In [1]:
```
import os
import pandas as pd #pandas import
import numpy as np #numpy import
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%pylab inline
df = pd.read_csv("data.csv",  index_col = None )
df.head()
```

```
Populating the interactive namespace from numpy and matplotlib
```

Out[1]:

	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 Data

In [2]:
```
#Finding out/listing the rows with null values
df_nan = df[df.isna().any(axis=1)]
df_nan.head()
```

Out[2]:

	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 [3]:
```
df=df.dropna(how='all')
df.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
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 [4]:
```
#nunique() gives the number of unique values on specified column
#unique() only list the unique values themselve
df['Product'].nunique()
df['Product'].unique()
```

Out[4]:
```
array(['USB-C Charging Cable', 'Bose SoundSport Headphones',
       'Google Phone', 'Wired Headphones', 'Macbook Pro Laptop',
       'Lightning Charging Cable', '27in 4K Gaming Monitor',
       'AA Batteries (4-pack)', 'Apple Airpods Headphones',
       'AAA Batteries (4-pack)', 'iPhone', 'Flatscreen TV',
       '27in FHD Monitor', '20in Monitor', 'LG Dryer', 'ThinkPad Laptop',
       'Vareebadd Phone', 'LG Washing Machine', '34in Ultrawide Monitor',
       'Product'], dtype=object)
```

In [5]:
```
#Count the number of value in each category
df['Product'].value_counts()
```

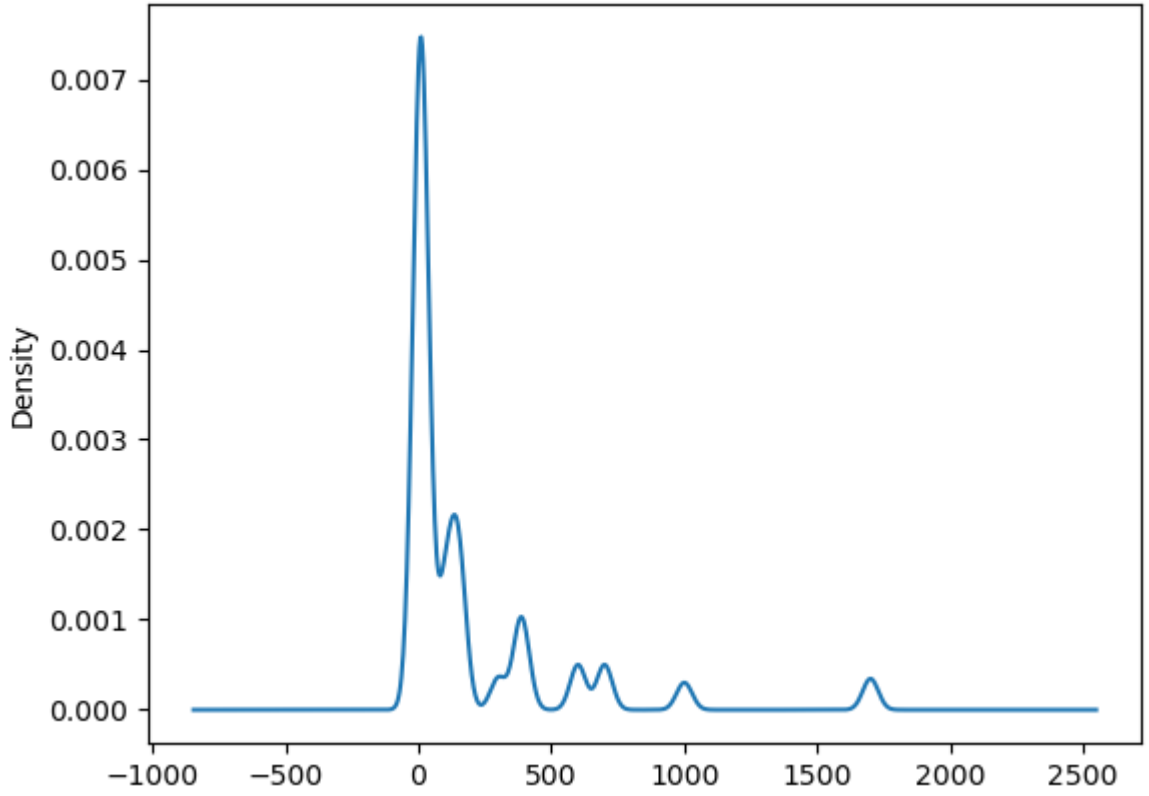
Out[5]:
```
USB-C Charging Cable          21903
Lightning Charging Cable      21658
AAA Batteries (4-pack)        20641
AA Batteries (4-pack)         20577
Wired Headphones              18882
Apple Airpods Headphones      15549
Bose SoundSport Headphones    13325
27in FHD Monitor               7507
iPhone                         6842
27in 4K Gaming Monitor         6230
34in Ultrawide Monitor         6181
Google Phone                   5525
Flatscreen TV                  4800
Macbook Pro Laptop             4724
ThinkPad Laptop                4128
20in Monitor                   4101
Vareebadd Phone                2065
LG Washing Machine              666
LG Dryer                        646
Product                         355
Name: Product, dtype: int64
```

In [6]:
```
df= df[df['Order Date'].str[0:2]!='Or']
#Changing the  data type of the two column
df['Quantity Ordered']=pd.to_numeric(df['Quantity Ordered'])
df['Price Each']=pd.to_numeric(df['Price Each'])
```

Checking Data Normality

In [7]:
```
#df['Price Each']=pd.to_numeric(df['Price Each'])
df['Price Each'].plot(kind='density')
```

Out[7]:
```
<AxesSubplot:ylabel='Density'>
```



Seprating Columns for Dates

In [13]:
```
#Creating seprate column for month

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

df.tail()
```

Out[13]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
186845	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	9
186846	259354	iPhone	1	700.00	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	9
186847	259355	iPhone	1	700.00	09/23/19 07:39	220 12th St, San Francisco, CA 94016	9
186848	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	9
186849	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	9

In [14]:
```
#Adding sale column
df['Sale']=df['Quantity Ordered']*df['Price Each']
df.head(2)
```

Out[14]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale
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 [15]:
```
#What was the best month for sale? How much was earned that year?
results=df.groupby('Month').sum()
results.head(2)
```

Out[15]:

Month	Quantity Ordered	Price Each	Sale
1	10903	1811768.38	1822256.73
2	13449	2188884.72	2202022.42

In [16]:
```
#What time shuld we advertise to maximize sale?
#Changing order date column to date and time
df['Date'] = df['Order Date'].apply(lambda x:x.split(' ')[0])

df.head(1)
```

Out[16]:

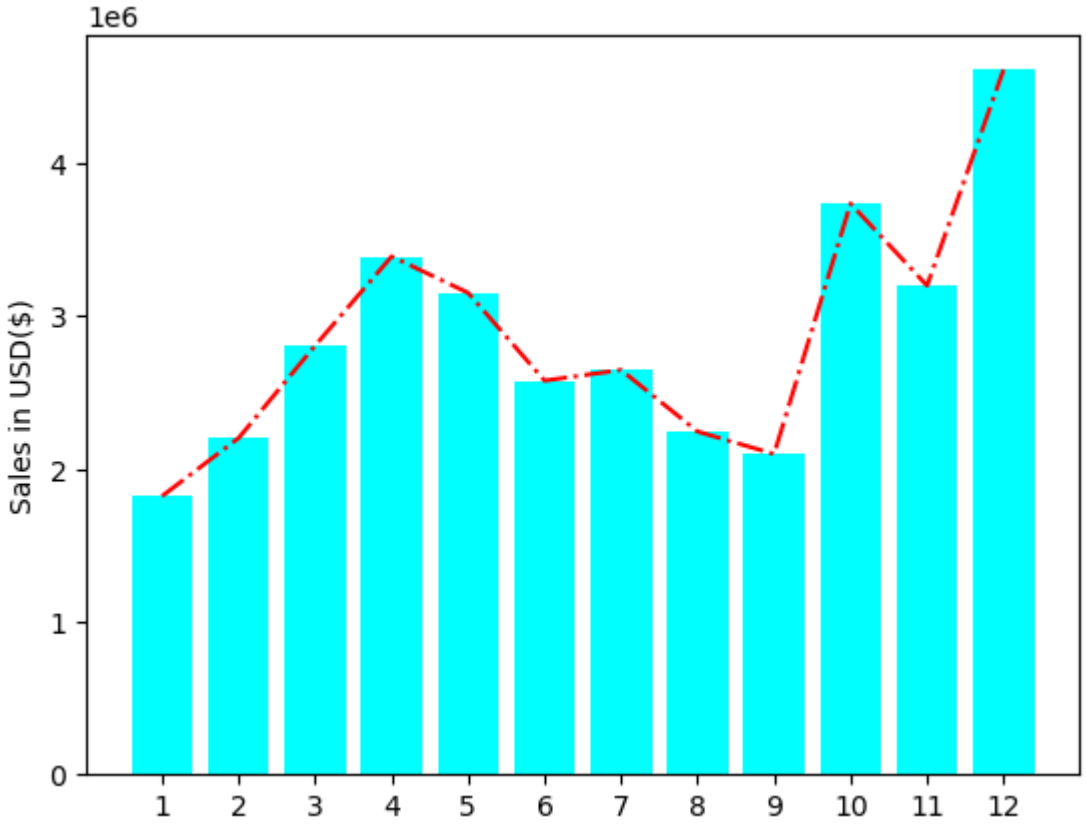
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.9	04/19/19

Which Month has the highest sale?

In [63]:
```
months = range(1,13)
plt.bar(months,results['Sale'],  color='cyan')
plt.plot(results['Sale'], 'r-.', color='red')
plt.xticks(months)
#plt.yticks(results['Sale'])
plt.ylabel("Sales in USD($)")
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_20056\2168574099.py:3: UserWarning:

color is redundantly defined by the 'color' keyword argument and the fmt string "r-." (-> color='r'). The keyword argum
ent will take precedence.
```



In [18]:
```
df['Order Date']=pd.to_datetime(df['Order Date'])
```

In [19]:
```
# what city has the hieghst number of sale?
#First create a city column
#.apply --> Let us to run function on our dataframe
def get_city(address):
    return address.split(',')[1]

def get_state(address):
    return address.split(',')[2].split(' ')[1]
df['City']= df['Purchase Address'].apply(lambda x: get_city(x)+' '+get_state(x))
df.head(3)
```

Out[19]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	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	04/19/19	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	04/07/19	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	04/12/19	Los Angeles CA

In [20]:
```
city=df.groupby("City").sum()
city
```

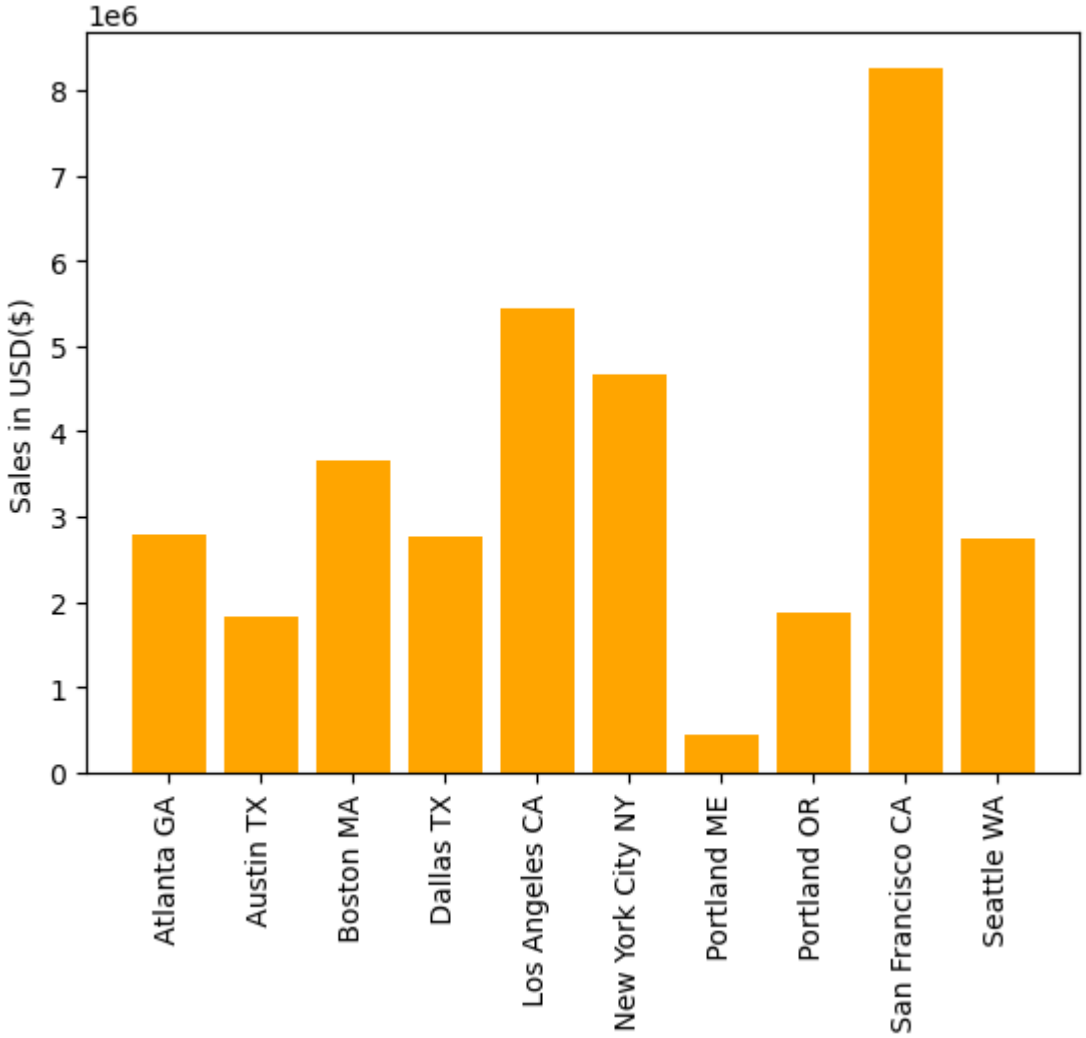
Out[20]:

City	Quantity Ordered	Price Each	Month	Sale
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

Which City has the highest Sale?

In [77]:

```
cityindex=[city for city, df in df.groupby("City")]
plt.bar(cityindex,city['Sale'], color='orange')
plt.xticks(cityindex, rotation=90)
#plt.yticks(results['Sale'])
plt.ylabel("Sales in USD($)")
plt.show()
```



In [22]:
```
df['Hour']= df['Order Date'].dt.hour
df['Minute']= df['Order Date'].dt.minute
df.head(1)
```

Out[22]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	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	23.9	04/19/19	Dallas TX	8	46

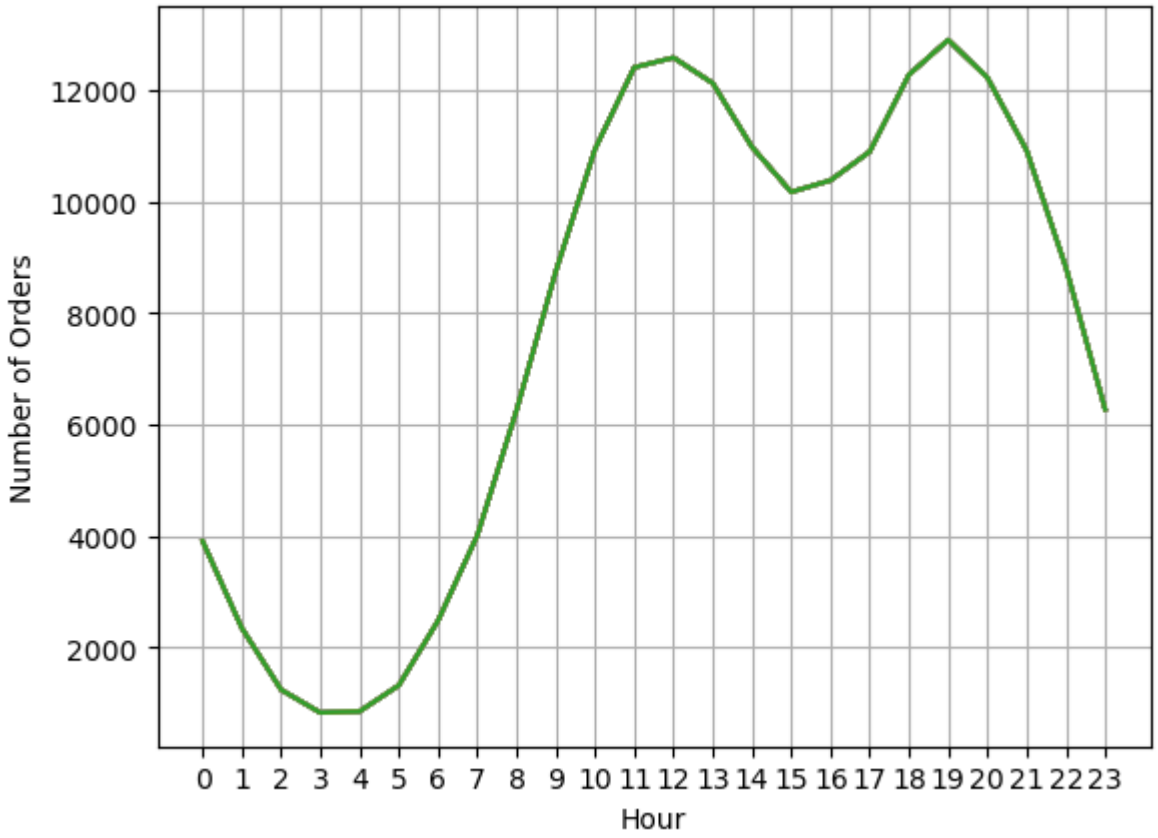
In [23]:
```
df['Date']= pd.to_datetime(df['Date'])
df.head()
```

Out[23]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	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	2019-04-19	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	2019-04-07	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	2019-04-12	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	2019-04-12	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	2019-04-30	Los Angeles CA	9	27

Which hour of the day has the highest number of sale?

In [69]:
```
hours=[hour for hour, df in df.groupby('Hour')]
plt.plot(hours, df.groupby(['Hour']).count())
plt.xticks(hours)
plt.xlabel("Hour")
plt.ylabel("Number of Orders")
plt.grid()
plt.show()
```



In [25]:
```
#What products were sold together?
df.head()
dfDuplicate =df[df['Order ID'].duplicated(keep=False)]

dfDuplicate['Grouped'] = dfDuplicate.groupby('Order ID')['Product'].transform(lambda x:','.join(x))
dfDuplicate.head(2)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_20056\2876416605.py:5: 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
  dfDuplicate['Grouped'] = dfDuplicate.groupby('Order ID')['Product'].transform(lambda x:','.join(x))
```

Out[25]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	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	2019-04-12	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	2019-04-12	Los Angeles CA	14	38	Google Phone,Wired Headphones

In [26]:
```
#Dropping duplicated columns
dfDuplicated = dfDuplicate[['Order ID','Grouped']].drop_duplicates()
dfDuplicated.head(2)
```

Out[26]:

	Order ID	Grouped
3	176560	Google Phone,Wired Headphones
18	176574	Google Phone,USB-C Charging Cable

In [27]:
```
from itertools import combinations
from collections import Counter
#Counting the number of times product have been ordered at the same time or together
count = Counter()
for row in dfDuplicated['Grouped']:
        row_list= row.split(',')
        count.update(Counter(combinations(row_list,2)))

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


#This comment only prints out the top 10
#count.most_common(10)
```

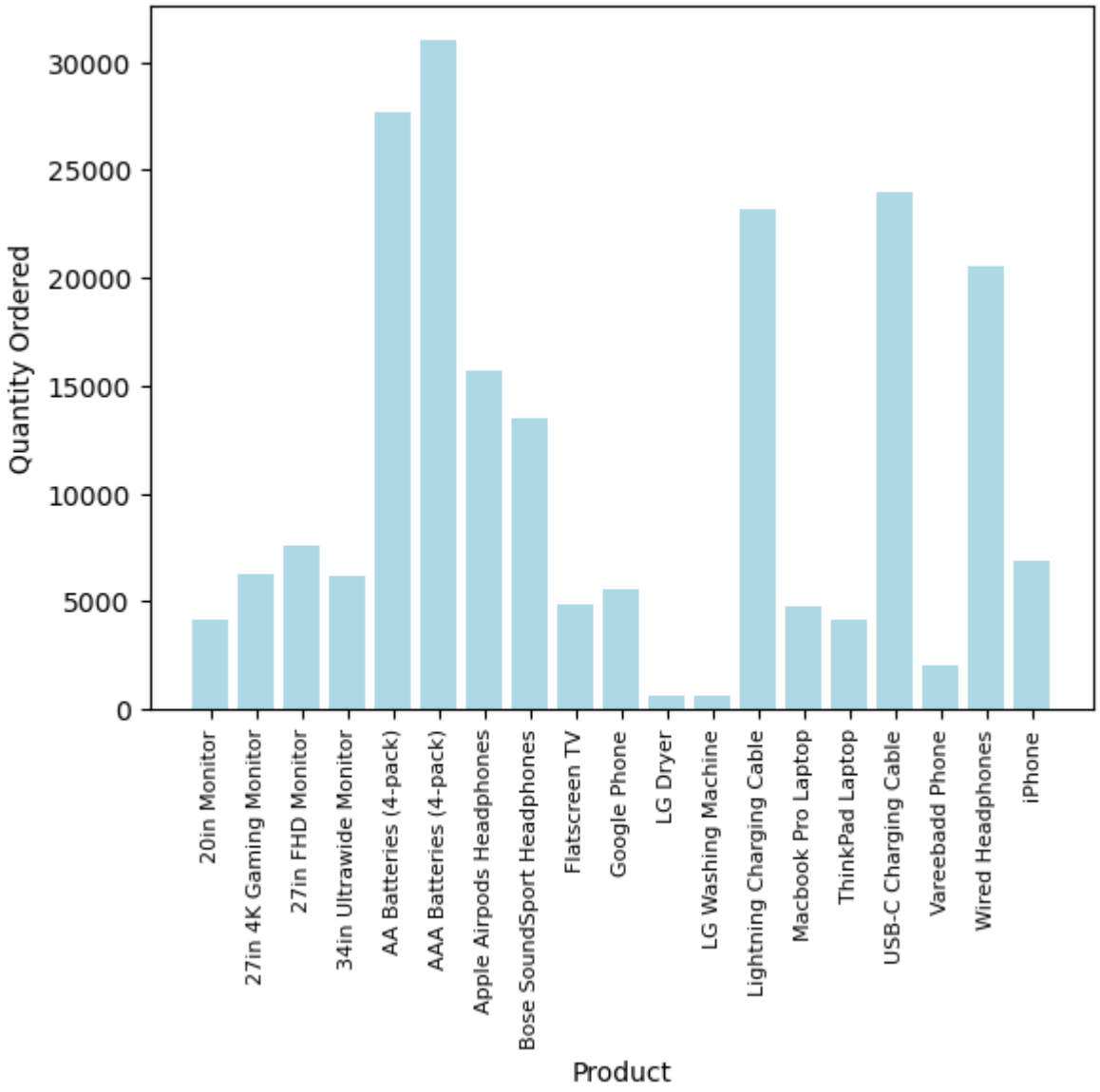
```
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
```

Which products were mostly sold together?

In [72]:
```
#What product sold the most?
productGrp=df.groupby('Product')
quantityOrd= productGrp.sum()['Quantity Ordered']

products=[product for product, df in productGrp]
plt.bar(products,quantityOrd ,color='lightblue')

plt.xticks(rotation=90, size=8)
plt.xlabel('Product')
plt.ylabel('Quantity Ordered')
plt.show()
```



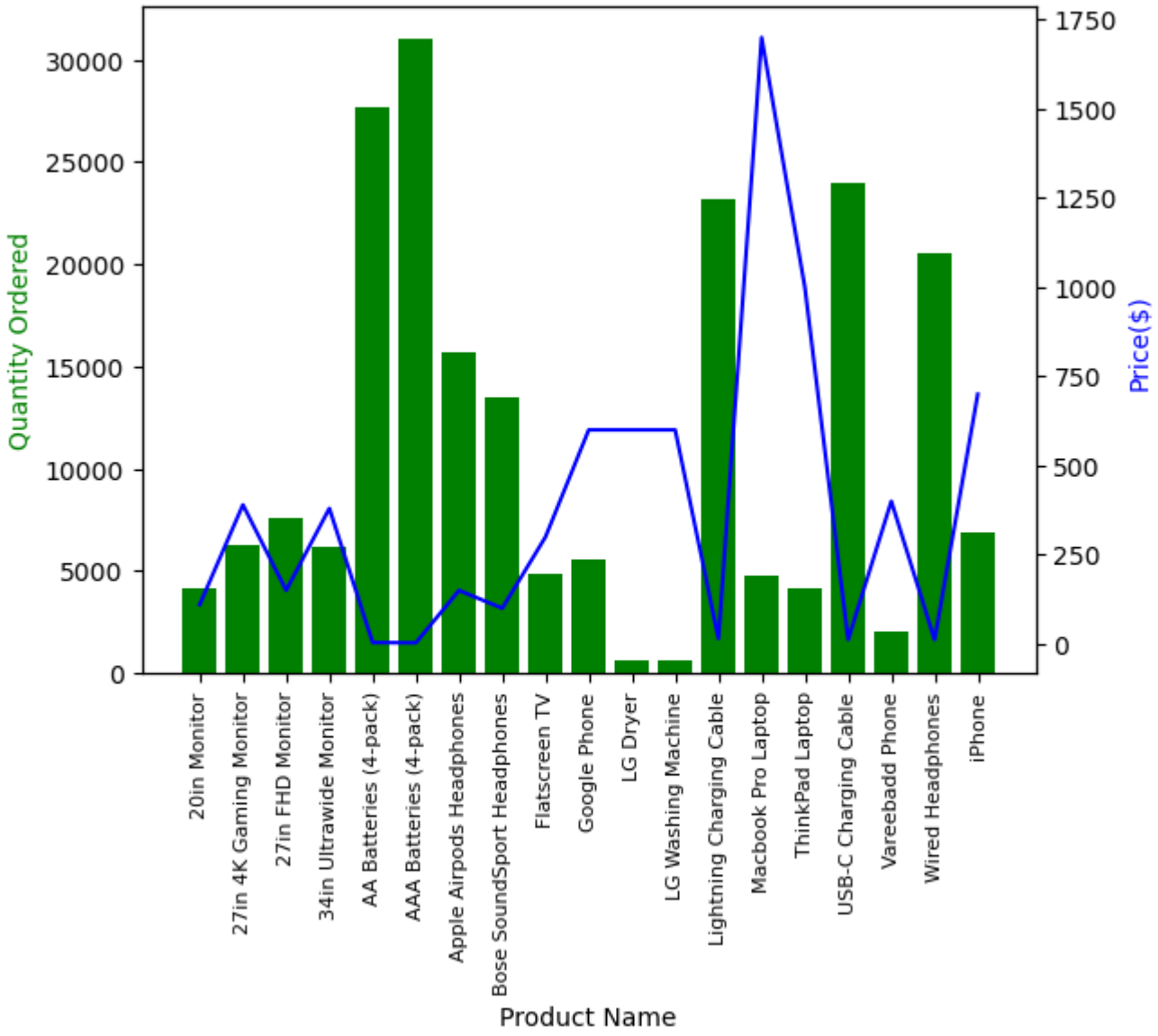
Which Produsts were sold together (number of product sold vs. sale amount)

In [29]:
```
prices = df.groupby('Product').mean()['Price Each']
```

```
fig, ax1= plt.subplots()
ax2=ax1.twinx()
ax1.bar(products,quantityOrd,color='g' )
ax2.plot(products, prices,'b-')

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

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_20056\2917419474.py:9: UserWarning: FixedFormatter should only be used toget
her with FixedLocator
  ax1.set_xticklabels(products,rotation='vertical', size=8)
```



Some Cleaning and adding changes to columns

Adding an specific amount to the price of all the products

In [30]:
```
def add(x):
    return(x//2)
df['Added lambda']=df['Price Each'].apply(add)
df.head(1)
```

Out[30]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	City	Hour	Minute	Added lambda
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.9	2019-04-19	Dallas TX	8	46	5.0

In [31]:
```
#We can do the same using the lambda function
df['Added lambda']=df['Price Each'].apply(lambda x: x//2)
df.head(1)
```

Out[31]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	City	Hour	Minute	Added lambda
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.9	2019-04-19	Dallas TX	8	46	5.0

In [32]:
```
for index, row in df.iterrows():
    if row['Quantity Ordered'] == 1 or row['Quantity Ordered'] == 4:
        df.loc[index, 'Gender'] = 'M'
    else:
        df.loc[index, 'Gender'] = 'F'
```

In [33]:
```
df.drop(columns='Added lambda', axis=1, inplace=True)
```

In [34]:
```
df.head(2)
```

Out[34]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	City	Hour	Minute	Gender
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	2019-04-19	Dallas TX	8	46	F
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	2019-04-07	Boston MA	22	30	M

In [35]:
```
#Listing the products and price each of female customer only
df[df['Gender']=='F'][['Product','Price Each']]
```

Out[35]:

	Product	Price Each
0	USB-C Charging Cable	11.95
28	AAA Batteries (4-pack)	2.99
32	AAA Batteries (4-pack)	2.99
40	Lightning Charging Cable	14.95
42	Wired Headphones	11.99
...	...	...
186829	AAA Batteries (4-pack)	2.99
186830	USB-C Charging Cable	11.95
186831	AA Batteries (4-pack)	3.84
186835	AAA Batteries (4-pack)	2.99
186845	AAA Batteries (4-pack)	2.99

16592 rows × 2 columns

In [36]:
```
#The isin function will do the same
df[df['Gender'].isin(['F'])][['Product','Price Each']]
```

Out[36]:

	Product	Price Each
0	USB-C Charging Cable	11.95
28	AAA Batteries (4-pack)	2.99
32	AAA Batteries (4-pack)	2.99
40	Lightning Charging Cable	14.95
42	Wired Headphones	11.99
...	...	...
186829	AAA Batteries (4-pack)	2.99
186830	USB-C Charging Cable	11.95
186831	AA Batteries (4-pack)	3.84
186835	AAA Batteries (4-pack)	2.99
186845	AAA Batteries (4-pack)	2.99

16592 rows × 2 columns

Heatmap

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

Out[37]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	City	Hour	Minute	Gender
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	2019-04-19	Dallas TX	8	46	F
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	2019-04-07	Boston MA	22	30	M

In [38]:
```
#df_m = df.groupby(["Quantity Ordered", "Price Each"]).size().unstack(level=0)
df_m = df[['Month', 'Price Each','Sale']]
df_m= df_m[df_m['Price Each']>20]
```

In [39]:
```
#corr(df_m)
#np.corrcoef(df_m)



correlation=df.corr()
df.corr()
```

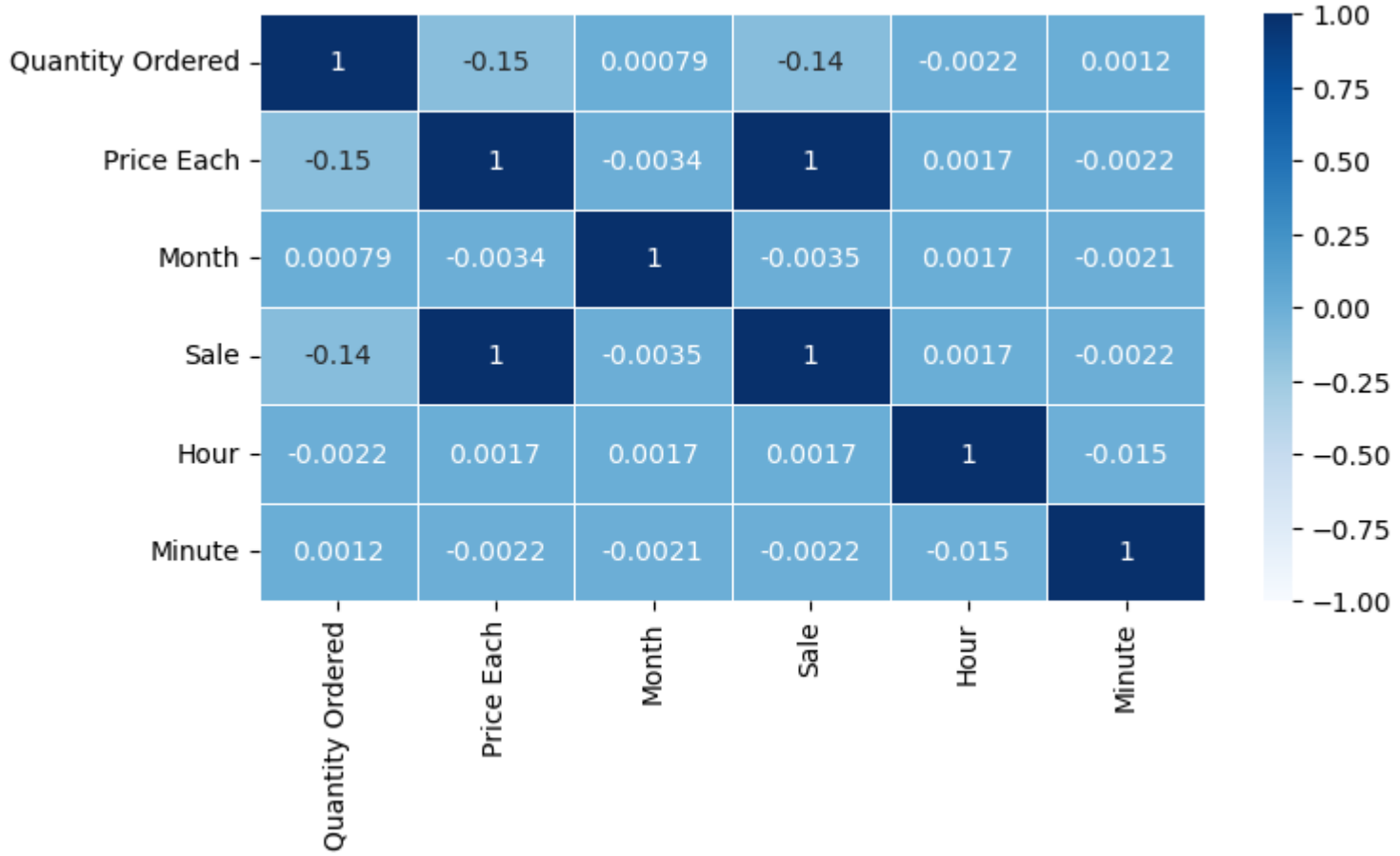
Out[39]:

	Quantity Ordered	Price Each	Month	Sale	Hour	Minute
Quantity Ordered	1.000000	-0.148272	0.000791	-0.139417	-0.002218	0.001225
Price Each	-0.148272	1.000000	-0.003375	0.999203	0.001721	-0.002163
Month	0.000791	-0.003375	1.000000	-0.003466	0.001731	-0.002075
Sale	-0.139417	0.999203	-0.003466	1.000000	0.001668	-0.002162
Hour	-0.002218	0.001721	0.001731	0.001668	1.000000	-0.015345
Minute	0.001225	-0.002163	-0.002075	-0.002162	-0.015345	1.000000

In [40]:
```
plt.figure(figsize=(8,4), dpi=100)
sns.heatmap(correlation, cmap='Blues', annot=True, linewidth=0.5, vmin=-1, vmax=1)

#A high correlation between sale and price of each product
#Moderate positive and negative correlation between other variables
```

Out[40]:
```
<AxesSubplot:>
```



In [41]:
```
df.groupby('Month')['Sale'].value_counts()
```

Out[41]:
```
Month  Sale
1      11.95      1074
       14.95       995
       11.99       934
       150.00      808
       3.84        758
                  ...
12     26.88         1
       59.80         1
       450.00        1
       779.98        1
       1200.00       1
Name: Sale, Length: 508, dtype: int64
```

In [42]:
```
months=(df.groupby('Month')
 ['Product']
 .value_counts()
 .unstack()
.fillna(0))
months

#months.transpose()
months.T
#Change/convert the matrix
```

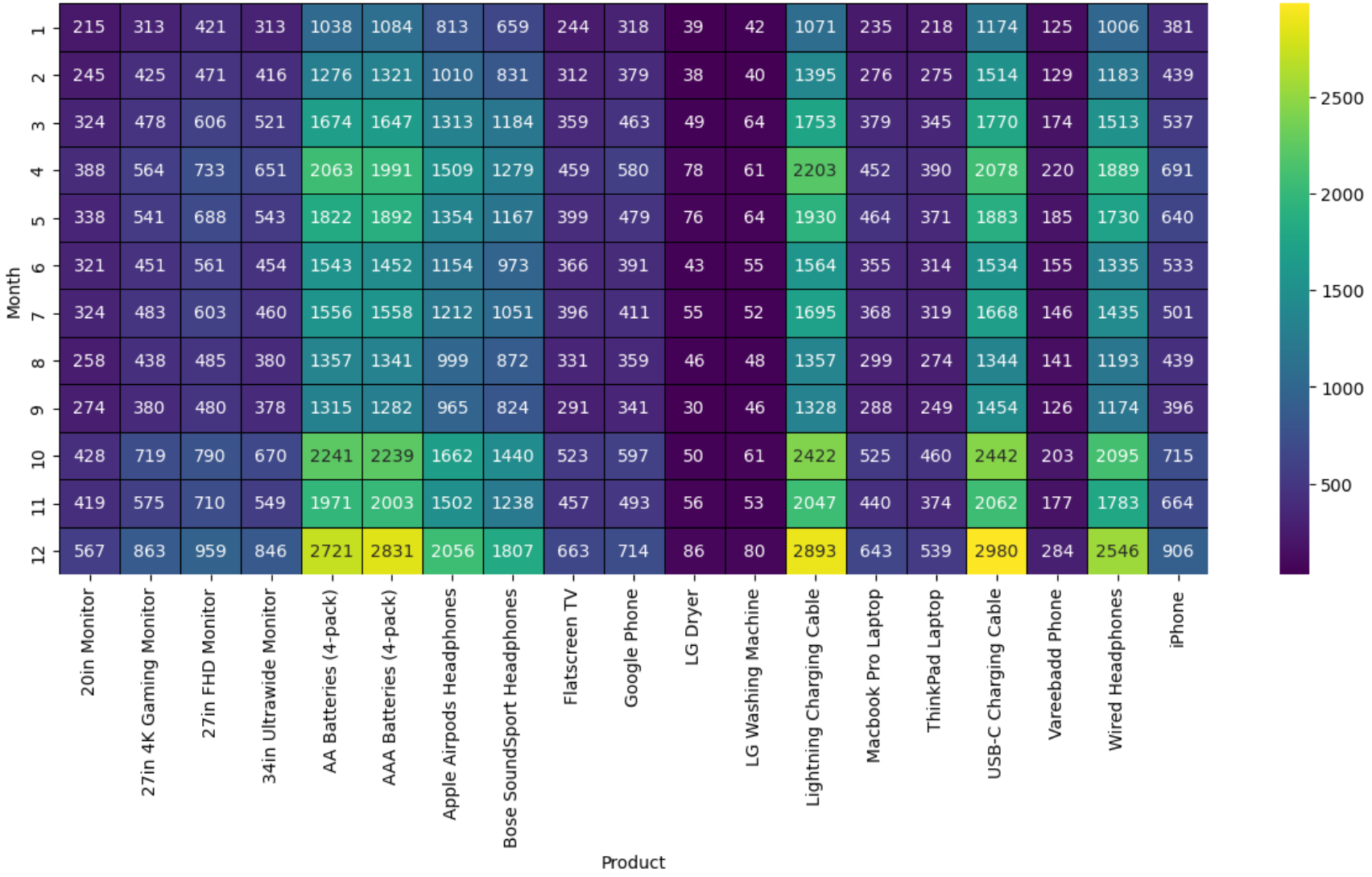
Out[42]:

Month	1	2	3	4	5	6	7	8	9	10	11	12
Product												
20in Monitor	215	245	324	388	338	321	324	258	274	428	419	567
27in 4K Gaming Monitor	313	425	478	564	541	451	483	438	380	719	575	863
27in FHD Monitor	421	471	606	733	688	561	603	485	480	790	710	959
34in Ultrawide Monitor	313	416	521	651	543	454	460	380	378	670	549	846
AA Batteries (4-pack)	1038	1276	1674	2063	1822	1543	1556	1357	1315	2241	1971	2721
AAA Batteries (4-pack)	1084	1321	1647	1991	1892	1452	1558	1341	1282	2239	2003	2831
Apple Airpods Headphones	813	1010	1313	1509	1354	1154	1212	999	965	1662	1502	2056
Bose SoundSport Headphones	659	831	1184	1279	1167	973	1051	872	824	1440	1238	1807
Flatscreen TV	244	312	359	459	399	366	396	331	291	523	457	663
Google Phone	318	379	463	580	479	391	411	359	341	597	493	714
LG Dryer	39	38	49	78	76	43	55	46	30	50	56	86
LG Washing Machine	42	40	64	61	64	55	52	48	46	61	53	80
Lightning Charging Cable	1071	1395	1753	2203	1930	1564	1695	1357	1328	2422	2047	2893
Macbook Pro Laptop	235	276	379	452	464	355	368	299	288	525	440	643
ThinkPad Laptop	218	275	345	390	371	314	319	274	249	460	374	539
USB-C Charging Cable	1174	1514	1770	2078	1883	1534	1668	1344	1454	2442	2062	2980
Vareebadd Phone	125	129	174	220	185	155	146	141	126	203	177	284
Wired Headphones	1006	1183	1513	1889	1730	1335	1435	1193	1174	2095	1783	2546
iPhone	381	439	537	691	640	533	501	439	396	715	664	906

Finding out max sold based on month of the years

In [43]:
```
#With no decimal place
plt.figure(figsize=(15,6), dpi=100)
sns.heatmap(months, cmap="viridis",  annot=True, linewidth=0.5, fmt=".0f", linecolor="black", square=False)
```

Out[43]: <AxesSubplot:xlabel='Product', ylabel='Month'>



Treemap

In [44]:
```
df.head(2)
```

Out[44]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	City	Hour	Minute	Gender
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	2019-04-19	Dallas TX	8	46	F
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	2019-04-07	Boston MA	22	30	M

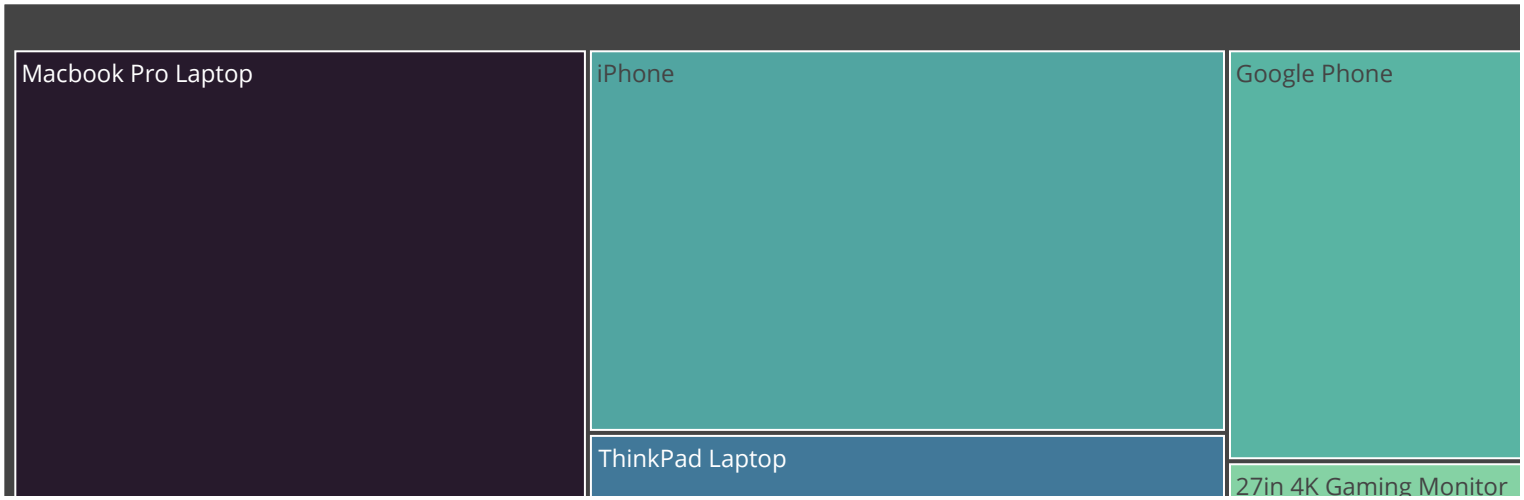
In [45]:
```
bins=[0,500,1000,float(inf)]
labels= ['Cheap', 'Middle-priced', 'Expensive']
df['Price Category']= pd.cut(df['Price Each'], bins=bins, labels=labels)
df.head(2)
```

Out[45]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sale	Date	City	Hour	Minute	Gender	Price Category
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	2019-04-19	Dallas TX	8	46	F	Cheap
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	2019-04-07	Boston MA	22	30	M	Cheap

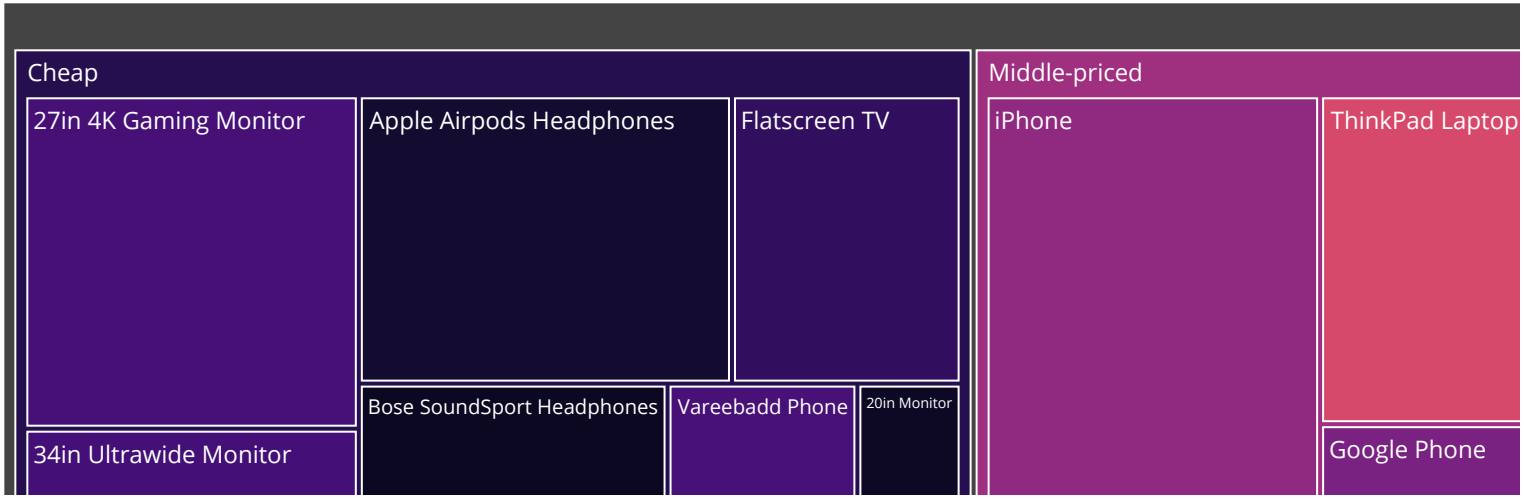
Sales amount of each product

In [46]:
```
px.treemap(data_frame=df, path=['Product'], values='Sale', color='Sale', color_continuous_scale='deep')
```



Categorizing based on Price (Cheap/Middle-Priced/Expensive)

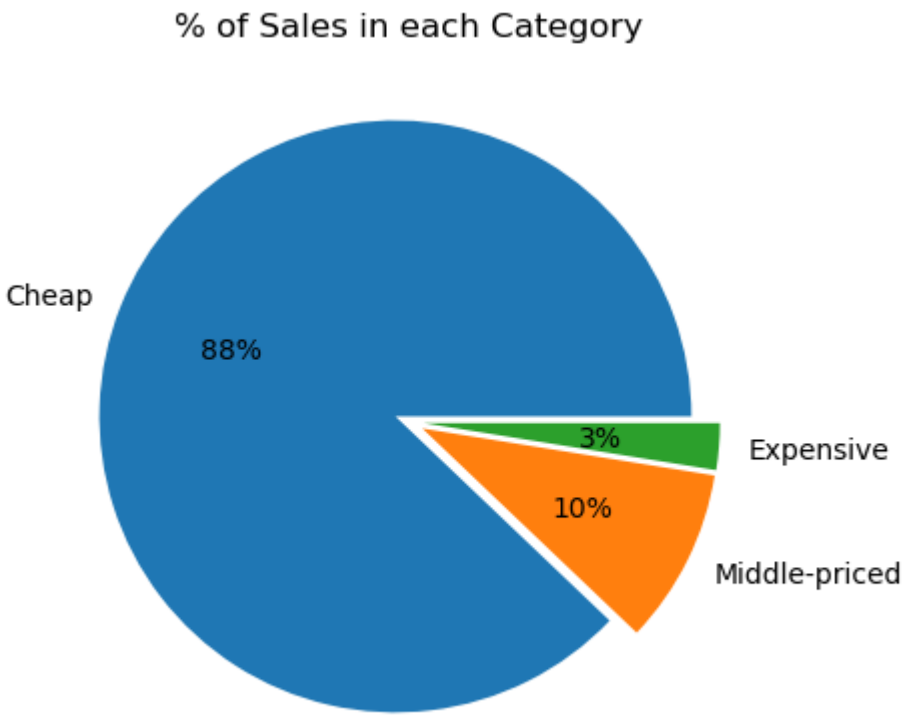
In [47]:
```
px.treemap(data_frame=df, path=['Price Category','Product'], values='Sale',color='Sale', color_continuous_scale='magma'
```



Percentage of Sales in each Category

In [61]:
```
explode = (0.05, 0.05, 0.05)
df.groupby("Price Category").size().plot(kind="pie",explode=explode, autopct='%1.0f%%',label="", title='% of Sales in e
```

Out[61]:
```
<AxesSubplot:title={'center':'% of Sales in each Category'}>
```


% of Sales in each Category
Cheap
88%
3%
Expensive
10%
Middle-priced