In [1]:
```
# Pandas for managing datasets
import pandas as pd

# Matplotlib for additional customization
from matplotlib import pyplot as plt

import matplotlib as mp

%matplotlib inline
#In Jupyter Notebook, you can also include %matplotlib inline
#to display your plots inside your notebook.

# Seaborn for plotting and styling
import seaborn as sns

import os
```

In [2]:
```
# https://elitedatascience.com/wp-content/uploads/2017/04/Pokemon.csv

os.getcwd()
```

Out[2]:
```
'C:\\Users\\admin'
```

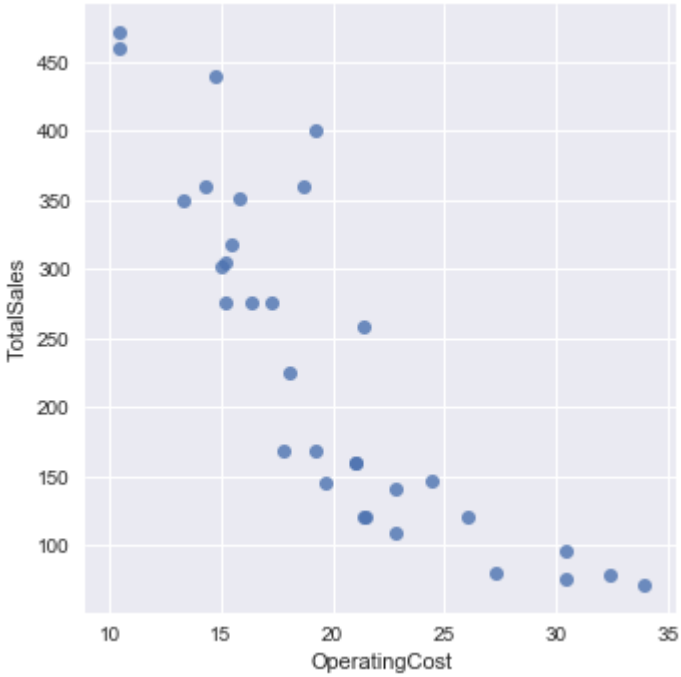
In [3]:
```
os.chdir('C:\\Users\\admin\\pandas\\DataSets')
```

In [4]:
```
df = pd.read_csv("Pokemon.csv")

stores = pd.read_csv("stores.csv")
```

In [ ]:
```
# Drawing a scatter plot
# Recommended way
sns.lmplot(x='Attack', y='Defense', data=df)

# Alternative way
# sns.lmplot(x=df.Attack, y=df.Defense)
```

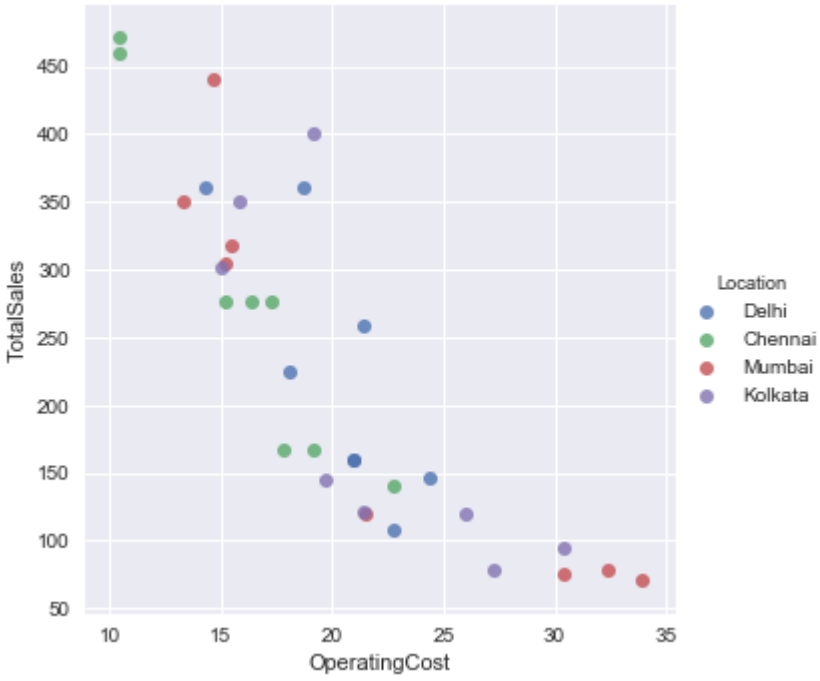
In [9]:
```
# Scatter plot between TotalSales and OperatingCost
sns.lmplot(x = "OperatingCost", y = "TotalSales", data = stores, fit_reg=False)
# fit_reg = False
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x12781668>



In [10]:
```
# Adding color to scatterplot

sns.lmplot(x = "OperatingCost", y = "TotalSales", data = stores,fit_reg=False,hue
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x123abbe0>



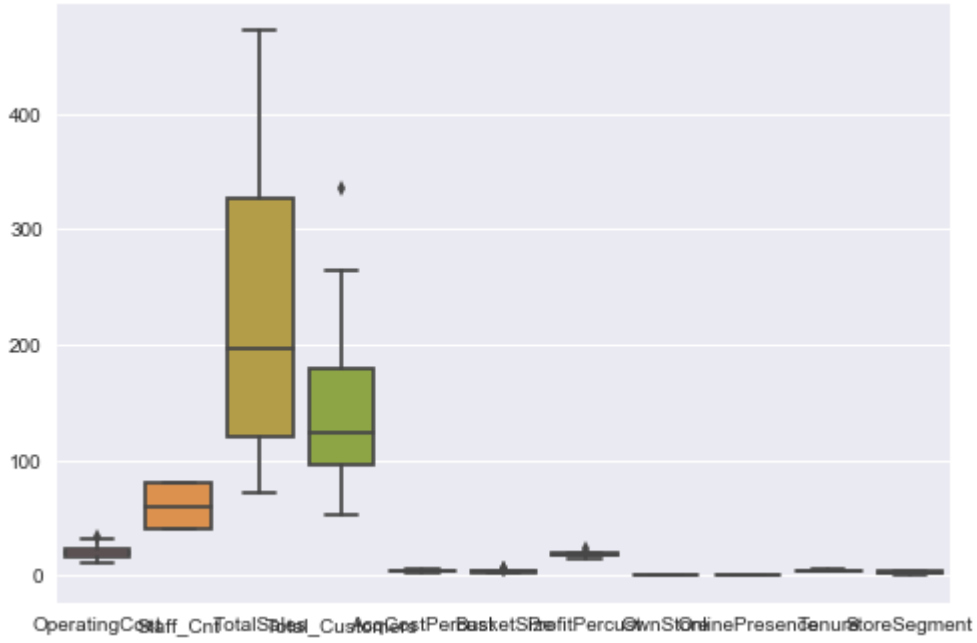
In [ ]:
```
sns.lmplot(x='Attack', y='Defense', data=df,
           fit_reg=False, # No regression line
           hue='Stage')   # Color by evolution stage
```

In [ ]:
```
# Plot using Seaborn
sns.lmplot(x='Attack', y='Defense', data=df,
           fit_reg=False,
           hue='Stage')

# Tweak using Matplotlib
plt.ylim(0, 500)
plt.xlim(0, 300)
```

In [ ]:
```
# Whiskers in Seaborn
```

In [12]:
```
sns.boxplot(data=stores)
```

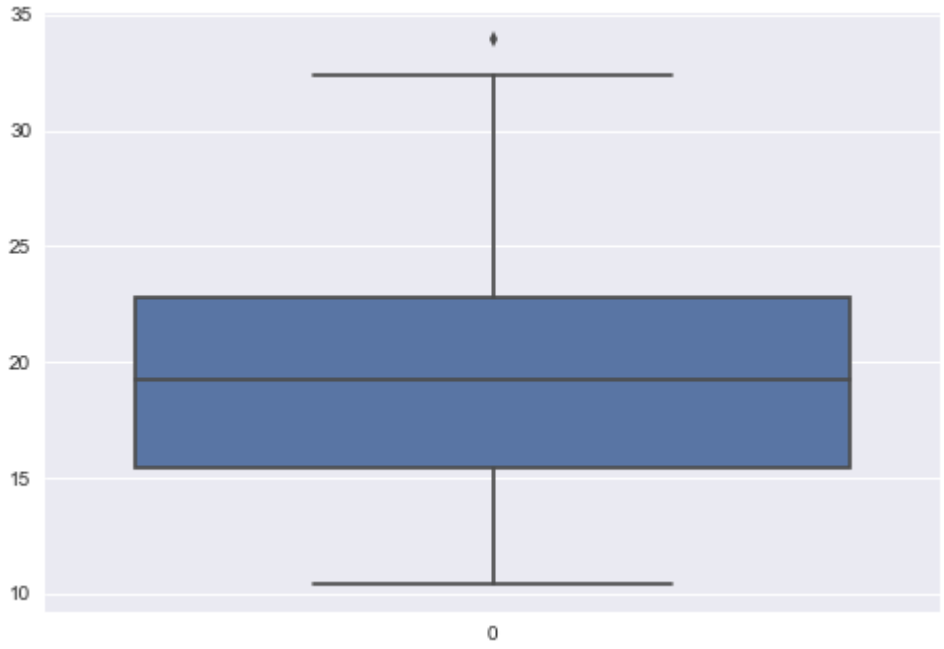
Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x13608dd8>
```



In [11]:
```
sns.boxplot(data=stores.OperatingCost)
```

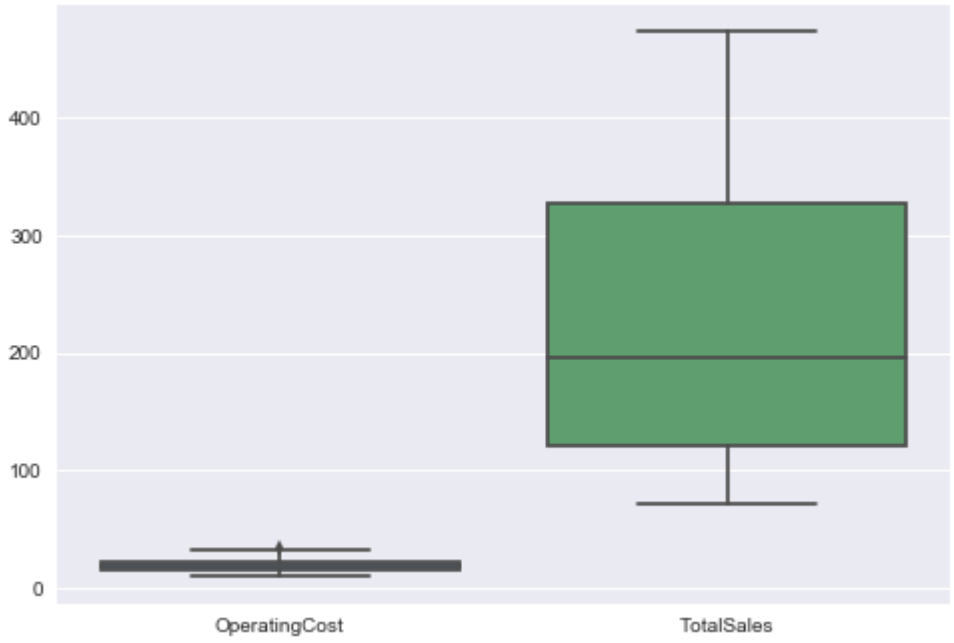
```
C:\Users\admin\Anaconda2\lib\site-packages\seaborn\categorical.py:454: FutureWa
rning: remove_na is deprecated and is a private function. Do not use.
  box_data = remove_na(group_data)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x133777b8>



In [13]:
```
sns.boxplot(data = stores.loc[:,["OperatingCost","TotalSales"]])
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x13c4e898>



In [ ]:
```
# Violin plots are useful alternatives to box plots.

# They show the distribution (through the thickness of the violin) instead of
# only the summary statistics.

# Set theme
sns.set_style('whitegrid')

# Violin plot
sns.violinplot(x='Location', y='TotalSales', data=stores)
```

In [ ]:
```
col_Pallete = ["red","blue","green","yellow"]
sns.violinplot(x='Location', y='TotalSales', data=stores,palette=col_Pallete)
```

In [ ]:
```
# color reference can be found at
# http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

# Generating colors automatically

# Getting all color names so as to generate random colors for the graphs
colorDict = mp.colors.cnames
# from matplotlib package, get the colors as mentioned in the above pdf

colors = []
for i in colorDict:
    colors.append(i)

# colors variable is a list data structure of all available colors

# to get n random values from colors list
import random
c = random.sample(colors,4)
print c

sns.violinplot(x='Location', y='TotalSales', data=stores,palette=c)
```

In [ ]:
```
# Violin plots are great for visualizing distributions.

# However, for a dataset, we may want to simply display each point.

# That's where the swarm plot comes in.

# This visualization will show each point, while "stacking" those with similar va

# Swarm plot with Pokemon color palette

sns.swarmplot(x='Location', y='TotalSales', data=stores,palette=c)
```

In [ ]:
```
# Overlaying plots.

# It's pretty straightforward to overlay plots using Seaborn,
# and it works the same way as with Matplotlib.

# First, we'll make our figure larger using Matplotlib.
# Then, we'll plot the violin plot. However, we'll set inner=None to remove the b
# Next, we'll plot the swarm plot. This time, we'll make the points black so they
# Finally, we'll set a title using Matplotlib.
```

In [ ]:
```
# Set figure size with matplotlib
plt.figure(figsize=(10,6))
sns.violinplot(x='Location', y='TotalSales', data=stores,palette=c,alpha=0.1)
sns.swarmplot(x='Location', y='TotalSales', data=stores,color='k')

# Set title with matplotlib
plt.title('TotalSales by Location')
```

Heatmap

Heatmaps help you visualize matrix-like data.

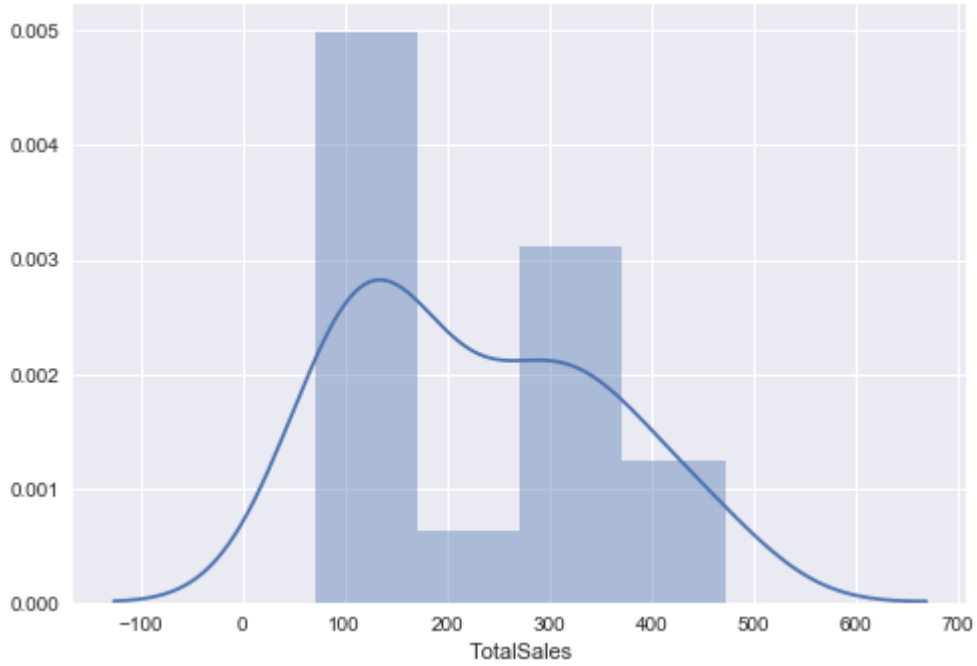
In [ ]:
```
stats_df = df.drop(['Total', 'Stage', 'Legendary'], axis=1)

# Calculate correlations
#corr = stats_df.corr()

# Heatmap
#sns.heatmap(corr)

stores1 = stores.iloc[:,4:]
corrStores = stores1.corr()
sns.heatmap(corrStores)
```

Histogram

Histograms allow you to plot the distributions of numeric variables.

In [15]:
```
# Distribution Plot (a.k.a. Histogram)
sns.distplot(stores.TotalSales)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x13e1a390>



Bar Plot

Bar plots help you visualize the distributions of categorical variables.

In [ ]:
```
# Count Plot (a.k.a. Bar Plot)
sns.countplot(x='Type 1', data=df, palette=pkmn_type_colors)

# Rotate x-labels
plt.xticks(rotation=-45)
```

In [ ]:
```
sns.countplot(x='StoreType', data=stores, palette=col_Pallete)

# Rotate x-labels
plt.xticks(rotation=-45)
```

Factor Plot

Factor plots make it easy to separate plots by categorical classes.

In [ ]:
```
# Factor Plot
g = sns.factorplot(x='Type 1',
                   y='Attack',
                   data=df,
                   hue='Stage',  # Color by stage
                   col='Stage',  # Separate by stage
                   kind='swarm') # Swarmplot

# Rotate x-axis labels
g.set_xticklabels(rotation=-45)
```

In [ ]:
```
g = sns.factorplot(x='Location',
                   y='TotalSales',
                   data=stores,
                   hue='Location',  # Color by stage
                   col='StoreType',  # Separate by stage
                   kind='bar') # barplot

# Rotate x-axis labels
g.set_xticklabels(rotation=-45)
```

Density Plot

Density plots display the distribution between two variables.

In [ ]:
```
# Density Plot
sns.kdeplot(df.Attack, df.Defense)
```

In [ ]:
```
sns.kdeplot(stores.TotalSales, stores.OperatingCost)
```

In [ ]:
```

```