# Milestone II: Descriptive Stats

In [1]:

```
#Classic Libraries
import pandas as pd
import numpy as np
import random
import math
import time
import datetime
import operator
import sys
import statsmodels.api as sm
import re

#Data Viz
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
pd.options.plotting.backend = "plotly"
#SQL Libraries
import sqlalchemy
import psycopg2
from sqlalchemy import create_engine
from sqlalchemy_utils import database_exists, create_database
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
import warnings
warnings.filterwarnings('ignore')
%load_ext sql
```

## First I am going to load the clean and joined data I went through in Milestone 1 and exported from Databricks

In [2]:

```
%sql postgresql+psycopg2://postgres:pass@localhost/postgres
```

Out[2]:

```
'Connected: postgres@postgres'
```

In [3]:

```
conn = psycopg2.connect("postgresql://postgres:pass@localhost/postgres")
```

# My Hypothesises

## 1. On a per capita basis smaller and pooere countires will probably have a higher medal count and some of them will be outliers.

## 2. Countries farther North will excel in the winter and vice versa for the South in the summer.

## 3. Competitors that are taller will have an advantage.

## 4. Competitors that weigh more will have an advantage.

# Hypothesis I and !!

## On a per capita basis smaller and pooere countires will probably have a higher medal count and some of them will be outliers.

In order to see the accuracy of this hypothesis about both smaller population and less wealthy countries and their relation to their medal counts I am going to need a little more data. I downloaded two data sets from the data.WorldBank to get the data i need in relation to GDP and the other in relation to population. First I'm going to have to load them data frame behind those dataframes with the medal count by countries to start.

As for the second hypothesis I am planning on making an animated graphic with this data that will be rotating between Summer and Winter Olympics so we should be able to look that way.

## GDP

In [4]:

```
gdp = pd.read_excel('worldgdp1960.xlsx')
gdp.columns = gdp.columns.str.replace(r"\(.*\)","")
gdp = gdp.rename(columns={'United States':'USA', 'United Kingdom':'UK'})
gdp = gdp.rename(columns ={'Country Name':'year'}).set_index('year')
stack = gdp.stack()
stack  =pd.DataFrame(stack) #
stack.index.names=['year', 'country']
stack.columns = ['gdp_per_capita']
stack
```

Out[4]:

| | | gdp_per_capita |
|---|---|---|
| **year** | **country** | |
| **1960** | **Africa Eastern and Southern** | 147.507808 |
| | **Afghanistan** | 59.773234 |
| | **Africa Western and Central** | 107.932233 |
| | **Australia** | 1810.619230 |
| | **Austria** | 935.460427 |
| **...** | **...** | ... |
| **2020** | **Samoa** | 4067.843459 |
| | **Kosovo** | 4346.637931 |
| | **South Africa** | 5655.867654 |
| | **Zambia** | 985.132436 |
| | **Zimbabwe** | 1214.509820 |

12836 rows × 1 columns

## Population

In [5]:

```
pop = pd.read_csv('country_populations.csv')
pop.columns = pop.columns.str.replace(r"\(.*\)","")
pop = pop.rename(columns={'United States':'USA', 'United Kingdom':'UK'})
pop = pop.rename(columns ={'Country Name':'year'}).set_index('year')
pop= pop.stack()
pop =pd.DataFrame(pop)
pop.index.names = ['year', 'country']
pop.columns = ['population']
pop
```

Out[5]:

| | | population |
|---|---|---|
| **year** | **country** | |
| **1960** | **Aruba** | 54208.0 |
| | **Africa Eastern and Southern** | 130836765.0 |
| | **Afghanistan** | 8996967.0 |
| | **Africa Western and Central** | 96396419.0 |
| | **Angola** | 5454938.0 |
| **...** | **...** | ... |
| **2020** | **Kosovo** | 1775378.0 |
| | **Yemen, Rep.** | 29825968.0 |
| | **South Africa** | 59308690.0 |
| | **Zambia** | 18383956.0 |
| | **Zimbabwe** | 14862927.0 |

16123 rows × 1 columns

## Medal Count

In [6]:

```
%%sql countrymedals <<
SELECT region, count(medal), year, "NOC"
FROM sportstats
Group by region, "NOC", year
order by year
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
3290 rows affected.
Returning data to local variable countrymedals
```

In [7]:

```
countrymedals = pd.DataFrame(countrymedals)
countrymedals.columns = ['country', 'medals', 'year', 'code']
countrymedals.at[3289,'year']=2016
countrymedals.year = countrymedals.year.astype(int)
medaldf = countrymedals
mdf = medaldf[medaldf['year']>1959]
s= mdf.groupby(['year', 'country', 'code'], as_index=True, sort=False)
medf = s.sum()
medf
```

Out[7]:

| | | | medals |
|---|---|---|---|
| **year** | **country** | **code** | |
| **1960** | **Bulgaria** | **BUL** | 7 |
| | **Fiji** | **FIJ** | 0 |
| | **Vietnam** | **VNM** | 0 |
| | **Puerto Rico** | **PUR** | 0 |
| | **Japan** | **JPN** | 31 |
| **...** | **...** | **...** | ... |
| **2016** | **Individual Olympic Athletes** | **IOA** | 2 |
| | **Madagascar** | **MAD** | 0 |

| | | |
|---|---|---|
| **Sierra Leone** | **SLE** | 0 |
| **Iran** | **IRI** | 8 |
| **Singapore** | **SIN** | 0 |

2736 rows × 1 columns

In [8]:
```
bdf = medf.merge(pop, how='inner', left_index=True, right_index=True)
df = bdf.merge(stack, how='inner', left_index=True, right_index=True)
df.gdp_per_capita=df.gdp_per_capita.round(2)
df = df.reset_index(['year', 'country', 'code'])
```

In [9]:
```
df
```

Out[9]:

| | year | country | code | medals | population | gdp_per_capita |
|---|---|---|---|---|---|---|
| **0** | 1960 | Afghanistan | AFG | 0 | 8996967.0 | 59.77 |
| **1** | 1960 | Australia | AUS | 46 | 10276477.0 | 1810.62 |
| **2** | 1960 | Austria | AUT | 9 | 7047539.0 | 935.46 |
| **3** | 1960 | Belgium | BEL | 4 | 9153489.0 | 1273.69 |
| **4** | 1960 | Bermuda | BER | 0 | 44400.0 | 1902.40 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2153** | 2016 | Uzbekistan | UZB | 13 | 31847900.0 | 2704.68 |
| **2154** | 2016 | Vanuatu | VAN | 0 | 278326.0 | 2805.67 |
| **2155** | 2016 | Vietnam | VIE | 2 | 93640435.0 | 2192.17 |
| **2156** | 2016 | Zambia | ZAM | 0 | 16363449.0 | 1280.81 |
| **2157** | 2016 | Zimbabwe | ZIM | 0 | 14030338.0 | 1464.59 |

2158 rows × 6 columns

### I am using the PyCountry-Convert Funtion to help me read in continent names for the data.

In [10]:
```
from pycountry_convert import country_alpha2_to_continent_code, country_name_to_count
def get_continent(col):
    try:
        cn_a2_code =  country_name_to_country_alpha2(col)
    except:
        cn_a2_code = 'NaN'
    try:
        cn_continent = country_alpha2_to_continent_code(cn_a2_code)
    except:
        cn_continent = 'NaN'
    return (cn_continent)


df["continent"] = df["country"].apply(lambda col: get_continent(col))
```

In [11]:
```
df['continent'][df.country == 'UK'] = 'EU'
df['continent'][df.country == 'Kosovo'] = 'AS'
df['continent'][df.country == 'Timor-Leste'] = 'AS'
df.loc[df.continent == 'AS', 'continent'] = 'Asia'
df.loc[df.continent == 'OC', 'continent'] = 'Oceania'
df.loc[df.continent == 'EU', 'continent'] = 'Europe'
df.loc[df.continent == 'AF', 'continent'] = 'Africa'
df.loc[df.continent == 'SA', 'continent'] = 'South America'
```

```
df.loc[df.continent == 'NA', 'continent'] = 'North America'
df
```

Out[11]:

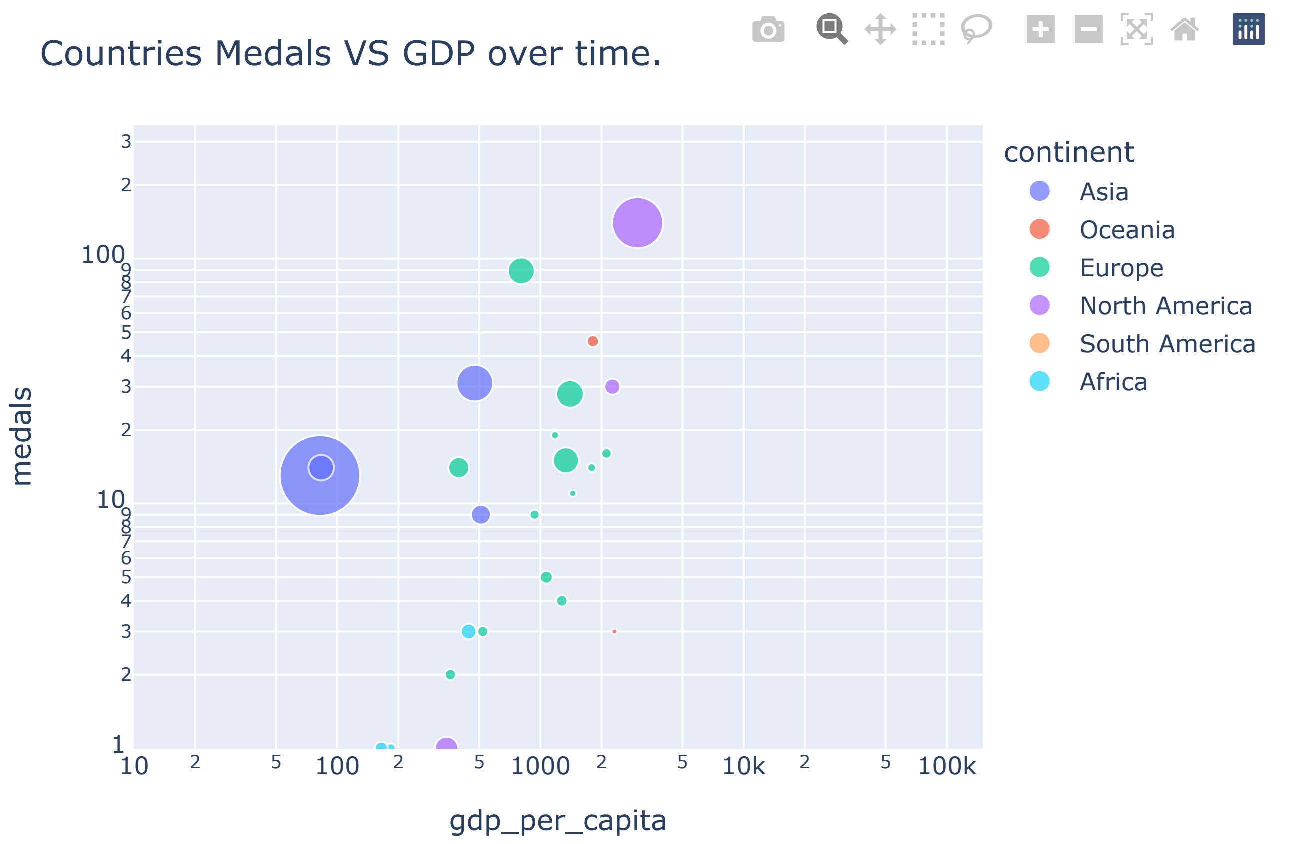
| | year | country | code | medals | population | gdp_per_capita | continent |
|---|---|---|---|---|---|---|---|
| **0** | 1960 | Afghanistan | AFG | 0 | 8996967.0 | 59.77 | Asia |
| **1** | 1960 | Australia | AUS | 46 | 10276477.0 | 1810.62 | Oceania |
| **2** | 1960 | Austria | AUT | 9 | 7047539.0 | 935.46 | Europe |
| **3** | 1960 | Belgium | BEL | 4 | 9153489.0 | 1273.69 | Europe |
| **4** | 1960 | Bermuda | BER | 0 | 44400.0 | 1902.40 | North America |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2153** | 2016 | Uzbekistan | UZB | 13 | 31847900.0 | 2704.68 | Asia |
| **2154** | 2016 | Vanuatu | VAN | 0 | 278326.0 | 2805.67 | Oceania |
| **2155** | 2016 | Vietnam | VIE | 2 | 93640435.0 | 2192.17 | Asia |
| **2156** | 2016 | Zambia | ZAM | 0 | 16363449.0 | 1280.81 | Africa |
| **2157** | 2016 | Zimbabwe | ZIM | 0 | 14030338.0 | 1464.59 | Africa |

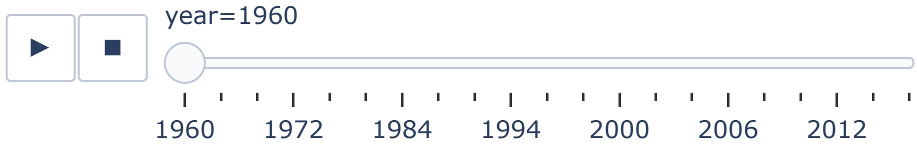
2158 rows × 7 columns

# Now that we have the data cleaned and ready I'm going to create a plotly animation that measures both the GDP per capita population and medal counts along with shedding light on which continent each country it is from.

In [12]:

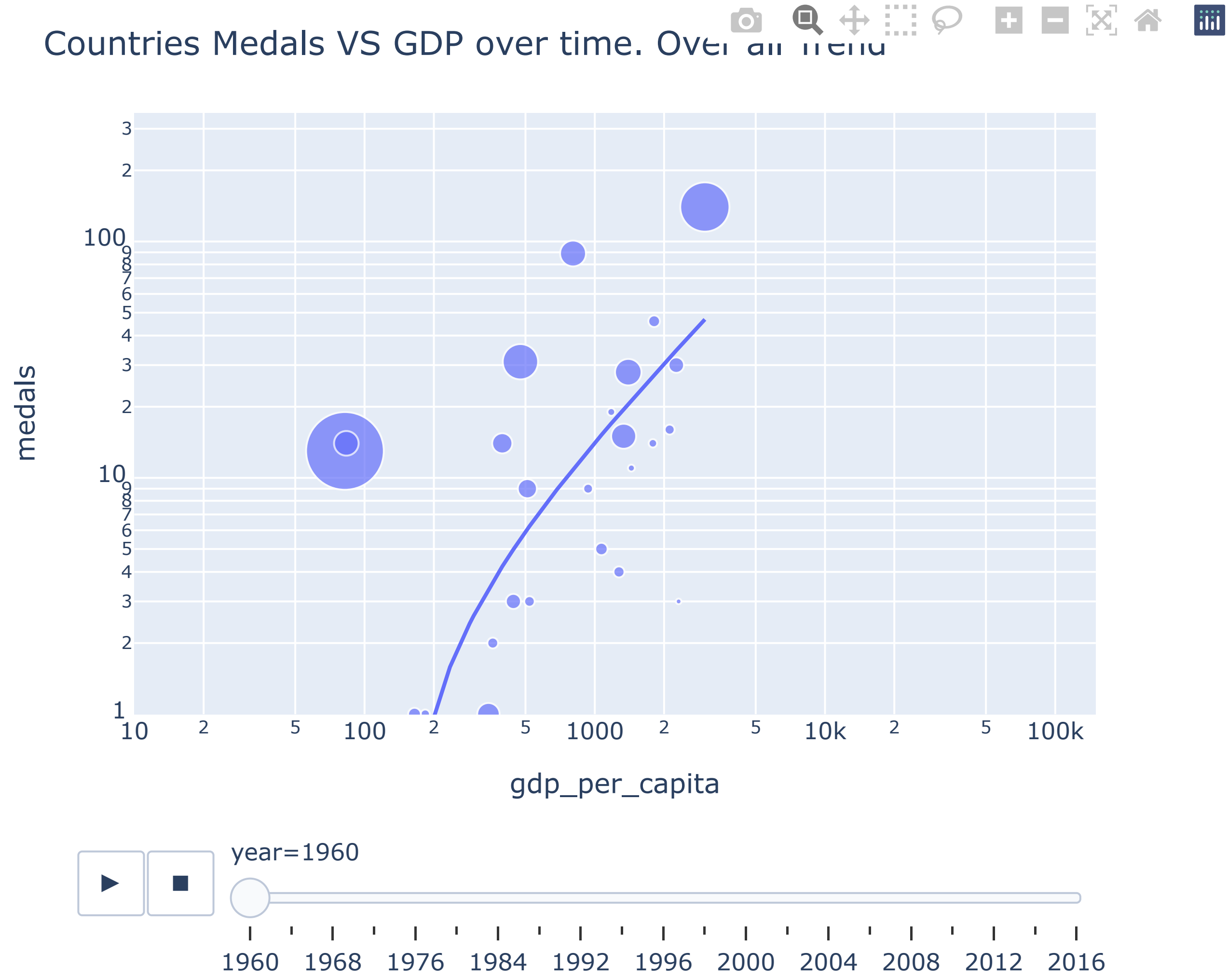
```
fig = px.scatter(df, x="gdp_per_capita", y="medals", animation_frame="year", animatic
           size="population", color="continent", hover_name="country",
           log_y=True, log_x=True, size_max=50, range_x=[10,150000], range_y=[1,350])

fig.update_layout(title="Countries Medals VS GDP over time.")
fig.show()
```






In [13]:

```
fig = px.scatter(df, x="gdp_per_capita", y="medals", animation_frame="year", animatic
           size="population", hover_name="country", trendline= 'ols',
           log_y=True, log_x=True, size_max=50, range_x=[10,150000], range_y=[1,350])

fig.update_layout(title="Countries Medals VS GDP over time. Over all Trend")
fig.show()
results = px.get_trendline_results(fig)
results.px_fit_results.iloc[0].summary()
```




Out[13]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **R-squared:** | 0.259 |
| **Model:** | OLS | **Adj. R-squared:** | 0.244 |
| **Method:** | Least Squares | **F-statistic:** | 17.13 |
| **Date:** | Wed, 22 Dec 2021 | **Prob (F-statistic):** | 0.000137 |
| **Time:** | 17:25:02 | **Log-Likelihood:** | -226.40 |
| **No. Observations:** | 51 | **AIC:** | 456.8 |
| **Df Residuals:** | 49 | **BIC:** | 460.7 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| **const** | -2.2490 | 4.200 | -0.535 | 0.595 | -10.690 | 6.192 |
| **x1** | 0.0163 | 0.004 | 4.139 | 0.000 | 0.008 | 0.024 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 52.578 | **Durbin-Watson:** | 1.872 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 274.522 |
| **Skew:** | 2.707 | **Prob(JB):** | 2.45e-60 |
| **Kurtosis:** | 12.993 | **Cond. No.** | 1.53e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.53e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

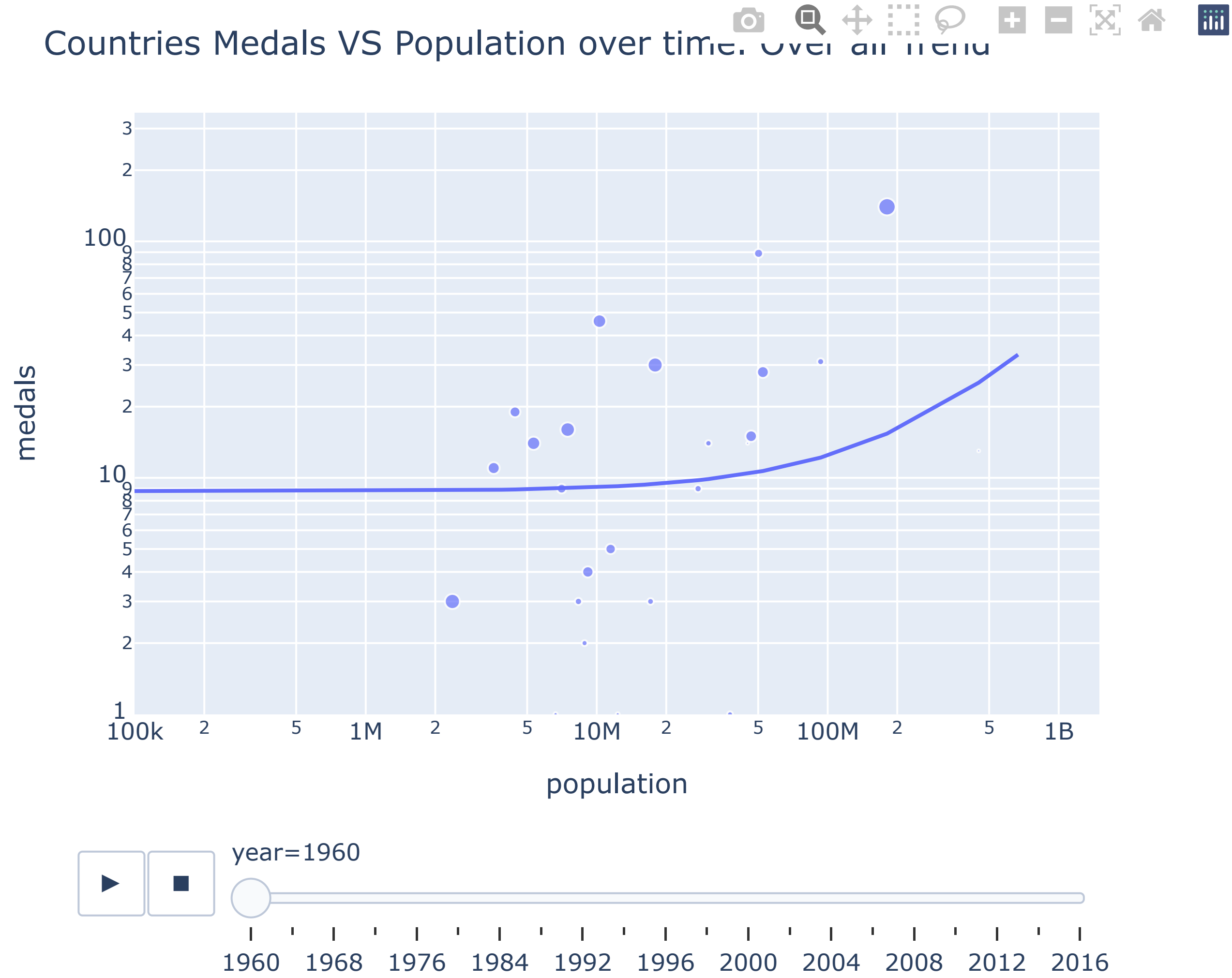
In [14]:

```
fig = px.scatter(df, x="population", y="medals", animation_frame="year", animation_gr
           size="gdp_per_capita", color='continent', hover_name="country",
           log_y=True, log_x=True, size_max=50, range_x=[100000,1500000000], range_y=

fig.update_layout(title="Countries Medals VS Population over time")
fig.show()
```


In [15]:

```
fig = px.scatter(df, x="population", y="medals", animation_frame="year", animation_gr
           size="gdp_per_capita", hover_name="country", trendline= 'ols',
           log_y=True, log_x=True, size_max=50, range_x=[100000,1500000000], range_y=

fig.update_layout(title="Countries Medals VS Population over time. Over all Trend")
fig.show()
```

```
results = px.get_trendline_results(fig)
results.px_fit_results.iloc[0].summary()
```




Out[15]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **R-squared:** | 0.029 |
| **Model:** | OLS | **Adj. R-squared:** | 0.009 |
| **Method:** | Least Squares | **F-statistic:** | 1.469 |
| **Date:** | Wed, 22 Dec 2021 | **Prob (F-statistic):** | 0.231 |
| **Time:** | 17:25:03 | **Log-Likelihood:** | -233.29 |
| **No. Observations:** | 51 | **AIC:** | 470.6 |
| **Df Residuals:** | 49 | **BIC:** | 474.4 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **const** | 8.7652 | 3.559 | 2.463 | 0.017 | 1.613 | 15.917 |
| **x1** | 3.662e-08 | 3.02e-08 | 1.212 | 0.231 | -2.41e-08 | 9.73e-08 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 69.656 | **Durbin-Watson:** | 1.679 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 612.253 |
| **Skew:** | 3.688 | **Prob(JB):** | 1.12e-133 |
| **Kurtosis:** | 18.288 | **Cond. No.** | 1.25e+08 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.25e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

In [16]:
```
%%sql totpercentsum <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PEF
FROM sportstats
WHERE season= 'Winter'
GROUP BY region
HAVING COUNT(medal) > 0
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
41 rows affected.
Returning data to local variable totpercentsum
```

In [17]:
```
%%sql goldpercentsum <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PEF
FROM sportstats
WHERE season= 'Winter'
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Gold'
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
33 rows affected.
Returning data to local variable goldpercentsum
```

In [18]:
```
%%sql silverpercentsum <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PEF
FROM sportstats
WHERE season= 'Winter'
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Silver'
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
36 rows affected.
Returning data to local variable silverpercentsum
```

In [19]:
```
%%sql bronzepercentsum <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PEF
FROM sportstats
WHERE season= 'Winter'
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Bronze'
Order by perc DESC;
```

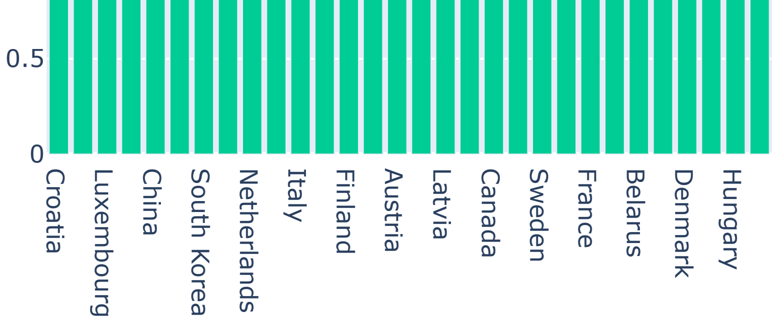
```
 * postgresql+psycopg2://postgres:***@localhost/postgres
35 rows affected.
Returning data to local variable bronzepercentsum
```

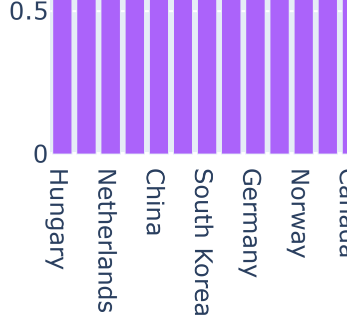
In [20]:
```
totalsum = pd.DataFrame(totpercentsum)
totalsum.columns = ['region', 'medals']
sumtot =totalsum.head(30)
goldsum = pd.DataFrame(goldpercentsum)
goldsum.columns = ['region', 'medals']
goldsum = goldsum.head(30)
```

```
silversum= pd.DataFrame(silverpercentsum)
silversum.columns = ['region', 'medals']
silversum = silversum.head(30)
bronzesum= pd.DataFrame(bronzepercentsum)
bronzesum.columns = ['region', 'medals']
bronzesum = bronzesum.head(30)
```

In [21]:

```
fig = make_subplots(rows=2,cols=2, subplot_titles=("Total", "Gold", "Silver", "Bronze

fig.add_trace(go.Bar(x = sumtot.region, y= sumtot.medals, name='Total'), row=1, col=1
fig.add_trace(go.Bar(x = goldsum.region, y= goldsum.medals, name='Gold'), row=1, col=
fig.add_trace(go.Bar(x = silversum.region, y= silversum.medals, name='Silver'), row=2
fig.add_trace(go.Bar(x = bronzesum.region, y= bronzesum.medals, name='Bronze'), row=2

fig.update_layout(height=1000, width=1000, title = 'Per Capita Medal Winners Winter')
fig.show()
```




# Hypothesis Check

## Hypothesis 1:

From the animation and what I have gathered it definitely seems like both have and effect with positive correlation towards more medals. The main difference between population and GDP per capita is though that population has always had a mild steady correlation with medals while GDP per capita use to have a much stronger correlation but that has slowly diminished over time to not much more of a correlation than population correlation with metals.

## Hypothesis 2 :

There are a few outliers and if we were just to consider the Winter olympics it appears that most of the world does not really compete and the vast majority of the medals are won by those from Northern nations in North America, Europe and Asia.

# Hypothesis III and IV

To start I think I'm just going to look at the breakdown by metal by country from there we can break it into height and weight as well. that way we have a general overview of how it has been an overtime.

In [23]:
```
%%sql
SELECT *
FROM sportstats
LIMIT 5
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
5 rows affected.
```

Out[23]:

| id | name | age | height | weight | Sex | team | games | year | season | sport | event | me |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 87378.0 | Ramazan Nuristani | None | None | None | M | Afghanistan | 1956 Summer | 1956.0 | Summer | Hockey | Hockey Men's Hockey | Nc |
| 87377.0 | Noor Ullah Nuristani | None | None | None | M | Afghanistan | 1956 Summer | 1956.0 | Summer | Hockey | Hockey Men's Hockey | Nc |
| 87375.0 | Mohammad Jahan Nuristani | None | None | None | M | Afghanistan | 1948 Summer | 1948.0 | Summer | Hockey | Hockey Men's Hockey | Nc |
| 87374.0 | Mohammad Amin Nuristani | None | None | None | M | Afghanistan | 1956 Summer | 1956.0 | Summer | Hockey | Hockey Men's Hockey | Nc |
| 87373.0 | Jahan Gulam Nuristani | None | None | None | M | Afghanistan | 1948 Summer | 1948.0 | Summer | Hockey | Hockey Men's Hockey | Nc |

In [59]:

```
%%sql
SELECT COUNT(medal) AS medals_won, region
FROM sportstats
Group BY region
ORDER BY medals_won desc
limit 10
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
10 rows affected.
```

Out[59]:

| medals_won | region |
|---|---|
| 5436 | USA |
| 3945 | Russia |
| 3752 | Germany |
| 2065 | UK |
| 1747 | France |
| 1637 | Italy |
| 1534 | Sweden |
| 1349 | Australia |
| 1344 | Canada |
| 1135 | Hungary |

In [25]:

```
%%sql gold_won <<
SELECT COUNT(medal) AS medals_won, region
FROM sportstats
Group BY region, medal
HAVING medal = 'Gold'
ORDER BY medals_won desc;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
99 rows affected.
Returning data to local variable gold_won
```

In [26]:

```
%%sql silver_won <<
SELECT COUNT(medal) AS medals_won, region
FROM sportstats
Group BY region, medal
HAVING medal = 'Silver'
ORDER BY medals_won desc;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
116 rows affected.
Returning data to local variable silver_won
```

In [27]:

```
%%sql bronze_won <<
SELECT COUNT(medal) AS medals_won, region
FROM sportstats
Group BY region, medal
HAVING medal = 'Bronze'
ORDER BY medals_won desc;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
112 rows affected.
Returning data to local variable bronze_won
```

In [28]:

```
%%sql total_medals_won <<
SELECT COUNT(medal) AS medals_won, region
FROM sportstats
```

```
Group BY region
ORDER BY medals_won desc
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
207 rows affected.
Returning data to local variable total_medals_won
```

In [29]:

```
total = pd.DataFrame(total_medals_won)
total.columns = ['medals', 'region']
toptot = total.head(30)
```

In [30]:

```
gold = pd.DataFrame(gold_won)
gold.columns = ['medals', 'region']
topgold = gold.head(30)
```

In [31]:

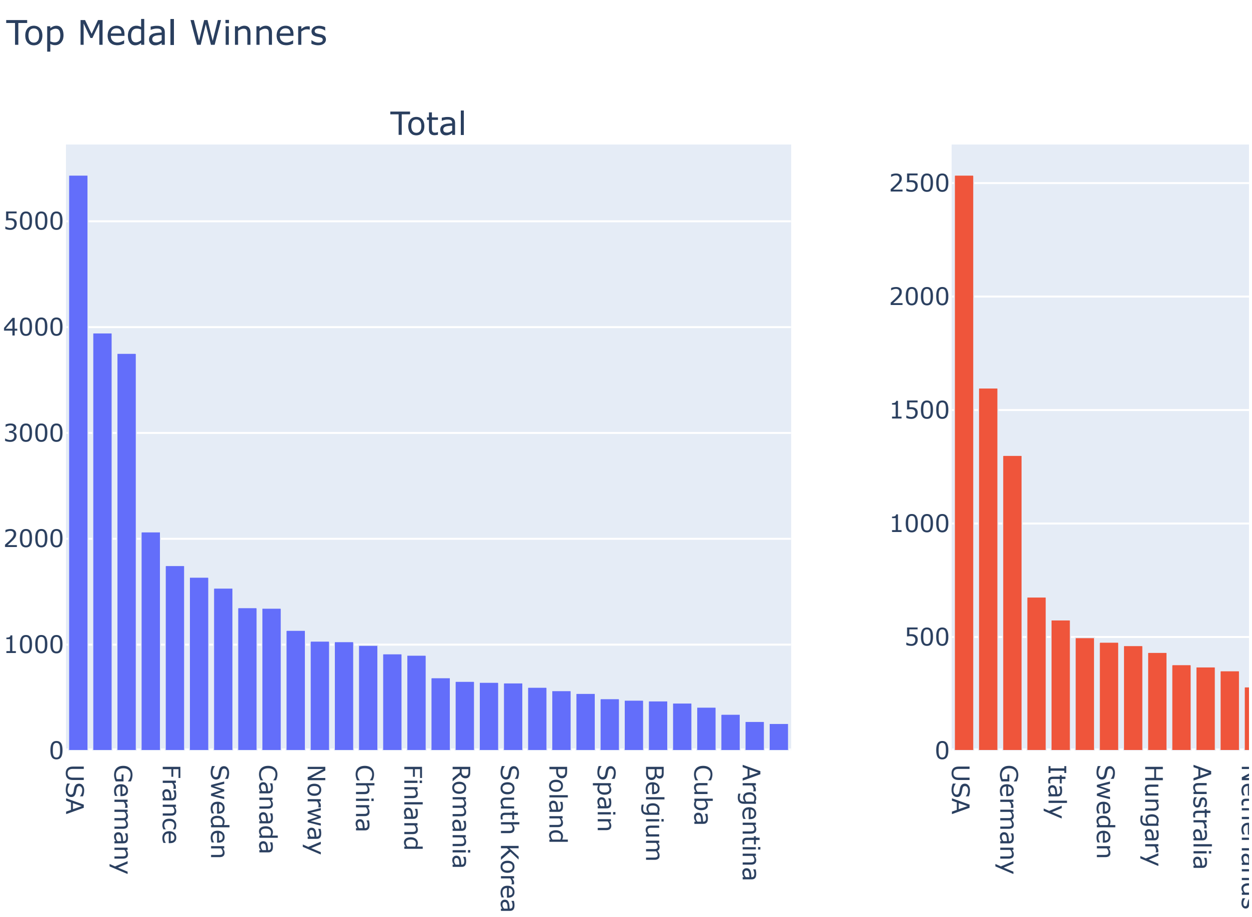
```
silver = pd.DataFrame(silver_won)
silver.columns = ['medals', 'region']
topslvr = silver.head(30)
```

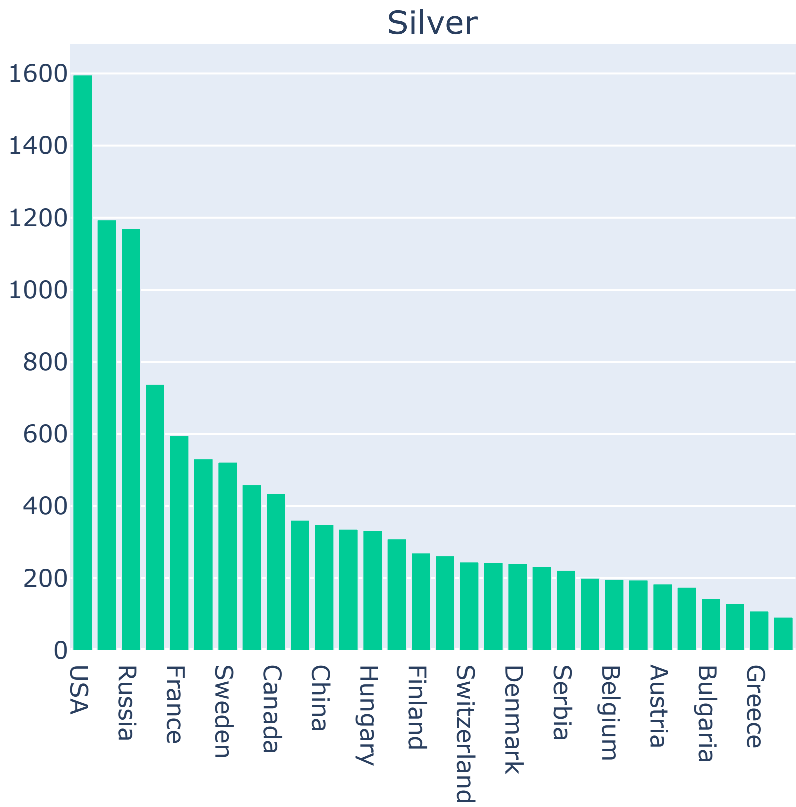
In [32]:

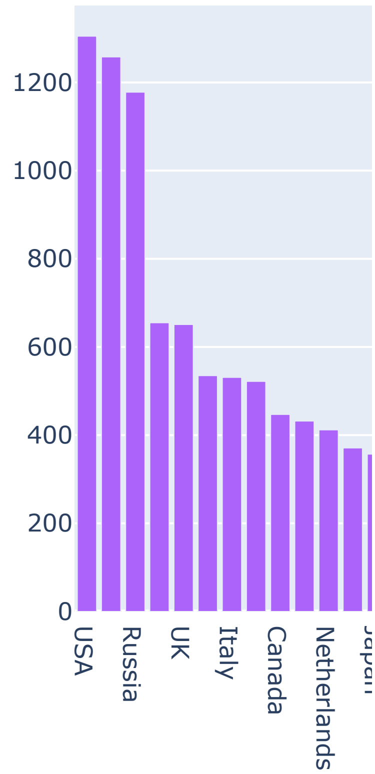
```
bronze = pd.DataFrame(bronze_won)
bronze.columns = ['medals', 'region']
topbr = bronze.head(30)
```

In [33]:

```
fig = make_subplots(rows=2,cols=2, subplot_titles=("Total", "Gold", "Silver", "Bronze

fig.add_trace(go.Bar(x = toptot.region, y= toptot.medals, name='Total'), row=1, col=1
fig.add_trace(go.Bar(x = topgold.region, y= topgold.medals, name='Gold'), row=1, col=
fig.add_trace(go.Bar(x = topslvr.region, y= topslvr.medals, name='Silver'), row=2, co
fig.add_trace(go.Bar(x = topbr.region, y= topbr.medals, name='Bronze'), row=2, col=2)

fig.update_layout(height=1000, width=1000, title = 'Top Medal Winners')
fig.show()
```






In [34]:

```
%%sql totpercent <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PE
FROM sportstats
GROUP BY region
HAVING COUNT(medal) > 0
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
136 rows affected.
Returning data to local variable totpercent
```

In [35]:

```
%%sql totpercent <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PE
FROM sportstats
GROUP BY region
HAVING COUNT(medal) > 0
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
136 rows affected.
Returning data to local variable totpercent
```

In [36]:

```
%%sql goldpercent <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PE
FROM sportstats
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Gold'
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
99 rows affected.
Returning data to local variable goldpercent
```

In [37]:
```
%%sql silverpercent <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PER
FROM sportstats
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Silver'
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
116 rows affected.
Returning data to local variable silverpercent
```

In [38]:
```
%%sql bronzepercent <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PER
FROM sportstats
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Bronze'
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
112 rows affected.
Returning data to local variable bronzepercent
```

In [39]:
```
%%sql goldpercent <<
SELECT region, CAST(COUNT(medal) AS float)/CAST(COUNT(DISTINCT name) AS float) as PER
FROM sportstats
GROUP BY region, medal
HAVING COUNT(medal) > 0
AND medal = 'Gold'
Order by perc DESC;
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
99 rows affected.
Returning data to local variable goldpercent
```

In [40]:
```
totalperc = pd.DataFrame(totpercent)
totalperc.columns = ['region', 'medals']
perctot =totalperc.head(30)
goldperc = pd.DataFrame(goldpercent)
goldperc.columns = ['region', 'medals']
percgold = goldperc.head(30)
silverperc= pd.DataFrame(silverpercent)
silverperc.columns = ['region', 'medals']
percslvr = silverperc.head(30)
bronzeperc= pd.DataFrame(bronzepercent)
bronzeperc.columns = ['region', 'medals']
percbrnz = bronzeperc.head(30)
```

In [41]:
```
silverperc= pd.DataFrame(silverpercent)
silverperc.columns = ['region', 'medals']
percslvr = silverperc.head(30)
percslvr
```

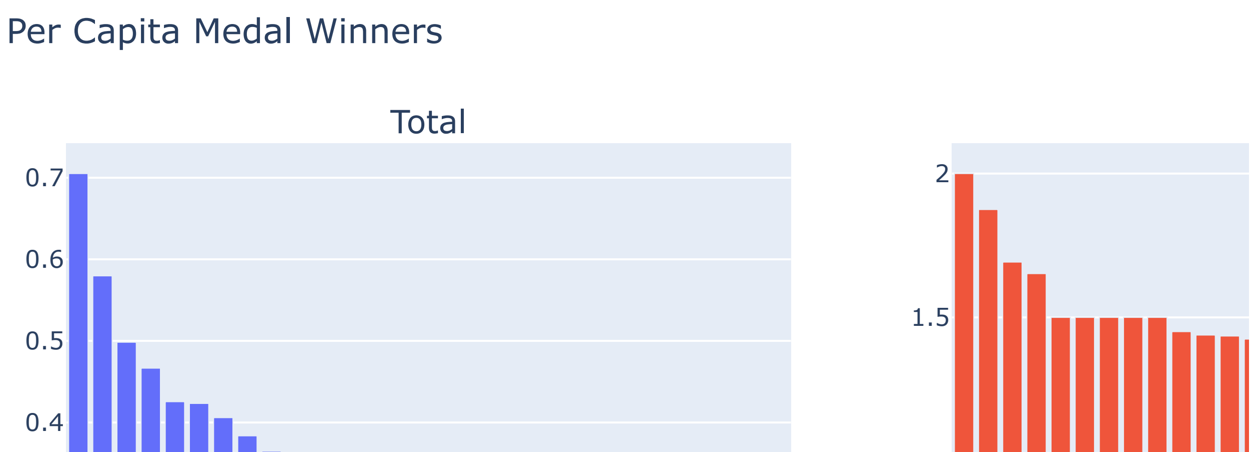
Out[41]:

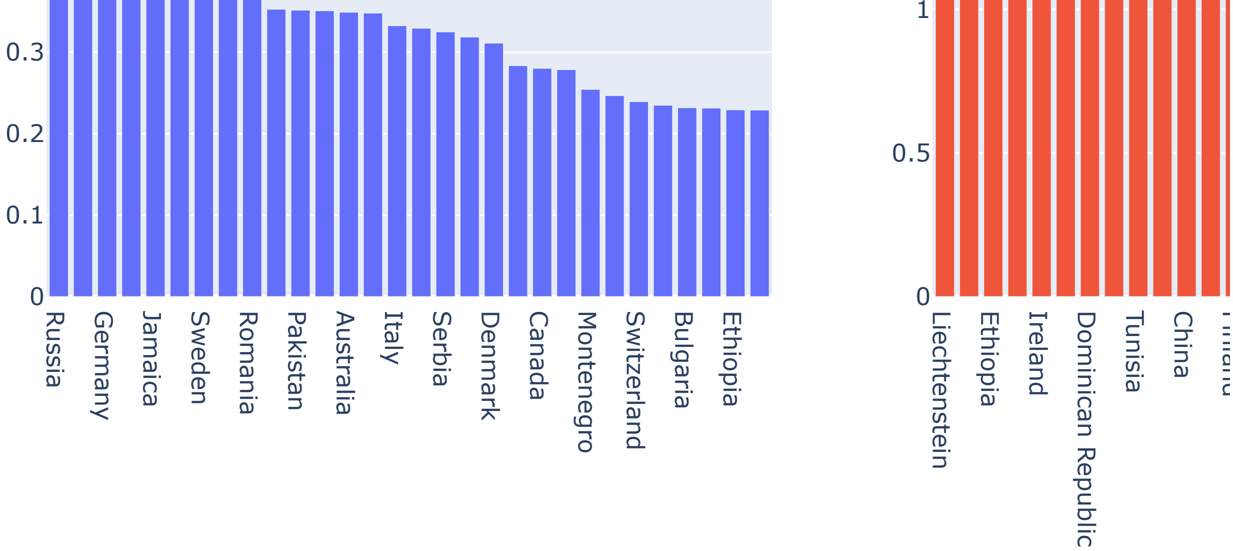
| | region | medals |
|---|---|---|
| **0** | Namibia | 4.000000 |
| **1** | Zimbabwe | 4.000000 |
| **2** | Tunisia | 1.500000 |
| **3** | Slovakia | 1.357143 |
| **4** | Jamaica | 1.339286 |
| **5** | Algeria | 1.333333 |

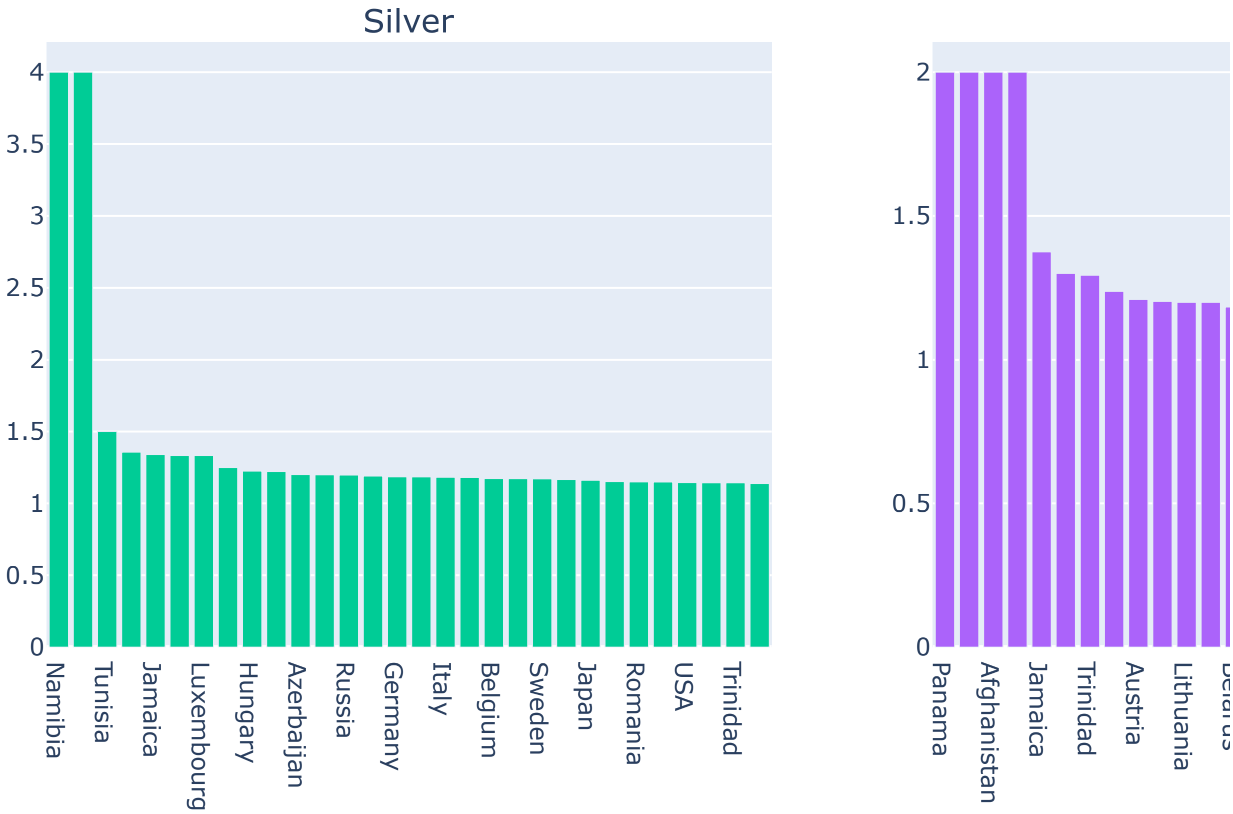
| | | |
|---|---|---|
| **6** | Luxembourg | 1.333333 |
| **7** | Norway | 1.249135 |
| **8** | Hungary | 1.225092 |
| **9** | Malaysia | 1.222222 |
| **10** | Azerbaijan | 1.200000 |
| **11** | Australia | 1.198433 |
| **12** | Russia | 1.197544 |
| **13** | Kazakhstan | 1.190476 |
| **14** | Germany | 1.184524 |
| **15** | Pakistan | 1.184211 |
| **16** | Italy | 1.182628 |
| **17** | Slovenia | 1.181818 |
| **18** | Belgium | 1.172619 |
| **19** | China | 1.171141 |
| **20** | Sweden | 1.170404 |
| **21** | Switzerland | 1.166667 |
| **22** | Japan | 1.161654 |
| **23** | Brazil | 1.151316 |
| **24** | Romania | 1.149425 |
| **25** | Finland | 1.148936 |
| **26** | USA | 1.144086 |
| **27** | Thailand | 1.142857 |
| **28** | Trinidad | 1.142857 |
| **29** | Kenya | 1.138889 |

In [42]:

```
fig = make_subplots(rows=2,cols=2, subplot_titles=("Total", "Gold", "Silver", "Bronze

fig.add_trace(go.Bar(x = perctot.region, y= perctot.medals, name='Total'), row=1, col
fig.add_trace(go.Bar(x = percgold.region, y= percgold.medals, name='Gold'), row=1, co
fig.add_trace(go.Bar(x = percslvr.region, y= percslvr.medals, name='Silver'), row=2,
fig.add_trace(go.Bar(x = percbrnz.region, y= percbrnz.medals, name='Bronze'), row=2,

fig.update_layout(height=1000, width=1000, title = 'Per Capita Medal Winners')
fig.show()
```






Now that we have really explored the medal counts per country and breaking it down by per capita participants battles. We have seen that on a per-capita basis smaller countries actually tend to do well. It is partially contradictory to my first hypothesis especially when you consider per capita it is the smaller countries who are at the top. Who would have guessed that Lichtenstein and Ethiopia hav the most Gold medals per capita.

# Let us move on to height and weight though.

In [43]:

```
%%sql height_weight <<
SELECT Year, AVG(height), AVG(weight) FROM sportstats group by year
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
36 rows affected.
Returning data to local variable height_weight
```

In [44]:
```
%%sql height_weightgold <<
SELECT Year, AVG(height), AVG(weight) FROM sportstats Where medal='Gold' group by yea
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
35 rows affected.
Returning data to local variable height_weightgold
```

In [45]:
```
%%sql height_weightsilver <<
SELECT Year, AVG(height), AVG(weight) FROM sportstats Where medal='Silver' group by y
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
35 rows affected.
Returning data to local variable height_weightsilver
```

In [46]:
```
%%sql height_weightbronze <<
SELECT Year, AVG(height), AVG(weight) FROM sportstats Where medal='Bronze' group by y
```

```
 * postgresql+psycopg2://postgres:***@localhost/postgres
35 rows affected.
Returning data to local variable height_weightbronze
```

In [47]:
```
hw = pd.DataFrame(height_weight)
hw.columns = ['year', 'height', 'weight']
hwg = pd.DataFrame(height_weightgold)
hwg.columns = ['year', 'height', 'weight']
hws = pd.DataFrame(height_weightsilver)
hws.columns = ['year', 'height', 'weight']
hwb = pd.DataFrame(height_weightbronze)
hwb.columns = ['year', 'height', 'weight']
```

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

Out[48]:

| | year | height | weight |
|---|---|---|---|
| 0 | 1896.0 | 170.666667 | 65.666667 |
| 1 | 1900.0 | 178.812500 | 74.200000 |
| 2 | 1904.0 | 176.304348 | 75.571429 |
| 3 | 1906.0 | 178.900000 | 80.000000 |
| 4 | 1908.0 | 177.108696 | 74.430769 |

In [49]:
```
t_g_hw = pd.merge(hw, hwg, on ='year')
s_b_hw = pd.merge(hws, hwb, on ='year')
mergehw = pd.merge(t_g_hw, s_b_hw, on ='year')
mergehw.columns= ['year', 'avg_height', 'avg_weight', 'gold_height', 'gold_weight', '
mergehw
```

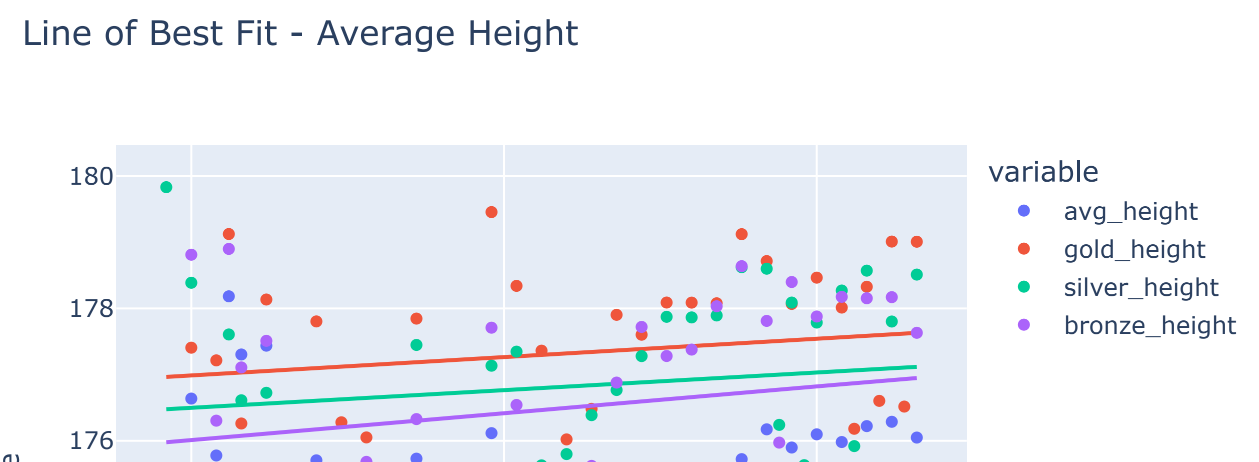
Out[49]:

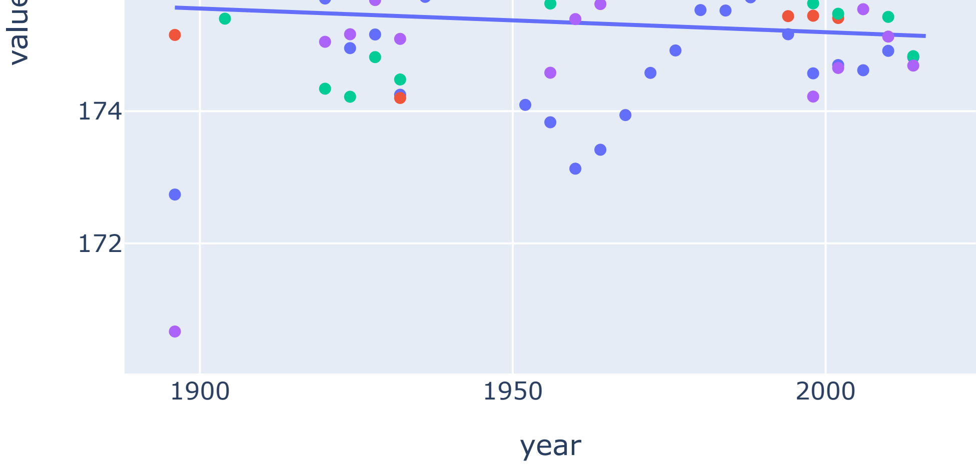
| | year | avg_height | avg_weight | gold_height | gold_weight | silver_height | silver_weight | bronze_height | brc |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1896.0 | 172.739130 | 71.387755 | 175.153846 | 72.153846 | 179.833333 | 76.833333 | 170.666667 | |
| 1 | 1900.0 | 176.637931 | 74.556962 | 177.407407 | 72.857143 | 178.388889 | 71.076923 | 178.812500 | |
| 2 | 1904.0 | 175.778302 | 72.275862 | 177.215686 | 71.733333 | 175.400000 | 72.238095 | 176.304348 | |
| 3 | 1906.0 | 178.183594 | 75.921569 | 179.125000 | 77.034483 | 177.607143 | 77.000000 | 178.900000 | |
| 4 | 1908.0 | 177.304348 | 75.070513 | 176.261538 | 76.423077 | 176.612245 | 76.514286 | 177.108696 | |
| 5 | 1912.0 | 177.437936 | 73.081081 | 178.135135 | 74.754098 | 176.725490 | 74.893617 | 177.511111 | |
| 6 | 1920.0 | 175.704724 | 72.950959 | 177.804348 | 74.037500 | 174.340426 | 74.304878 | 175.050000 | |
| 7 | 1924.0 | 174.954119 | 71.681223 | 176.279661 | 70.698630 | 174.218750 | 74.814286 | 175.164179 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **8** | 1928.0 | 175.160455 | 70.994366 | 176.051020 | 72.484848 | 174.818182 | 72.309091 | 175.681818 |
| **9** | 1932.0 | 174.248938 | 70.410714 | 174.200000 | 71.123077 | 174.479675 | 71.175439 | 175.094340 |
| **10** | 1936.0 | 175.730118 | 71.488277 | 177.846847 | 73.829787 | 177.448718 | 72.490909 | 176.328125 |
| **11** | 1948.0 | 176.115917 | 71.543058 | 179.456790 | 74.114943 | 177.135135 | 73.285714 | 177.709677 |
| **12** | 1952.0 | 174.095588 | 69.987109 | 178.341463 | 76.062992 | 177.347368 | 73.314607 | 176.541667 |
| **13** | 1956.0 | 173.832202 | 70.305545 | 177.362245 | 75.251309 | 175.628866 | 72.451282 | 174.582090 |
| **14** | 1960.0 | 173.130440 | 69.297825 | 176.021084 | 72.432927 | 175.799373 | 72.740506 | 175.393293 |
| **15** | 1964.0 | 173.416752 | 69.672856 | 176.483461 | 73.588235 | 176.387755 | 73.817010 | 175.620779 |
| **16** | 1968.0 | 173.941503 | 69.581497 | 177.904077 | 74.346988 | 176.767726 | 73.293399 | 176.879808 |
| **17** | 1972.0 | 174.579646 | 70.027027 | 177.603412 | 74.377919 | 177.280353 | 73.789357 | 177.721174 |
| **18** | 1976.0 | 174.918925 | 70.130863 | 178.089463 | 74.929577 | 177.874239 | 74.287755 | 177.281437 |
| **19** | 1980.0 | 175.530201 | 70.719890 | 178.088462 | 74.451923 | 177.864762 | 74.022945 | 177.378531 |
| **20** | 1984.0 | 175.523165 | 70.228378 | 178.078431 | 73.725490 | 177.894444 | 73.022181 | 178.033451 |
| **21** | 1988.0 | 175.722973 | 70.469093 | 179.122924 | 74.858804 | 178.623946 | 74.300169 | 178.639291 |
| **22** | 1992.0 | 176.173326 | 71.136291 | 178.716239 | 74.650602 | 178.600977 | 74.467532 | 177.812006 |
| **23** | 1994.0 | 175.164028 | 71.033030 | 175.438095 | 71.800000 | 176.240741 | 73.333333 | 175.972727 |
| **24** | 1996.0 | 175.898702 | 70.907016 | 178.068783 | 73.664273 | 178.089127 | 73.482206 | 178.400000 |
| **25** | 1998.0 | 174.572484 | 70.918089 | 175.444444 | 72.958333 | 175.631944 | 72.388889 | 174.221477 |
| **26** | 2000.0 | 176.097879 | 71.124095 | 178.465961 | 74.412747 | 177.786910 | 73.321157 | 177.878698 |
| **27** | 2002.0 | 174.698234 | 71.185559 | 175.409938 | 73.043750 | 175.474359 | 73.346154 | 174.654088 |
| **28** | 2004.0 | 175.981542 | 71.291159 | 178.013575 | 74.200603 | 178.271212 | 74.040909 | 178.176036 |
| **29** | 2006.0 | 174.619657 | 70.513402 | 176.181818 | 73.744318 | 175.920000 | 72.514286 | 175.542857 |
| **30** | 2008.0 | 176.222098 | 71.411892 | 178.328358 | 74.286145 | 178.571644 | 74.358974 | 178.154173 |
| **31** | 2010.0 | 174.913330 | 70.723594 | 176.603448 | 74.075145 | 175.426901 | 72.255952 | 175.128655 |
| **32** | 2012.0 | 176.287254 | 71.341748 | 179.011094 | 74.443730 | 177.802862 | 73.166934 | 178.170877 |
| **33** | 2014.0 | 174.814914 | 70.754604 | 176.517413 | 73.529101 | 174.832487 | 72.015789 | 174.690355 |
| **34** | 2016.0 | 176.049047 | 71.005510 | 179.009063 | 74.503030 | 178.512232 | 73.840491 | 177.632479 |

In [50]:

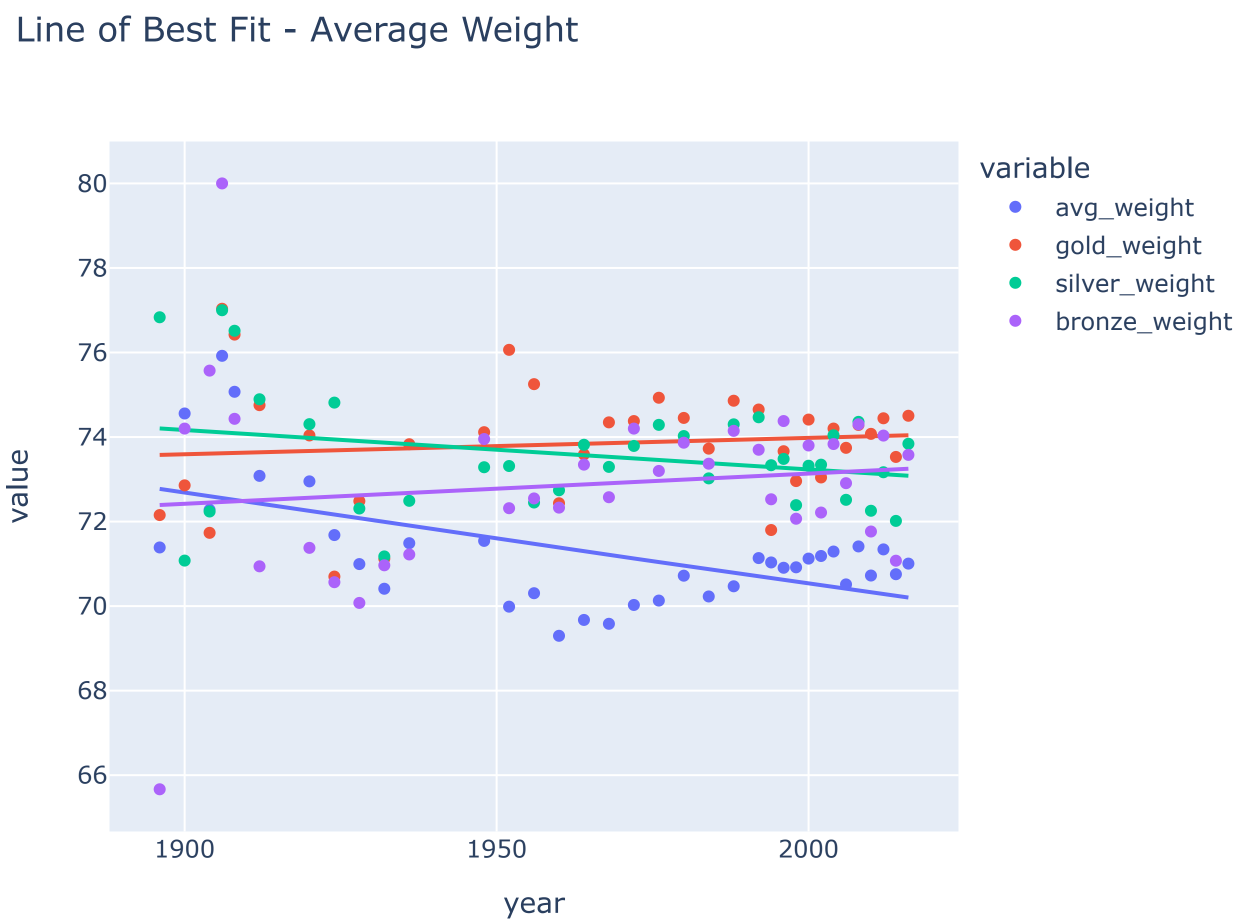
```
fig = px.scatter(mergehw, x='year', y=['avg_height', 'gold_height', 'silver_height',
                 title='Line of Best Fit - Average Height')
fig.show()
```






In [51]:

```
fig = px.scatter(mergehw, x='year', y=['avg_weight', 'gold_weight', 'silver_weight',
            title='Line of Best Fit - Average Weight')
fig.show()
results = px.get_trendline_results(fig)
results.px_fit_results.iloc[0].summary()
```




Out[51]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **R-squared:** | 0.322 |
| **Model:** | OLS | **Adj. R-squared:** | 0.301 |
| **Method:** | Least Squares | **F-statistic:** | 15.66 |
| **Date:** | Wed, 22 Dec 2021 | **Prob (F-statistic):** | 0.000380 |
| **Time:** | 17:25:12 | **Log-Likelihood:** | -55.836 |
| **No. Observations:** | 35 | **AIC:** | 115.7 |

| | | | |
|---|---|---|---|
| **Df Residuals:** | 33 | **BIC:** | 118.8 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 388.9011 | 80.258 | 4.846 | 0.000 | 225.616 | 552.186 |
| **x1** | -96.4440 | 24.370 | -3.957 | 0.000 | -146.026 | -46.862 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3.255 | **Durbin-Watson:** | 0.894 |
| **Prob(Omnibus):** | 0.196 | **Jarque-Bera (JB):** | 2.071 |
| **Skew:** | 0.556 | **Prob(JB):** | 0.355 |
| **Kurtosis:** | 3.426 | **Cond. No.** | 1.39e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [52]:

```
results = px.get_trendline_results(fig)
results.px_fit_results.iloc[0].summary()
```

Out[52]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **R-squared:** | 0.322 |
| **Model:** | OLS | **Adj. R-squared:** | 0.301 |
| **Method:** | Least Squares | **F-statistic:** | 15.66 |
| **Date:** | Wed, 22 Dec 2021 | **Prob (F-statistic):** | 0.000380 |
| **Time:** | 17:25:12 | **Log-Likelihood:** | -55.836 |
| **No. Observations:** | 35 | **AIC:** | 115.7 |
| **Df Residuals:** | 33 | **BIC:** | 118.8 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 388.9011 | 80.258 | 4.846 | 0.000 | 225.616 | 552.186 |
| **x1** | -96.4440 | 24.370 | -3.957 | 0.000 | -146.026 | -46.862 |

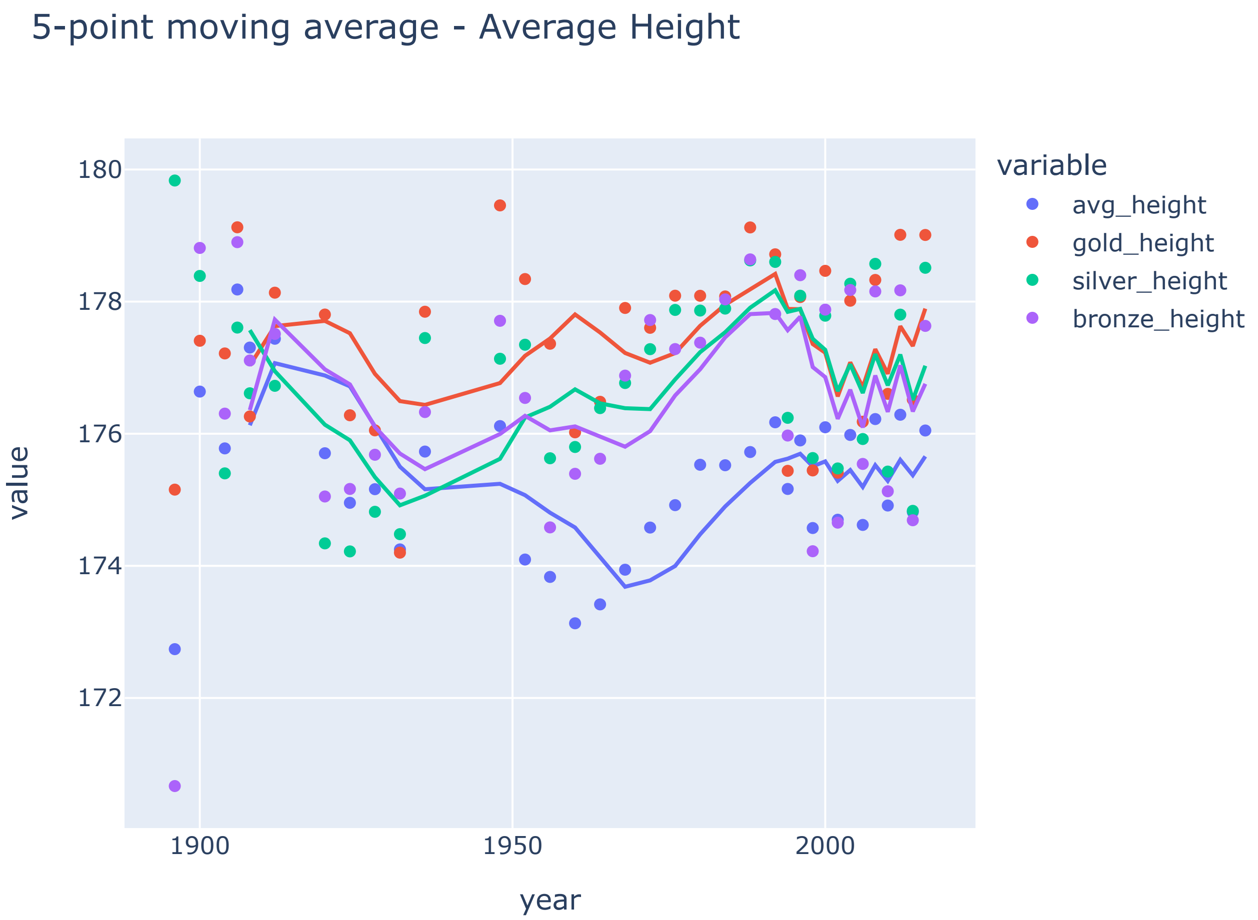
| | | | |
|---|---|---|---|
| **Omnibus:** | 3.255 | **Durbin-Watson:** | 0.894 |
| **Prob(Omnibus):** | 0.196 | **Jarque-Bera (JB):** | 2.071 |
| **Skew:** | 0.556 | **Prob(JB):** | 0.355 |
| **Kurtosis:** | 3.426 | **Cond. No.** | 1.39e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.39e+03. This might indicate that there are strong multicollinearity or other numerical problems.

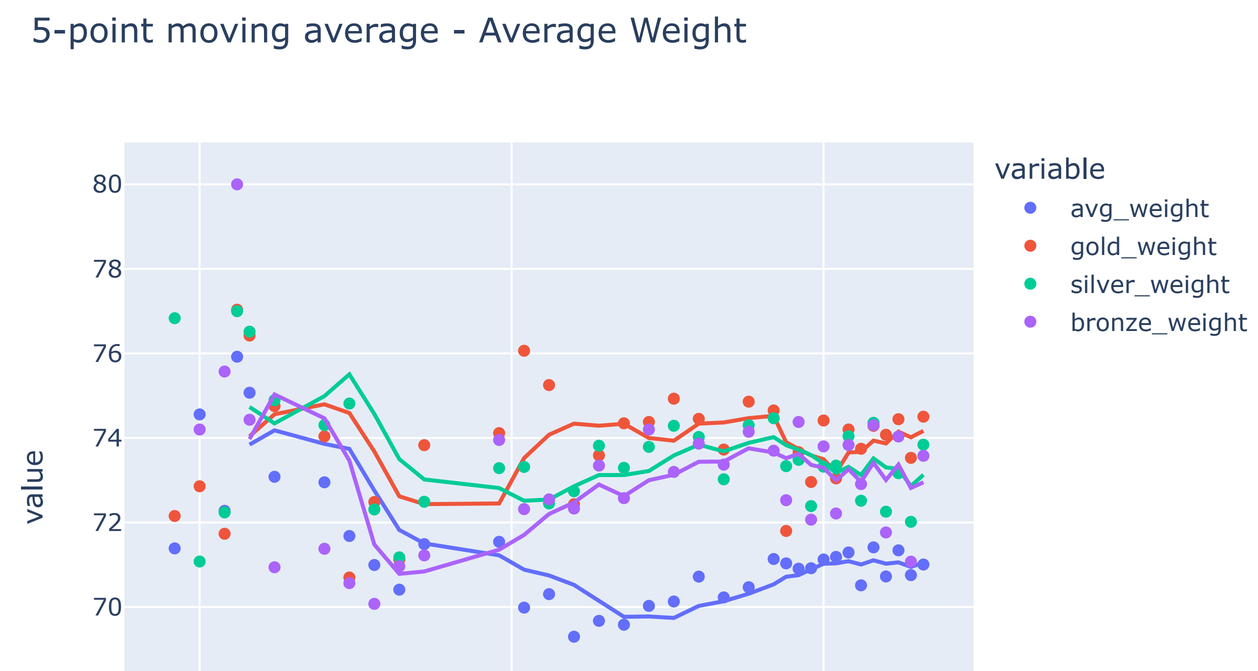
In [53]:

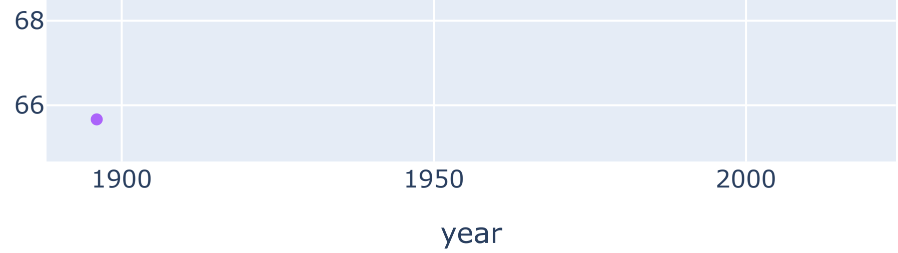
```
fig = px.scatter(mergehw, x='year', y=['avg_height', 'gold_height', 'silver_height',
                 title="5-point moving average - Average Height")
fig.show()
```




In [54]:

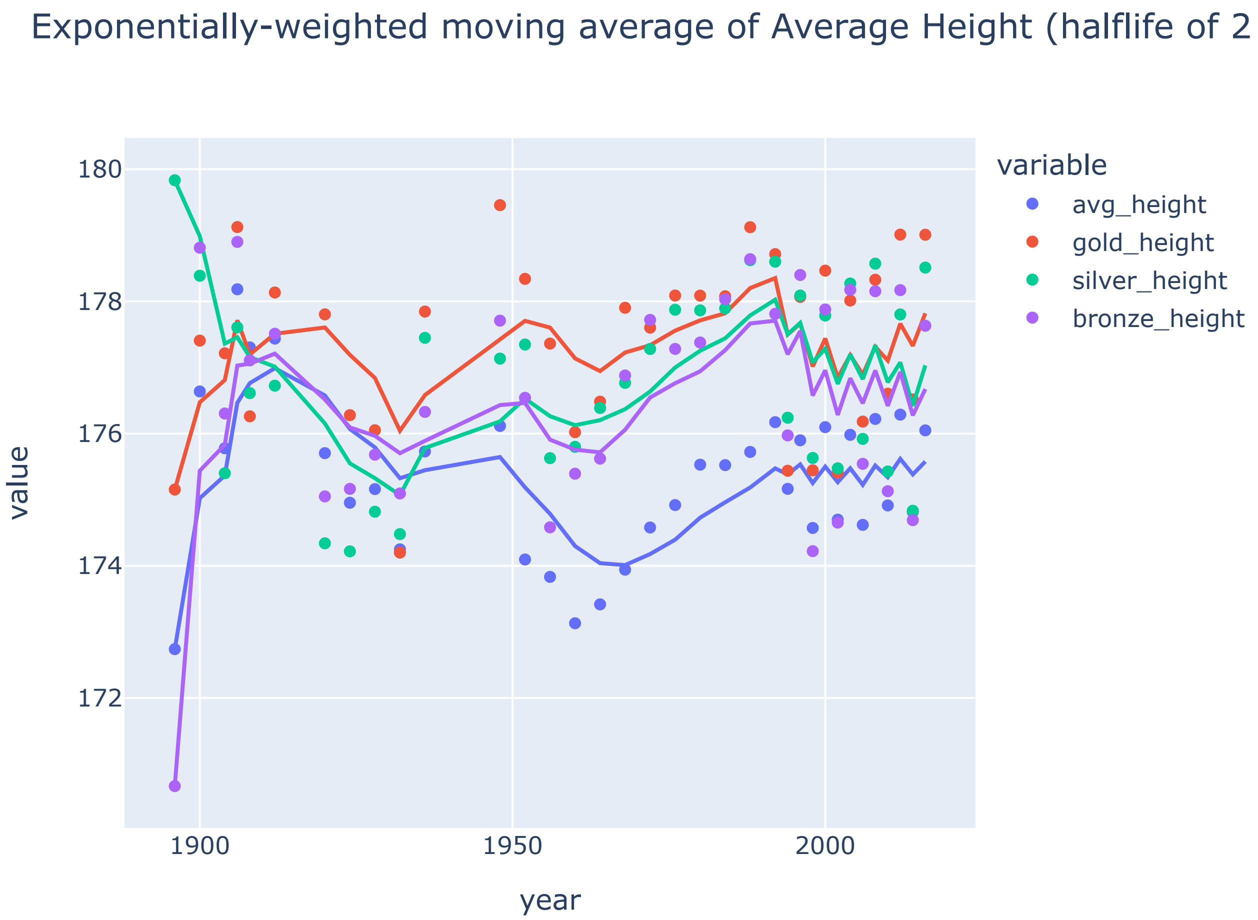
```
fig = px.scatter(mergehw, x='year', y=['avg_weight', 'gold_weight', 'silver_weight',
                 title="5-point moving average - Average Weight")
fig.show()
```





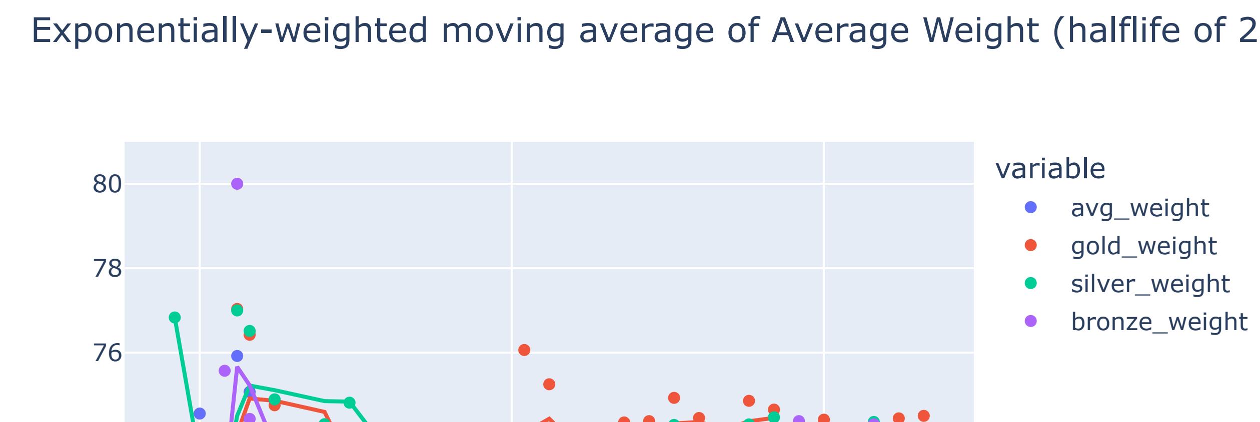

In [55]:

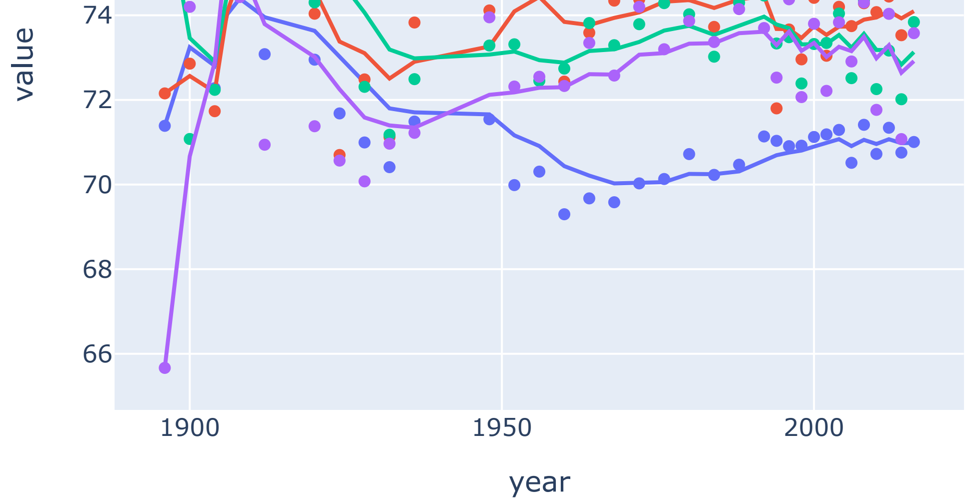
```
fig = px.scatter(mergehw, x='year', y=['avg_height', 'gold_height', 'silver_height', 
                title="Exponentially-weighted moving average of Average Height (halfl
fig.show()
```




In [56]:

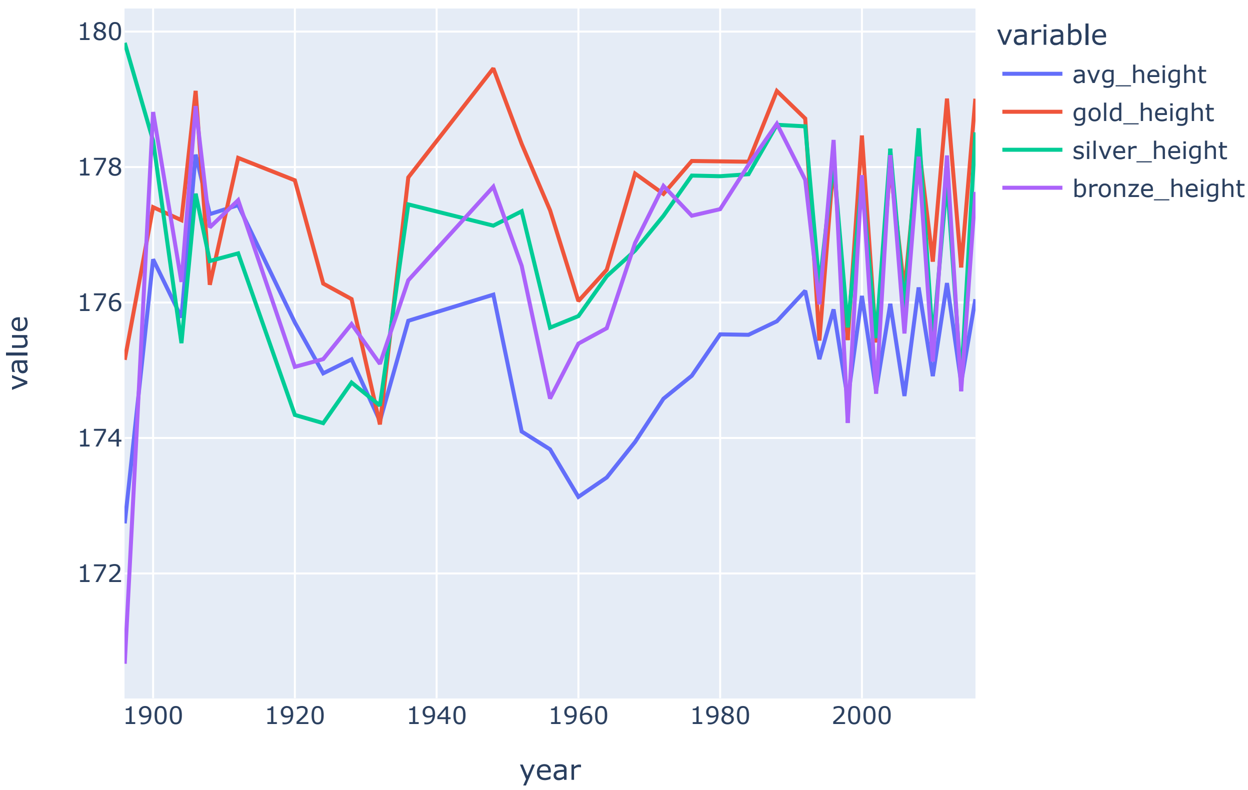
```
fig = px.scatter(mergehw, x='year', y=['avg_weight', 'gold_weight', 'silver_weight', 
                title="Exponentially-weighted moving average of Average Weight (halfl
fig.show()
```





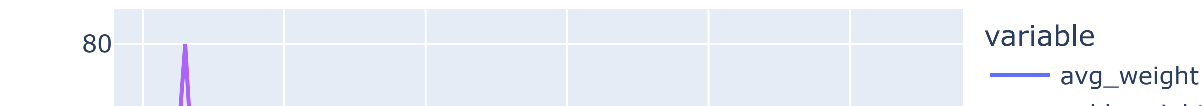

In [57]:
```
mergehw.plot(x = mergehw.year, y= [mergehw.avg_height, mergehw.gold_height, mergehw.s
            title= 'Average Height Over Time by Medal Winner')
```

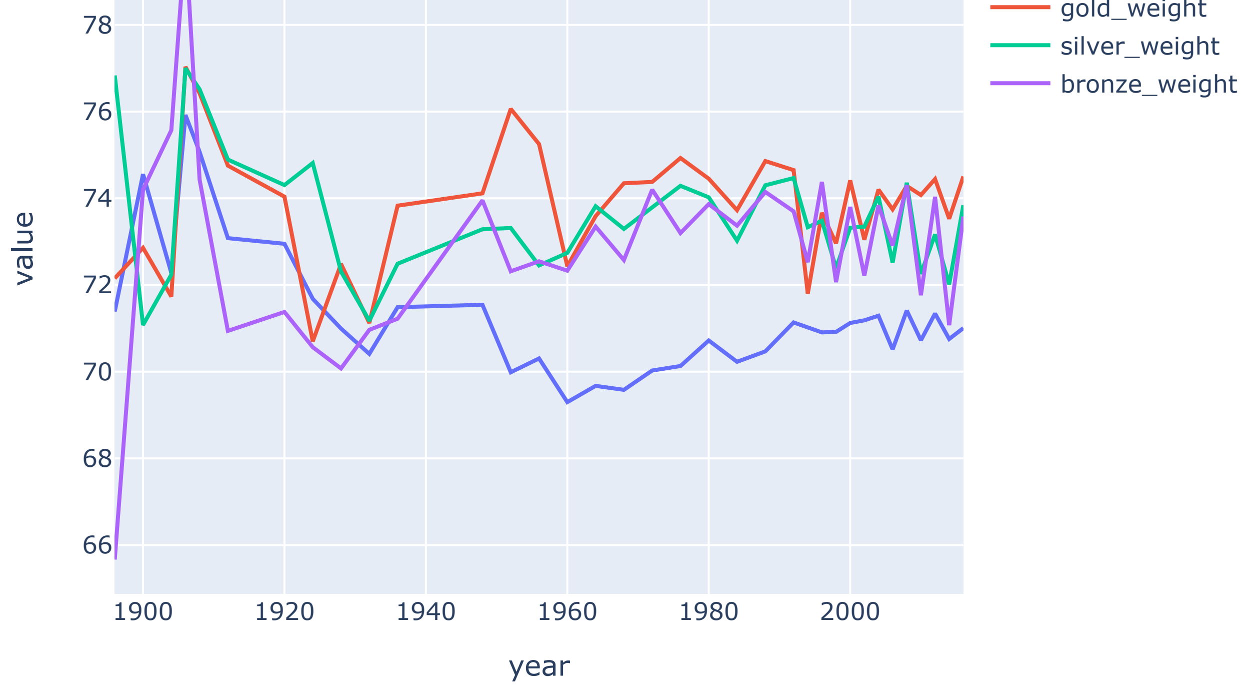
Average Height Over Time by Medal Winner




In [58]:
```
mergehw.plot(x = mergehw.year, y= [mergehw.avg_weight, mergehw.gold_weight, mergehw.s
            title= 'Average Weight Over Time by Medal Winner')
```

Average Weight Over Time by Medal Winner






It does appear after looking at many variations of trend lines for both height and weight over time and medal winners whether it be gold silver or bronze. That having a higher Heights and larger weight definitely leads to more success in winning a medal. Of course I'm sure there is outliers if I was to break this down by event I'm sure gymnasts probably need to be on the smaller side while weightlifters and those kids. And field competitions benefit from a height and weight.

# Conclusion

After going through this analysis I was able to both prove my hypothesis true. The last two about weight and height we're quite accurate and even the one regarding the season and which part of the world would win primarily the the North in the Winter. Also per capita smaller countires do tend to when more medals than larger ones like China, USA, Etc.

Honestly could probably go through a million other questions I have at the moment but I will save that for later like breaking it down amongst events in who excels in which events weather certain countries excel at certain events we're certain demographics to. There is an endless amounts of analysis to be found in this data.

I look forward to exploring it some more.