question-2

November 15, 2023

1 QUESTION 2

NAME: RISHAV KUMAR

ROLL NO. 2301560042 MY GITHUB LINK: click me

2 pandas-astype

This was taken from: 'pandas-astype.ipynb Link'

[ ]:
```
import pandas as pd

data = {
    'Product_ID': [101, 102, 103, 104, 105],
    'Product_Name': ['iPhone', 'MacBook Pro', 'iPad', 'Apple Watch', 'AirPods'],
    'Release_Year': ['2020', '2019', '2018', '2021', '2019'],
    'Price ($)': [999, 1299, 329, 399, 159],
    'Units_Sold_Millions': ['75', '20', '40', '25', '30']
}

df = pd.DataFrame(data)
```

[ ]:
```
df
```

	Product_ID	Product_Name	Release_Year	Price ($)	Units_Sold_Millions
0	101	iPhone	2020	999	75
1	102	MacBook Pro	2019	1299	20
2	103	iPad	2018	329	40
3	104	Apple Watch	2021	399	25
4	105	AirPods	2019	159	30

[ ]:
```
df.dtypes
```

```
Product_ID              int64
Product_Name           object
Release_Year           object
Price ($)               int64
```

```
Units_Sold_Millions    object
dtype: object
```

[ ]:
```
df.astype({"Release_Year":'int64'}).dtypes
```

```
Product_ID              int64
Product_Name           object
Release_Year            int64
Price ($)               int64
Units_Sold_Millions    object
dtype: object
```

[ ]:
```
df.astype({'Units_Sold_Millions':'int64'}).dtypes
```

```
Product_ID              int64
Product_Name           object
Release_Year           object
Price ($)               int64
Units_Sold_Millions     int64
dtype: object
```

[ ]:
```
df.astype({'Price ($)': 'float64'}).dtypes
```

```
Product_ID              int64
Product_Name           object
Release_Year           object
Price ($)             float64
Units_Sold_Millions     object
dtype: object
```

[ ]:
```
df.astype({'Product_ID': 'str'}).dtypes
```

```
Product_ID             object
Product_Name           object
Release_Year           object
Price ($)               int64
Units_Sold_Millions    object
dtype: object
```

[ ]:
```
df
```

	Product_ID	Product_Name	Release_Year	Price ($)	Units_Sold_Millions
0	101	iPhone	2020	999	75
1	102	MacBook Pro	2019	1299	20
2	103	iPad	2018	329	40
3	104	Apple Watch	2021	399	25
4	105	AirPods	2019	159	30

[ ]:
```
df.loc[:, ['Units_Sold_Millions']].astype('float64').apply(lambda x: x *␣
↪1000000)
```

```
   Units_Sold_Millions
0           75000000.0
1           20000000.0
2           40000000.0
3           25000000.0
4           30000000.0
```

[ ]:
```
df.astype('category').dtypes
```

```
Product_ID             category
Product_Name           category
Release_Year           category
Price ($)              category
Units_Sold_Millions    category
dtype: object
```

[ ]:
```
df.select_dtypes(include=['object']).map(lambda x: x.upper() if isinstance(x,␣
↪str) else x).astype('category')
```

```
  Product_Name Release_Year Units_Sold_Millions
0       IPHONE         2020                  75
1  MACBOOK PRO         2019                  20
2         IPAD         2018                  40
3  APPLE WATCH         2021                  25
4      AIRPODS         2019                  30
```

[ ]:
```
df.astype({'Product_Name':'category'}).dtypes
```

```
Product_ID                int64
Product_Name           category
Release_Year             object
Price ($)                 int64
Units_Sold_Millions      object
dtype: object
```

[ ]:
```
df.astype({'Product_Name':'category'}).loc[:, 'Product_Name']
```

```
0         iPhone
1    MacBook Pro
2           iPad
3    Apple Watch
4        AirPods
Name: Product_Name, dtype: category
Categories (5, object): ['AirPods', 'Apple Watch', 'MacBook Pro', 'iPad',␣
↪'iPhone']
```

[ ]:
```
name_map = {'iPhone':'IP', 'MacBook Pro':'MBP', 'iPad':'IPD', 'Apple Watch':
 ↪'AW', 'AirPods':'AP'}
name_map
```

```
{'iPhone': 'IP',
 'MacBook Pro': 'MBP',
 'iPad': 'IPD',
 'Apple Watch': 'AW',
 'AirPods': 'AP'}
```

[ ]:
```
df['Product_Name'] = df['Product_Name'].astype('category')
```

[ ]:
```
df.dtypes
```

```
Product_ID               int64
Product_Name          category
Release_Year            object
Price ($)                int64
Units_Sold_Millions     object
dtype: object
```

[ ]:
```
df['Product_Name']
```

```
0         iPhone
1    MacBook Pro
2           iPad
3    Apple Watch
4        AirPods
Name: Product_Name, dtype: category
Categories (5, object): ['AirPods', 'Apple Watch', 'MacBook Pro', 'iPad',␣
 ↪'iPhone']
```

[ ]:
```
df['Product_Name']= df['Product_Name'].cat.rename_categories(name_map)
```

[ ]:
```
df['Product_Name']
```

```
0     IP
1    MBP
2    IPD
3     AW
4     AP
Name: Product_Name, dtype: category
Categories (5, object): ['AP', 'AW', 'MBP', 'IPD', 'IP']
```

[ ]:
```
df.astype({'Product_Name': 'category'}).loc[:, ['Product_Name']].dtypes
```

```
Product_Name    category
dtype: object
```

[ ]:
```
df.astype({'Release_Year':'int64', 'Units_Sold_Millions':'int64'}).dtypes
```

```
Product_ID               int64
Product_Name          category
Release_Year             int64
Price ($)                int64
Units_Sold_Millions      int64
dtype: object
```

[ ]:
```

```

This was taken from: 'pandas-size.ipynb Link'

[ ]:
```
import pandas as pd
from datetime import datetime
import numpy as np
```

[ ]:
```
data = {
    'Order_ID': [1, 2, 3, 4, 5],
    'Customer_ID': [101, 102, 103, 104, 105],
    'Restaurant_ID': [201, 202, 203, 204, 205],
    'Order_Date': [datetime(2023, 8, 1), datetime(2023, 8, 2), datetime(2023,␣
 ↪8, 3), datetime(2023, 8, 4), datetime(2023, 8, 5)],
    'Order_Time': ['12:30:00', '18:45:00', '19:20:00', '13:15:00', '09:30:00'],
    'Food_Item': ['Burger', 'Sushi', 'Pasta', 'Pizza', 'Salad'],
    'Quantity': [2, 1, 3, 1, 1],
    'Price': [12.99, 18.50, 10.99, 14.99, 7.99],
    'Delivery_Time(min)': [30, 45, 40, 35, 25],
    'Rating': [4.5, 4.0, 5.0, 3.5, 4.0]
}
```

[ ]:
```
df = pd.DataFrame(data)
df
```

	Order_ID	Customer_ID	Restaurant_ID	Order_Date	Order_Time	Food_Item
0	1	101	201	2023-08-01	12:30:00	Burger
1	2	102	202	2023-08-02	18:45:00	Sushi
2	3	103	203	2023-08-03	19:20:00	Pasta
3	4	104	204	2023-08-04	13:15:00	Pizza
4	5	105	205	2023-08-05	09:30:00	Salad

	Quantity	Price	Delivery_Time(min)	Rating
0	2	12.99	30	4.5
1	1	18.50	45	4.0
2	3	10.99	40	5.0
3	1	14.99	35	3.5
4	1	7.99	25	4.0

What will the size attribute return when applied to the df DataFrame?

[ ]:
```
size= len(df)* len(df.loc[0])
size, df.size
```

```
(50, 50)
```

How would you verify if the size attribute counts the number of rows, columns, or individual elements in the DataFrame?

[ ]:
```
x=(len(df) * len(df.loc[0]) == df.size)
x
```

```
True
```

If you apply a filter to df to only include rows where Rating is greater than or equal to 4.5, what will the size attribute return for the filtered DataFrame?

[ ]:
```
l=df[df['Rating'] > 4.5].size
 l
```

```
10
```

Compare the output of df.size and df.shape. What mathematical operation could you apply to the values in shape to derive the value of size?

[ ]:
```
com=(df.size == df.shape[0]* df.shape[1])
com
```

```
True
```

How would the size attribute change if you dropped a row from df? Can you confirm this by coding it?

[ ]:
```
print("Before :",df.size)
df = df.drop(4)
print("After :",df.size)
```

```
Before : 50
After : 40
```

Create a Series object by selecting the Price column from df. What does the size attribute show for this Series object?

[ ]:
```
si= df['Price'].size
si
```

```
4
```

Does the size attribute include NaN values if they were present in the DataFrame?

[ ]:
```
print("without nan : ", df.size)
df.loc[4, 'Food_Item'] = np.nan
```

```
print("with nan : ", df.size)
```

```
without nan :  40
with nan :  50
```

How would the output of uber_eats_df.size change if you added a new column to the DataFrame? Can you confirm your answer by coding it?

[ ]:
```
df.loc[:, 'Location'] = ['NewYork', "California", "Texas", "Dallas", "Colorado"]
df.size
```

```
55
```

[ ]:
```
```

This was taken from: 'pandas-concat.ipynb Link'

[ ]:
```
import pandas as pd
```

[ ]:
```
data = {
    'Shop_ID' : [1, 2, 3, 4, 2, 5, 6],
    'Shop_Name': ['Starbucks', 'Blue Bottle', 'Verve Coffee', 'Stumptown',␣
 ↪'Blue Bottle', 'Gregorys Coffee', 'Cafe Grumpy'],
    'Location': ['Hollywood', 'San Diego', 'Hollywood', 'San Diego', 'New␣
 ↪York', 'New York', 'New York'],
    'Rating': [4.5, 4.8, 4.4, 4.3, 4.6, 4.2, 4.7],
    'Revenue': [50000, 30000, 25000, 28000, 52000, 20000, 24000]
}
shop = pd.DataFrame(data)
```

[ ]:
```
data = {
    'Shop_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Shop_Name': ['Bean Town Brews', 'Cafe Arabica', 'Espresso Junction',␣
 ↪'Morning Mocha', 'Perk Up Cafe', 'The Grindhouse', 'Java Jive', 'Steamy␣
 ↪Cups', 'Cappuccino Corner', 'Latte Lounge'],
    'Location': ['New York, NY', 'Los Angeles, CA', 'Chicago, IL', 'San␣
 ↪Francisco, CA', 'Miami, FL', 'Seattle, WA', 'Austin', 'Denver CO',␣
 ↪'Philadelphia, PH', 'Boston, BO'],
    'Rating': [4.5, 4.2, 4.0, 4.7, 4.3, 4.8, 4.0, 4.6, 4.4, 4.1],
    'Revenue': [3500, 2800, 2200, 4100, 3200, 4500, 2300, 3900, 3400, 2700]
}
shop1 = pd.DataFrame(data)
```

[ ]:
```
shop
```

	Shop_ID	Shop_Name	Location	Rating	Revenue
0	1	Starbucks	Hollywood	4.5	50000
1	2	Blue Bottle	San Diego	4.8	30000

```
2         3     Verve Coffee  Hollywood     4.4    25000
3         4        Stumptown  San Diego     4.3    28000
4         2      Blue Bottle   New York     4.6    52000
5         5  Gregorys Coffee   New York     4.2    20000
6         6      Cafe Grumpy   New York     4.7    24000
```

[ ]:
```
shop1
```

```
   Shop_ID          Shop_Name           Location  Rating  Revenue
0        1    Bean Town Brews       New York, NY     4.5     3500
1        2       Cafe Arabica    Los Angeles, CA     4.2     2800
2        3  Espresso Junction        Chicago, IL     4.0     2200
3        4      Morning Mocha  San Francisco, CA     4.7     4100
4        5       Perk Up Cafe          Miami, FL     4.3     3200
5        6     The Grindhouse        Seattle, WA     4.8     4500
6        7          Java Jive              Austin     4.0     2300
7        8        Steamy Cups          Denver CO     4.6     3900
8        9  Cappuccino Corner   Philadelphia, PH     4.4     3400
9       10       Latte Lounge         Boston, BO     4.1     2700
```

[ ]:
```
suppliers_data = {
'Shop_ID': [1, 2, 3, 4, 5, 6, 7],
'Shop_Name': ['Starbucks', 'Blue Bottle', 'Verve Coffee', 'Stumptown',␣
 ↪'Gregorys Coffee', 'Blue Bottle', 'Gregorys Coffee'],
'Supplier_Name': ['Beans R Us', 'Premium Beans', 'South Coffee Suppliers', 'San␣
 ↪Diego Beans', 'NY Fresh Beans', 'Mexico tea', 'Chayyos'],
'Delivery_Days': [7, 5, 7, 4, 6,9, 10]
}

sdf = pd.DataFrame(suppliers_data)
```

2.0.1 Basic Concatenation

How would you concatenate coffee_shops and more_coffee_shops vertically ?

[ ]:
```
pd.concat([shop, shop1])
```

```
   Shop_ID          Shop_Name           Location  Rating  Revenue
0        1           Starbucks          Hollywood     4.5    50000
1        2        Blue Bottle          San Diego     4.8    30000
2        3       Verve Coffee          Hollywood     4.4    25000
3        4          Stumptown          San Diego     4.3    28000
4        2        Blue Bottle           New York     4.6    52000
5        5    Gregorys Coffee           New York     4.2    20000
6        6        Cafe Grumpy           New York     4.7    24000
0        1    Bean Town Brews       New York, NY     4.5     3500
1        2       Cafe Arabica    Los Angeles, CA     4.2     2800
2        3  Espresso Junction        Chicago, IL     4.0     2200
```

3	4	Morning Mocha	San Francisco, CA	4.7	4100
4	5	Perk Up Cafe	Miami, FL	4.3	3200
5	6	The Grindhouse	Seattle, WA	4.8	4500
6	7	Java Jive	Austin	4.0	2300
7	8	Steamy Cups	Denver CO	4.6	3900
8	9	Cappuccino Corner	Philadelphia, PH	4.4	3400
9	10	Latte Lounge	Boston, BO	4.1	2700

2.0.2 Ignoring index

reseting the index

[ ]:
```
pd.concat([shop, shop1], ignore_index=True)
```

	Shop_ID	Shop_Name	Location	Rating	Revenue
0	1	Starbucks	Hollywood	4.5	50000
1	2	Blue Bottle	San Diego	4.8	30000
2	3	Verve Coffee	Hollywood	4.4	25000
3	4	Stumptown	San Diego	4.3	28000
4	2	Blue Bottle	New York	4.6	52000
5	5	Gregorys Coffee	New York	4.2	20000
6	6	Cafe Grumpy	New York	4.7	24000
7	1	Bean Town Brews	New York, NY	4.5	3500
8	2	Cafe Arabica	Los Angeles, CA	4.2	2800
9	3	Espresso Junction	Chicago, IL	4.0	2200
10	4	Morning Mocha	San Francisco, CA	4.7	4100
11	5	Perk Up Cafe	Miami, FL	4.3	3200
12	6	The Grindhouse	Seattle, WA	4.8	4500
13	7	Java Jive	Austin	4.0	2300
14	8	Steamy Cups	Denver CO	4.6	3900
15	9	Cappuccino Corner	Philadelphia, PH	4.4	3400
16	10	Latte Lounge	Boston, BO	4.1	2700

2.0.3 Adding multiIndex

[ ]:
```
pd.concat([shop, shop1], keys=['First','Second'])
```

		Shop_ID	Shop_Name	Location	Rating	Revenue
First	0	1	Starbucks	Hollywood	4.5	50000
	1	2	Blue Bottle	San Diego	4.8	30000
	2	3	Verve Coffee	Hollywood	4.4	25000
	3	4	Stumptown	San Diego	4.3	28000
	4	2	Blue Bottle	New York	4.6	52000
	5	5	Gregorys Coffee	New York	4.2	20000
	6	6	Cafe Grumpy	New York	4.7	24000
Second	0	1	Bean Town Brews	New York, NY	4.5	3500
	1	2	Cafe Arabica	Los Angeles, CA	4.2	2800
	2	3	Espresso Junction	Chicago, IL	4.0	2200
	3	4	Morning Mocha	San Francisco, CA	4.7	4100

4	5	Perk Up Cafe	Miami, FL	4.3	3200
5	6	The Grindhouse	Seattle, WA	4.8	4500
6	7	Java Jive	Austin	4.0	2300
7	8	Steamy Cups	Denver CO	4.6	3900
8	9	Cappuccino Corner	Philadelphia, PH	4.4	3400
9	10	Latte Lounge	Boston, BO	4.1	2700

[ ]:
```
sdf
```

	Shop_ID	Shop_Name	Supplier_Name	Delivery_Days
0	1	Starbucks	Beans R Us	7
1	2	Blue Bottle	Premium Beans	5
2	3	Verve Coffee	South Coffee Suppliers	7
3	4	Stumptown	San Diego Beans	4
4	5	Gregorys Coffee	NY Fresh Beans	6
5	6	Blue Bottle	Mexico tea	9
6	7	Gregorys Coffee	Chayyos	10

Copying sdf data to sdf1 and dropping two index

[ ]:
```
sdf1= sdf.drop(['Shop_ID', 'Shop_Name'], axis=1)
sdf1
```

	Supplier_Name	Delivery_Days
0	Beans R Us	7
1	Premium Beans	5
2	South Coffee Suppliers	7
3	San Diego Beans	4
4	NY Fresh Beans	6
5	Mexico tea	9
6	Chayyos	10

2.0.4 Adding sdf1 to shop data

[ ]:
```
shop=pd.concat([shop, sdf1], axis=1)
```

[ ]:
```
shop
```

	Shop_ID	Shop_Name	Location	Rating	Revenue	\
0	1	Starbucks	Hollywood	4.5	50000	
1	2	Blue Bottle	San Diego	4.8	30000	
2	3	Verve Coffee	Hollywood	4.4	25000	
3	4	Stumptown	San Diego	4.3	28000	
4	2	Blue Bottle	New York	4.6	52000	
5	5	Gregorys Coffee	New York	4.2	20000	
6	6	Cafe Grumpy	New York	4.7	24000	

	Supplier_Name	Delivery_Days
0	Beans R Us	7

```
1           Premium Beans             5
2  South Coffee Suppliers             7
3         San Diego Beans             4
4          NY Fresh Beans             6
5              Mexico tea             9
6                 Chayyos            10
```

[ ]:

This was taken from: 'pandas-explode.ipynb Link'

[ ]:
```
import pandas as pd
```

[ ]:
```
data = {
    'Coffee_Shop': ['Brewed Awakening', 'Coffee Cloud', 'Bean Dream', 'Espresso␣
 ↪Express', 'Latte Love',
    'Mocha Magic', 'Cafe Comfort', 'Seattle Sip', 'Drip Drop', 'Grind Ground'],
    'Location': ['Downtown', 'Capitol Hill', 'Green Lake', 'Ballard', 'West␣
 ↪Seattle',
    'Fremont', 'Queen Anne', 'Belltown', 'University District', 'Magnolia'],
    'Avg_Rating': [4.5, 4.2, 5.0, 4.8, 4.6, 4.3, 4.9, 4.0, 4.7, 4.6],
    'Coffee_Type': ['Espresso', 'Latte', 'Cappuccino', 'Americano', 'Cold Brew',
    'Macchiato', 'Espresso', 'Latte', 'Drip Coffee', 'Americano'],
    'Tables_Available': [5, 8, 6, 7, 5, 9, 7, 8, 6, 5]
}
```

[ ]:

[ ]:
```
coffee_shops_df = pd.DataFrame(data)
```

[ ]:
```
coffee_shops_df
```

```
        Coffee_Shop             Location  Avg_Rating  Coffee_Type  \
0  Brewed Awakening             Downtown         4.5     Espresso
1      Coffee Cloud         Capitol Hill         4.2        Latte
2         Bean Dream           Green Lake         5.0   Cappuccino
3  Espresso Express              Ballard         4.8    Americano
4        Latte Love         West Seattle         4.6    Cold Brew
5       Mocha Magic              Fremont         4.3    Macchiato
6      Cafe Comfort           Queen Anne         4.9     Espresso
7       Seattle Sip             Belltown         4.0        Latte
8         Drip Drop  University District         4.7  Drip Coffee
9      Grind Ground             Magnolia         4.6    Americano

   Tables_Available
0                 5
1                 8
2                 6
```

```
3                 7
4                 5
5                 9
6                 7
7                 8
8                 6
9                 5
```

If you had a column in coffee_shops_df that contained lists of coffee varieties offered, how would you use explode to create a separate row for each coffee type?

[ ]:
```
coffee_varieties = [["Arabica", "Robusta"], ["Typica", "Bourbon", "Geisha"],␣
↪"Hawaiian", "Sumatra"]
```

[ ]:
```
temp = coffee_shops_df.loc[0:3].copy()
```

[ ]:
```
temp
```

```
        Coffee_Shop      Location  Avg_Rating Coffee_Type  Tables_Available
0  Brewed Awakening      Downtown         4.5    Espresso                 5
1      Coffee Cloud  Capitol Hill         4.2       Latte                 8
2        Bean Dream    Green Lake         5.0  Cappuccino                 6
3  Espresso Express       Ballard         4.8   Americano                 7
```

[ ]:
```
temp['Coffee_Varieties'] = coffee_varieties
```

[ ]:
```
temp
```

```
        Coffee_Shop      Location  Avg_Rating Coffee_Type  Tables_Available  \
0  Brewed Awakening      Downtown         4.5    Espresso                 5
1      Coffee Cloud  Capitol Hill         4.2       Latte                 8
2        Bean Dream    Green Lake         5.0  Cappuccino                 6
3  Espresso Express       Ballard         4.8   Americano                 7

           Coffee_Varieties
0         [Arabica, Robusta]
1  [Typica, Bourbon, Geisha]
2                   Hawaiian
3                    Sumatra
```

[ ]:
```
temp.explode('Coffee_Varieties')
```

```
        Coffee_Shop      Location  Avg_Rating Coffee_Type  Tables_Available  \
0  Brewed Awakening      Downtown         4.5    Espresso                 5
0  Brewed Awakening      Downtown         4.5    Espresso                 5
1      Coffee Cloud  Capitol Hill         4.2       Latte                 8
1      Coffee Cloud  Capitol Hill         4.2       Latte                 8
1      Coffee Cloud  Capitol Hill         4.2       Latte                 8
2        Bean Dream    Green Lake         5.0  Cappuccino                 6
```

```
3  Espresso Express       Ballard         4.8   Americano                7

  Coffee_Varieties
0          Arabica
0          Robusta
1           Typica
1          Bourbon
1           Geisha
2         Hawaiian
3          Sumatra
```

[ ]:
```
temp.explode('Coffee_Varieties', ignore_index=True)
```

```
        Coffee_Shop      Location  Avg_Rating Coffee_Type  Tables_Available  \
0  Brewed Awakening      Downtown         4.5    Espresso                 5
1  Brewed Awakening      Downtown         4.5    Espresso                 5
2      Coffee Cloud  Capitol Hill         4.2       Latte                 8
3      Coffee Cloud  Capitol Hill         4.2       Latte                 8
4      Coffee Cloud  Capitol Hill         4.2       Latte                 8
5        Bean Dream    Green Lake         5.0  Cappuccino                 6
6  Espresso Express       Ballard         4.8   Americano                 7

  Coffee_Varieties
0          Arabica
1          Robusta
2           Typica
3          Bourbon
4           Geisha
5         Hawaiian
6          Sumatra
```

[ ]:
```
data = {'A' : [[1,2,3], "foo", 1, 3],
        'B' : ['bar', 'text', 'str', 'night']}
df = pd.DataFrame(data)
df
```

```
           A      B
0  [1, 2, 3]    bar
1        foo   text
2          1    str
3          3  night
```

[ ]:
```
df.explode('A')
```

```
     A      B
0    1    bar
0    2    bar
0    3    bar
```

```
1  foo   text
2    1    str
3    3  night
```

[ ]:
```
data = {'A' : ['a', 'b', 'c'],
        'B' : [1 ,3, 5],
        'C' : [2, 4, 6]}
df= pd.DataFrame(data)
df
```

```
   A  B  C
0  a  1  2
1  b  3  4
2  c  5  6
```

[ ]:
```
df.melt(id_vars=['A'], value_vars=['B', 'C'])
```

```
   A variable  value
0  a        B      1
1  b        B      3
2  c        B      5
3  a        C      2
4  b        C      4
5  c        C      6
```

[ ]:
```
df.explode('A')
```

```
   A  B  C
0  a  1  2
1  b  3  4
2  c  5  6
```

[ ]:
```

```

This was taken from: 'pandas-group-z2.ipynb Link'

[ ]:
```
import pandas as pd
```

[ ]:
```
data = {
'Shop_Name': ['Starbucks', 'Blue Bottle', 'Dunkin', 'Peets', 'Starbucks', 'La␣
 ↪Colombe', 'Blue Bottle', 'Peets', 'Starbucks', 'Dunkin'],
'Location': ['Hollywood', 'San Diego', 'San Diego', 'Hollywood', 'New York',␣
 ↪'New York', 'Hollywood', 'San Diego', 'New York', 'Hollywood'],
'Rating': [4.5, 4.7, 4.1, 4.3, 4.5, 4.8, 4.6, 4.2, 4.4, 3.9],
'Type': ['Chain', 'Independent', 'Chain', 'Chain', 'Chain', 'Independent',␣
 ↪'Independent', 'Chain', 'Chain', 'Chain']
}
```

[ ]:
```
df = pd.DataFrame(data)
```

[ ]:
```
df
```

	Shop_Name	Location	Rating	Type
0	Starbucks	Hollywood	4.5	Chain
1	Blue Bottle	San Diego	4.7	Independent
2	Dunkin	San Diego	4.1	Chain
3	Peets	Hollywood	4.3	Chain
4	Starbucks	New York	4.5	Chain
5	La Colombe	New York	4.8	Independent
6	Blue Bottle	Hollywood	4.6	Independent
7	Peets	San Diego	4.2	Chain
8	Starbucks	New York	4.4	Chain
9	Dunkin	Hollywood	3.9	Chain

1) Group by the Location column and calculate the mean of the Rating for each location. Which location has the highest average rating?

[ ]:
```
rt_mn = df.groupby('Location').agg({'Rating':'mean'})
rt_mn
```

Location	Rating
Hollywood	4.325000
New York	4.566667
San Diego	4.333333

[ ]:
```
rt_mn.sort_values('Rating', ascending=False).iloc[0]
```

```
Rating    4.566667
Name: New York, dtype: float64
```

2) Group by the Type column and calculate the total count of each type. Which type of coffee shop is most prevalent?

[ ]:
```
df.groupby('Type').agg({'Shop_Name':'count'})
```

Type	Shop_Name
Chain	7
Independent	3

[ ]:
```
df.groupby('Type').agg({'Shop_Name':'count'}).sort_values('Shop_Name').iloc[-1]
```

```
Shop_Name    7
Name: Chain, dtype: int64
```

3) Group the data by both Location and Type. What's the maximum rating a 'Chain' coffee shop received in 'Hollywood'?

[ ]:
```
ratings_loc_type = df.groupby(['Location', 'Type']).agg({'Rating':'max'})
```

[ ]:
```
ratings_loc_type
```

```
                      Rating
Location  Type
Hollywood Chain          4.5
          Independent    4.6
New York  Chain          4.5
          Independent    4.8
San Diego Chain          4.2
          Independent    4.7
```

[ ]:
```
ratings_loc_type.sort_values('Rating', ascending=False).iloc[0]
```

```
Rating    4.8
Name: (New York, Independent), dtype: float64
```

4) By grouping based on the Shop_Name column, can you calculate the range (max-min) of Rating each shop received?

[ ]:
```
df.loc[:, ['Shop_Name', 'Rating']].sort_values('Shop_Name')
```

```
     Shop_Name  Rating
1  Blue Bottle     4.7
6  Blue Bottle     4.6
2       Dunkin     4.1
9       Dunkin     3.9
5   La Colombe     4.8
3        Peets     4.3
7        Peets     4.2
0    Starbucks     4.5
4    Starbucks     4.5
8    Starbucks     4.4
```

[ ]:
```
df.groupby('Shop_Name').agg({'Rating': lambda x : x.astype(float).max() - x.
↪astype(float).min()})
```

```
             Rating
Shop_Name
Blue Bottle     0.1
Dunkin          0.2
La Colombe      0.0
Peets           0.1
Starbucks       0.1
```

5) Group by the Location and find the shop with the highest rating in each location.

```
[ ]: df.groupby('Location').max().loc[:, 'Shop_Name']
```

```
Location
Hollywood    Starbucks
New York     Starbucks
San Diego        Peets
Name: Shop_Name, dtype: object
```

6) Calculate the median rating for each Shop_Name by grouping based on the shop name. Which shop has the median rating of 4.5?

```
[ ]: med_ratings = df.groupby('Shop_Name').agg({'Rating':'median'})
```

```
[ ]: med_ratings
```

```
             Rating
Shop_Name
Blue Bottle    4.65
Dunkin         4.00
La Colombe     4.80
Peets          4.25
Starbucks      4.50
```

```
[ ]: med_ratings[med_ratings['Rating'] == 4.65]
```

```
             Rating
Shop_Name
Blue Bottle    4.65
```

7) Group by Type and find the standard deviation of Rating for both 'Independent' and 'Chain' shops. Which type has a more consistent rating?

```
[ ]: df.groupby('Type').agg({'Rating':'std'})
```

```
               Rating
Type
Chain        0.221467
Independent  0.100000
```

8) After grouping by Location, find the total number of reviews each location received (assuming each row is a unique review).

```
[ ]: df.groupby('Location').count()['Type']
```

```
Location
Hollywood    4
New York     3
San Diego    3
Name: Type, dtype: int64
```

9) Group by both Location and Type and calculate the sum of the Rating. Which combination of location and type has the highest total rating?

[ ]:
```
df.sort_values(['Location', 'Type'])
```

```
     Shop_Name   Location  Rating         Type
0    Starbucks  Hollywood     4.5        Chain
3        Peets  Hollywood     4.3        Chain
9       Dunkin  Hollywood     3.9        Chain
6  Blue Bottle  Hollywood     4.6  Independent
4    Starbucks   New York     4.5        Chain
8    Starbucks   New York     4.4        Chain
5   La Colombe   New York     4.8  Independent
2       Dunkin  San Diego     4.1        Chain
7        Peets  San Diego     4.2        Chain
1  Blue Bottle  San Diego     4.7  Independent
```

[ ]:
```
sum_ratings = df.groupby(['Location', 'Type']).agg({'Rating':'sum'})
```

[ ]:
```
sum_ratings
```

```
                      Rating
Location  Type
Hollywood Chain         12.7
          Independent    4.6
New York  Chain          8.9
          Independent    4.8
San Diego Chain          8.3
          Independent    4.7
```

[ ]:
```
sum_ratings.idxmax()
```

```
Rating    (Hollywood, Chain)
dtype: object
```

10) Group by Location and find out the 25th percentile of the Rating for each location.

[ ]:
```
df.groupby('Location')['Rating'].quantile(0.25)
```

```
Location
Hollywood    4.20
New York     4.45
San Diego    4.15
Name: Rating, dtype: float64
```

[ ]:
```

```

This file was created by me and i used it to prefom different methods of pandas libraries.

[ ]:
```
import pandas as pd
```

[ ]:
```
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
    'Age': [25, 30, 22, 35, 28],
    'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Boston'],
    'Salary': [60000, 80000, 55000, 75000, 70000]
}
```

[ ]:
```
df = pd.DataFrame(data)
```

[ ]:
```
selected_data = df.loc[df['Age'] > 25, ['Name', 'City']]
selected_data
```

```
    Name           City
1    Bob  San Francisco
3  David        Chicago
4  Emily         Boston
```

[ ]:
```
filtered_data = df[df['City'].isin(['New York', 'Chicago'])]
filtered_data
```

```
    Name  Age      City  Salary
0  Alice   25  New York   60000
3  David   35   Chicago   75000
```

[ ]:
```
missing_values = df.isnull()
missing_values
```

```
    Name    Age   City  Salary
0  False  False  False   False
1  False  False  False   False
2  False  False  False   False
3  False  False  False   False
4  False  False  False   False
```

[ ]:
```
selected_rows = df.iloc[1:4, 0:2]
selected_rows
```

```
      Name  Age
1      Bob   30
2  Charlie   22
3    David   35
```

[ ]:
```

```