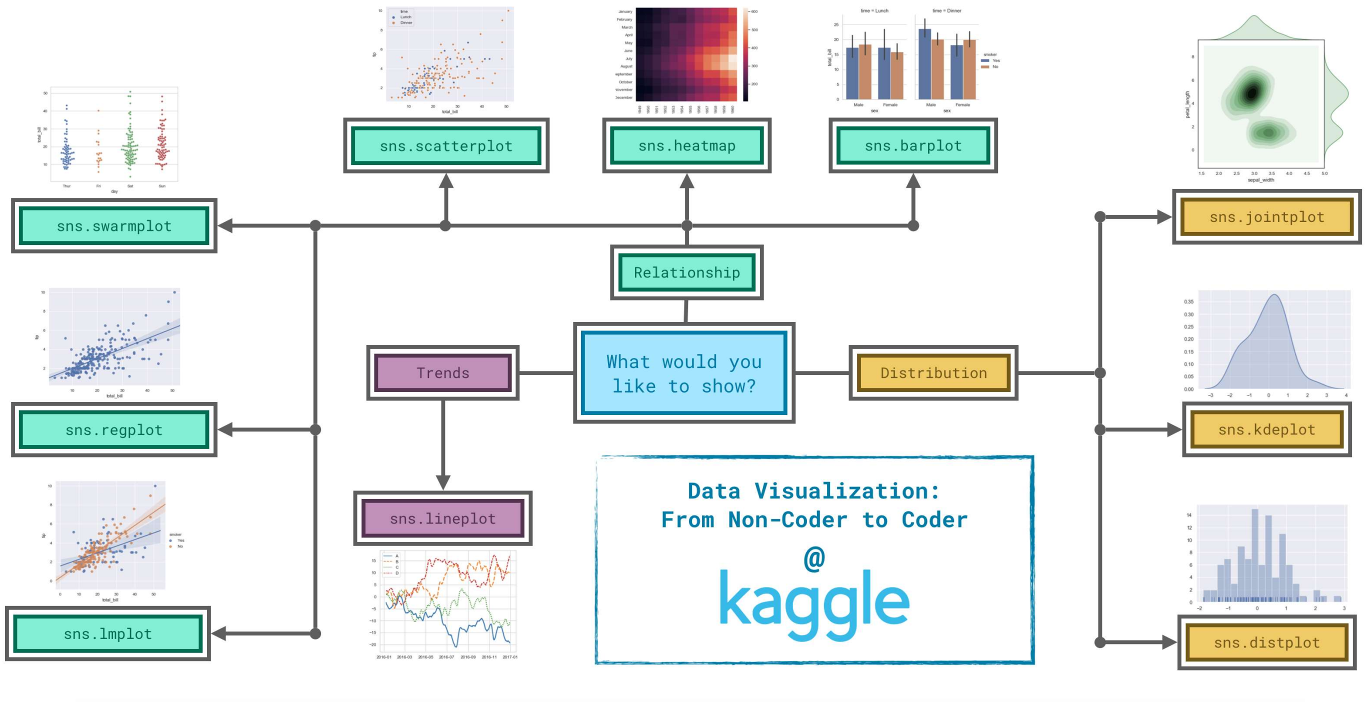
In this micro-course, you've learned how to create many different chart types. Now, you'll organize your knowledge, before learning some quick commands that you can use to change the style of your charts.

What have you learned?



Since it's not always easy to decide how to best tell the story behind your data, we've broken the chart types into three broad categories to help with this.

- **Trends** - A trend is defined as a pattern of change.
  - `sns.lineplot` - **Line charts** are best to show trends over a period of time, and multiple lines can be used to show trends in more than one group.

- **Relationship** - There are many different chart types that you can use to understand relationships between variables in your data.
  - `sns.barplot` - **Bar charts** are useful for comparing quantities corresponding to different groups.
  - `sns.heatmap` - **Heatmaps** can be used to find color-coded patterns in tables of numbers.
  - `sns.scatterplot` - **Scatter plots** show the relationship between two continuous variables; if color-coded, we can also show the relationship with a third categorical variable.
  - `sns.regplot` - Including a **regression line** in the scatter plot makes it easier to see any linear relationship between two variables.
  - `sns.lmplot` - This command is useful for drawing multiple regression lines, if the scatter plot contains multiple, color-coded groups.
  - `sns.swarmplot` - **Categorical scatter plots** show the relationship between a continuous variable and a categorical variable.

- **Distribution** - We visualize distributions to show the possible values that we can expect to see in a variable, along with how likely they are.
  - `sns.distplot` - **Histograms** show the distribution of a single numerical variable.

- `sns.kdeplot` **- KDE plots** (or **2D KDE plots**) show an estimated, smooth distribution of a single numerical variable (or two numerical variables).
- `sns.jointplot` - This command is useful for simultaneously displaying a 2D KDE plot with the corresponding KDE plots for each individual variable.

Changing styles with seaborn

All of the commands have provided a nice, default style to each of the plots. However, you may find it useful to customize how your plots look, and thankfully, this can be accomplished by just adding one more line of code!

As always, we need to begin by setting up the coding environment. (*This code is hidden, but you can un-hide it by clicking on the "Code" button immediately below this text, on the right.*)

Hide Output

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")
```

We'll work with the same code that we used to create a line chart in a previous tutorial. The code below loads the dataset and creates the chart.

In [2]:

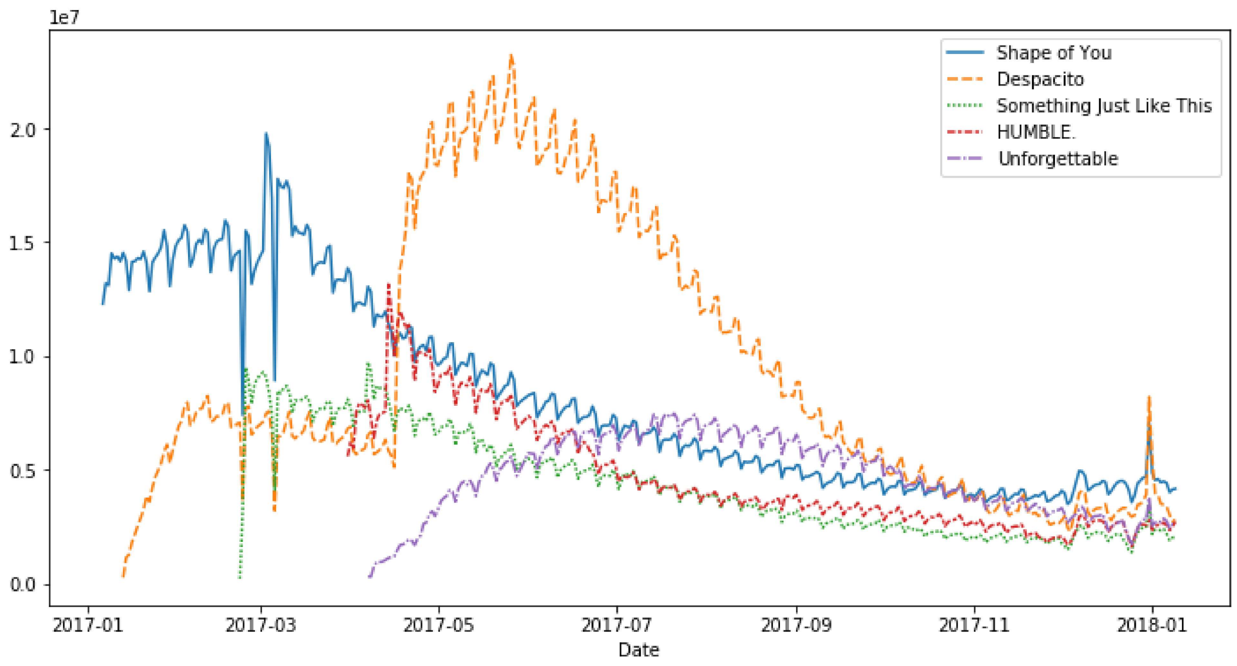
```
# Path of the file to read
spotify_filepath = "../input/spotify.csv"

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

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

Out[2]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f303783eef0>
```



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

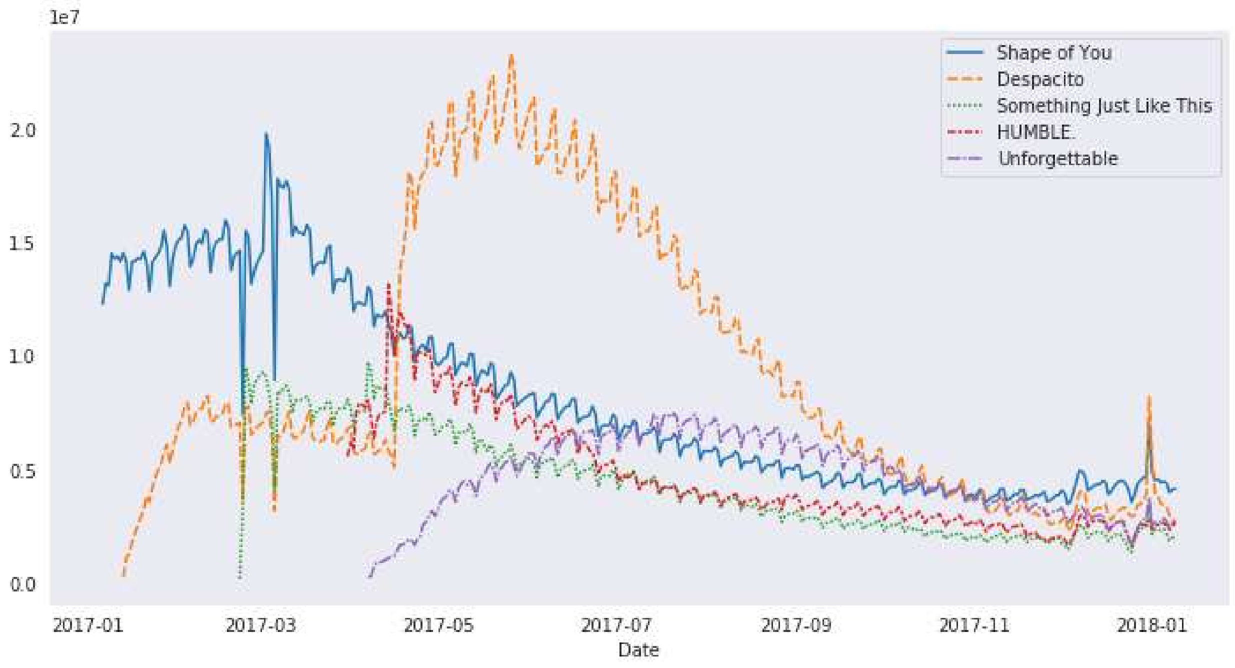
In [3]:

```python
# 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[3]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f30451e2c88>
```



Seaborn has five different themes: (1) `"darkgrid"`, (2) `"whitegrid"`, (3) `"dark"`, (4) `"white"`, and (5) `"ticks"`, and you need only use a command similar to the one in the code cell above (with the chosen theme filled in) to change it.

In the upcoming exercise, you'll experiment with these themes to see which one you like most!

What's next?

Explore seaborn styles in a quick **coding exercise**!