Sales Analysis

Importing libraries

In [46]:
```
import pandas as pd
import numpy as np
```

Lets read a single csv file

In [5]:
```
df=pd.read_csv('Sales_April_2019.csv')
df.head()
```

Out[5]:

	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

Step:1 Merge 12 months of sales data in single csv file

In [6]:
```
import glob
csv_files=glob.glob('*.{}'.format('csv'))
csv_files
```

Out[6]:
```
['Sales_April_2019.csv',
 'Sales_August_2019.csv',
 'Sales_December_2019.csv',
 'Sales_February_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']
```

In [7]:
```
df_append=pd.DataFrame()
for file in csv_files:
    df=pd.read_csv(file)
    df_append=df_append.append(df,ignore_index=True)
df_append
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2893495819.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use panda
s.concat instead.
  df_append=df_append.append(df,ignore_index=True)
```

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
...	...	...	...	...	...	...
186845	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
186846	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
186847	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
186848	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
186849	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

186850 rows × 6 columns

In [8]:
```
df_append.describe()
```

Out[8]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
count	186305	186305	186305	186305	186305	186305
unique	178438	20	10	24	142396	140788
top	Order ID	USB-C Charging Cable	1	11.95	Order Date	Purchase Address
freq	355	21903	168552	21903	355	355

Clean up the data

In [9]:
```
df_nan=df_append[df_append.isna().any(axis=1)]
df_nan.head()
```

Out[9]:

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

In [10]:
```
df_nan.shape
```

Out[10]:
```
(545, 6)
```

Drop rows with nan

In [11]:
```
df_append=df_append.dropna(how='all')
df_append.head()
```

Out[11]:

	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

In [13]:
```
df_append.shape
```

Out[13]:
```
(186305, 6)
```

In [14]:
```
df_append.dtypes
```

Out[14]:
```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
dtype: object
```

Data clean-up: deleting duplicate rows

In [15]:
```
df_temp=df_append[df_append['Order Date'].str[0:2]=='Or']
df_temp.head()
```

Out[15]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
519	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1149	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1155	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2878	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2893	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

In [16]:
```
df_temp.shape
```

Out[16]:
```
(355, 6)
```

In [17]:
```
del df_temp
```

In [18]:
```
df_append=df_append[df_append['Order Date'].str[0:2]!='Or']
df_append.shape
```

Out[18]:
```
(185950, 6)
```

In [19]:
```
df_new=df_append.drop_duplicates()
df_new.shape
```

Out[19]:
```
(185686, 6)
```

In [20]:
```
df_new['Quantity Ordered']=pd.to_numeric(df_new['Quantity Ordered'])
df_new['Price Each']=pd.to_numeric(df_new['Price Each'])
df_new.head()
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\3292657684.py:1: 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_new['Quantity Ordered']=pd.to_numeric(df_new['Quantity Ordered'])
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\3292657684.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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new['Price Each']=pd.to_numeric(df_new['Price Each'])
```

Out[20]:

	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	600.00	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

In [ ]:

Making additional columns

In [21]:
```
#slicing first two characters from order date for month
df_new['Month']=df_new['Order Date'].str[0:2]
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\4138533325.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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new['Month']=df_new['Order Date'].str[0:2]
```

In [22]:
```
#convert month from str into int 32
df_new['Month']=df_new['Month'].astype('int32')
df_new.head()
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\1851674791.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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new['Month']=df_new['Month'].astype('int32')
```

Out[22]:

	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

Adding sales column

In [23]:
```
df_new['sales']=df_new['Quantity Ordered']*df_new['Price Each']
df_new.head()
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\1290726864.py:1: 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_new['sales']=df_new['Quantity Ordered']*df_new['Price Each']
```

Out[23]:

	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

Adding city column

In [24]:
```
#using apply function
def get_city(address):
    return address.split(',')[1]
def get_state(address):
    return (address.split(',')[2]).split(' ')[1]
df_new['city']=df_new['Purchase Address'].apply(lambda x: get_city(x)+'('+get_state(x)+')')
df_new.head()
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\3727653491.py:6: 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_new['city']=df_new['Purchase Address'].apply(lambda x: get_city(x)+'('+get_state(x)+')')
```

Out[24]:

	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)

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

In [25]:
```
results=df_new.groupby('Month').sum()
results
```

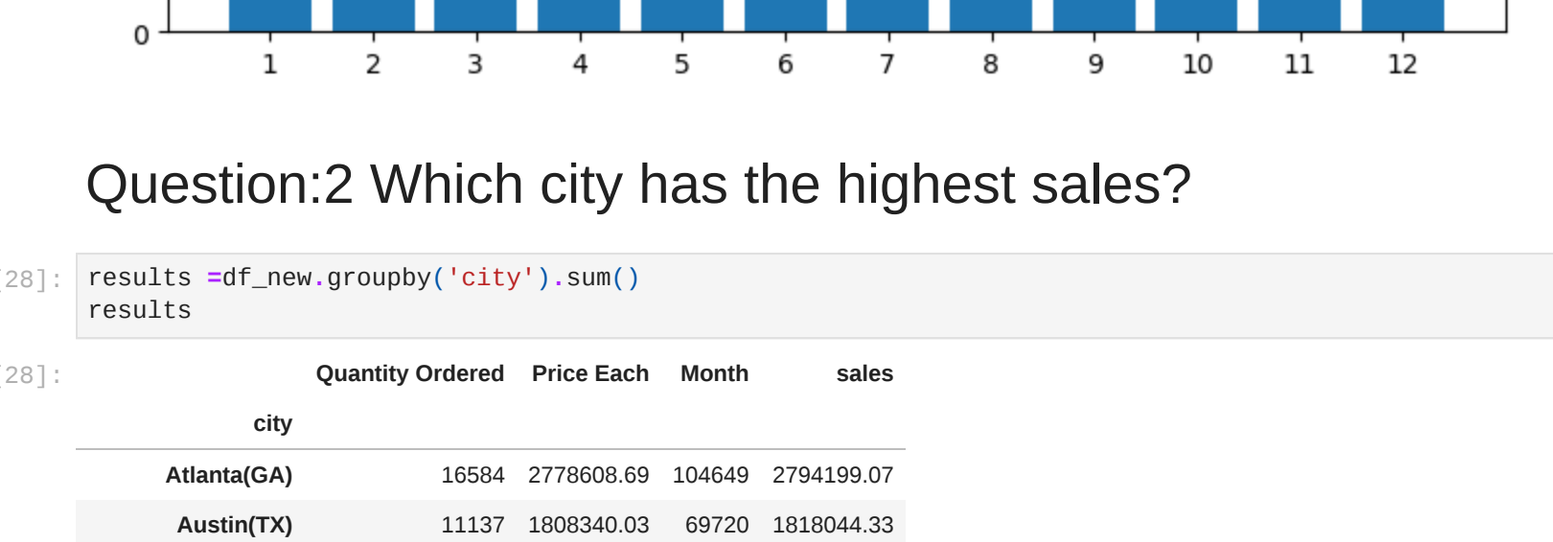
Out[25]:

Month	Quantity Ordered	Price Each	sales
1	10893	1810924.81	1821413.16
2	13431	2186940.38	2200078.08
3	16979	2789084.64	2804973.35
4	20536	3366218.76	3389217.98
5	18653	3133134.61	3150616.23
6	15234	2560503.50	2576280.15
7	16054	2631225.12	2646461.32
8	13429	2226963.90	2241083.37
9	13091	2081897.65	2094465.69
10	22669	3713608.80	3734777.86
11	19769	3178872.53	3197875.05
12	28074	4583267.77	4608295.70

In [26]:
```
import matplotlib.pyplot as plt
```

In [27]:
```
months=range(1,13)
plt.figure(figsize=(9,4))
plt.bar(months,results['sales'])

plt.xticks(months)
plt.ylabel('Total sales in a month in USD')
plt.grid(axis='y')
plt.show()
```



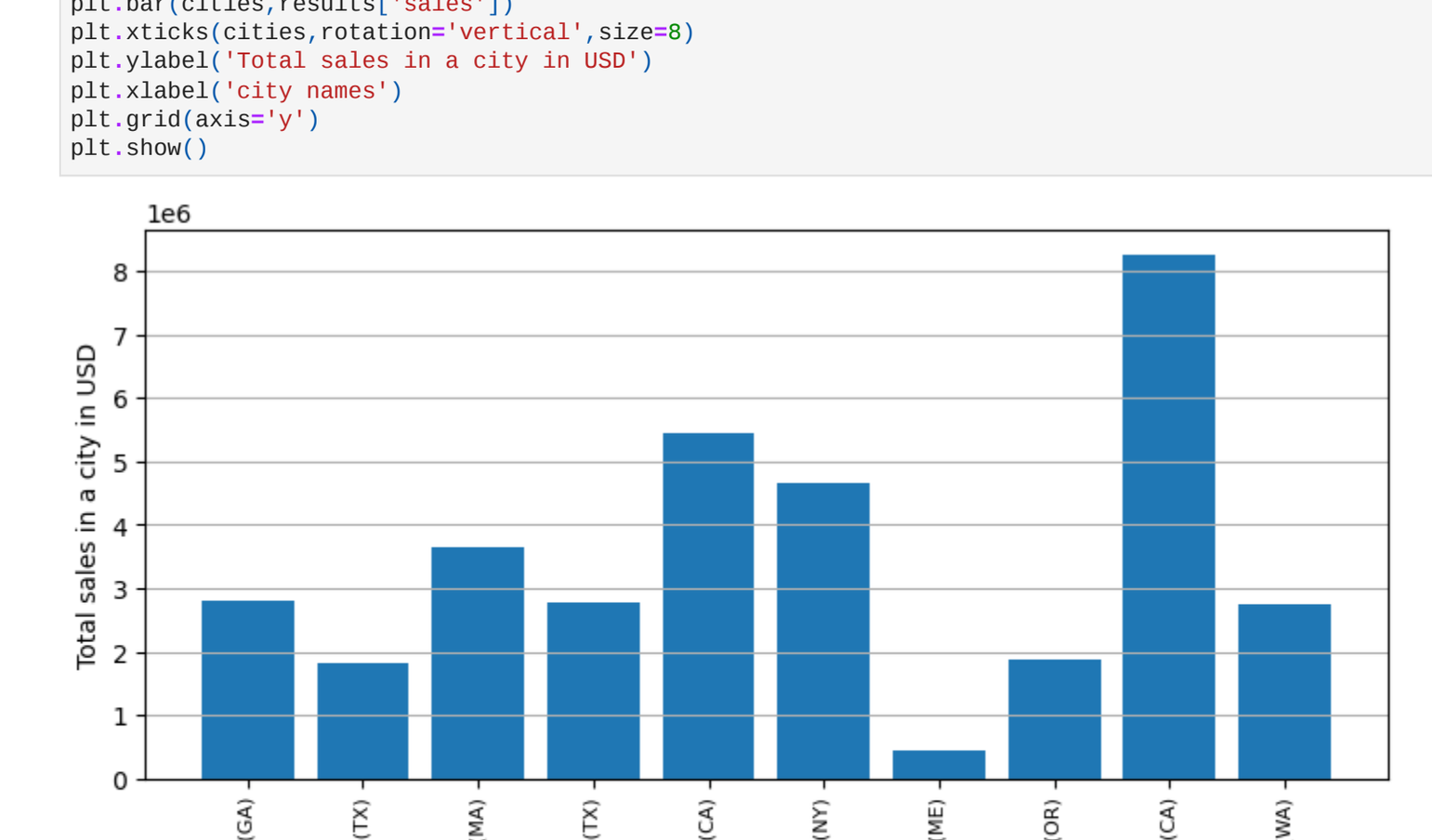
Question:2 Which city has the highest sales?

In [28]:
```
results =df_new.groupby('city').sum()
results
```

Out[28]:

city	Quantity Ordered	Price Each	Month	sales
Atlanta(GA)	16584	2778608.69	104649	2794199.07
Austin(TX)	11137	1808340.03	69720	1818044.33
Boston(MA)	22494	3634398.40	140898	3658627.65
Dallas(TX)	16707	2750026.38	104447	2765373.96
Los Angeles(CA)	33247	5417171.70	208020	5448304.28
New York City(NY)	27903	4632920.54	175557	4661867.14
Portland(ME)	2746	446752.36	17119	449321.38
Portland(OR)	11291	1859836.44	70517	1870010.56
San Francisco(CA)	50169	8204001.38	314949	8254743.55
Seattle(WA)	16534	2730586.55	104817	2745046.02

In [29]:
```
cities=results.index
plt.figure(figsize=(9,4))
plt.bar(cities,results['sales'])
plt.xticks(cities,rotation='vertical',size=8)
plt.ylabel('Total sales in a city in USD')
plt.xlabel('city names')
plt.grid(axis='y')
plt.show()
```



Question 3: What time we display ads to maximize profits?

First converting order data from str to date time

In [30]:
```
df_new['Order Date']=pd.to_datetime(df_new['Order Date'])
df_new.head()
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\3930132402.py:1: 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_new['Order Date']=pd.to_datetime(df_new['Order Date'])
```

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

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2136656433.py:1: 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_new['Hour']=df_new['Order Date'].dt.hour
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\2136656433.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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new['Minute']=df_new['Order Date'].dt.minute
```

Out[31]:

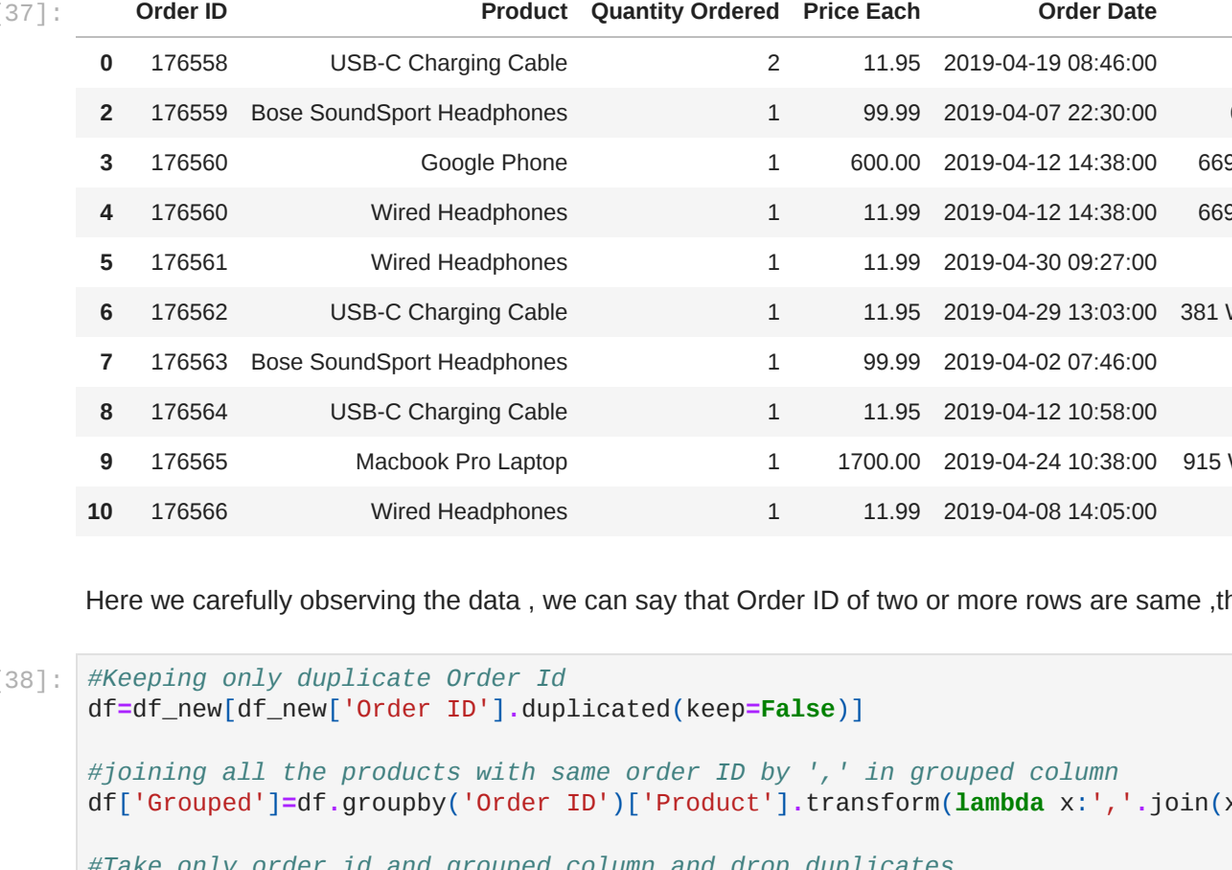
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	sales	city	Hour	Minute
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas(TX)	8	46
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	Boston(MA)	22	30
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	38
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	38
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	27

In [35]:
```
results=df_new.groupby('Hour').count()
results
```

Out[35]:

Hour	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	sales	city	Minute
0	3902	3902	3902	3902	3902	3902	3902	3902	3902	3902
1	2347	2347	2347	2347	2347	2347	2347	2347	2347	2347
2	1242	1242	1242	1242	1242	1242	1242	1242	1242	1242
3	830	830	830	830	830	830	830	830	830	830
4	852	852	852	852	852	852	852	852	852	852
5	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320
6	2481	2481	2481	2481	2481	2481	2481	2481	2481	2481
7	4002	4002	4002	4002	4002	4002	4002	4002	4002	4002
8	6252	6252	6252	6252	6252	6252	6252	6252	6252	6252
9	8740	8740	8740	8740	8740	8740	8740	8740	8740	8740
10	10929	10929	10929	10929	10929	10929	10929	10929	10929	10929
11	12392	12392	12392	12392	12392	12392	12392	12392	12392	12392
12	12573	12573	12573	12573	12573	12573	12573	12573	12573	12573
13	12115	12115	12115	12115	12115	12115	12115	12115	12115	12115
14	10965	10965	10965	10965	10965	10965	10965	10965	10965	10965
15	10159	10159	10159	10159	10159	10159	10159	10159	10159	10159
16	10359	10359	10359	10359	10359	10359	10359	10359	10359	10359
17	10884	10884	10884	10884	10884	10884	10884	10884	10884	10884
18	12263	12263	12263	12263	12263	12263	12263	12263	12263	12263
19	12886	12886	12886	12886	12886	12886	12886	12886	12886	12886
20	12218	12218	12218	12218	12218	12218	12218	12218	12218	12218
21	10905	10905	10905	10905	10905	10905	10905	10905	10905	10905
22	8808	8808	8808	8808	8808	8808	8808	8808	8808	8808
23	6262	6262	6262	6262	6262	6262	6262	6262	6262	6262

In [36]:
```
hours=results.index
plt.plot(hours,results['Order ID'])
plt.xticks(hours)
plt.xlabel('Hour of the day')
plt.ylabel('Total number of sales in an hours')
plt.grid()
plt.show()
```



In [ ]:
```
From the plot ,it clearly evident that around 12pm and 7pm maximum ads should be displayed
```

Question 4: What products are often sold together?

In [37]:
```
df_new.head(10)
```

Out[37]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	sales	city	Hour	Minute
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	46
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	30
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	38
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	38
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	27
6	176562	USB-C Charging Cable	1	11.95	2019-04-29 13:03:00	381 Wilson St, San Francisco, CA 94016	4	11.95	San Francisco(CA)	13	3
7	176563	Bose SoundSport Headphones	1	99.99	2019-04-02 07:46:00	668 Center St, Seattle, WA 98101	4	99.99	Seattle(WA)	7	46
8	176564	USB-C Charging Cable	1	11.95	2019-04-12 10:58:00	790 Ridge St, Atlanta, GA 30301	4	11.95	Atlanta(GA)	10	58
9	176565	Macbook Pro Laptop	1	1700.00	2019-04-24 10:38:00	915 Willow St, San Francisco, CA 94016	4	1700.00	San Francisco(CA)	10	38
10	176566	Wired Headphones	1	11.99	2019-04-08 14:05:00	83 7th St, Boston, MA 02215	4	11.99	Boston(MA)	14	5

Here we carefully observing the data , we can say that Order ID of two or more rows are same ,these corresponding products were sold together

In [38]:
```
#Keeping only duplicate Order Id
df=df_new[df_new['Order ID'].duplicated(keep=False)]

#joining all the products with same order ID by ',' in grouped column
df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x:','.join(x))

#Take only order id and grouped column and drop duplicates
df=df[['Order ID','Grouped']].drop_duplicates()
df.head(100)
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\3968361358.py: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))
```

Out[38]:

	Order ID	Grouped
3	176560	Google Phone,Wired Headphones
18	176574	Google Phone,USB-C Charging Cable
32	176586	AAA Batteries (4-pack),Google Phone
119	176672	Lightning Charging Cable,USB-C Charging Cable
129	176681	Apple Airpods Headphones,ThinkPad Laptop
...	...	...
2718	179162	Google Phone,USB-C Charging Cable
2783	179226	34in Ultrawide Monitor,Macbook Pro Laptop
2829	179270	iPhone,Lightning Charging Cable
2871	179311	Wired Headphones,USB-C Charging Cable
2890	179328	Wired Headphones,AA Batteries (4-pack)

100 rows × 2 columns

Now we can count number of occurences of the combinations

In [39]:
```
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)
```

```
('iPhone', 'Lightning Charging Cable') 1002
('Google Phone', 'USB-C Charging Cable') 985
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 413
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 159
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
```

iphones and lighting charging cable are bought together

In [ ]:

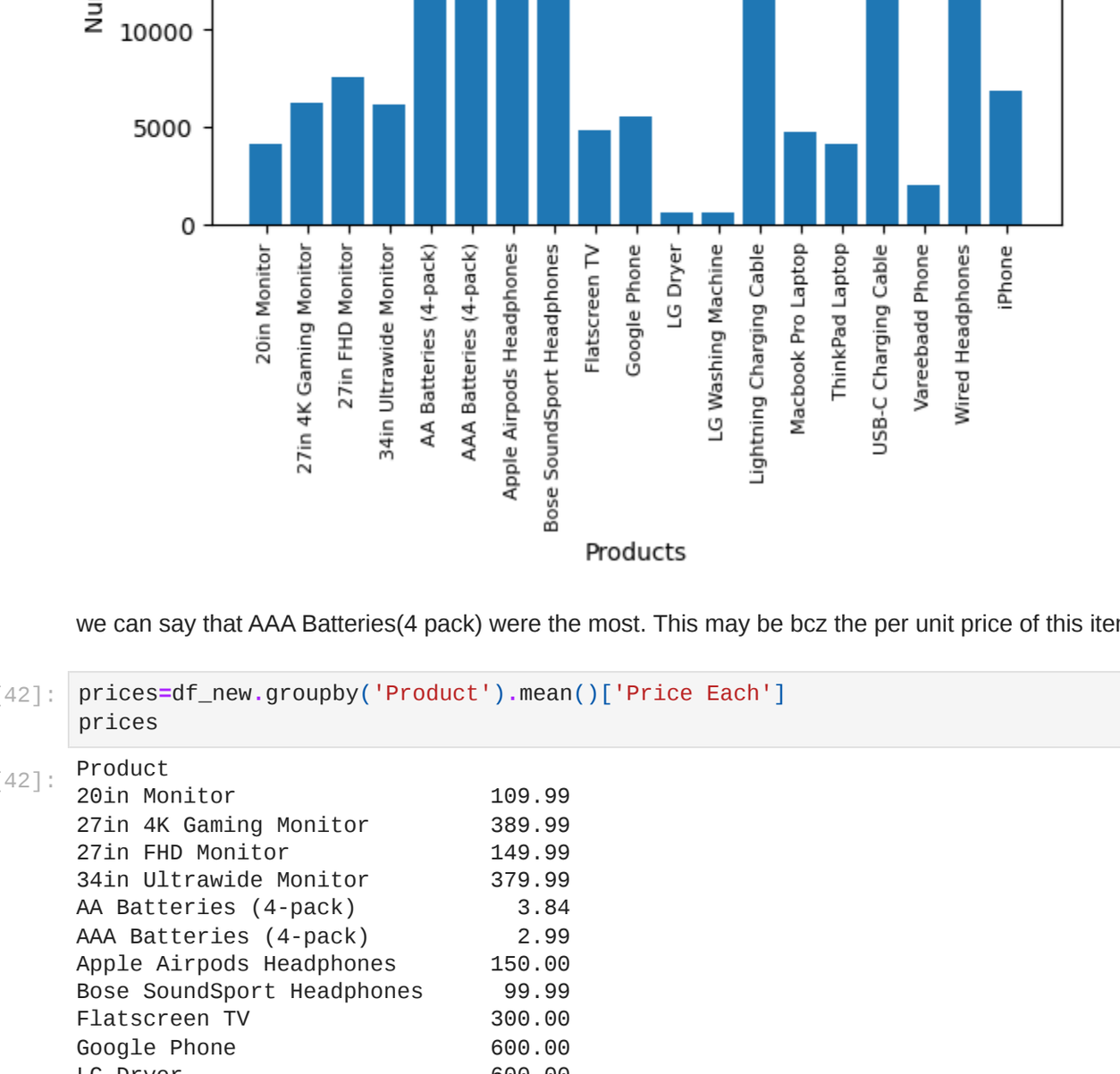
In [ ]:

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

In [40]:
```
product_group=df_new.groupby('Product')
quantity_ordered=product_group.sum()['Quantity Ordered']
quantity_ordered
```

Out[40]:
```
Product
20in Monitor                   4126
27in 4K Gaming Monitor         6239
27in FHD Monitor               7541
34in Ultrawide Monitor         6192
AA Batteries (4-pack)         27615
AAA Batteries (4-pack)        30986
Apple Airpods Headphones      15637
Bose SoundSport Headphones    13430
Flatscreen TV                  4813
Google Phone                   5529
LG Dryer                        646
LG Washing Machine              666
Lightning Charging Cable      23169
Macbook Pro Laptop             4725
ThinkPad Laptop                4128
USB-C Charging Cable          23931
Vareebadd Phone                2068
Wired Headphones              20524
iPhone                         6847
Name: Quantity Ordered, dtype: int64
```

In [41]:
```
products=quantity_ordered.index
plt.bar(products,quantity_ordered)
plt.xticks(products,rotation='vertical',size=8)
plt.ylabel('Number of units sold')
plt.xlabel('Products')
plt.show()
```



we can say that AAA Batteries(4 pack) were the most. This may be bcz the per unit price of this item is lowest.

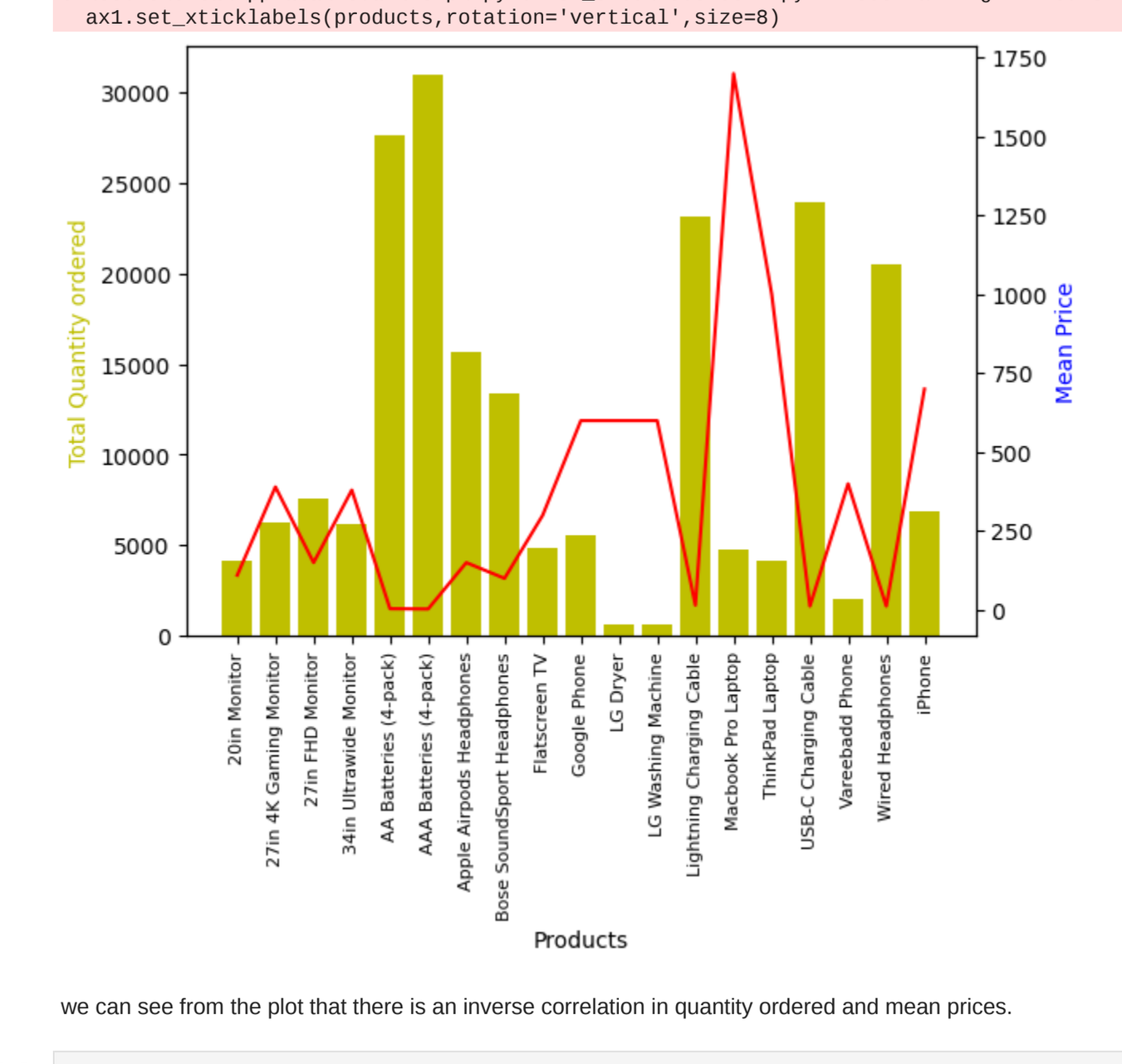
In [42]:
```
prices=df_new.groupby('Product').mean()['Price Each']
prices
```

Out[42]:
```
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
```

In [45]:
```
fig,ax1=plt.subplots()
ax2=ax1.twinx()
ax1.bar(products,quantity_ordered,color='y')
ax1.set_xticklabels(products,rotation='vertical',size=8)
ax1.set_ylabel('Total Quantity ordered',color='y')
ax1.set_xlabel('Products')

ax2.plot(products,prices,color='r')
ax2.set_ylabel('Mean Price',color='b')
plt.show()
```

```
C:\Users\ekta\AppData\Local\Temp\ipykernel_5160\790545827.py:4: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(products,rotation='vertical',size=8)
```



we can see from the plot that there is an inverse correlation in quantity ordered and mean prices.

In [ ]:

In [ ]:

In [ ]: