Set up the notebook

In [1]:
```
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

```
Setup Complete
```

Load the data

parse_dates = True - This tells the notebook to understand the each row label as a date (as opposed to a number or other text with a different meaning).

In [2]:
```
# Path of the file to read
fifa_filepath = r'D:\Data Analytics\Python\Kaggle Data Visualization\fifa.csv'

# Read the file into a variable fifa_data
fifa_data = pd.read_csv(fifa_filepath, index_col="Date", parse_dates=True)
```

Examine the data

In [3]:
```
# Print the first 5 rows of the data
fifa_data.head()
```

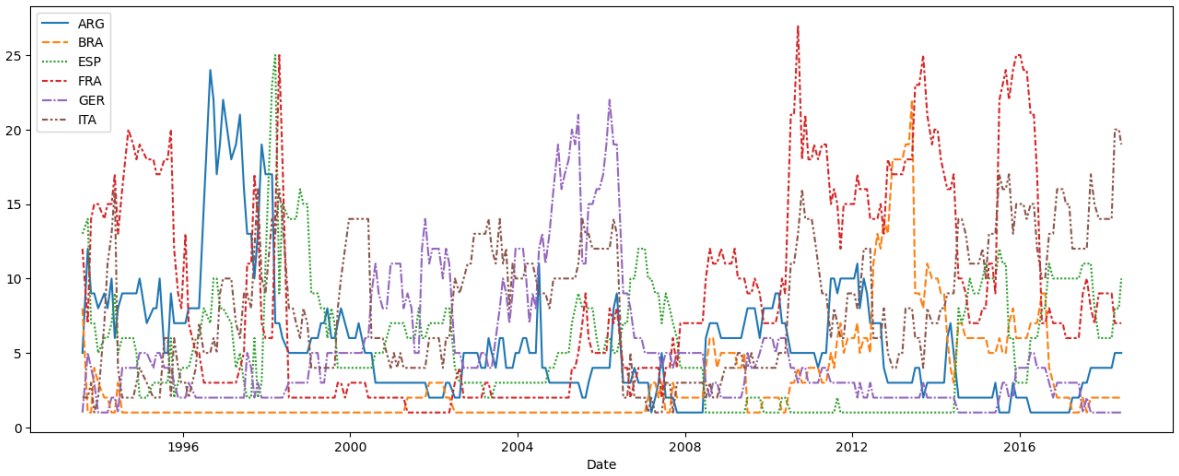
Out[3]:

Date	ARG	BRA	ESP	FRA	GER	ITA
1993-08-08	5.0	8.0	13.0	12.0	1.0	2.0
1993-09-23	12.0	1.0	14.0	7.0	5.0	2.0
1993-10-22	9.0	1.0	7.0	14.0	4.0	3.0
1993-11-19	9.0	4.0	7.0	15.0	3.0	1.0
1993-12-23	8.0	3.0	5.0	15.0	1.0	2.0

Plot the data

In [4]:
```
# Set the width and height of the figure
plt.figure(figsize=(16,6))

# Line chart showing how FIFA rankings evolved over time
sns.lineplot(data = fifa_data)
```

Out[4]:
```
<Axes: xlabel='Date'>
```



Line Charts

In [5]:
```
# Path of the file to read
spotify_filepath = r'D:\Data Analytics\Python\Kaggle Data Visualization\spotif

# Read the file into a variable spotify_data
spotify_data = pd.read_csv(spotify_filepath, index_col="Date", parse_dates=Tru
```

Examine the data

In [6]:
```
# Print the first 5 rows of the data
spotify_data.head()
```

Out[6]:

Date	Shape of You	Despacito	Something Just Like This	HUMBLE.	Unforgettable
2017-01-06	12287078	NaN	NaN	NaN	NaN
2017-01-07	13190270	NaN	NaN	NaN	NaN
2017-01-08	13099919	NaN	NaN	NaN	NaN
2017-01-09	14506351	NaN	NaN	NaN	NaN
2017-01-10	14275628	NaN	NaN	NaN	NaN

In [7]:
```
# Print the last five rows of the data
spotify_data.tail()
```

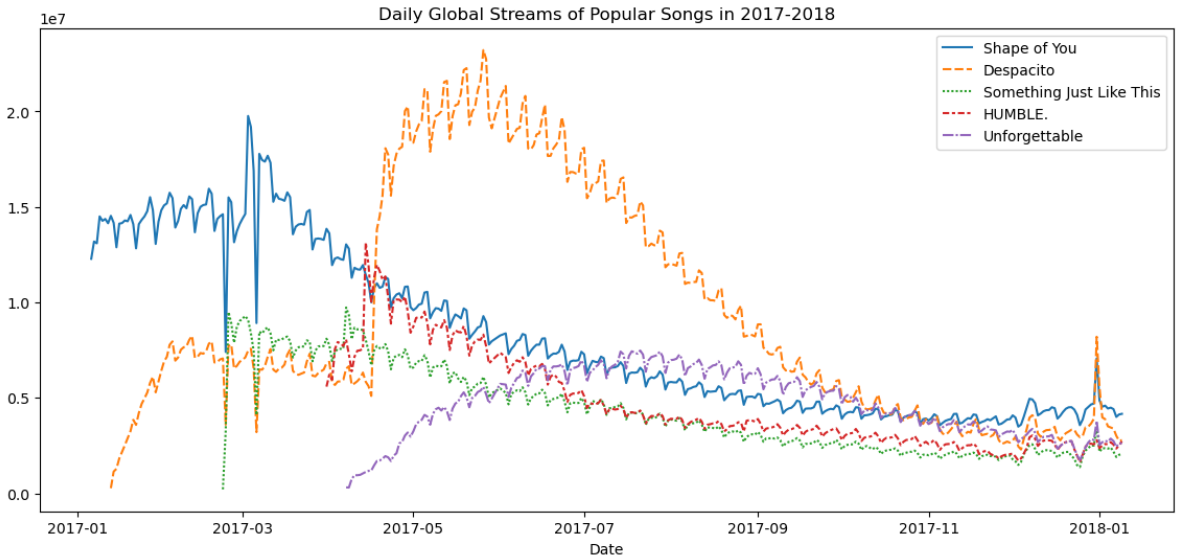
Out[7]:

Date	Shape of You	Despacito	Something Just Like This	HUMBLE.	Unforgettable
2018-01-05	4492978	3450315.0	2408365.0	2685857.0	2869783.0
2018-01-06	4416476	3394284.0	2188035.0	2559044.0	2743748.0
2018-01-07	4009104	3020789.0	1908129.0	2350985.0	2441045.0
2018-01-08	4135505	2755266.0	2023251.0	2523265.0	2622693.0
2018-01-09	4168506	2791601.0	2058016.0	2727678.0	2627334.0

Plot the data

In [8]:
```
# Set the width and height of the figure
plt.figure(figsize=(14,6))

# Add title
plt.title("Daily Global Streams of Popular Songs in 2017-2018")

# Line chart showing daily global streams of each song
sns.lineplot(data=spotify_data)
```

Out[8]:
```
<Axes: title={'center': 'Daily Global Streams of Popular Songs in 2017-201
8'}, xlabel='Date'>
```



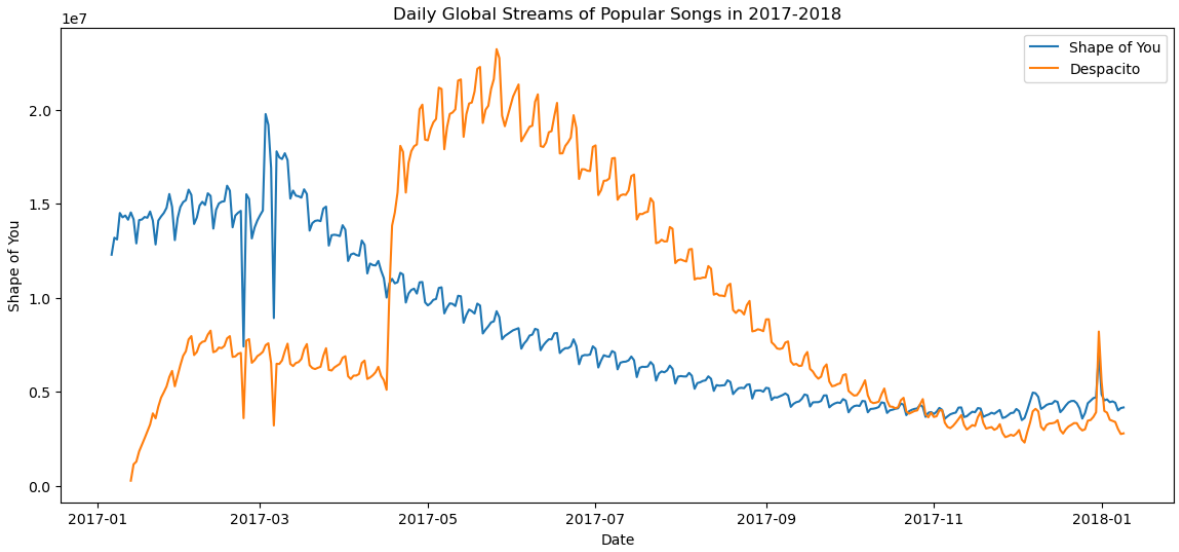
Plot a subset of the data

In [9]:
```
list(spotify_data.columns)
```

Out[9]:
```
['Shape of You',
 'Despacito',
 'Something Just Like This',
 'HUMBLE.',
 'Unforgettable']
```

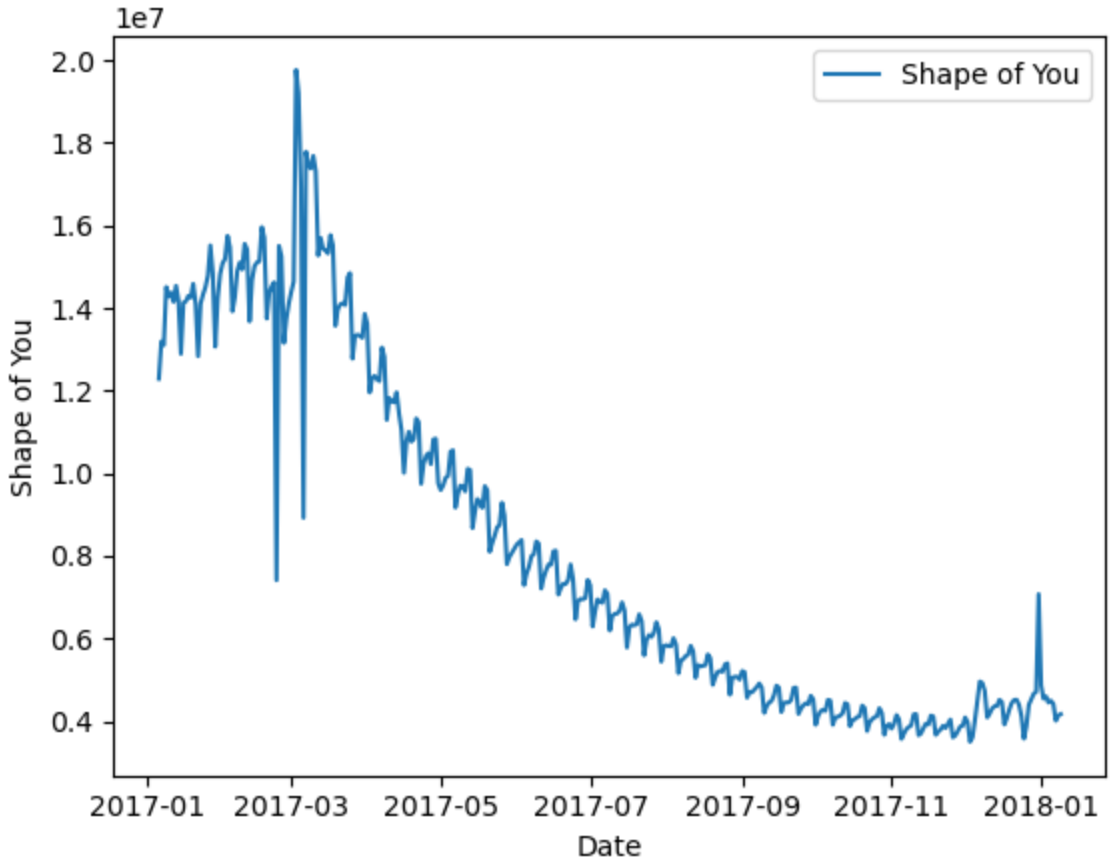
In [10]:
```
# Set the width and height of the figure
plt.figure(figsize=(14,6))

# Add title
plt.title("Daily Global Streams of Popular Songs in 2017-2018")

# Line chart showing daily global streams of 'Shape of You'
sns.lineplot(data=spotify_data['Shape of You'], label="Shape of You")

# Line chart showing daily global streams of 'Despacito'
sns.lineplot(data=spotify_data['Despacito'], label="Despacito")

# Add label for horizontal axis
plt.xlabel("Date")
```

Out[10]:
```
Text(0.5, 0, 'Date')
```



In [11]:
```
# Line chart showing daily global streams of 'Shape of You'
sns.lineplot(data=spotify_data['Shape of You'], label="Shape of You")
```

Out[11]: <Axes: xlabel='Date', ylabel='Shape of You'>



Exercise

In [12]:
```
# How many listeners did listen Shape of You in 2018-01-06?

spotify_data['Shape of You'].loc['2018-01-06']
```

Out[12]: 4416476

In [13]:
```
# In 2018-01-06, how many more listeners did listen Shape of You than Somethin

spotify_data['Shape of You'].loc['2018-01-06'] - spotify_data['Something Just
```

Out[13]: 2228441.0

Bar Charts and Heatmaps

In [14]:
```
# Path of the file to read
flight_filepath = r'D:\Data Analytics\Python\Kaggle Data Visualization\flight_

# Read the file into a variable flight_data
flight_data = pd.read_csv(flight_filepath, index_col="Month")
```

Examine the data

In [15]:
```
# Print the data
flight_data
```

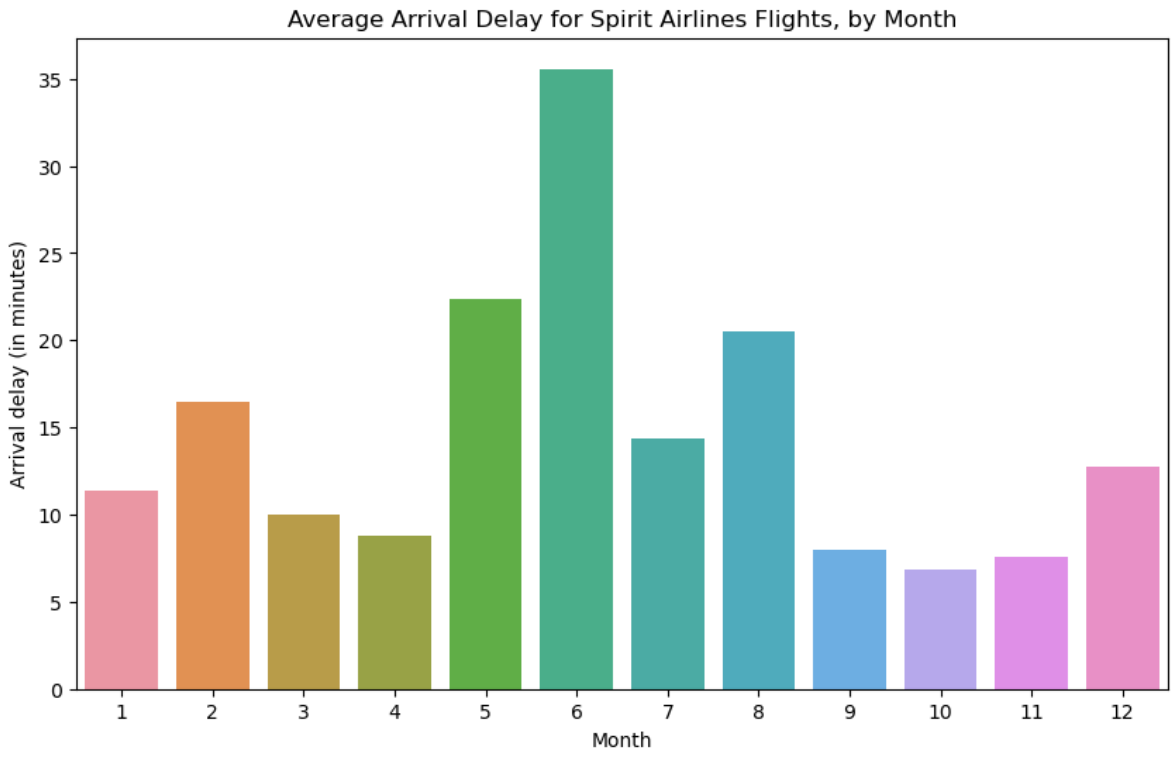
Out[15]:

Month	AA	AS	B6	DL	EV	F9	HA	MQ	
1	6.955843	-0.320888	7.347281	-2.043847	8.537497	18.357238	3.512640	18.164974	1
2	7.530204	-0.782923	18.657673	5.614745	10.417236	27.424179	6.029967	21.301627	1
3	6.693587	-0.544731	10.741317	2.077965	6.730101	20.074855	3.468383	11.018418	1
4	4.931778	-3.009003	2.780105	0.083343	4.821253	12.640440	0.011022	5.131228	
5	5.173878	-1.716398	-0.709019	0.149333	7.724290	13.007554	0.826426	5.466790	2
6	8.191017	-0.220621	5.047155	4.419594	13.952793	19.712951	0.882786	9.639323	3
7	3.870440	0.377408	5.841454	1.204862	6.926421	14.464543	2.001586	3.980289	1
8	3.193907	2.503899	9.280950	0.653114	5.154422	9.175737	7.448029	1.896565	2
9	-1.432732	-1.813800	3.539154	-3.703377	0.851062	0.978460	3.696915	-2.167268	
10	-0.580930	-2.993617	3.676787	-5.011516	2.303760	0.082127	0.467074	-3.735054	
11	0.772630	-1.916516	1.418299	-3.175414	4.415930	11.164527	-2.719894	0.220061	
12	4.149684	-1.846681	13.839290	2.504595	6.685176	9.346221	-1.706475	0.662486	1

Bar chart

In [16]:
```
# Set the width and height of the figure
plt.figure(figsize=(10,6))

# Add title
plt.title("Average Arrival Delay for Spirit Airlines Flights, by Month")

# Bar chart showing average arrival delay for Spirit Airlines flights by month
sns.barplot(x=flight_data.index, y=flight_data['NK'])

# Add label for vertical axis
plt.ylabel("Arrival delay (in minutes)")
```

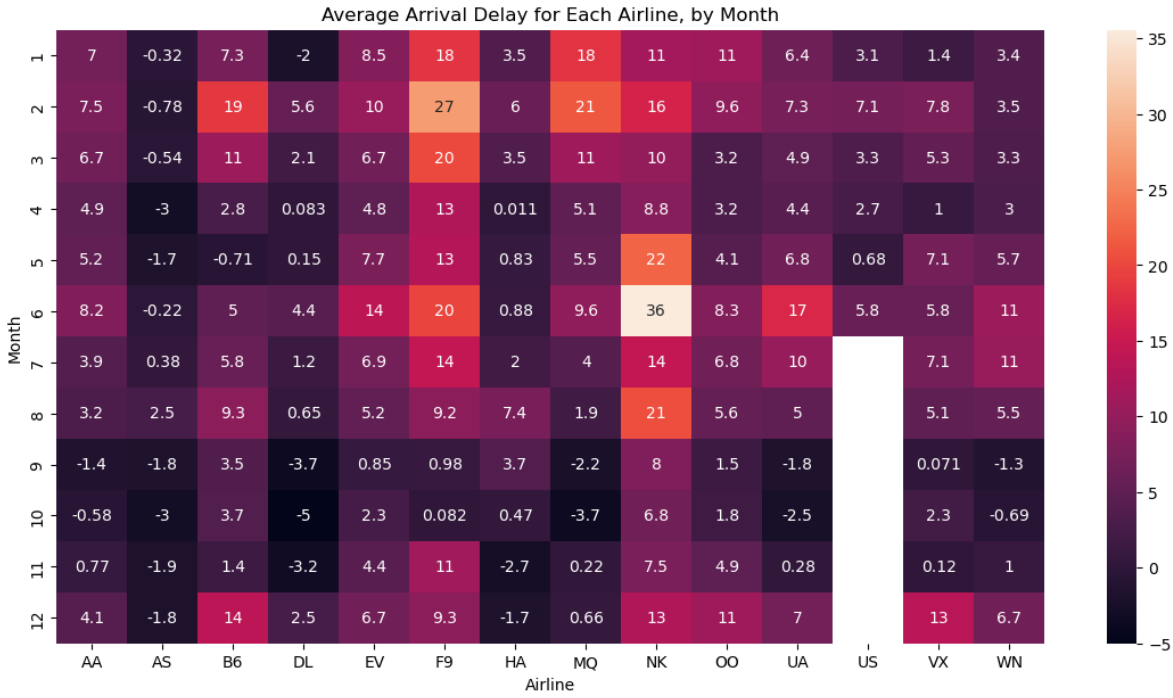
Out[16]:
```
Text(0, 0.5, 'Arrival delay (in minutes)')
```



Heatmap

In [17]:
```
# Set the width and height of the figure
plt.figure(figsize=(14,7))

# Add title
plt.title("Average Arrival Delay for Each Airline, by Month")

# Heatmap showing average arrival delay for each airline by month
sns.heatmap(data=flight_data, annot=True)

# Add label for horizontal axis
plt.xlabel("Airline")
```

Out[17]:
```
Text(0.5, 47.7222222222222, 'Airline')
```



Exercise

In [18]:
```
ign_data = pd.read_csv(r'D:\Data Analytics\Python\Kaggle Data Visualization\ig

# Print the data
ign_data
```

Out[18]:

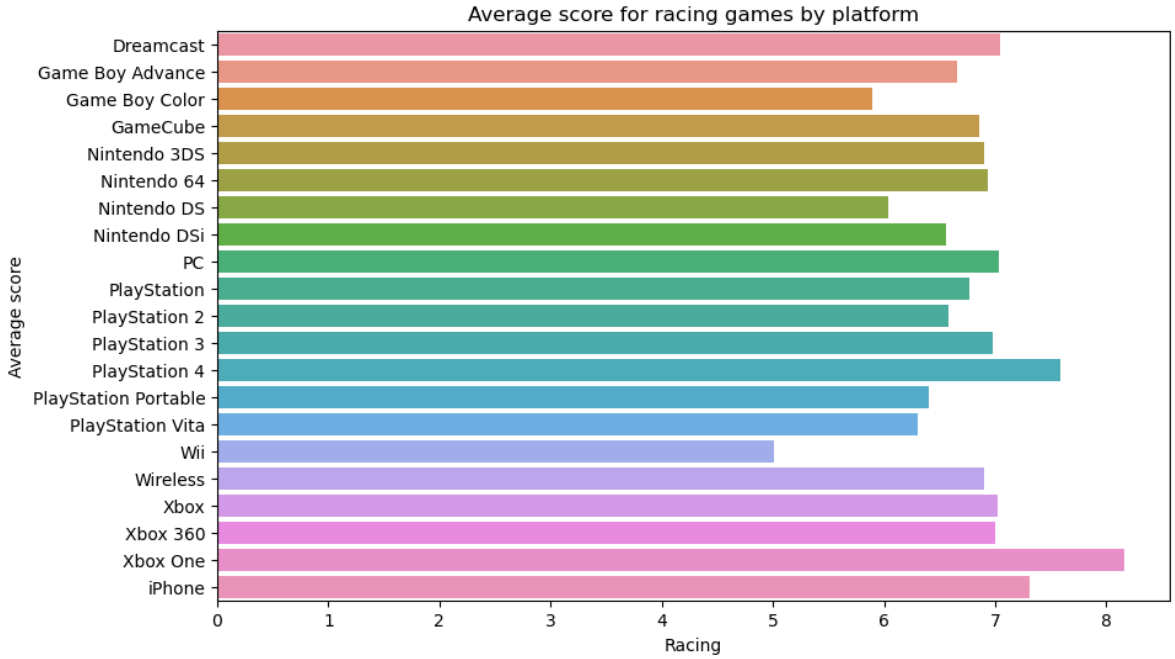
Platform	Action	Action, Adventure	Adventure	Fighting	Platformer	Puzzle	RPG	Racing
Dreamcast	6.882857	7.511111	6.281818	8.200000	8.340000	8.088889	7.700000	7.042500
Game Boy Advance	6.373077	7.507692	6.057143	6.226316	6.970588	6.532143	7.542857	6.657143
Game Boy Color	6.272727	8.166667	5.307692	4.500000	6.352941	6.583333	7.285714	5.897436
GameCube	6.532584	7.608333	6.753846	7.422222	6.665714	6.133333	7.890909	6.852632
Nintendo 3DS	6.670833	7.481818	7.414286	6.614286	7.503448	8.000000	7.719231	6.900000
Nintendo 64	6.649057	8.250000	7.000000	5.681250	6.889655	7.461538	6.050000	6.939623
Nintendo DS	5.903608	7.240000	6.259804	6.320000	6.840000	6.604615	7.222619	6.038636
Nintendo DSi	6.827027	8.500000	6.090909	7.500000	7.250000	6.810526	7.166667	6.563636
PC	6.805791	7.334746	7.136798	7.166667	7.410938	6.924706	7.759930	7.032418
PlayStation	6.016406	7.933333	6.313725	6.553731	6.579070	6.757895	7.910000	6.773387
PlayStation 2	6.467361	7.250000	6.315152	7.306349	7.068421	6.354545	7.473077	6.585065
PlayStation 3	6.853819	7.306154	6.820988	7.710938	7.735714	7.350000	7.436111	6.978571
PlayStation 4	7.550000	7.835294	7.388571	7.280000	8.390909	7.400000	7.944000	7.590000
PlayStation Portable	6.467797	7.000000	6.938095	6.822222	7.194737	6.726667	6.817778	6.401961
PlayStation Vita	7.173077	6.133333	8.057143	7.527273	8.568750	8.250000	7.337500	6.300000
Wii	6.262718	7.294643	6.234043	6.733333	7.054255	6.426984	7.410345	5.011667
Wireless	7.041699	7.312500	6.972414	6.740000	7.509091	7.360550	8.260000	6.898305
Xbox	6.819512	7.479032	6.821429	7.029630	7.303448	5.125000	8.277778	7.021591
Xbox 360	6.719048	7.137838	6.857353	7.552239	7.559574	7.141026	7.650000	6.996154
Xbox One	7.702857	7.566667	7.254545	7.171429	6.733333	8.100000	8.291667	8.163636
iPhone	6.865445	7.764286	7.745833	6.087500	7.471930	7.810784	7.185185	7.315789

In [19]:
```
# What is the highest average score received by PC games,
high_score = max(ign_data[ign_data.index== "PC"].max())
print(high_score)

# On the Playstation Vita platform, which genre has the lowest average score?
worst_genre = ign_data[ign_data.index== "PlayStation Vita"].idxmin(axis = 1)[0
print(worst_genre)
```

```
7.759930313588847
Simulation
```

In [20]:
```
# Bar chart showing average score for racing games by platform
plt.figure(figsize=(10,6))
plt.title("Average score for racing games by platform")
sns.barplot(x=ign_data['Racing'], y=ign_data.index)
plt.ylabel("Average score")
```

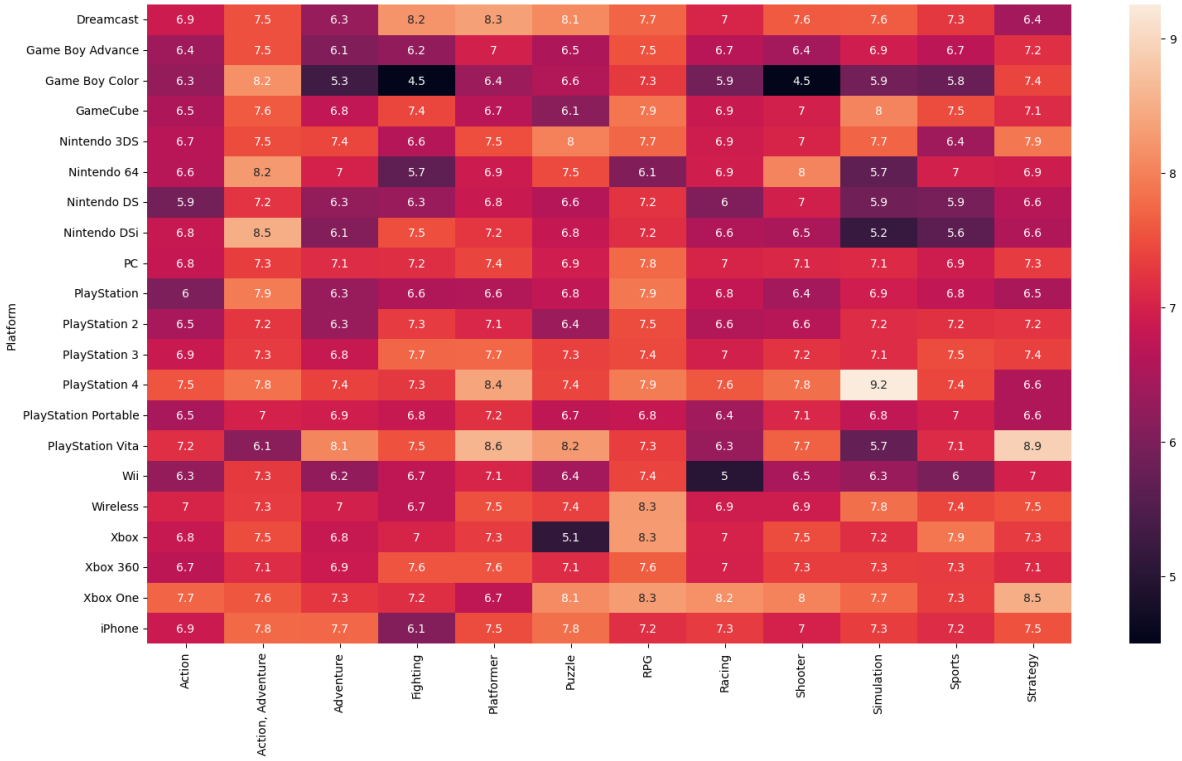
Out[20]:
```
Text(0, 0.5, 'Average score')
```



Based on the data, we should not expect a racing game for the Wii platform to receive a high rating. In fact, on average, racing games for Wii score lower than any other platform. Xbox One seems to be the best alternative, since it has the highest average ratings.

In [21]:
```
# Heatmap showing average game score by platform and genre
plt.figure(figsize = (18,10))
sns.heatmap(data=ign_data, annot=True)
```

Out[21]: <Axes: ylabel='Platform'>



Simulation games for **Playstation 4** receive the highest average ratings (9.2). **Shooting** and **Fighting** games for **Game Boy Color** receive the lowest average rankings (4.5).

Scatter Plots

In [22]:
```
# Path of the file to read
insurance_filepath = r'D:\Data Analytics\Python\Kaggle Data Visualization\insu

# Read the file into a variable insurance_data
insurance_data = pd.read_csv(insurance_filepath)
```

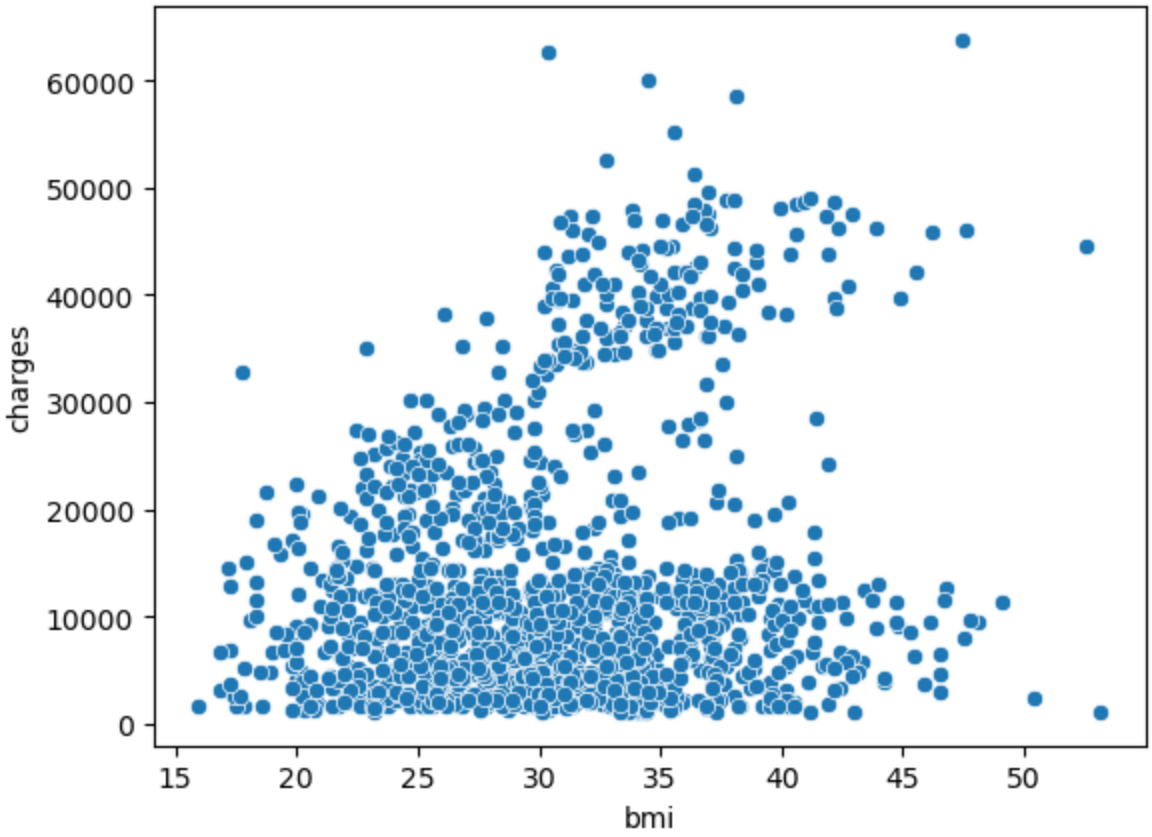
In [23]:
```
insurance_data.head()
```

Out[23]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [24]:
```
sns.scatterplot(x=insurance_data['bmi'], y=insurance_data['charges'])
```

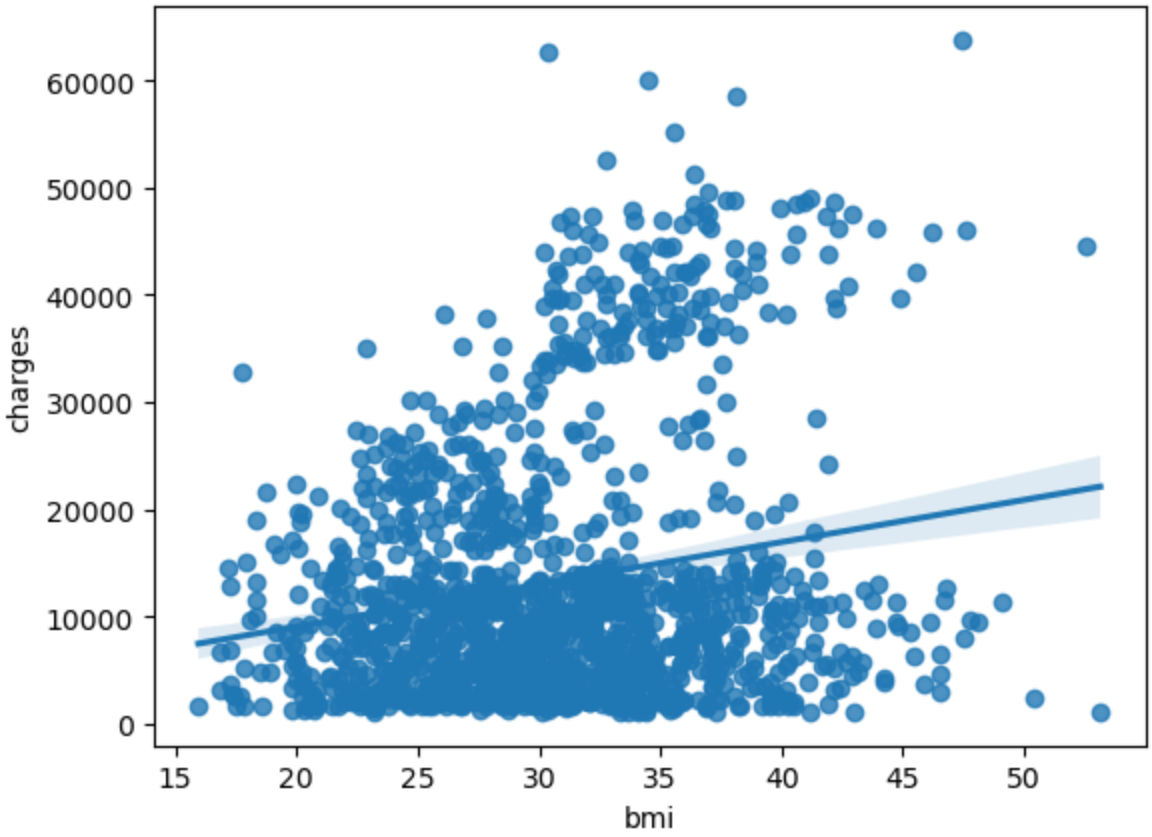
Out[24]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



The scatterplot above suggests that body mass index (BMI) and insurance costs (charges) are **positively correlated**, where customers with higher BMI typically also tend to pay more in insurance costs. (This pattern makes sense, since high BMI is typically associated with higher risk of chronic disease.)

In [25]:
```
sns.regplot(x=insurance_data['bmi'], y=insurance_data['charges'])
```

Out[25]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```

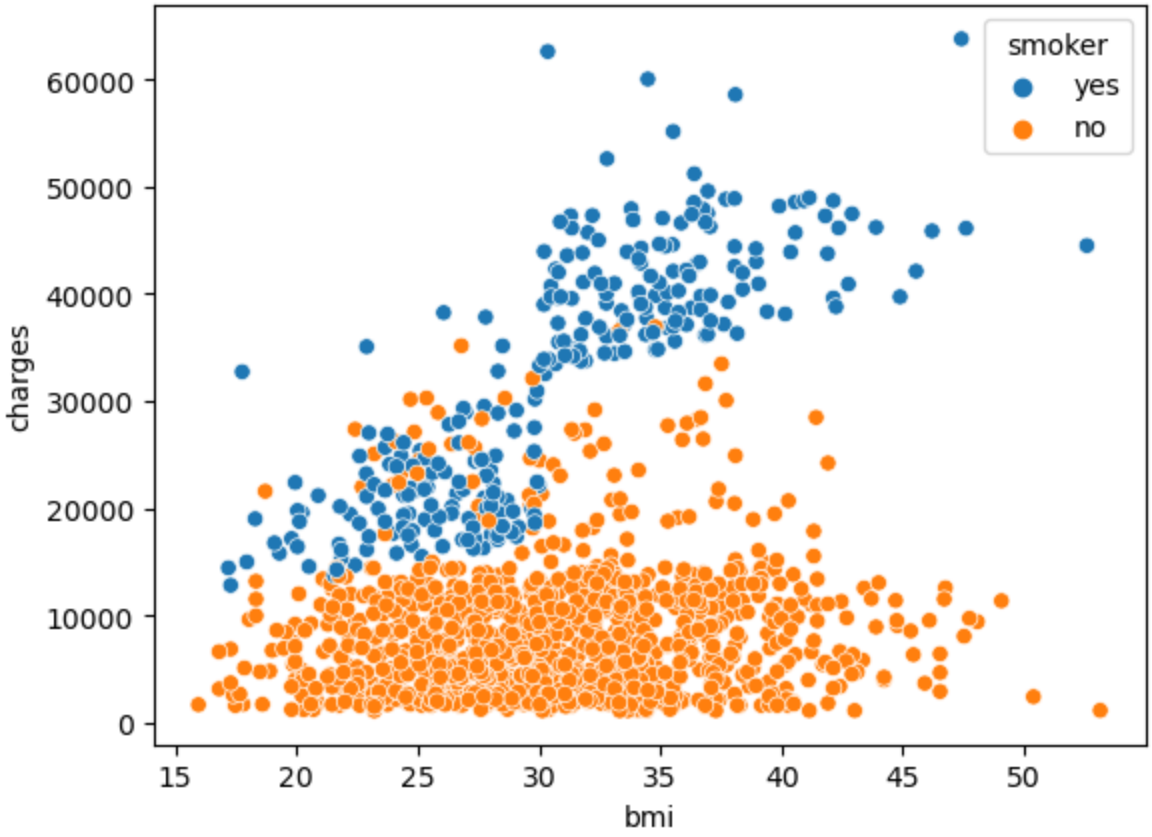


To double-check the strength of this relationship, you might like to add a **regression line**, or the line that best fits the data. We do this by changing the command to sns.regplot.

Color-coded scatter plots

In [26]:
```
sns.scatterplot(x=insurance_data['bmi'], y=insurance_data['charges'], hue = in
```

Out[26]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```

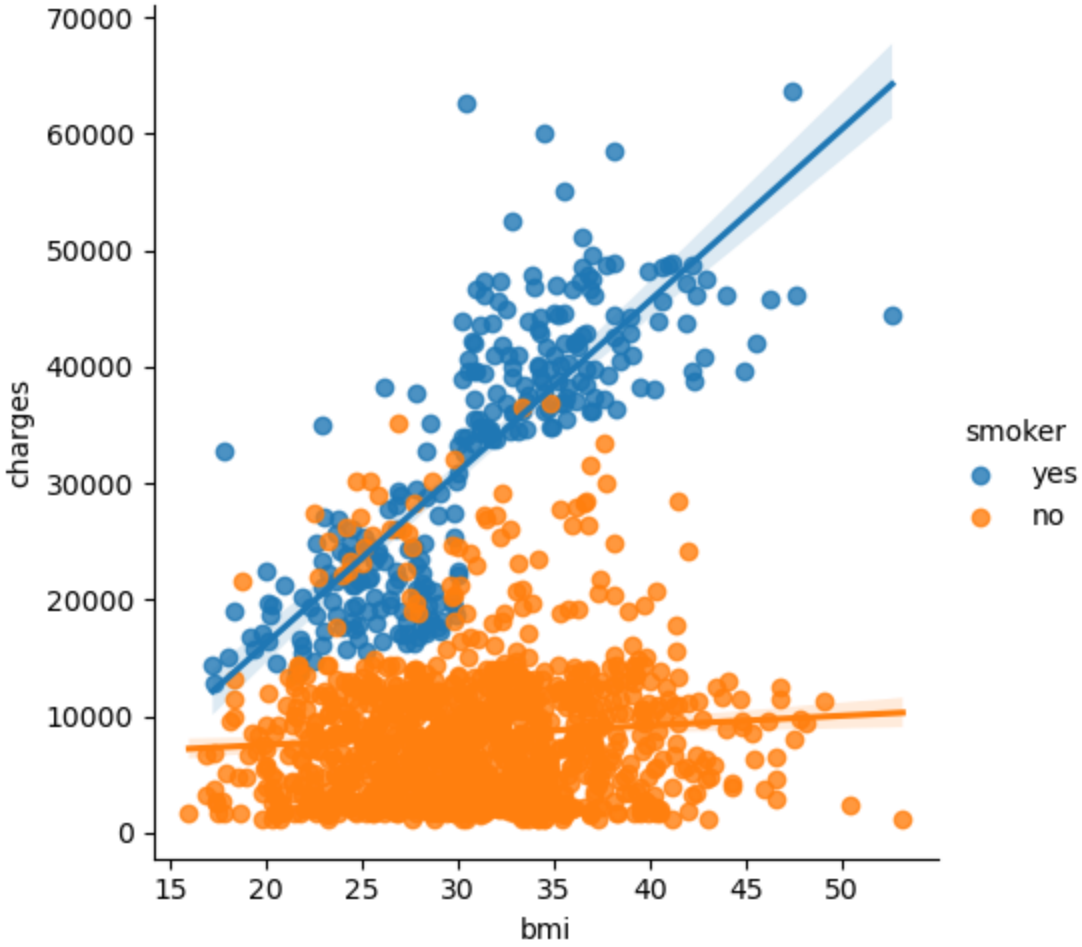


This scatter plot shows that while nonsmokers to tend to pay slightly more with increasing BMI, smokers pay MUCH more.

In [27]:
```
sns.lmplot(x="bmi", y="charges", hue="smoker", data=insurance_data)
```

```
D:\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figu
re layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[27]:
```
<seaborn.axisgrid.FacetGrid at 0x2530b31f510>
```



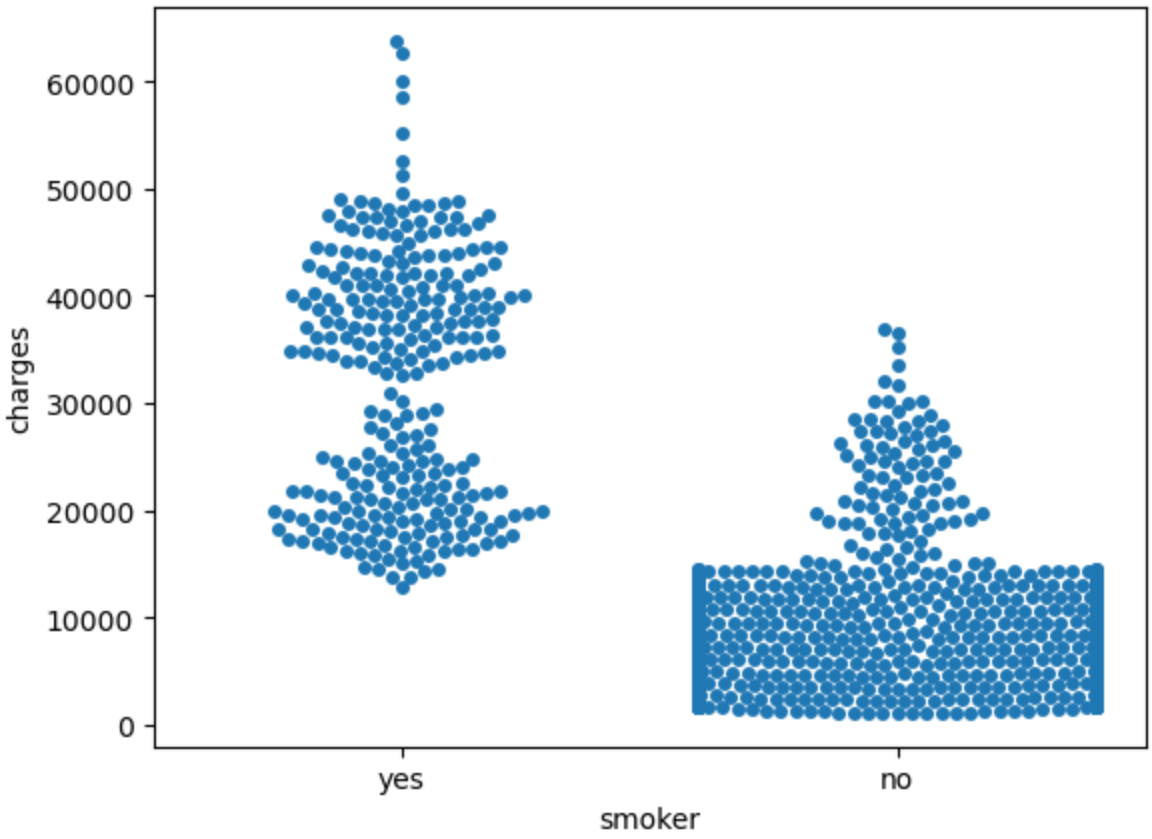
To further emphasize this fact, we can use the sns.lmplot command to add two regression lines, corresponding to smokers and nonsmokers. (*You'll notice that the regression line for smokers has a much steeper slope, relative to the line for nonsmokers!*)

In [28]:
```
sns.swarmplot(x=insurance_data['smoker'],
              y=insurance_data['charges'])
```

```
D:\anaconda3\Lib\site-packages\seaborn\categorical.py:3544: UserWarning: 37.
4% of the points cannot be placed; you may want to decrease the size of the m
arkers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[28]:
```
<Axes: xlabel='smoker', ylabel='charges'>
```

```
D:\anaconda3\Lib\site-packages\seaborn\categorical.py:3544: UserWarning: 60.
8% of the points cannot be placed; you may want to decrease the size of the m
arkers or use stripplot.
  warnings.warn(msg, UserWarning)
```



We can adapt the design of the scatter plot to feature a categorical variable (like "smoker") on one of the main axes. We'll refer to this plot type as a **categorical scatter plot**, and we build it with the sns.swarmplot command.

Distributions

In [29]:
```
# Path of the file to read
iris_filepath = r'D:\Data Analytics\Python\Kaggle Data Visualization\iris.csv'

# Read the file into a variable iris_data
iris_data = pd.read_csv(iris_filepath, index_col="Id")

# Print the first 5 rows of the data
iris_data.head()
```

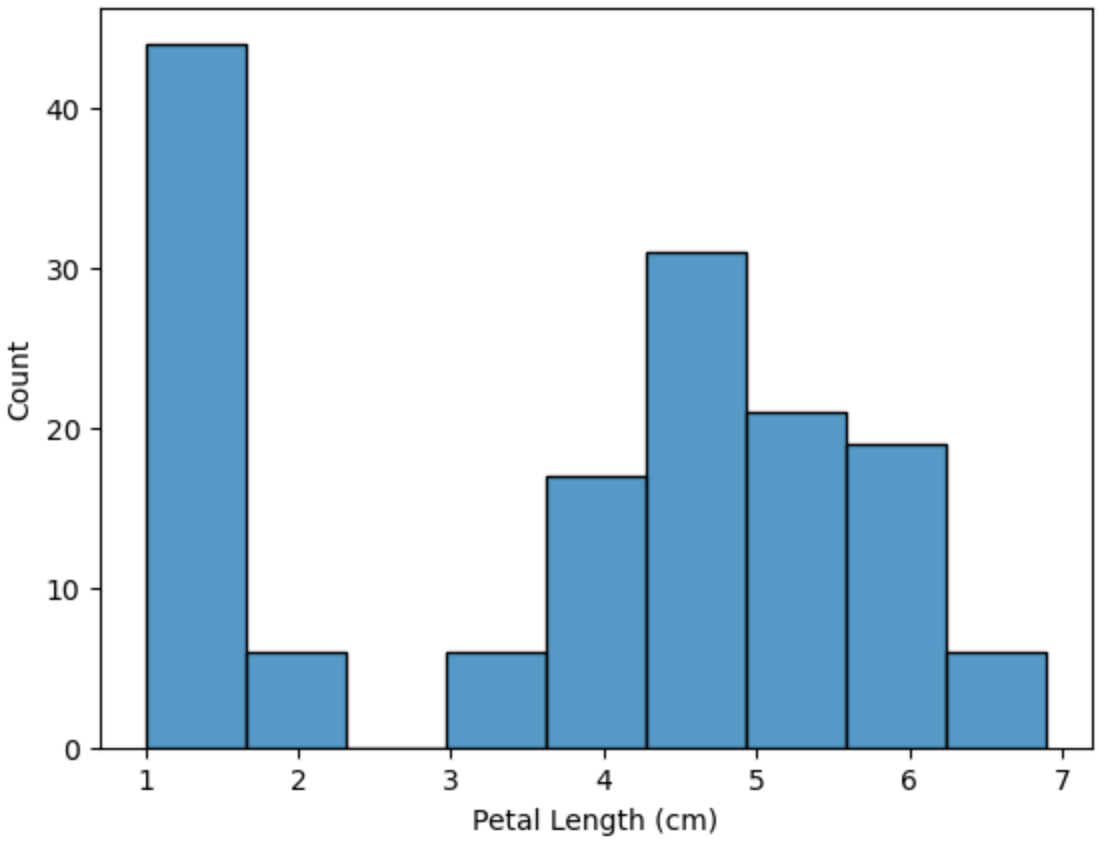
Out[29]:

Id	Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

Histograms

In [30]:
```
# Histogram
sns.histplot(iris_data['Petal Length (cm)'])
```

Out[30]:
```
<Axes: xlabel='Petal Length (cm)', ylabel='Count'>
```

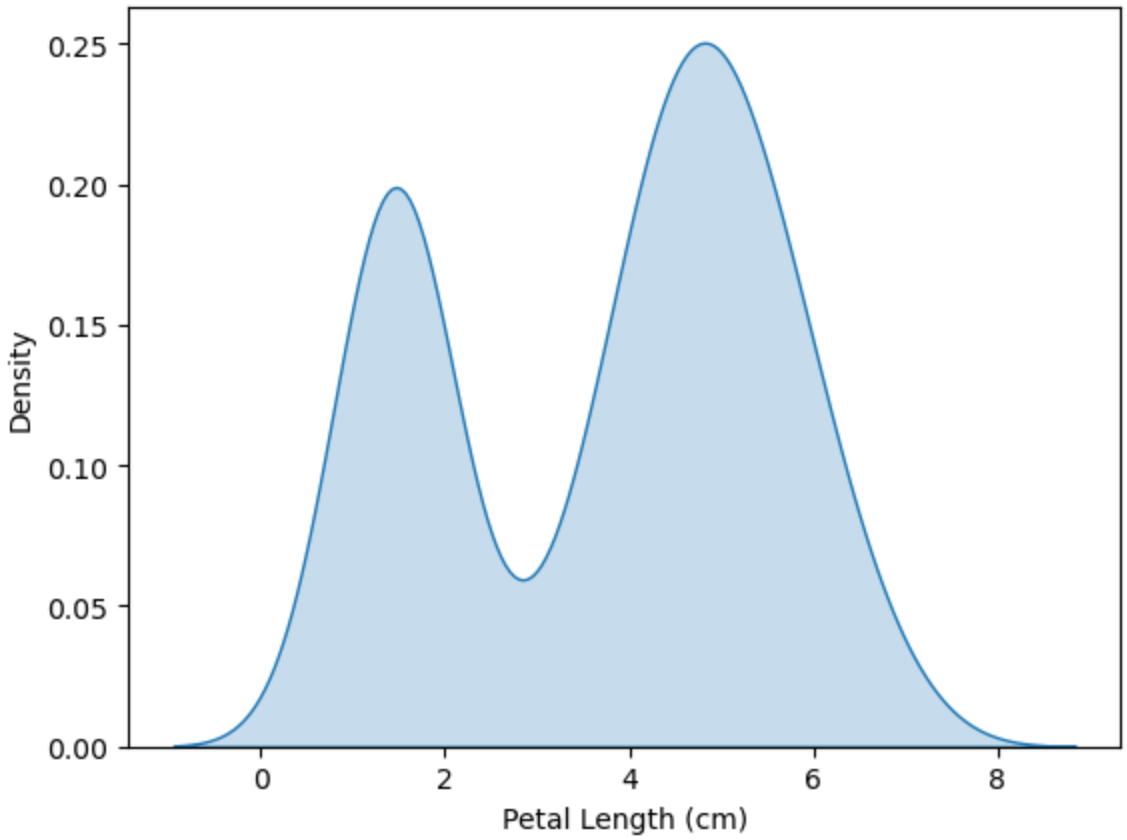


Density plots

Kernel density estimate (KDE) plot

In [31]:
```
# KDE plot
sns.kdeplot(data=iris_data['Petal Length (cm)'], fill=True)
```

Out[31]:
```
<Axes: xlabel='Petal Length (cm)', ylabel='Density'>
```

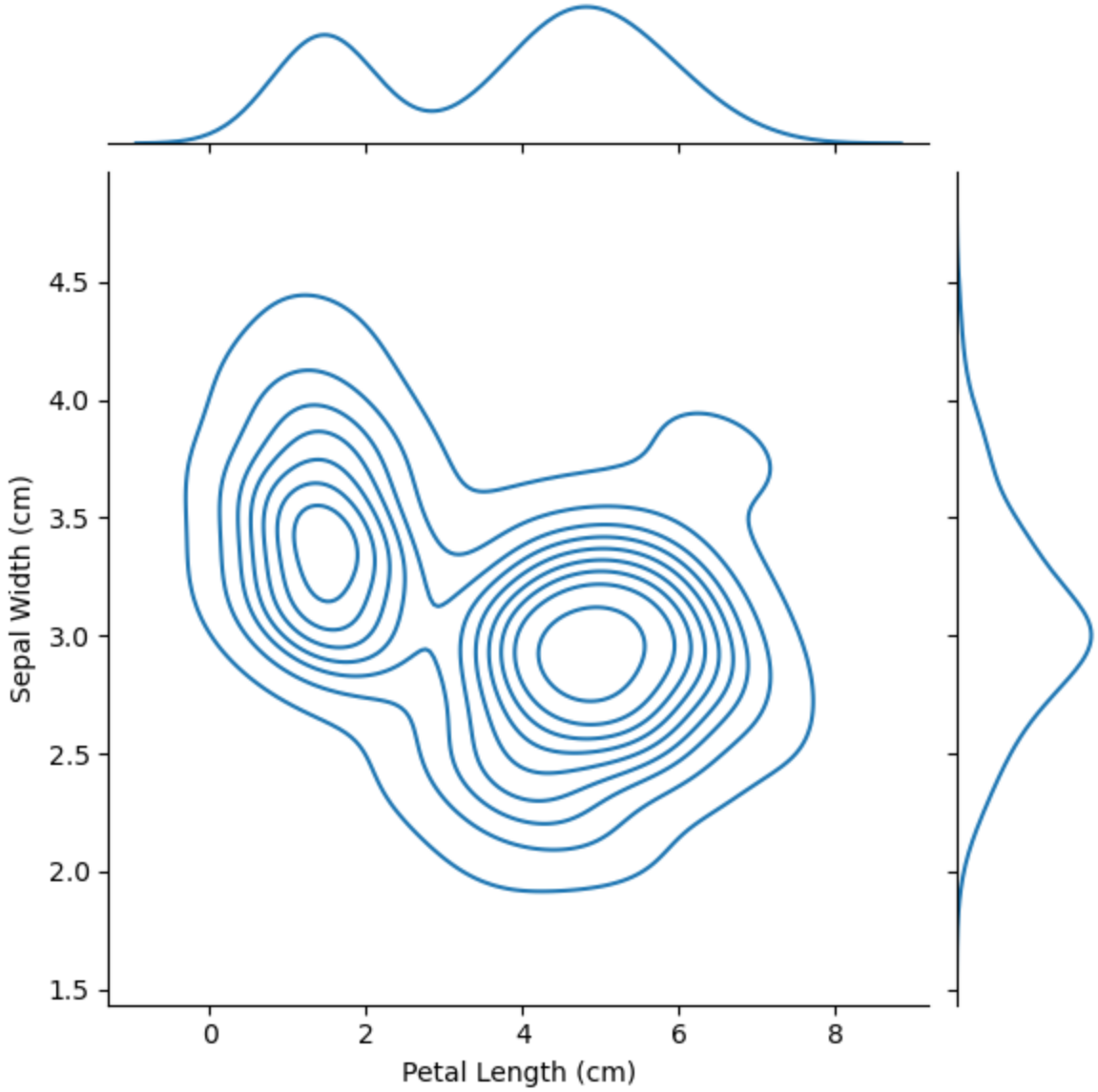


2D KDE plots

Two-dimensional (2D) KDE plot

In [32]:
```
# 2D KDE plot
sns.jointplot(x=iris_data['Petal Length (cm)'], y=iris_data['Sepal Width (cm)'
```

Out[32]:
```
<seaborn.axisgrid.JointGrid at 0x2530bdbe490>
```



Note that in addition to the 2D KDE plot in the center,

- The curve at the top of the figure is a KDE plot for the data on the x-axis (in this case, iris_data['Petal Length (cm)']), and
- The curve on the right of the figure is a KDE plot for the data on the y-axis (in this case, iris_data['Sepal Width (cm)']).

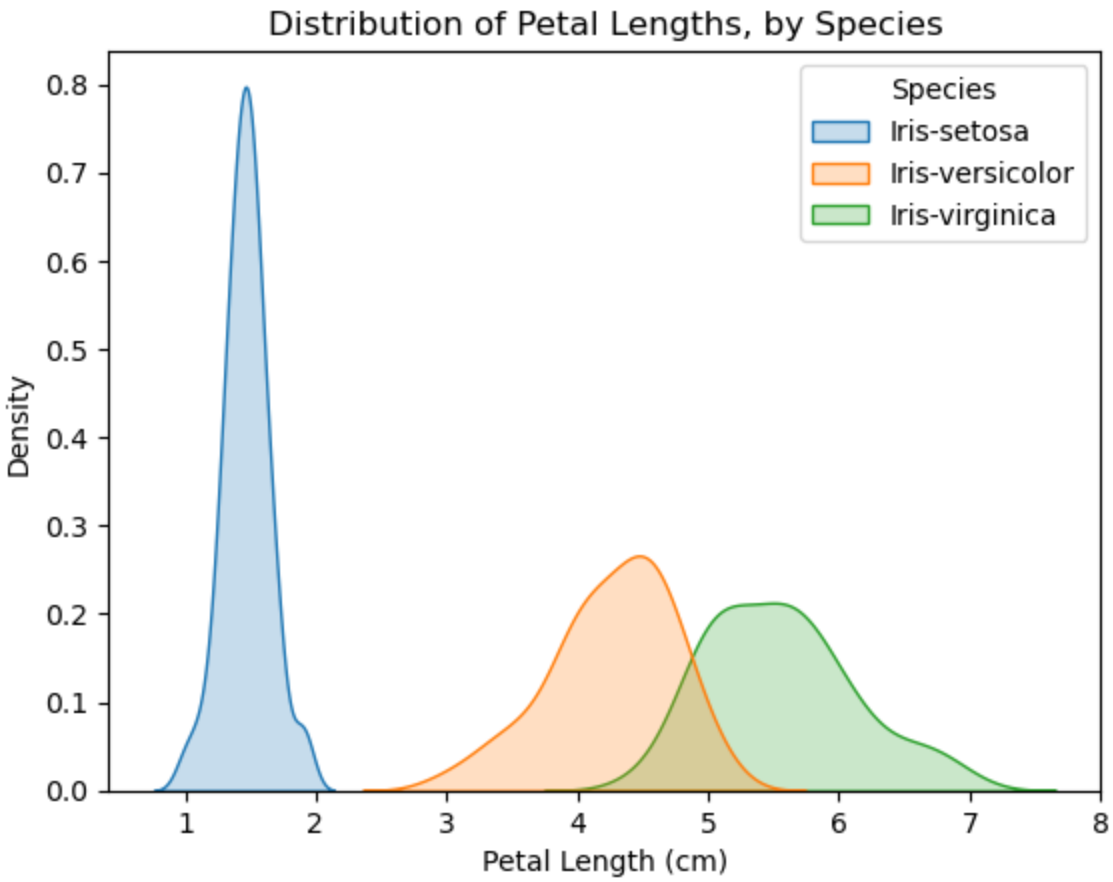
Color-coded plots

In [33]:
```
# Histograms for each species
sns.histplot(data=iris_data, x='Petal Length (cm)', hue='Species')

# Add title
plt.title("Histogram of Petal Lengths, by Species")
```

Out[33]:
```
Text(0.5, 1.0, 'Histogram of Petal Lengths, by Species')
```

In [34]:
```
# KDE plots for each species
sns.kdeplot(data=iris_data, x='Petal Length (cm)', hue='Species', fill=True)

# Add title
plt.title("Distribution of Petal Lengths, by Species")
```

Out[34]:
```
Text(0.5, 1.0, 'Distribution of Petal Lengths, by Species')
```

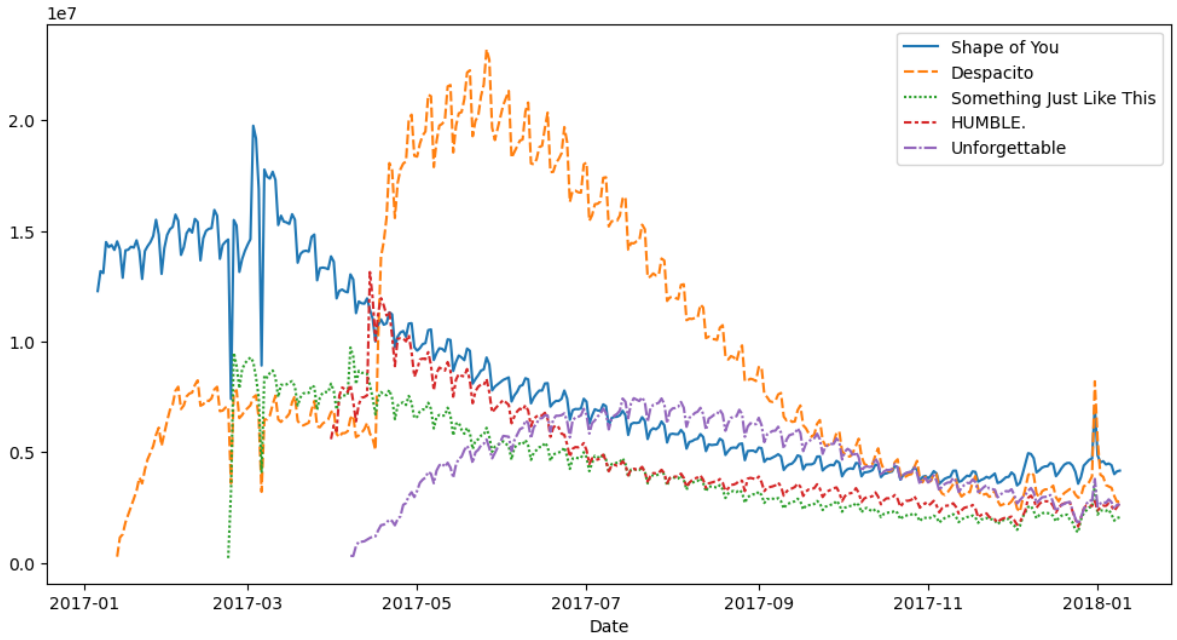


Choosing Plot Types and Custom Styles

Changing styles with seaborn

In [35]:
```
# Line chart
plt.figure(figsize=(12,6))
sns.lineplot(data=spotify_data)
```

Out[35]:
```
<Axes: xlabel='Date'>
```



We can quickly change the style of the figure to a different theme with only a single line of code.

In [36]:
```
# Change the style of the figure to the "dark" theme
sns.set_style("dark")

# Line chart
plt.figure(figsize=(12,6))
sns.lineplot(data=spotify_data)
```

Out[36]: <Axes: xlabel='Date'>