



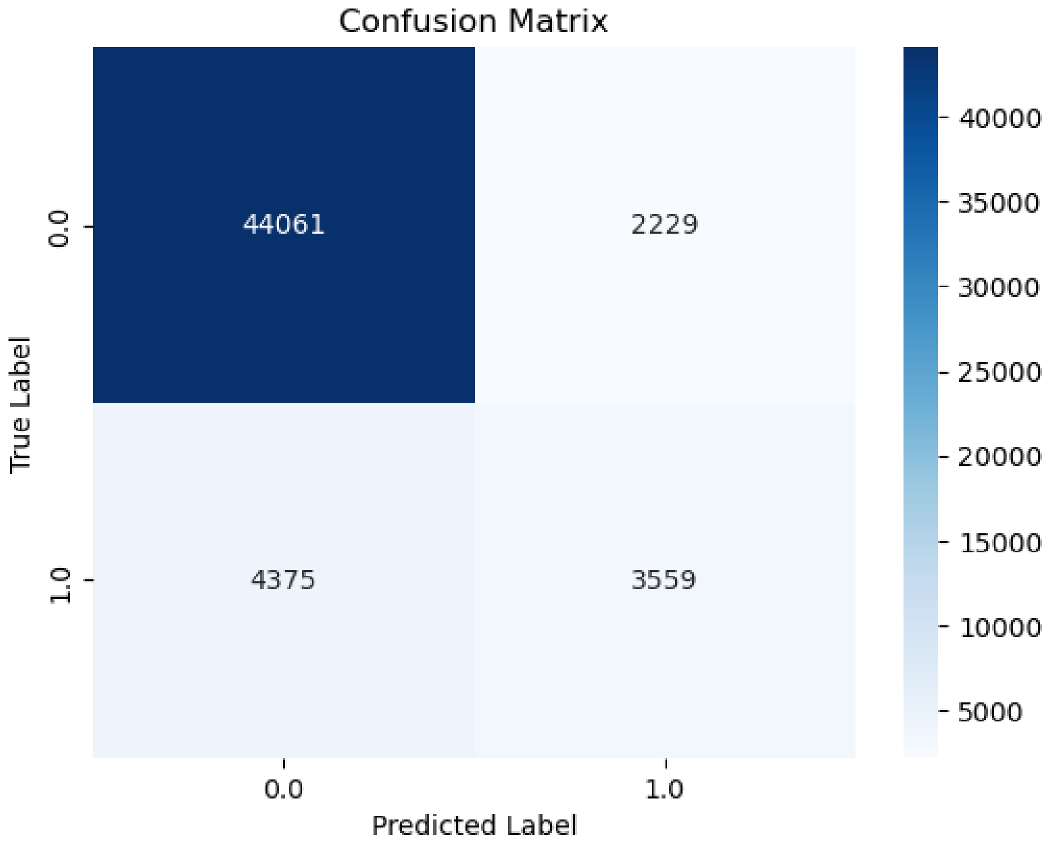
6 3. Hyperparameter tuning(Computationally expensive code)

[25]:
```
# from sklearn.ensemble import RandomForestClassifier

# # Create the RandomForestClassifier instance with balanced class weights
# rf = RandomForestClassifier(class_weight='balanced', n_jobs=-1)
# # Define the parameter grid for RandomizedSearchCV
# param_dist = {
#     'n_estimators': [10, 20],  # Number of trees in the forest
#     'max_depth': [10, 20, 30, None],  # Regularization: More options for max_depth
#     'min_samples_split': np.arange(2, 10, 2),  # Regularization: Expanded range for min_samples_split
#     'min_samples_leaf': np.arange(1, 6),  # Regularization: Expanded range for min_samples_leaf
#     'max_features': ['auto', 'sqrt', 0.2, 0.5, 0.7],  # Different options for max_features
# }

# # Create the RandomizedSearchCV instance with more iterations and folds
# random_search = RandomizedSearchCV(rf, param_distributions=param_dist, cv=10, n_iter=50, n_jobs=-1)

# # Perform the random search on your dataset
# random_search.fit(X_train, y_train)

# # Print the best hyperparameters and corresponding score
# print("Best Hyperparameters:", random_search.best_params_)
# print("Best Score:", random_search.best_score_)

# # Get the best model from the random search
# best_model = random_search.best_estimator_

# # Predict on the test set (assuming you have a separate test set)
# y_pred = best_model.predict(X_test)

# # Create the confusion matrix
# cm = confusion_matrix(y_test, y_pred)

# # Plot the confusion matrix heatmap
# labels = np.unique(y_test)
# sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
# plt.xlabel("Predicted Label")
# plt.ylabel("True Label")
# plt.title("Confusion Matrix")
# plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

```
Best Hyperparameters: {'n_estimators': 20, 'min_samples_split': 2,
'min_samples_leaf': 1, 'max_features': 0.5, 'max_depth': 30}
Best Score: 0.8765284301800967
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```



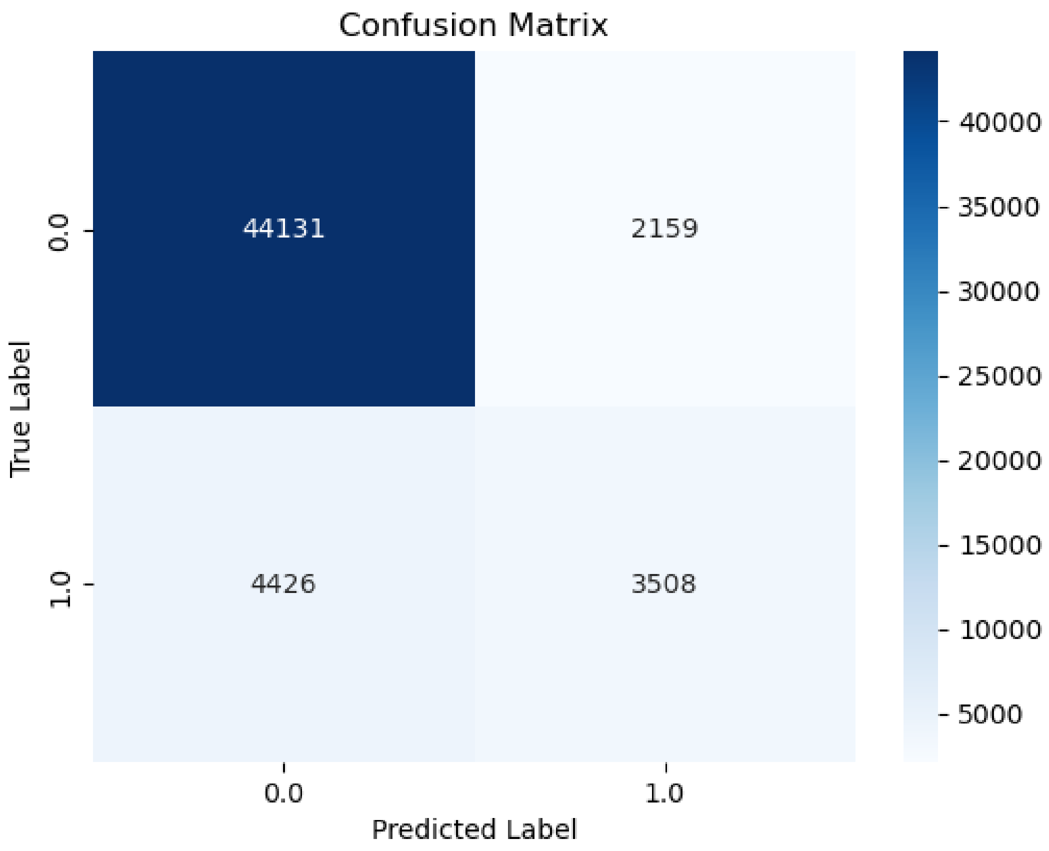
7 Random Forest

```
[26]: import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score

# Create the RandomForestClassifier instance with balanced class weights
rf = RandomForestClassifier(class_weight='balanced', n_jobs=-1)

# Fit the model on the training data
rf.fit(X_train, y_train)
```

```python
# Make predictions on the test set
y_pred = rf.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Initial Accuracy:", accuracy)

# Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix heatmap
labels = np.unique(y_test)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

```
Initial Accuracy: 0.878559309530835
```



8 Hyperparameter In RF

[27]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score

# Create the RandomForestClassifier instance with balanced class weights
rf = RandomForestClassifier(class_weight='balanced', n_jobs=-1)

# Define the parameter grid for RandomizedSearchCV
param_dist = {
    'n_estimators': [100, 150, 200, 250],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': np.arange(2, 10, 2),
    'min_samples_leaf': np.arange(1, 6),
    'max_features': [0.5, 0.6, 0.7, 0.8],
}

# Create the RandomizedSearchCV instance with a reasonable number of iterations
```

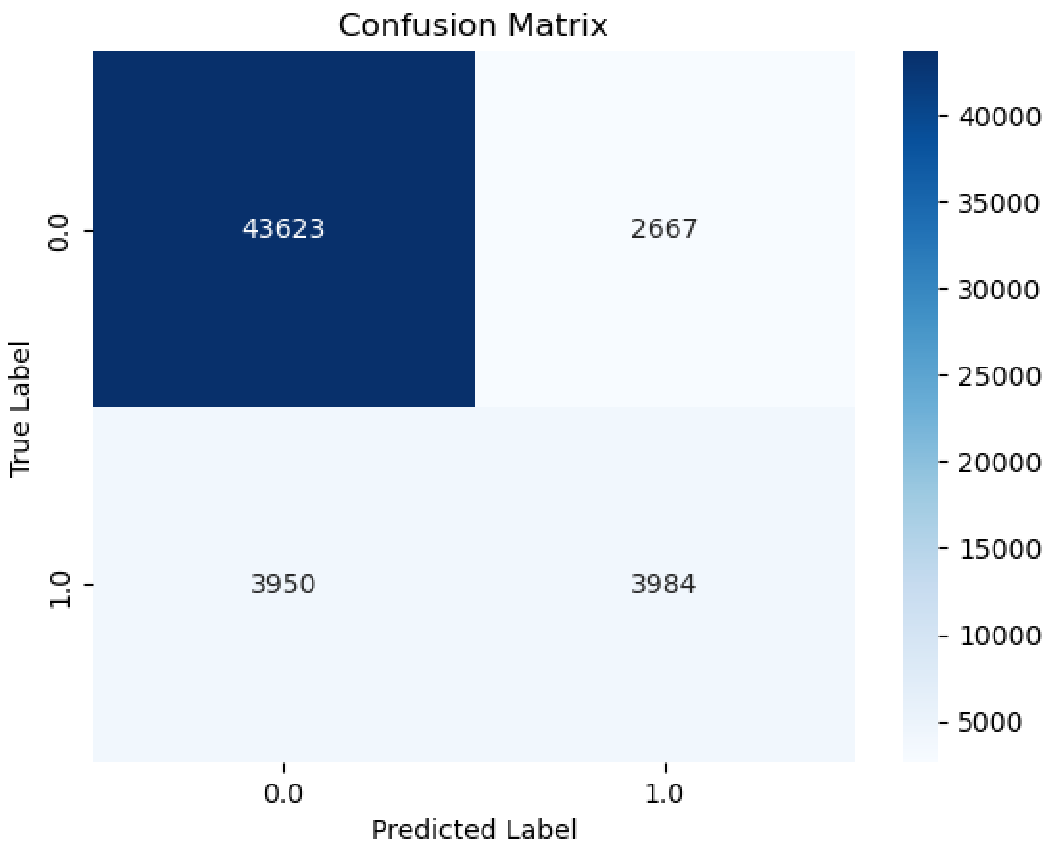
```
random_search = RandomizedSearchCV(rf, param_distributions=param_dist, cv=5,␣
  ↪n_iter=20, n_jobs=-1)

# Perform the random search on the full dataset
random_search.fit(X_train, y_train)

# Print the best hyperparameters and corresponding score
print("Best Hyperparameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)

# Get the best model from the random search
best_model = random_search.best_estimator_

# Predict on the test set (assuming you have a separate test set)
y_pred = best_model.predict(X_test)

# Calculate the accuracy score for the best model
accuracy = accuracy_score(y_test, y_pred)
print("Best Model Accuracy:", accuracy)

# Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix heatmap
labels = np.unique(y_test)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
  ↪yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

```
Best Hyperparameters: {'n_estimators': 200, 'min_samples_split': 4,
'min_samples_leaf': 1, 'max_features': 0.8, 'max_depth': None}
Best Score: 0.8772891518760447
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

```
Best Model Accuracy: 0.8779691649454117
```



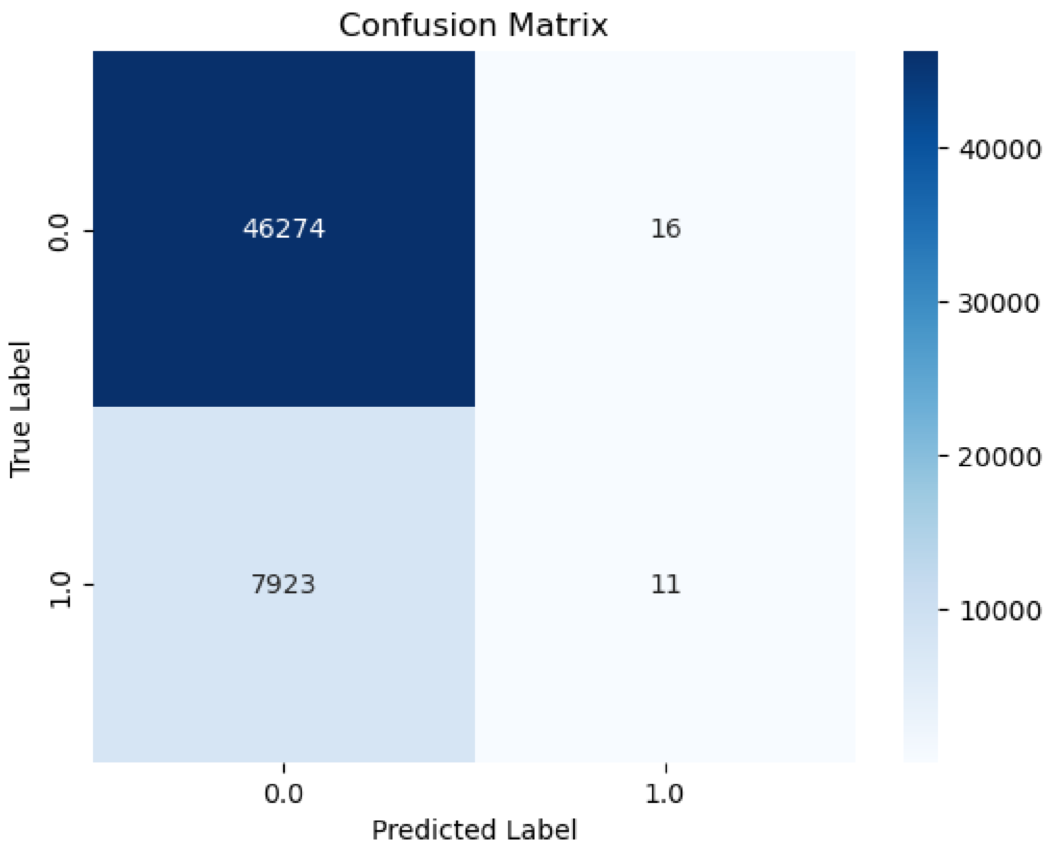
9 Logistic regression

[28]:
```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score

# Create the LogisticRegression instance
logreg = LogisticRegression()

# Fit the model on the training data
logreg.fit(X_train, y_train)

# Make predictions on the test set
y_pred = logreg.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Initial Accuracy:", accuracy)
```

```
# Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix heatmap
labels = np.unique(y_test)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
```

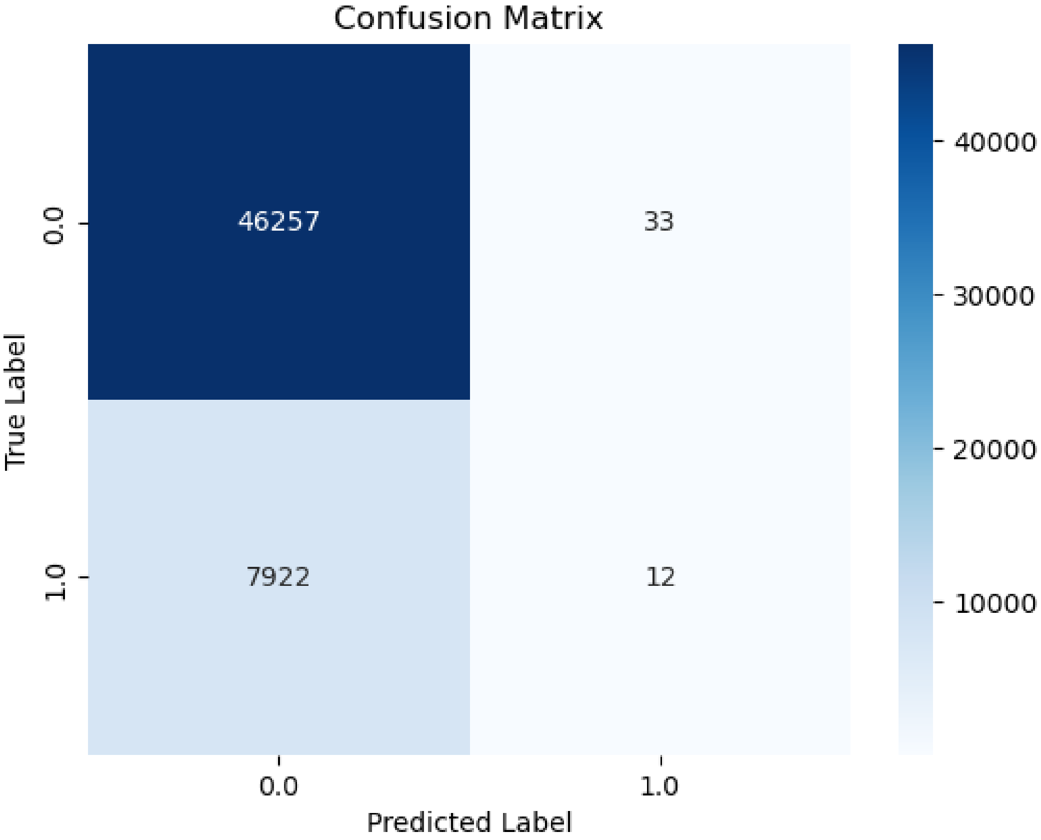
```
Initial Accuracy: 0.8535888167601062
```



[29]:
```
from sklearn.model_selection import RandomizedSearchCV
import numpy as np

# Create the LogisticRegression instance
logreg = LogisticRegression()

# Define the parameter grid for RandomizedSearchCV
param_dist = {
    'C': np.logspace(-3, 3, 7),  # Vary C from 0.001 to 1000
}

# Create the RandomizedSearchCV instance with a reasonable number of iterations
random_search = RandomizedSearchCV(logreg, param_distributions=param_dist,␣
 ↪cv=5, n_iter=20, n_jobs=-1)

# Perform the random search on the full dataset
random_search.fit(X_train, y_train)

# Print the best hyperparameters and corresponding score
print("Best Hyperparameters:", random_search.best_params_)
```

```python
print("Best Score:", random_search.best_score_)

# Get the best model from the random search
best_model = random_search.best_estimator_

# Predict on the test set (assuming you have a separate test set)
y_pred = best_model.predict(X_test)

# Calculate the accuracy score for the best model
accuracy = accuracy_score(y_test, y_pred)
print("Best Model Accuracy:", accuracy)

# Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix heatmap
labels = np.unique(y_test)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\model_selection\_search.py:305: UserWarning: The total space of
parameters 7 is smaller than n_iter=20. Running 7 iterations. For exhaustive
searches, use GridSearchCV.
  warnings.warn(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\Kundan Mourya\anaconda3\lib\site-
packages\sklearn\utils\validation.py:768: UserWarning: pandas.DataFrame with
sparse columns found.It will be converted to a dense numpy array.
  warnings.warn(
Best Hyperparameters: {'C': 0.001}
Best Score: 0.8529268009214984
Best Model Accuracy: 0.8532937444673945
```



```
[ ]:
```