In [1]:

```
import pandas as pd
import os
```

In [2]:

```
#merge whole year of sales data in a single CSV file
```

In [3]:

```
df=pd.read_csv("./Sales_data/Sales_April_2019.csv")

all_files=[f for f in os.listdir('./Sales_data')]

all_months_data=pd.DataFrame()

for file in all_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)

#So looks like we have concateneted all the csvs into a single one
```

UPDATED DATAFRAME

In [4]:

```
all_data=pd.read_csv('all_data.csv')
all_data.head(10)
```

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
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
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001
6	176562	USB-C Charging Cable	1	11.95	04/29/19 13:03	381 Wilson St, San Francisco, CA 94016
7	176563	Bose SoundSport Headphones	1	99.99	04/02/19 07:46	668 Center St, Seattle, WA 98101
8	176564	USB-C Charging Cable	1	11.95	04/12/19 10:58	790 Ridge St, Atlanta, GA 30301
9	176565	Macbook Pro Laptop	1	1700	04/24/19 10:38	915 Willow St, San Francisco, CA 94016

Before moving on to further data anaylsis, let's augment a few more columns

In [5]:

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

all_data.head()
```

Out[5]:

	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	04
1	NaN	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	04
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04

In [6]:

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

Out[6]:

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

In [7]:

```
#lets drop these
```

In [8]:

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

Out[8]:

	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	04
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

In [9]:

```
#all_data['Month']=all_data['Month'].astype('int32')

#all_data.head()
```

In [10]:

```
#FIND or and delete it
```

In [11]:

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

Out[11]:

	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	04
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

In [12]:

```
#OH OKAY
all_data['Month']=all_data['Month'].astype('int32')

all_data.head()
```

Out[12]:

	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	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 [13]:

```
#Okay so the month column is created and we converted it to int as well
```

AIM 1: Best month for sales, earned how much

In [14]:

```
#We need a sales column for that
#all_data['Sales']=all_data['Quantity Ordered']*all_data['Price Each']
#now before this , some columns actually look like numbers but are actually strings
```

Lets convert columns to their correct types

In [15]:

```
all_data['Quantity Ordered']=pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each']=pd.to_numeric(all_data['Price Each'])
all_data['Sales']=all_data['Quantity Ordered']*all_data['Price Each']
all_data.head(10)
```

Out[15]:

	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
6	176562	USB-C Charging Cable	1	11.95	04/29/19 13:03	381 Wilson St, San Francisco, CA 94016	4	11.95
7	176563	Bose SoundSport Headphones	1	99.99	04/02/19 07:46	668 Center St, Seattle, WA 98101	4	99.99
8	176564	USB-C Charging Cable	1	11.95	04/12/19 10:58	790 Ridge St, Atlanta, GA 30301	4	11.95
9	176565	Macbook Pro Laptop	1	1700.00	04/24/19 10:38	915 Willow St, San Francisco, CA 94016	4	1700.00
10	176566	Wired Headphones	1	11.99	04/08/19 14:05	83 7th St, Boston, MA 02215	4	11.99

In [16]:

```
#Okay so we have the month and sales column , let's answer , what was the best month
#and what were the sales
```

In [17]:

```
all_data.groupby('Month').sum()
```

Out[17]:

	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

In [18]:

```
results=all_data.groupby('Month').sum()
```

In [19]:

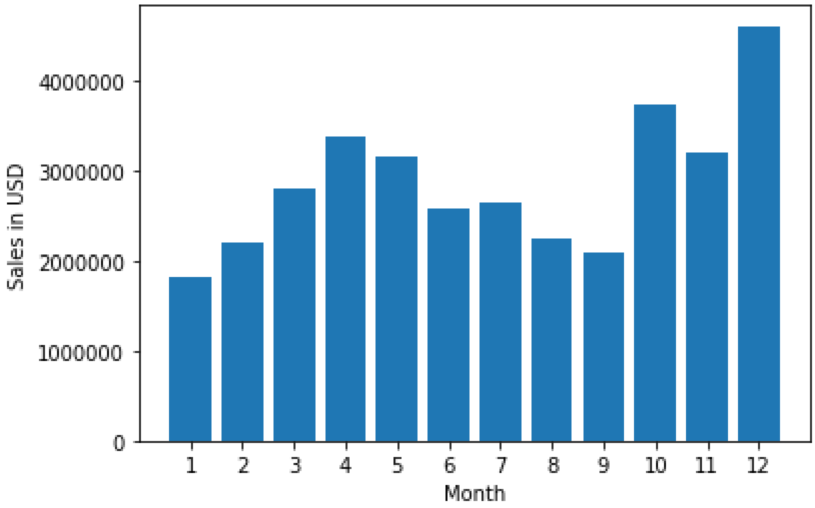
```
all_data[all_data['Sales']==all_data['Sales'].max()]
```

Out[19]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
4717	181069	Macbook Pro Laptop	2	1700.0	04/27/19 21:01	668 Park St, San Francisco, CA 94016	4	3400.0
5219	181544	Macbook Pro Laptop	2	1700.0	04/22/19 12:48	731 11th St, New York City, NY 10001	4	3400.0
92026	210292	Macbook Pro Laptop	2	1700.0	06/08/19 09:00	953 Ridge St, San Francisco, CA 94016	6	3400.0
127265	200528	Macbook Pro Laptop	2	1700.0	05/13/19 13:40	643 4th St, Boston, MA 02215	5	3400.0

In [20]:

```
#lets plot to visualise a bit more
import matplotlib.pyplot as plt
months=range(1,13)
y=results
plt.bar(months,results['Sales'])
plt.xticks(months)
plt.ylabel('Sales in USD')
plt.xlabel('Month')
plt.show()
```



AIM-2 : Which city had the highest number of sales

So we need a city column

In [21]:

```
#Let's use the .apply() method
def forcity(name):
    whole=name.split(",")
    return whole[1]
def forstate(name):
    whole=name.split(',')
    return whole[2][1:3]


#all_data['City']=all_data['Purchase Address'].apply(forcity)
all_data['City']=all_data['Purchase Address'].apply(lambda x:forcity(x) + " (" + forstate(x)+")")
```

In [22]:

```
all_data.head()
```

Out[22]:

	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)

In [23]:

```
result_city=all_data.groupby('City').sum()
result_city.head(10)
```

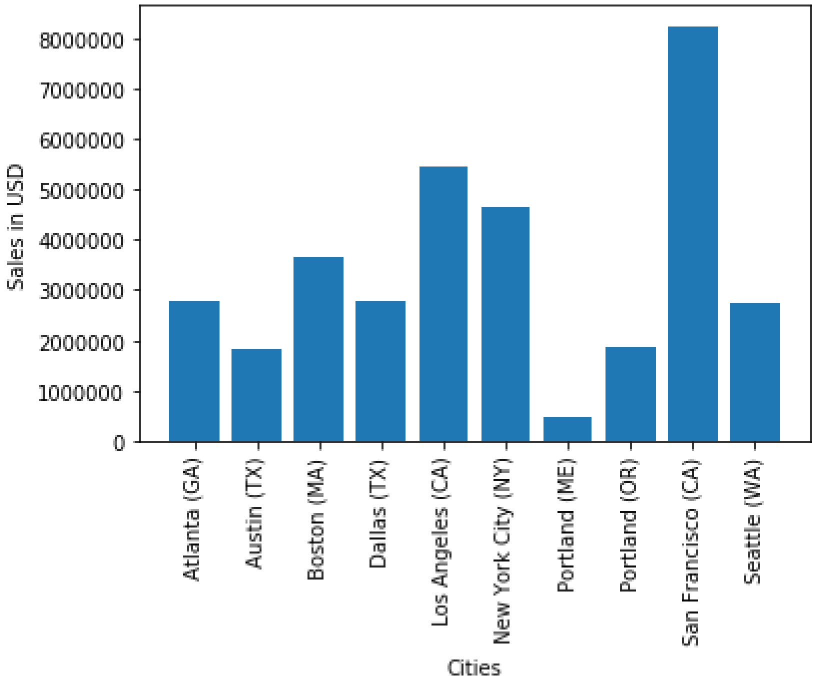
Out[23]:

	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

In [24]:

```
import matplotlib.pyplot as plt
#cities=all_data['City'].unique(), messed up order
cities=[city for city,df in all_data.groupby('City')]
plt.bar(cities,result_city['Sales'])
plt.xticks(cities,rotation='vertical',size=10)
plt.ylabel('Sales in USD')
plt.xlabel('Cities')

plt.show()
```



In [25]:

```
#Now why this discrepancy?? we clearly saw that san franciso had the highest sale,
#It has something to do with the order imposed of x by unique function
#X and Y are not in order so let's fix that cities=[city for city,df in all_data.groupby('City')]
```

What time should advertisements be displayed to maximise the liklihood of customer's buying product

In [26]:

```
all_data.head()
```

Out[26]:

	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)

In [27]:

```
#So we need to figure out a way to aggregate the order dates, into their distribution
#over  a 24-h period
```

In [28]:

```
#Best to create a date-time object rather than pciking and converting(parsing the strin
g)
```

In [29]:

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

In [30]:

```
all_data.head()
```

Out[30]:

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

In [31]:

```
all_data['Hour']=all_data['Order Date'].dt.hour
all_data['Minutes']=all_data['Order Date'].dt.minute
```

In [32]:

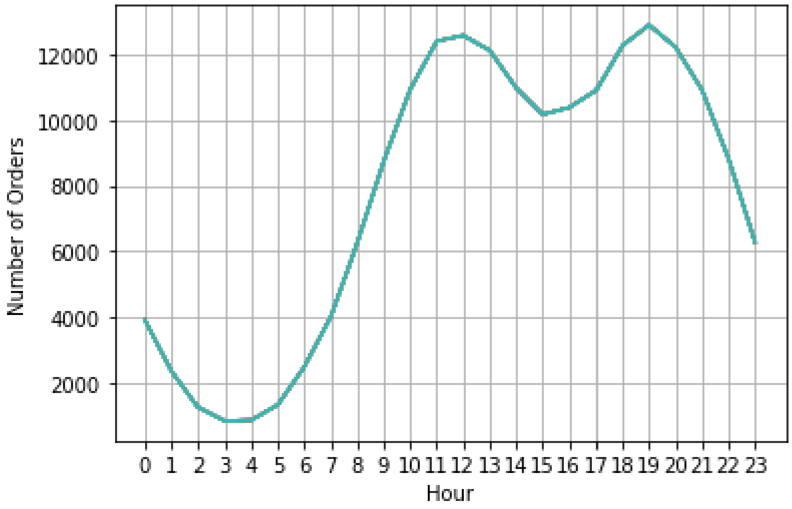
```
all_data.head()
```

Out[32]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour
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 [33]:

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



In [34]:

```
#So right before 11 am or right before 7pm might be a good time for ads
```

AIM : What products are often sold together??

In [35]:

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

In [36]:

```
df.head()
```

Out[36]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour
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
18	176574	Google Phone	1	600.00	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	19
19	176574	USB-C Charging Cable	1	11.95	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	11.95	Los Angeles (CA)	19
30	176585	Bose SoundSport Headphones	1	99.99	2019-04-07 11:31:00	823 Highland St, Boston, MA 02215	4	99.99	Boston (MA)	11

In [37]:

```
#Now let's try to get the products on the same line
```

In [38]:

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

```
C:\Users\anike\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Settin
gWithCopyWarning:
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
  """Entry point for launching an IPython kernel.
```

In [39]:

```
df.head()
```

Out[39]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour
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
18	176574	Google Phone	1	600.00	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	19
19	176574	USB-C Charging Cable	1	11.95	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	11.95	Los Angeles (CA)	19
30	176585	Bose SoundSport Headphones	1	99.99	2019-04-07 11:31:00	823 Highland St, Boston, MA 02215	4	99.99	Boston (MA)	11

In [40]:

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

In [41]:

```
df.head()
```

Out[41]:

	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 [42]:

```
#Now we have the count pairs of what occurs together most frequently
# I referred from stack overflow, a bit on the trickier side
```

In [43]:

```
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))) #change then number and we can get
 results for 3,4 etc items sold together

count.most_common(10)
for a,b in count.most_common(10) :
    print(a,b)
```

```
('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
```

AIM : What product is sold the most and why do think that is?

In [45]:

```
all_data.groupby('Product').sum()['Quantity Ordered']
```

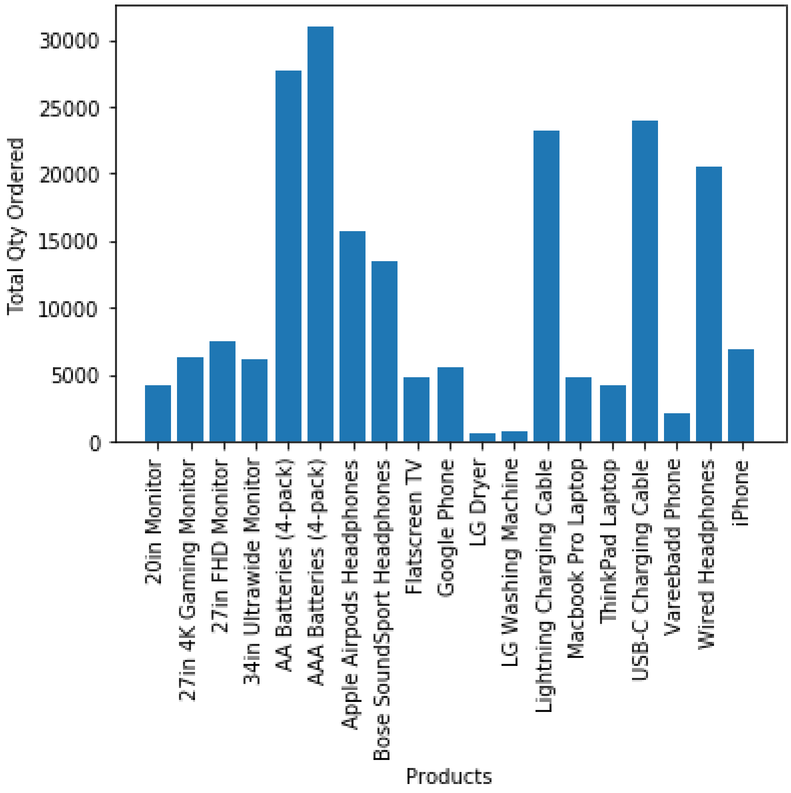
Out[45]:

```
Product
20in Monitor                   4129
27in 4K Gaming Monitor         6244
27in FHD Monitor               7550
34in Ultrawide Monitor         6199
AA Batteries (4-pack)         27635
AAA Batteries (4-pack)        31017
Apple Airpods Headphones      15661
Bose SoundSport Headphones    13457
Flatscreen TV                  4819
Google Phone                   5532
LG Dryer                        646
LG Washing Machine              666
Lightning Charging Cable      23217
Macbook Pro Laptop             4728
ThinkPad Laptop                4130
USB-C Charging Cable          23975
Vareebadd Phone                2068
Wired Headphones              20557
iPhone                         6849
Name: Quantity Ordered, dtype: int64
```

In [48]:

```
import matplotlib.pyplot as plt
#cities=all_data['City'].unique(), messed up order
prod_group=all_data.groupby('Product')
qty_order=all_data.groupby('Product').sum()['Quantity Ordered']
Products=[prod for prod,df in prod_group]
plt.bar(Products,qty_order)
plt.xticks(Products,rotation='vertical',size=10)
plt.ylabel('Total Qty Ordered')
plt.xlabel('Products')

plt.show()
```



In [50]:

```
#We can see and interpret a few things , but it is good to prove our hypothesis
#Let's overlay with prices
```

In [56]:

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

In [57]:

```
costs.head()
```

Out[57]:

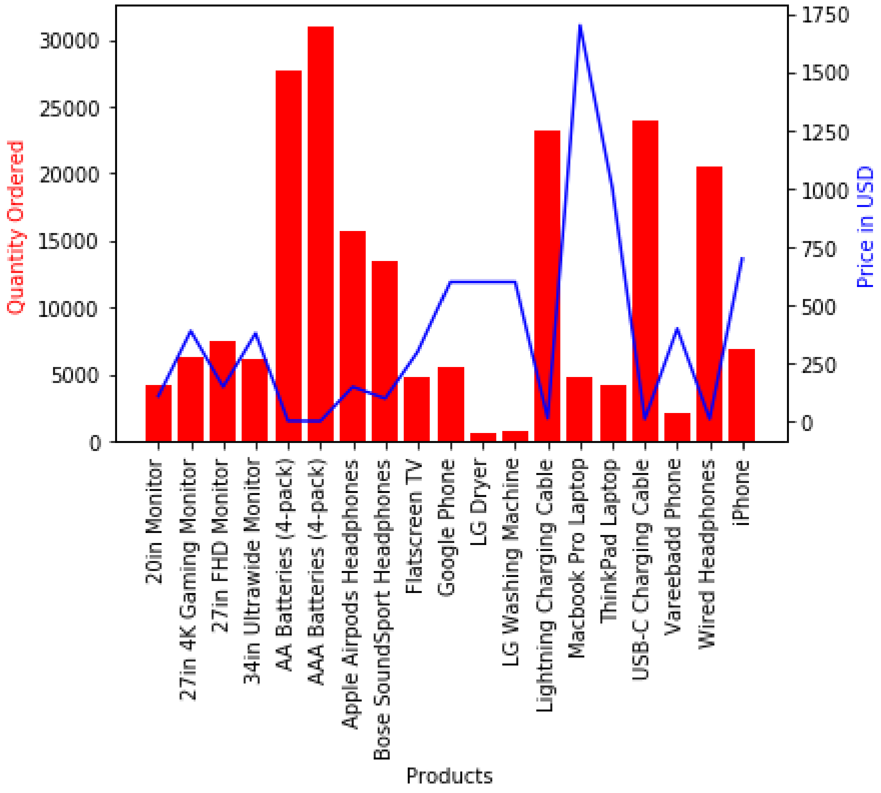
```
Product
20in Monitor              109.99
27in 4K Gaming Monitor    389.99
27in FHD Monitor          149.99
34in Ultrawide Monitor    379.99
AA Batteries (4-pack)       3.84
Name: Price Each, dtype: float64
```

In [65]:

```
fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(Products,qty_order,color='r')
ax2.plot(Products, costs, 'b-')

ax1.set_xlabel('Products')
ax1.set_ylabel('Quantity Ordered', color='r')
ax2.set_ylabel('Price in USD', color='b')
ax1.set_xticklabels(Products,rotation='vertical',size=10)
plt.show()
```



In [66]:

```
#Here we have a qty ordered overlayed with price graph, which shows less price is equal
to more quantity ordered
```

I TRIED TO DO SOME ANALYSIS TO WITH SOME SALES DATA , ANSWERING QUESTIONS THAT ARE USUALLY

ASKED BY THE SALES AND MARKETING TEAMS ALL AROUND THE WORLD.

THANK YOU

In [ ]:

```

```