In [3]:

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

Merging 12 months of sales data into a single file

In [4]:

```
files = [file for file in os.listdir('./data/raw/') ]


all_months_data = pd.DataFrame()

for file in files:
    df = pd.read_csv("./data/raw/"+file)
    all_months_data = pd.concat([all_months_data, df])

all_months_data.head()
```

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

In [6]:

```
all_months_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 186850 entries, 0 to 11685
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        186305 non-null  object
 4   Order Date        186305 non-null  object
 5   Purchase Address  186305 non-null  object
dtypes: object(6)
memory usage: 10.0+ MB
```

In [7]:

```
#Saving single file transformed
all_months_data.to_csv("./data/transformed/all_data.csv", index=False)
```

Read in updated dataframe

In [4]:

```
all_data = pd.read_csv("./data/transformed/all_data.csv")
all_data.head()
```

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

Clean up the data

In [5]:

```
# Check rows of NAN
nan_df = all_data[all_data.isna().any(axis=1)]
nan_df.head()
# Drop rows of NAN
all_data = all_data.dropna(how='all')
```

In [6]:

```
# Find 'Or' and delete and update all_data df
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']
```

In [7]:

```
#Convert columns to the correct type

#to int
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
#to float
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])

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
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

Augment data with additional columns

2: Add Month Column

In [8]:

```python
# Transforming "order date" column
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
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	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

3: Add a sales column

In [9]:

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

Out[9]:

	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

4: Add a city column

In [10]:

```
# # Methond 1: Let's use .apply() method

# all_data['Column'] = all_data['Purchase Address'].apply(lambda x: x.split(',')[1]
# all_data.head()

# Methon 2: Function tips with same line above

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: get_city(x) + ' ' +
                                               #apply(lambda x: f"{get_city(x)} ({g
all_data.head()
```

Out[10]:

	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 [11]:

```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 185950 entries, 0 to 186849
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Order ID          185950 non-null  object 
 1   Product           185950 non-null  object 
 2   Quantity Ordered  185950 non-null  int64  
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  object 
 5   Purchase Address  185950 non-null  object 
 6   Month             185950 non-null  int32  
 7   Sales             185950 non-null  float64
 8   City              185950 non-null  object 
dtypes: float64(2), int32(1), int64(1), object(5)
memory usage: 13.5+ MB
```

In [36]:

```
#Saving single file transformed
all_data.to_csv("./data/transformed/transformed_data.csv", index=False)
```

What was the best month for sales? how much was earned that month?

In [15]:

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

In [17]:

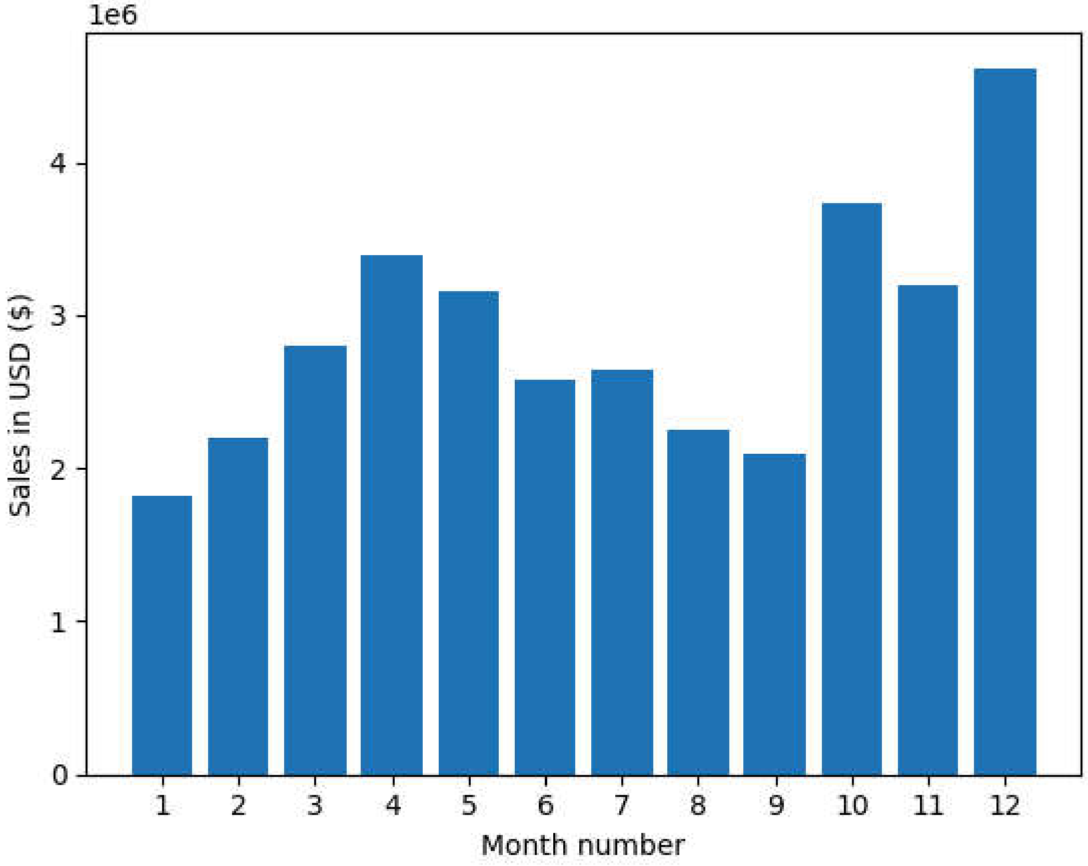
```
#Sales only
all_data.groupby('Month').sum()['Sales']
```

Out[17]:

```
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
```

In [18]:

```python
import matplotlib.pyplot as plt

months = range(1,13)

plt.bar(months, results['Sales'])
plt.xticks(months)
plt.xlabel('Month number')
plt.ylabel('Sales in USD ($)')
plt.show()
```



Conclusion

- Revenue tends to increase from the beginning of the year to the end of the year: Especially in the last months of the year (October, November, December), revenue grows significantly. This can be due to factors such as shopping seasonality, year-end focused marketing campaigns, or special events.

What city had the highest number of sales

In [13]:

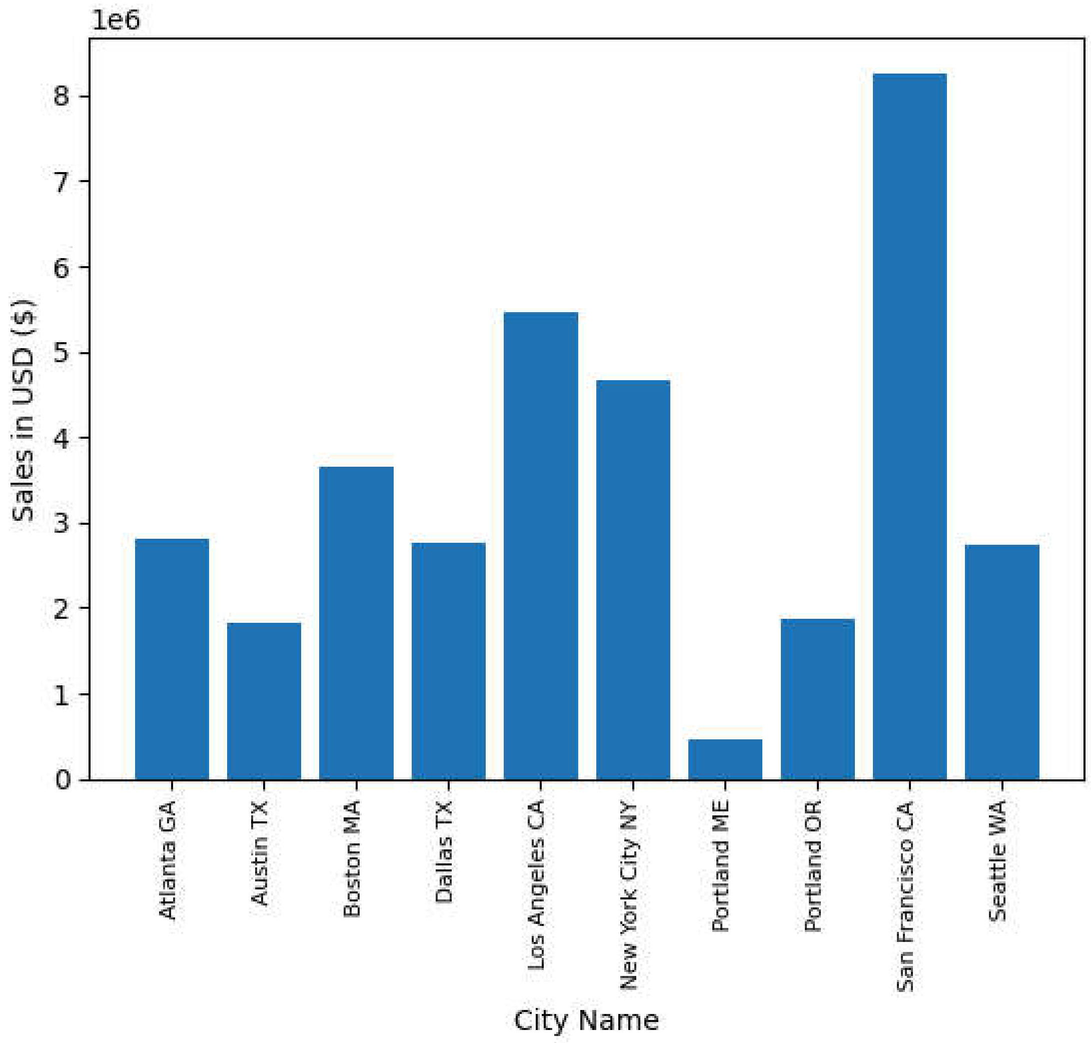
```
results = all_data.groupby('City')['Sales'].sum()
results
```

Out[13]:

```
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
```

In [16]:

```
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.xlabel('City Name')
plt.ylabel('Sales in USD ($)')
plt.show()
```



Conclusion

- Highest Revenue: San Francisco leads in revenue, followed by New York City and Los Angeles. This shows that the Western US market, especially the large cities, has great business potential.
- Significant differences between cities: Revenue between cities varies greatly, from a few hundred thousand to more than 8 million. This shows that the business potential and market size of each city are different.
- Concentration in large cities: Large cities such as New York, Los Angeles and San Francisco contribute a large part of the total revenue. This shows the importance of large urban markets.

What time should we display advertisements to maximize likelihood of customers buying products?

```
In [17]: all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_8752\3842191188.py:1: UserWarning: Coul
d not infer format, so each element will be parsed individually, falling back to `
dateutil`. To ensure parsing is consistent and as-expected, please specify a forma
t.
  all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

In [18]:

```
all_data.head()
```

Out[18]:

	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 [19]:

```
# By hour column
all_data['Hour'] = all_data['Order Date'].dt.hour
all_data.head()
```

Out[19]:

	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 [20]:

```
# by Minute column
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data.head()
```

Out[20]:

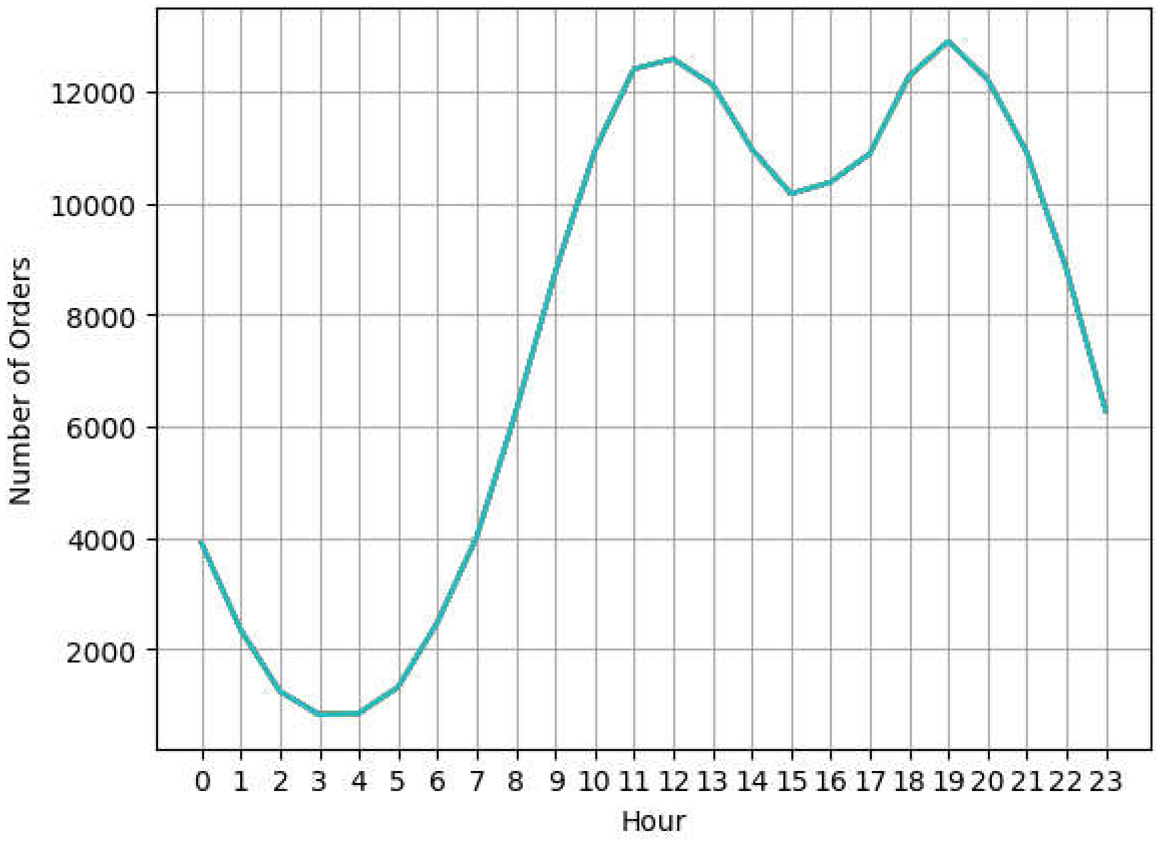
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minu
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas TX	8	
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	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 [21]:

```python
import matplotlib.pyplot as plt

hours = [hour for hour, df in all_data.groupby('Hour')]

plt.xticks(hours)
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.grid()
plt.plot(hours, all_data.groupby(['Hour']).count())
```

Out[21]:

```
[<matplotlib.lines.Line2D at 0x17fe56b8c90>,
 <matplotlib.lines.Line2D at 0x17fe589ec90>,
 <matplotlib.lines.Line2D at 0x17fe597a550>,
 <matplotlib.lines.Line2D at 0x17fe58e8250>,
 <matplotlib.lines.Line2D at 0x17fe5870ed0>,
 <matplotlib.lines.Line2D at 0x17fe59ae990>,
 <matplotlib.lines.Line2D at 0x17fe59afb50>,
 <matplotlib.lines.Line2D at 0x17fe59a80d0>,
 <matplotlib.lines.Line2D at 0x17fe59a04d0>,
 <matplotlib.lines.Line2D at 0x17fe59af810>]
```



Conclusion

- The time frame to advertise products is from 11am to 1pm and 6pm to 8pm.

What products are most often sold together?

In [22]:

```
#Get duplicated Order ID's
# https://stackoverflow.com/questions/43348194/pandas-select-rows-if-id-appear-seve
df = all_data[all_data['Order ID'].duplicated(keep=False)]

# Referenced: https://stackoverflow.com/questions/27298178/concatenate-strings-from
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
df2 = df[['Order ID', 'Grouped']].drop_duplicates()
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_8752\3234889904.py:6: SettingWithCopyWa
rning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join
(x))
```

In [23]:

```
df.head(10)
```

Out[23]:

	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
31	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
32	176586	AAA Batteries (4-pack)	2	2.99	2019-04-10 17:00:00	365 Center St, San Francisco, CA 94016	4	5.98	San Francisco CA	17
33	176586	Google Phone	1	600.00	2019-04-10 17:00:00	365 Center St, San Francisco, CA 94016	4	600.00	San Francisco CA	17
119	176672	Lightning Charging Cable	1	14.95	2019-04-12 11:07:00	778 Maple St, New York City, NY 10001	4	14.95	New York City NY	11

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	I
120	176672	USB-C Charging Cable	1	11.95	2019-04-12 11:07:00	778 Maple St, New York City, NY 10001	4	11.95	New York City NY	11	

In [24]:

```python
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') 2140
('Google Phone', 'USB-C Charging Cable') 2116
('iPhone', 'Wired Headphones') 987
('Google Phone', 'Wired Headphones') 949
('iPhone', 'Apple Airpods Headphones') 799
('Vareebadd Phone', 'USB-C Charging Cable') 773
('Google Phone', 'Bose SoundSport Headphones') 503
('USB-C Charging Cable', 'Wired Headphones') 452
('Vareebadd Phone', 'Wired Headphones') 327
('Lightning Charging Cable', 'Wired Headphones') 253
```

Conclusion

- Trend of selling products together: Customers often buy charging accessories together with their phones. (Over 2000 orders)

5. What products sold the most? Why do you think it sold the most?

In [28]:

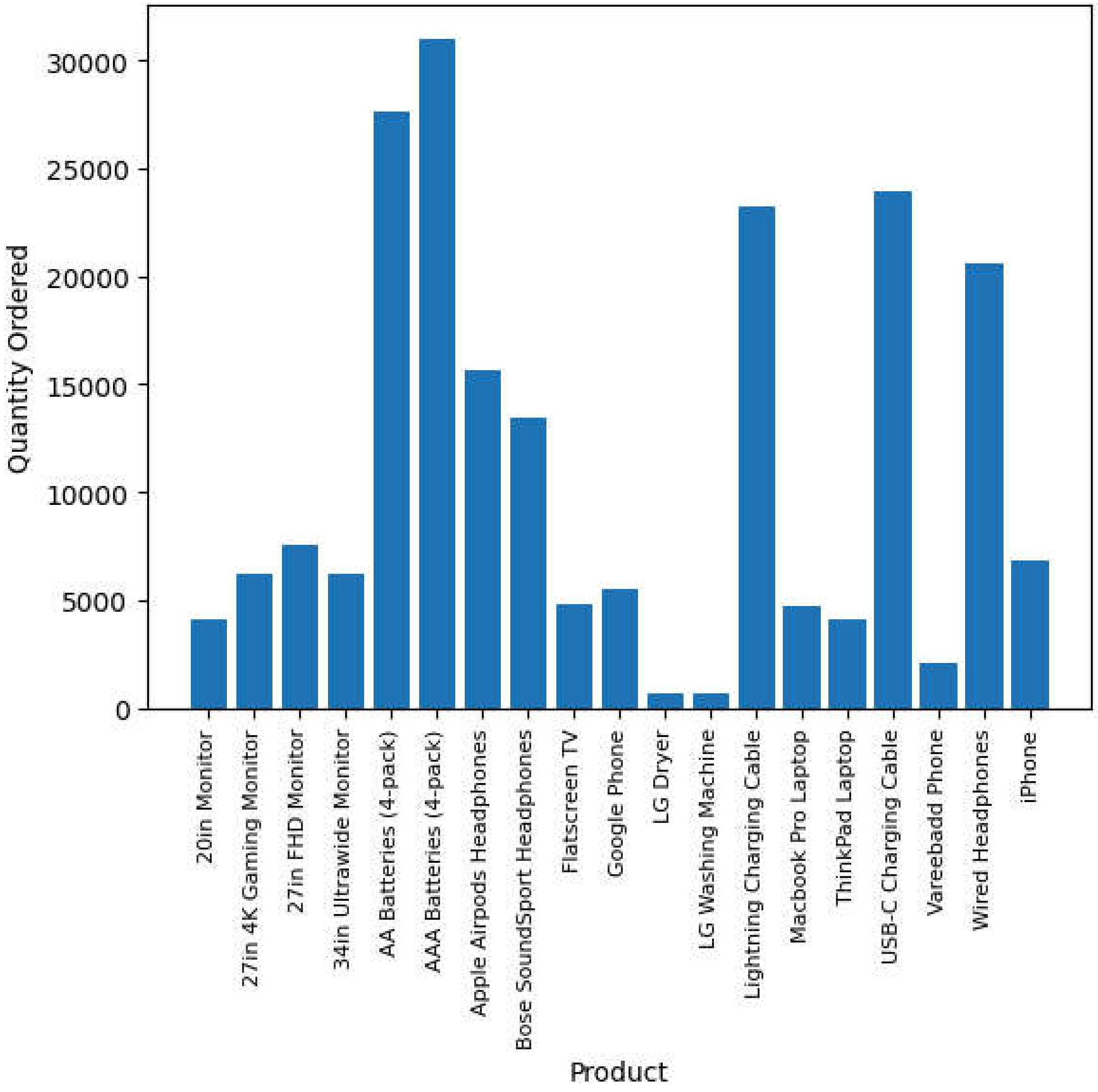
```python
product_group = all_data.groupby('Product')['Quantity Ordered']
quantity_ordered = product_group.sum()
```

In [29]:

```python
quantity_ordered
```

```
Out[29]: 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 [30]: 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 [33]:

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

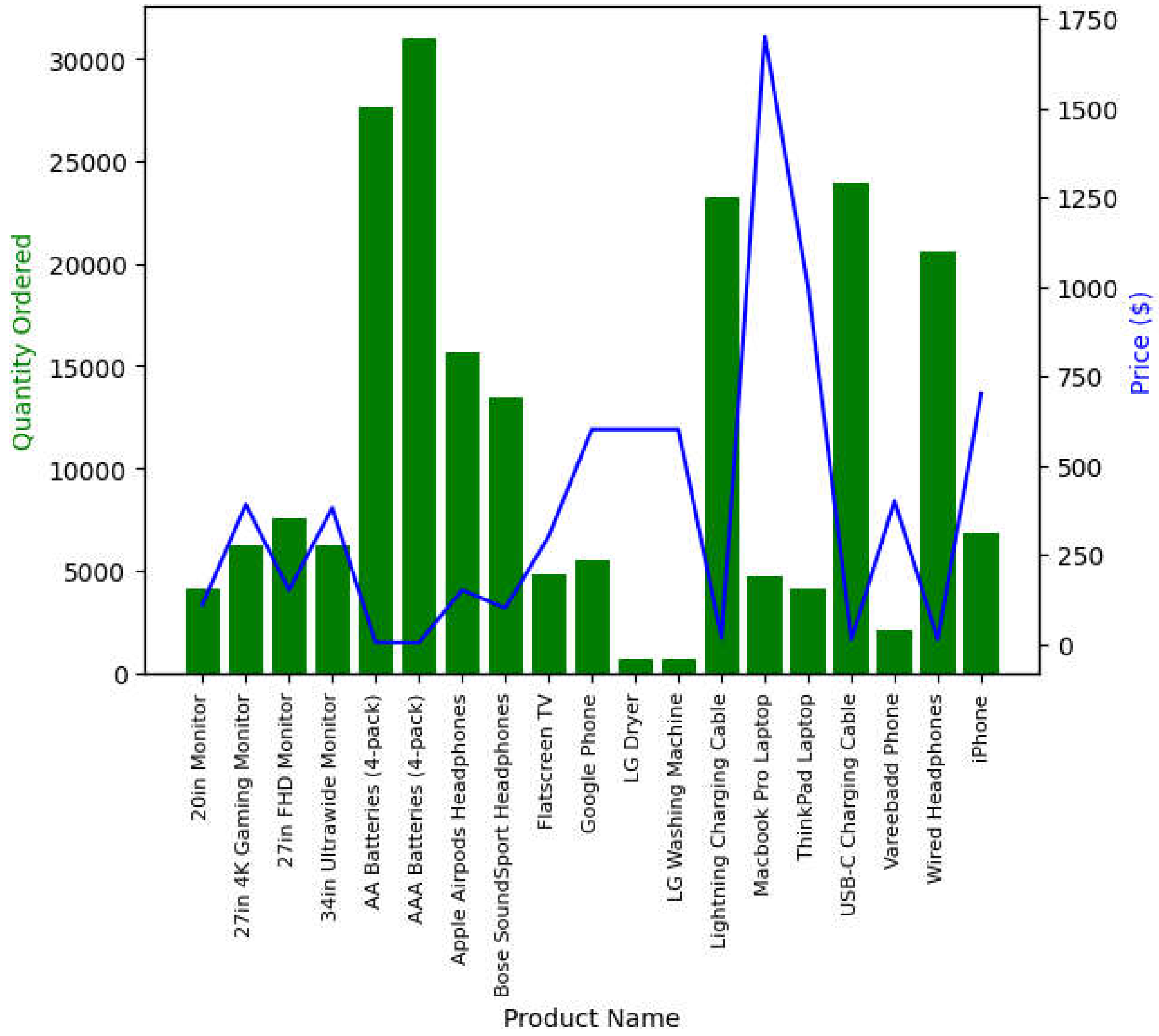
Out[33]:

```
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 [34]:

```
# Referenced: https://stackoverflow.com/questions/14762181/adding-a-y-axis-label-to

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(products, quantity_ordered, color='g')
ax2.plot(products, prices, color='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)

fig.show()
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_8752\3693591103.py:12: UserWarning: set
_ticklabels() should only be used with a fixed number of ticks, i.e. after set_tic
ks() or using a FixedLocator.
  ax1.set_xticklabels(products, rotation='vertical', size=8)
C:\Users\Acer\AppData\Local\Temp\ipykernel_8752\3693591103.py:14: UserWarning: Fig
ureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```



Conclusion:

- Best-selling products: AA Batteries (4-pack) led in sales volume with 27,635 units, followed by AAA Batteries (4-pack) with 31,017 units. This shows a high demand for daily consumer products.
- Best-selling phone accessories: Charging cables (Lightning Charging Cable and USB-C Charging Cable) and headphones (Wired Headphones, Apple Airpods Headphones) are the best-selling phone accessories.
- High-value products: Macbook Pro Laptop and iPhone are the two products with the highest value, contributing significantly to revenue.
- Popular computer monitors: Computer monitors such as 27in FHD Monitor, 27in 4K Gaming Monitor and 34in Ultrawide Monitor all had quite high sales volume, showing a demand for computer peripherals.