Sales Data Analyst

Purpose

Analyze sales data to identify trends,top-selling products, and revenue metrics for business decision-making.

Description

In this project, you will dive into a large sales dataset to extract valuable insights. You will explore sales trends over time, identify the bestselling products, calculate revenue metrics such as total sales and profit margins, and create visualizations to present your findings effectively. This project showcases your ability to manipulate and derive insights from large datasets, enabling you to make data-driven recommendations for optimizing sales strategies.

```
In [1]: import os
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly.graph_objs as go
        from plotly.offline import iplot
```

```
In [2]: all_data=pd.read_csv('Sales Data.csv')
        all_data.head()
```

Out[2]:

	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City
1	1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City
2	2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City
3	3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco
4	4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta

Data Cleaning and Formating

In [3]:

```
all_data.dtypes
```

Out[3]:

```
Unnamed: 0            int64
Order ID              int64
Product              object
Quantity Ordered      int64
Price Each          float64
Order Date           object
Purchase Address     object
Month                 int64
Sales               float64
City                 object
Hour                  int64
dtype: object
```

In [4]:

```
all_data.head()
```

Out[4]:

	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City
1	1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City
2	2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City
3	3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco
4	4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta

In [5]:

```
all_data.isnull().sum()
```

Out[5]:
```
Unnamed: 0          0
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
Month               0
Sales               0
City                0
Hour                0
dtype: int64
```

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

Out[6]:
```
(185950, 11)
```

What is the best mont for sale?

In [7]:
```
'04/19/19 08:46'.split('/')[0]
```

Out[7]:
```
'04'
```

In [8]:
```
def month(x):
  return x.split('/')[0]
```

add month col

In [9]:
```
all_data['Month'] = all_data['Order Date'].apply(month)
```

In [10]:
```
all_data['Month'].unique()
```

Out[10]:
```
array(['2019-12-30 00:01:00', '2019-12-29 07:03:00',
       '2019-12-12 18:21:00', ..., '2019-06-09 22:07:00',
       '2019-06-26 18:35:00', '2019-06-25 14:33:00'], dtype=object)
```

In [11]:
```
filter = all_data['Month'] == 'Order Date'
len(all_data[~filter])
```

Out[11]:
```
185950
```

In [12]:
```
all_data = all_data[~filter]
```

In [13]:
```
all_data.shape
```

Out[13]:
```
(185950, 11)
```

In [14]:
```
all_data.head()
```

Out[14]:

	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	2019-12-30 00:01:00	1700.00	New York Cit
1	1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	2019-12-29 07:03:00	600.00	New York Cit
2	2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	2019-12-12 18:21:00	11.95	New York Cit
3	3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	2019-12-22 15:13:00	149.99	San Francisco
4	4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	2019-12-18 12:38:00	11.95	Atlanta

In [15]:

```
all_data["Month"]
```

Out[15]:

```
0         2019-12-30 00:01:00
1         2019-12-29 07:03:00
2         2019-12-12 18:21:00
3         2019-12-22 15:13:00
4         2019-12-18 12:38:00
                 ...
185945    2019-06-07 19:02:00
185946    2019-06-01 19:29:00
185947    2019-06-22 18:57:00
185948    2019-06-26 18:35:00
185949    2019-06-25 14:33:00
Name: Month, Length: 185950, dtype: object
```

In [16]:

```
all_data.dtypes
```

Out[16]:

```
Unnamed: 0            int64
Order ID              int64
Product              object
Quantity Ordered      int64
Price Each          float64
Order Date           object
Purchase Address     object
Month                object
Sales               float64
City                 object
Hour                  int64
dtype: object
```

In [17]:

```
all_data['Price Each'] = all_data['Price Each'].astype(float)
```

In [18]:

```
all_data['Quantity Ordered'] = all_data['Quantity Ordered'].astype(int)
all_data.head(5)
```

Out[18]:

	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	2019-12-30 00:01:00	1700.00	New York Cit
1	1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	2019-12-29 07:03:00	600.00	New York Cit
2	2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	2019-12-12 18:21:00	11.95	New York Cit
3	3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	2019-12-22 15:13:00	149.99	San Francisco
4	4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	2019-12-18 12:38:00	11.95	Atlanta

In [19]:

```
all_data['sales']=all_data['Quantity  Ordered']*all_data['Price  Each']
all_data.head(5)
```

Out[19]:

	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	2019-12-30 00:01:00	1700.00	New York Cit
1	1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	2019-12-29 07:03:00	600.00	New York Cit
2	2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	2019-12-12 18:21:00	11.95	New York Cit
3	3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	2019-12-22 15:13:00	149.99	San Francisco
4	4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	2019-12-18 12:38:00	11.95	Atlanta

In [20]:

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

Out[20]:

```
Month
2019-01-01 03:07:00     11.99
2019-01-01 03:40:00     11.95
2019-01-01 04:56:00    150.00
2019-01-01 05:53:00      2.99
2019-01-01 06:03:00     23.90
                        ...  
2020-01-01 04:06:00    149.99
2020-01-01 04:13:00      2.99
2020-01-01 04:21:00     11.95
2020-01-01 04:54:00     99.99
2020-01-01 05:13:00    114.94
Name: sales, Length: 142395, dtype: float64
```

Which city has max order

In [21]:

```
'917 1st St, Dallas, TX 75001'.split(',')[1]
```

Out[21]:

```
' Dallas'
```

In [22]:

```
def city(x):
  return  x.split(',')[1]
```

In [23]:

```
all_data['city']  =  all_data['Purchase  Address'].apply(city)
```

In [24]:

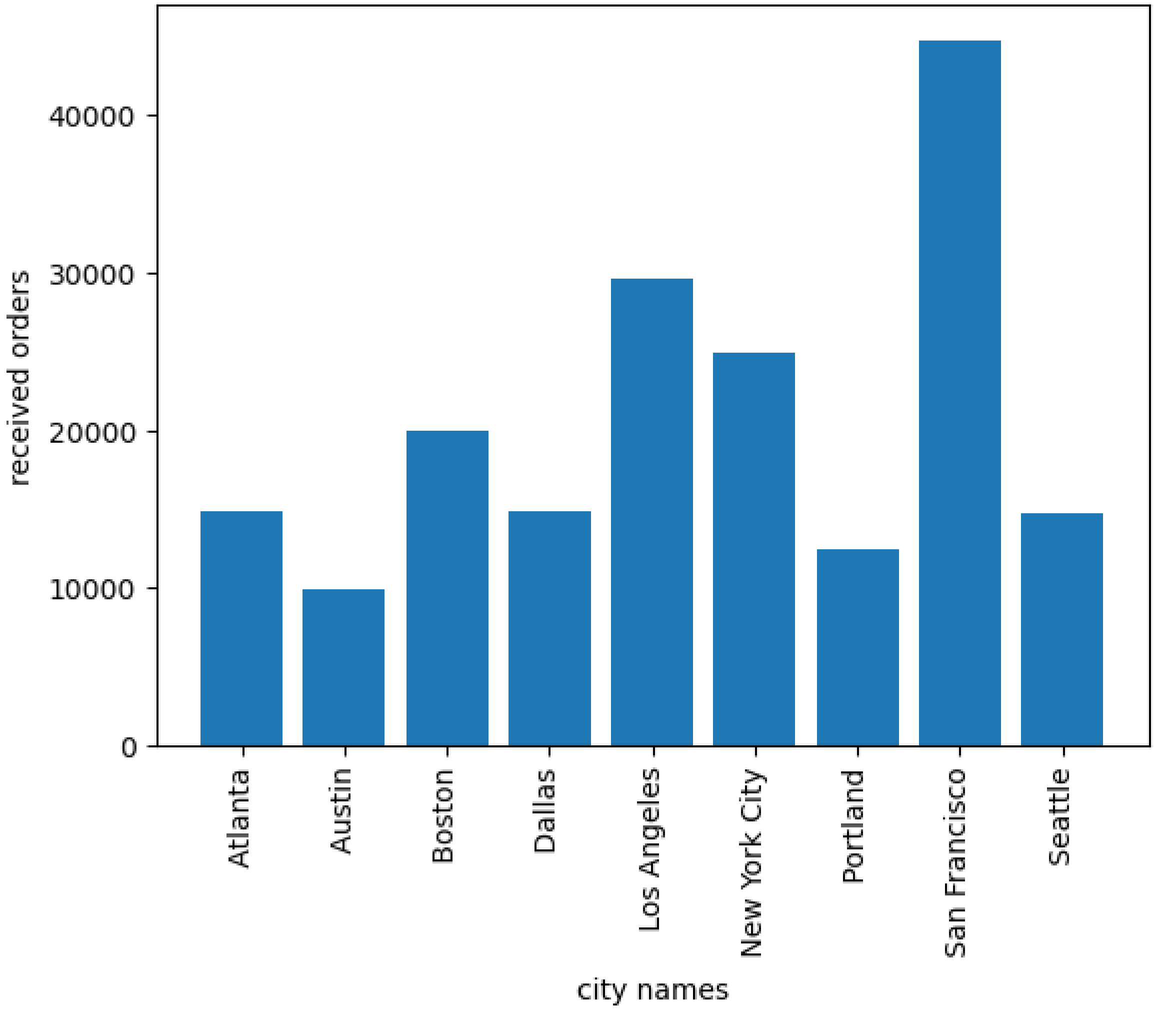
```
all_data.groupby('city')['city'].count()
```

Out[24]:

```
city
 Atlanta          14881
 Austin            9905
 Boston           19934
 Dallas           14820
 Los Angeles      29605
 New York City    24876
 Portland         12465
 San Francisco    44732
 Seattle          14732
Name: city, dtype: int64
```

In [25]:

```
plt.bar(all_data.groupby('city')['city'].count().index,all_data.groupby('city')['c
plt.xticks(rotation='vertical')
plt.ylabel('received  orders')
plt.xlabel('city  names')
plt.show()
```



what time should we display advertisements to maximise for product purchase?

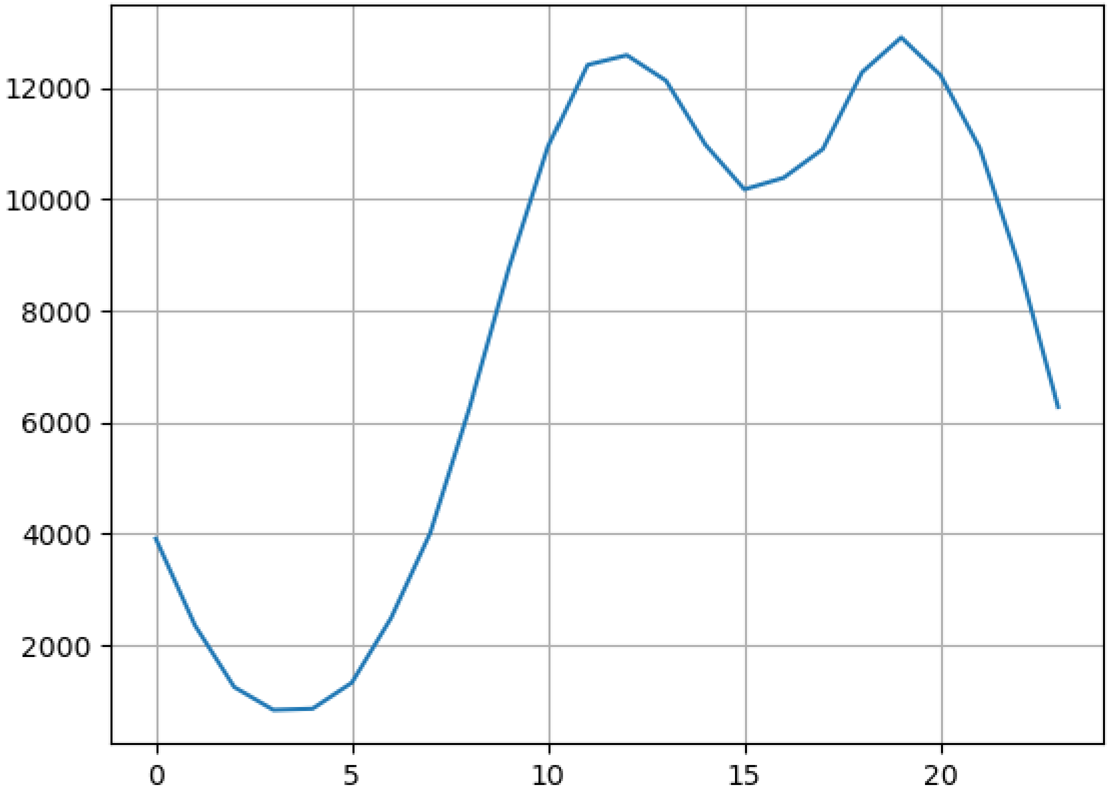
In [26]:
```
#all_data['Order  Date'][0].dtype
```

In [27]:
```
all_data['Hour'] = pd.to_datetime(all_data['Order  Date']).dt.hour
```

In [28]:
```
keys=[]
hour=[]
for key,hour_df in all_data.groupby('Hour'):
  keys.append(key)
  hour.append(len(hour_df))
```

In [29]:
```
plt.grid()
plt.plot(keys,hour)
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x23349587100>]
```

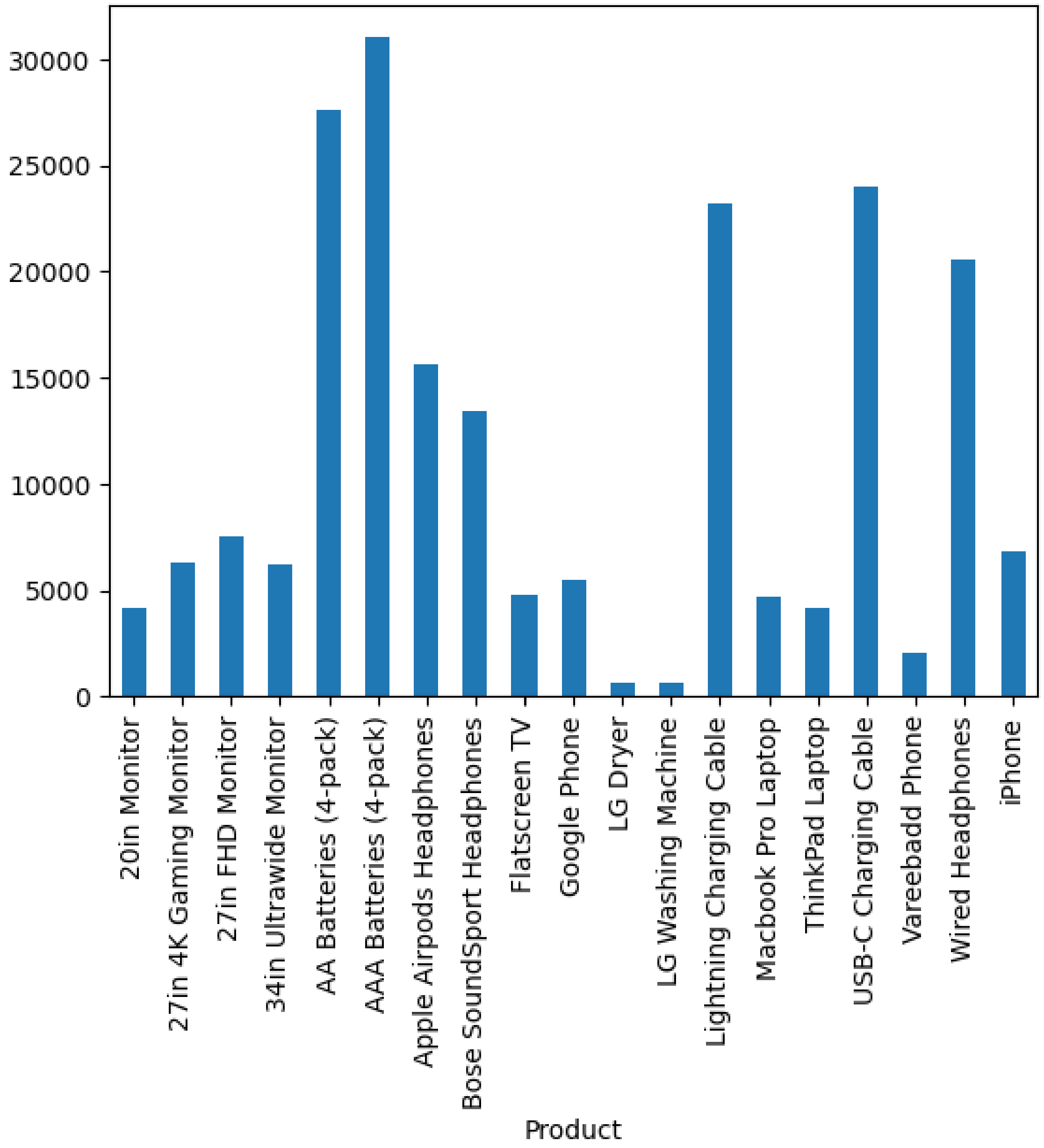


Betwen 12pm and 7pm is probably the best time to advertise to maximise product purchase

what product sold be most? & why?

```
In [30]: all_data.groupby('Product')['Quantity  Ordered'].sum().plot(kind='bar')
```

```
Out[30]: <AxesSubplot:xlabel='Product'>
```



In [31]:
```
all_data.groupby('Product')['Price  Each'].mean()
```

Out[31]:
```
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 [32]:

```
products=all_data.groupby('Product')['Quantity Ordered'].sum().index
quantity=all_data.groupby('Product')['Quantity Ordered'].sum()
prices=all_data.groupby('Product')['Price Each'].mean()
```

In [33]:

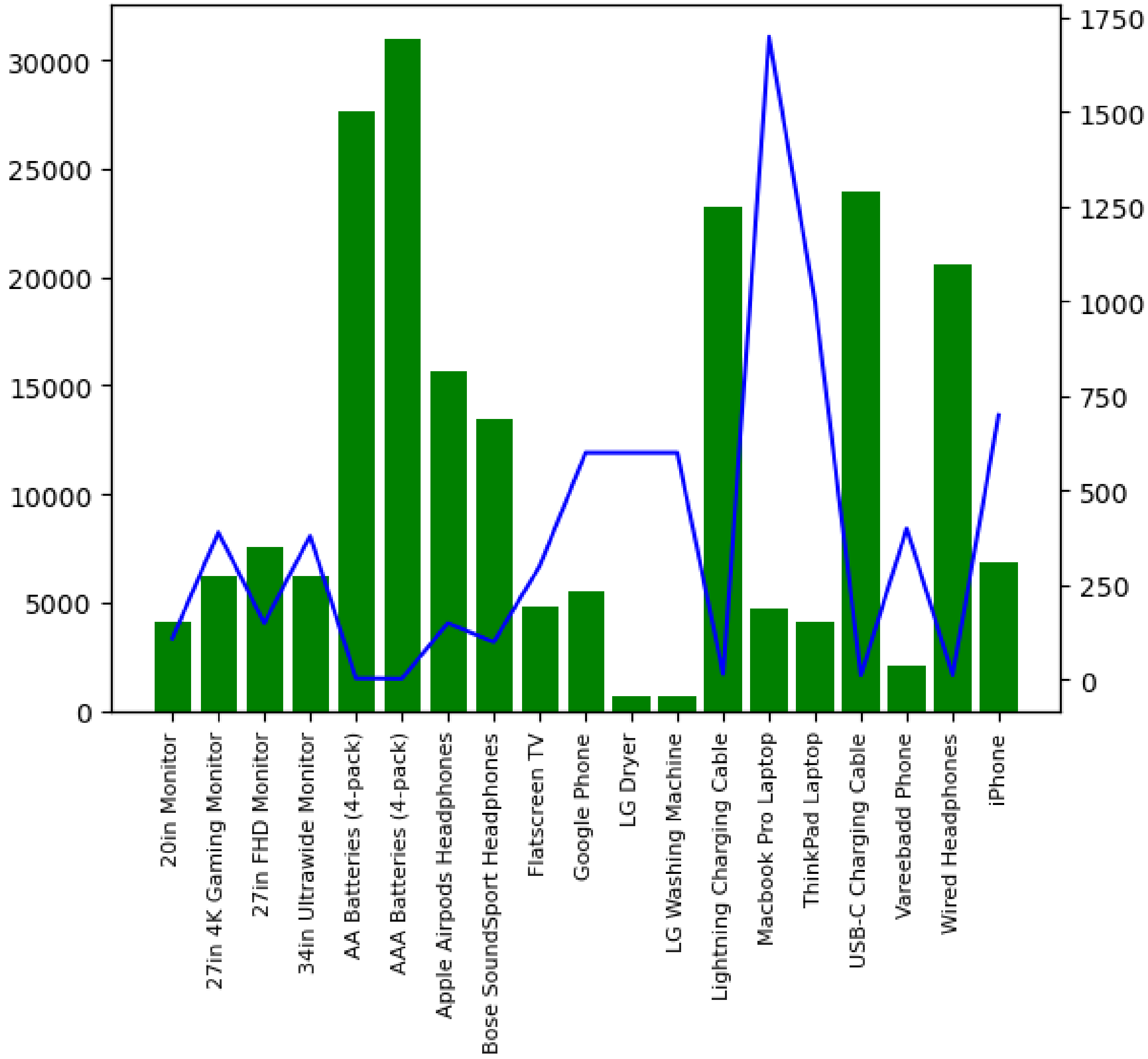
```
plt.figure(figsize=(40,24))
fig,ax1 = plt.subplots()
ax2=ax1.twinx()
ax1.bar(products, quantity, color='g')
ax2.plot(products, prices, 'b-')
ax1.set_xticklabels(products, rotation='vertical', size=8)
```

```
C:\Users\ACER\AppData\Local\Temp\ipykernel_4672\2263540929.py:6: UserWarning:

FixedFormatter should only be used together with FixedLocator
```

Out[33]:

```
[Text(0, 0, '20in Monitor'),
 Text(1, 0, '27in 4K Gaming Monitor'),
 Text(2, 0, '27in FHD Monitor'),
 Text(3, 0, '34in Ultrawide Monitor'),
 Text(4, 0, 'AA Batteries (4-pack)'),
 Text(5, 0, 'AAA Batteries (4-pack)'),
 Text(6, 0, 'Apple Airpods Headphones'),
 Text(7, 0, 'Bose SoundSport Headphones'),
 Text(8, 0, 'Flatscreen TV'),
 Text(9, 0, 'Google Phone'),
 Text(10, 0, 'LG Dryer'),
 Text(11, 0, 'LG Washing Machine'),
 Text(12, 0, 'Lightning Charging Cable'),
 Text(13, 0, 'Macbook Pro Laptop'),
 Text(14, 0, 'ThinkPad Laptop'),
 Text(15, 0, 'USB-C Charging Cable'),
 Text(16, 0, 'Vareebadd Phone'),
 Text(17, 0, 'Wired Headphones'),
 Text(18, 0, 'iPhone')]
<Figure size 4000x2400 with 0 Axes>
```



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

In [34]:
```
all_data.shape
```

Out[34]:
```
(185950, 13)
```

In [35]:
```
df=all_data[all_data['Order ID'].duplicated(keep=False)]
df.head(5)
```

Out[35]:

	Unnamed: 0	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
16	16	295681	Google Phone	1	600.00	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	2019-12-25 12:37:00	600.00	Boston
17	17	295681	USB-C Charging Cable	1	11.95	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	2019-12-25 12:37:00	11.95	Boston
18	18	295681	Bose SoundSport Headphones	1	99.99	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	2019-12-25 12:37:00	99.99	Boston
19	19	295681	Wired Headphones	1	11.99	2019-12-25 12:37:00	79 Elm St, Boston, MA 02215	2019-12-25 12:37:00	11.99	Boston
36	36	295698	Vareebadd Phone	1	400.00	2019-12-13 14:32:00	175 1st St, New York City, NY 10001	2019-12-13 14:32:00	400.00	New York City

In [36]:

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

```
C:\Users\ACER\AppData\Local\Temp\ipykernel_4672\2345761670.py:1: SettingWithCopyWa
rning:


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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [37]:

```
df.shape
```

Out[37]:

```
(14649, 14)
```

In [38]:

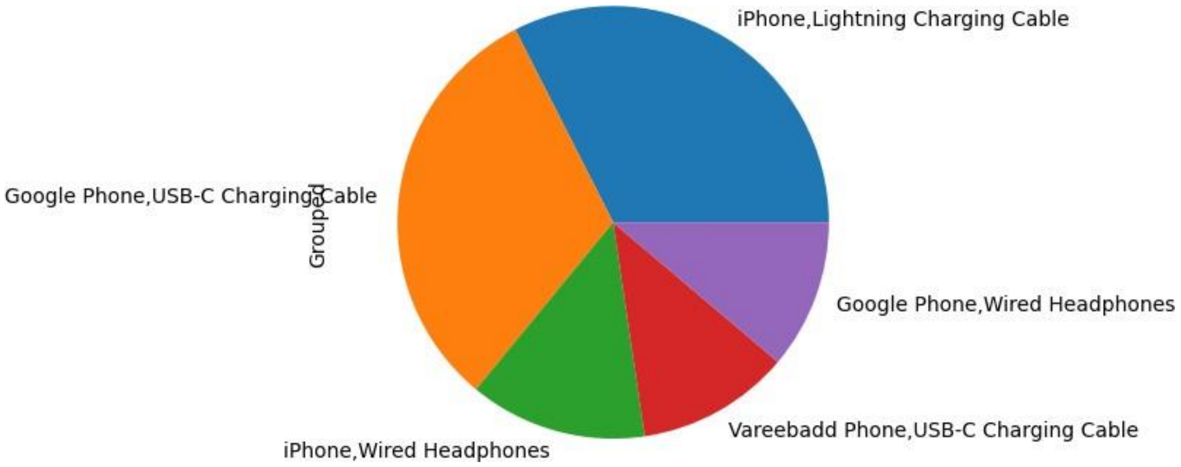
```
#lets drop out all duplicate Order ID
df2 = df.drop_duplicates(subset=['Order ID'])
```

In [39]:

```
df2['Grouped'].value_counts()[0:5].plot.pie()
```

Out[39]:

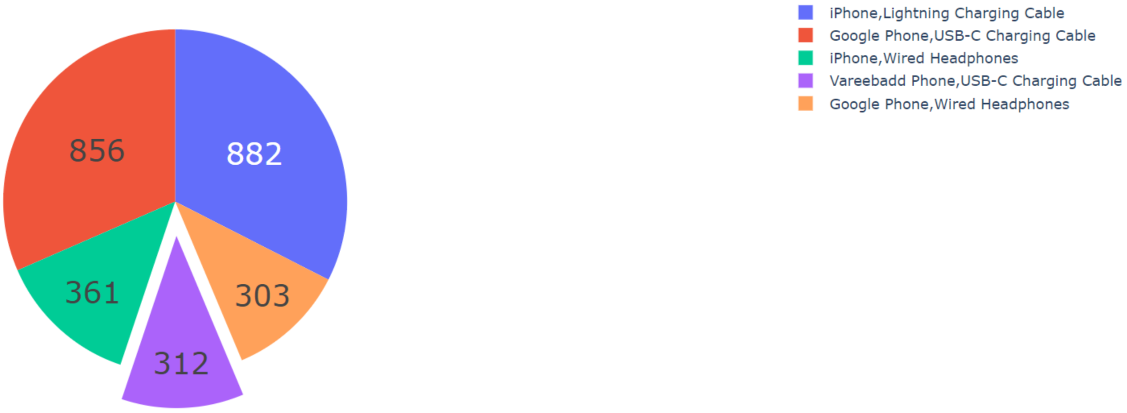
```
<AxesSubplot:ylabel='Grouped'>
```



```
In [40]: values=df2['Grouped'].value_counts()[0:5]
         labels=df['Grouped'].value_counts()[0:5].index
```

```
In [41]: trace=go.Pie(labels=labels,  values=values,
                      hoverinfo='label+percent', textinfo='value',
                      textfont=dict(size=25),
                      pull=[0, 0, 0, 0.2, 0]
                      )
```

```
In [42]: iplot([trace])
```



Results:

1. The dataset shows 19 products with order quantity of 209,000 units, sold in different 9 cities with a revenue of $34.48M
2. Top 5 selling products are: AAA batteries (4pack), AA batteries (4pack),USB-C Charging Cables, Lighting changing cable, Wired Headphone and their sales units 31012,27635,23971,23211 and 20553 respectively.
3. Top 3 low selling products are: Macbook Pro Laptop, Tinkpad Laptop, 20in Monitor, LG Washing Machine LG Dryer with their sales units 4727, 4128, 4126, 666 and 646 respectively.
4. Looking at the city with highest sales order, San Francisco ranged highest while Austin have the least order.
5. The month with the highest sales is December 2019 while September 2019 was recorded to have the least sales order. 2020 was not the spot light, however January 2020 was also analyzed and have the least sales order.
6. The products Macbook Pro Laptop have the highest revenue generation while AA batteries (4pack) is the least.
7. Product by category, digital devices which comprises Phone, Laptop and others have the highest sales order while home appliances such as LG Dryer, LG washing Machine ranked to have lowest sales order.