Import Libraries

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sn
```

In [3]:
```
import os
from os.path import isfile, join
from os import listdir

cwd = os.path.abspath('')
files = os.listdir(cwd)
files
```

Out[3]:
```
['.ipynb_checkpoints',
 '.vscode',
 'DS_Business.ipynb',
 'helloworld.cpp',
 'maskify.py',
 'merged_data.csv',
 'Pandas-Data-Science-Tasks-master',
 'Sales_Data']
```

Download the data using pandas and os

In [4]:
```
mypath = './Sales_Data/'

onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]

print(onlyfiles)


#Merge 12 months of data into one single file

merged_data = pd.DataFrame()

for file in onlyfiles:
    df = pd.read_csv(mypath + file)
    merged_data = pd.concat([merged_data, df])

merged_data.to_csv("merged_data.csv", index = False)

merged_data.head()
```

```
['Sales_April_2019.csv', 'Sales_August_2019.csv', 'Sales_December_2019.csv', 'Sales_Febr
uary_2019.csv', 'Sales_January_2019.csv', 'Sales_July_2019.csv', 'Sales_June_2019.csv',
'Sales_March_2019.csv', 'Sales_May_2019.csv', 'Sales_November_2019.csv', 'Sales_October_
2019.csv', 'Sales_September_2019.csv']
```

Out[4]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

Clean the data and remove uneccessary features

In [31]:
```
nan_df= merged_data[merged_data.isna().any(axis = 1)]
nan_df.head()

merged_data = merged_data.dropna(how='all')
merged_data
```

Out[31]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
...	...	...	...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	9	8.97	Los Angeles (CA)
11682	259354	iPhone	1	700.00	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	9	700.00	San Francisco (CA)

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City
11683	259355	iPhone	1	700.00	09/23/19 07:39	220 12th St, San Francisco, CA 94016	9	700.00	San Francisco (CA)
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	9	379.99	San Francisco (CA)
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	9	11.95	San Francisco (CA)

185950 rows × 9 columns

Solve issue with Month column having the string 'Or' and delete it

In [32]:
```
#Solve the [ invalid literal for int() with base 10: 'Or'] issue

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

Augment data with additional columns

In [33]:
```
#Add month column

merged_data['Month'] = merged_data['Order Date'].str[0:2]

merged_data.head()
```

Out[33]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	04	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04	11.99	Los Angeles (CA)

In [34]:
```
merged_data['Month'] = merged_data['Month'].astype('int32')
```

Convert columns to the right data type

In [35]:
```
merged_data['Quantity Ordered'] = pd.to_numeric(merged_data['Quantity Ordered']) #Conve

merged_data['Price Each'] = pd.to_numeric(merged_data['Price Each']) #Convert to float
```

In [36]:
```
#Add a sales column

merged_data['Sales_Data'] = merged_data['Quantity Ordered'] * merged_data['Price Each']

merged_data.head()
```

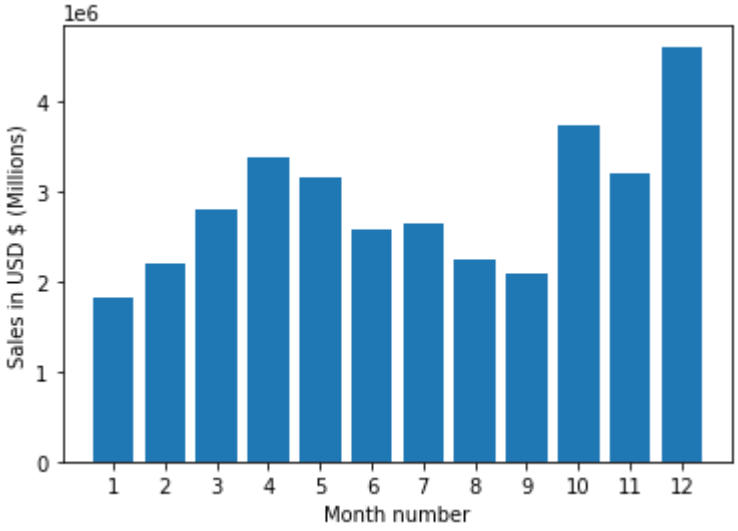
Out[36]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)

Question 1. What was the best month for sales and how much was earned that month?

In [37]:
```
results = merged_data.groupby('Month').sum()
```

In [38]:
```
#using matplotlib lets visualize the data

months = range(1,13) #months in the year

plt.bar(months, results['Sales_Data'])
plt.xticks(months)
plt.xlabel('Month number')
plt.ylabel('Sales in USD $ (Millions)')

plt.show()
```



Question 2. What city had the highest number of sales?

- Add a city column

In [39]:

```
#Let's use the .aply() method, use lambda to disect the cell contents

#or you can create a function
def get_city(address):

    return address.split(',')[1]

def get_state(address):

    return address.split(',')[2].split(' ')[1]

#Not really efficient for thousands of data since we are using a custom function
merged_data['City'] = merged_data['Purchase Address'].apply(lambda x: f"{get_city(x)}

merged_data.head()
```

Out[39]:

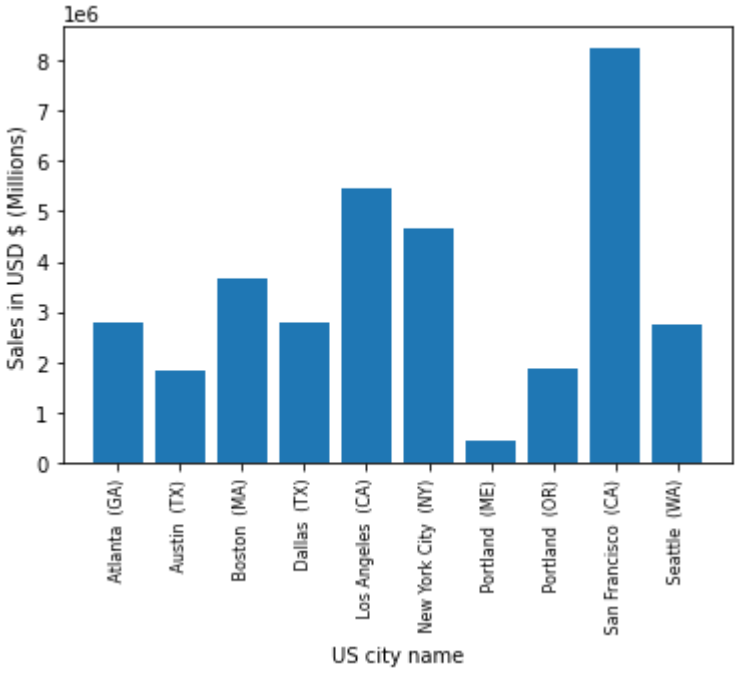
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)

In [40]:
```
results2 = merged_data.groupby('City').sum()
results2
```

Out[40]:

City	Quantity Ordered	Price Each	Month	Sales_Data
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

In [46]:
```
#unique method gives all values in a column

cities = [city for city, df in merged_data.groupby('City')] #Fixes order with respect t

plt.bar(cities, results2['Sales_Data'])
plt.xticks(cities, rotation = 'vertical', size=8)
plt.xlabel('US city name')
plt.ylabel('Sales in USD $ (Millions)')

plt.show()
```



Resons why this can be the case

- Christmas holiday sales
- San Francisco may have higher GDP per capita
- Silicon Valley uses more electronics relative to the products sold by this company

Question 3: What time should we display advertisements to maximize the likelihood of customers buying products?

In [48]:
```
merged_data

#use datetime library to extract the date
merged_data['Order Date'] = pd.to_datetime(merged_data['Order Date'])
```

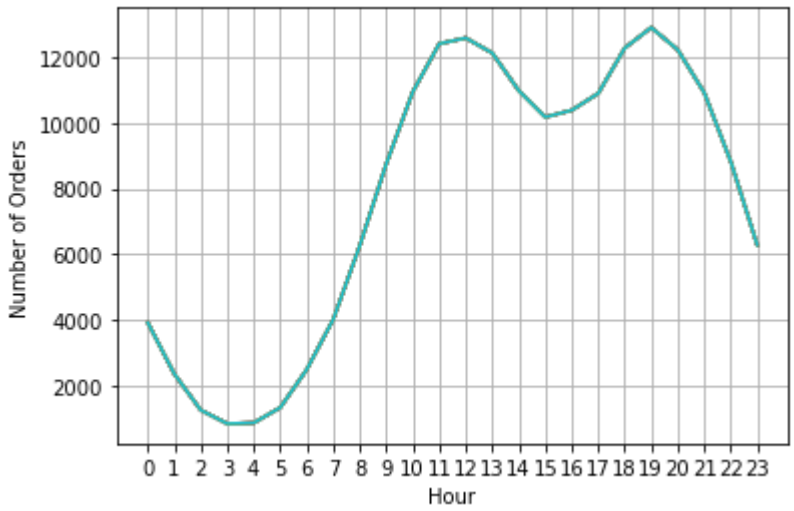
In [50]:
```
#Add Hour and minute of purchase using datetime as dt

merged_data['Hour'] = merged_data['Order Date'].dt.hour

merged_data['Minute'] = merged_data['Order Date'].dt.minute
```

Plot dates

In [57]:
```
#Number of occurences for products at the specific hour
hour = [hour for hour, df in merged_data.groupby('Hour')]

plt.plot(hour, merged_data.groupby(['Hour']).count())
plt.xticks(hour)
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.grid()


merged_data.groupby(['Hour']).count().head()
```

Out[57]:

Hour	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City	Minute
0	3910	3910	3910	3910	3910	3910	3910	3910	3910	3910
1	2350	2350	2350	2350	2350	2350	2350	2350	2350	2350
2	1243	1243	1243	1243	1243	1243	1243	1243	1243	1243
3	831	831	831	831	831	831	831	831	831	831
4	854	854	854	854	854	854	854	854	854	854



- Peak time are at 11:00am and 7:00pm, this would be a good time to place an ad
- This data corresponds to all the orders in every city, a possibility is to break down the data and find best products sold for that specific city

Question 4: What products are most often sold together?

In [59]:
```
#Which products were sold together the most, use duplicates from
# ...the Product column

merged_data.head()
```

Out[59]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales_Data	City	Hour	Minut
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)	8	4
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)	22	3
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	14	3
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	14	3
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	9	2

In [64]:
```
#Create new dataframe, keep=False results in keeping all duplicates
df = merged_data[merged_data['Order ID'].duplicated(keep=False)]

#merge duplicates together
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

```
<ipython-input-64-41de8908c585>: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: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

In [67]:
```
df = df[['Order ID', 'Grouped']].drop_duplicates()
df.head(10)
```

Out[67]:

	Order ID	Grouped
3	176560	Google Phone,Wired Headphones
18	176574	Google Phone,USB-C Charging Cable
30	176585	Bose SoundSport Headphones,Bose SoundSport Hea...
32	176586	AAA Batteries (4-pack),Google Phone
119	176672	Lightning Charging Cable,USB-C Charging Cable
129	176681	Apple Airpods Headphones,ThinkPad Laptop
138	176689	Bose SoundSport Headphones,AAA Batteries (4-pack)
189	176739	34in Ultrawide Monitor,Google Phone
225	176774	Lightning Charging Cable,USB-C Charging Cable
233	176781	iPhone,Lightning Charging Cable

In [72]:
```
from itertools import combinations
from collections import Counter

count = Counter()

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

for key, value in count.most_common(10):
    print(key, value)
#Most sold: {('iPhone', 'Lightning Charging Cable'): 1005}
```

```
('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
```

Question 5: what product sold the most? Why do you think it sold the most?

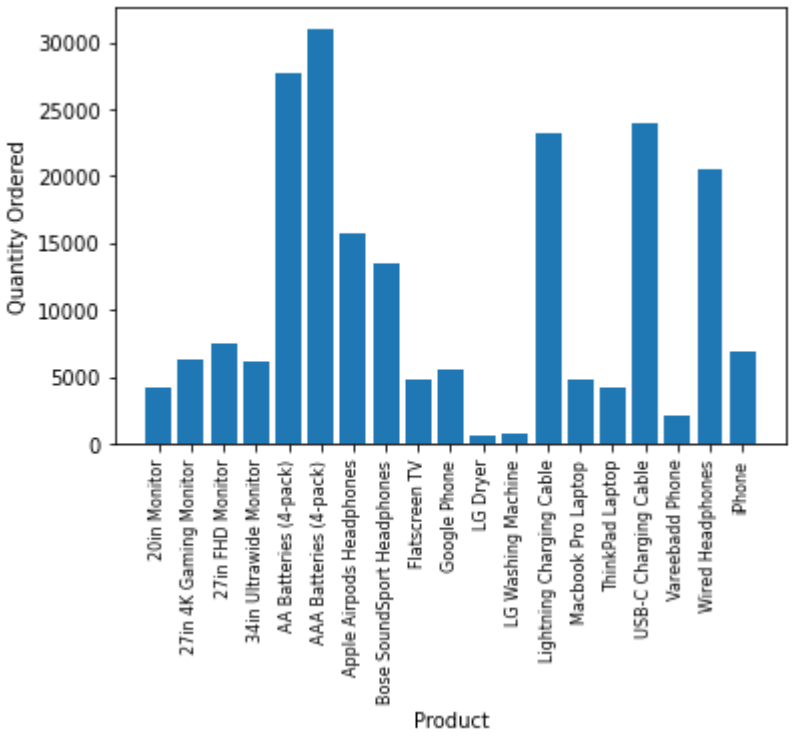
In [73]:
```
# Sum Quantity Ordred based on the Product
product_group = merged_data.groupby('Product')

product_group.sum()
```

Out[73]:

Product	Quantity Ordered	Price Each	Month	Sales_Data	Hour	Minute
20in Monitor	4129	451068.99	29336	454148.71	58764	122252
27in 4K Gaming Monitor	6244	2429637.70	44440	2435097.56	90916	184331
27in FHD Monitor	7550	1125974.93	52558	1132424.50	107540	219948
34in Ultrawide Monitor	6199	2348718.19	43304	2355558.01	89076	183480
AA Batteries (4-pack)	27635	79015.68	145558	106118.40	298342	609039
AAA Batteries (4-pack)	31017	61716.59	146370	92740.83	297332	612113
Apple Airpods Headphones	15661	2332350.00	109477	2349150.00	223304	455570
Bose SoundSport Headphones	13457	1332366.75	94113	1345565.43	192445	392603
Flatscreen TV	4819	1440000.00	34224	1445700.00	68815	142789
Google Phone	5532	3315000.00	38305	3319200.00	79479	162773
LG Dryer	646	387600.00	4383	387600.00	9326	19043
LG Washing Machine	666	399600.00	4523	399600.00	9785	19462
Lightning Charging Cable	23217	323787.10	153092	347094.15	312529	634442
Macbook Pro Laptop	4728	8030800.00	33548	8037600.00	68261	137574
ThinkPad Laptop	4130	4127958.72	28950	4129958.70	59746	121508
USB-C Charging Cable	23975	261740.85	154819	286501.25	314645	647586
Vareebadd Phone	2068	826000.00	14309	827200.00	29472	61835
Wired Headphones	20557	226395.18	133397	246478.43	271720	554023
iPhone	6849	4789400.00	47941	4794300.00	98657	201688

In [80]:
```
quantity_ordered = product_group.sum()['Quantity Ordered']

products = [product for product, df in product_group]

plt.bar(products, quantity_ordered)
```

```
plt.ylabel('Quantity Ordered')
plt.xlabel("Product")
plt.xticks(products, rotation='vertical', size=8)

plt.show()
```



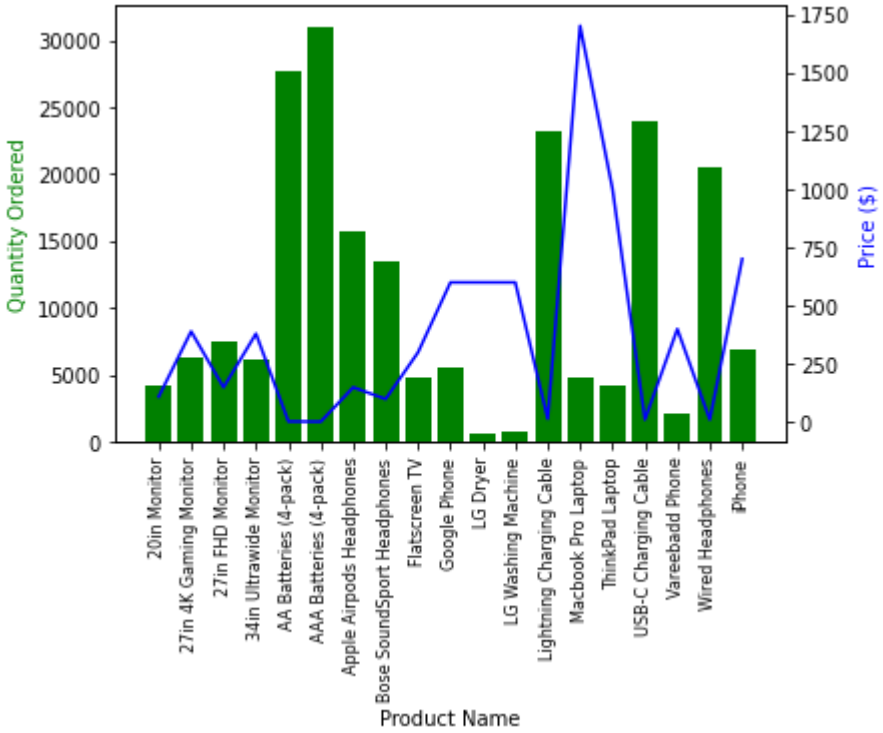
In [88]:

```
#Determine why products are sold the most or sold the least
#With the help of stackoverflow...

prices = merged_data.groupby('Product').mean()['Price Each']
print(prices)

fig, ax1 = plt.subplots()

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

ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered', color='g')
ax2.set_ylabel('Price ($)', color='b')

ax1.set_xticklabels(products, rotation='vertical', size=8)

plt.show()
```

```
Product
20in Monitor                  109.99
27in 4K Gaming Monitor        389.99
27in FHD Monitor              149.99
34in Ultrawide Monitor        379.99
AA Batteries (4-pack)           3.84
AAA Batteries (4-pack)          2.99
Apple Airpods Headphones      150.00
Bose SoundSport Headphones     99.99
```

```
Flatscreen TV                 300.00
Google Phone                  600.00
LG Dryer                      600.00
LG Washing Machine            600.00
Lightning Charging Cable       14.95
Macbook Pro Laptop           1700.00
ThinkPad Laptop               999.99
USB-C Charging Cable           11.95
Vareebadd Phone               400.00
Wired Headphones               11.99
iPhone                        700.00
Name: Price Each, dtype: float64
```

```
<ipython-input-88-4afee4b8d532>:17: UserWarning: FixedFormatter should only be used toge
ther with FixedLocator
  ax1.set_xticklabels(products, rotation='vertical', size=8)
```



- When quantity ordered is high, price tends to be low (vice versa)
- Outliers do exist however

In [ ]: