

SalesAnalysis

January 10, 2021

1 Sales Analysis

Import necessary libraries

```
[1]: import os
import pandas as pd
```

Merge data from each month into one CSV

```
[2]: path = "./Sales_Data"
files = [file for file in os.listdir(path) if not file.startswith('.')] #_
↳ Ignore hidden files

all_months_data = pd.DataFrame()

for file in files:
    current_data = pd.read_csv(path+"/"+file)
    all_months_data = pd.concat([all_months_data, current_data])

all_months_data.to_csv("all_data_copy.csv", index=False)
```

Read in updated dataframe

```
[2]: all_data = pd.read_csv("all_data.csv")
all_data.head()
```

```
[2]:
```

	Order ID	Product	Quantity	Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95		
1	NaN	NaN	NaN	NaN	NaN	
2	176559	Bose SoundSport Headphones	1	99.99		
3	176560	Google Phone	1	600		
4	176560	Wired Headphones	1	11.99		

	Order Date	Purchase Address
0	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

4 04/12/19 14:38 669 Spruce St, Los Angeles, CA 90001

1.0.1 Clean up the data!

The first step in this is figuring out what I need to clean. I have found in practice, that you find things you need to clean as you perform operations and get errors. Based on the error, you decide how you should go about cleaning the data

Drop rows of NaN

```
[21]: # Find NaN
nan_df = all_data[all_data.isna().any(axis=1)]
display(nan_df.head())

all_data = all_data.dropna(how='all')
all_data.head()
```

	Order ID	Product	Quantity	Ordered	Price Each	Order Date	Purchase Address
1	NaN	NaN		NaN	NaN	NaN	NaN
356	NaN	NaN		NaN	NaN	NaN	NaN
735	NaN	NaN		NaN	NaN	NaN	NaN
1433	NaN	NaN		NaN	NaN	NaN	NaN
1553	NaN	NaN		NaN	NaN	NaN	NaN

```
[21]: Order ID          Product Quantity Ordered Price Each \
0  176558      USB-C Charging Cable          2      11.95
2  176559  Bose SoundSport Headphones          1      99.99
3  176560          Google Phone              1        600
4  176560      Wired Headphones              1      11.99
5  176561      Wired Headphones              1      11.99
```

	Order Date	Purchase Address
0	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Get rid of text in order date column

```
[22]: all_data = all_data[all_data['Order Date'].str[0:2]!='Or']
```

Make columns correct type

```
[23]: all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
```

1.0.2 Augment data with additional columns

Add month column

```
[24]: all_data['Month'] = all_data['Order Date'].str[0:2]
      all_data['Month'] = all_data['Month'].astype('int32')
      all_data.head()
```

```
[24]:  Order ID          Product  Quantity Ordered  Price Each  \
0    176558  USB-C Charging Cable                2      11.95
2    176559  Bose SoundSport Headphones            1      99.99
3    176560        Google Phone                1     600.00
4    176560      Wired Headphones                1      11.99
5    176561      Wired Headphones                1      11.99

      Order Date          Purchase Address  Month
0  04/19/19 08:46      917 1st St, Dallas, TX 75001      4
2  04/07/19 22:30      682 Chestnut St, Boston, MA 02215      4
3  04/12/19 14:38      669 Spruce St, Los Angeles, CA 90001      4
4  04/12/19 14:38      669 Spruce St, Los Angeles, CA 90001      4
5  04/30/19 09:27      333 8th St, Los Angeles, CA 90001      4
```

Add month column (alternative method)

```
[47]: all_data['Month 2'] = pd.to_datetime(all_data['Order Date']).dt.month
      all_data.head()
```

```
[47]:  Order ID          Product  Quantity Ordered  Price Each  \
0    176558  USB-C Charging Cable                2      11.95
2    176559  Bose SoundSport Headphones            1      99.99
3    176560        Google Phone                1       600
4    176560      Wired Headphones                1      11.99
5    176561      Wired Headphones                1      11.99

      Order Date          Purchase Address  Month  Month 2
0  04/19/19 08:46      917 1st St, Dallas, TX 75001      4      4
2  04/07/19 22:30      682 Chestnut St, Boston, MA 02215      4      4
3  04/12/19 14:38      669 Spruce St, Los Angeles, CA 90001      4      4
4  04/12/19 14:38      669 Spruce St, Los Angeles, CA 90001      4      4
5  04/30/19 09:27      333 8th St, Los Angeles, CA 90001      4      4
```

Add city column

```
[25]: def get_city(address):
      return address.split(",")[1].strip(" ")

      def get_state(address):
          return address.split(",")[2].split(" ")[1]
```

```
all_data['City'] = all_data['Purchase Address'].apply(lambda x: f"{get_city(x)}\n
↳ ({get_state(x)})")
all_data.head()
```

```
[25]:   Order ID          Product  Quantity Ordered  Price Each  \
0    176558      USB-C Charging Cable             2         11.95
2    176559  Bose SoundSport Headphones             1         99.99
3    176560          Google Phone             1        600.00
4    176560      Wired Headphones             1         11.99
5    176561      Wired Headphones             1         11.99
```

```
      Order Date          Purchase Address  Month  \
0  04/19/19 08:46      917 1st St, Dallas, TX 75001      4
2  04/07/19 22:30      682 Chestnut St, Boston, MA 02215      4
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4
5  04/30/19 09:27      333 8th St, Los Angeles, CA 90001      4
```

```
      City
0    Dallas (TX)
2    Boston (MA)
3  Los Angeles (CA)
4  Los Angeles (CA)
5  Los Angeles (CA)
```

1.1 Data Exploration!

Question 1: What was the best month for sales? How much was earned that month?

```
[26]: all_data['Sales'] = all_data['Quantity Ordered'].astype('int') *
↳ all_data['Price Each'].astype('float')
```

```
[27]: all_data.groupby(['Month']).sum()
```

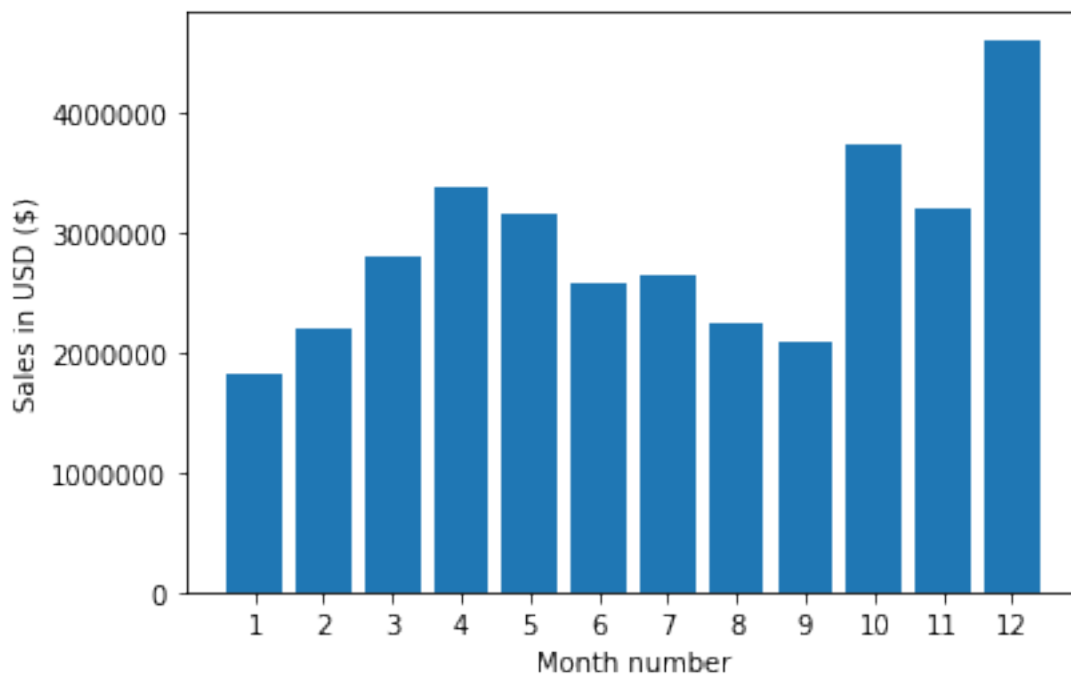
```
[27]:   Quantity Ordered  Price Each  Sales
Month
1           10903  1.811768e+06  1.822257e+06
2           13449  2.188885e+06  2.202022e+06
3           17005  2.791208e+06  2.807100e+06
4           20558  3.367671e+06  3.390670e+06
5           18667  3.135125e+06  3.152607e+06
6           15253  2.562026e+06  2.577802e+06
7           16072  2.632540e+06  2.647776e+06
8           13448  2.230345e+06  2.244468e+06
9           13109  2.084992e+06  2.097560e+06
10          22703  3.715555e+06  3.736727e+06
11          19798  3.180601e+06  3.199603e+06
12          28114  4.588415e+06  4.613443e+06
```

```
[28]: import matplotlib.pyplot as plt

months = range(1,13)
print(months)

plt.bar(months,all_data.groupby(['Month']).sum()['Sales'])
plt.xticks(months)
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month number')
plt.show()
```

```
range(1, 13)
```



Question 2: What city sold the most product?

```
[29]: all_data.groupby(['City']).sum()
```

```
[29]:
```

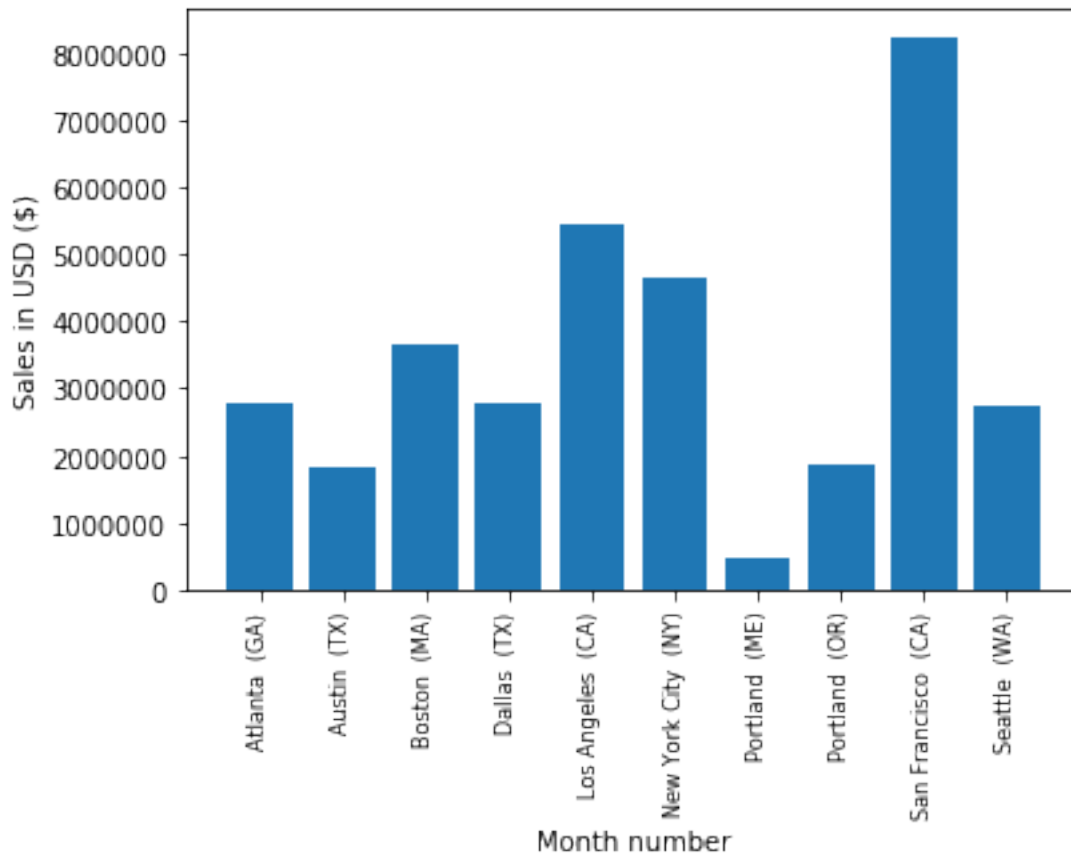
	Quantity Ordered	Price Each	Month	Sales
City				
Atlanta (GA)	16602	2.779908e+06	104794	2.795499e+06
Austin (TX)	11153	1.809874e+06	69829	1.819582e+06
Boston (MA)	22528	3.637410e+06	141112	3.661642e+06
Dallas (TX)	16730	2.752628e+06	104620	2.767975e+06
Los Angeles (CA)	33289	5.421435e+06	208325	5.452571e+06
New York City (NY)	27932	4.635371e+06	175741	4.664317e+06

Portland (ME)	2750	4.471893e+05	17144	4.497583e+05
Portland (OR)	11303	1.860558e+06	70621	1.870732e+06
San Francisco (CA)	50239	8.211462e+06	315520	8.262204e+06
Seattle (WA)	16553	2.733296e+06	104941	2.747755e+06

```
[30]: import matplotlib.pyplot as plt

keys = [city for city, df in all_data.groupby(['City'])]

plt.bar(keys, all_data.groupby(['City']).sum()['Sales'])
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month number')
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



Question 3: What time should Idisplay advertisements to maximize likelihood of customer's buying product?

```
[31]: # Add hour column
all_data['Hour'] = pd.to_datetime(all_data['Order Date']).dt.hour
```

```
all_data['Minute'] = pd.to_datetime(all_data['Order Date']).dt.minute
all_data['Count'] = 1
all_data.head()
```

```
[31]:
```

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

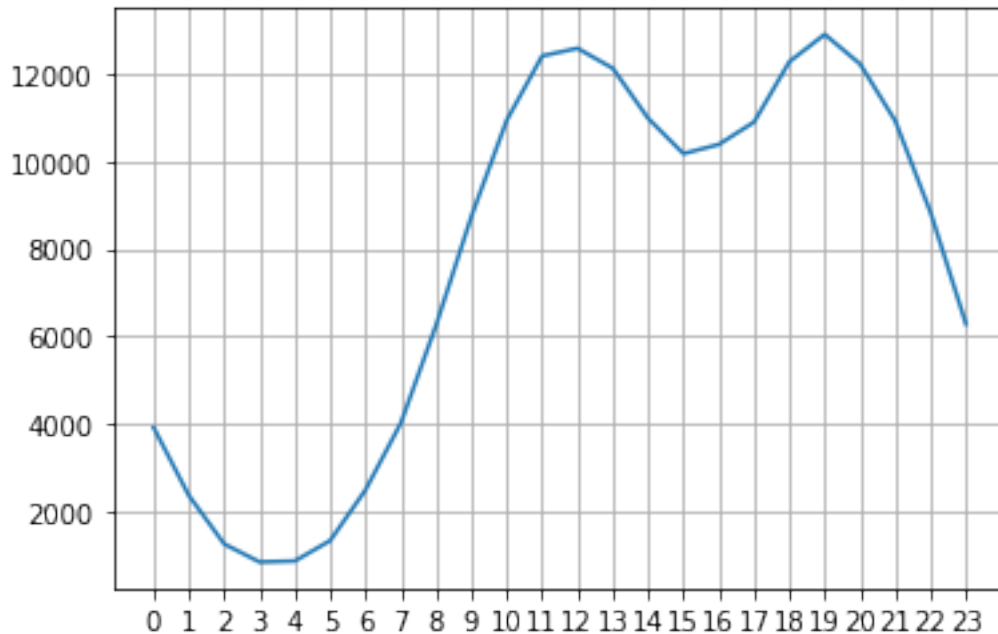
	Order Date	Purchase Address	Month	\
0	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	
4	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	
5	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	

		City	Sales	Hour	Minute	Count
0	Dallas	(TX)	23.90	8	46	1
2	Boston	(MA)	99.99	22	30	1
3	Los Angeles	(CA)	600.00	14	38	1
4	Los Angeles	(CA)	11.99	14	38	1
5	Los Angeles	(CA)	11.99	9	27	1

```
[32]: keys = [pair for pair, df in all_data.groupby(['Hour'])]

plt.plot(keys, all_data.groupby(['Hour']).count()['Count'])
plt.xticks(keys)
plt.grid()
plt.show()

# My recommendation is slightly before 11am or 7pm
```



Question 4: What products are most often sold together?

```
[48]: # https://stackoverflow.com/questions/43348194/
      ↪ pandas-select-rows-if-id-appear-several-time
df = all_data[all_data['Order ID'].duplicated(keep=False)]

# Referenced: https://stackoverflow.com/questions/27298178/
      ↪ concatenate-strings-from-several-rows-using-pandas-groupby
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ', '.join(x))
df2 = df[['Order ID', 'Grouped']].drop_duplicates()
```

C:\Users\keith\Anaconda3\lib\site-packages\ipykernel_launcher.py:5:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

"""

```
[47]: # Referenced: https://stackoverflow.com/questions/52195887/
      ↪ counting-unique-pairs-of-numbers-into-a-python-dictionary
from itertools import combinations
from collections import Counter
```



```

count = Counter()

for row in df2['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

for key,value in count.most_common(10):
    print(key, value)

```

```

('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple AirPods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92

```

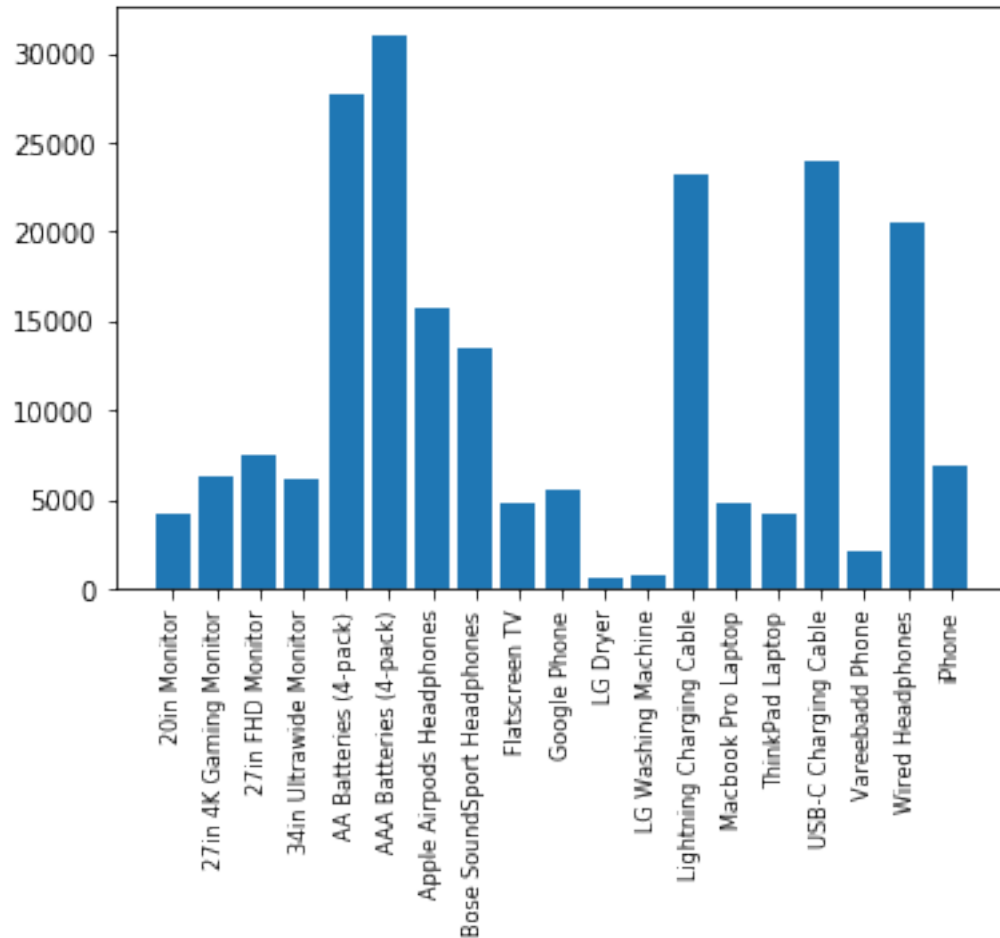
What product sold the most? Why do you think it sold the most?

```

[76]: product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']

keys = [pair for pair, df in product_group]
plt.bar(keys, quantity_ordered)
plt.xticks(keys, rotation='vertical', size=8)
plt.show()

```



[75]: # Referenced: <https://stackoverflow.com/questions/14762181/adding-a-y-axis-label-to-secondary-y-axis-in-matplotlib>

```
prices = all_data.groupby('Product').mean()['Price Each']

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(keys, quantity_ordered, color='g')
ax2.plot(keys, prices, color='b')

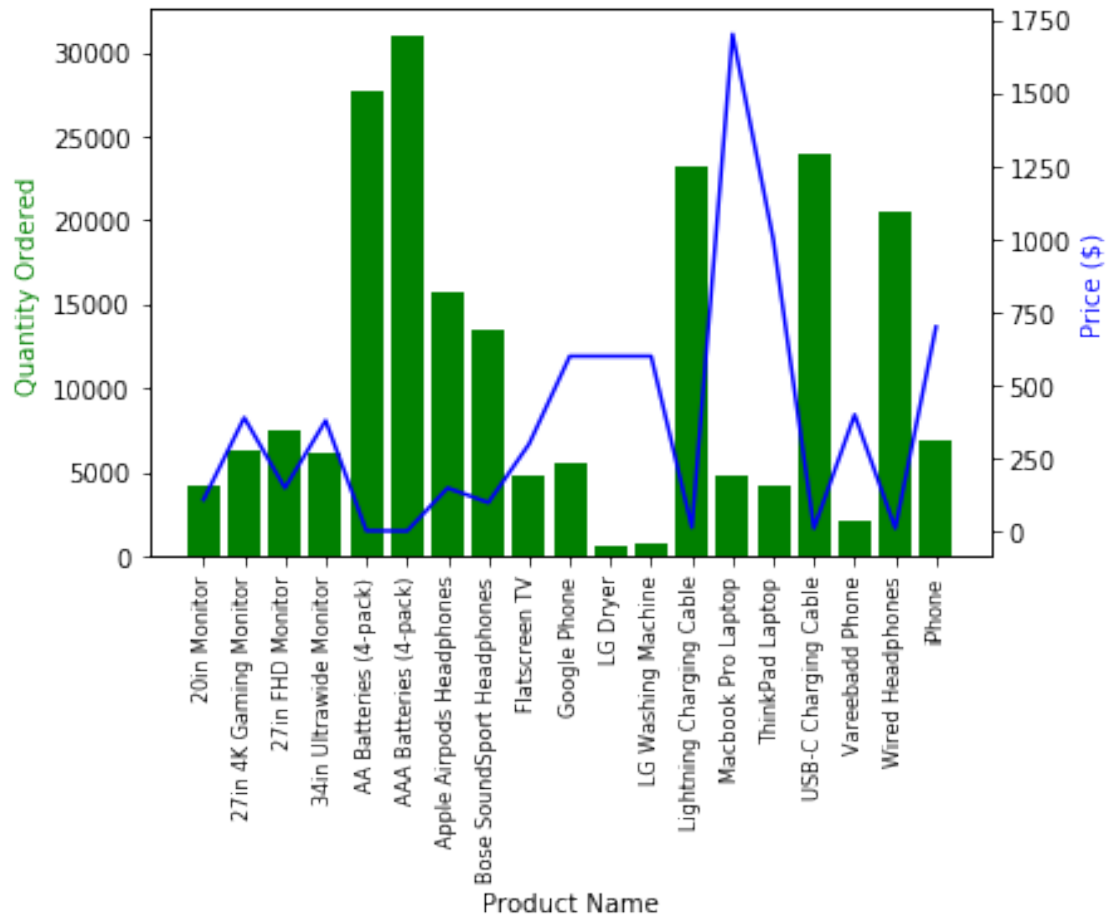
ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered', color='g')
ax2.set_ylabel('Price ($)', color='b')
ax1.set_xticklabels(keys, rotation='vertical', size=8)

fig.show()
```

```

C:\Users\keith\Anaconda3\lib\site-packages\ipykernel_launcher.py:16:
UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
app.launch_new_instance()

```



```
[ ]:
```