analysis

June 26, 2023

Import Necessary Libraries

[7]:
```
import pandas as pd
import os
```

Task 1 : Merge the 12 months of sales data into a single CSV file

[9]:
```
df = pd.read_csv("./Sales_Data/Sales_April_2019.csv")

files = [file for file in os.listdir('./Sales_Data')]

all_months_data = pd.DataFrame()

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

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

Read in updated dataframe

[10]:
```
all_data= pd.read_csv("all_data.csv")
all_data.head()
```

[10]:

	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
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

Cleaning the data

Drop rows of NaN

[14]:
```
all_data = all_data.dropna(how='all')
all_data.head()
```

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

Find 'Or' and Delete it

[15]:
```
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']
```

[ ]:
```

```

Convert colomns to the correct type

[32]:
```
all_data['Quantity Ordered'] = all_data['Quantity Ordered'].astype('float')
all_data['Price Each'] = all_data['Price Each'].astype('float')
all_data.head()
```

[32]:
```
  Order ID                    Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable               2.0       11.95
2   176559  Bose SoundSport Headphones               1.0       99.99
3   176560                Google Phone               1.0      600.00
4   176560            Wired Headphones               1.0       11.99
5   176561            Wired Headphones               1.0       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

   Qunatity Ordered
0               2.0
2               1.0
3               1.0
4               1.0
```

```
5               1.0
```

[31]:
```
type(all_data['Price Each'])
```

[31]:
```
pandas.core.series.Series
```

Augment data with additional Colomns

Task 2 : Add Month Colomn

[16]:
```
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
```

[16]:
```
  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
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
```

Task 3 : Add a sales colomn

[33]:
```
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
all_data.head()
```

[33]:
```
  Order ID                    Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable               2.0       11.95
2   176559  Bose SoundSport Headphones               1.0       99.99
3   176560                Google Phone               1.0      600.00
4   176560            Wired Headphones               1.0       11.99
5   176561            Wired Headphones               1.0       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

   Qunatity Ordered   Sales
0               2.0   23.90
```

```
2              1.0   99.99
3              1.0  600.00
4              1.0   11.99
5              1.0   11.99
```

Task 4 : Add a city colomn

[45]:
```
def get_city(address):
    return address.split(',')[1]
def get_state(address):
    return address.split(',')[2].split(' ')[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x: get_city(x) + '␣
↪(' + get_state(x) + ')')

all_data.head()
```

[45]:
```
  Order ID                    Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable               2.0       11.95
2   176559  Bose SoundSport Headphones               1.0       99.99
3   176560                Google Phone               1.0      600.00
4   176560            Wired Headphones               1.0       11.99
5   176561            Wired Headphones               1.0       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

   Qunatity Ordered   Sales              City
0               2.0   23.90        Dallas (TX)
2               1.0   99.99        Boston (MA)
3               1.0  600.00   Los Angeles (CA)
4               1.0   11.99   Los Angeles (CA)
5               1.0   11.99   Los Angeles (CA)
```

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

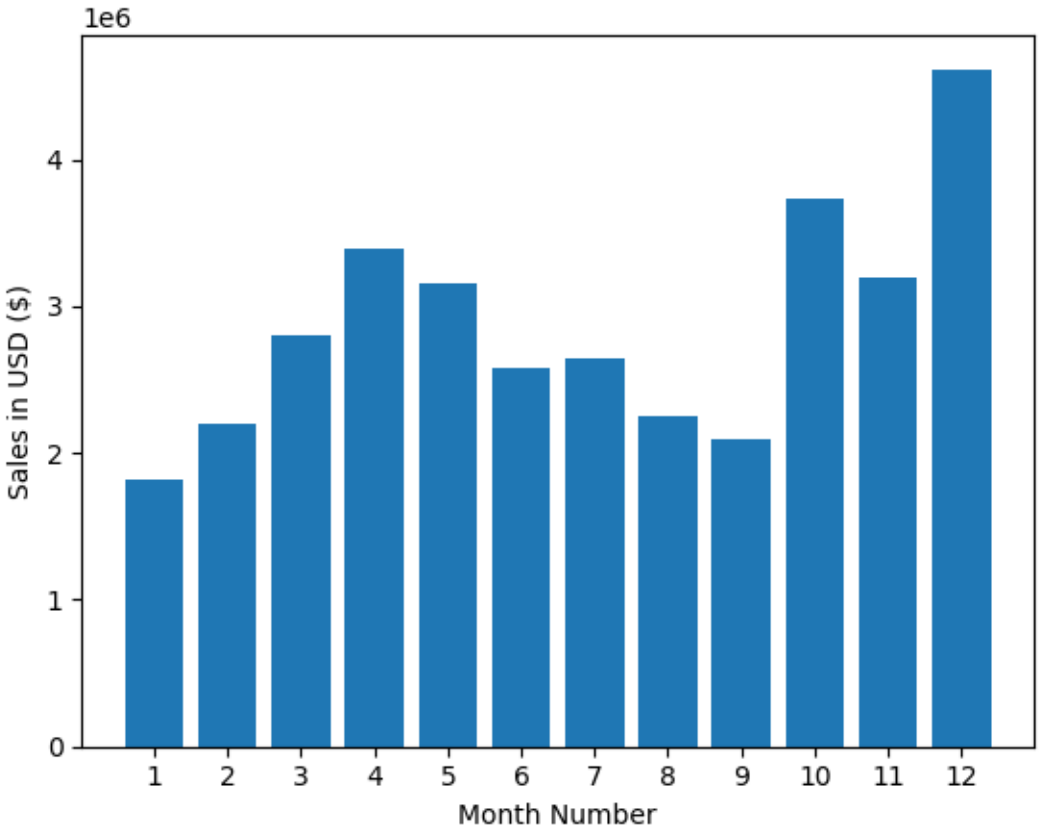
[49]:
```
results = all_data.groupby('Month').sum()
results
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_15000\894706187.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a
future version, numeric_only will default to False. Either specify numeric_only
or select only columns which should be valid for the function.
  results = all_data.groupby('Month').sum()
```

[49]:

Month	Quantity Ordered	Price Each	Qunatity Ordered	Sales
1	10903.0	1811768.38	10903.0	1822256.73
2	13449.0	2188884.72	13449.0	2202022.42
3	17005.0	2791207.83	17005.0	2807100.38
4	20558.0	3367671.02	20558.0	3390670.24
5	18667.0	3135125.13	18667.0	3152606.75
6	15253.0	2562025.61	15253.0	2577802.26
7	16072.0	2632539.56	16072.0	2647775.76
8	13448.0	2230345.42	13448.0	2244467.88
9	13109.0	2084992.09	13109.0	2097560.13
10	22703.0	3715554.83	22703.0	3736726.88
11	19798.0	3180600.68	19798.0	3199603.20
12	28114.0	4588415.41	28114.0	4613443.34

[51]:
```
import matplotlib.pyplot as plt

months = range(1,13)

plt.bar(months, results['Sales'])
plt.xticks(months)
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month Number')
plt.show()
```



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

[58]:
```
results = all_data.groupby('City').sum()
results
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_15000\2386508471.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a
future version, numeric_only will default to False. Either specify numeric_only
or select only columns which should be valid for the function.
  results = all_data.groupby('City').sum()
```

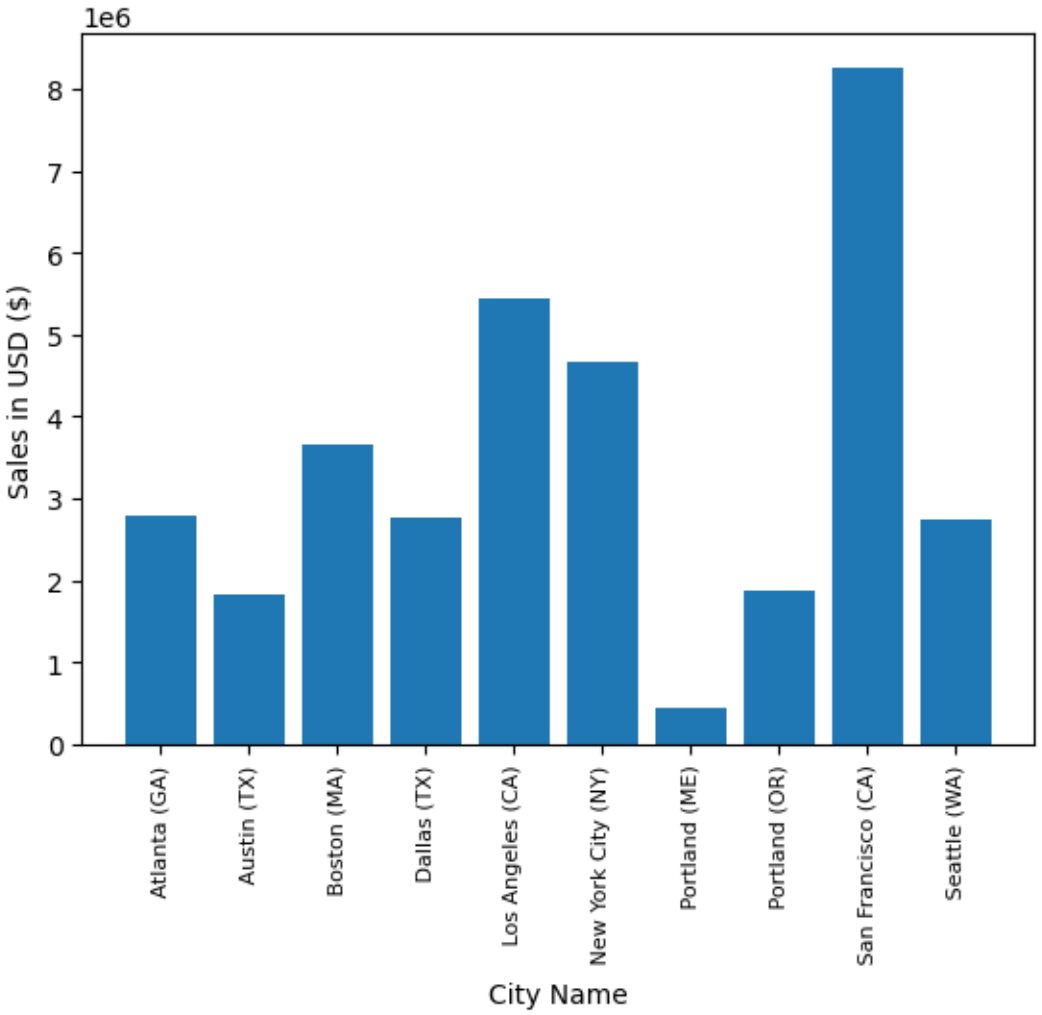
[58]:

City	Quantity Ordered	Price Each	Month	Qunatity Ordered
Atlanta (GA)	16602.0	2779908.20	104794	16602.0
Austin (TX)	11153.0	1809873.61	69829	11153.0
Boston (MA)	22528.0	3637409.77	141112	22528.0
Dallas (TX)	16730.0	2752627.82	104620	16730.0
Los Angeles (CA)	33289.0	5421435.23	208325	33289.0
New York City (NY)	27932.0	4635370.83	175741	27932.0
Portland (ME)	2750.0	447189.25	17144	2750.0
Portland (OR)	11303.0	1860558.22	70621	11303.0

```
 San Francisco (CA)           50239.0  8211461.74  315520           50239.0
 Seattle (WA)                 16553.0  2733296.01  104941           16553.0
```

```
                          Sales
City
 Atlanta (GA)        2795498.58
 Austin (TX)         1819581.75
 Boston (MA)         3661642.01
 Dallas (TX)         2767975.40
 Los Angeles (CA)    5452570.80
 New York City (NY)  4664317.43
 Portland (ME)        449758.27
 Portland (OR)       1870732.34
 San Francisco (CA)  8262203.91
 Seattle (WA)        2747755.48
```

[61]:
```
import matplotlib.pyplot as plt

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

plt.bar(cities, results['Sales'])
plt.xticks(cities, rotation = 'vertical', size=8)
plt.ylabel('Sales in USD ($)')
plt.xlabel('City Name')
plt.show()
```



Qustion 3 : What time should we display advertisments to maximise likelihood of cutomer's buying product?

[62]:
```
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

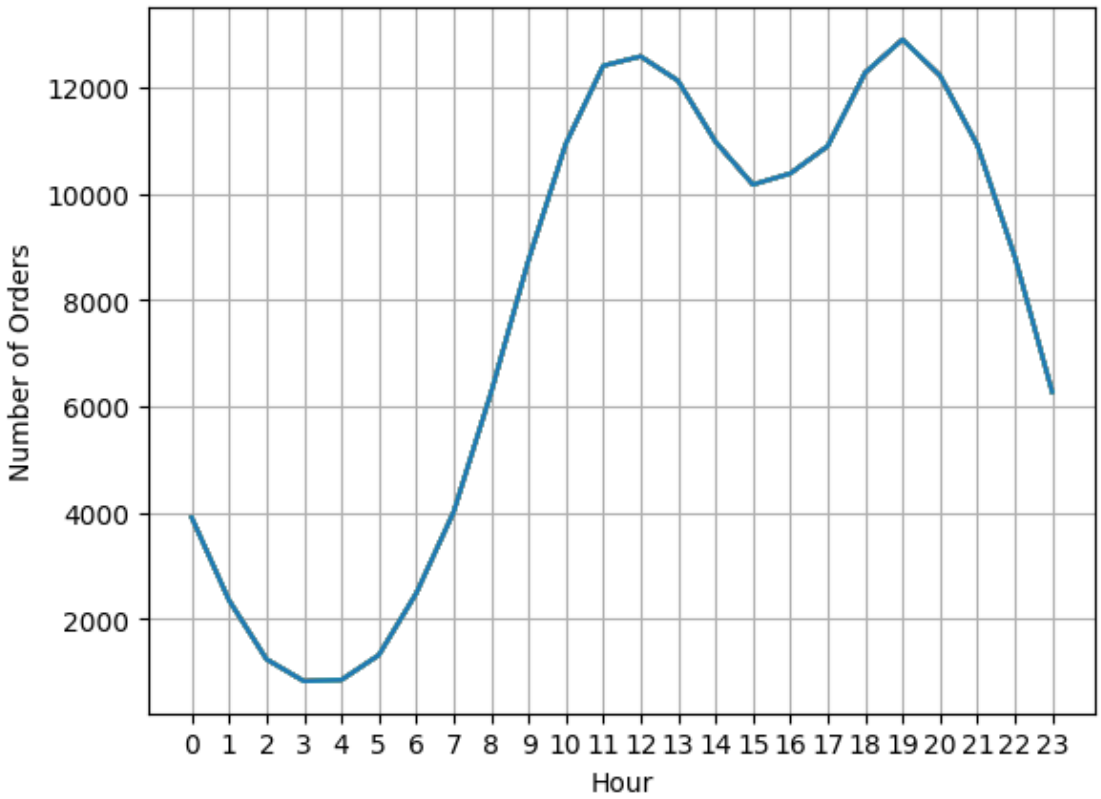
[65]:
```
all_data['Hour'] = all_data['Order Date'].dt.hour
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data.head()
```

[65]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2.0	11.95	
2	176559	Bose SoundSport Headphones	1.0	99.99	
3	176560	Google Phone	1.0	600.00	
4	176560	Wired Headphones	1.0	11.99	
5	176561	Wired Headphones	1.0	11.99	

```
           Order Date                      Purchase Address  Month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   Qunatity Ordered   Sales              City  Hour  Minute
0               2.0   23.90        Dallas (TX)     8      46
2               1.0   99.99        Boston (MA)    22      30
3               1.0  600.00   Los Angeles (CA)    14      38
4               1.0   11.99   Los Angeles (CA)    14      38
5               1.0   11.99   Los Angeles (CA)     9      27
```

[67]:
```
hours = [hour for hour, df in all_data.groupby('Hour')]

plt.plot(hours, all_data.groupby(['Hour']).count())
plt.xticks(hours)
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.grid()
plt.show()
```



Question 4 : What products are most often sold together?

[72]:
```
df = all_data[all_data['Order ID'].duplicated(keep=False)]

df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x : ','.
 ↪join(x))

df = df[['Order ID', 'Grouped']].drop_duplicates()

df.head()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_15000\119640104.py:3:
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))
```

[72]:

	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

[75]:
```
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)))

count.most_common(10)
```

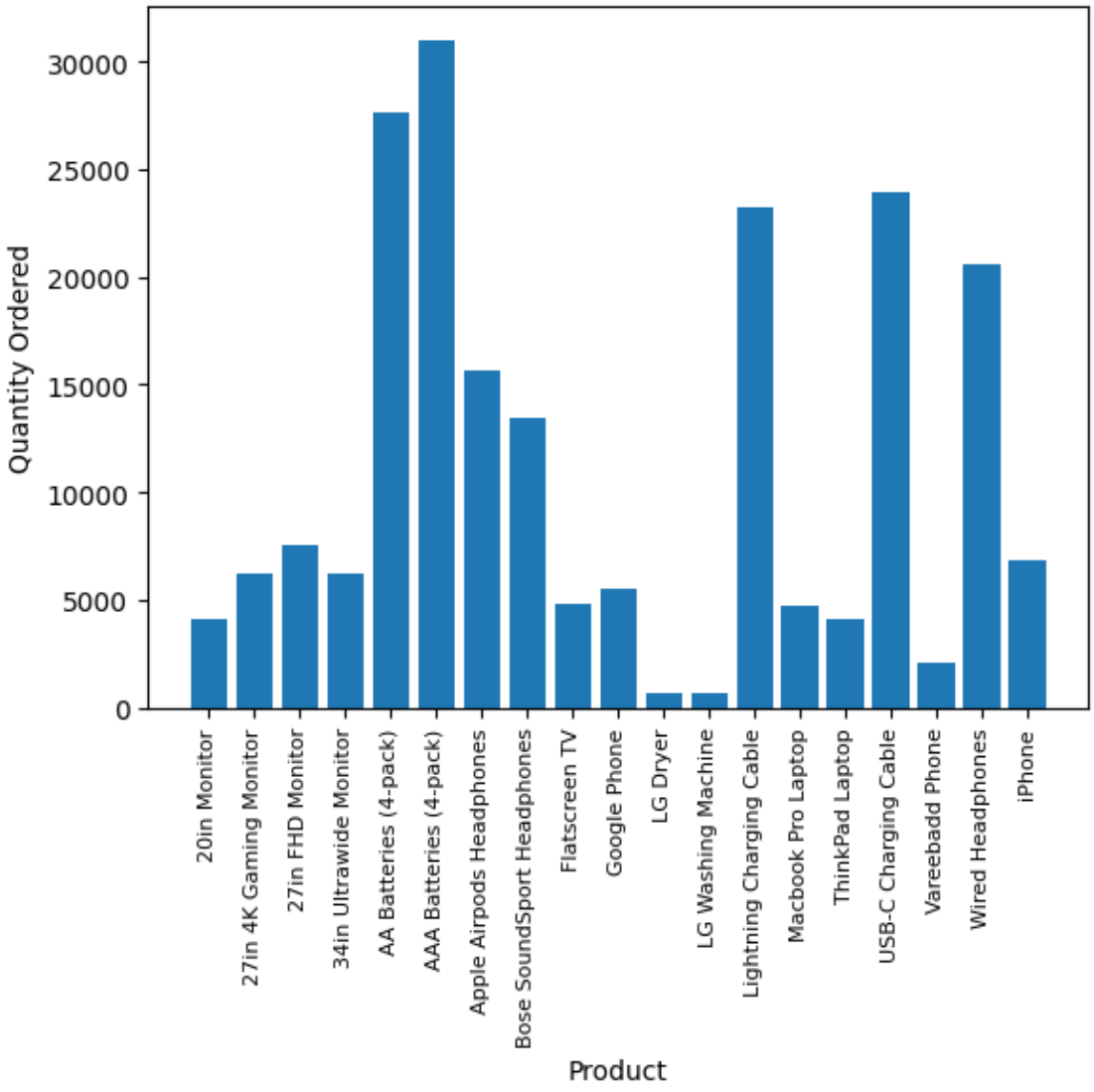
[75]:
```
[(('iPhone', 'Lightning Charging Cable'), 1005),
 (('Google Phone', 'USB-C Charging Cable'), 987),
 (('iPhone', 'Wired Headphones'), 447),
 (('Google Phone', 'Wired Headphones'), 414),
 (('Vareebadd Phone', 'USB-C Charging Cable'), 361),
 (('iPhone', 'Apple Airpods Headphones'), 360),
 (('Google 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?

[79]:
```
product_group = all_data.groupby('Product')
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()
```

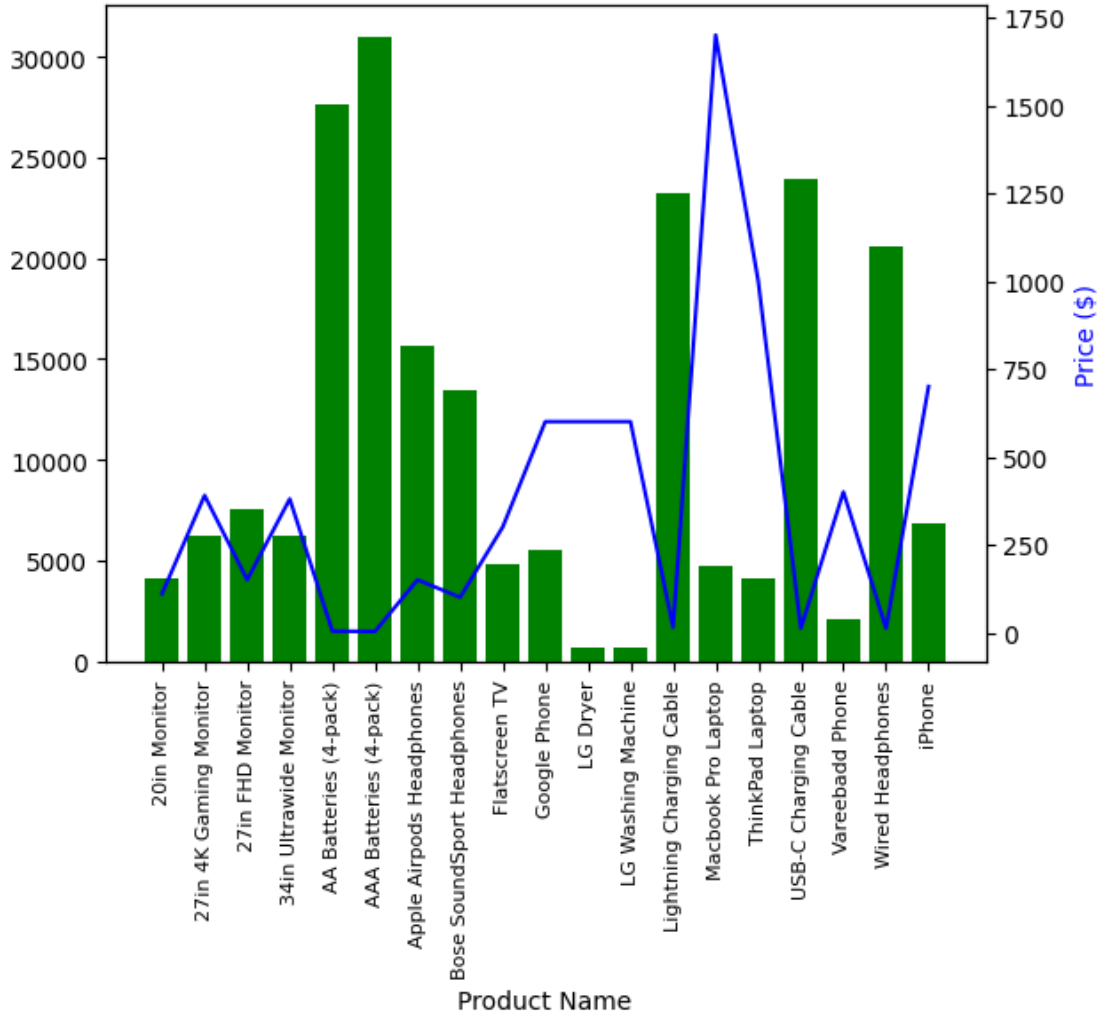
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_15000\1070783197.py:2: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a
future version, numeric_only will default to False. Either specify numeric_only
or select only columns which should be valid for the function.
  quantity_ordered = product_group.sum()['Quantity Ordered']
```



[81]:
```
prices = all_data.groupby('Product').mean()['Price Each']

fig, ax1 = plt.subplots()

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

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

```
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_15000\296489534.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a
future version, numeric_only will default to False. Either specify numeric_only
or select only columns which should be valid for the function.
  prices = all_data.groupby('Product').mean()['Price Each']
C:\Users\DELL\AppData\Local\Temp\ipykernel_15000\296489534.py:12: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(products, rotation = 'vertical', size = 8)
```



[ ]: