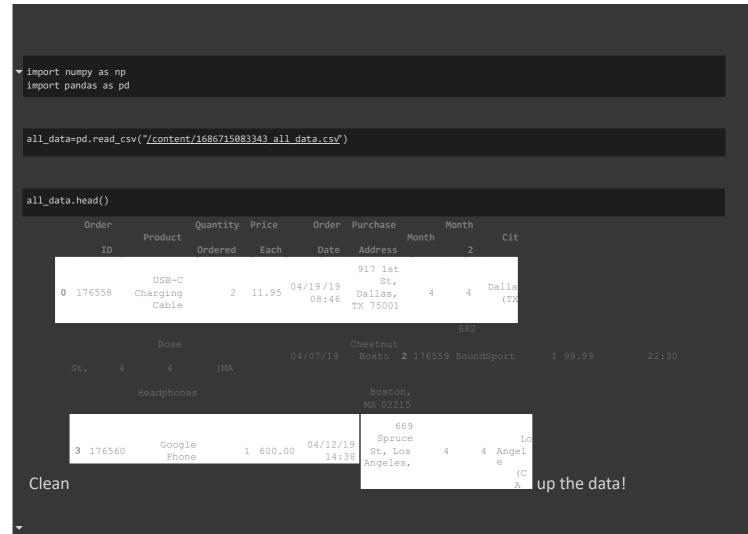
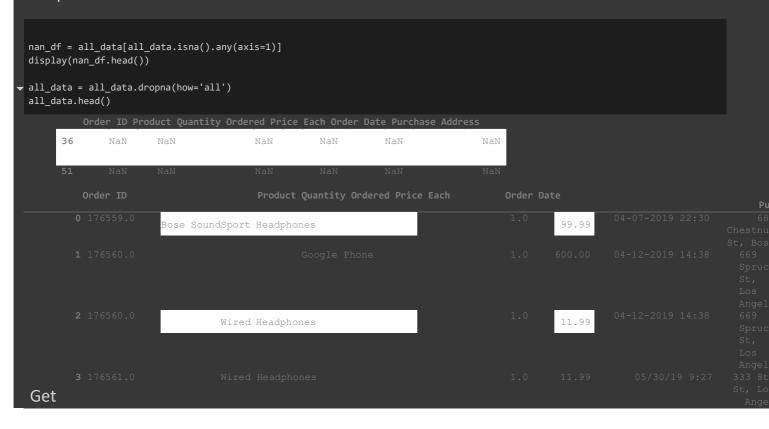
Name=Lokesh sarode Roll=158



Drop rows of NAN



4 176562.0 USB-C Charging Cable 1.0 11.95 04/29/19 13:03 381 Wilson

St, Sa Francis

rid of text in order date column

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

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'])
```

Augment data with additional columns

Add month column¶

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

Order ID Product Quantity Ordered Price Each Order Date Pur
```

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

Ad 1 month column (alternative method)

```
all_d ata['Month 2'] = pd.to_datetime(all_data['Order Date']).dt.month
all_d ata.head()
```

	Order ID			Product Quantity Ordered	Price Each
		Bose SoundSport Headphones	1.0		99.99
		Google Phone	1.0		600.00
		Wired Headphones	1.0		11.99
Add	3 176561.0	Wired Headphones	1.0		11.99

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(x)} ({get_state(x)})") all_data.head()
```

Order ID	Product Quantity Orde	ered Price Each	Order Date	Pur
0 176559.0	Bose SoundSport Headphones	1.0 99.99	04-07-2019 22:30	
1 176560.0	Google Phone	1.0 600.00	04-12-2019 14:38	669 Spruce St, Los Angel
2 176560.0	Wired Headphones	1.0	04-12-2019 14:38	
3 176561.0	Wired Headphones	1.0 11.99	05/30/19 9:27	
4 176562.0	USB-C Charging Cable	1.0	04/29/19 13:03	381 Wilson St, San Francis

→ Data Exploration!

Questic 1: What was the best month for sales? How much was earned that month?

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

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

<pre><ipython-input-14-dce0a735c05d>:1: FutureWarning:</ipython-input-14-dce0a735c05d></pre>	The default	value of	numeric_only	in DataFrameGroupB	y.sum is	5
<pre>deprecated. all data.groupby(['Month']).sum()</pre>						

	Order ID Qua	intity Ordered Price	Each Month	2 Sal	les Month
4	7335546.0	123.0	885.80	160	1210.76
6	184076.0	1.0	14.95	6	14.95
8	726962.0	9.0	23.92	32	50.83
9	2378802.0	17.0	591.44	90	616.62
10	550924.0	11.0	10.67	30	39.69
11	740314.0	19.0	13.66	44	65.31
12	550635.0	17.0	8.97	36	50.83

Question 2: What city sold the most product?

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

م [د ک

<ipython-input-15-79b556d70b46>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is
deprecated. city_max=all_data.groupby(['City']).sum()

€

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-strings-from-several-rows-using-pandas-groupby df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x)) df2 = df[['Order ID', 'Grouped']].drop_duplicates() print(df['Grouped'])
 - 1 Google Phone, Wired Headphones
 - 2 Google Phone, Wired Headphones Name: Grouped, dtype: object <ipython-input-16-9a93a24e3a06>:4: 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

```
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)

    ('Google Phone', 'Wired Headphones') 1
```

→ What 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']
     <ipython-input-18-4815a60ac quantity_ordered =</pre>
                                                                    30b>:2: FutureWarning: The default value of numeric_only in DataFram
      product_g
                                                            roup.sum()['Quantity Ordered']
print(quantity_ordered)
     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
prices = all_data.groupby('Product').mean()['Price Each']
     <ipython-input-20-225049d1ed32>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprec
      all data.groupby('Product').mean()['Price Each']
                                                                                                                            >
print(prices)
```

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