Sales Analysis

Import Necessary Libraries

```
In [1]: import pandas as pd


import os
```

Task_1: Merging the_12 monthes of sales data into a single CSV file

```
In [2]: df = pd.read_csv("C:/Users/KSA/Downloads/Python/Sales_Analysis/Sales_April_2019.csv")
files = [file for file in os.listdir("./Sales_Analysis")]
all_months_data = pd.DataFrame()

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

all_months_data.to_csv("all_data.csv",index=False)
```

Read in updated DataFrame

```
In [3]: all_data = pd.read_csv("all_data.csv")
all_data.head()
```

Out[3]:

	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

cleaning up the data!

Drop rows of NAN

```
In [4]: df = all_data[all_data.isna().any(axis=1)]

df.head()
```

Out[4]:

	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 [5]: all_data = all_data.dropna(how="all")

all_data.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
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

finding "Or" and delete it

```
In [6]: all_data = all_data [all_data["Order Date"].str[0:2]!="Or"]
```

Convert column to the correct type

```
In [7]: all_data ["Quantity Ordered"]=pd.to_numeric(all_data["Quantity Ordered"])
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

```
In [ ]:
```

Augment data with additional columns

Task_2 Adding month column

```
In [8]: 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

Task_3 Adding a sales column

```
In [9]: 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

Task_2: Add a city column

```
In [10]: 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)+ " (" +get_State(x)+") ")

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)

Question_1: What was the best month for sales? How much was earned that months?

```
In [11]: results= all_data.groupby("Month").sum()
all_data.head()
```

```
C:\Users\KSA\AppData\Local\Temp\ipykernel_7580\1224932306.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  results= all_data.groupby("Month").sum()
```

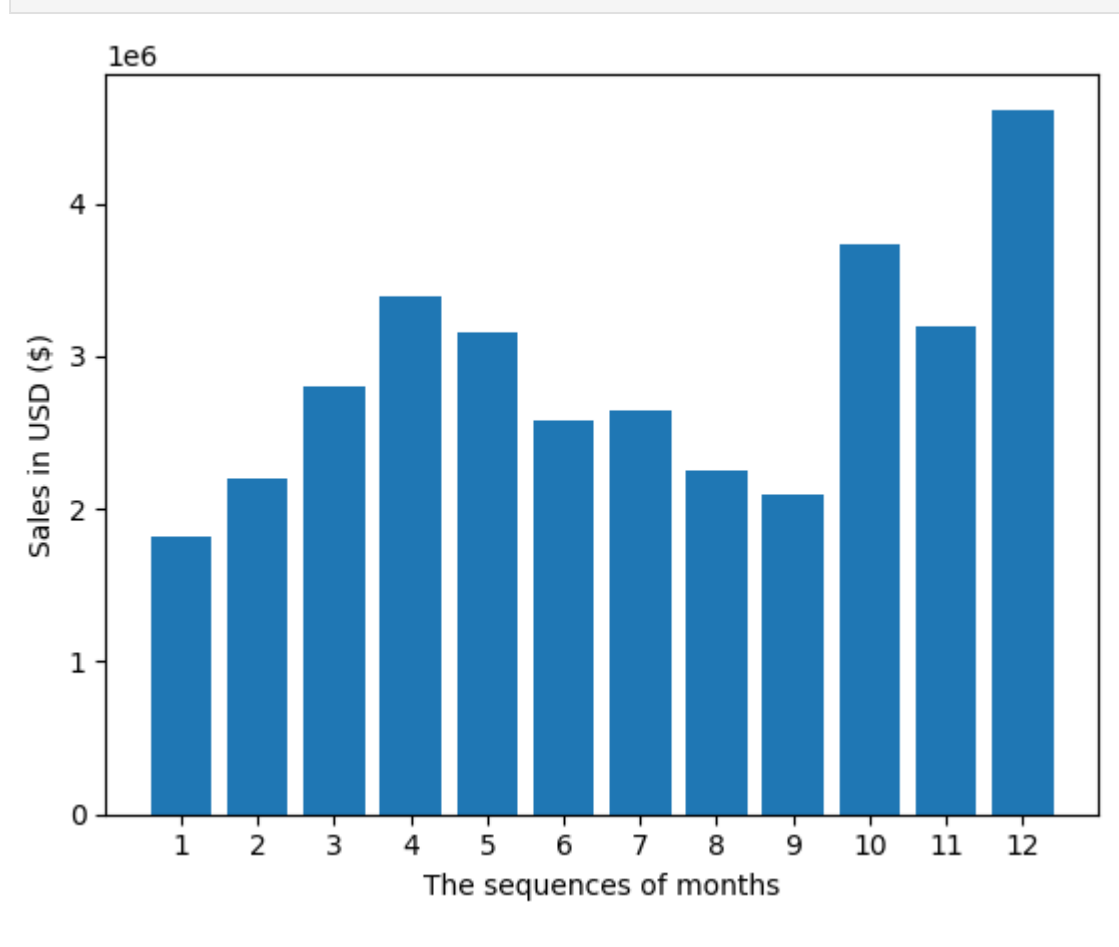
Out[11]:

	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 [50]: !pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (3.6.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.19 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.24.0)
Requirement already satisfied: packaging>=20.0 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (22.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\ksa\appdata\local\programs\python\python311\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
In [12]: import matplotlib.pyplot as plt
Month = range(1,13)
plt.bar(Month,results["Sales"])
plt.xticks(Month)
plt.ylabel("Sales in USD ($)")
plt.xlabel("The sequences of months")
plt.show()
```



Question_1: What was the best month for sales? How much was earned that months? as you can see in the chart the best month is (December). The reveution in December was 4,613,443.444 USD.

The first QS has been answered correctly

Question_2: what city had the highest number of sales ??

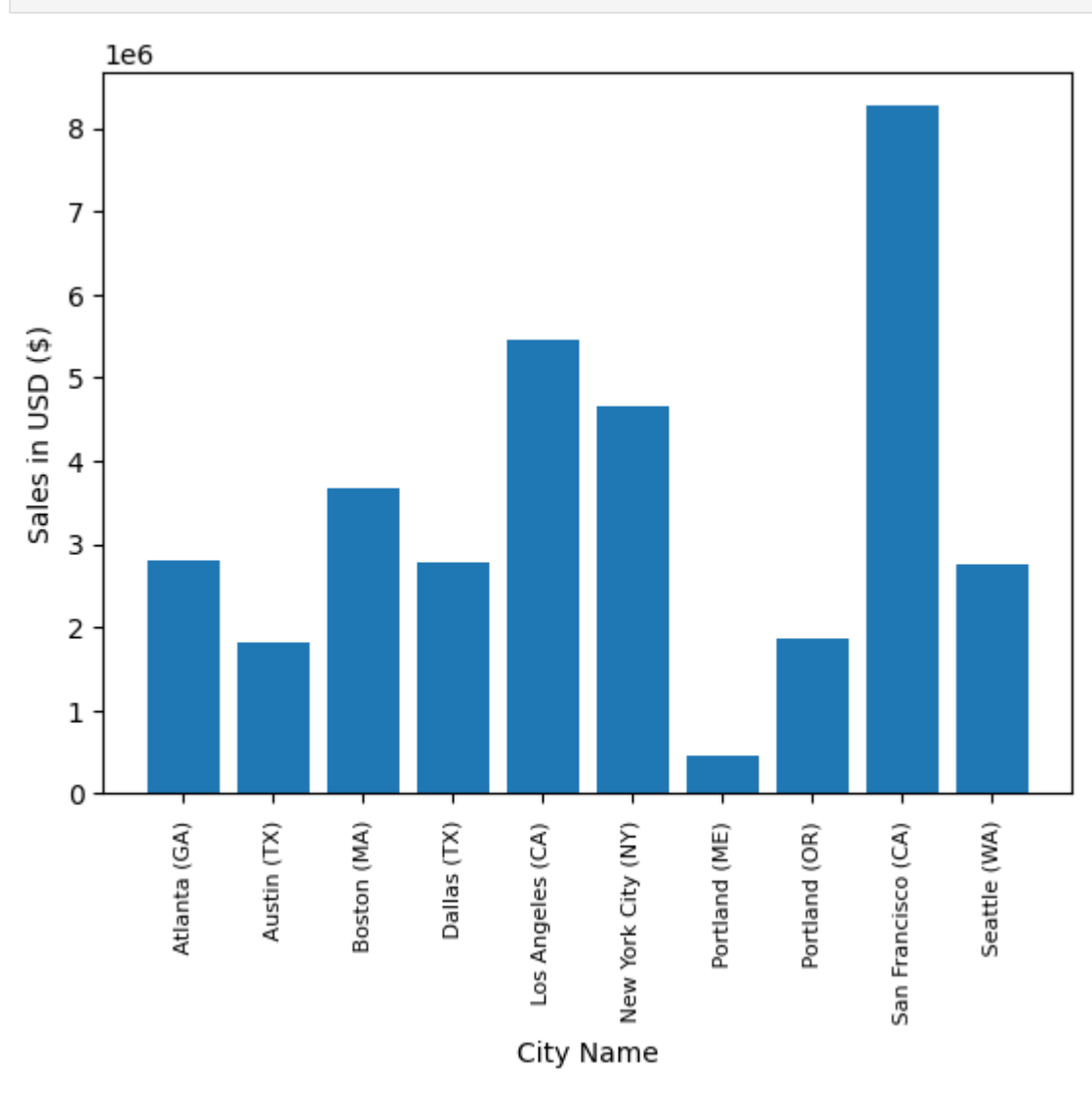
```
In [13]: results= all_data.groupby('City').sum()
results
```

```
C:\Users\KSA\AppData\Local\Temp\ipykernel_7580\3152931966.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  results= all_data.groupby('City').sum()
```

Out[13]:

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 [14]: import matplotlib.pyplot as plt
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.show()
```



Question_3 : What is the best time to display the adverstisements likelihood of customer's buying product?

```
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	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 [15]: all_data["Order Date"]= pd.to_datetime(all_data["Order Date"])
all_data.head()
```

Out[15]:

	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 [16]: all_data["Hour"]=all_data["Order Date"].dt.hour
all_data["Minute"]=all_data["Order Date"].dt.minute
all_data.head()
```

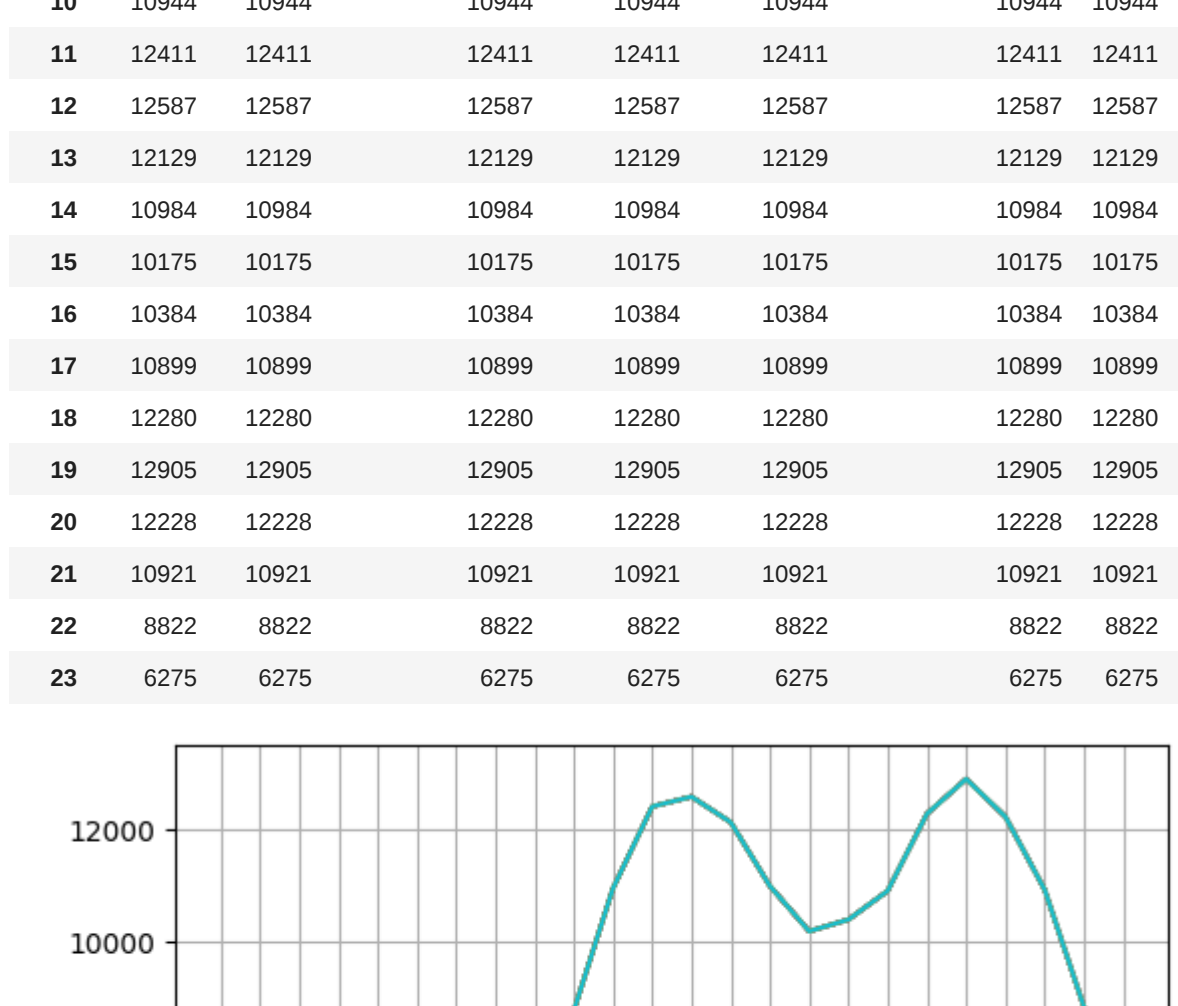
Out[16]:

	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 [17]: Hours = [hour for hour, df in all_data.groupby("Hour")]

plt.plot(Hours,all_data.groupby(["Hour"]).count())
plt.xlabel("The hours")
plt.ylabel("Orders quantity")
plt.xticks(Hours)
plt.grid()



all_data.groupby(["Hour"]).count()
```

Out[17]:

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



What products are most often sold together ?

```
In [24]: all_data.head()
```

Out[24]:

	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 [18]: df= all_data[all_data["Order ID"].duplicated(keep=False)]
df["Grouped"]= df.groupby("Order ID")["Product"].transform(lambda x :",".join(x))

df= df[["Order ID","Grouped"]].drop_duplicates()

df.head()
```

```
C:\Users\KSA\AppData\Local\Temp\ipykernel_7580\848779116.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["Grouped"]= df.groupby("Order ID")["Product"].transform(lambda x :",".join(x))
```

Out[18]:

	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 [19]: 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(20):
    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
('Lightning Charging Cable', 'Apple Airpods Headphones') 81
('Vareebadd Phone', 'Bose SoundSport Headphones') 80
('USB-C Charging Cable', 'Bose SoundSport Headphones') 77
('Apple Airpods Headphones', 'Wired Headphones') 69
('Lightning Charging Cable', 'USB-C Charging Cable') 58
('Lightning Charging Cable', 'AA Batteries (4-pack)') 55
('Lightning Charging Cable', 'Lightning Charging Cable') 54
('Bose SoundSport Headphones', 'Wired Headphones') 53
('AA Batteries (4-pack)', 'Lightning Charging Cable') 51
('AAA Batteries (4-pack)', 'USB-C Charging Cable') 50
```

What product sold the most? why do you think it sold the most?

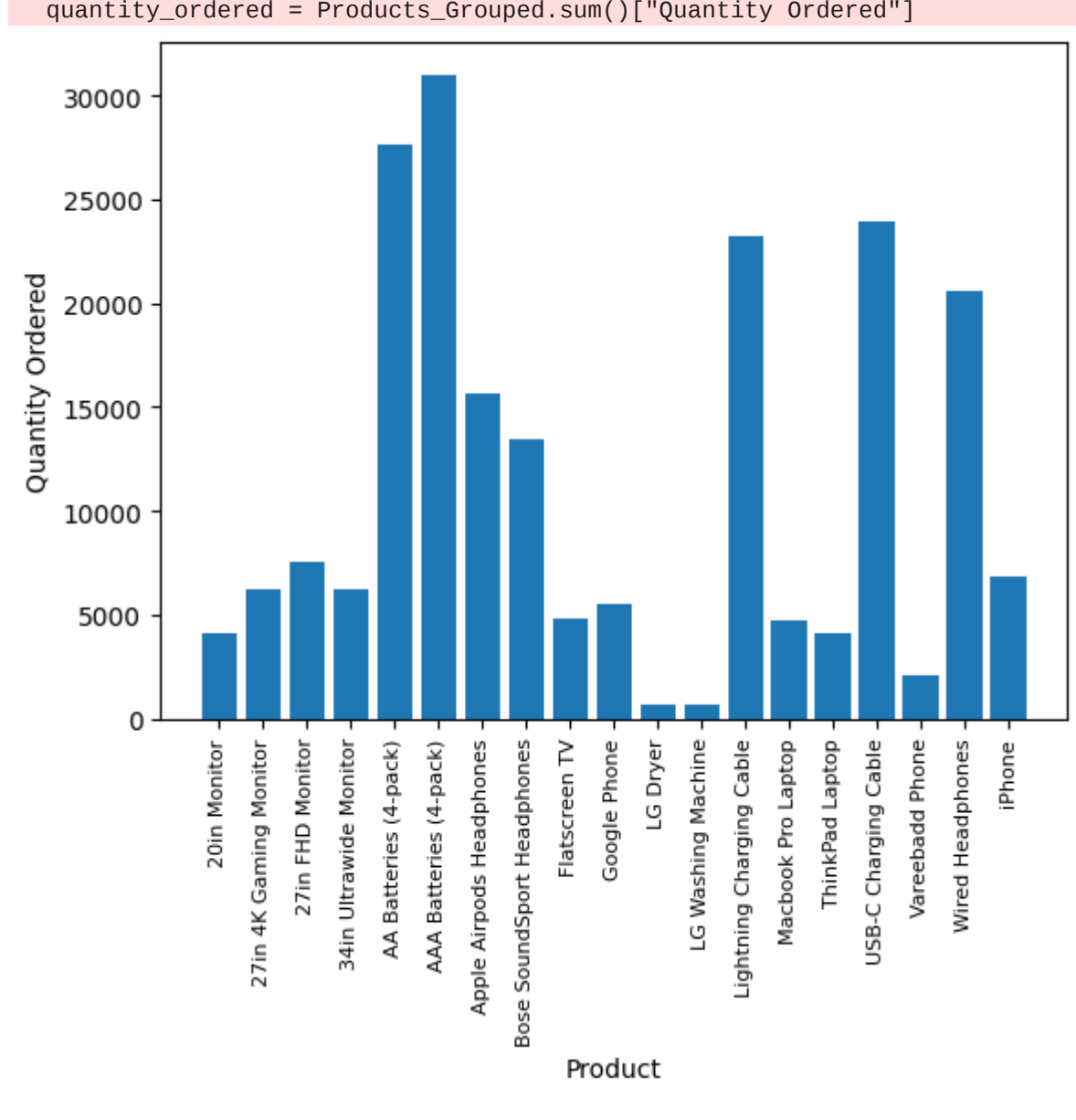
```
In [20]: all_data.head()
```

Out[20]:

	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

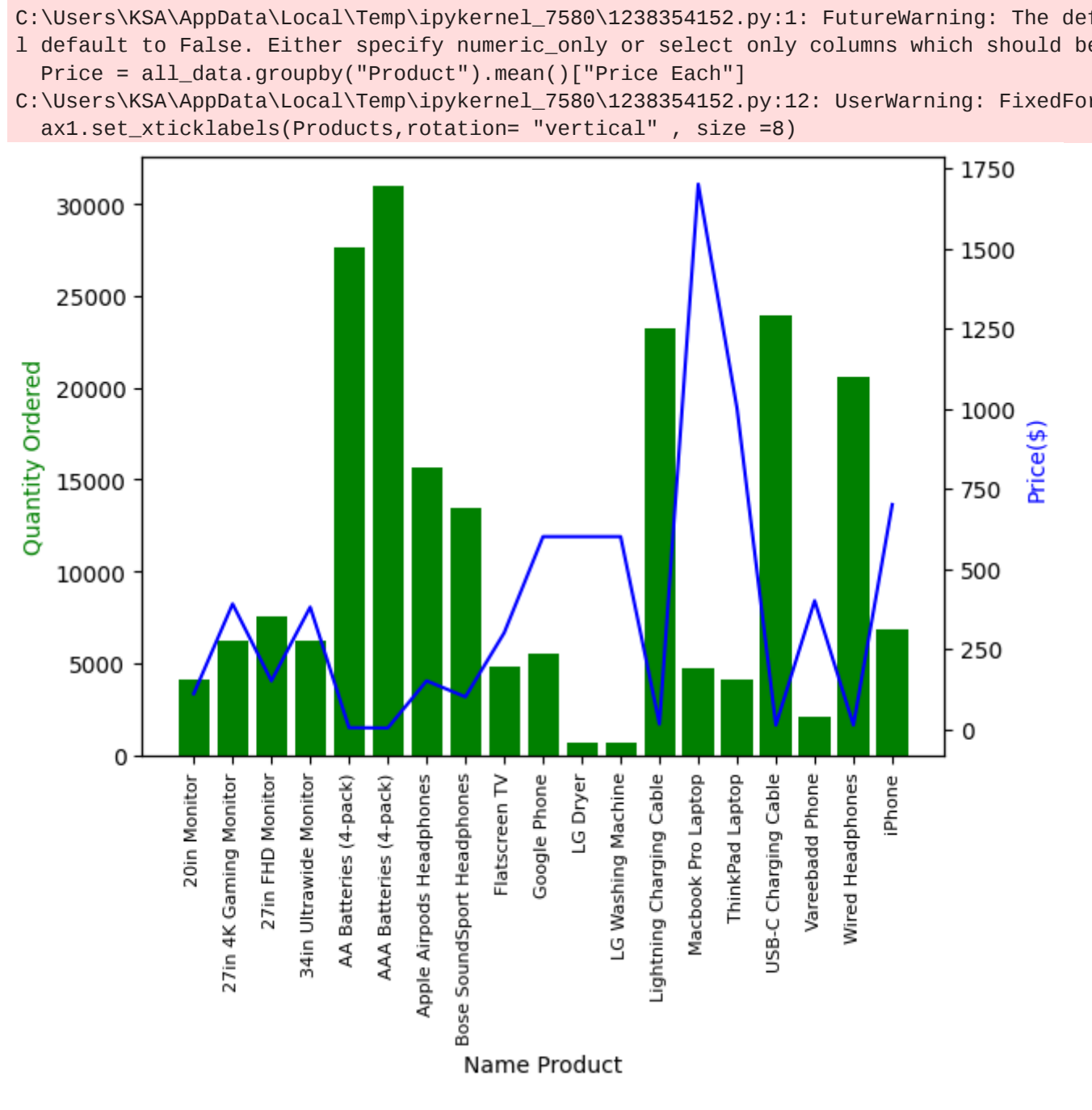
```
In [29]: Products_Grouped = all_data.groupby("Product")


quantity_ordered = Products_Grouped.sum()["Quantity Ordered"]

Products = [Product for Product, df in Products_Grouped]

plt.bar(Products,quantity_ordered)
plt.ylabel("Quantity Ordered")
plt.xlabel("Product")
plt.xticks(Products,rotation= "vertical" , size =8)

plt.show()
```

```
C:\Users\KSA\AppData\Local\Temp\ipykernel_7580\1507785933.py:4: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  quantity_ordered = Products_Grouped.sum()["Quantity Ordered"]
```




```
In [38]: Price = all_data.groupby("Product").mean()["Price Each"]

fig, ax1 = plt.subplots()

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

ax1.set_xlabel('Name Product')
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\KSA\AppData\Local\Temp\ipykernel_7580\1238354152.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only wil
l default to False. Either specify numeric_only or select only columns which should be valid for the function.
  Price = all_data.groupby("Product").mean()["Price Each"]
C:\Users\KSA\AppData\Local\Temp\ipykernel_7580\1238354152.py:12: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(Products,rotation= "vertical" , size =8)
```




```
In [ ]:
```