

import numpy as np

import pandas as pd

all\_data=pd.read\_csv("/content/sample\_data/csv1.csv")

all\_data.head()

In [1]:

Order ID

Product

Quantity

Ordered

Price

Each

Order Date

Purchase Address

0

176559.0

Bose SoundSport

Headphones

1.0

99.99

4

/

7/2019

22:30

Chestnut St,

682

Boston, MA 02215

1

176560.0

Google Phone

1.0

600.00

4/12/2019

14:38

Spruce St, Los

669

Angeles, CA 90001

2

176560.0

Wired Headphones

1.0

11.99

4/12/2019

14:38

669

Spruce St, Los

Angeles, CA 90001

3

176561.0

Wired Headphones

1.0

11.99

5/30/2019

9:27

333 8

th St, Los

Angeles, CA 90001

4

176562.0

USB-C Charging

Cable

1.0

11.95

4/29/2019

13:03

381

Wilson St, San

Francisco, CA 94016

Out [1]:

#Clean up the data!

all\_data.shape

In [2]:

(69

,

6)

Out [2]:

#Drop rows of NAN

#find NAN

nan\_df= all\_data[all\_data.isna().any(axis=1)]

display(nan\_df.head())

all\_data.shape

all\_data =all\_data.dropna(how='all')

all\_data.head()

all\_data.shape

In [3]:

Order ID

Product

Quantity Ordered

Price Each

Order Date

Purchase Address

36

NaN

NaN

NaN

NaN

NaN

NaN

51

NaN

NaN

NaN

NaN

NaN

NaN

6)

(67

,

Out [3]:

#Get rid of text in order date column

all\_data= all\_data[all\_data['Order Date'].str[0:2]!='Or']

print(all\_data)

In [4]:

Order ID Product Quantity Ordered Price Each \

0

176559.0 Bose SoundSport Headphones 1.0 99.99

1

176560.0 Google Phone 1.0 600.00

2

176560.0 Wired Headphones 1.0 11.99

3

176561.0 Wired Headphones 1.0 11.99

4

176562.0 USB-C Charging Cable 1.0 11.95

.. ... ... ... ...

64

259329.0 Lightning Charging Cable 1.0 14.95

65

259330.0 AA Batteries (4-pack) 2.0 3.84

259331.0 Apple Airpods Headphones 1.0 150.00

66

259332.0 Apple Airpods Headphones 1.0 150.00

67

259333.0 Bose SoundSport Headphones 1.0 99.99

68

Order Date Purchase Address

0

4/7/2019 22:30 682 Chestnut St, Boston, MA 02215

1

4/12/2019 14:38 669 Spruce St, Los Angeles, CA 90001

4/12/2019 14:38 669 Spruce St, Los Angeles, CA 90001

2

3

5/30/2019 9:27 333 8th St, Los Angeles, CA 90001

4

4/29/2019 13:03 381 Wilson St, San Francisco, CA 94016

.. ... ...

64

9/5/2019 19:00 480 Lincoln St, Atlanta, GA 30301

9/25/2019 22:01 763 Washington St, Seattle, WA 98101

65

9/29/2019 7:00 770 4th St, New York City, NY 10001

66

67

9/16/2019 19:21 782 Lake St, Atlanta, GA 30301

68

9/19/2019 18:03 347 Ridge St, San Francisco, CA 94016

]

[67

rows x 6 columns

#Make columns correct type

all\_data['Quantity Ordered']= pd.to\_numeric(all\_data['Quantity Ordered'])

all\_data['Price Each']= pd.to\_numeric(all\_data['Price Each'])

In [5]:

#Augment data with additional columns

#Add month column (alternative method)

all\_data['Month 2']= pd.to\_datetime(all\_data['Order Date']).dt.month

all\_data.head()

In [7]:

Order ID

Product

Quantity

Ordered

Price

Each

Order Date

Purchase

Address

Month

Month

2

0

176559.0

Bose

SoundSport

Headphones

1.0

99.99

4

/

7/2019

22:30

682

Chestnut

St, Boston,

MA 02215

4

/

4

1

176560.0

Google Phone

1.0

600.00

4/12/2019

14:38

669

Spruce

St, Los

Angeles, CA

90001

4

/

4

2

176560.0

Wired

Headphones

1.0

11.99

4/12/2019

14:38

Spruce

669

St, Los

Angeles, CA

90001

/

4

4

3

176561.0

Wired

Headphones

1.0

11.99

5/30/2019

9:27

333 8

th St,

Los Angeles,

CA 90001

5

/

5

4

176562.0

USB-C

Charging

Cable

1.0

11.95

4/29/2019

13:03

381

Wilson

St, San

Francisco,

CA 94016

4

/

4

Out [7]:

#Add city column

def get\_city(address):

return address.split(",")[1].strip(" ")

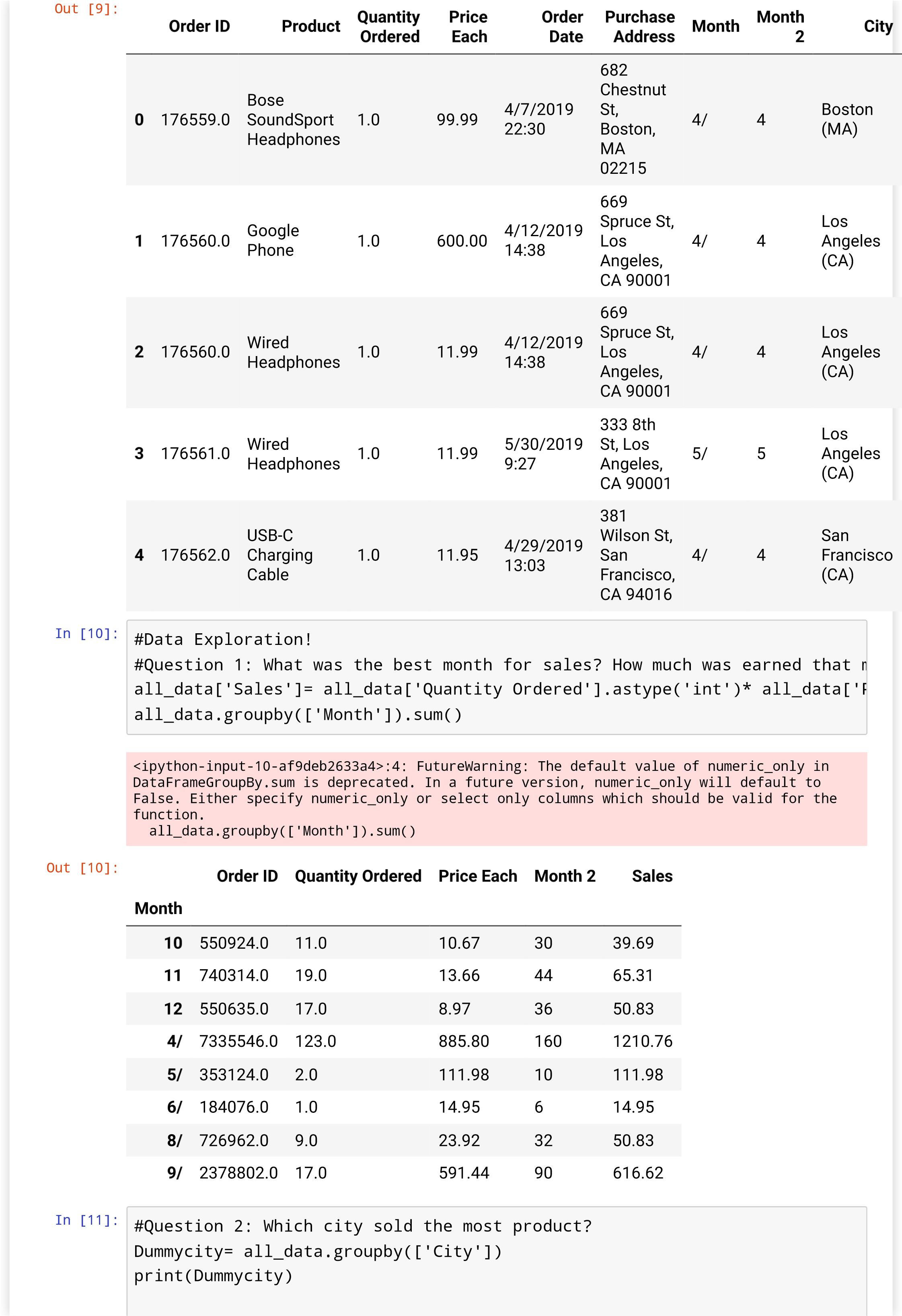
def get\_state(address):

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

all\_data['City']= all\_data['Purchase Address'].apply(lambda x: f"{get\_city

all\_data.head()

In [9]:



#city\_max= all\_data.groupby(['City']).sum()

#print(max(city\_max))

<

pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f2ddc22ed

10>

#Question 4: What products are most often sold together?

df =all\_data[all\_data['Order ID'].duplicated(keep=False)]

#Referenced: https://stackoverflow.com/questions/27298178/concatenate-stri

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

df2= df[['Order ID','Grouped']].drop\_duplicates()

print(df['Grouped'])

In [13]:

1

Google Phone,Wired Headphones

2

Google Phone,Wired Headphones

Name: Grouped, dtype: object

ipython-input-13-9bcb872e5b74>: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

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

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)

In [14]:

(

'Google Phone', 'Wired Headphones')

1

#Which Product sold the most? Why do you think it sold the most?

product\_group= all\_data.groupby('Product')

quantity\_ordered= product\_group.sum()['Quantity Ordered']

print(quantity\_ordered)

In [16]:

Product

AA Batteries (4-pack) 64.0

AAA Batteries (4-pack) 109.0

Apple Airpods Headphones 3.0

Bose SoundSport Headphones 3.0

Google Phone 1.0

Lightning Charging Cable 4.0

USB-C Charging Cable 8.0

Wired Headphones 7.0

Name: Quantity Ordered, dtype: float64

<

ipython-input-16-3b5f90fb32af>:3: FutureWarning: The default value of numeric\_only in

DataFrameGroupBy.sum is deprecated. In a future version, numeric\_only will default to

False. Either specify numeric\_only or select only columns which should be valid for the

function.

quantity\_ordered= product\_group.sum()['Quantity Ordered']

prices= all\_data.groupby('Product').mean()['Price Each']

In [17]:

<

ipython-input-17-2c255f3ab494>:1: FutureWarning: The default value of numeric\_only in

DataFrameGroupBy.mean is deprecated. In a future version, numeric\_only will default to

False. Either specify numeric\_only or select only columns which should be valid for the

function.

prices= all\_data.groupby('Product').mean()['Price Each']

print(prices)

In [18]:

Product

AA Batteries (4-pack) 3.84

AAA Batteries (4-pack) 2.99

Apple Airpods Headphones 150.00

Bose SoundSport Headphones 99.99

Google Phone 600.00

Lightning Charging Cable 14.95

USB-C Charging Cable 11.95

Wired Headphones 11.99

Name: Price Each, dtype: float64