In [ ]:

1.. What is Feather & how to read data from data !

... Feather is a binary file format that is used for storing data Feather is a fast, lightweight, and easy-to-use binary file format for storing data # It shows high I/O speed, doesn't take too much memory on the disk and doesn't need any unpacking when loaded back into RAM. ...

In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

... In order to work with feather files , do install pyarrow package ...

In [ ]:

In [2]:
```
!pip install pyarrow
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pyarrow in d:\windows security\new folder\lib\site-pac
kages (11.0.0)
Requirement already satisfied: numpy>=1.16.6 in d:\windows security\new folder\lib\si
te-packages (from pyarrow) (1.24.3)
```

In [ ]:

In [3]:
```
all_data = pd.read_feather(r"C:\Users\User\E-commerce Sales_data analysis/Sales_data.f
```

In [4]:
```
all_data.head(6)
```

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	None	None	None	None	None	None
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 [5]:
```
all_data.isnull().sum()
```

Out[5]:
```
Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
Order Date          545
Purchase Address    545
dtype: int64
```

''' Insight: The data contains hundreds of thousands of electronics store purchases broken down by month, product type, cost , purchase address, etc '''

In [ ]:
```
```

Data cleaning and formatting

In [6]:
```
all_data = all_data.dropna(how="all")
```

In [80]:
```
all_data.isnull().sum() ##checking out total missing values we have
```

Out[80]:
```
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
Month               0
sales               0
city                0
dtype: int64
```

... check whether u have duplicate rows or not ...

In [8]:
```
all_data.duplicated()
```

Out[8]:
```
0         False
2         False
3         False
4         False
5         False
          ...
186845    False
186846    False
186847    False
186848    False
186849    False
Length: 186305, dtype: bool
```

In [9]:
```
all_data[all_data.duplicated()] ## total 618 duplicate rows ..
```

Out[9]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
31	176585	Bose SoundSport Headphones	1	99.99	04/07/19 11:31	823 Highland St, Boston, MA 02215
1149	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1155	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1302	177795	Apple Airpods Headphones	1	150	04/27/19 19:45	740 14th St, Seattle, WA 98101
1684	178158	USB-C Charging Cable	1	11.95	04/28/19 21:13	197 Center St, San Francisco, CA 94016
...	...	...	...	...	...	...
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
186782	259296	Apple Airpods Headphones	1	150	09/28/19 16:48	894 6th St, Dallas, TX 75001
186785	259297	Lightning Charging Cable	1	14.95	09/15/19 18:54	138 Main St, Boston, MA 02215

618 rows × 6 columns

In [10]:
```
all_data = all_data.drop_duplicates() ## Dropping all the duplicate rows ..
```

In [11]:
```
all_data.shape
```

Out[11]:
```
(185687, 6)
```

In [12]:
```
all_data [all_data.duplicated()]
```

Out[12]:

Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

In [ ]:

In [ ]:

2.. Which is the best month for sale ?

... Lets first understand what this term 'best' is all about : if any month has maximum sales, we will consider that as best ...

In [13]:
```
all_data.head(2)
```

Out[13]:

	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

In [ ]:

In [14]:
```
all_data['Order Date'][0]
```

Out[14]:
```
'04/19/19 08:46'
```

In [15]:
```
'04/19/19 08:46'.split(' ')
```

Out[15]:
```
['04/19/19', '08:46']
```

In [16]:
```
'04/19/19 08:46'.split(' ') [0]
```

Out[16]:
```
'04/19/19'
```

In [17]:
```
'04/19/19 08:46'.split(' ') [0].split('/')[0]
```

Out[17]:
```
'04'
```

In [18]:
```
def return_month(x):
    return x.split('/')[0]
```

In [19]:
```
all_data['Month'] = all_data['Order Date'].apply(return_month)
```

In [20]:
```
all_data.dtypes ## applying return_month function on top of "Order Date" feature
```

Out[20]:
```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
Month               object
dtype: object
```

In [22]:
```
all_data['Month'].unique()
```

Out[22]:
```
array(['04', '05', 'Order Date', '08', '09', '12', '01', '02', '03', '07',
       '06', '11', '10'], dtype=object)
```

In [23]:
```
filter1 = all_data['Month'] == 'Order Date'
```

In [24]:
```
all_data[~filter1]
```

Out[24]:

	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	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
...	...	...	...	...	...	...	...
186845	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	09
186846	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	09
186847	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016	09
186848	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	09
186849	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	09

185686 rows × 7 columns

In [25]:
```
all_data = all_data[~filter1] ## excluding all those rows which have entry as "Order D
```

In [26]:
```
all_data.shape
```

Out[26]:
```
(185686, 7)
```

In [27]:
```
all_data['Month'].astype(int)
```

Out[27]:
```
0         4
2         4
3         4
4         4
5         4
         ..
186845    9
186846    9
186847    9
186848    9
186849    9
Name: Month, Length: 185686, dtype: int32
```

... use warnings package to get rid of any warnings ...

In [28]:
```
import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

In [29]:
```
all_data['Month'] = all_data['Month'].astype(int)
```

In [30]:
```
all_data.dtypes
```

Out[30]:
```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
Month                int32
dtype: object
```

In [31]:
```
all_data['Quantity Ordered'] = all_data['Quantity Ordered'].astype(int)
all_data['Price Each'] = all_data['Price Each'].astype(float)
```

In [32]:
```
all_data.dtypes
```

Out[32]:
```
Order ID             object
Product              object
Quantity Ordered      int32
Price Each          float64
Order Date           object
Purchase Address     object
Month                 int32
dtype: object
```

In [77]:
```
all_data['sales'] = all_data['Quantity Ordered'] * all_data['Price Each'] ## creating
```

In [34]:
```
all_data['sales']
```

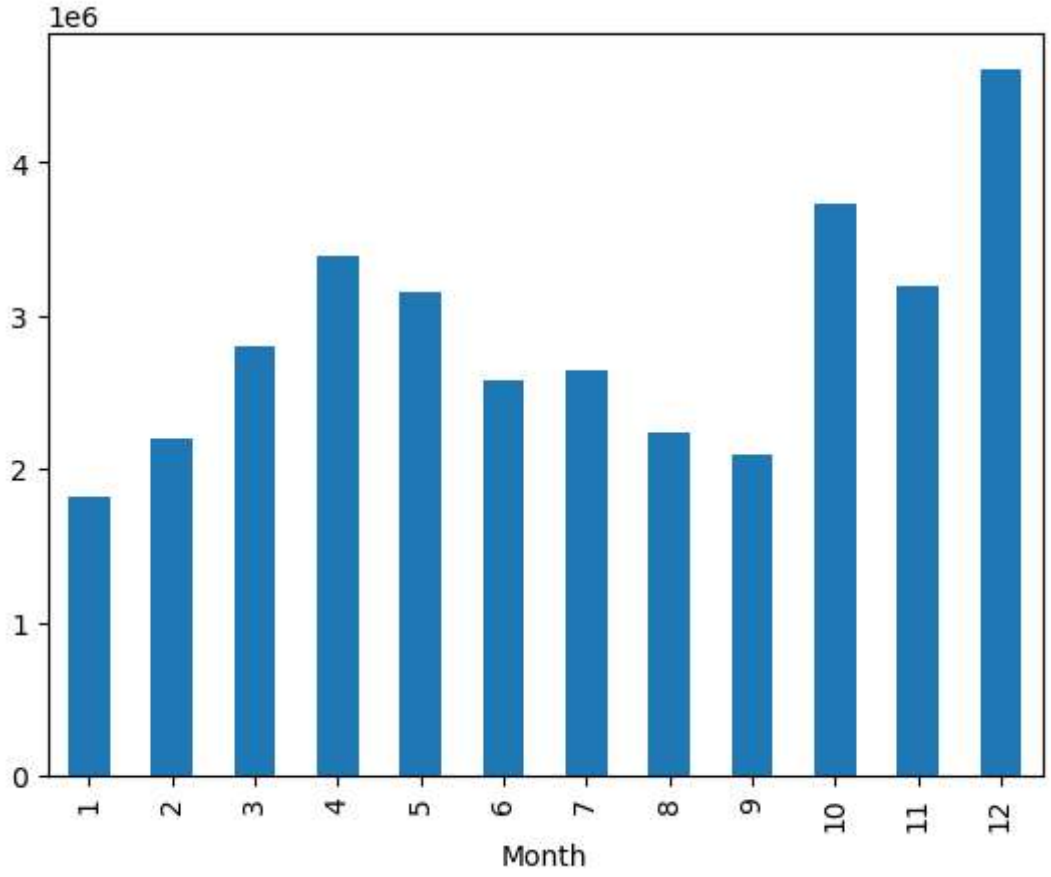
Out[34]:
```
0          23.90
2          99.99
3         600.00
4          11.99
5          11.99
           ...
186845      8.97
186846    700.00
186847    700.00
186848    379.99
186849     11.95
Name: sales, Length: 185686, dtype: float64
```

In [35]:
```
all_data.groupby(['Month'])['sales'].sum()
```

Out[35]:
```
Month
1     1821413.16
2     2200078.08
3     2804973.35
4     3389217.98
5     3150616.23
6     2576280.15
7     2646461.32
8     2241083.37
9     2094465.69
10    3734777.86
11    3197875.05
12    4608295.70
Name: sales, dtype: float64
```

In [36]:
```
all_data.groupby(['Month'])['sales'].sum().plot(kind='bar')
```

Out[36]:
```
<Axes: xlabel='Month'>
```



In [ ]:

... Insight: December is the best month of sales '''

In [ ]:

3.. Which city has max order ?

In [ ]:

In [37]:
```
all_data.head(2)
```

Out[37]:

	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

In [38]:
```
all_data['Purchase Address'][0]
```

Out[38]:
```
'917 1st St, Dallas, TX 75001'
```

In [78]:
```
all_data['Purchase Address'][0].split(',')[1] ## extracting city from "Purchase Addres
```

Out[78]:
```
' Dallas'
```

In [ ]:
```
```

In [40]:
```
all_data['city'] = all_data['Purchase Address'].str.split(',').str.get(1)
```

In [41]:
```
all_data['city']
```

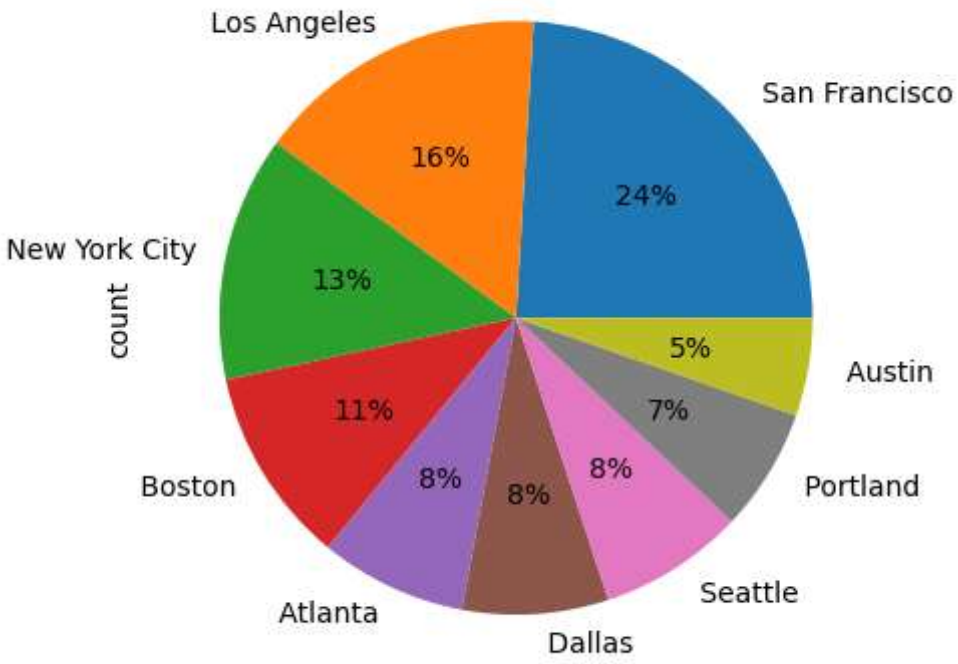
Out[41]:
```
0                 Dallas
2                 Boston
3            Los Angeles
4            Los Angeles
5            Los Angeles
                ...
186845       Los Angeles
186846     San Francisco
186847     San Francisco
186848     San Francisco
186849     San Francisco
Name: city, Length: 185686, dtype: object
```

In [79]:
```
pd.value_counts(all_data['city']) ## frequency table..
```

Out[79]:
```
city
 San Francisco    44662
 Los Angeles      29564
 New York City    24847
 Boston           19901
 Atlanta          14863
 Dallas           14797
 Seattle          14713
 Portland         12449
 Austin            9890
Name: count, dtype: int64
```

In [43]:
```
pd.value_counts(all_data['city']).plot(kind='pie' , autopct = '%1.0f%%') ## Pandas pie
```

Out[43]:
```
<Axes: ylabel='count'>
```



In [ ]:

''' Insight: New York , Los Angeles , San Francisco are the Top 3 cities which has max order '''

In [ ]:

4.. What product sold the most & Why?

In [ ]:

In [44]:
```
all_data.columns
```

Out[44]:
```
Index(['Order ID', 'Product', 'Quantity Ordered', 'Price Each', 'Order Date',
       'Purchase Address', 'Month', 'sales', 'city'],
      dtype='object')
```

In [45]:
```
count_df = all_data.groupby(['Product']).agg({'Quantity Ordered':'sum' , 'Price Each':
```

In [46]:
```
count_df = count_df.reset_index()
```

In [47]:
```
count_df
```

Out[47]:

	Product	Quantity Ordered	Price Each
0	20in Monitor	4126	109.99
1	27in 4K Gaming Monitor	6239	389.99
2	27in FHD Monitor	7541	149.99
3	34in Ultrawide Monitor	6192	379.99
4	AA Batteries (4-pack)	27615	3.84
5	AAA Batteries (4-pack)	30986	2.99
6	Apple Airpods Headphones	15637	150.00
7	Bose SoundSport Headphones	13430	99.99
8	Flatscreen TV	4813	300.00
9	Google Phone	5529	600.00
10	LG Dryer	646	600.00
11	LG Washing Machine	666	600.00
12	Lightning Charging Cable	23169	14.95
13	Macbook Pro Laptop	4725	1700.00
14	ThinkPad Laptop	4128	999.99
15	USB-C Charging Cable	23931	11.95
16	Vareebadd Phone	2068	400.00
17	Wired Headphones	20524	11.99
18	iPhone	6847	700.00

In [ ]:

''' When we say twin axes, it means a figure can have dual x or y-axes.. plt.twinx() : function which is used to create a twin Axes that are sharing the x-axis.. Similarly, the function twiny() is used to create a second x axis in your figure, which means twiny() sharing the y-axis.. '''

In [48]:
```
products = count_df['Product'].values
```

In [ ]:
```
count_df
```

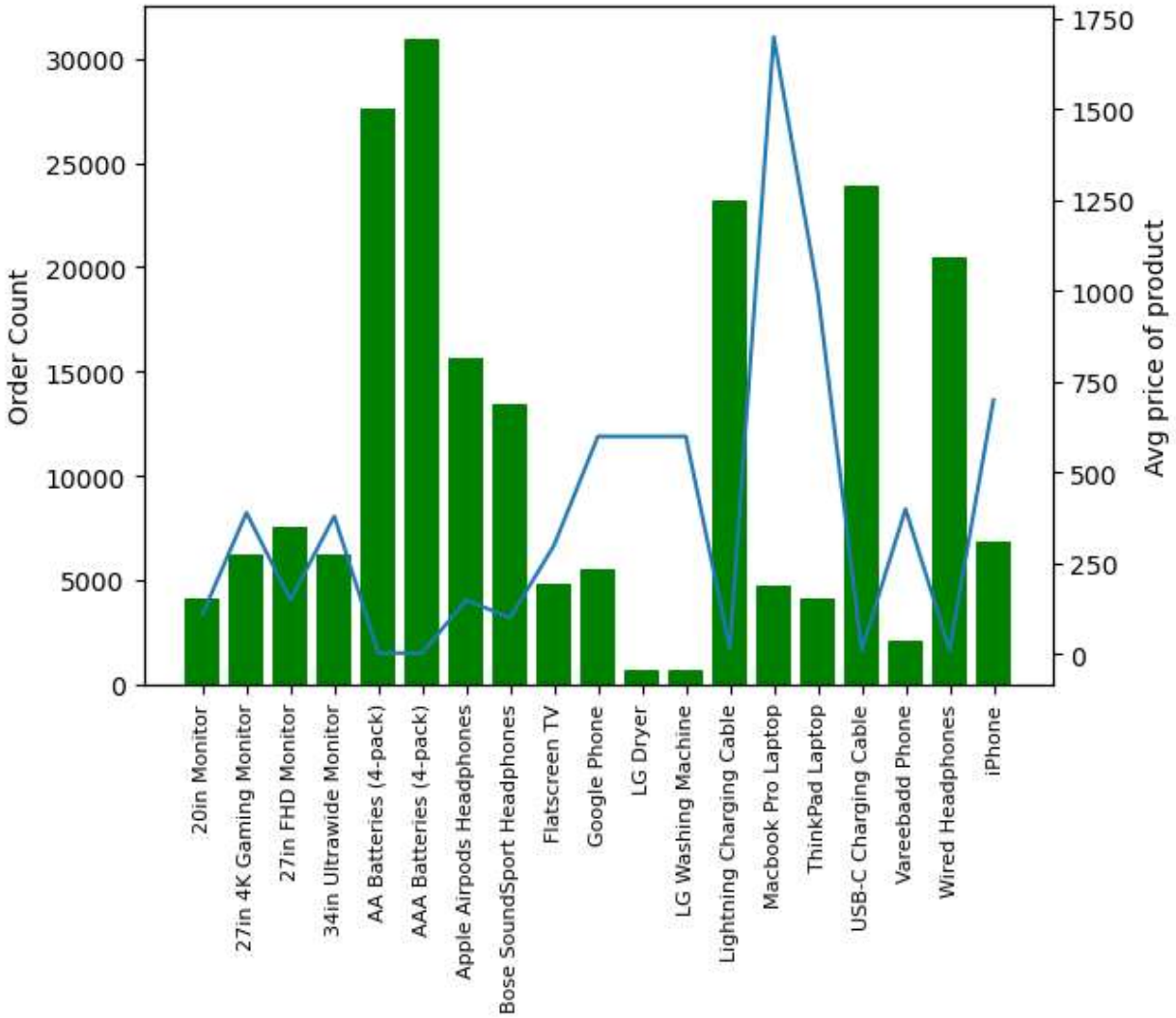
In [49]:
```
fig , ax1 = plt.subplots()

ax2 = ax1.twinx() ## as X-axis is same for both plots , ie we are sharing x-axis , ie

ax1.bar(count_df['Product'] , count_df['Quantity Ordered'] , color='g')
ax2.plot(count_df['Product'] , count_df['Price Each'] )
ax1.set_xticklabels(products , rotation='vertical' , fontsize=8 )

ax1.set_ylabel('Order Count')
ax2.set_ylabel('Avg price of product')
```

Out[49]:
```
Text(0, 0.5, 'Avg price of product')
```



In [ ]:

''' Insights : The top selling product is 'AAA Batteries'. The top selling products seems to have a correlation with the price of the product. The cheaper the product higher the quantity ordered and vice versa. '''

In [ ]:

5.. Understanding Trend of the most sold product ?

In [ ]:

In [50]:
```
all_data['Product'].value_counts()
```

Out[50]:
```
Product
USB-C Charging Cable          21859
Lightning Charging Cable      21610
AAA Batteries (4-pack)        20612
AA Batteries (4-pack)         20558
Wired Headphones              18849
Apple Airpods Headphones      15525
Bose SoundSport Headphones    13298
27in FHD Monitor               7498
iPhone                         6840
27in 4K Gaming Monitor         6225
34in Ultrawide Monitor         6174
Google Phone                   5522
Flatscreen TV                  4794
Macbook Pro Laptop             4721
ThinkPad Laptop                4126
20in Monitor                   4098
Vareebadd Phone                2065
LG Washing Machine              666
LG Dryer                        646
Name: count, dtype: int64
```

In [51]:
```
all_data['Product'].value_counts()[0:5].index
```

Out[51]:
```
Index(['USB-C Charging Cable', 'Lightning Charging Cable',
       'AAA Batteries (4-pack)', 'AA Batteries (4-pack)', 'Wired Headphones'],
      dtype='object', name='Product')
```

In [59]:
```
most_sold_product = all_data['Product'].value_counts()[0:5].index
```

In [ ]:
```

```

In [60]:
```
all_data['Product'].isin(most_sold_product)
```

Out[60]:
```
0          True
2         False
3         False
4          True
5          True
          ...
186845     True
186846    False
186847    False
186848    False
186849     True
Name: Product, Length: 185686, dtype: bool
```

In [61]:
```
most_sold_product_df = all_data[all_data['Product'].isin(most_sold_product)]  ## data
```

In [62]:
```
most_sold_product_df.head(4)
```

Out[62]:

	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
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles
6	176562	USB-C Charging Cable	1	11.95	04/29/19 13:03	381 Wilson St, San Francisco, CA 94016	4	11.95	San Francisco

In [ ]:

In [ ]:
```
## Since we have learnt how to create frequency table or pivot table using crosstab()
## lets learn how to do it using groupby + unstack()
```

In [63]:
```
most_sold_product_df.groupby(['Month' , 'Product']).size()
```

Out[63]:

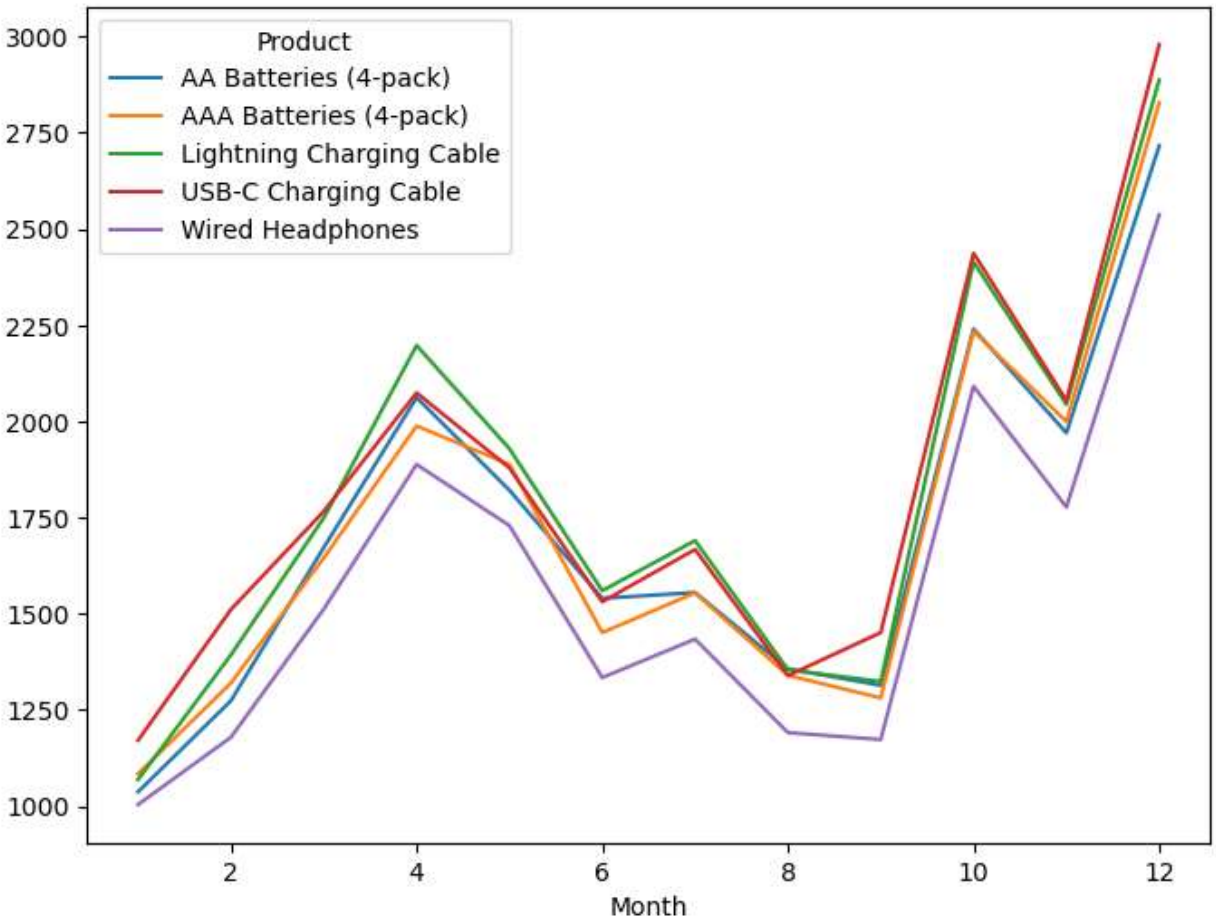
Month	Product	
1	AA Batteries (4-pack)	1037
	AAA Batteries (4-pack)	1084
	Lightning Charging Cable	1069
	USB-C Charging Cable	1171
	Wired Headphones	1004
2	AA Batteries (4-pack)	1274
	AAA Batteries (4-pack)	1320
	Lightning Charging Cable	1393
	USB-C Charging Cable	1511
	Wired Headphones	1179
3	AA Batteries (4-pack)	1672
	AAA Batteries (4-pack)	1645
	Lightning Charging Cable	1749
	USB-C Charging Cable	1766
	Wired Headphones	1512
4	AA Batteries (4-pack)	2062
	AAA Batteries (4-pack)	1988
	Lightning Charging Cable	2197
	USB-C Charging Cable	2074
	Wired Headphones	1888
5	AA Batteries (4-pack)	1821
	AAA Batteries (4-pack)	1888
	Lightning Charging Cable	1929
	USB-C Charging Cable	1879
	Wired Headphones	1729
6	AA Batteries (4-pack)	1540
	AAA Batteries (4-pack)	1451
	Lightning Charging Cable	1560
	USB-C Charging Cable	1531
	Wired Headphones	1334
7	AA Batteries (4-pack)	1555
	AAA Batteries (4-pack)	1554
	Lightning Charging Cable	1690
	USB-C Charging Cable	1667
	Wired Headphones	1434
8	AA Batteries (4-pack)	1357
	AAA Batteries (4-pack)	1340
	Lightning Charging Cable	1354
	USB-C Charging Cable	1339
	Wired Headphones	1191
9	AA Batteries (4-pack)	1314
	AAA Batteries (4-pack)	1281
	Lightning Charging Cable	1324
	USB-C Charging Cable	1451
	Wired Headphones	1173
10	AA Batteries (4-pack)	2240
	AAA Batteries (4-pack)	2234
	Lightning Charging Cable	2414
	USB-C Charging Cable	2437
	Wired Headphones	2091
11	AA Batteries (4-pack)	1970
	AAA Batteries (4-pack)	1999
	Lightning Charging Cable	2044
	USB-C Charging Cable	2054
	Wired Headphones	1777
12	AA Batteries (4-pack)	2716
	AAA Batteries (4-pack)	2828
	Lightning Charging Cable	2887
	USB-C Charging Cable	2979

```
Wired Headphones            2537
dtype: int64
```

In [64]:
```
pivot = most_sold_product_df.groupby(['Month' , 'Product']).size().unstack()
```

In [65]:
```
pivot.plot(figsize=(8,6))
```

Out[65]:
```
<Axes: xlabel='Month'>
```



''' Insight : Products have been sold more in Oct , Nov , Dec '''

In [ ]:
```
```

6.. What products are most often sold together ?

In [66]:
```
all_data.columns
```

Out[66]:
```
Index(['Order ID', 'Product', 'Quantity Ordered', 'Price Each', 'Order Date',
       'Purchase Address', 'Month', 'sales', 'city'],
      dtype='object')
```

In [67]:
```
all_data['Order ID']
```

Out[67]:
```
0         176558
2         176559
3         176560
4         176560
5         176561
           ...
186845    259353
186846    259354
186847    259355
186848    259356
186849    259357
Name: Order ID, Length: 185686, dtype: object
```

In [68]:
```
df_duplicated = all_data[all_data['Order ID'].duplicated(keep=False)]
```

In [69]:
```
df_duplicated
```

Out[69]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	sales	city
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles
18	176574	Google Phone	1	600.00	04/03/19 19:42	20 Hill St, Los Angeles, CA 90001	4	600.00	Los Angeles
19	176574	USB-C Charging Cable	1	11.95	04/03/19 19:42	20 Hill St, Los Angeles, CA 90001	4	11.95	Los Angeles
32	176586	AAA Batteries (4-pack)	2	2.99	04/10/19 17:00	365 Center St, San Francisco, CA 94016	4	5.98	San Francisco
...	...	...	...	...	...	...	...	...	...
186792	259303	AA Batteries (4-pack)	1	3.84	09/20/19 20:18	106 7th St, Atlanta, GA 30301	9	3.84	Atlanta
186803	259314	Wired Headphones	1	11.99	09/16/19 00:25	241 Highland St, Atlanta, GA 30301	9	11.99	Atlanta
186804	259314	AAA Batteries (4-pack)	2	2.99	09/16/19 00:25	241 Highland St, Atlanta, GA 30301	9	5.98	Atlanta
186841	259350	Google Phone	1	600.00	09/30/19 13:49	519 Maple St, San Francisco, CA 94016	9	600.00	San Francisco
186842	259350	USB-C Charging Cable	1	11.95	09/30/19 13:49	519 Maple St, San Francisco, CA 94016	9	11.95	San Francisco

14128 rows × 9 columns

In [70]:
```
dup_products = df_duplicated.groupby(['Order ID'])['Product'].apply(lambda x : ','.joi

## for every Order-Id , collect all the products ..
```

In [71]:
```
dup_products
```

Out[71]:

	Order ID	grouped_products
0	141275	USB-C Charging Cable,Wired Headphones
1	141290	Apple Airpods Headphones,AA Batteries (4-pack)
2	141365	Vareebadd Phone,Wired Headphones
3	141384	Google Phone,USB-C Charging Cable
4	141450	Google Phone,Bose SoundSport Headphones
...	...	...
6874	319536	Macbook Pro Laptop,Wired Headphones
6875	319556	Google Phone,Wired Headphones
6876	319584	iPhone,Wired Headphones
6877	319596	iPhone,Lightning Charging Cable
6878	319631	34in Ultrawide Monitor,Lightning Charging Cable

6879 rows × 2 columns

In [ ]:

In [72]:
```
dup_products_df = df_duplicated.merge(dup_products , how='left' , on='Order ID') ## me
```

In [73]:
```
dup_products_df
```

Out[73]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	sales	city	grouped
0	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles	Ph He
1	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles	Ph He
2	176574	Google Phone	1	600.00	04/03/19 19:42	20 Hill St, Los Angeles, CA 90001	4	600.00	Los Angeles	Ph Char
3	176574	USB-C Charging Cable	1	11.95	04/03/19 19:42	20 Hill St, Los Angeles, CA 90001	4	11.95	Los Angeles	Ph Char
4	176586	AAA Batteries (4-pack)	2	2.99	04/10/19 17:00	365 Center St, San Francisco, CA 94016	4	5.98	San Francisco	AAA B pa
...	...	...	...	...	...	...	...	...	...	
14123	259303	AA Batteries (4-pack)	1	3.84	09/20/19 20:18	106 7th St, Atlanta, GA 30301	9	3.84	Atlanta	34ir N Batteri
14124	259314	Wired Headphones	1	11.99	09/16/19 00:25	241 Highland St, Atlanta, GA 30301	9	11.99	Atlanta	Headp Batteri
14125	259314	AAA Batteries (4-pack)	2	2.99	09/16/19 00:25	241 Highland St, Atlanta, GA 30301	9	5.98	Atlanta	Headp Batteri
14126	259350	Google Phone	1	600.00	09/30/19 13:49	519 Maple St, San Francisco, CA 94016	9	600.00	San Francisco	Ph Char
14127	259350	USB-C Charging Cable	1	11.95	09/30/19 13:49	519 Maple St, San Francisco, CA 94016	9	11.95	San Francisco	Ph Char

14128 rows × 10 columns

In [ ]:

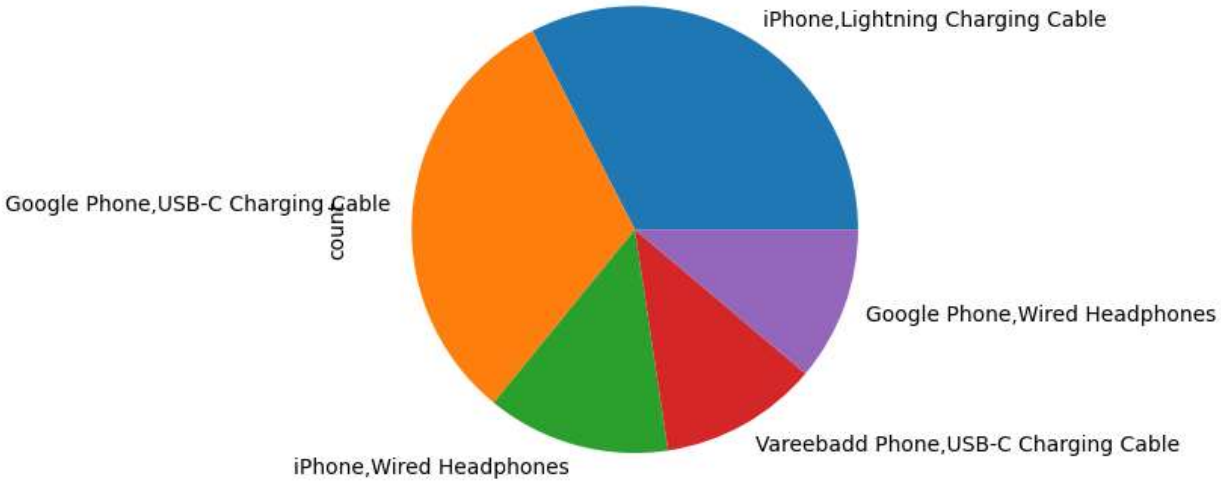
In [74]:
```
no_dup_df = dup_products_df.drop_duplicates(subset=['Order ID']) # lets drop out all d
```

In [75]:
```
no_dup_df.shape
```

Out[75]:
```
(6879, 10)
```

In [76]:
```
no_dup_df['grouped_products'].value_counts()[0:5].plot.pie()
```

Out[76]:
```
<Axes: ylabel='count'>
```



''' Insight: As soon as any Person will bought Iphone , we can recommend him charging cable , wired headphones
As soon as any Person will bought Google phone , we can recommend him USB-c charging cable '''

In [ ]: