

```
In [1]: #Akanksha Kumbhar
#202201040112
#739 G
```

```
import numpy as np
import pandas as pd
```

```
In [3]: all_data=pd.read_csv("D:\Akanksha\all_data.csv")
```

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

```
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 8:46	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

CLEAN UP THE DATA!

```
In [5]: # 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
366	NaN	NaN	NaN	NaN	NaN	NaN
736	NaN	NaN	NaN	NaN	NaN	NaN
1433	NaN	NaN	NaN	NaN	NaN	NaN

```
356 NaN NaN NaN NaN NaN
735 NaN NaN NaN NaN NaN
1433 NaN NaN NaN NaN NaN
1553 NaN NaN NaN NaN NaN
```

Out[5]:

	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 rid of text in order date column

```
In [7]: all_data = all_data[all_data['Order Date'].str[0:2]!='Or']
```

Make columns correct type

```
In [8]: all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
```

Augment data with additional columns

```
In [9]: all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
```

Out[9]:

	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

Out[9]:

	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

In [10]: `all_data['Month 2'] = pd.to_datetime(all_data['Order Date']).dt.month
all_data.head()`

Out[10]:

	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 [11]: `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)} ({get_state(x)})")
all_data.head()`

Out[11]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Month 2
--	----------	---------	------------------	------------	------------	------------------	-------	---------



Out[11]:

	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)

In [12]: `all_data['Sales'] = all_data['Quantity Ordered'].astype('int') * all_data['Price Each'].astype('float')`

In [13]: `all_data.groupby(['Month']).sum()`

C:\Users\Omkar\AppData\Local\Temp\ipykernel_28816\2666040485.py:1: 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.

Out[13]:

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

In [14]: `city_max=all_data.groupby(['City']).sum()
print(max(city_max))`

Sales

C:\Users\Omkar\AppData\Local\Temp\ipykernel_28816\801093808.py:1: 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.

In [17]: `product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']`

C:\Users\Omkar\AppData\Local\Temp\ipykernel_28816\1112885426.py:2: 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.

In [18]: `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 [19]: `prices = all_data.groupby('Product').mean()['Price Each']`

```
C:\Users\Omkar\AppData\Local\Temp\ipykernel_28816\1171195910.py: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']
```

In [20]: `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
```

```
In [19]: prices = all_data.groupby('Product').mean()['Price Each']
```

C:\Users\Omkar\AppData\Local\Temp\ipykernel_28816\1171195910.py: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.

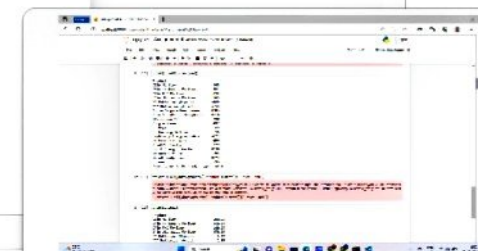
```
prices = all_data.groupby('Product').mean()['Price Each']
```

```
In [20]: 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 [ ]:
```



Snipping Tool

Screenshot copied to clipboard and saved
Select here to mark up and share the image