import necessary libraries

In [5]:
```
import pandas as pd
import os
```

merging 12 month sale data into single file

In [6]:
```
df= pd.read_csv("./Sales_Data/Sales_April_2019.csv")
files=[file for file in os.listdir('./Sales_Data')]
all_month_data=pd.DataFrame()
for file in files:
    df= pd.read_csv("./Sales_Data/"+file)
    all_month_data=pd.concat([all_month_data, df])

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

In [ ]:

In [ ]:

Task1:read in update dataframe

In [7]:
```
all_data=pd.read_csv("all_data.csv")
all_data.head()
```

Out[7]:

	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

clean up the data!

while performing the operation we find the error .Based on that error , we can clean up the data (from my point of view)

drop rows of NAN

In [9]:
```
nan_df=all_data[all_data.isna().any(axis=1)]
nan_df.head()
all_data=all_data.dropna(how='all')
all_data.head()
```

Out[9]:

	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
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
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

find 'or 'and delete it

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

#convert column to the correct type

In [19]:
```
all_data['Quantity Ordered']=pd.to_numeric(all_data['Quantity Ordered']) #make
all_data['Price Each']=pd.to_numeric(all_data['Price Each']) #make float
```

```
Augment data with additional column
```

Task2- add month column

In [20]:
```
all_data['Month']=all_data['Order Date'].str[0:2]
all_data['Month']=all_data['Month'].astype('int32')
all_data.head()
```

Out[20]:

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

In [ ]:
```
```

In [ ]:
```
```

add sales column

In [21]:
```
all_data['Sales']=all_data['Quantity Ordered']*all_data['Price Each']
all_data.head()
```

Out[21]:

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

add a city column

In [39]:
```
#let's use .apply()
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: f"{get_city(x)}

all_data.head()
```

Out[39]:

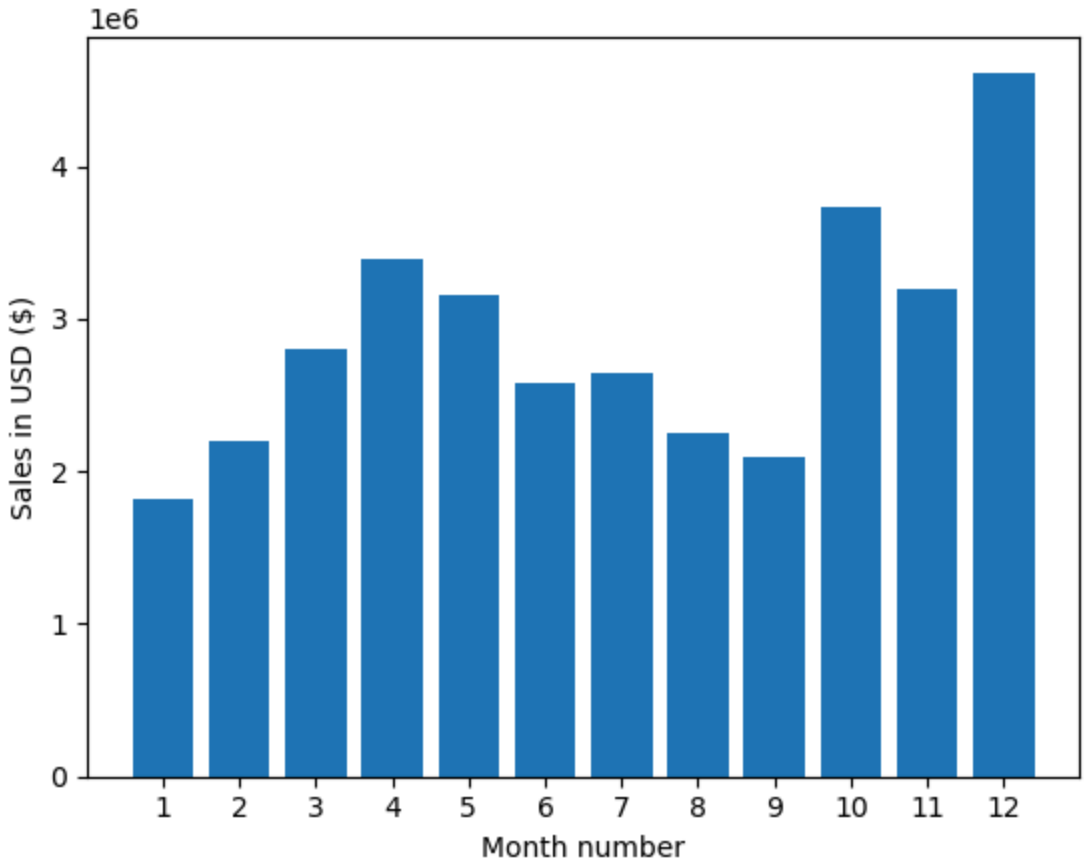
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
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)

Q-1. what was the best month for sale? How much was earned that month?

In [23]:
```
results=all_data.groupby('Month').sum()
```

```
C:\Users\Pradeep Ahir\AppData\Local\Temp\ipykernel_3664\3809692125.py:1: Futu
reWarning: The default value of numeric_only in DataFrameGroupBy.sum is depre
cated. In a future version, numeric_only will default to False. Either specif
y numeric_only or select only columns which should be valid for the function.
  results=all_data.groupby('Month').sum()
```

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



what city has the highest number of sale?

In [41]:
```
results=all_data.groupby('City').sum()
results
```

```
C:\Users\Pradeep Ahir\AppData\Local\Temp\ipykernel_3664\3338049859.py:1: Futu
reWarning: The default value of numeric_only in DataFrameGroupBy.sum is depre
cated. In a future version, numeric_only will default to False. Either specif
y numeric_only or select only columns which should be valid for the function.
  results=all_data.groupby('City').sum()
```

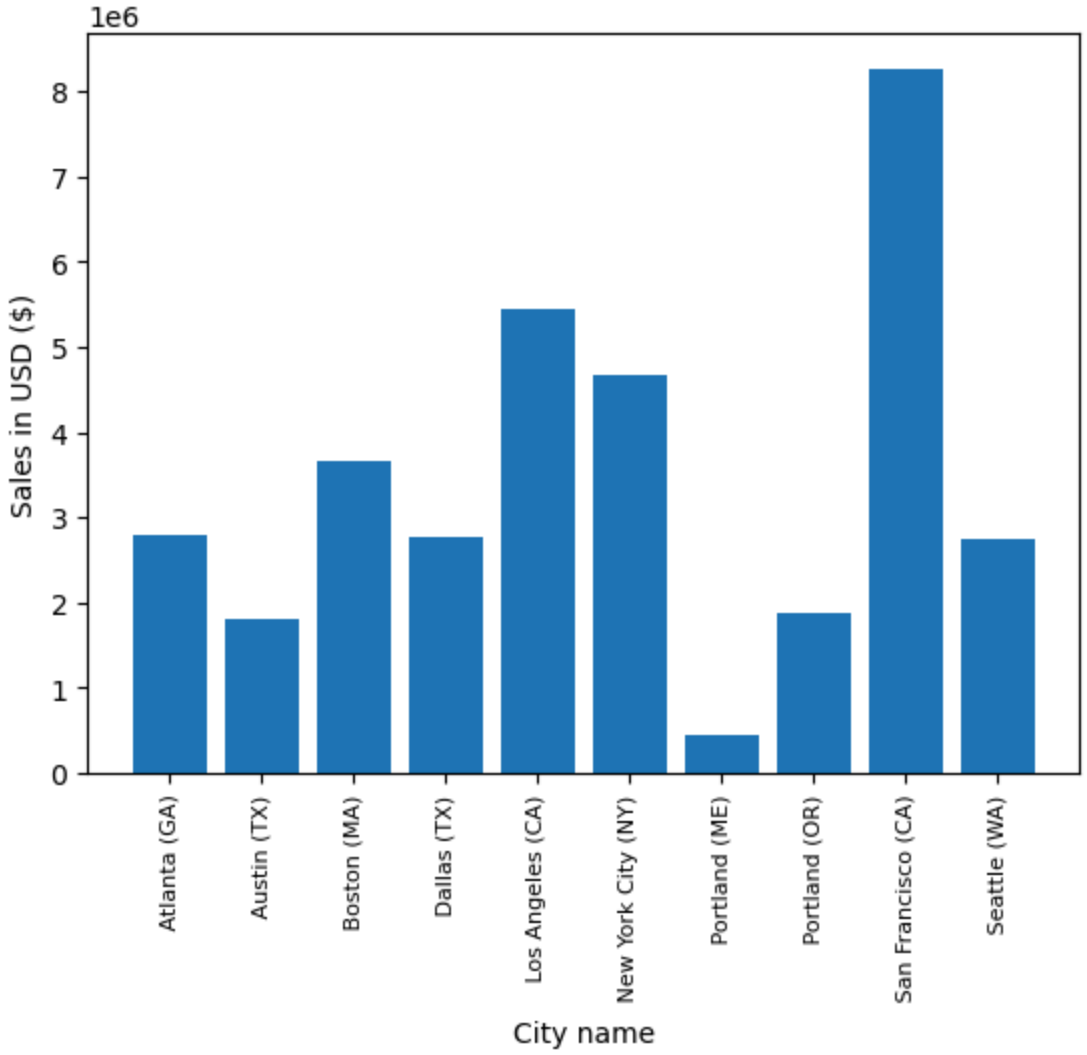
Out[41]:

City	Quantity Ordered	Price Each	Month	Sales
Atlanta (GA)	16602	2779908.20	104794	2795498.58
Austin (TX)	11153	1809873.61	69829	1819581.75
Boston (MA)	22528	3637409.77	141112	3661642.01
Dallas (TX)	16730	2752627.82	104620	2767975.40
Los Angeles (CA)	33289	5421435.23	208325	5452570.80
New York City (NY)	27932	4635370.83	175741	4664317.43
Portland (ME)	2750	447189.25	17144	449758.27
Portland (OR)	11303	1860558.22	70621	1870732.34
San Francisco (CA)	50239	8211461.74	315520	8262203.91
Seattle (WA)	16553	2733296.01	104941	2747755.48

In [45]:
```
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()
```



what time should we display advertisement to maximize likelihood of customer's buying product?

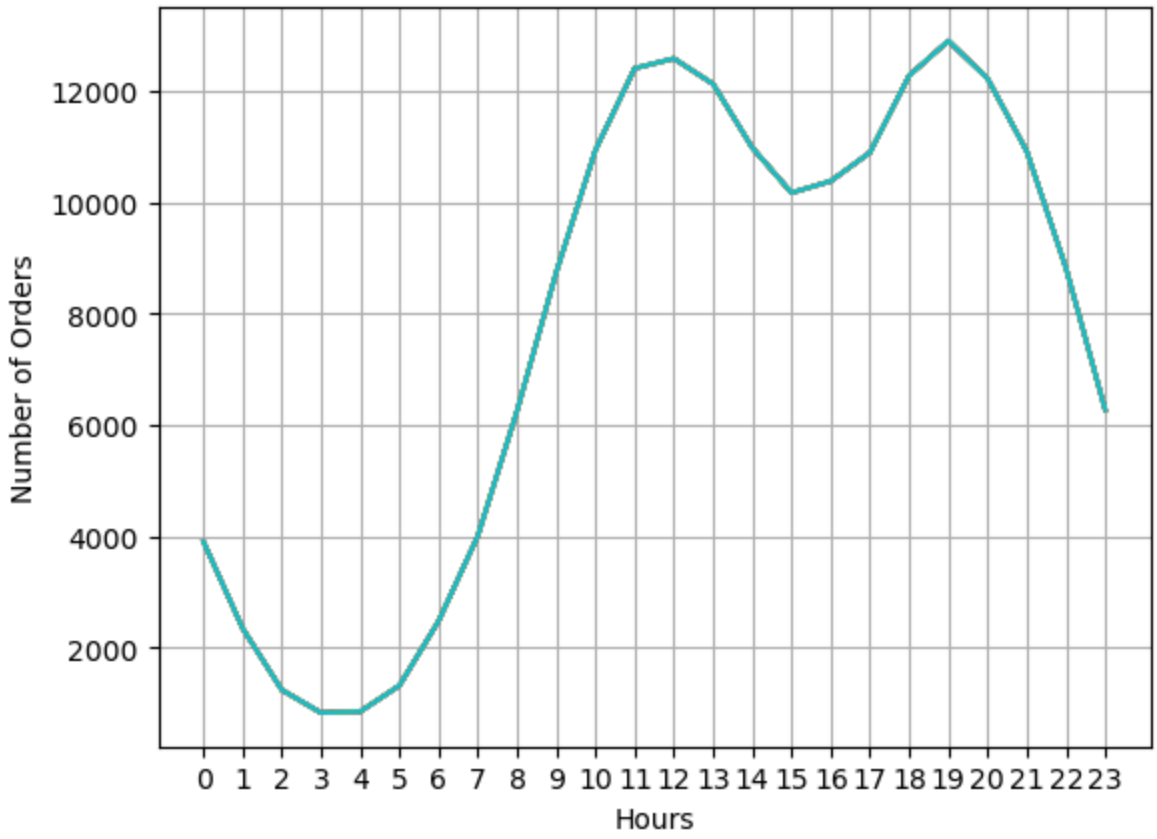
In [47]:
```
all_data['Order Date']=pd.to_datetime(all_data['Order Date'])
all_data['Hour']= all_data['Order Date'].dt.hour
all_data['Minute']=all_data['Order Date'].dt.minute
all_data.head()
```

Out[47]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Min
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	
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	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	
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	
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	

In [53]:
```
hours=[hour for hour ,df in all_data.groupby('Hour')]
plt.plot(hours, all_data.groupby(['Hour']).count())
plt.xticks(hours)
plt.xlabel('Hours')
plt.ylabel('Number of Orders')
plt.grid()
plt.show()

#my recommendation to display advertizement around(11am) or 7pm(19) to maximize
```



what products are most often sold together?

In [58]:
```
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()     #to remove duplicate
df.head()
```

```
C:\Users\Pradeep Ahir\AppData\Local\Temp\ipykernel_3664\1093302264.py:2: Sett
ingWithCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x: ','.joi
n(x))
```

Out[58]:

	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

In [60]:
```
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)))     #here 2 is the item ar
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
```

Q-5 what product sold the most ? why do you think it sold the most?

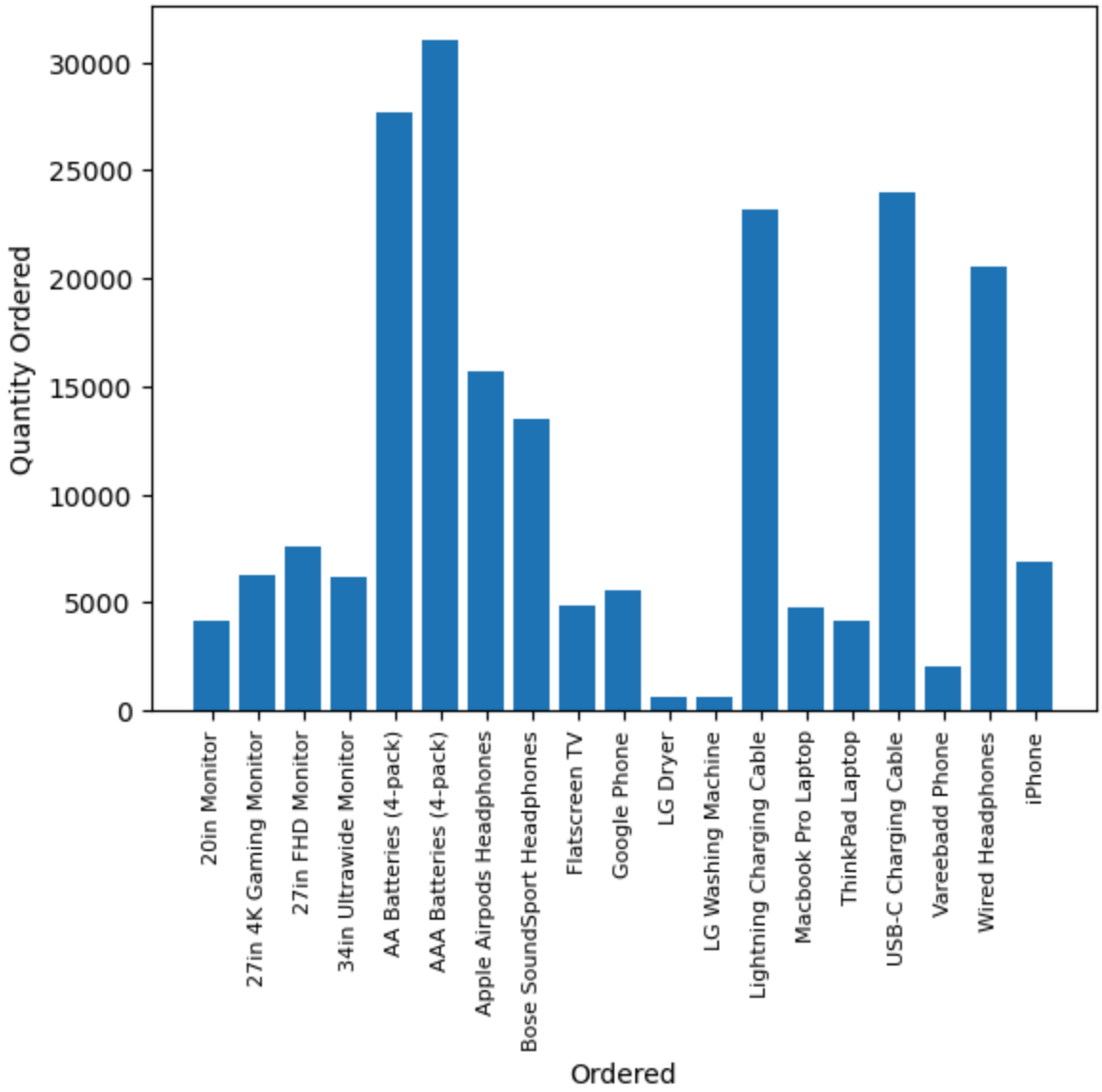
In [61]:
```
all_data.head()
```

Out[61]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Min
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	
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	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	
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	
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	

In [65]:
```
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('Ordered')
plt.xticks(products, rotation='vertical', size=8)
plt.show()
```

```
C:\Users\Pradeep Ahir\AppData\Local\Temp\ipykernel_3664\1983284032.py:2: Futu
reWarning: The default value of numeric_only in DataFrameGroupBy.sum is depre
cated. In a future version, numeric_only will default to False. Either specif
y numeric_only or select only columns which should be valid for the function.
  quantity_ordered=product_group.sum()['Quantity Ordered']
```



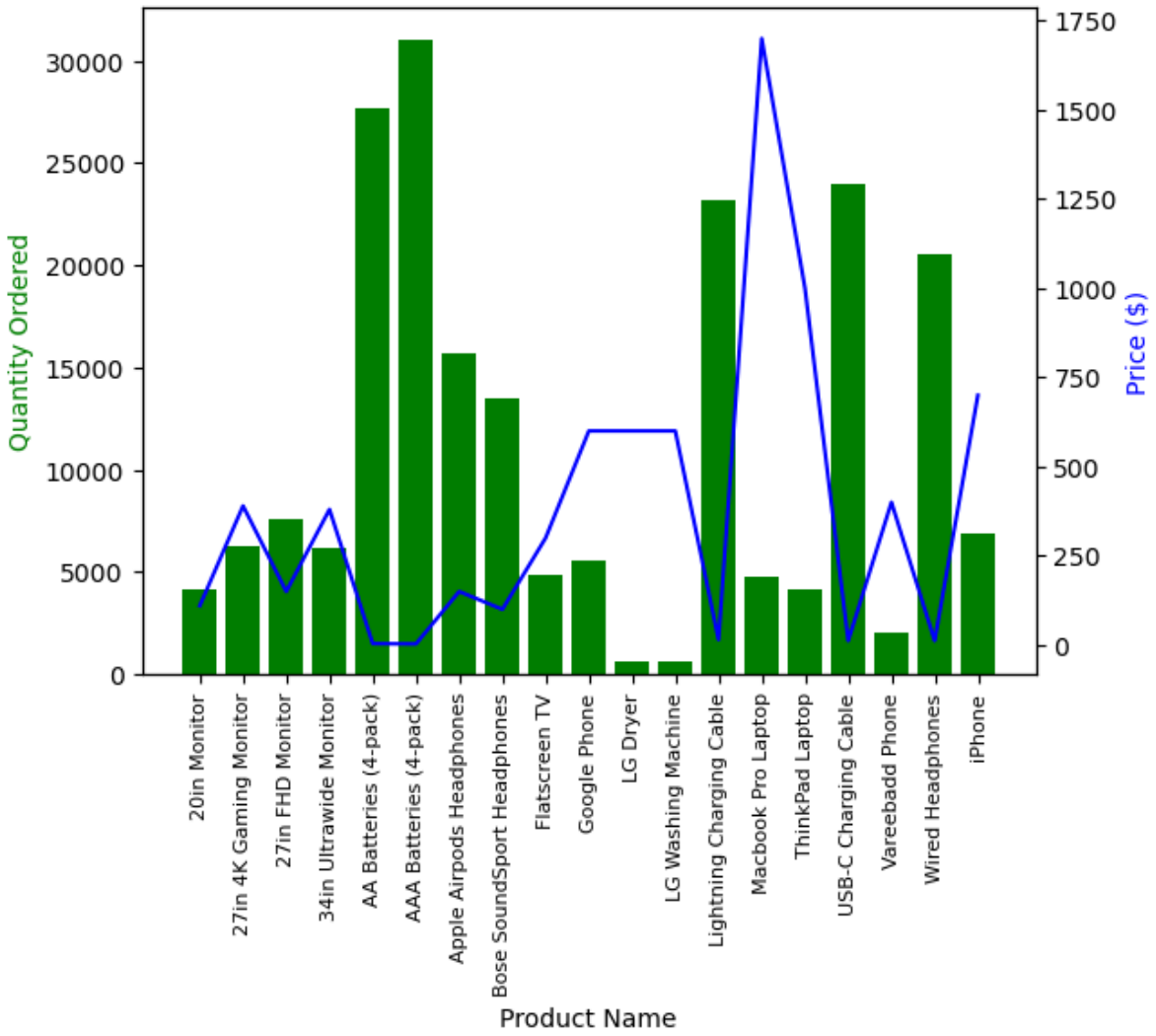
In [70]:
```
prices=all_data.groupby('Product').mean()['Price Each'] # just google how to pl

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()
```

```
C:\Users\Pradeep Ahir\AppData\Local\Temp\ipykernel_3664\1175560429.py:1: Futu
reWarning: The default value of numeric_only in DataFrameGroupBy.mean is depr
ecated. In a future version, numeric_only will default to False. Either speci
fy numeric_only or select only columns which should be valid for the functio
n.
  prices=all_data.groupby('Product').mean()['Price Each'] # just google how t
o plot second y axis
C:\Users\Pradeep Ahir\AppData\Local\Temp\ipykernel_3664\1175560429.py:12: Use
rWarning: FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(products, rotation='vertical', size=8)
```


Quantity Ordered
Price ($)
Product Name
0
5000
10000
15000
20000
25000
30000
0
250
500
750
1000
1250
1500
1750
20in Monitor
27in 4K Gaming Monitor
27in FHD Monitor
34in Ultrawide Monitor
AA Batteries (4-pack)
AAA Batteries (4-pack)
Apple Airpods Headphones
Bose SoundSport Headphones
Flatscreen TV
Google Phone
LG Dryer
LG Washing Machine
Lightning Charging Cable
Macbook Pro Laptop
ThinkPad Laptop
USB-C Charging Cable
Vareebadd Phone
Wired Headphones
iPhone

In [ ]: