In [32]:
```python
import pandas as pd
import glob
import os
import matplotlib.pyplot as plt
#import seaborn as sns

# 2 line below for html export
import plotly.io as pio
pio.renderers.default='notebook'

# 2 line below for pdf export
!pip install pyppeteer
!pyppeteer-install
```

```
Requirement already satisfied: pyppeteer in c:\users\karol\anaconda3\lib\site-packages
(1.0.2)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\karol\anaconda3\lib\sit
e-packages (from pyppeteer) (1.4.4)
Requirement already satisfied: certifi>=2021 in c:\users\karol\anaconda3\lib\site-packag
es (from pyppeteer) (2023.7.22)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\karol\anaconda3\lib\s
ite-packages (from pyppeteer) (6.0.0)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in c:\users\karol\anaconda3\lib\site-p
ackages (from pyppeteer) (8.2.2)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\karol\anaconda3\lib\site-
packages (from pyppeteer) (4.65.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\karol\anaconda3\lib\si
te-packages (from pyppeteer) (1.26.16)
Requirement already satisfied: websockets<11.0,>=10.0 in c:\users\karol\anaconda3\lib\si
te-packages (from pyppeteer) (10.4)
Requirement already satisfied: zipp>=0.5 in c:\users\karol\anaconda3\lib\site-packages
(from importlib-metadata>=1.4->pyppeteer) (3.11.0)
Requirement already satisfied: colorama in c:\users\karol\anaconda3\lib\site-packages (f
rom tqdm<5.0.0,>=4.42.1->pyppeteer) (0.4.6)
```

```
chromium is already installed.
```

Task1: Join all the csv fille into one dataframe

In [33]:
```python
#define path to CSV files
path = r'C:\Users\karol\sales_data'

#identify all CSV files
all_files = glob.glob(os.path.join("*.csv"))

#merge all CSV files into one DataFrame
df = pd.concat((pd.read_csv(f) for f in all_files), ignore_index=True)
```

```
C:\Users\karol\AppData\Local\Temp\ipykernel_63764\1257142584.py:8: DtypeWarning:

Columns (6,8,9) have mixed types. Specify dtype option on import or set low_memory=Fals
e.
```

In [34]:
```python
df.to_csv('all_data.csv')
df.head()
```

Out[34]:

	Unnamed: 0.5	Unnamed: 0.4	Unnamed: 0.2	Unnamed: 0.3	Unnamed: 0.1	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	
0	0.0	0.0	0.0	0.0	0.0	0.0	176558.0	USB-C Charging Cable	2.0	11.95	0

1	1.0	1.0	1.0	1.0	1.0	1.0	NaN	NaN	NaN	NaN	
2	2.0	2.0	2.0	2.0	2.0	2.0	176559.0	Bose SoundSport Headphones	1.0	99.99	0
3	3.0	3.0	3.0	3.0	3.0	3.0	176560.0	Google Phone	1.0	600.0	0
4	4.0	4.0	4.0	4.0	4.0	4.0	176560.0	Wired Headphones	1.0	11.99	0

In [35]:
```
df = df.drop(['Unnamed: 0.5','Unnamed: 0.4','Unnamed: 0.2', 'Unnamed: 0.3', 'Unnamed: 0.
df
```

Out[35]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.0	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
...	...	...	...	...	...	...
1492475	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
1492476	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
1492477	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
1492478	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
1492479	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

1492480 rows × 6 columns

In [36]:
```
df.shape
```

Out[36]:
```
(1492480, 6)
```

In [37]:
```
df.dtypes
```

```
Out[37]: Order ID            object
         Product             object
         Quantity Ordered    object
         Price Each          object
         Order Date          object
         Purchase Address    object
         dtype: object
```

Clean the data!

Drop rows of NAN

```
In [38]: df = df.dropna(how='all')
         df.head()
```

Out[38]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.0	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561.0	Wired Headphones	1.0	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Find 'Or' and delete it

```
In [39]: df = df[df['Order Date'].str[0:2] != 'Or']
         df.head()
```

Out[39]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.0	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561.0	Wired Headphones	1.0	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

```
In [ ]:
```

Convert columns to the correct type

```
In [41]: # Make float
```

```
df['Price Each'] = df['Price Each'].astype('float')
df['Price Each'].dtype
```

Out[41]:
```
dtype('float64')
```

In [45]:
```
# convert to numeric
df['Quantity Ordered'] = pd.to_numeric(df['Quantity Ordered'])
df['Quantity Ordered'].dtype
```

Out[45]:
```
dtype('float64')
```

In [48]:
```
# Make int
df['Quantity Ordered'] = df['Quantity Ordered'].astype('int32')
df['Quantity Ordered'].dtype
```

Out[48]:
```
dtype('int32')
```

Augment data with additional columns

Task 2: Add month column

In [49]:
```
df['Month'] = df['Order Date'].str[0:2]
df['Month'] = df['Month'].astype('int32')
df.head()
```

Out[49]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558.0	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	176559.0	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560.0	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560.0	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561.0	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

In [50]:
```
df['Month'].unique()
```

Out[50]:
```
array([ 4,  5,  8,  9, 12,  1,  2,  3,  7,  6, 11, 10])
```

In [51]:
```
df['Month'].dtype
```

Out[51]:
```
dtype('int32')
```

Task 3: Add a sales column

In [52]:
```
df['Sales'] = df['Quantity Ordered'] * df['Price Each']
df.head()
```

Out[52]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	176558.0	USB-C Charging	2	11.95	04/19/19	917 1st St, Dallas, TX 75001	4	23.90

		Cable			08:46			
2	176559.0	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
3	176560.0	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
4	176560.0	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
5	176561.0	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

Task 4: Add a city column

In [53]:

```python
# get city
def get_city(address):
    return address.split(',')[1]

# get the state
def get_state(address):
    return address.split(',')[2].split(' ')[1]

df['City'] = df['Purchase Address'].apply(lambda x: f"{get_city(x)} ({get_state(x)})")
df.head()
```

Out[53]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	176558.0	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559.0	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560.0	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560.0	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561.0	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 [54]:

```python
results = df.groupby('Month').sum()
results
```

```
C:\Users\karol\AppData\Local\Temp\ipykernel_63764\4082709524.py:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future ver
sion, numeric_only will default to False. Either specify numeric_only or select only col
umns which should be valid for the function.
```

Out[54]:

	Quantity Ordered	Price Each	Sales
Month			

1	87224	14494147.04	14578053.84
2	107592	17511077.76	17616179.36
3	136040	22329662.64	22456803.04
4	164464	26941368.16	27125361.92
5	149336	25081001.04	25220854.00
6	122024	20496204.88	20622418.08
7	128576	21060316.48	21182206.08
8	107584	17842763.36	17955743.04
9	104872	16679936.72	16780481.04
10	181624	29724438.64	29893815.04
11	158384	25444805.44	25596825.60
12	224912	36707323.28	36907546.72

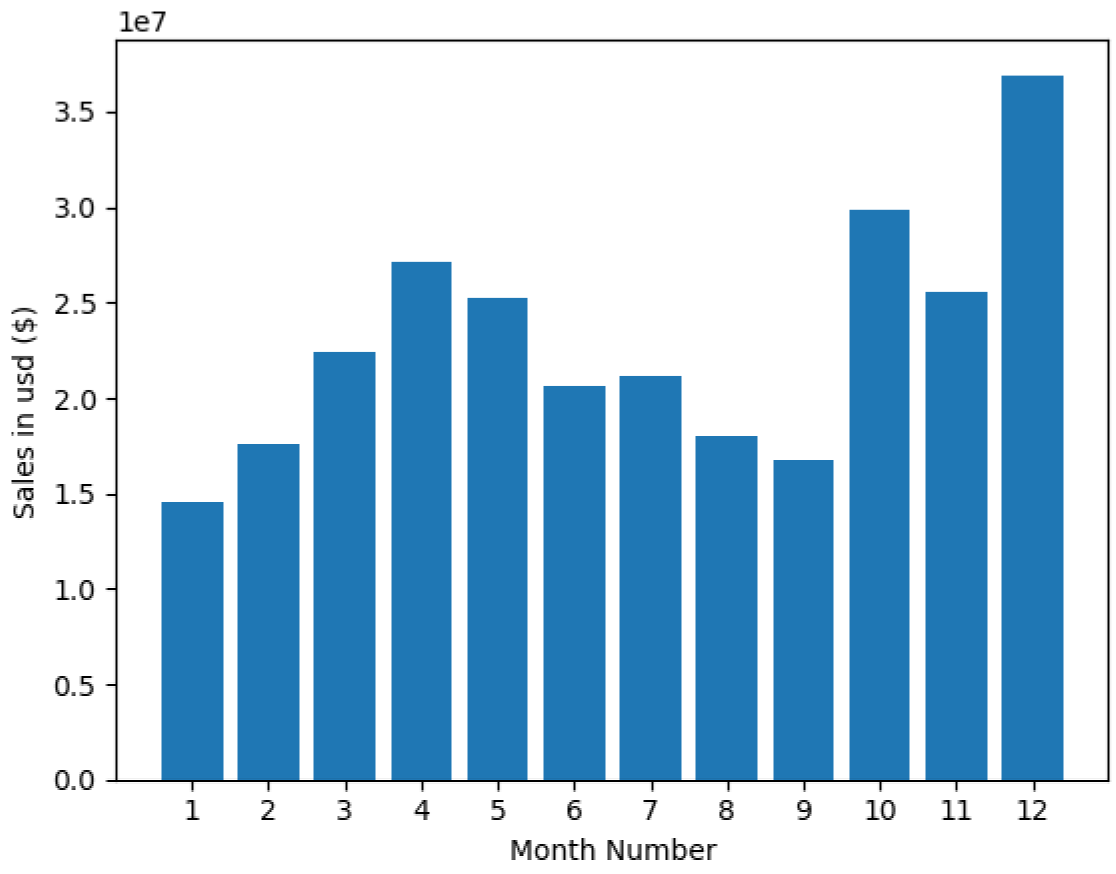
In [55]:
```
# reset index
df = df.reset_index(drop=True)
df.head()
```

Out[55]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	176558.0	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
1	176559.0	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
2	176560.0	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
3	176560.0	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
4	176561.0	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)

In [56]:
```
#import calendar

months = range(1,13)
plt.bar(months, results['Sales'])
plt.xticks(months)
plt.ylabel('Sales in usd ($)')
plt.xlabel('Month Number')


# Best month of sales is december
```

Out[56]:
```
Text(0.5, 0, 'Month Number')
```



Question 2: what city hasd the highest number of sales?

```
In [57]: cities_sales = df.groupby('City').sum()
         cities_sales
```

```
C:\Users\karol\AppData\Local\Temp\ipykernel_63764\2315319973.py:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future ver
sion, numeric_only will default to False. Either specify numeric_only or select only col
umns which should be valid for the function.
```

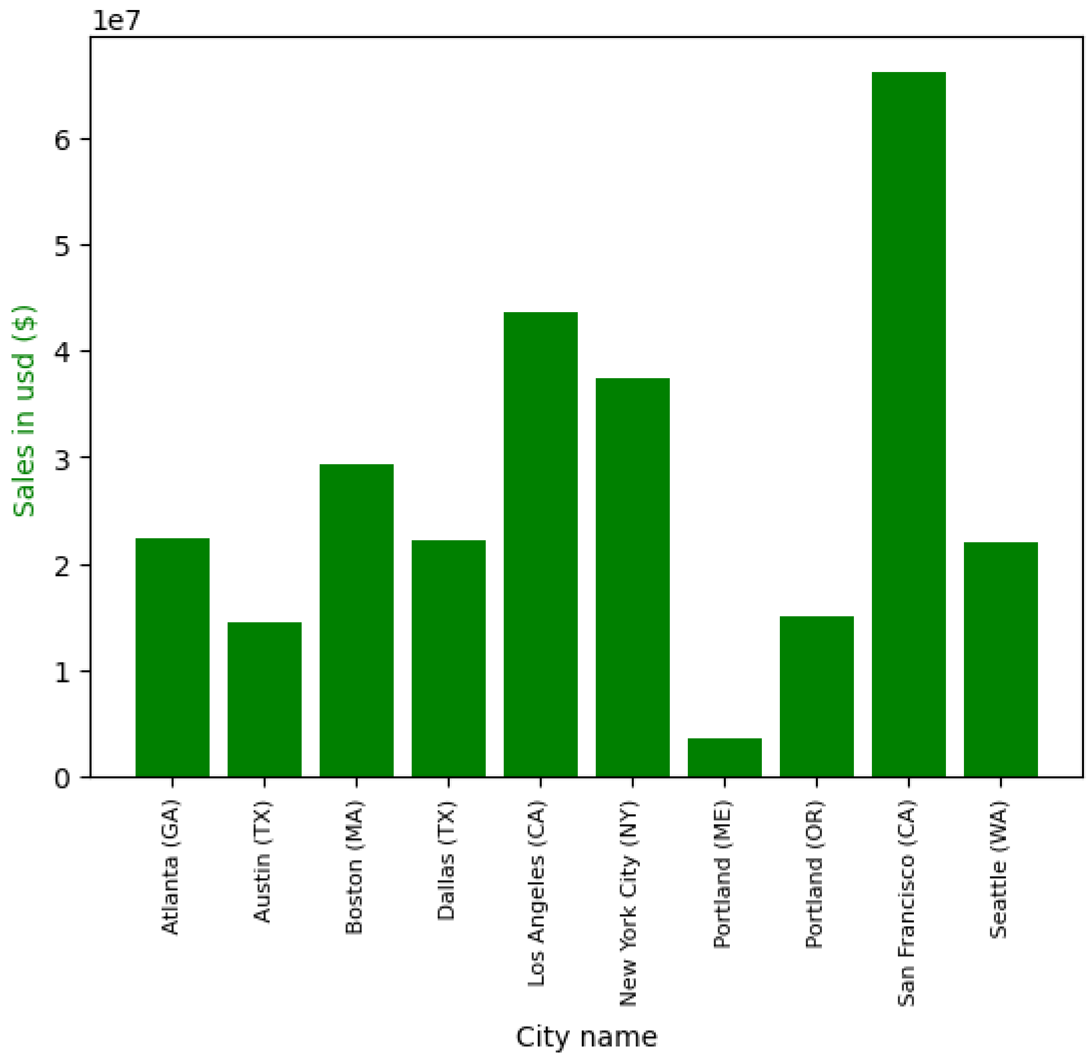
Out[57]:

City	Quantity Ordered	Price Each	Month	Sales
Atlanta (GA)	132816	22239265.60	838352	22363988.64
Austin (TX)	89224	14478988.88	558632	14556654.00
Boston (MA)	180224	29099278.16	1128896	29293136.08
Dallas (TX)	133840	22021022.56	836960	22143803.20
Los Angeles (CA)	266312	43371481.84	1666600	43620566.40
New York City (NY)	223456	37082966.64	1405928	37314539.44
Portland (ME)	22000	3577514.00	137152	3598066.16
Portland (OR)	90424	14884465.76	564968	14965858.72
San Francisco (CA)	401912	65691693.92	2524160	66097631.28
Seattle (WA)	132424	21866368.08	839528	21982043.84

```
In [58]: cities = [city for city, df in df.groupby('City')]
```

```
plt.bar(cities, cities_sales['Sales'],color='g')
plt.xticks(cities, rotation ='vertical', fontsize=8)
plt.ylabel('Sales in usd ($)', color='g')
plt.xlabel('City name')

# city with the highest sales is San Francisco (CA)
```

Out[58]:
```
Text(0.5, 0, 'City name')
```



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

In [70]:
```
# convert to datetime format
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Order Date'].dtype
```

Out[70]:
```
dtype('<M8[ns]')
```

Task 5: create Hour and minutes column

In [61]:
```
df['Hour'] = df['Order Date'].dt.hour
```

In [62]:
```
df['Minute'] = df['Order Date'].dt.minute
```

In [63]:
```
df.head()
```

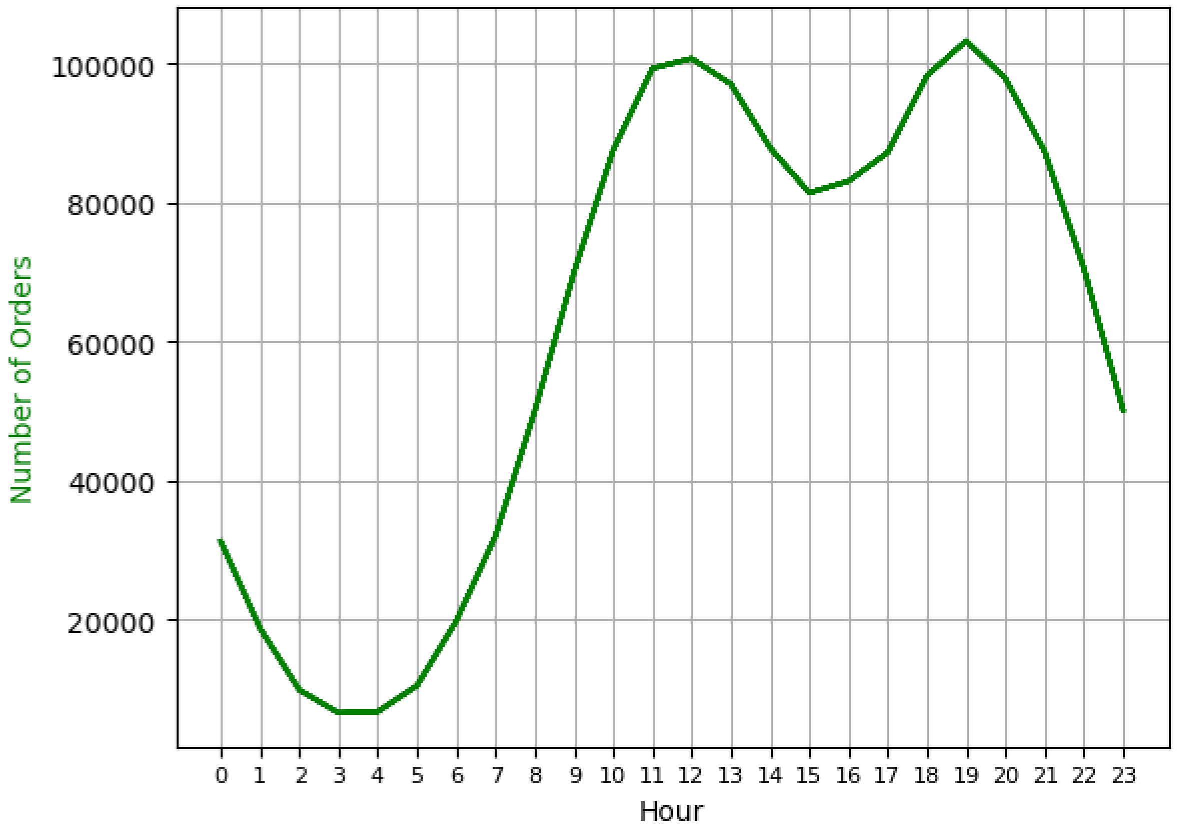
Out[63]:

Order	Product	Quantity	Price	Order	Purchase	Month	Sales	City	Hour	Minute

	ID		Ordered	Each	Date	Address					
0	176558.0	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
1	176559.0	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
2	176560.0	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
3	176560.0	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
4	176561.0	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 [77]:
```python
# groupby 'Hour' and Counted number of rows by each hour.
hours = [hour for hour, df in df.groupby('Hour')]
plt.plot(hours, df.groupby(['Hour']).count(), color='g')
plt.xticks(hours, fontsize=8)
plt.grid()
plt.ylabel('Number of Orders', color='g')
plt.xlabel('Hour')

# My recomendantion is arround 11am (11) or 7pm (19)
```

Out[77]:
```
Text(0.5, 0, 'Hour')
```



Question 4: What product sold the most? Why do you think it sold the most?

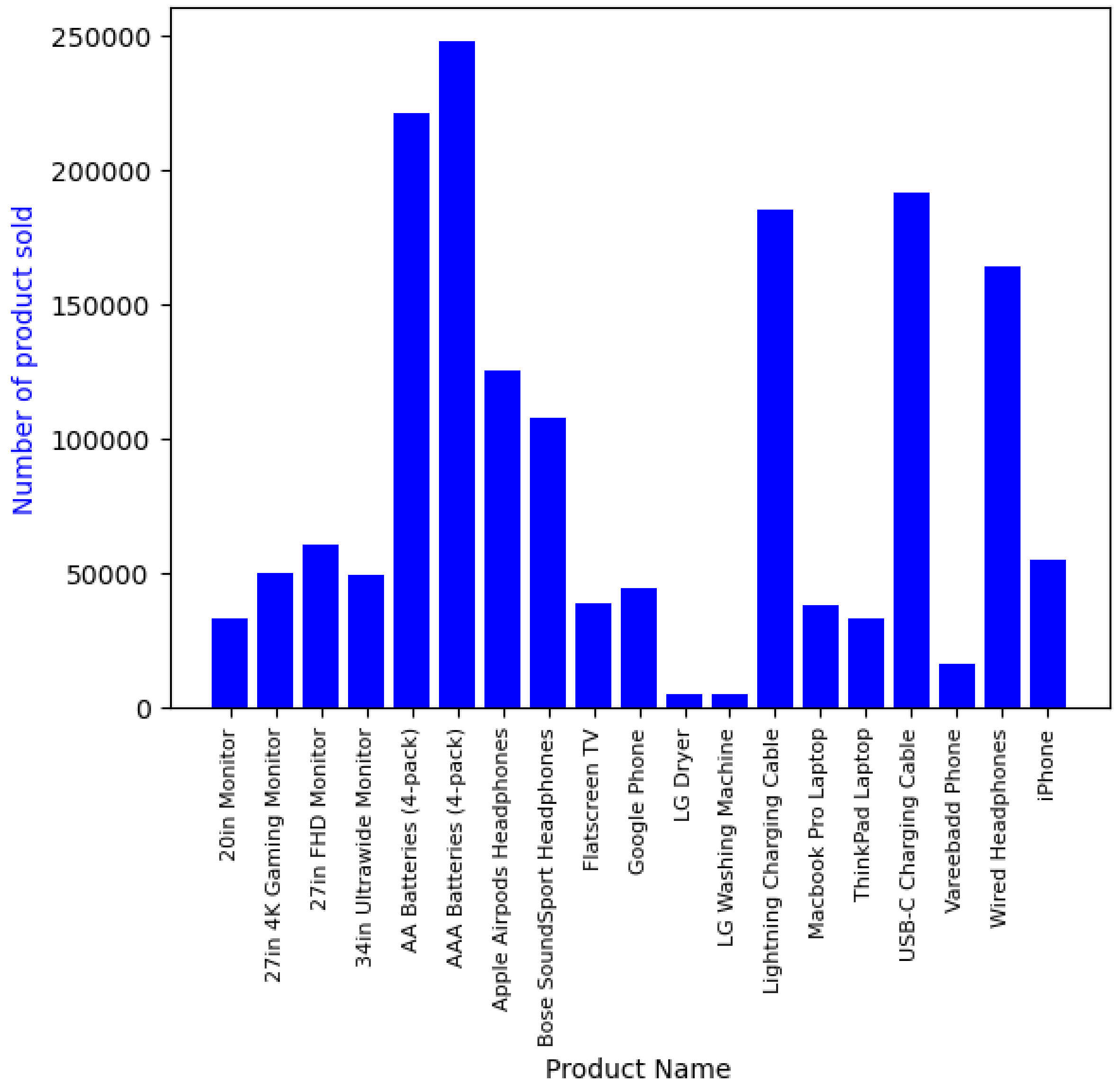
```
In [78]: product_group = df.groupby('Product')
         quantity_ordered = product_group.sum()['Quantity Ordered']
         quantity_ordered

         # the most sold product: AAA Batteries (4-pack)
         # number of times sold: 155.085
```

```
C:\Users\karol\AppData\Local\Temp\ipykernel_63764\2496607105.py:2: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future ver
sion, numeric_only will default to False. Either specify numeric_only or select only col
umns which should be valid for the function.
```

```
Out[78]: Product
         20in Monitor                   33032
         27in 4K Gaming Monitor         49952
         27in FHD Monitor               60400
         34in Ultrawide Monitor         49592
         AA Batteries (4-pack)         221080
         AAA Batteries (4-pack)        248136
         Apple Airpods Headphones      125288
         Bose SoundSport Headphones    107656
         Flatscreen TV                  38552
         Google Phone                   44256
         LG Dryer                        5168
         LG Washing Machine              5328
         Lightning Charging Cable      185736
         Macbook Pro Laptop             37824
         ThinkPad Laptop                33040
         USB-C Charging Cable          191800
         Vareebadd Phone                16544
         Wired Headphones              164456
         iPhone                         54792
         Name: Quantity Ordered, dtype: int32
```

```
In [79]: # bar chart
         products = [product for product, df in product_group]
         plt.bar(products, quantity_ordered, color='b')
         plt.xticks(products, rotation ='vertical', fontsize=8)
         plt.ylabel('Number of product sold', color='b')
         plt.xlabel('Product Name')
```

```
Out[79]: Text(0.5, 0, 'Product Name')
```



Question 5: Why the AAA Batteries (4-pack) is the most sold product?

number of times sold: 155.085

```
In [80]: product_prices = df.groupby('Product').mean()['Price Each']
         print(product_prices)
```

```
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
```

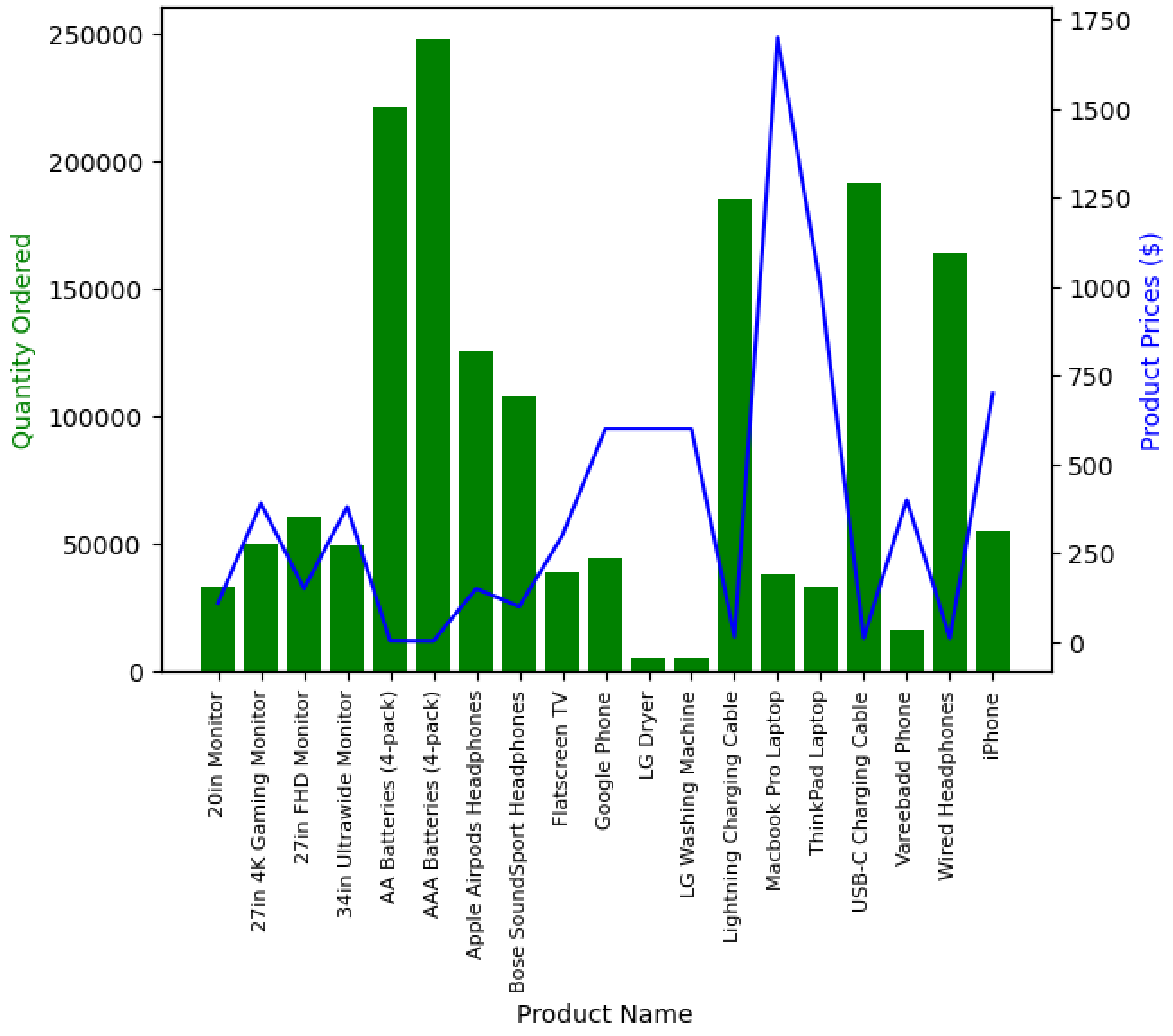
```
iPhone                             700.00
Name: Price Each, dtype: float64
```

```
C:\Users\karol\AppData\Local\Temp\ipykernel_63764\927660572.py:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future ve
rsion, numeric_only will default to False. Either specify numeric_only or select only co
lumns which should be valid for the function.
```

In [83]:

```
product_prices = df.groupby('Product').mean()['Price Each']
#print(product_prices)

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(products, quantity_ordered, color='g')
ax2.plot(products, product_prices, 'b-')

ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered', color='g')
ax2.set_ylabel('Product Prices ($)', color='b')
ax1.set_xticklabels(products, rotation='vertical', size=8)
plt.show()
```

```
C:\Users\karol\AppData\Local\Temp\ipykernel_63764\446822023.py:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future ve
rsion, numeric_only will default to False. Either specify numeric_only or select only co
lumns which should be valid for the function.

C:\Users\karol\AppData\Local\Temp\ipykernel_63764\446822023.py:13: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



The AAA Batteries (4-pack) it's the most cheap product:

AAA Batteries (4-pack) Price: 2.99$

As you can see in the graph above, when the product price is high, the quantity order is low.

When the product price is low, quantity ordered is high.

```
 Why Macbook Pro Laptop and ThinkPad Laptop prices are hight, but there's a
high qunatity ordered?
```

One of the reasons is because, there's many students and business that need a computer to function.