In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline

#Plotly Express is the easy-to-use, high-level interface to Plotly, which operates on a variet
import plotly.express as px

# Functions in plotly.graph_objs module generate graph objects
import plotly.graph_objects as go
```

EXERCISE 1

Create line and bar plotly charts for the IRIS dataset available in plotly.

- Load the dataset and check the data.
- Both line and bar charts should attempt to display the sepal length in function of petal length for the 3 iris species in the dataset.

In [2]:
```
df_iris = px.data.iris()
df_iris.head()

fig_line = px.line(
    df_iris,
    x="petal_length",
    y="sepal_length",
    color="species",
    title="Line Plot: Sepal Length vs Petal Length by Iris Species"
)
fig_line.show()


df_avg = df_iris.groupby(["species", "petal_length"]).agg({"sepal_length": "mean"}).reset_ind

fig_bar = px.bar(
    df_avg,
    x="petal_length",
    y="sepal_length",
    color="species",
    barmode="group",
    title="Bar Chart: Average Sepal Length vs Petal Length by Species"
)
fig_bar.show()
```

EXERCISE 2

Create various scatter plots charts for the TIPS dataset available in plotly.

- Load the dataset and check the data.
- Explore the dataset using scatter plots.
- Try all of the above exemplified features.

In [3]:
```
df_tips = px.data.tips()
print(df_tips.head())

fig_1 = px.scatter(
    df_tips,
    x='total_bill',
    y='tip',
```

```
    title='Tip for Total bill'
)
fig_1.show()

fig_2 = px.scatter(
    df_tips,
    x='total_bill',
    y='tip',
    color='day',
    title='Tip for Total bill per day'
)
fig_2.show()

fig_3 = px.scatter(
    df_tips,
    x='total_bill',
    y='tip',
    color='time',
    size='size',
    hover_data='tip'
)
fig_3.show()

fig_4 = px.scatter(
    df_tips,
    x='total_bill',
    y='tip',
    color='day',
    size='size',
    facet_col='sex'
)
fig_4.show()


fig_5 = px.scatter(
    df_tips,
    x='total_bill',
    y='tip',
    color='sex',
    size='size',
    log_x=True,
    animation_frame='sex'
)
fig_5.show()
```

```
   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
```

EXERCISE 3

Create various animated plots to visualize the GAPMINDER dataset available in plotly.

- Load the dataset and check the data.
- Explore the dataset using bar charts, scatter plots, animated maps, etc.
- Try to produce the most useful visualizations for this dataset (for example, check how the life exp depends on gdp per capita; or how the population is increasing each year per continent).

In [4]:
```
df_gapminder = px.data.gapminder()
print(df_gapminder.head())
```

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

   iso_num
0        4
1        4
2        4
3        4
4        4
```

In [5]:
```
latest_year = df_gapminder['year'].max()
df_latest = df_gapminder[df_gapminder['year'] == latest_year]

fig_1 = px.pie(
    df_latest,
    names="continent",
    values="pop",
    title=f"Population Distribution Across Continents in {latest_year}"
)
fig_1.show()
```

In [6]:
```
fig_2 = px.histogram(
    df_latest,
    x="gdpPercap",
    nbins=30,
    title=f"Distribution of GDP per Capita in {latest_year}",
    labels={"gdpPercap": "GDP per Capita"}
)
fig_2.show()
```

In [7]:
```
fig_3 = px.box(
    df_latest,
    x="continent",
    y="lifeExp",
    points="all",
    title="Box Plot: Life Expectancy across Continents",
    hover_data='country'
)
fig_3.show()
```

In [8]:
```
fig_4 = px.violin(
    df_latest,
    x="continent",
    y="gdpPercap",
    box=True,
    points="all",
    title="Violin Plot: GDP per capita distribution across continents",
    labels={"gdpPercap": "GDP per Capita"},
    hover_data='country'
)
fig_4.show()
```

In [9]:
```
fig_5 = px.scatter_3d(
    df_gapminder,
    x="gdpPercap",
    y="lifeExp",
    z="pop",
    color="continent",
    symbol="continent",
```

```
    animation_frame="year",
    log_x=True,
    title="3D Scatter Plot: GDP per Capita vs Life Expectancy vs Population Over Time",
    labels={"gdpPercap": "GDP per Capita", "lifeExp": "Life Expectancy", "pop": "Population"}
)
fig_5.show()
```

In [10]:
```
# Animated Scatter Plot: Life Expectancy vs GDP per capita over time
fig_6 = px.scatter(
    df_gapminder,
    x="gdpPercap",
    y="lifeExp",
    animation_frame="year",  # Animate over the years
    color="continent",
    hover_name="country",
    log_x=True,  # Log scale for GDP per capita
    size="pop",
    title="Animated Scatter Plot: Life Expectancy vs GDP per capita",
    labels={"gdpPercap": "GDP per Capita", "lifeExp": "Life Expectancy"}
)
fig_6.show()
```

In [11]:
```
# Corrected Animated Bar Chart: Population of the top 10 most populated countries over time
# Select the top 10 most populated countries by each year
df_sorted = df_gapminder.groupby('year').apply(lambda x: x.nlargest(10, 'pop')).reset_index(d

fig_7 = px.bar(
    df_sorted,
    x="country",
    y="pop",
    animation_frame="year",
    color="continent",
    title="Animated Bar Chart: Population of the Top 10 Most Populated Countries Over Time",
    labels={"pop": "Population"}
)
fig_7.show()
```

```
C:\Users\astal\AppData\Local\Temp\ipykernel_5788\2952095325.py:3: DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a
future version of pandas the grouping columns will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select the grouping columns afte
r groupby to silence this warning.
```

In [12]:
```
fig_8 = px.choropleth(
    df_gapminder,
    locations="country",
    locationmode="country names",
    color="pop",
    animation_frame="year",
    title="Animated Choropleth Map: Population Growth Over Time",
    color_continuous_scale="Viridis",
    hover_name="country"
)

fig_8.show()
```