Sales Analysis [Real World Data] By Aman Kumar

Import Necessary Libraries

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

In [2]:
```
files =[file for file in os.listdir('./Sales')]

all_months_data = pd.DataFrame()

for file in files:
    pdf = pd.read_csv("./Sales/"+file)
    all_months_data = pd.concat([all_months_data, pdf] ,axis=0,ignore_index=True)

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

Task 1: Clean up the data

Drop rows of NAN

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

all_data = all_data.dropna(how='all')
all_data.head(517)
```

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
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
...	...	...	...	...	...	...
514	177050	Apple Airpods Headphones	1	150	04/08/19 09:53	510 Elm St, Boston, MA 02215
515	177051	Wired Headphones	2	11.99	04/07/19 08:41	777 Adams St, Boston, MA 02215
516	177052	USB-C Charging Cable	2	11.95	04/02/19 09:30	532 Walnut St, San Francisco, CA 94016
517	177053	Wired Headphones	1	11.99	04/24/19 20:45	5 Adams St, Boston, MA 02215
518	177054	Apple Airpods Headphones	1	150	04/09/19 19:18	800 Jackson St, Atlanta, GA 30301

517 rows × 6 columns

Find 'Or' and delete it

In [5]:
```
temp_df = all_data[all_data['Order Date'].str[0:2] != 'Or']
```

Convert columns to be correct type

In [6]:
```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 186305 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: 9.9+ MB
```

In [7]:
```
all_data.head(190)
```

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	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
...	...	...	...	...	...	...
186	176736	AA Batteries (4-pack)	1	3.84	04/17/19 11:48	421 2nd St, Los Angeles, CA 90001
187	176737	Apple Airpods Headphones	1	150	04/07/19 01:28	109 Center St, Los Angeles, CA 90001
188	176738	AA Batteries (4-pack)	1	3.84	04/03/19 13:17	661 Center St, San Francisco, CA 94016
189	176739	34in Ultrawide Monitor	1	379.99	04/05/19 17:38	730 6th St, Austin, TX 73301
190	176739	Google Phone	1	600	04/05/19 17:38	730 6th St, Austin, TX 73301

190 rows × 6 columns

Removing Rows that contain string

In [8]:
```
all_data [all_data['Price Each'] == 'Price Each']
```

Out[8]:

	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
...	...	...	...	...	...	...
185164	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
185551	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
186563	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
186632	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
186738	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

355 rows × 6 columns

In [9]:
```
# To remove rows that contain String
all_data = all_data.loc[~all_data['Quantity Ordered'].str.contains('Quantity Ordered')]
```

Convert columns Datatypes

In [10]:
```
all_data[["Quantity Ordered", "Price Each"]] = all_data[["Quantity Ordered", "Price Each

all_data.head()
```

Out[10]:

	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 [11]:
```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 6 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
dtypes: float64(1), int64(1), object(4)
memory usage: 9.9+ MB
```

Task 2: Add Month Column

In [12]:
```
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month']
all_data.head()
```

Out[12]:

	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	04
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

Task 3: Add a sales column

In [13]:
```
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
all_data.head()
```

Out[13]:

	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	04	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04	11.99

Task 4: Add a city column

In [14]:
```
# Lets use the .apply()
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_st

all_data.head()
```

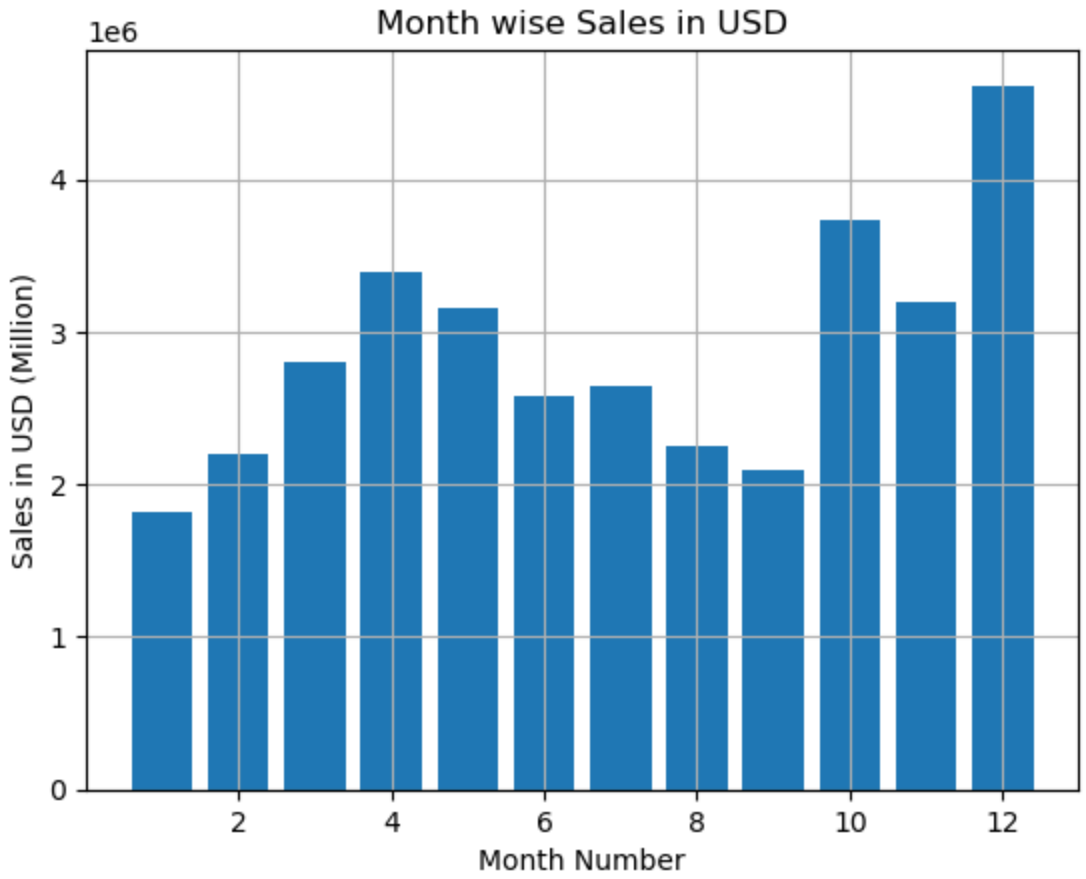
Out[14]:

	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	04	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04	11.99	Los Angeles (CA)

Question 1: What was the best month for sales? How much was earned that month?

In [15]:
```
results = all_data.groupby('Month').sum()
```

In [16]:
```
import matplotlib.pyplot as plt

months = range(1,13)
plt.bar(months, results['Sales'])
plt.title('Month wise Sales in USD')
plt.ylabel('Sales in USD (Million)')
plt.xlabel('Month Number')
plt.grid()
plt.show()
```



Heighest sales recorded in December, Total Sales for december month crossed 4.5 million USD.

Question 2: What City has the highest number of Sales ?

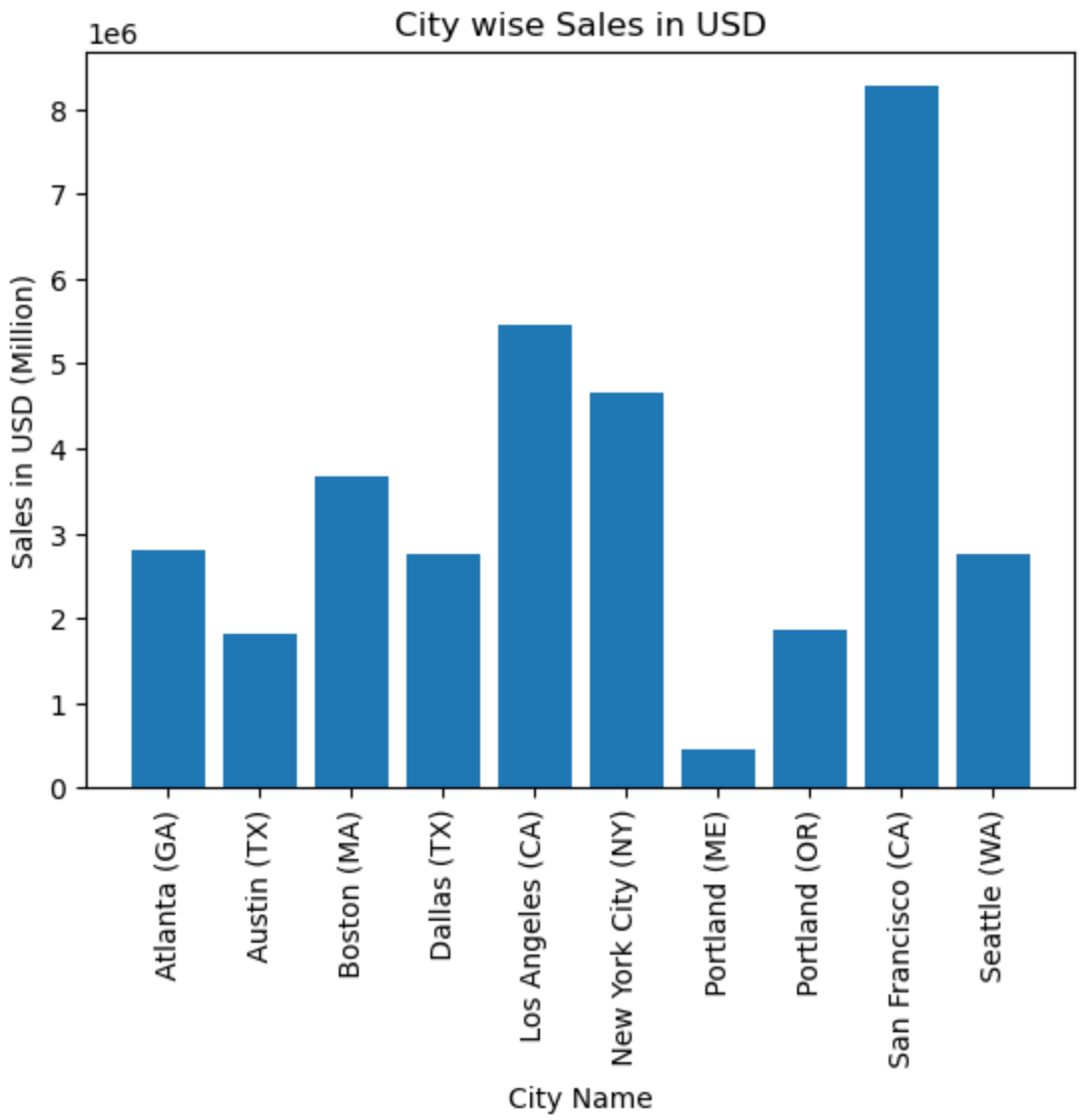
In [17]:
```
results = all_data.groupby('City').sum()
results
```

Out[17]:

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

In [18]:

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

plt.bar(cities, results['Sales'])
plt.title('City wise Sales in USD')
plt.ylabel('Sales in USD (Million)')
plt.xlabel('City Name')
plt.xticks(rotation=90)
plt.show()
```



San Francisco (CA) has the heighest sales.

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

In [19]:
```
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

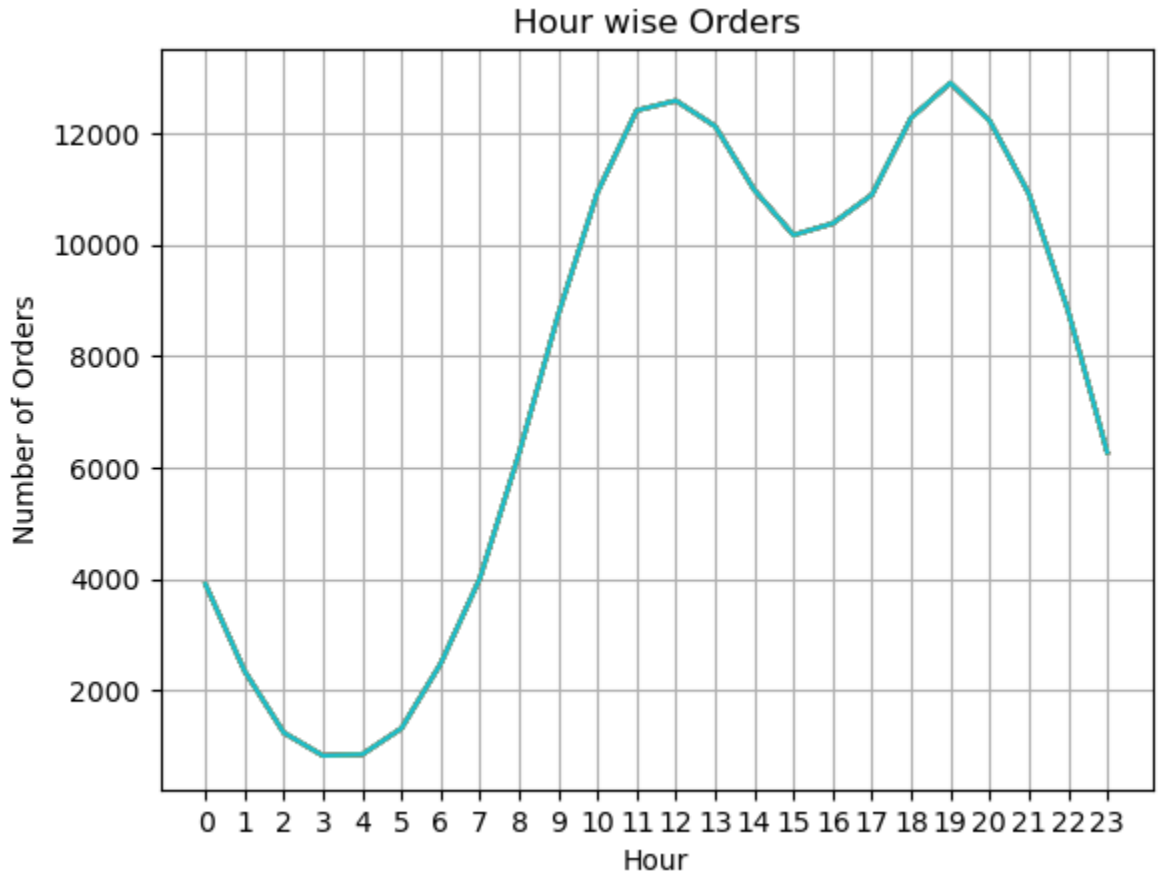
In [20]:
```
all_data['Hour'] = all_data['Order Date'].dt.hour
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data.head()
```

Out[20]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	04	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	04	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	04	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	04	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	04	11.99	Los Angeles (CA)	9	27

In [21]:

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

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



We should display advertisements Between 11am to 12 noon and between 6 to 7 PM to maximize likelihood ...ying product

Question 4: What products are most often sold together ?

In [22]:
```
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\amanc\AppData\Local\Temp\ipykernel_21052\4061286189.py:3: SettingWithCopyWarnin
g:
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[22]:

	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 [23]:
```
from itertools import combinations
from collections import Counter
```

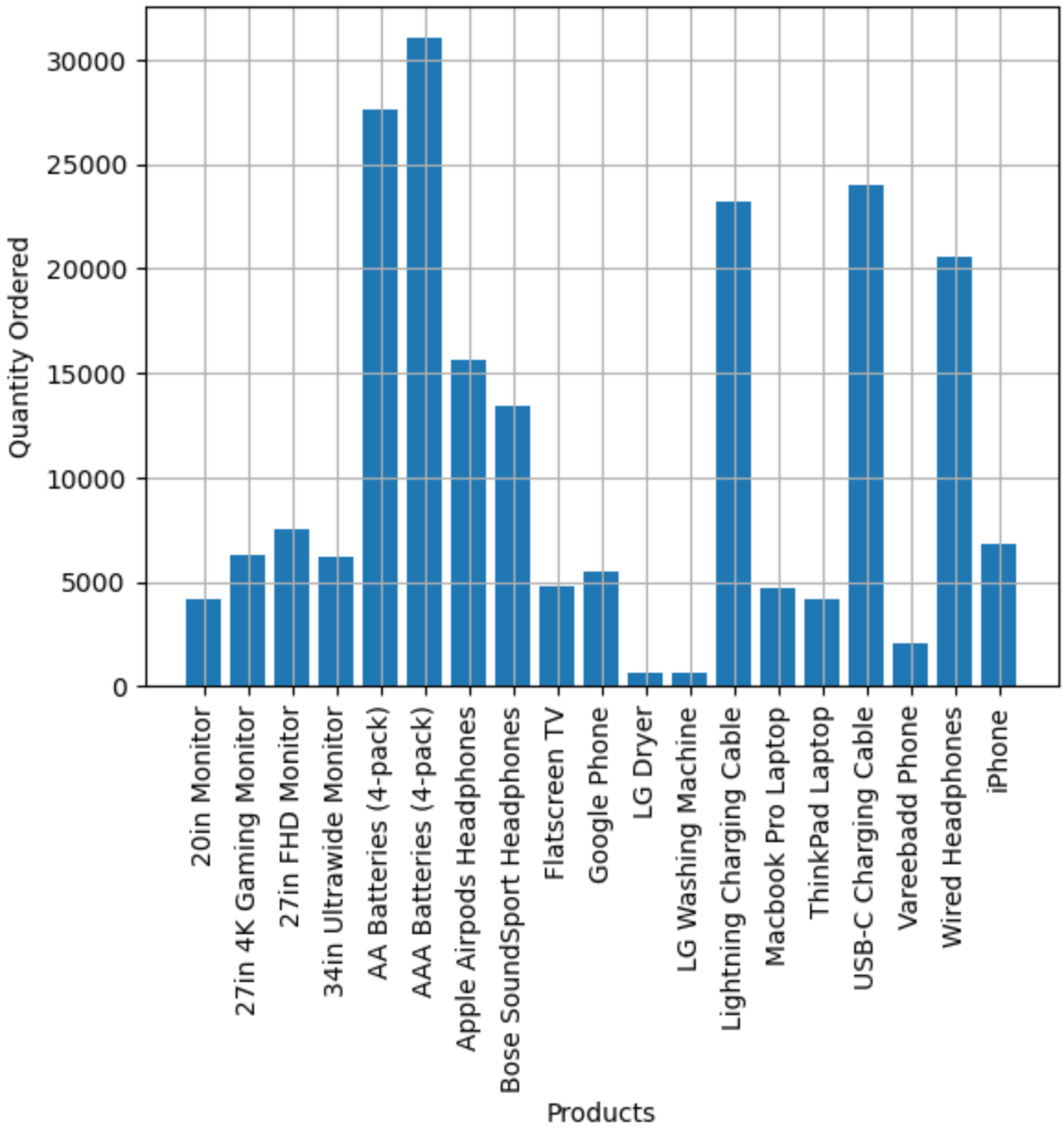
In [24]:
```
count = Counter()

for row in df['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

count.most_common(10)
```

Out[24]:
```
[(('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)]
```

Question 5: What product sold the most ? Why do you think it sold the most ?

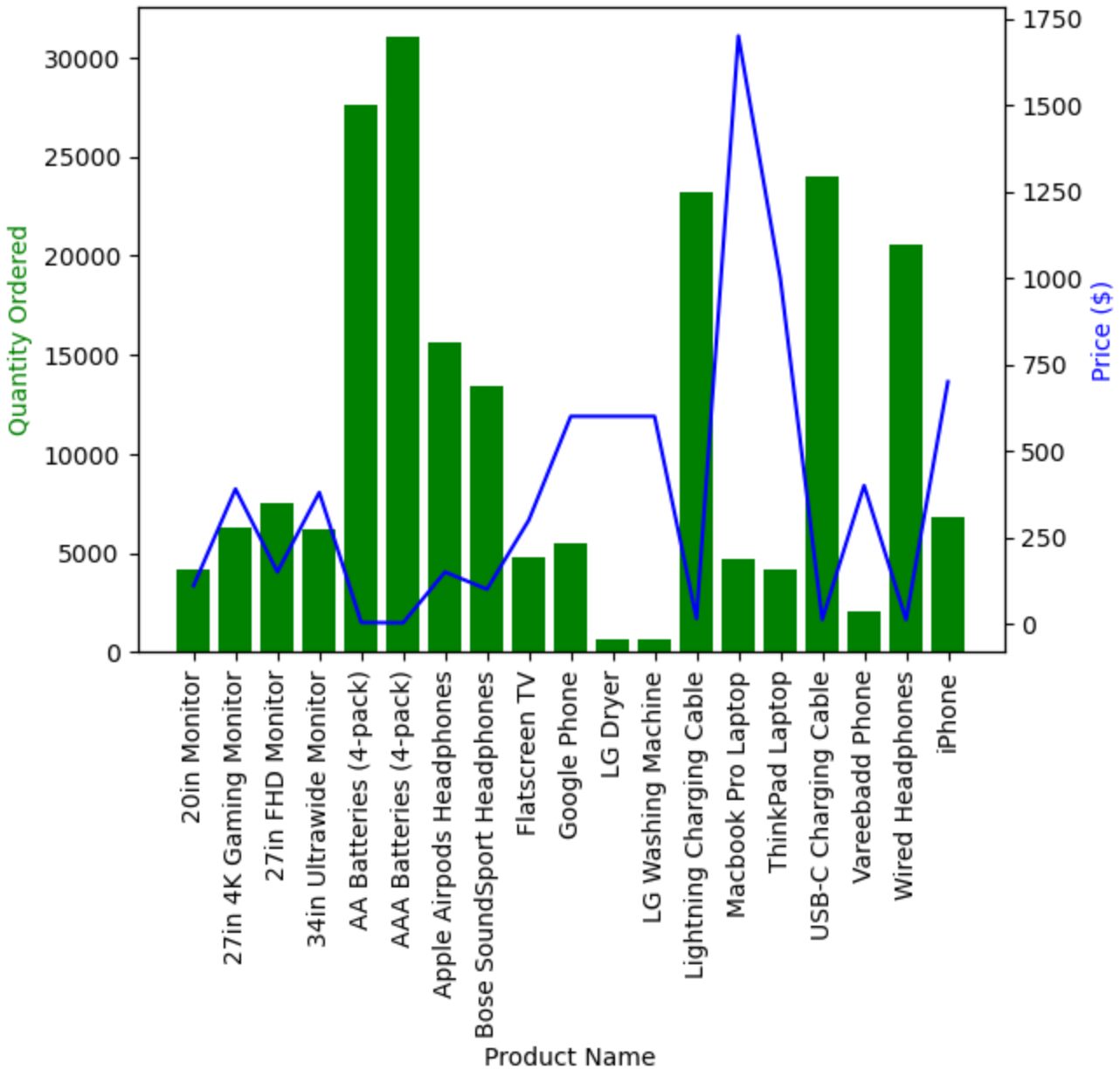
In [25]:
```
product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']

products = [products for products, df in product_group]
```

In [26]:
```
plt.bar(products, quantity_ordered)
plt.xticks(products, rotation=90)
plt.ylabel('Quantity Ordered')
plt.xlabel('Products')
plt.grid()
plt.show()
```



In [27]:
```
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=90)

plt.show()
```

```
C:\Users\amanc\AppData\Local\Temp\ipykernel_21052\4073329059.py:12: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(products, rotation=90)
```



Batteries have the heighest sales. As I can see, the items with less price have high sales and vice-versa.

In [ ]: