```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Data Preparation

orders_df=pd.read_excel("D:/Data Science/Case
Study/global_superstore_2016.xlsx",sheet_name="Orders")
orders_df.columns
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Postal Code',
'City',
       'State', 'Country', 'Region', 'Market', 'Product ID',
'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity',
'Discount',
       'Profit', 'Shipping Cost', 'Order Priority'],
      dtype='object')
```

```python
returns_df=pd.read_excel("D:/Data Science/Case
Study/global_superstore_2016.xlsx",sheet_name="Returns")
returns_df.columns
```

```
Index(['Returned', 'Order ID', 'Region'], dtype='object')
```

```python
people_df=pd.read_excel("D:/Data Science/Case
Study/global_superstore_2016.xlsx",sheet_name="People")
people_df.columns
```

```
Index(['Person', 'Region'], dtype='object')
```

```python
merged_df=pd.merge(orders_df,returns_df,on="Order ID",how="left")
merged_df.columns
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Postal Code',
'City',
       'State', 'Country', 'Region_x', 'Market', 'Product ID',
'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity',
'Discount',
       'Profit', 'Shipping Cost', 'Order Priority', 'Returned',
'Region_y'],
      dtype='object')
```

```python
merged_df=merged_df.rename(columns={'Region_x':'Region'})
merged_df.columns
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Postal Code',
```

```
'City',
       'State', 'Country', 'Region', 'Market', 'Product ID',
'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity',
'Discount',
       'Profit', 'Shipping Cost', 'Order Priority', 'Returned',
'Region_y'],
      dtype='object')
```

```
merged_df=pd.merge(merged_df,people_df,on="Region",how="left")
merged_df.columns
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Postal Code',
'City',
       'State', 'Country', 'Region', 'Market', 'Product ID',
'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity',
'Discount',
       'Profit', 'Shipping Cost', 'Order Priority', 'Returned',
'Region_y',
       'Person'],
      dtype='object')
```

```
merged_df.head()
```

```
   Row ID                   Order ID Order Date  Ship Date     Ship
Mode  \
0   40098  CA-2014-AB10015140-41954 2014-11-11 2014-11-13   First
Class
1   26341    IN-2014-JR162107-41675 2014-02-05 2014-02-07  Second
Class
2   25330    IN-2014-CR127307-41929 2014-10-17 2014-10-18   First
Class
3   13524   ES-2014-KM1637548-41667 2014-01-28 2014-01-30   First
Class
4   47221   SG-2014-RH9495111-41948 2014-11-05 2014-11-06      Same
Day

    Customer ID     Customer Name      Segment  Postal Code
City  \
0  AB-100151402     Aaron Bergman     Consumer      73120.0  Oklahoma
City
1     JR-162107     Justin Ritter    Corporate          NaN
Wollongong
2     CR-127307      Craig Reiter     Consumer          NaN
Brisbane
3    KM-1637548  Katherine Murray  Home Office          NaN
Berlin
4    RH-9495111       Rick Hansen     Consumer          NaN
```

```
Dakar

   ...                              Product Name     Sales Quantity
Discount  \
0  ...                          Samsung Convoy 3   221.980         2
0.0
1  ...  Novimex Executive Leather Armchair, Black  3709.395         9
0.1
2  ...          Nokia Smart Phone, with Caller ID  5175.171         9
0.1
3  ...             Motorola Smart Phone, Cordless  2892.510         5
0.1
4  ...             Sharp Wireless Fax, High-Speed  2832.960         8
0.0

     Profit Shipping Cost Order Priority Returned  Region_y
Person
0   62.1544          40.77           High      NaN       NaN      Lon
Bonher
1 -288.7650         923.63       Critical      NaN       NaN     Kauri
Anaru
2  919.9710         915.49         Medium      NaN       NaN     Kauri
Anaru
3  -96.5400         910.16         Medium      NaN       NaN   Gilbert
Wolff
4  311.5200         903.04       Critical      NaN       NaN  Katlego
Akosua

[5 rows x 27 columns]

merged_df.tail()

       Row ID                  Order ID Order Date  Ship Date
Ship Mode  \
51285   29002   IN-2015-KE1642066-42174 2015-06-19 2015-06-19
Same Day
51286   34337  US-2014-ZD21925140-41765 2014-05-06 2014-05-10
Standard Class
51287   31315  CA-2012-ZD21925140-41147 2012-08-26 2012-08-31
Second Class
51288    9596   MX-2013-RB1979518-41322 2013-02-17 2013-02-21
Standard Class
51289    6147   MX-2013-MC1810093-41416 2013-05-22 2013-05-26
Second Class

        Customer ID       Customer Name      Segment  Postal Code  \
51285    KE-1642066     Katrina Edelman    Corporate          NaN
51286  ZD-219251408  Zuschuss Donatelli     Consumer      37421.0
51287  ZD-219251404  Zuschuss Donatelli     Consumer      94109.0
51288    RB-1979518          Ross Baird  Home Office          NaN
```

```
51289    MC-1810093       Mick Crebagga     Consumer          NaN

                City  ...                                    Product 
Name  \
51285           Kure  ...                  Advantus Thumb Tacks, 12 
Pack
51286    Chattanooga  ...  Eldon Image Series Desk Accessories, 
Burgundy
51287  San Francisco  ...                                     Newell 
341
51288       Valinhos  ...                        Acco Index Tab, 
Economy
51289       Tipitapa  ...        Eaton Computer Printout Paper, 8.5 x 
11

       Sales Quantity Discount  Profit Shipping Cost Order Priority 
Returned  \
51285  65.10        5      0.0  4.5000         1.010         Medium 
NaN
51286  16.72        5      0.2  3.3440         1.930           High 
NaN
51287   8.56        2      0.0  2.4824         1.580           High 
NaN
51288  13.44        2      0.0  2.4000         1.003         Medium 
NaN
51289  61.38        3      0.0  1.8000         1.002           High 
NaN

       Region_y            Person
51285       NaN      Hadia Bousaid
51286       NaN    Flannery Newton
51287       NaN    Derrick Snyders
51288       NaN    Vasco Magalhães
51289       NaN  Nicodemo Bautista

[5 rows x 27 columns]
```

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 27 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   Row ID          51290 non-null  int64         
 1   Order ID        51290 non-null  object        
 2   Order Date      51290 non-null  datetime64[ns]
 3   Ship Date       51290 non-null  datetime64[ns]
 4   Ship Mode       51290 non-null  object        
 5   Customer ID     51290 non-null  object        
```

```
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   Postal Code     9994 non-null   float64
 9   City            51290 non-null  object
 10  State           51290 non-null  object
 11  Country         51290 non-null  object
 12  Region          51290 non-null  object
 13  Market          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object
 24  Returned        2220 non-null   object
 25  Region_y        2220 non-null   object
 26  Person          50906 non-null  object
dtypes: datetime64[ns](2), float64(5), int64(2), object(18)
memory usage: 10.6+ MB
```

```
merged_df.isnull().sum()
```

```
Row ID                0
Order ID              0
Order Date            0
Ship Date             0
Ship Mode             0
Customer ID           0
Customer Name         0
Segment               0
Postal Code       41296
City                  0
State                 0
Country               0
Region                0
Market                0
Product ID            0
Category              0
Sub-Category          0
Product Name          0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping Cost         0
Order Priority        0
```

```
Returned           49070
Region_y           49070
Person               384
dtype: int64
```

```python
#Data Cleaning
```

```python
merged_df['Returned'].value_counts()
```

```
Returned
Yes    2220
Name: count, dtype: int64
```

```python
merged_df['Returned']=merged_df['Returned'].replace('Yes',True)
```

```python
merged_df['Returned'].value_counts()
```

```
Returned
True    2220
Name: count, dtype: int64
```

```python
merged_df['Returned'].fillna(False,inplace=True)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_13284\3219814076.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  merged_df['Returned'].fillna(False,inplace=True)
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_13284\3219814076.py:1:
FutureWarning: Downcasting object dtype arrays
on .fillna, .ffill, .bfill is deprecated and will change in a future
version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  merged_df['Returned'].fillna(False,inplace=True)
```

```python
merged_df['Returned'].value_counts()
```

```
Returned
False    49070
True      2220
Name: count, dtype: int64
```

```
region_return=merged_df.groupby('Region').agg(total_orders=('Order ID','count'),total_returns=('Returned','sum'))
region_return
```

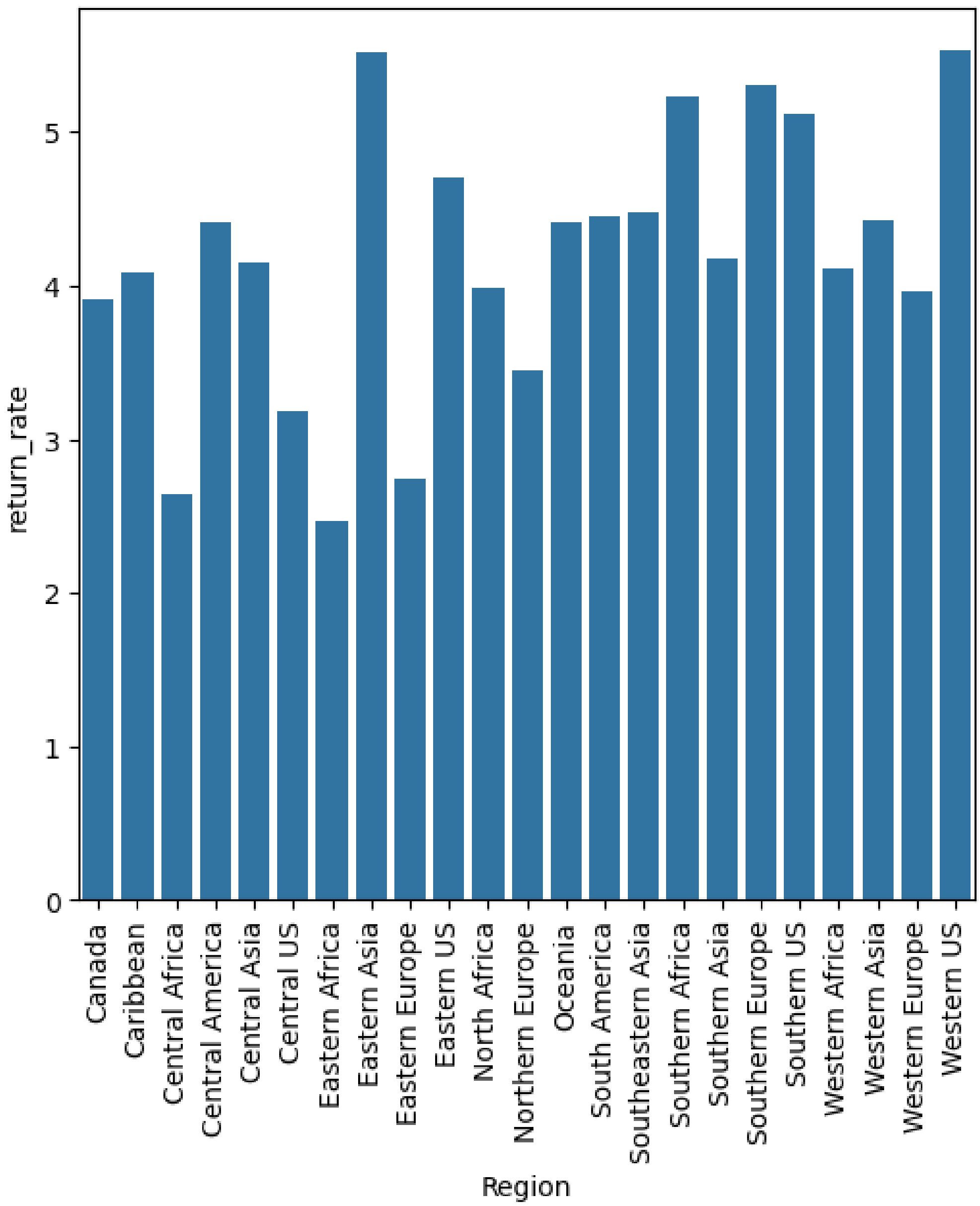
	total_orders	total_returns
Region		
Canada	384	15
Caribbean	1690	69
Central Africa	643	17
Central America	5616	248
Central Asia	217	9
Central US	2323	74
Eastern Africa	728	18
Eastern Asia	2374	131
Eastern Europe	1529	42
Eastern US	2848	134
North Africa	1278	51
Northern Europe	2204	76
Oceania	3487	154
South America	2988	133
Southeastern Asia	3129	140
Southern Africa	478	25
Southern Asia	2655	111
Southern Europe	2113	112
Southern US	1620	83
Western Africa	1460	60
Western Asia	2440	108
Western Europe	5883	233
Western US	3203	177

```
region_return['return_rate']=(region_return['total_returns']/region_return['total_orders'])*100
region_return
```

	total_orders	total_returns	return_rate
Region			
Canada	384	15	3.906250
Caribbean	1690	69	4.082840
Central Africa	643	17	2.643857
Central America	5616	248	4.415954
Central Asia	217	9	4.147465
Central US	2323	74	3.185536
Eastern Africa	728	18	2.472527
Eastern Asia	2374	131	5.518113
Eastern Europe	1529	42	2.746893
Eastern US	2848	134	4.705056
North Africa	1278	51	3.990610
Northern Europe	2204	76	3.448276
Oceania	3487	154	4.416404
South America	2988	133	4.451138

```
Southeastern Asia           3129            140     4.474273
Southern Africa              478             25     5.230126
Southern Asia               2655            111     4.180791
Southern Europe             2113            112     5.300521
Southern US                 1620             83     5.123457
Western Africa              1460             60     4.109589
Western Asia                2440            108     4.426230
Western Europe              5883            233     3.960564
Western US                  3203            177     5.526069

#Visualize data on the bases of region and return rate
plt.figure(figsize=(6,6))
sns.barplot(data=region_return,x='Region',y='return_rate')
plt.xticks(rotation=90)
plt.show()
```




```
#Question
#Q.1 What are the regions with the highest and lowest return rates?

region_return[(region_return['return_rate'] ==
region_return['return_rate'].min())]
```

	total_orders	total_returns	return_rate
Region			
Eastern Africa	728	18	2.472527

```
region_return[(region_return['return_rate'] ==
region_return['return_rate'].max())]
```

```
             total_orders  total_returns  return_rate
Region
Western US           3203            177     5.526069
```

```python
# Q2 2.    Are there specific product categories or sub-categories
associated with higher return rates?

def calculated_grouped_return_rate(col_name):
    Group_return=merged_df.groupby(col_name).agg(total_orders=('Order
ID','count'),total_returns=('Returned','sum'))

Group_return['return_rate']=(Group_return['total_returns']/Group_retur
n['total_orders'])*100
    return Group_return


product_return_rate=calculated_grouped_return_rate('Category')
product_return_rate
```

```
                 total_orders  total_returns  return_rate
Category
Furniture                9860            427     4.330629
Office Supplies         31289           1348     4.308223
Technology              10141            445     4.388127
```

```python
product_return_rate=calculated_grouped_return_rate('Sub-Category')
product_return_rate
```

```
              total_orders  total_returns  return_rate
Sub-Category
Accessories           3075            138     4.487805
Appliances            1742             59     3.386912
Art                   4864            217     4.461349
Binders               6146            269     4.376830
Bookcases             2411            104     4.313563
Chairs                3434            147     4.280722
Copiers               2223             99     4.453441
Envelopes             2387             99     4.147465
Fasteners             2601            102     3.921569
Furnishings           3154            135     4.280279
Labels                2601            137     5.267205
Machines              1486             63     4.239569
Paper                 3492            150     4.295533
Phones                3357            145     4.319333
Storage               5049            212     4.198851
Supplies              2407            103     4.279186
Tables                 861             41     4.761905
```

```python
#Q3 Is there a correlation between shipping mode and return rates?
```

```
ship_mode_return_rate=calculated_grouped_return_rate('Ship Mode')
ship_mode_return_rate.reset_index()
```

```
        Ship Mode  total_orders  total_returns  return_rate
0     First Class          7505            312     4.157229
1        Same Day          2701            120     4.442799
2    Second Class         10309            396     3.841304
3  Standard Class         30775           1392     4.523152
```

```
correlation=ship_mode_return_rate[['total_orders','total_returns']].co
rr()
correlation
```

```
               total_orders  total_returns
total_orders       1.000000       0.998487
total_returns      0.998487       1.000000
```

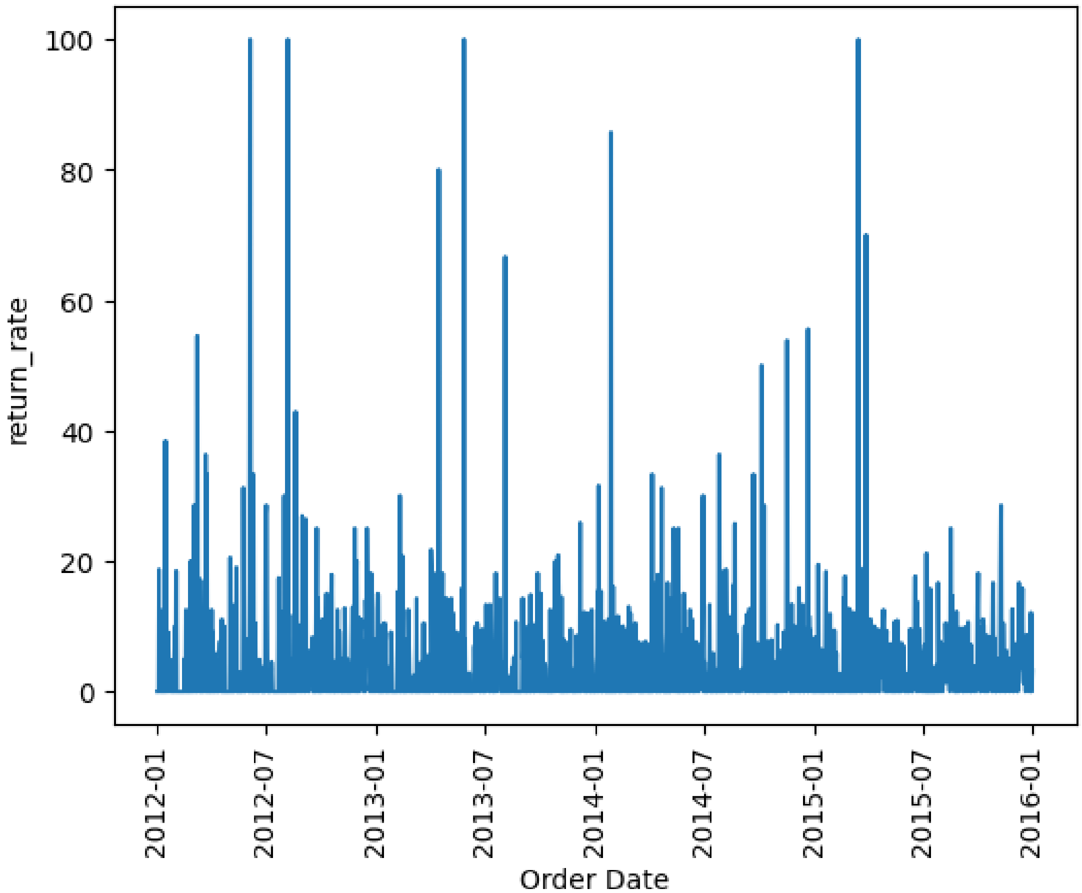
```
#Q4 How do return rates vary over time?
```

```
Order_date_return_rate=calculated_grouped_return_rate('Order Date')
Order_date_return_rate
```

```
            total_orders  total_returns  return_rate
Order Date
2012-01-01             6              0     0.000000
2012-01-02             1              0     0.000000
2012-01-03            20              0     0.000000
2012-01-04            16              3    18.750000
2012-01-05             7              0     0.000000
...                  ...            ...          ...
2015-12-27            54              0     0.000000
2015-12-28            13              0     0.000000
2015-12-29           116             14    12.068966
2015-12-30            79              0     0.000000
2015-12-31            62              2     3.225806

[1430 rows x 3 columns]
```

```
sns.lineplot(data=Order_date_return_rate,x='Order
Date',y='return_rate')
plt.xticks(rotation=90)
plt.show()
```

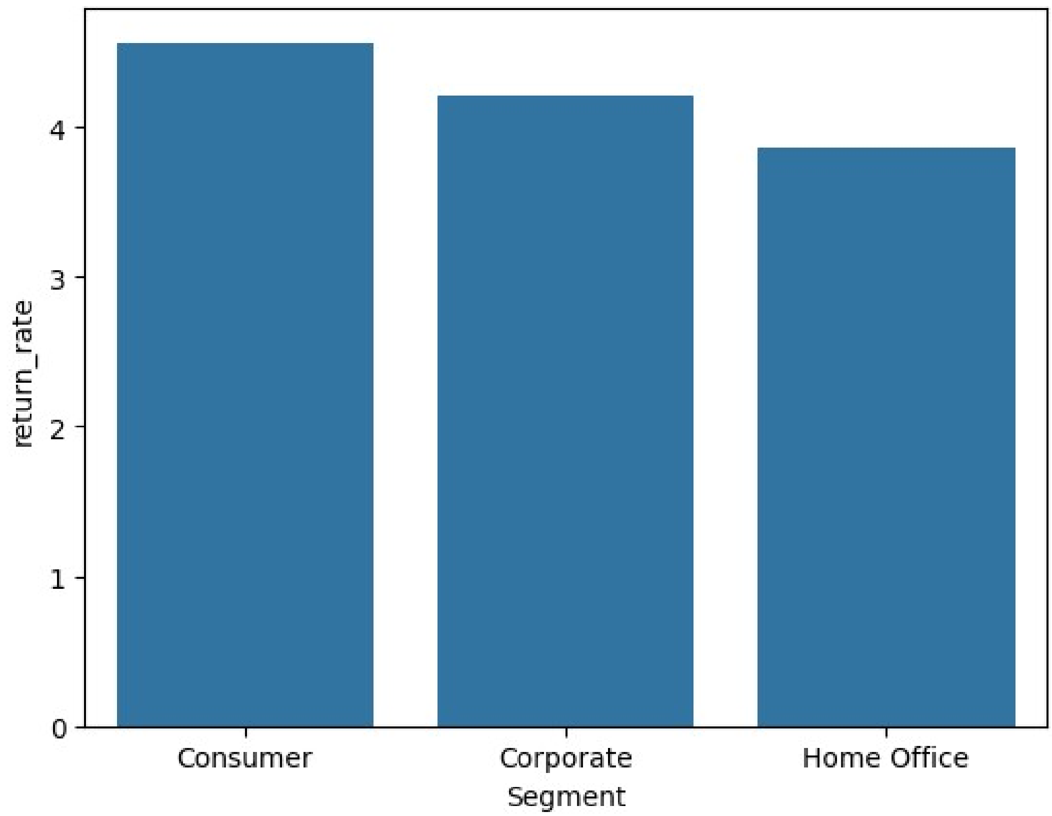



```
#Q5 Are there any correlations between customer segments (e.g.,
consumer, corporate) and return behavior?

segment_return_rate=calculated_grouped_return_rate('Segment')
segment_return_rate
```

Segment	total_orders	total_returns	return_rate
Consumer	26518	1210	4.562938
Corporate	15429	650	4.212846
Home Office	9343	360	3.853152

```
sns.barplot(data=segment_return_rate,x='Segment',y='return_rate')
plt.show()
```




```
#Q6 Do specific cities or states within regions show higher return rates?

city_return_rate=calculated_grouped_return_rate(['Region','State'])
city_return_rate
```

```
                                total_orders  total_returns  return_rate
Region     State
Canada     Alberta                        49              5    10.204082
           British Columbia               46              0     0.000000
           Manitoba                       14              0     0.000000
           Newfoundland                    1              0     0.000000
           Nova Scotia                     3              0     0.000000
...                                      ...            ...          ...
Western US New Mexico                     37              1     2.702703
           Oregon                        124              2     1.612903
           Utah                           53              3     5.660377
           Washington                    506             16     3.162055
           Wyoming                         1              0     0.000000

[1120 rows x 3 columns]
```

```
city_return_rate[(city_return_rate['return_rate']==city_return_rate['return_rate'].max())]
```

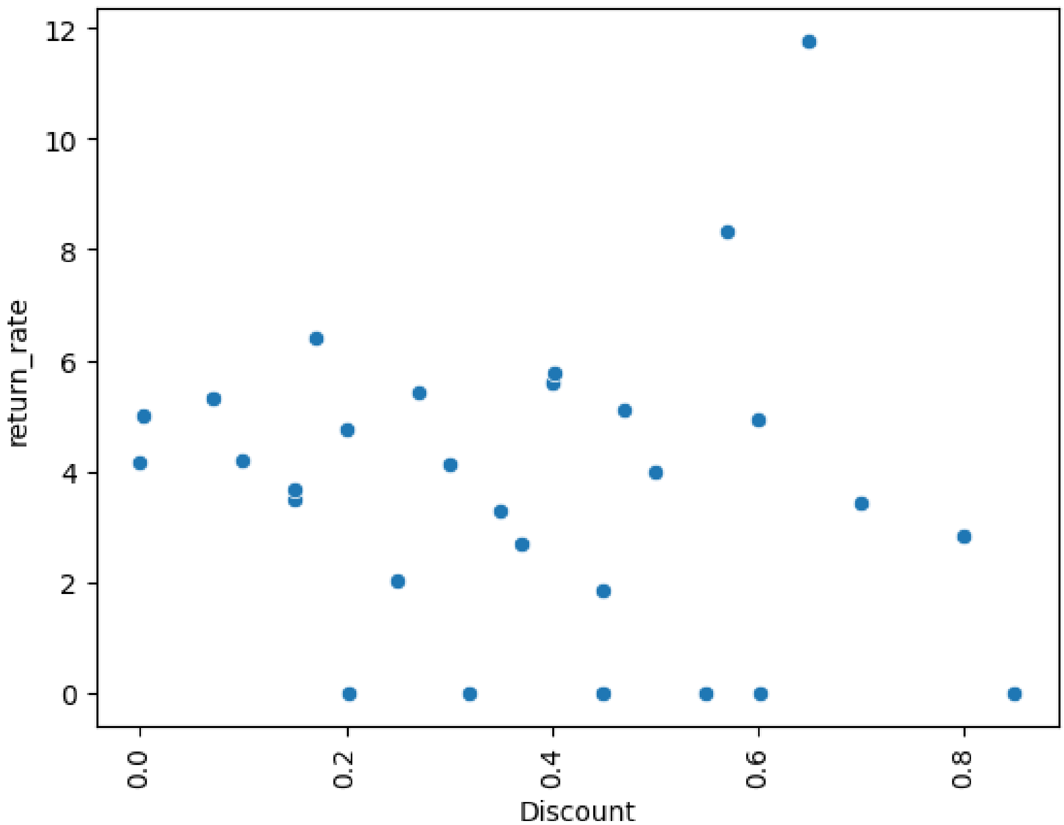
		total_orders	total_returns	return_rate
Region	State			
Central Africa	Litoral	3	3	100.0
Eastern Europe	Yambol	1	1	100.0
North Africa	Tlemcen	2	2	100.0
Southern Europe	Elbasan	1	1	100.0
	Korçë	3	3	100.0
	Ljubljana	3	3	100.0

```
#Q7 What is the impact of discounts on return rates?

discount_return_rate=calculated_grouped_return_rate('Discount')
discount_return_rate.sort_values(by='return_rate',ascending=False)
```

	total_orders	total_returns	return_rate
Discount			
0.650	17	2	11.764706
0.570	12	1	8.333333
0.170	735	47	6.394558
0.402	104	6	5.769231
0.400	3177	178	5.602770
0.270	388	21	5.412371
0.070	150	8	5.333333
0.470	725	37	5.103448
0.002	461	23	4.989154
0.600	2006	99	4.935194
0.200	4998	238	4.761905
0.100	4068	171	4.203540
0.000	29009	1205	4.153883
0.300	340	14	4.117647
0.500	1633	65	3.980404
0.150	82	3	3.658537
0.150	459	16	3.485839
0.700	1786	61	3.415454
0.350	122	4	3.278689
0.800	316	9	2.848101
0.370	74	2	2.702703
0.250	198	4	2.020202
0.450	325	6	1.846154
0.550	10	0	0.000000
0.450	2	0	0.000000
0.602	23	0	0.000000
0.320	27	0	0.000000
0.202	41	0	0.000000
0.850	2	0	0.000000

```
sns.scatterplot(data=discount_return_rate,x='Discount',y='return_rate'
)
plt.xticks(rotation=90)
plt.show()
```




```
#Q8 How do return rates differ between new customers and repeat
customers?

repeat=merged_df['Customer Name'].value_counts().reset_index()
repeat.head()

     Customer Name  count
0  Muhammed Yedwab    108
1      Steven Ward    106
2      Bill Eplett    102
3       Gary Hwang    102
4  Patrick O'Brill    102

#Q9 Is there a relationship between shipping cost and return rates?

shipcost_return_rate=calculated_grouped_return_rate('Shipping
Cost').reset_index()
shipcost_return_rate

       Shipping Cost  total_orders  total_returns  return_rate
0              1.002             1              0          0.0
1              1.003             1              0          0.0
2              1.010             6              0          0.0
3              1.019             1              0          0.0
4              1.020             6              0          0.0
```

```
...              ...            ...            ...          ...
16748        903.040              1              0          0.0
16749        910.160              1              0          0.0
16750        915.490              1              0          0.0
16751        923.630              1              0          0.0
16752        933.570              1              0          0.0

[16753 rows x 4 columns]
```

```
corr=shipcost_return_rate[['Shipping Cost','return_rate']].corr()
corr
```

```
               Shipping Cost  return_rate
Shipping Cost       1.000000     0.011215
return_rate         0.011215     1.000000
```