In [2]:
```
import pandas as pd
import seaborn as sns
import plotly.express as px
import math
import numpy as np
import datetime as dt

import matplotlib.pyplot as plt
import plotly.io as pio
pio.renderers.default = "plotly_mimetype+notebook"
```

In [3]:
```
# In case you use VS code and face an SSL certificate error, uncomment the following code and run the cell.

# import ssl
# ssl._create_default_https_context = ssl._create_unverified_context
```

Matplotlib

For this exercise, the following code has been provided to load the stock dataset.

In [4]:
```
stocks = px.data.stocks()
print(f'Data frame shape: {stocks.shape}')
stocks.head()
```

```
Data frame shape: (105, 7)
```

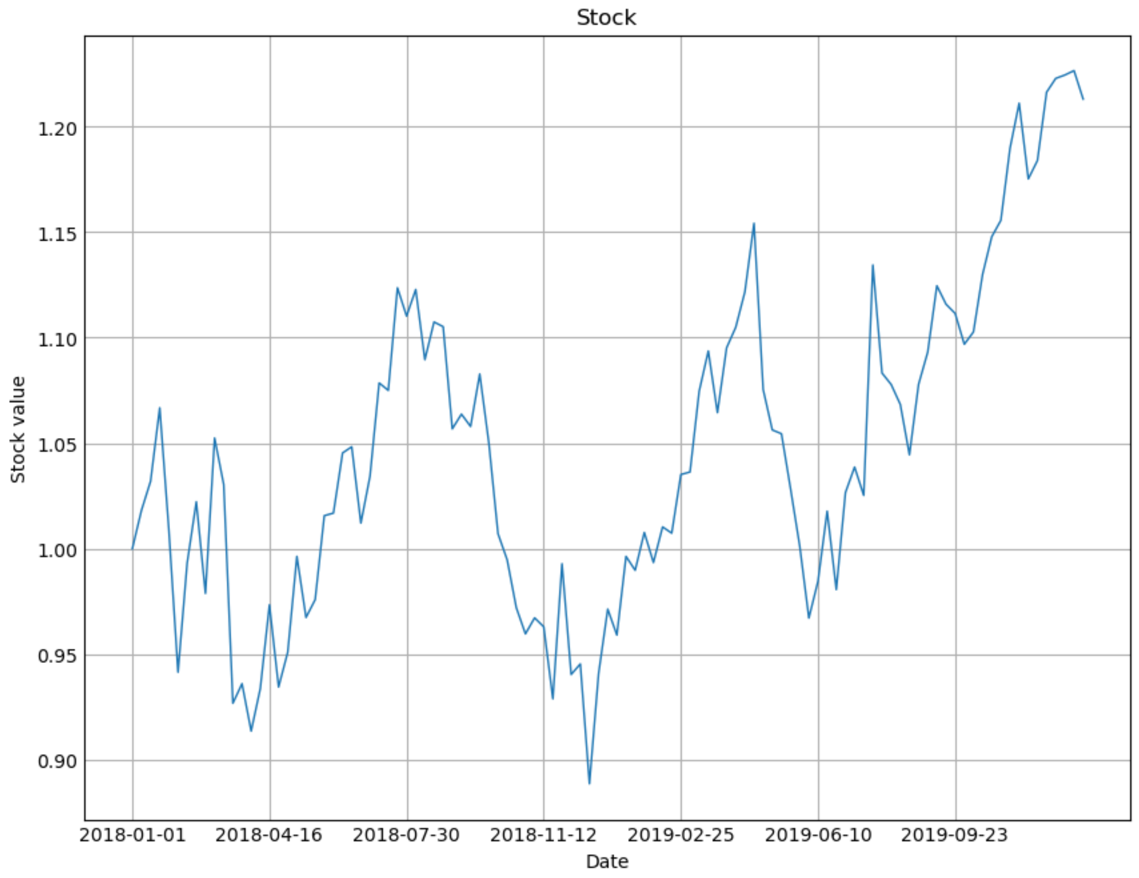
Out[4]:

	date	GOOG	AAPL	AMZN	FB	NFLX	MSFT
0	2018-01-01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	2018-01-08	1.018172	1.011943	1.061881	0.959968	1.053526	1.015988
2	2018-01-15	1.032008	1.019771	1.053240	0.970243	1.049860	1.020524
3	2018-01-22	1.066783	0.980057	1.140676	1.016858	1.307681	1.066561
4	2018-01-29	1.008773	0.917143	1.163374	1.018357	1.273537	1.040708

Question 1:

Select a specific stock from the dataset and create a well-constructed plot. Make sure the plot is readable with relevant information, such as date, values. Additionally, provide a title for the plot and appropriately label the axes

In [5]:
```
# YOUR CODE HERE

plt.style.use('_mpl-gallery')

# make data
x = [i for i in stocks['date']]
y = stocks['GOOG']
labels = stocks['date']

# plot
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(x, y, linewidth=1)

ax.set_title('Stock')
ax.set_ylabel('Stock value')
ax.set_xlabel('Date')
ax.set(xticks= x[::15], autoscaley_on=True)
plt.show()
```



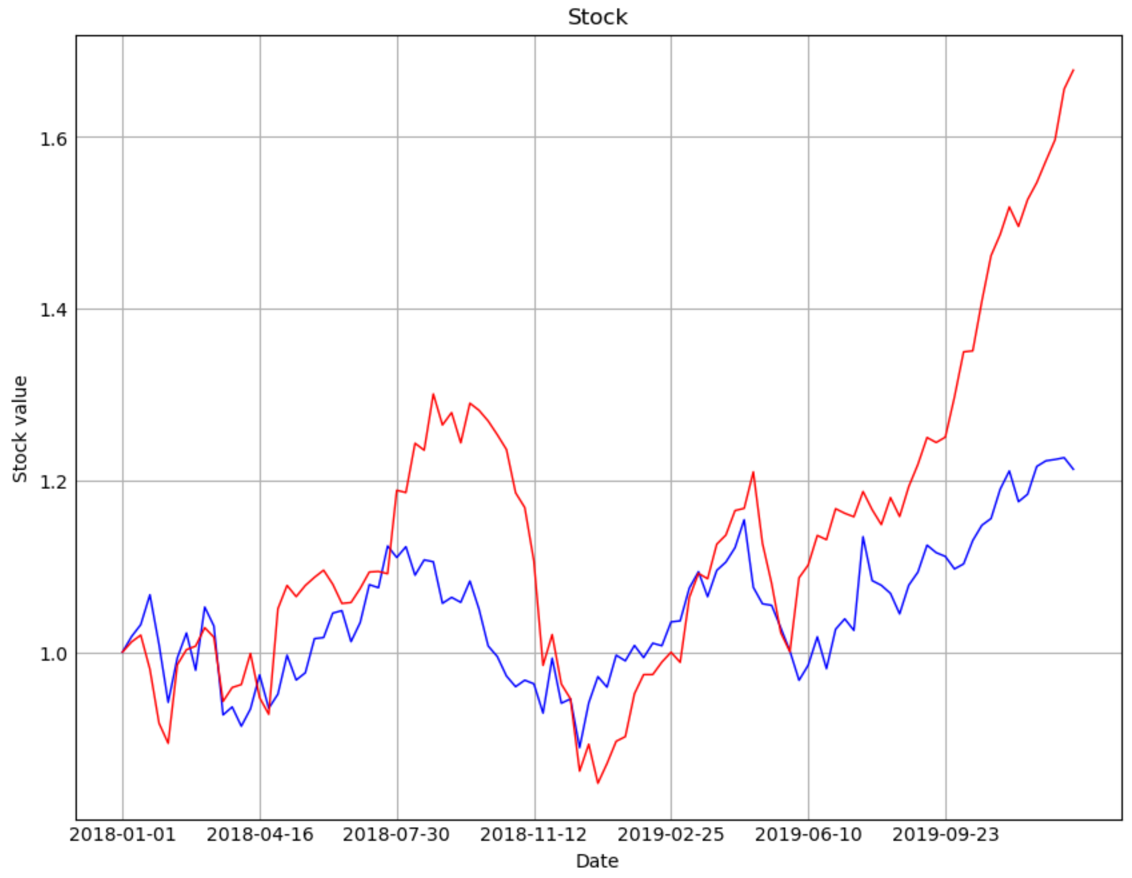
In [6]:
```
# Do not run this cell.
```

Question 2

You have already plotted data from one stock. It is possible to plot multiple stocks for comparison.
Customise line styles, markers, colours, and include a legend in the plot to differentiate between them.

In [7]:
```
# YOUR CODE HERE

plt.style.use('_mpl-gallery')

# make data
x = [i for i in stocks['date']]
y1 = stocks['GOOG']
y2 = stocks['AAPL']
labels = stocks['date']

# plot
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(x, y1, linewidth=1, color = 'blue')
```

```
# plot
ax.plot(x, y2, linewidth=1, color='red')

ax.set_title('Stock')
ax.set_ylabel('Stock value')
ax.set_xlabel('Date')
ax.set(xticks= x[::15], autoscaley_on=True)
plt.show()
```



In [8]:
```
# Do not run this cell.
```

Seaborn

For this exercise, the following code has been provided to load the tips dataset

In [9]:
```
tips = sns.load_dataset('tips')
print(f'Data frame shape: {stocks.shape}')
tips.head()
```

```
Data frame shape: (105, 7)
```

Out[9]:

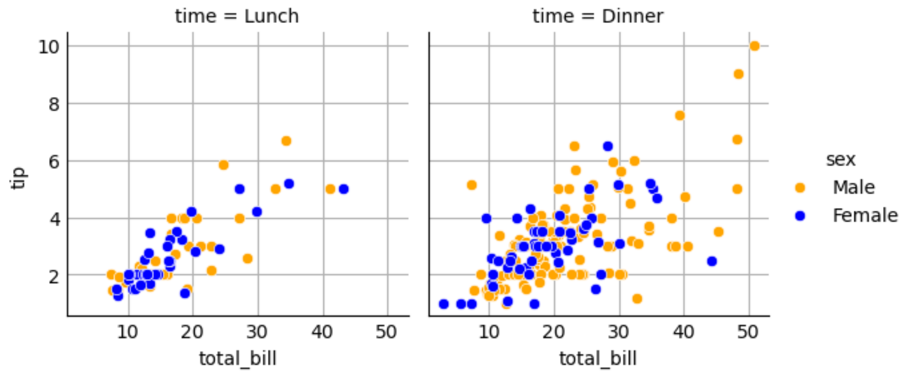
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Question 3:

- Is there a significant difference in tipping behavior between male and female patrons who are smokers and non-smokers?

In [10]:

```
# Your code here (recreate the following plot)
# make data:
# plt.style.use('_mpl-gallery')

# smokers = tips[tips['smoker'] == 'Yes']
# non_smokers = tips[tips['smoker'] == 'No']

# # plot
# x = smokers['tip']
# y = smokers['total_bill']
# col = ['blue' if i == 'Male' else 'orange' for i in smokers['sex']]

# plt.scatter(x, y, c=col, alpha=1)
# plt.show()

g = sns.FacetGrid(tips, col='time', hue='sex', palette=['orange', 'blue'])
g.map(sns.scatterplot, 'total_bill', 'tip')
g.add_legend()
plt.show()
```



In [11]:

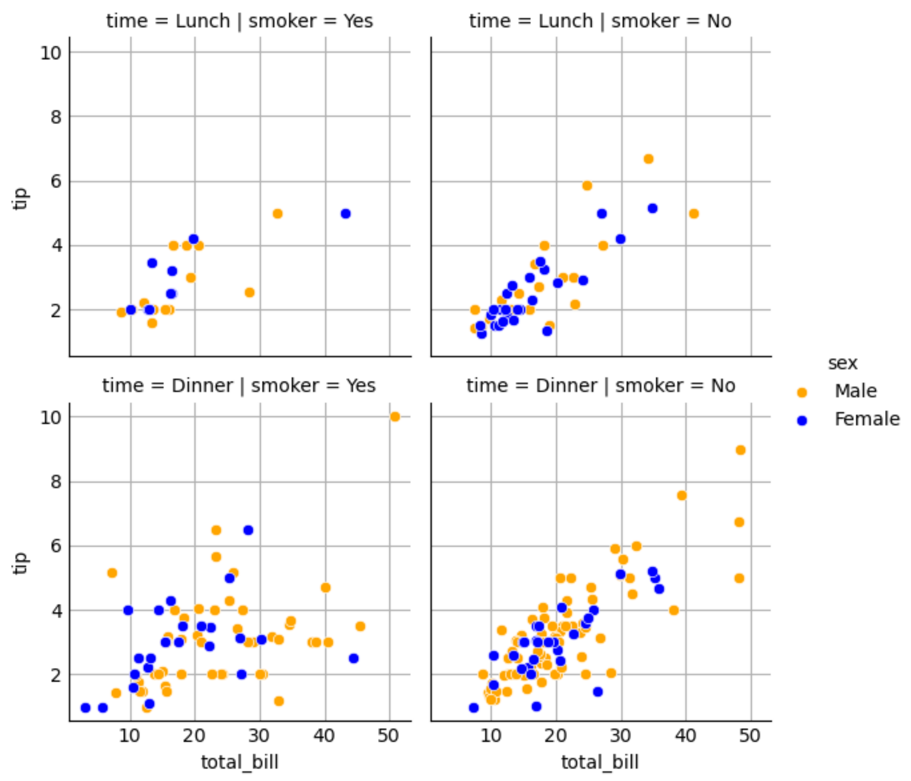
```
# Do not run this cell.
```

- Create a visualization to explore how the 'total_bill' and 'tip' vary among different groups, considering factors like gender, smoking habits, and meal type."

In [12]:

```
# Your code here (recreate the following plot)
smokers = tips[tips['smoker'] == 'Yes']
non_smokers = tips[tips['smoker'] == 'No']
```

```
g = sns.FacetGrid(tips, col='smoker', row='time', hue='sex', palette=['orange', 'blue'])
g.map(sns.scatterplot, 'total_bill', 'tip')
g.add_legend()
plt.show()
```



In [13]:
```
# Do not run this cell.
```

Plotly Express

Question 4:

Redo the above exercises (Questions 2 & 3) with plotly express. Create diagrams which you can interact with.

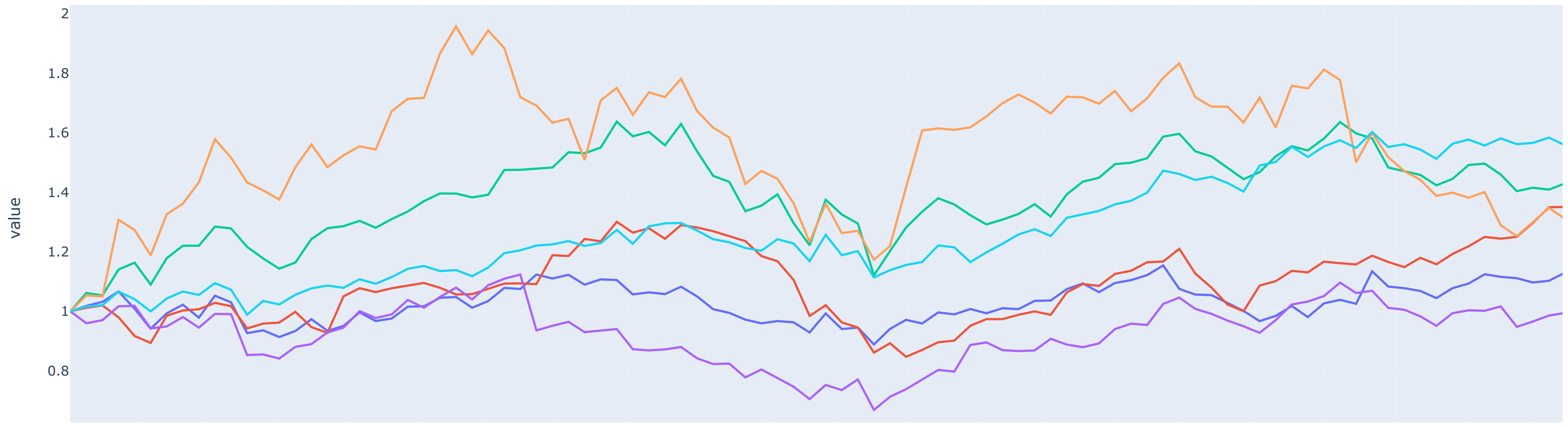
The stocks dataset

Hints:

- Restructure the stocks dataframe in a format that can be readily utilized with Plotly Express.

In [14]:
```
# YOUR CODE HERE (Convert data to the following format)
df = px.data.stocks()
```

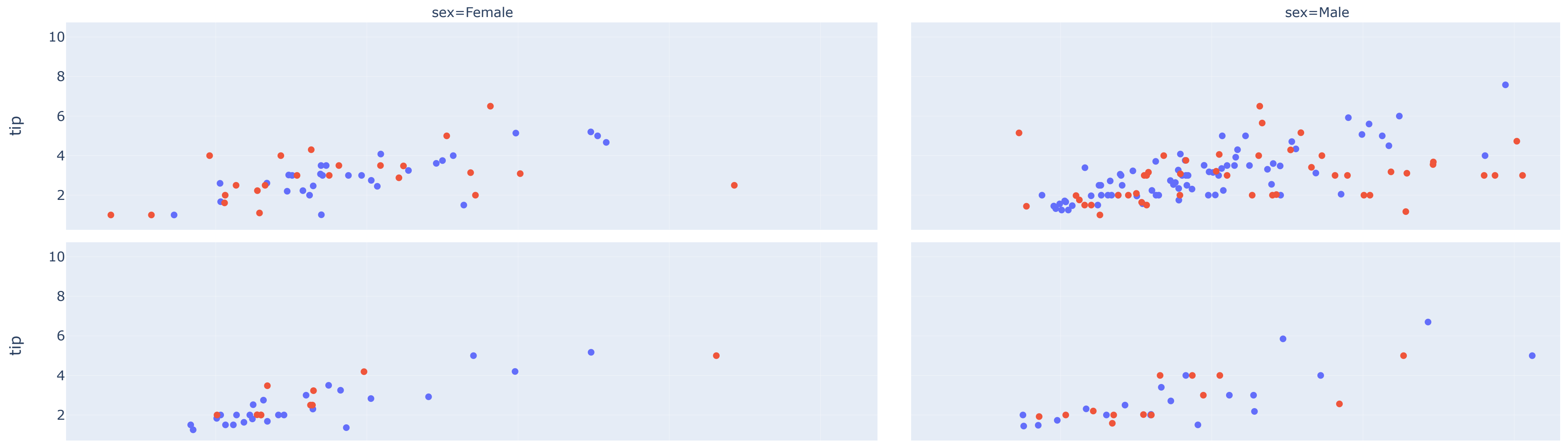
```
fig = px.line(df, x="date", y=df.columns,
              hover_data={"date": "|%B %d, %Y"},
              )
fig.update_xaxes(
    dtick="M1",
    tickformat="%b\n%Y",
    ticklabelmode="period")
fig.show()
```



In [15]:
```
# Do not run this cell.
```

The tips dataset

In [16]:
```
# YOUR CODE HERE
df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", color="smoker", facet_col="sex", facet_row="time")
fig.show()
```



In [17]:
```
# Do not run this cell.
```

Question 5:

Recreate the barplot below that shows the population of different continents for the year 2007.

Hints:

1. **Data Extraction:** Begin by extracting the data for the year 2007 from the DataFrame. Be sure to process the data appropriately.
2. **Plotly Bar Chart:** For visualization, use Plotly Express to create a bar chart. You can find the relevant documentation here.
3. **Color Coding:** Apply different colors to the bars representing each continent.
4. **Custom Sorting:** Arrange the bars on the x-axis in a specific order based on continents. This can be achieved by configuring the x-axis layout settings, which are detailed here.
5. **Bar Labels:** Add text labels to each bar to display population information clearly.

In [18]:
```
#load data
df = px.data.gapminder()
print(f'Data frame shape: {stocks.shape}')
df.head()
```

```
Data frame shape: (105, 7)
```

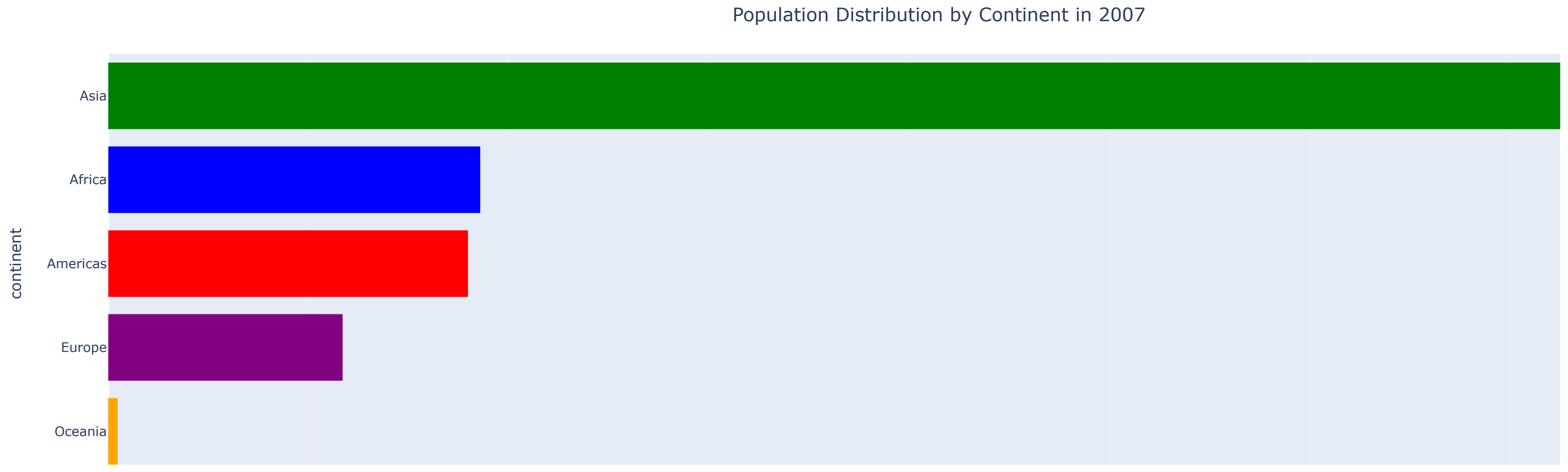
Out[18]:

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

In [19]:

```python
df = px.data.gapminder()
df_2007 = df[df['year']==2007]
df_2007_new = df_2007.groupby('continent').sum()

# Sort the DataFrame by population in descending order
df_2007_new = df_2007_new.sort_values(by='pop', ascending=False)

# Define a color scale based on the sorted order
# color_scale = px.colors.qualitative.Plotly[::-1]
color_scale = ['green', 'blue', 'red', 'purple', 'orange']

fig = px.bar(df_2007_new, x="pop", y=df_2007_new.index, orientation='h',
             color=df_2007_new.index, color_discrete_sequence=color_scale)

fig.update_layout(title={
   'text': 'Population Distribution by Continent in 2007',
   'x': 0.5})

fig.update_xaxes(autorange=True)

fig.show()
```

```
C:\Users\jelme\AppData\Local\Temp\ipykernel_1488\3084227494.py:3: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be vali
d for the function.
```



In [20]:
```
# Do not run this cell.
```

In [ ]: