In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import seaborn as sns
```

Merging 12 months of sales data into single CSV file

In [2]:
```python
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:
    #print(file)
    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)
```

In [3]:
```python
df.head()
```

Out[3]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	209921	USB-C Charging Cable	1	11.95	06/23/19 19:34	950 Walnut St, Portland, ME 04101
1	209922	Macbook Pro Laptop	1	1700.0	06/30/19 10:05	80 4th St, San Francisco, CA 94016
2	209923	ThinkPad Laptop	1	999.99	06/24/19 20:18	402 Jackson St, Los Angeles, CA 90001
3	209924	27in FHD Monitor	1	149.99	06/05/19 10:21	560 10th St, Seattle, WA 98101
4	209925	Bose SoundSport Headphones	1	99.99	06/25/19 18:58	545 2nd St, San Francisco, CA 94016

In [4]:
```python
all_data = pd.read_csv('all_data.csv')
all_data.head()
```

Out[4]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301

Cleaning Data

In [5]:
```python
nan_df = all_data[all_data.isna().any(axis = 1)]
nan_df.head()
```

Out[5]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
264	NaN	NaN	NaN	NaN	NaN	NaN
648	NaN	NaN	NaN	NaN	NaN	NaN
680	NaN	NaN	NaN	NaN	NaN	NaN
1385	NaN	NaN	NaN	NaN	NaN	NaN
1495	NaN	NaN	NaN	NaN	NaN	NaN

In [6]:
```python
all_data = all_data.dropna(how = 'all')
```

In [7]:
```python
all_data.isnull().values.any()
```

Out[7]: False

In [8]:
```python
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']
```

In [9]:
```python
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
# we use inplace = true becase we don't need to assing it to a new variable again
```

Out[9]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001	12
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001	12
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001	12
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016	12
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301	12

Converting str to int

In [10]:
```python
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
```

In [11]:
```python
all_data['Sales'] = all_data['Quantity Ordered']* all_data['Price Each']
```

In [12]:
```python
all_data.head()
```

Out[12]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	295665	Macbook Pro Laptop	1	1700.00	12/30/19 00:01	136 Church St, New York City, NY 10001	12	1700.00
1	295666	LG Washing Machine	1	600.00	12/29/19 07:03	562 2nd St, New York City, NY 10001	12	600.00
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001	12	11.95
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016	12	149.99
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301	12	11.95

What was the best month for sales? How much was earned thatt month?

In [13]:
```python
new_sales = all_data.groupby('Month').sum()
new_sales = new_sales.reset_index() # imoprtant to plot
```

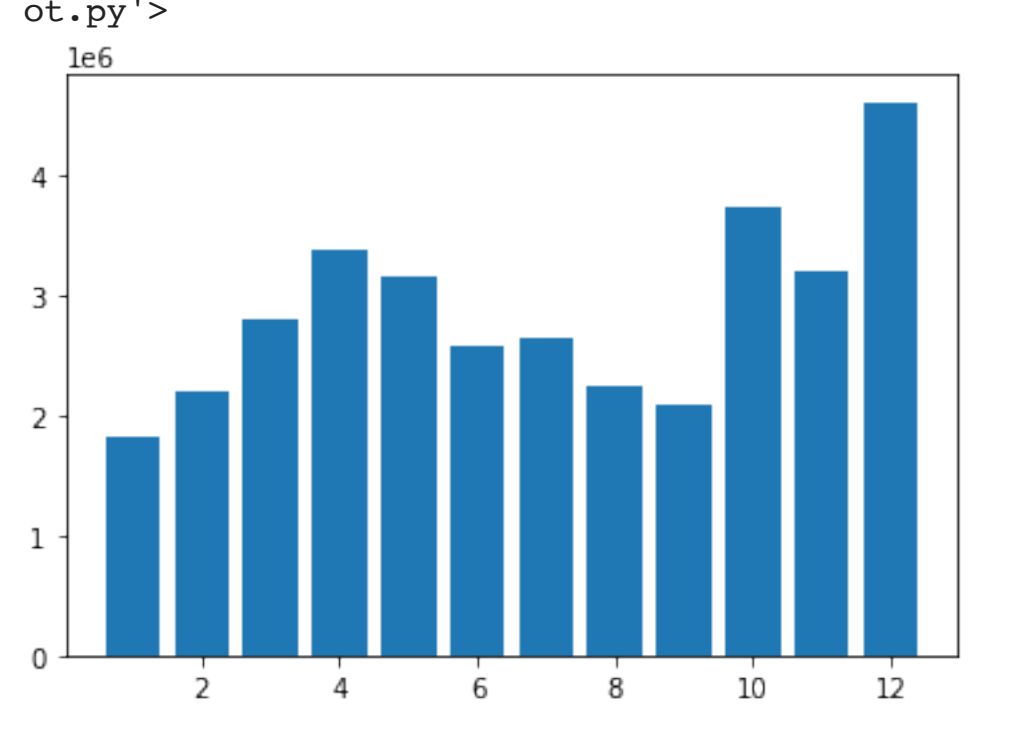
In [14]:
```python
new_sales
```

Out[14]:

	Month	Quantity Ordered	Price Each	Sales
0	1	10903	1.811768e+06	1.822257e+06
1	2	13449	2.188885e+06	2.202022e+06
2	3	17005	2.791208e+06	2.807100e+06
3	4	20558	3.367671e+06	3.390670e+06
4	5	18667	3.135125e+06	3.152607e+06
5	6	15253	2.562026e+06	2.577802e+06
6	7	16072	2.632540e+06	2.647776e+06
7	8	13448	2.230345e+06	2.244468e+06
8	9	13109	2.084992e+06	2.097560e+06
9	10	22703	3.715555e+06	3.736727e+06
10	11	19798	3.180601e+06	3.199603e+06
11	12	28114	4.588415e+06	4.613443e+06

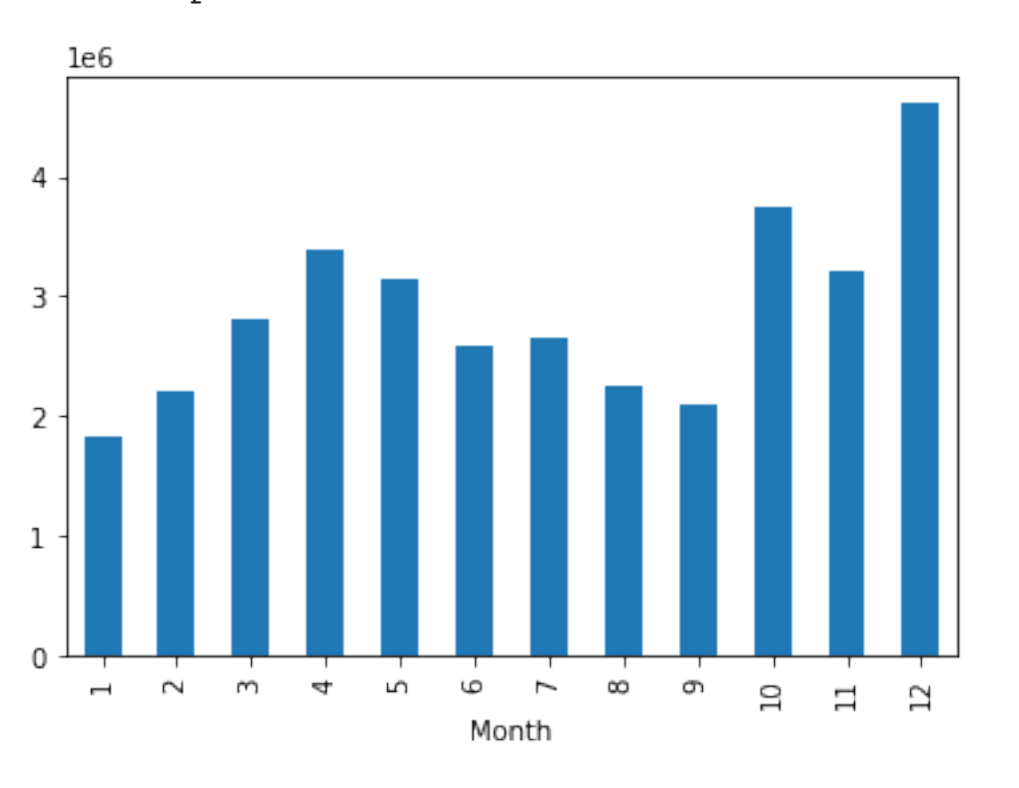
In [15]:
```python
months = range(1,13)
plt.bar(months, new_sales['Sales'])
plt
```

Out[15]: <module 'matplotlib.pyplot' from '/Users/samituttarkar/opt/anaconda3/lib/python3.8/site-packages/matplotlib/pyplot.py'>



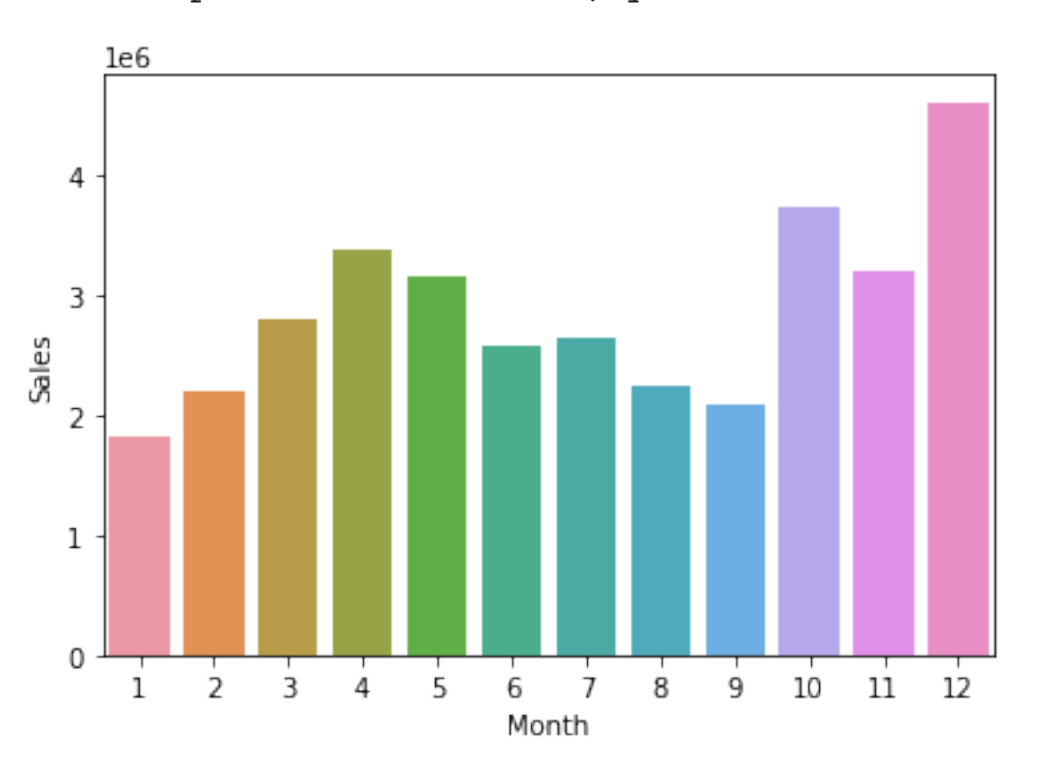
In [16]:
```python
all_data.groupby('Month')['Sales'].sum().plot(kind = 'bar')
```

Out[16]: <AxesSubplot:xlabel='Month'>



In [17]:
```python
sns.barplot(x="Month",
            y="Sales",
            data=new_sales)
```

Out[17]: <AxesSubplot:xlabel='Month', ylabel='Sales'>



What US City has to higest sales

In [18]:
```python
# let's use the .apply() method
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)} ({get_state(x)})") #lambda x:x.s
all_data.head()
```

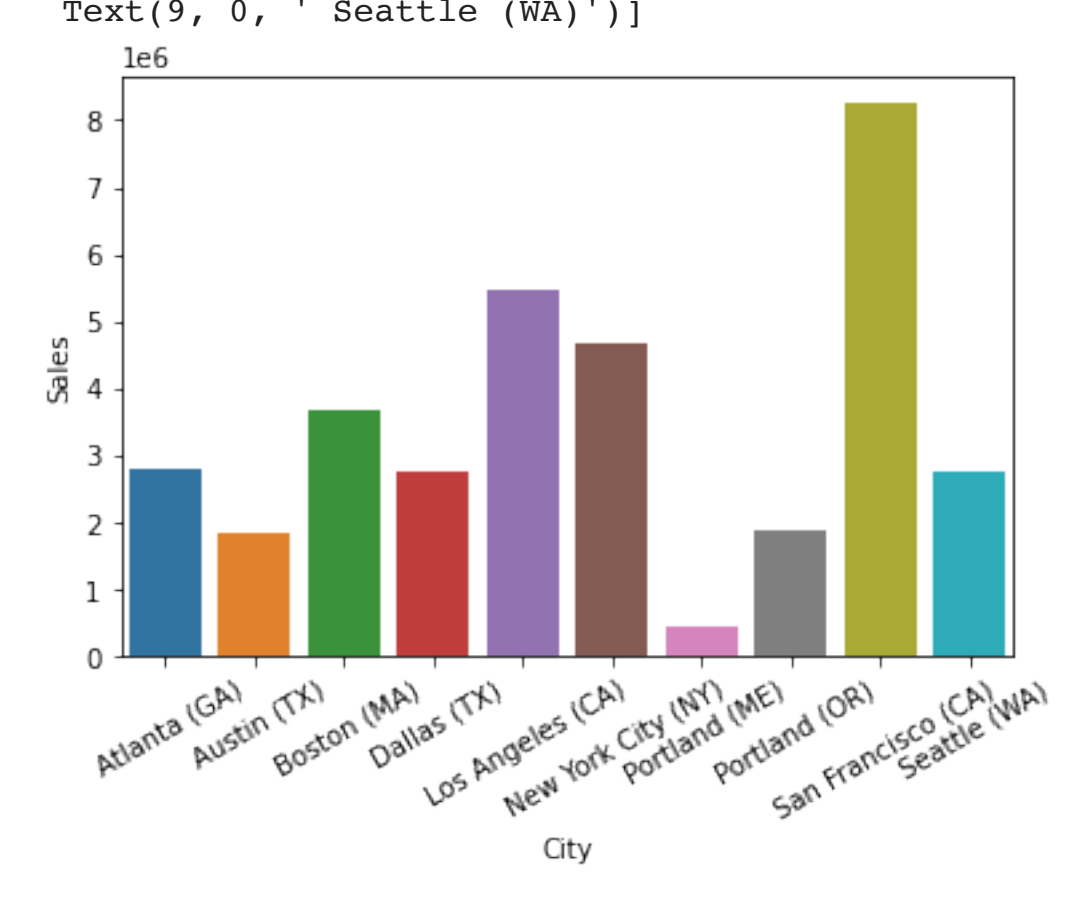
Out[18]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	295665	Macbook Pro Laptop	1	1700.00	12/30/19 00:01	136 Church St, New York City, NY 10001	12	1700.00	New York City (NY)
1	295666	LG Washing Machine	1	600.00	12/29/19 07:03	562 2nd St, New York City, NY 10001	12	600.00	New York City (NY)
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001	12	11.95	New York City (NY)
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco (CA)
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta (GA)

In [19]:
```python
city_sales = all_data.groupby('City')['Sales'].sum()
city_sales = city_sales.reset_index()
```

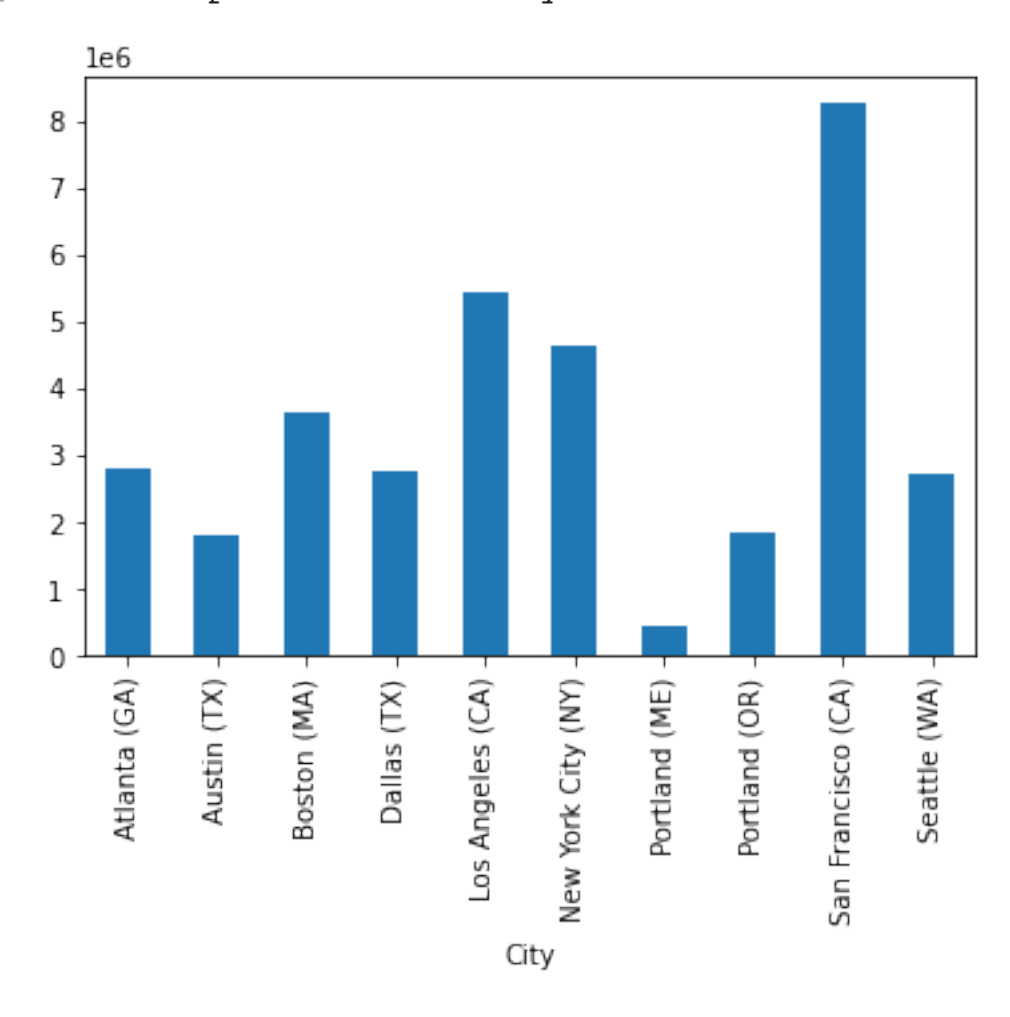
In [20]:
```python
g = sns.barplot(x = city_sales['City'], y = city_sales['Sales'] ,data = city_sales)
g.set_xticklabels(g.get_xticklabels(), rotation=30)
```

Out[20]:
```
[Text(0, 0, ' Atlanta (GA)'),
 Text(1, 0, ' Austin (TX)'),
 Text(2, 0, ' Boston (MA)'),
 Text(3, 0, ' Dallas (TX)'),
 Text(4, 0, ' Los Angeles (CA)'),
 Text(5, 0, ' New York City (NY)'),
 Text(6, 0, ' Portland (ME)'),
 Text(7, 0, ' Portland (OR)'),
 Text(8, 0, ' San Francisco (CA)'),
 Text(9, 0, ' Seattle (WA)')]
```

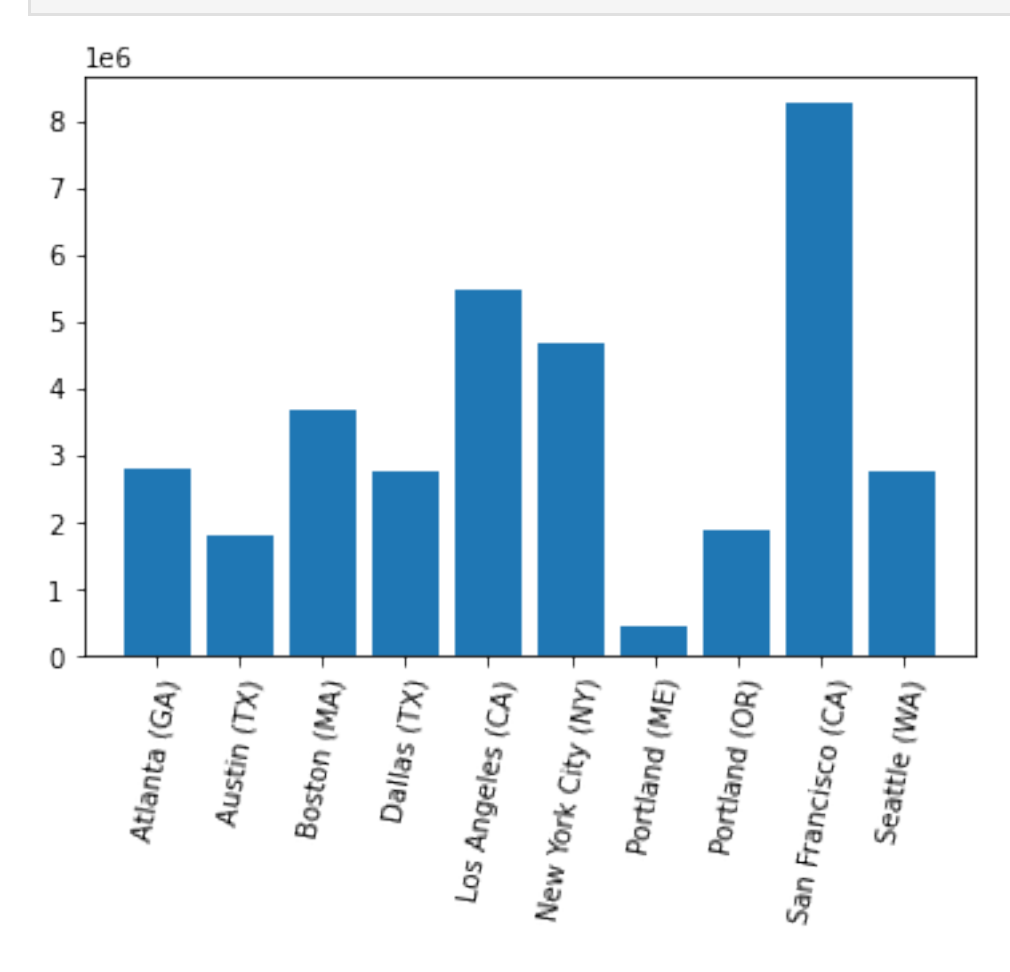


In [21]:
```python
all_data.groupby('City')['Sales'].sum().plot(kind = 'bar')
```

Out[21]: <AxesSubplot:xlabel='City'>



In [22]:
```python
plt.bar(city_sales['City'],city_sales['Sales'])
plt.xticks(rotation = 80)
plt.show()
```



What time should we display advertisements to maximize likelihood of customer's buying product ?

In [23]:
```python
#using date_time library cause it will help us get diffrent types of date and times
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

In [24]:
```python
#all_data.drop(columns = 'Column', inplace = True)
all_data.head()
```

Out[24]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City (NY)
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City (NY)
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City (NY)
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco (CA)
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta (GA)

In [25]:
```python
all_data['Hour'] = all_data['Order Date'].dt.hour
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data.head()
```

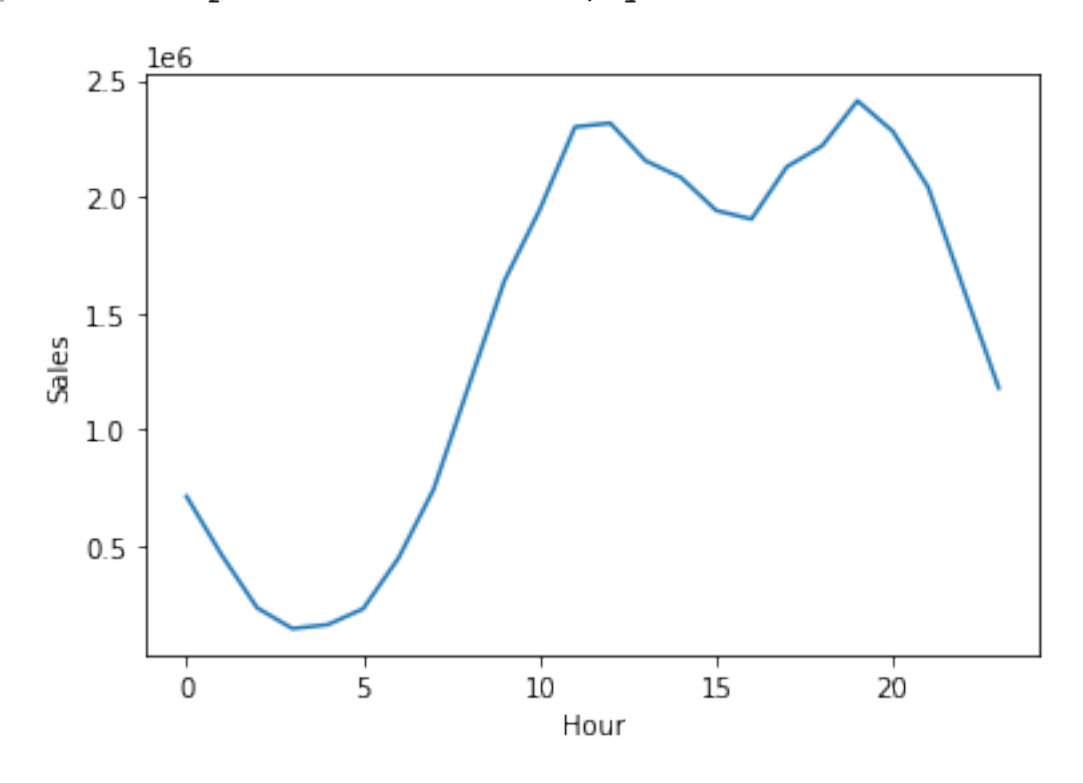
Out[25]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City (NY)	0	1
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City (NY)	7	3
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City (NY)	18	21
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco (CA)	15	13
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta (GA)	12	38

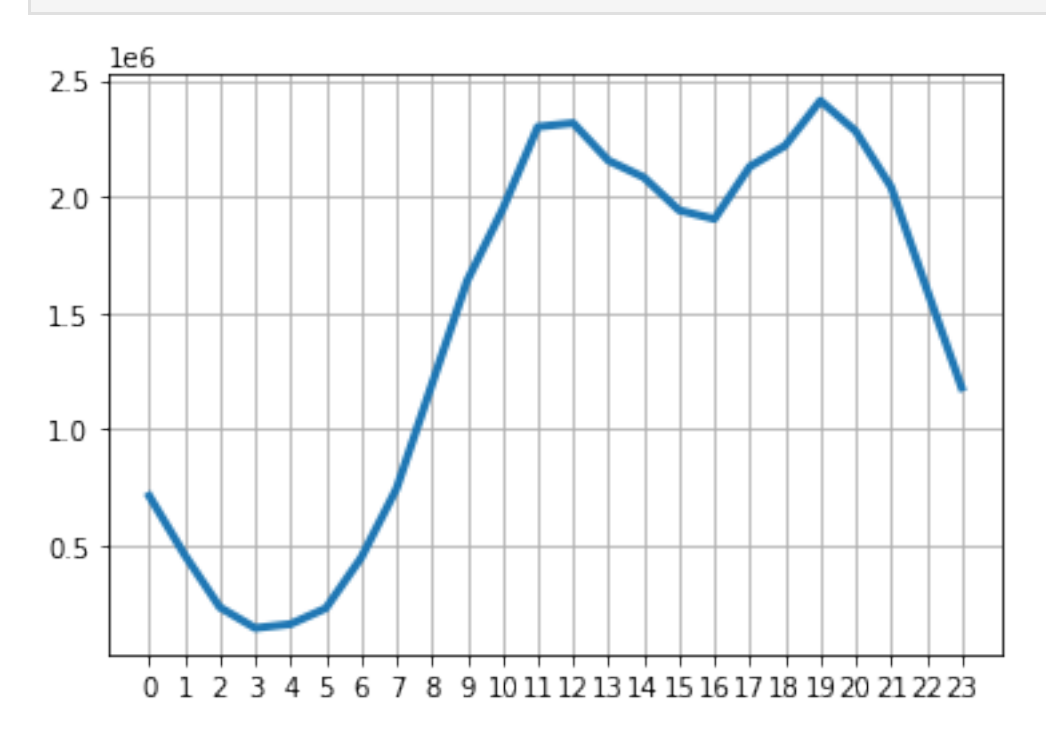
In [26]:
```python
by_hour = all_data.groupby('Hour')['Sales'].sum()
by_hour = by_hour.reset_index()
```

In [27]:
```python
sns.lineplot(x = by_hour['Hour'], y = by_hour['Sales'])
```

Out[27]: <AxesSubplot:xlabel='Hour', ylabel='Sales'>



In [28]:
```python
plt.plot(by_hour['Hour'], by_hour['Sales'],linewidth=3.0)
plt.xticks(by_hour['Hour'])
plt.grid()
plt.show()
```



In [29]:
```python
s = 0
for i in all_data.duplicated():
    if i == True:
        s += 1
print(s)
```

264

Which products are sold the most ?

In [48]:
```python
df = all_data[all_data['Order ID'].duplicated(keep = False)]
```

In [49]:
```python
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x:',' .join(x))
df.head()
```

```
<ipython-input-49-b32f6ddaa522>: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['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x:',' .join(x))
```

Out[49]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute	Grouped
16	295681	Google Phone	1	600.00	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	12	600.00	Boston (MA)	12	37	Google Phone,USB-C Charging Cable,Bose SoundSp...
17	295681	USB-C Charging Cable	1	11.95	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	12	11.95	Boston (MA)	12	37	Google Phone,USB-C Charging Cable,Bose SoundSp...
18	295681	Bose SoundSport Headphones	1	99.99	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	12	99.99	Boston (MA)	12	37	Google Phone,USB-C Charging Cable,Bose SoundSp...
19	295681	Wired Headphones	1	11.99	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	12	11.99	Boston (MA)	12	37	Google Phone,USB-C Charging Cable,Bose SoundSp...
36	295698	Vareebadd Phone	1	400.00	2019-12-13 14:32:00	175 1st St, New York City, NY 10001	12	400.00	New York City (NY)	14	32	Vareebadd Phone,USB-C Charging Cable

In [50]:
```python
df = df[['Order ID','Grouped']].drop_duplicates()
```

In [51]:
```python
df.head()
```

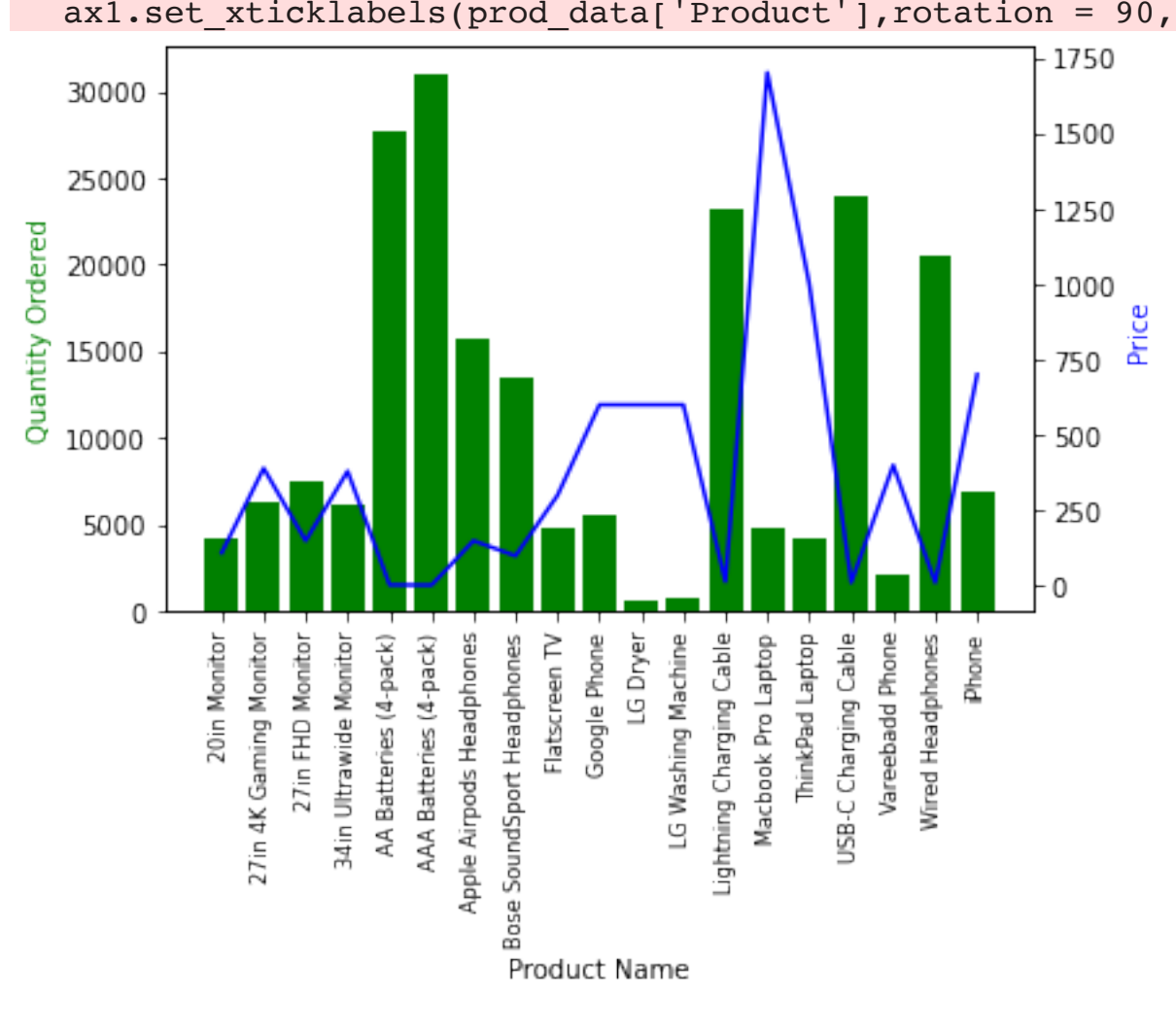
Out[51]:

	Order ID	Grouped
16	295681	Google Phone,USB-C Charging Cable,Bose SoundSp...
36	295698	Vareebadd Phone,USB-C Charging Cable
42	295703	AA Batteries (4-pack),Bose SoundSport Headphones
66	295726	iPhone,Lightning Charging Cable
76	295735	iPhone,Apple Airpods Headphones,Wired Headphones

In [56]:
```python
prod_data = all_data.groupby('Product')['Quantity Ordered'].sum()
prod_data = prod_data.reset_index()
```

In [60]:
```python
prices = all_data.groupby('Product').mean()['Price Each']
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.bar(prod_data['Product'],prod_data['Quantity Ordered'], color = 'g')
ax2.plot(prod_data['Product'],prices,'b-')
ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered',color = 'g')
ax2.set_ylabel('Price',color = 'b')
ax1.set_xticklabels(prod_data['Product'],rotation = 90,size = 8)
plt.show()
```

```
<ipython-input-60-5c868dfd64d9>:9: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(prod_data['Product'],rotation = 90,size = 8)
```



In [ ]: