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

In [6]:
```
all_data=pd.read_csv("D:\\kanishka_723\\all_data.csv")
```

In [7]:
```
all_data.head
```

Out[7]:
```
<bound method NDFrame.head of        Order ID                     Product Quan
tity Ordered Price Each  \
0       176558        USB-C Charging Cable                2      11.95
1          NaN                         NaN              NaN        NaN
2       176559  Bose SoundSport Headphones                1      99.99
3       176560                Google Phone                1        600
4       176560            Wired Headphones                1      11.99
...        ...                         ...              ...        ...
15874   191703                Google Phone                1        600
15875   191703  Bose SoundSport Headphones                1      99.99
15876   191704    Lightning Charging Cable                1      14.95
15877   191705    Apple Airpods Headphones                1        150
15878   191706      AA Batteries (4-pack)                1       3.84

           Order Date                      Purchase Address
0       04/19/19 8:46          917 1st St, Dallas, TX 75001
1                 NaN                                   NaN
2      04/07/19 22:30     682 Chestnut St, Boston, MA 02215
3      04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4      04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
...               ...                                   ...
15874  04/05/19 18:05  689 11th St, San Francisco, CA 94016
15875  04/05/19 18:05  689 11th St, San Francisco, CA 94016
15876  04/16/19 14:10    841 10th St, Los Angeles, CA 90001
15877  04/30/19 15:34  992 Jefferson St, Portland, OR 97035
15878  04/20/19 18:15     742 Highland St, Austin, TX 73301

[15879 rows x 6 columns]>
```

In [8]:
```
# Find NAN
nan_df = all_data[all_data.isna().any(axis=1)]
display(nan_df.head())

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

	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

Out[8]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 8:46	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 9:27	333 8th St, Los Angeles, CA 90001

get read of text in order date column

In [9]:
```
all_data = all_data[all_data['Order Date'].str[0:2]!='Or']
```

make columns correct type

In [10]:
```
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
```

argument data with additional columns

In [11]:
```
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
```

Out[11]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 8:46	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 9:27	333 8th St, Los Angeles, CA 90001	4

add month column (alternative method)

In [12]:
```
all_data['Month 2'] = pd.to_datetime(all_data['Order Date']).dt.month
all_data.head()
```

Out[12]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Month 2
0	176558	USB-C Charging Cable	2	11.95	04/19/19 8:46	917 1st St, Dallas, TX 75001	4	4
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	4
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	4
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	4
5	176561	Wired Headphones	1	11.99	04/30/19 9:27	333 8th St, Los Angeles, CA 90001	4	4

add city column

In [13]:
```
def get_city(address):
    return address.split(",")[1].strip(" ")

def get_state(address):
    return address.split(",")[2].split(" ")[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x: f"{get_city(x)
all_data.head()
```

Out[13]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Month 2	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 8:46	917 1st St, Dallas, TX 75001	4	4	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	4	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	4	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	4	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 9:27	333 8th St, Los Angeles, CA 90001	4	4	Los Angeles (CA)

data exploration

Q_1 what was the best month for sales? how much was earned that month?

In [15]:
```
all_data['Sales'] = all_data['Quantity Ordered'].astype('int') * all_data['Pri
```

In [16]:
```
all_data.groupby(['Month']).sum()
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_19520\2666040485.py:1: FutureWarn
ing: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
In a future version, numeric_only will default to False. Either specify numer
ic_only or select only columns which should be valid for the function.
  all_data.groupby(['Month']).sum()
```

Out[16]:

	Quantity Ordered	Price Each	Month 2	Sales
Month				
4	17739	2899439.68	63088	2918954.40
5	26	8851.62	125	8855.46

Q_2 what city sold the most product?

In [17]:
```
city_max=all_data.groupby(['City']).sum()
print(max(city_max))
```

```
Sales

C:\Users\kanis\AppData\Local\Temp\ipykernel_19520\801093808.py:1: FutureWarni
ng: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
In a future version, numeric_only will default to False. Either specify numer
ic_only or select only columns which should be valid for the function.
  city_max=all_data.groupby(['City']).sum()
```

Q_3 what products are most often sold together

In [18]:
```
df = all_data[all_data['Order ID'].duplicated(keep=False)]

# Referenced: https://stackoverflow.com/questions/27298178/concatenate-strings
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.joir
df2 = df[['Order ID', 'Grouped']].drop_duplicates()
print(df['Grouped'])
```

```
3                             Google Phone,Wired Headphones
4                             Google Phone,Wired Headphones
18                        Google Phone,USB-C Charging Cable
19                        Google Phone,USB-C Charging Cable
30         Bose SoundSport Headphones,Bose SoundSport Hea...
                                ...
15787                 USB-C Charging Cable,Wired Headphones
15818              Vareebadd Phone,Lightning Charging Cable
15819              Vareebadd Phone,Lightning Charging Cable
15874               Google Phone,Bose SoundSport Headphones
15875               Google Phone,Bose SoundSport Headphones
Name: Grouped, Length: 1269, dtype: object

C:\Users\kanis\AppData\Local\Temp\ipykernel_19520\4070466232.py:4: SettingWit
hCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.j
oin(x))
```

In [21]:
```python
from itertools import combinations
from collections import Counter

count = Counter()

for row in df2['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

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

```
('iPhone', 'Lightning Charging Cable') 94
('Google Phone', 'USB-C Charging Cable') 92
('Google Phone', 'Wired Headphones') 34
('iPhone', 'Wired Headphones') 33
('Vareebadd Phone', 'USB-C Charging Cable') 32
('iPhone', 'Apple Airpods Headphones') 29
('Google Phone', 'Bose SoundSport Headphones') 20
('Vareebadd Phone', 'Wired Headphones') 15
('USB-C Charging Cable', 'Wired Headphones') 11
('AA Batteries (4-pack)', 'Apple Airpods Headphones') 7
```

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

In [22]:
```python
product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_19520\1112885426.py:2: FutureWarn
ing: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
In a future version, numeric_only will default to False. Either specify numer
ic_only or select only columns which should be valid for the function.
  quantity_ordered = product_group.sum()['Quantity Ordered']
```

In [23]:
```
print(quantity_ordered)
```

```
Product
20in Monitor                   345
27in 4K Gaming Monitor         491
27in FHD Monitor               633
34in Ultrawide Monitor         563
AA Batteries (4-pack)         2446
AAA Batteries (4-pack)        2559
Apple Airpods Headphones      1303
Bose SoundSport Headphones    1110
Flatscreen TV                  398
Google Phone                   497
LG Dryer                        69
LG Washing Machine              56
Lightning Charging Cable      2027
Macbook Pro Laptop             400
ThinkPad Laptop                329
USB-C Charging Cable          1938
Vareebadd Phone                185
Wired Headphones              1823
iPhone                         593
Name: Quantity Ordered, dtype: int64
```

In [24]:
```
prices = all_data.groupby('Product').mean()['Price Each']
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_19520\1171195910.py:1: FutureWarn
ing: The default value of numeric_only in DataFrameGroupBy.mean is deprecate
d. In a future version, numeric_only will default to False. Either specify nu
meric_only or select only columns which should be valid for the function.
  prices = all_data.groupby('Product').mean()['Price Each']
```

In [25]:
```
print(prices)
```

```
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 [ ]:
```