In [107…
```python
import pandas as pd
import os
import matplotlib.pyplot as plt
```

In [72]:
```python
df = pd.read_csv("./Sales_Data/Sales_April_2019.csv")
files = [file for file in os.listdir('./Sales_Data')]  #create a list for all files
all_months_data = pd.DataFrame()                        #create empty data frame

for file in files:                                      #start a loop, iterates each file
    df = pd.read_csv("./Sales_Data/"+file)              #read each csv file iterated in loop and store content in data frame df
    all_months_data = pd.concat([all_months_data, df]) #combine the data from all csv file into a single data frame

all_months_data.to_csv("all_data.csv" , index=False)   #exports the combine data frame to a csv file
```

Read in updated data frame

In [73]:
```python
all_data = pd.read_csv("all_data.csv")
all_data.head(6)
```

Out[73]:

	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

In [74]:
```python
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186850 entries, 0 to 186849
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: 8.6+ MB
```

In [75]:
```python
nan_df = all_data[all_data.isna().any(axis=1)]   #To find Nan values in Data
nan_df.head()
```

Out[75]:

	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 [76]:
```python
all_data.dropna(inplace=True)  #drop Nan values
```

In [77]:
```python
all_data.head()
```

Out[77]:

	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

Find 'Or' and delete it

In [78]:
```python
temp_df = all_data[all_data['Order Date'].str[0:2] == 'Or'] #to check if first two characters in order data column have 'Or'
temp_df.head()
```

Out[78]:

	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 [79]:
```python
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']  #filtering out rows for'Or'
```

Convert columns to the correct data type

In [80]:
```python
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered']) #Make int
all_data['Price Each'] = pd.to_numeric(all_data['Price Each']) #Make float
```

Augment data with additional columns

Add Month Column

In [81]:
```python
all_data['Month'] = all_data['Order Date'].str[0:2]    #create new column Month
all_data['Month'] = all_data['Month'].astype('int32')  #change data type
all_data.head()
```

Out[81]:

	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

Add a Sales column

In [82]:
```python
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']   #calculate total sales for each row
all_data.head()
```

Out[82]:

	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

Add a city column

In [88]:
```python
# use.apply
def get_city(address):
    return address.split(',')[1]   #split address to get city from

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

all_data['City'] = all_data['Purchase Address'].apply(lambda x: f"{get_city(x)} ({get_state(x)})") #create City column and  concatenated city and state using a

all_data.head()
```

Out[88]:

	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)

Question1: What was the best month for sales? How much was earned that month?

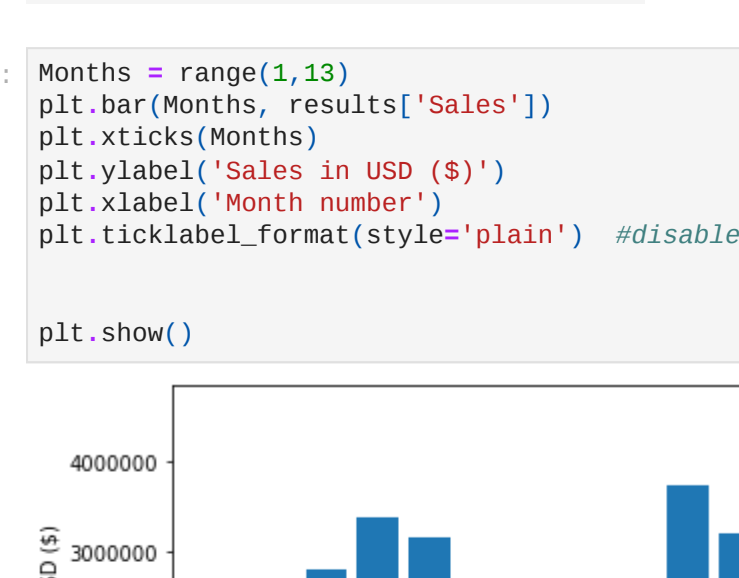
In [84]:
```python
results = all_data.groupby('Month').sum()
results.head(12)
```

Out[84]:

Month	Quantity Ordered	Price Each	Sales
1	10903	1811768.38	1822256.73
2	13449	2188884.72	2202022.42
3	17005	2791207.83	2807100.38
4	20558	3367671.02	3390670.24
5	18667	3135125.13	3152606.75
6	15253	2562025.61	2577802.26
7	16072	2632539.56	2647775.76
8	13448	2230345.42	2244467.88
9	13109	2084992.09	2097560.13
10	22703	3715554.83	3736726.88
11	19798	3180600.68	3199603.20
12	28114	4588415.41	4613443.34

In [85]:
```python
Months = range(1,13)
plt.bar(Months, results['Sales'])
plt.xticks(Months)
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month number')
plt.ticklabel_format(style='plain')  #disable the scientific notation on the y-axis


plt.show()
```



December was the best month for sales

Question2: What city had the highest number of sales?

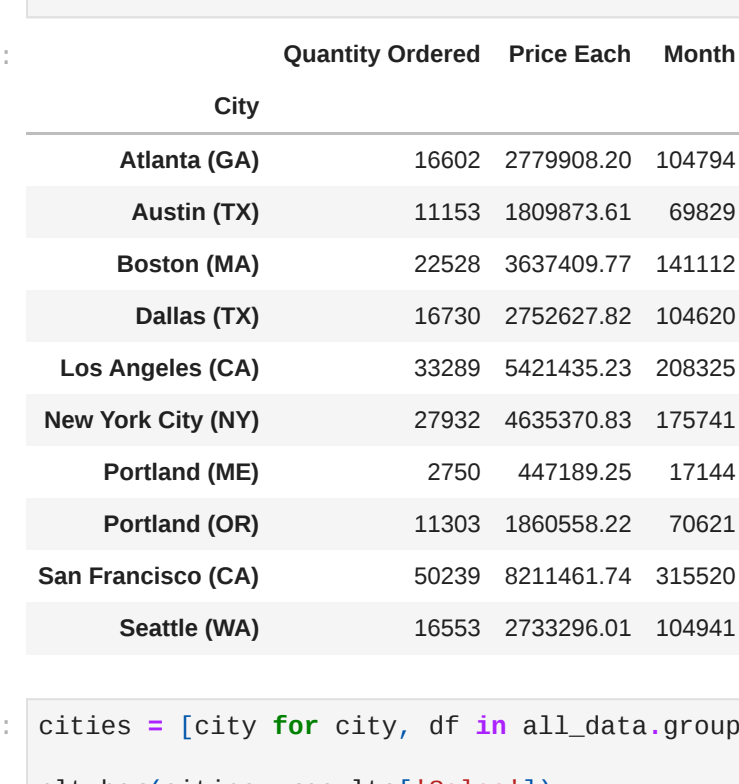
In [89]:
```python
results = all_data.groupby('City').sum()
results
```

Out[89]:

City	Quantity Ordered	Price Each	Month	Sales
Atlanta (GA)	16602	2779908.20	104794	2795498.58
Austin (TX)	11153	1809873.61	69829	1819581.75
Boston (MA)	22528	3637409.77	141112	3661642.01
Dallas (TX)	16730	2752627.82	104620	2767975.40
Los Angeles (CA)	33289	5421435.23	208325	5452570.80
New York City (NY)	27932	4635370.83	175741	4664317.43
Portland (ME)	2750	447189.25	17144	449758.27
Portland (OR)	11303	1860558.22	70621	1870732.34
San Francisco (CA)	50239	8211461.74	315520	8262203.91
Seattle (WA)	16553	2733296.01	104941	2747755.48

In [96]:
```python
cities = [city for city, df in all_data.groupby('City')]

plt.bar(cities, results['Sales'])
plt.xticks(cities, rotation='vertical', size=8)
plt.ylabel('Sales in USD ($)')
plt.xlabel('City name')
plt.ticklabel_format(style='plain' , axis='y')  #disable the scientific notation on the y-axis
plt.show()
```



San Francisco

Question3: What time should we display advertisements to maximize likelihood of customer's buying product?

In [97]:
```python
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])  #change into datetime object
all_data.head()
```

Out[97]:

	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 [100…
```python
all_data['Hour'] = all_data['Order Date'].dt.hour      #create Hour column
all_data['Minute'] = all_data['Order Date'].dt.minute  #create Minute column
all_data.head()
```

Out[100]:

	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 [112…
```python
hourly_counts = all_data.groupby('Hour').count()   #calculate the count in each hour
hourly_counts
```

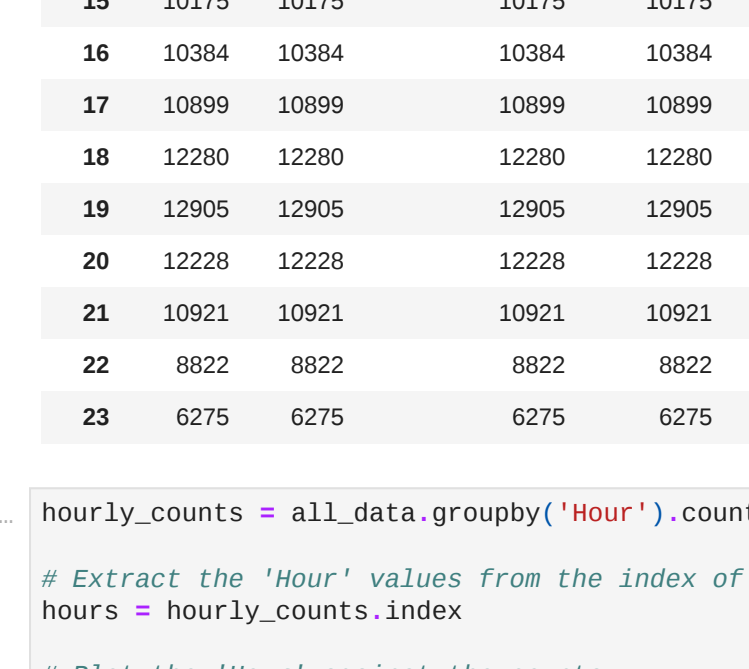
Out[112]:

Hour	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Minute
0	3910	3910	3910	3910	3910	3910	3910	3910	3910	3910
1	2350	2350	2350	2350	2350	2350	2350	2350	2350	2350
2	1243	1243	1243	1243	1243	1243	1243	1243	1243	1243
3	831	831	831	831	831	831	831	831	831	831
4	854	854	854	854	854	854	854	854	854	854
5	1321	1321	1321	1321	1321	1321	1321	1321	1321	1321
6	2482	2482	2482	2482	2482	2482	2482	2482	2482	2482
7	4011	4011	4011	4011	4011	4011	4011	4011	4011	4011
8	6256	6256	6256	6256	6256	6256	6256	6256	6256	6256
9	8748	8748	8748	8748	8748	8748	8748	8748	8748	8748
10	10944	10944	10944	10944	10944	10944	10944	10944	10944	10944
11	12411	12411	12411	12411	12411	12411	12411	12411	12411	12411
12	12587	12587	12587	12587	12587	12587	12587	12587	12587	12587
13	12129	12129	12129	12129	12129	12129	12129	12129	12129	12129
14	10984	10984	10984	10984	10984	10984	10984	10984	10984	10984
15	10175	10175	10175	10175	10175	10175	10175	10175	10175	10175
16	10384	10384	10384	10384	10384	10384	10384	10384	10384	10384
17	10899	10899	10899	10899	10899	10899	10899	10899	10899	10899
18	12280	12280	12280	12280	12280	12280	12280	12280	12280	12280
19	12905	12905	12905	12905	12905	12905	12905	12905	12905	12905
20	12228	12228	12228	12228	12228	12228	12228	12228	12228	12228
21	10921	10921	10921	10921	10921	10921	10921	10921	10921	10921
22	8822	8822	8822	8822	8822	8822	8822	8822	8822	8822
23	6275	6275	6275	6275	6275	6275	6275	6275	6275	6275

In [124…
```python
hourly_counts = all_data.groupby('Hour').count()

# Extract the 'Hour' values from the index of the resulting DataFrame
hours = hourly_counts.index

# Plot the 'Hour' against the counts
plt.plot(hours, hourly_counts['Quantity Ordered'])
plt.xticks(hours)
plt.grid()
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.title('Hourly Ordered Quantity')
plt.show()
# Around 11am (11) or 7pm (19)
```



Around 11am (11) or 7pm (19)

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

In [125…
```python
df = all_data[all_data['Order ID'].duplicated(keep=False)]   #check for duplicates as per order id
df.head(20)
```

Out[125]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
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
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	42
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	42
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
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	31
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	0
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	0
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	7
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	7
129	176681	Apple Airpods Headphones	1	150.00	2019-04-20 10:39:00	331 Cherry St, Seattle, WA 98101	4	150.00	Seattle (WA)	10	39
130	176681	ThinkPad Laptop	1	999.99	2019-04-20 10:39:00	331 Cherry St, Seattle, WA 98101	4	999.99	Seattle (WA)	10	39
138	176689	Bose SoundSport Headphones	1	99.99	2019-04-24 17:15:00	659 Lincoln St, New York City, NY 10001	4	99.99	New York City (NY)	17	15
139	176689	AAA Batteries (4-pack)	2	2.99	2019-04-24 17:15:00	659 Lincoln St, New York City, NY 10001	4	5.98	New York City (NY)	17	15
189	176739	34in Ultrawide Monitor	1	379.99	2019-04-05 17:38:00	730 6th St, Austin, TX 73301	4	379.99	Austin (TX)	17	38
190	176739	Google Phone	1	600.00	2019-04-05 17:38:00	730 6th St, Austin, TX 73301	4	600.00	Austin (TX)	17	38
225	176774	Lightning Charging Cable	1	14.95	2019-04-25 15:06:00	372 Church St, Los Angeles, CA 90001	4	14.95	Los Angeles (CA)	15	6
226	176774	USB-C Charging Cable	1	11.95	2019-04-25 15:06:00	372 Church St, Los Angeles, CA 90001	4	11.95	Los Angeles (CA)	15	6
233	176781	iPhone	1	700.00	2019-04-03 07:37:00	976 Hickory St, Dallas, TX 75001	4	700.00	Dallas (TX)	7	37
234	176781	Lightning Charging Cable	1	14.95	2019-04-03 07:37:00	976 Hickory St, Dallas, TX 75001	4	14.95	Dallas (TX)	7	37

In [127…
```python
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))   #create new column Grouped, contain string which is group by order id
df.head()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13964\759627908.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['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

Out[127]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute	Grouped
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	Google Phone,Wired Headphones
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	Google Phone,Wired Headphones
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	42	Google Phone,USB-C Charging Cable
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	42	Google Phone,USB-C Charging Cable
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	Bose SoundSport Headphones,Bose SoundSport Hea...

In [128…
```python
df = df[['Order ID', 'Grouped']].drop_duplicates()
df.head()
```

Out[128]:

	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 [135…
```python
from itertools import combinations
from collections import Counter
```

Question3: What products are most often sold together?

In [138…
```python
count = Counter()

for row in df['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))
 #iterates over each row in the 'Grouped' column, splits the row into individual product names, generates combinations of pairs of product names from each row,
#count.most_common(10)
for key, value in count.most_common(10):
    print(key, value)
```

```
('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
```

iphone and lighting charging cable in pair

In [139…
```python
count = Counter()

for row in df['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 3)))

#count.most_common(10)
for key, value in count.most_common(10):
    print(key, value)
```

```
('Google Phone', 'USB-C Charging Cable', 'Wired Headphones') 87
('iPhone', 'Lightning Charging Cable', 'Wired Headphones') 62
('iPhone', 'Lightning Charging Cable', 'Apple Airpods Headphones') 47
('Google Phone', 'USB-C Charging Cable', 'Bose SoundSport Headphones') 35
('Vareebadd Phone', 'USB-C Charging Cable', 'Wired Headphones') 33
('iPhone', 'Apple Airpods Headphones', 'Wired Headphones') 27
('Google Phone', 'Bose SoundSport Headphones', 'Wired Headphones') 24
('Vareebadd Phone', 'USB-C Charging Cable', 'Bose SoundSport Headphones') 16
('USB-C Charging Cable', 'Bose SoundSport Headphones', 'Wired Headphones') 5
('Vareebadd Phone', 'Bose SoundSport Headphones', 'Wired Headphones') 5
```

Question4: What Product sold the most? Why do you think it sold the most?

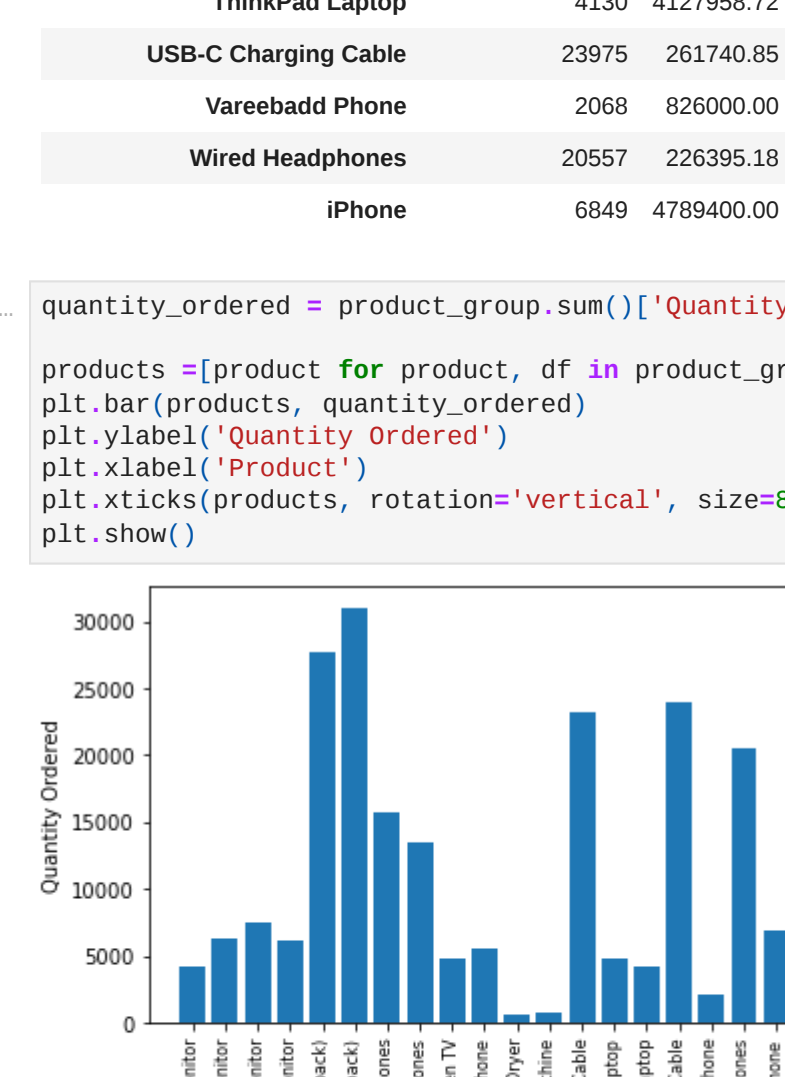
In [140…
```python
product_group = all_data.groupby('Product')
product_group.sum()
```

Out[140]:

Product	Quantity Ordered	Price Each	Month	Sales	Hour	Minute
20in Monitor	4129	451068.99	29336	454148.71	58764	122252
27in 4K Gaming Monitor	6244	2429637.70	44440	2435097.56	90916	184331
27in FHD Monitor	7550	1125974.93	52558	1132424.50	107540	219948
34in Ultrawide Monitor	6199	2348718.19	43304	2355558.01	89076	183480
AA Batteries (4-pack)	27635	79015.68	145558	106118.40	298342	609039
AAA Batteries (4-pack)	31017	61716.59	146370	92740.83	297332	612113
Apple Airpods Headphones	15661	2332350.00	109477	2349150.00	223304	455570
Bose SoundSport Headphones	13457	1332366.75	94113	1345565.43	192445	392603
Flatscreen TV	4819	1440000.00	34224	1445700.00	68815	142789
Google Phone	5532	3315000.00	38305	3319200.00	79479	162773
LG Dryer	646	387600.00	4383	387600.00	9326	19043
LG Washing Machine	666	399600.00	4523	399600.00	9785	19462
Lightning Charging Cable	23217	323787.10	153092	347094.15	312529	634442
Macbook Pro Laptop	4728	8030800.00	33548	8037600.00	68261	137574
ThinkPad Laptop	4130	4127958.72	28950	4129958.70	59746	121508
USB-C Charging Cable	23975	261740.85	154819	286501.25	314645	647586
Vareebadd Phone	2068	826000.00	14309	827200.00	29472	61835
Wired Headphones	20557	226395.18	133397	246478.43	271720	554023
iPhone	6849	4789400.00	47941	4794300.00	98657	201688

In [143…
```python
quantity_ordered = product_group.sum()['Quantity Ordered']

products =[product for product, df in product_group]
plt.bar(products, quantity_ordered)
plt.ylabel('Quantity Ordered')
plt.xlabel('Product')
plt.xticks(products, rotation='vertical', size=8)
plt.show()
```



In [144…
```python
prices = all_data.groupby('Product').mean()['Price Each'] #calculate average price
print(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
```

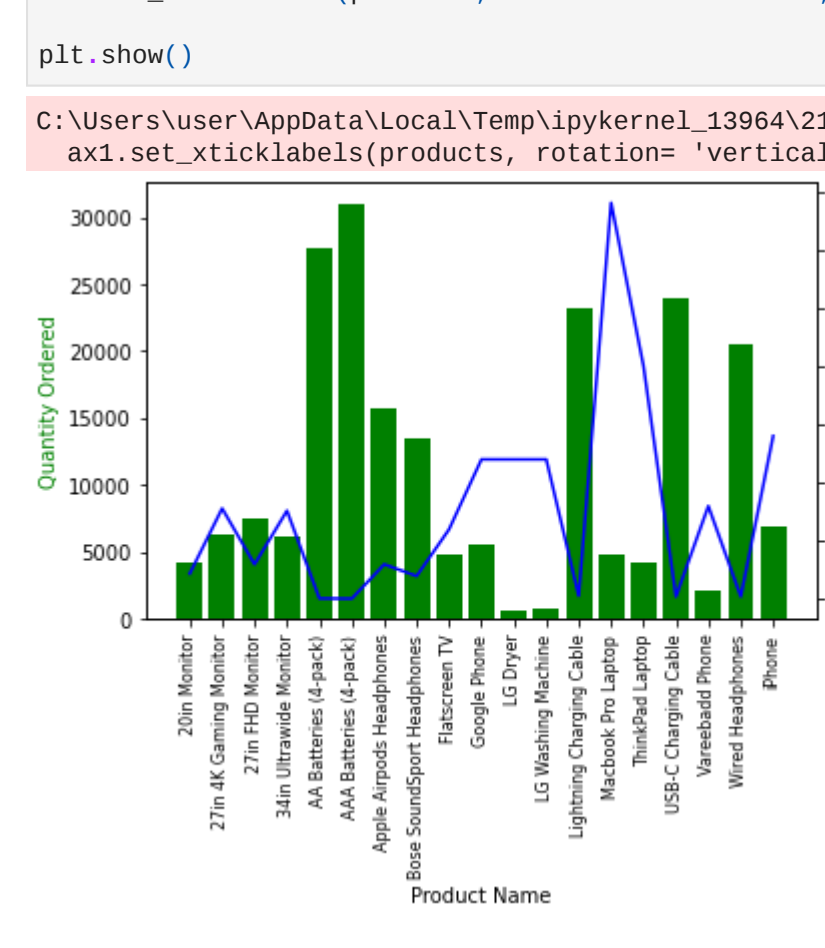
In [146…
```python
prices = all_data.groupby('Product').mean()['Price Each']
fig, ax1 = plt.subplots()

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

plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13964\2148397207.py:11: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(products, rotation= 'vertical', size=8)
```



AAA batteries as they are cheap and can be use in many gadgets and instruments or objects

In [ ]: