In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import re
import time
from datetime import datetime
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
from urllib.request import urlopen
from bs4 import BeautifulSoup
import requests
import warnings
warnings.filterwarnings('ignore')
print('Setup Complete!')
```

Setup Complete!

Web Scraping

In [2]:
```
no_pages = 2

def get_data(pageNo):
    headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0", "Accept-Encoding":"gzip, deflate", "Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8", "DNT":"1","Connection":"close", "Upgrade-Insecure-Requests":"1"}

    r = requests.get('https://www.amazon.com/gp/bestsellers/books/ref=zg_bs_pg_'+str(pageNo)+'?ie=UTF8&pg='+str(pageNo), headers=headers)# proxies=proxies)
    content = r.content
    soup = BeautifulSoup(content)
    #print(soup)

    alls = []
    for d in soup.findAll('div', attrs={'class':'a-section a-spacing-none aok-relative'}):
        #print(d)
        name = d.find('span', attrs={'class':'zg-text-center-align'})
        n = d.find('img', alt=True)
        #print(n['alt'])
        author = d.find('a', attrs={'class':'a-size-small a-link-child'})
        rating = d.find('span', attrs={'class':'a-icon-alt'})
        users_rated = d.find('a', attrs={'class':'a-size-small a-link-normal'})
        price = d.find('span', attrs={'class':'p13n-sc-price'})

        all1=[]

        if name is not None:
            #print(n['alt'])
            all1.append(n['alt'])
        else:
            all1.append("unknown-product")

        if author is not None:
            #print(author.text)
            all1.append(author.text)
        elif author is None:
            author = d.find('span', attrs={'class':'a-size-small a-color-base'})
            if author is not None:
                all1.append(author.text)
            else:
                all1.append('0')

        if rating is not None:
            #print(rating.text)
            all1.append(rating.text)
        else:
            all1.append('-1')

        if users_rated is not None:
            #print(price.text)
            all1.append(users_rated.text)
        else:
            all1.append('0')

        if price is not None:
            #print(price.text)
            all1.append(price.text)
        else:
            all1.append('0')
        alls.append(all1)
    return alls
```

In [3]:
```
results = []
for i in range(1, no_pages+1):
    results.append(get_data(i))
flatten = lambda l: [item for sublist in l for item in sublist]
df = pd.DataFrame(flatten(results),columns=['Book Name','Author','Rating','Customers_Rated', 'Price'])
df.to_csv('amazon_products.csv', index=False, encoding='utf-8')
```

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

In [5]:
```
df.shape
```

Out[5]:
```
(0, 5)
```

In [6]:
```
df.head()
```

Out[6]:

	Book Name	Author	Rating	Customers_Rated	Price

In [7]:
```
df.tail()
```

Out[7]:

	Book Name	Author	Rating	Customers_Rated	Price

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Book Name       0 non-null      object
 1   Author          0 non-null      object
 2   Rating          0 non-null      object
 3   Customers_Rated 0 non-null      object
 4   Price           0 non-null      object
dtypes: object(5)
memory usage: 0.0+ bytes
```

In [9]:
```
df['Rating'] = df['Rating'].apply(lambda x: x.split()[0])
df['Rating'] = pd.to_numeric(df['Rating'])
df["Price"] = df["Price"].str.replace('$', '')
df["Price"] = df["Price"].str.replace(',', '')
df['Price'] = df['Price'].apply(lambda x: x.split('.')[0])
df['Price'] = df['Price'].astype(int)
df["Customers_Rated"] = df["Customers_Rated"].str.replace(',', '')
df['Customers_Rated'] = pd.to_numeric(df['Customers_Rated'], errors='ignore')
df.head()
```

Out[9]:

	Book Name	Author	Rating	Customers_Rated	Price

In [10]:
```
df.dtypes
```

Out[10]:
```
Book Name          object
Author             object
Rating             int64
Customers_Rated    int64
Price              int32
dtype: object
```

In [11]:
```
## Replace the zero values in the DataFrame to NaN.
df.replace(str(0), np.nan, inplace=True)
df.replace(0, np.nan, inplace=True)
```

In [12]:
```
## Counting the Number of NANs in the DataFrame
count_nan = len(df) - df.count()
count_nan
```

Out[12]:
```
Book Name          0
Author             0
Rating             0
Customers_Rated    0
Price              0
dtype: int64
```

In [13]:
```
## Let's drop these NaNs.
df = df.dropna()
```

In [14]:
```
data = df.sort_values(["Price"], axis=0, ascending=False)[:15]
data
```

Out[14]:

	Book Name	Author	Rating	Customers_Rated	Price

In [15]:
```
from bokeh.models import ColumnDataSource
from bokeh.transform import dodge
from bokeh.io import curdoc
curdoc().clear()
from bokeh.io import push_notebook, show, output_notebook
from bokeh.layouts import row
from bokeh.plotting import figure
from bokeh.transform import factor_cmap
from bokeh.models import Legend
output_notebook()
```

BokehJS 2.4.3 successfully loaded.

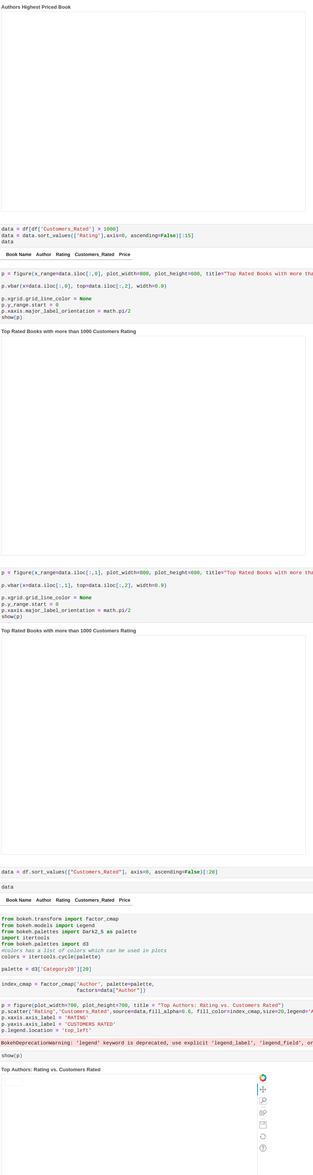
In [16]:
```
p = figure(x_range=data.iloc[:,1], plot_width=800, plot_height=550, title="Authors Highest Priced Book", toolbar_location=None, tools="")

p.vbar(x=data.iloc[:,1], top=data.iloc[:,4], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.xaxis.major_label_orientation = math.pi/2
```

In [17]:
```
show(p)
```

Authors Highest Priced Book



In [18]:
```
data = df[df['Customers_Rated'] > 1000]
data = data.sort_values(['Rating'],axis=0, ascending=False)[:15]
data
```

Out[18]:

	Book Name	Author	Rating	Customers_Rated	Price

In [19]:
```
p = figure(x_range=data.iloc[:,0], plot_width=800, plot_height=600, title="Top Rated Books with more than 1000 Customers Rating", toolbar_location=None, tools="")

p.vbar(x=data.iloc[:,0], top=data.iloc[:,2], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.xaxis.major_label_orientation = math.pi/2
show(p)
```

Top Rated Books with more than 1000 Customers Rating

In [20]:
```
p = figure(x_range=data.iloc[:,1], plot_width=800, plot_height=600, title="Top Rated Books with more than 1000 Customers Rating", toolbar_location=None, tools="")

p.vbar(x=data.iloc[:,1], top=data.iloc[:,2], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0
p.xaxis.major_label_orientation = math.pi/2
show(p)
```

Top Rated Books with more than 1000 Customers Rating

In [21]:
```
data = df.sort_values(["Customers_Rated"], axis=0, ascending=False)[:20]
```

In [22]:
```
data
```

Out[22]:

	Book Name	Author	Rating	Customers_Rated	Price

In [23]:
```
from bokeh.transform import factor_cmap
from bokeh.models import Legend
from bokeh.palettes import Dark2_8 as palette
import itertools
from bokeh.palettes import d3
#colors has a list of colors which can be used in plots
colors = itertools.cycle(palette)

palette = d3['Category20'][20]
```

In [24]:
```
index_cmap = factor_cmap('Author', palette=palette,
                         factors=data["Author"])
```

In [25]:
```
p = figure(plot_width=700, plot_height=700, title = "Top Authors: Rating vs. Customers Rated")
p.scatter('Rating','Customers_Rated',source=data,fill_alpha=0.6, fill_color=index_cmap,size=20,legend='Author')
p.xaxis.axis_label = 'RATING'
p.yaxis.axis_label = 'CUSTOMERS RATED'
p.legend.location = 'top_left'
```

```
BokehDeprecationWarning: 'legend' keyword is deprecated, use explicit 'legend_label', 'legend_field', or 'legend_group' keywords instead
```

In [26]:
```
show(p)
```

Top Authors: Rating vs. Customers Rated