Import necessary libraries

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

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

In [ ]:
```
df = pd.read_csv("/home/owekitiibwa/Desktop/dataAnalysis/Pandas-Data-Scie

files = [file for file in os.listdir('/home/owekitiibwa/Desktop/dataAnaly
all_months_data = pd.DataFrame()

for file in files:
    df = pd.read_csv("/home/owekitiibwa/Desktop/dataAnalysis/Pandas-Data-
    all_months_data = pd.concat([all_months_data,df])
all_months_data.to_csv("all_data.csv",index=False)
```

Read in updated dataframe

In [ ]:
```
all_data = pd.read_csv("/home/owekitiibwa/Desktop/dataAnalysis/Pandas-Dat
all_data.head()
```

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101

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

Cleaning Up the data

Drop rows of NaN

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

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

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101

Find 'Or' and delete it

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

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101

Augment data with additional colums

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

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5

Add column for Sales

Convert columns to the correct type

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

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5

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

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5	11.99
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5	3.84
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5	149.99
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5	11.99
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5	5.98

Best Month for Sales

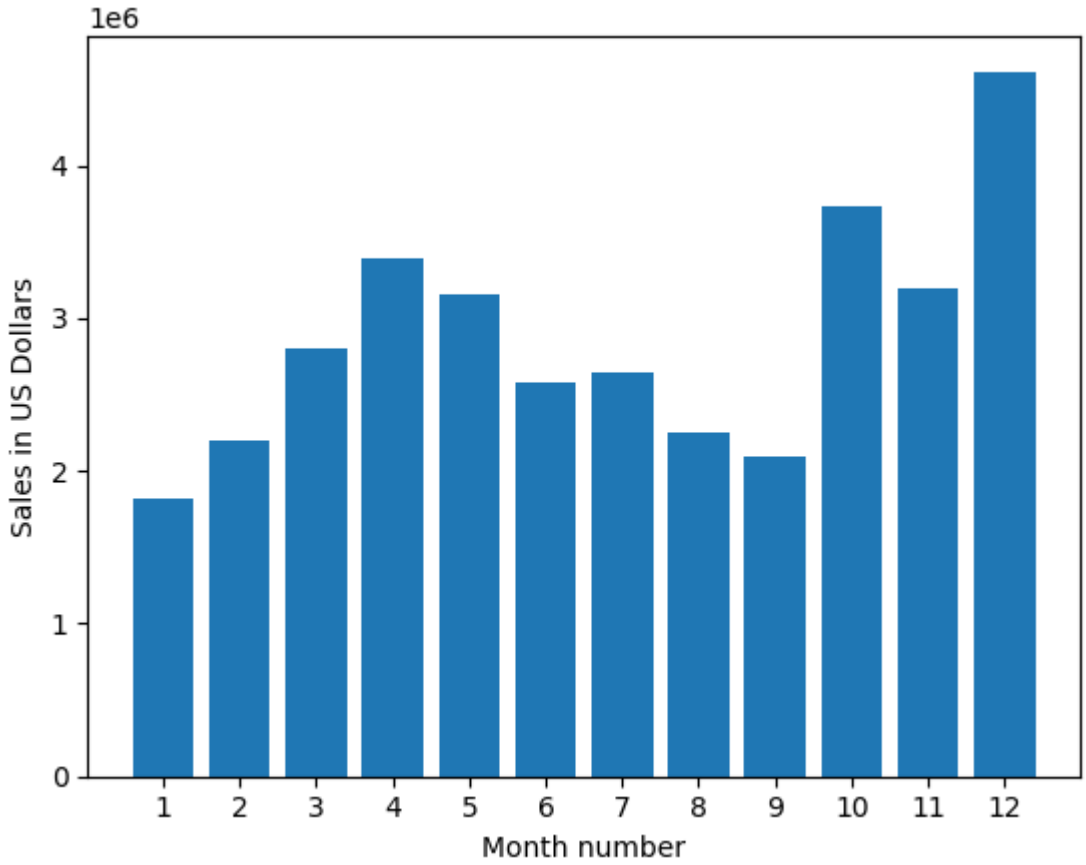
In [ ]:
```
results = all_data.groupby('Month').sum()['Sales']
results
```

Out[ ]:
```
Month
1     1822256.73
2     2202022.42
3     2807100.38
4     3390670.24
5     3152606.75
6     2577802.26
7     2647775.76
8     2244467.88
9     2097560.13
10    3736726.88
11    3199603.20
12    4613443.34
Name: Sales, dtype: float64
```

Data Visualization using a bar graph

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

months = range(1,13)
plt.bar(months, results)
plt.xticks(months)
plt.ylabel("Sales in US Dollars")
plt.xlabel('Month number')
plt.show()
```



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

Task 1: Add a city column

In [ ]:
```
#Let's use apply()
def get_state(address):
    return address.split(',')[2].split(" ")[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x:f"{x.split
all_data.head()
```

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5	11.99	New York City (NY)
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5	3.84	Dallas (TX)
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5	149.99	New York City (NY)
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5	11.99	Dallas (TX)
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5	5.98	Seattle (WA)

Task 2 : Group by City

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

Out[ ]:
```
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
Name: Sales, dtype: float64
```

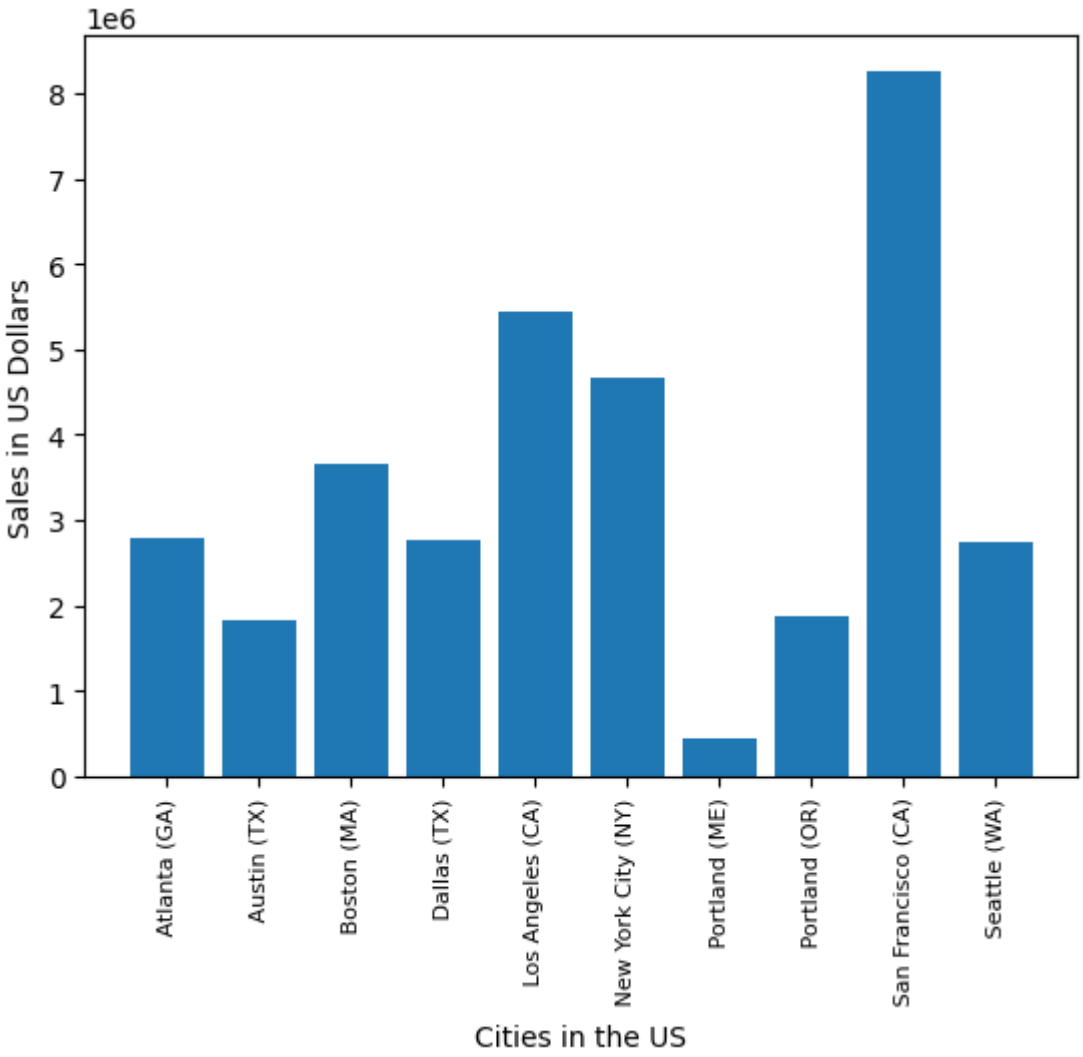
Plotting a bar graph for City Sales

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

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

plt.bar(cities, results)
plt.xticks(cities, rotation = "vertical",size=8 )
plt.ylabel("Sales in US Dollars")
```

```
plt.xlabel('Cities in the US')
plt.show()
```



Question 3: What time should we display advertisements to maximise likelihood of customer's buying product ?

Change the Order Date to hours and minutes

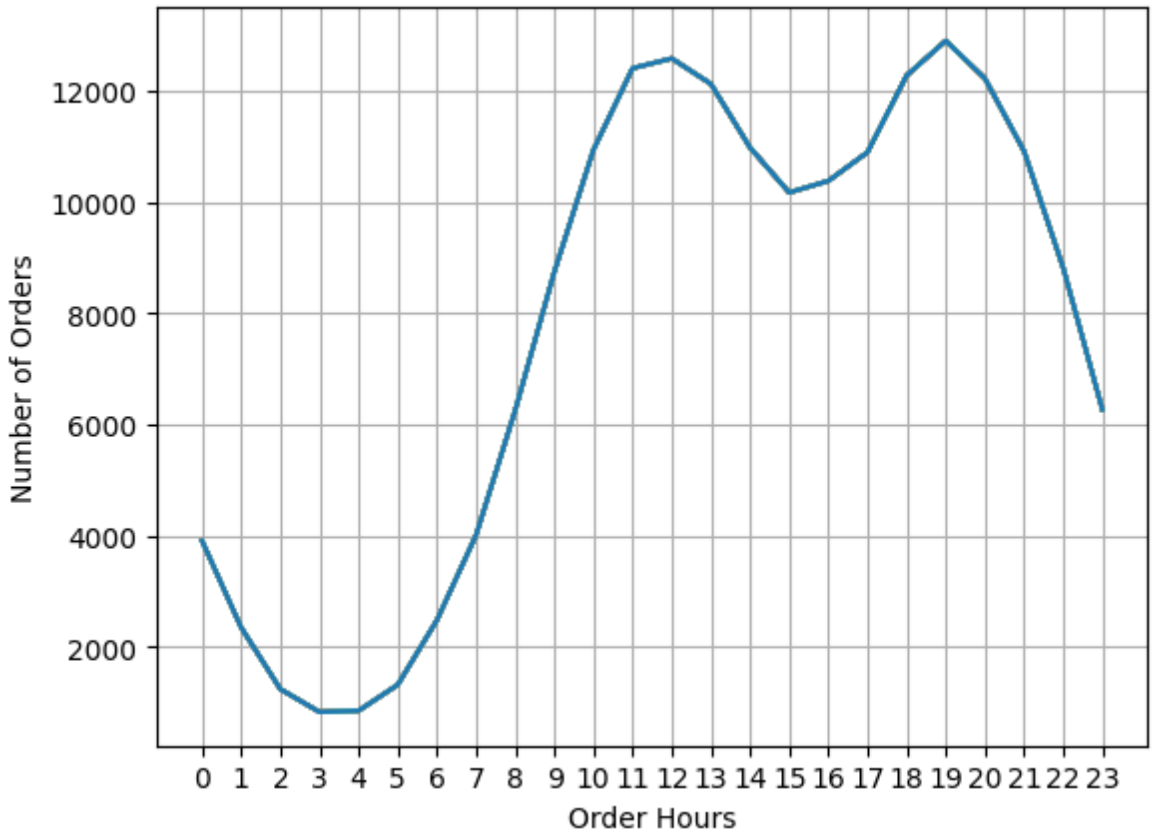
In [ ]:
```
all_data["Order Date"] = pd.to_datetime(all_data["Order Date"])
all_data["hour"] = all_data["Order Date"].dt.hour
all_data["Minute"] = all_data["Order Date"].dt.minute
all_data["Count"] = 1
all_data.head()
```

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	194095	Wired Headphones	1	11.99	2019-05-16 17:14:00	669 2nd St, New York City, NY 10001	5	11.99	New York City (NY)
1	194096	AA Batteries (4-pack)	1	3.84	2019-05-19 14:43:00	844 Walnut St, Dallas, TX 75001	5	3.84	Dallas (TX)
2	194097	27in FHD Monitor	1	149.99	2019-05-24 11:36:00	164 Madison St, New York City, NY 10001	5	149.99	New York City (NY)
3	194098	Wired Headphones	1	11.99	2019-05-02 20:40:00	622 Meadow St, Dallas, TX 75001	5	11.99	Dallas (TX)
4	194099	AAA Batteries (4-pack)	2	2.99	2019-05-11 22:55:00	17 Church St, Seattle, WA 98101	5	5.98	Seattle (WA)

In [ ]:

```
hours = [hour for hour,df in all_data.groupby('hour')]
plt.plot(hours,all_data.groupby(['hour']).count())
plt.xticks(hours)
plt.xlabel("Order Hours")
plt.ylabel("Number of Orders")
plt.grid()
plt.show()
#My recommendation is
```



Question 4: What products are most often sold together

In [ ]:
```
df = all_data[all_data["Order ID"].duplicated(keep=False)]
df["Grouped"] = df.groupby('Order ID')["Product"].transform(lambda x:','.
df = df[["Order ID","Grouped"]].drop_duplicates()
df.head()
```

```
/tmp/ipykernel_172375/3407403923.py:2: 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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["Grouped"] = df.groupby('Order ID')["Product"].transform(lambda
x:','.join(x))
```

Out[ ]:

	Order ID	Grouped
15	194110	Google Phone,Wired Headphones
77	194170	Google Phone,USB-C Charging Cable
82	194174	iPhone,Lightning Charging Cable,Wired Headphones
89	194179	Flatscreen TV,AAA Batteries (4-pack)
103	194192	Wired Headphones,Bose SoundSport Headphones

Counting items that are sold together

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

```
for row in df["Grouped"]:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list,2)))
count.most_common(20)
```

Out[ ]:
```
[(('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),
 (('Lightning Charging Cable', 'Apple Airpods Headphones'), 81),
 (('Vareebadd Phone', 'Bose SoundSport Headphones'), 80),
 (('USB-C Charging Cable', 'Bose SoundSport Headphones'), 77),
 (('Apple Airpods Headphones', 'Wired Headphones'), 69),
 (('Lightning Charging Cable', 'USB-C Charging Cable'), 58),
 (('Lightning Charging Cable', 'AA Batteries (4-pack)'), 55),
 (('Lightning Charging Cable', 'Lightning Charging Cable'), 54),
 (('Bose SoundSport Headphones', 'Wired Headphones'), 53),
 (('AA Batteries (4-pack)', 'Lightning Charging Cable'), 51),
 (('AAA Batteries (4-pack)', 'USB-C Charging Cable'), 50)]
```

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

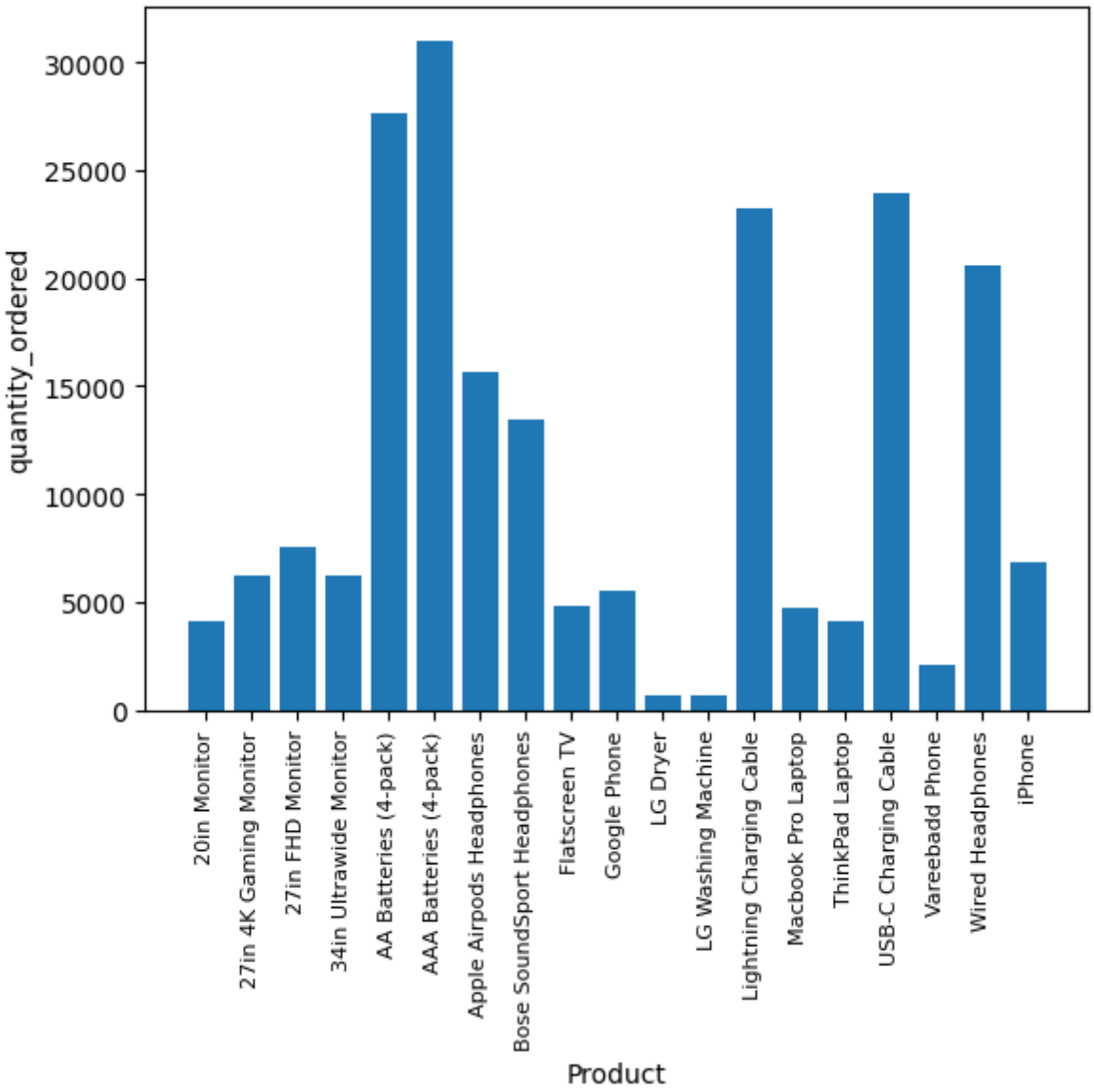
In [ ]:
```
all_data.head()
```

Out[ ]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5	11.99	New York City (NY)
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5	3.84	Dallas (TX)
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5	149.99	New York City (NY)
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5	11.99	Dallas (TX)
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5	5.98	Seattle (WA)

In [ ]:

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



In [ ]:
```
prices = product_group.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')
ax1.set_ylabel('Quantity Ordered',color='g')
ax2.set_ylabel('Price ($)', color='b')
ax1.set_xticklabels(products,rotation='vertical',size=8)

plt.show()
```

```
/tmp/ipykernel_172375/187488013.py:9: UserWarning: FixedFormatter should o
nly be used together with FixedLocator
  ax1.set_xticklabels(products,rotation='vertical',size=8)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js