Part 0: Critical Thinking

Unsupported Cell Type. Double-Click to inspect/edit the content.

Part 1: Descriptive Analysis

```
# importing Libraries

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
pd.set_option('display.max_rows', None)


# Lodaing the dataset from the given excel sheet i.e Funnel

df = pd.read_excel('assignment.xlsx', sheet_name='Funnel')


df.head() # to display the top five rows
```

	Year	Month	Segment	Region	KPI	Value Type	Value
0	2020	12	Clients	India	Lv1_Visitors	Actuals	3665558
1	2020	12	Clients	India	Lv2_Visitors	Actuals	2689569
2	2020	12	Clients	India	Lv3_Visitors	Actuals	1300571
3	2020	12	Clients	India	Lv4_Visitors	Actuals	717608
4	2020	12	Clients	India	Lv3_Visitors	Actuals	706677

The data is about visitors visited to a company they may be form differnet regions and they can be coustomers or clients.

- The Data belongs to FITTLYF
- The Data set contains columns of year month region KPI ValueType Value
- year means it gives the toatl number of year the visitors have visited the company
- month gives monthly how many times they arrived
- Segment means weather they are coustomers or clients
- Region which region they are they belongs to
- KPI means type of visitors based on the levels

- valueType they visited website or some other
- value is total number visitors

This Dataset has no patterns since all the columns are independent .

```
# to find the rows and columns in the data

df.shape
```

```
(1572, 7)
```

```
#  to disply the information about the data i.e data types and total values in dat a

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1572 entries, 0 to 1571
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Year        1572 non-null   int64 
 1   Month       1572 non-null   int64 
 2   Segment     1572 non-null   object
 3   Region      1572 non-null   object
 4   KPI         1572 non-null   object
 5   Value Type  1572 non-null   object
 6   Value       1572 non-null   int64 
dtypes: int64(3), object(4)
memory usage: 86.1+ KB
```

```
# to find the null values in the data

df.isnull().sum()
```

```
Year          0
Month         0
Segment       0
Region        0
KPI           0
Value Type    0
Value         0
dtype: int64
```

To find the values counts to find the unique values and count of values

```
df['Year'].value_counts()
```

```
2022    660
2020    456
2021    456
Name: Year, dtype: int64
```

```
df['Month'].value_counts()
```

```
12    131
11    131
10    131
9     131
8     131
7     131
6     131
5     131
4     131
3     131
2     131
1     131
Name: Month, dtype: int64
```

```
df['Month'].unique()
```

```
array([12, 11, 10,  9,  8,  7,  6,  5,  4,  3,  2,  1])
```

```
df['Segment'].value_counts()
```

```
Customers    1080
Clients       492
Name: Segment, dtype: int64
```

```
df['Region'].value_counts()
```

```
India         432
Uddepy        240
Dehradun      240
Ujjain        240
Faridabad     180
Aurangabad    180
Indore         60
Name: Region, dtype: int64
```

```
df['KPI'].value_counts()
```

```
Lv3_Visitors    324
Lv4_Visitors    324
Lv5_Visitors    324
Lv1_Visitors    300
Lv2_Visitors    300
Name: KPI, dtype: int64
```

```
df['Value Type'].value_counts()
```

```
Actuals    1572
Name: Value Type, dtype: int64
```

```
df['Value'].value_counts()
```

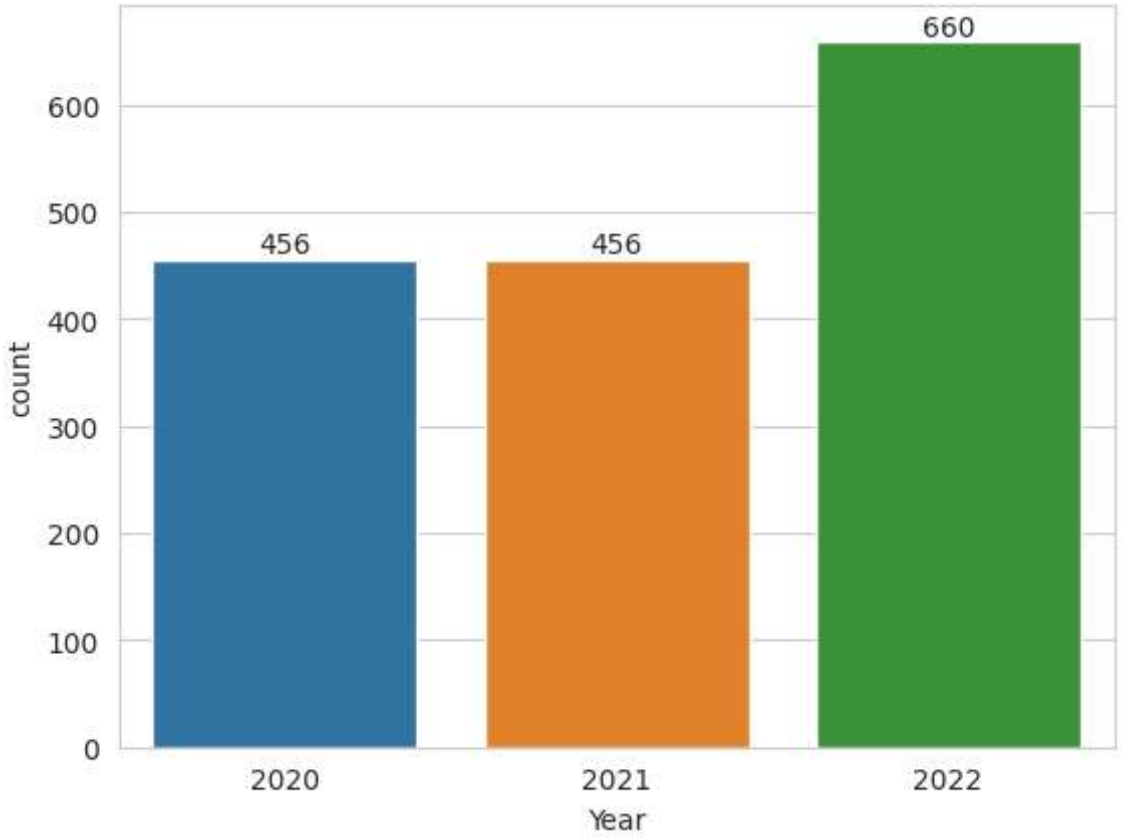
```
4085        1
4625        1
4815        1
6207        1
6821        1
8196        1
11600       1
16241       1
14781       1
15050       1
18591       1
56614       1
21519       1
22713       1
23730       1
28265       1
29359       1
32552       1
32823       1
34477       1
38057       1
70151       1
15260       1
81416       1
95001       1
222199      1
466208      1
512155      1
584005      1
18368       1
19556       1
34695       1
20744       1
12201       1
19538       1
232479      1
20867       1
22751       1
24103       1
24376       1
34848       1
40317       1
47155       1
71616       1
88949       1
330165      1
9334        1
469431      1
897         1
2250        1
3201        1
3616        1
4399        1
6103        1
6178        1
8489        1
527         1
Name: Value, dtype: int64
```

```
df['Value'].nunique()
```

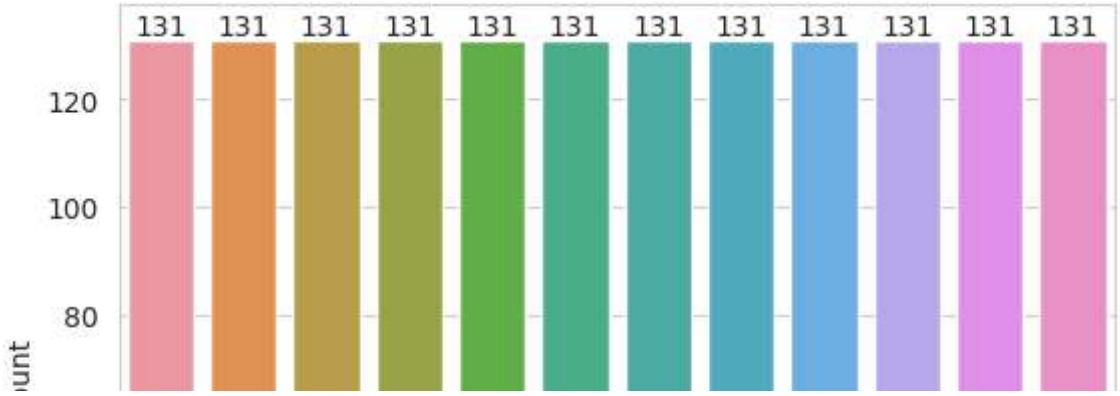
```
1558
```

Univerate analysis to find the value counts using plots

```
ax = sns.countplot(x=df["Year"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



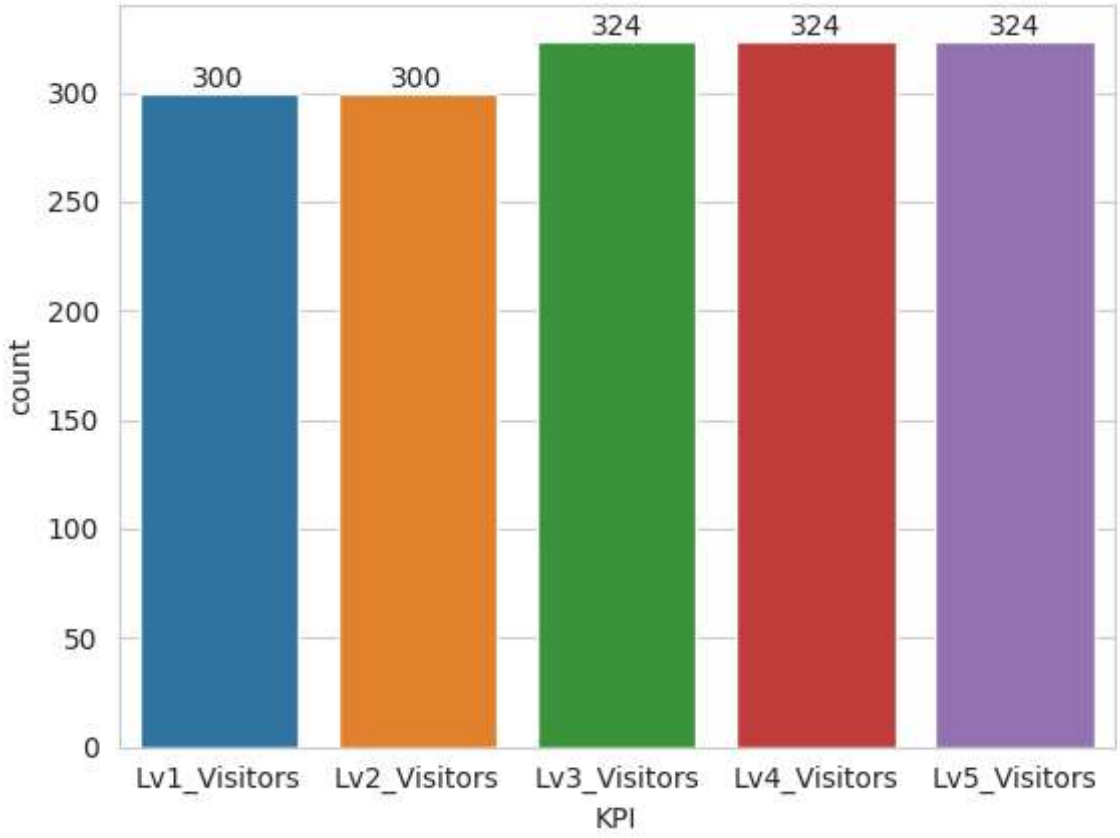

```
ax = sns.countplot(x=df["Month"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



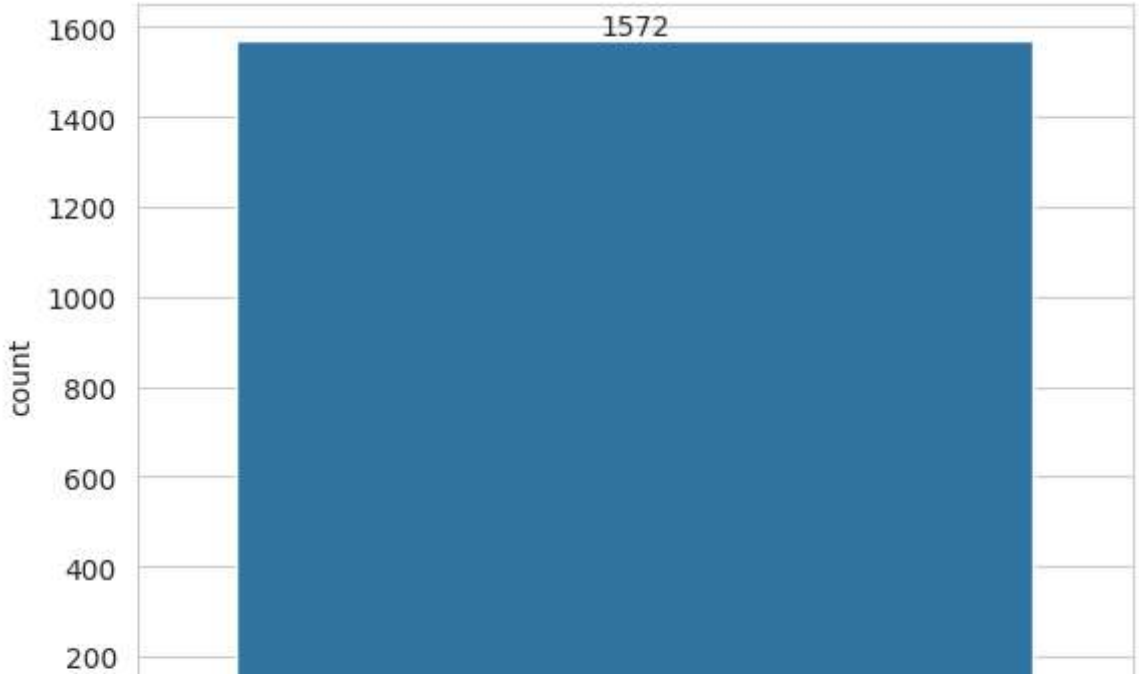

```
ax = sns.countplot(x=df["Segment"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```


```
ax = sns.countplot(x=df["Region"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```

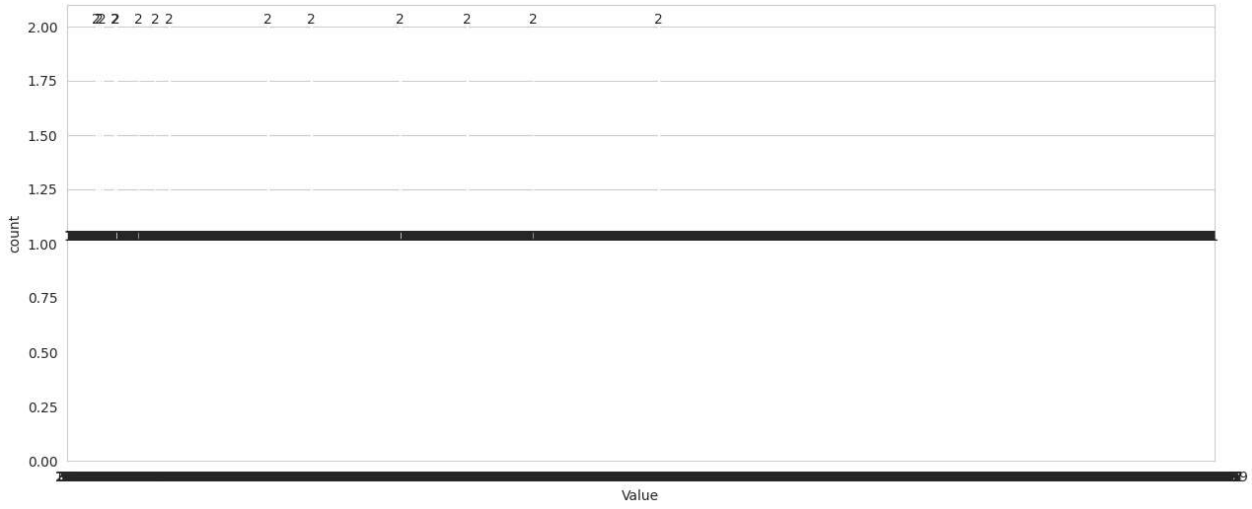

```
ax = sns.countplot(x=df["KPI"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```




```
ax = sns.countplot(x=df["Value Type"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```




```
plt.figure(figsize=(15, 6))
ax = sns.countplot(x=df["Value"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```




```
df.head()
```

	Year	Month	Segment	Region	KPI	Value Type	Value
0	2020	12	Clients	India	Lv1_Visitors	Actuals	3665558
1	2020	12	Clients	India	Lv2_Visitors	Actuals	2689569
3	2020	12	Clients	India	Lv4_Visitors	Actuals	717608

The total number of visitors segmented by each level, every month in each year

```
table = pd.pivot_table(df,index=['Segment','Month','Year'],aggfunc={'Value':np.sum})
table
```

	2021	2832286
	2022	1461679
3	**2020**	2333410
	2021	3250596
	2022	1843820
4	**2020**	3098412
	2021	2589415
	2022	1515534
5	**2020**	3273228
	2021	2408595
	2022	1542012
6	**2020**	2994226
	2021	2185250
	2022	1521724
7	**2020**	2635507
	2021	2101509
	2022	1459655
8	**2020**	2294441
	2021	1820148
	2022	1455818
9	**2020**	2536696
	2021	1690797
	2022	1380606
10	**2020**	3342355
	2021	1528523
	2022	1430120
11	**2020**	2817300
	2021	1438409
	2022	1533911
12	**2020**	2449978
	2021	1528252
	2022	1374120

```
table = pd.pivot_table(df,index=['Region','Year'],aggfunc={'Value':np.sum})
table
```

Region	Year	Value
Aurangabad	2020	285497
	2021	300058
	2022	196448
Dehradun	2020	1172070
	2021	930998
	2022	8212445
Faridabad	2020	1170281
	2021	1139358
	2022	921404
India	2020	183457306
	2021	116142601
	2022	110865565
Indore	2022	3543866
Uddepy	2020	4376357
	2021	3130906
	2022	22307017
Ujjain	2020	1176987
	2021	920026
	2022	5743265

percentage difference in the number of visitors between different regions and years

```
table['% of Year'] = (table.Value / table.groupby(level=0).Value.transform(sum) * 100).ast
```

```
table
```

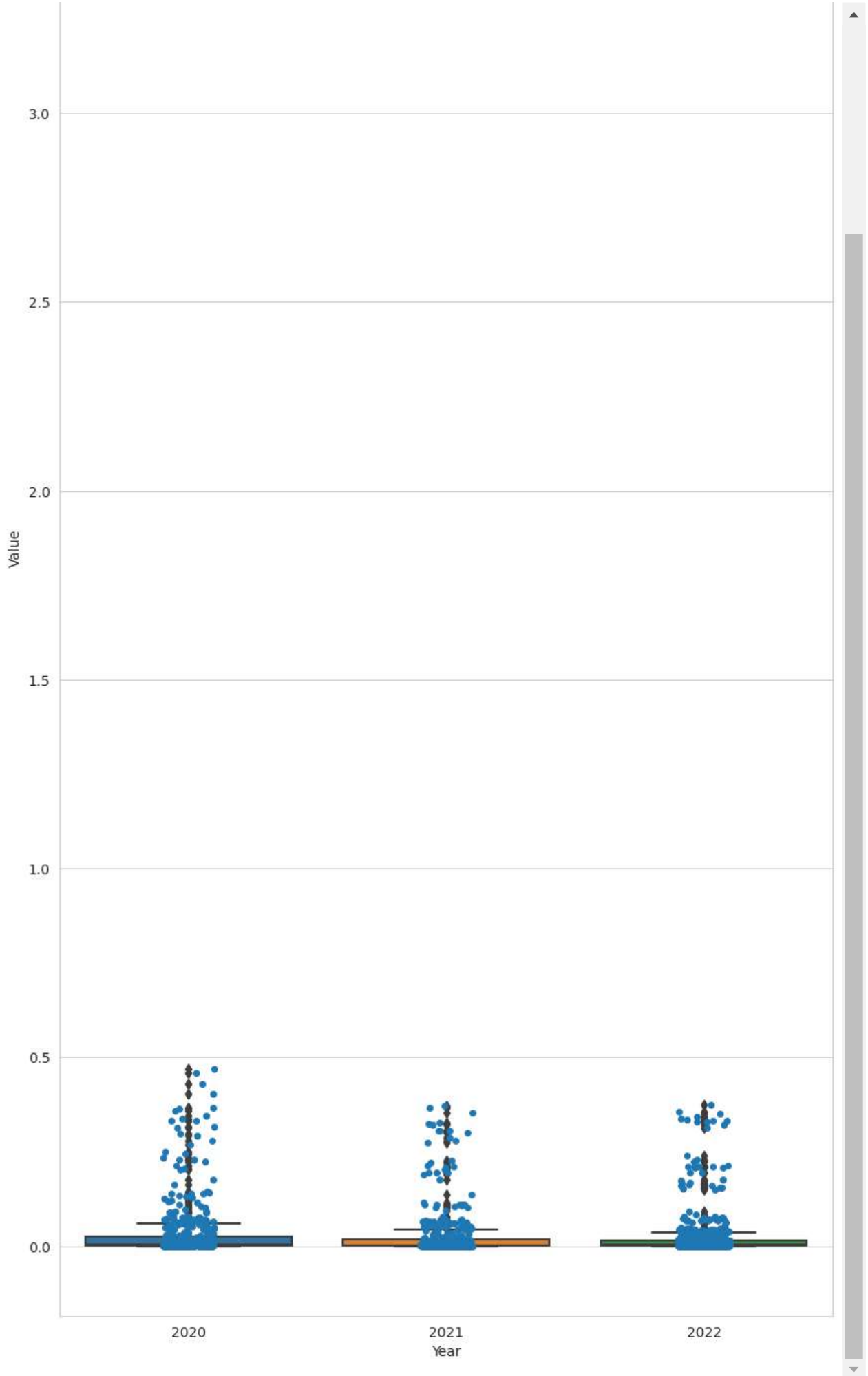
Region	Year	Value	% of Year
Aurangabad	**2020**	285497	36.50842771702922%
	2021	300058	38.370441034113675%
	2022	196448	25.121131248857104%
Dehradun	**2020**	1172070	11.362207579981723%
	2021	930998	9.02522249741724%
	2022	8212445	79.61256992260104%
Faridabad	**2020**	1170281	36.21991412680054%
	2021	1139358	35.26285474999868%
	2022	921404	28.517231123200776%
India	**2020**	183457306	44.69494233122732%
	2021	116142601	28.295340028015804%
	2022	110865565	27.009717640756882%
Indore	**2022**	3543866	100.0%
Uddepy	**2020**	4376357	14.678727777427461%
	2021	3130906	10.501363776016056%
	2022	22307017	74.81990844655648%

finding the outliers using boxplot

	2021	920026	11.7346094105336 58%

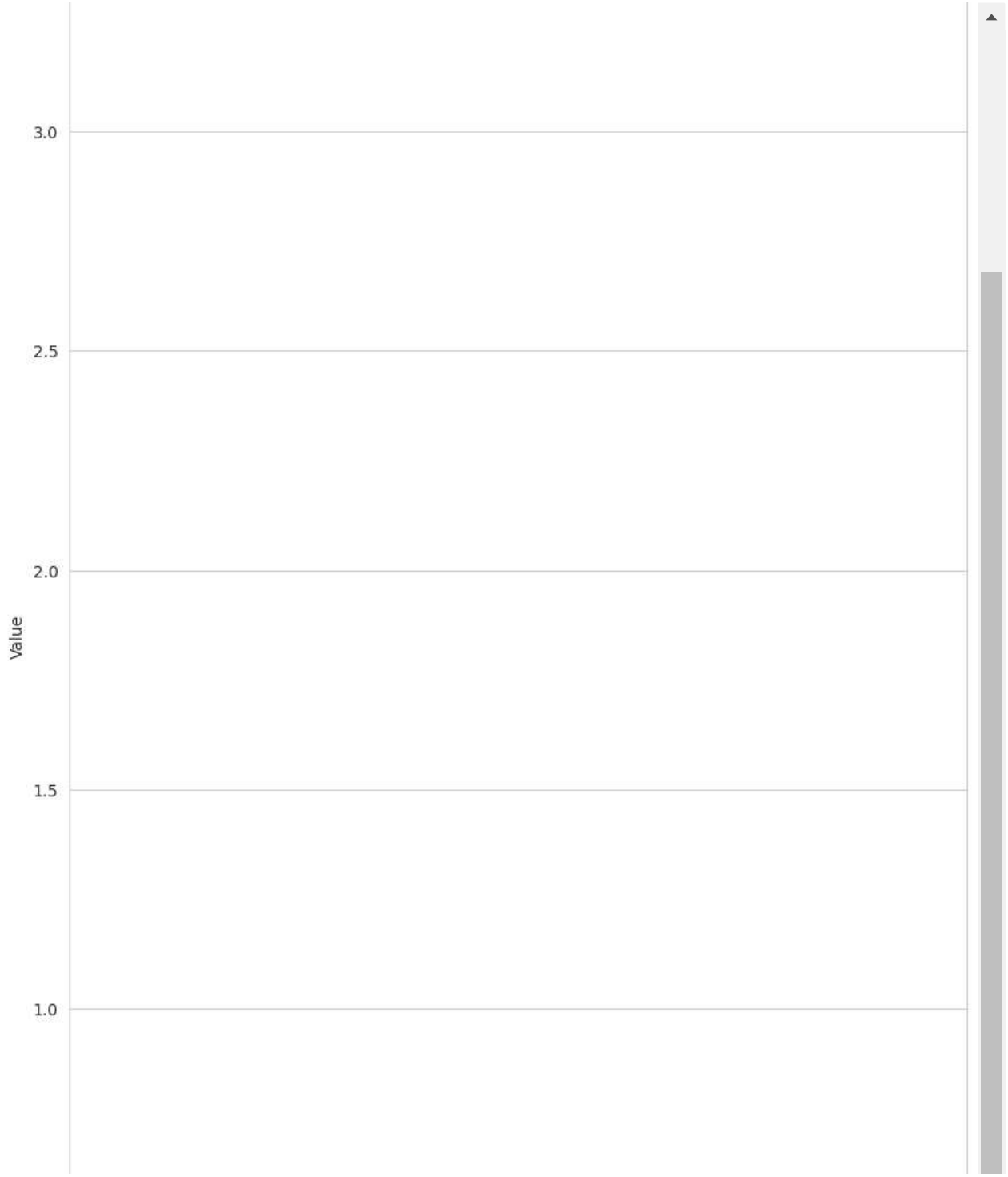
```
fig, ax = plt.subplots(figsize=(10, 20))

sns.set_style('whitegrid')
ax= sns.boxplot(x='Year',y='Value',data=df)
ax = sns.stripplot(x='Year',y='Value',data=df)
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Value
2020
2021
2022
Year

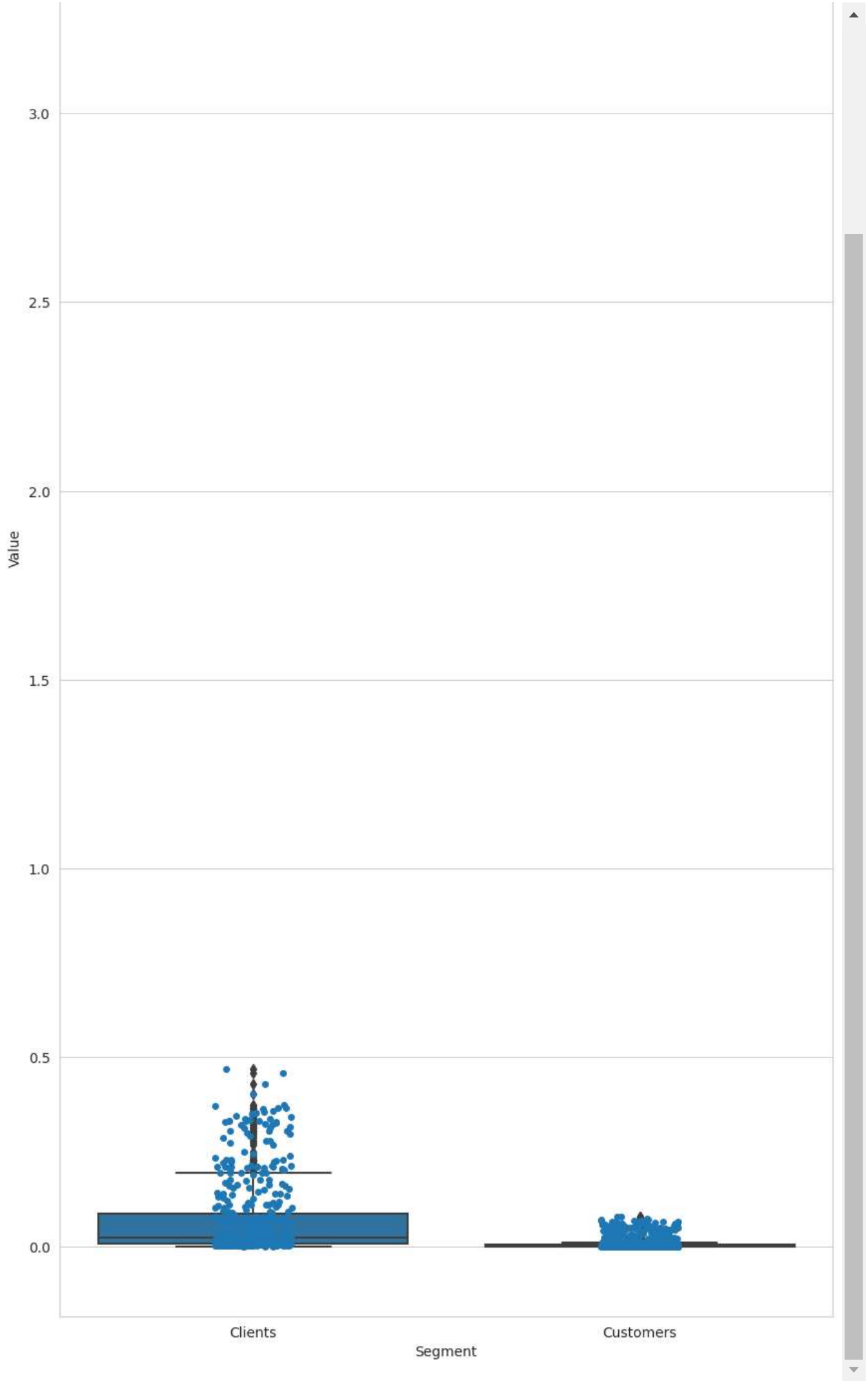
```
fig, ax = plt.subplots(figsize=(10, 20))

sns.set_style('whitegrid')
ax= sns.boxplot(x='Month',y='Value',data=df)
ax = sns.stripplot(x='Month',y='Value',data=df)
```



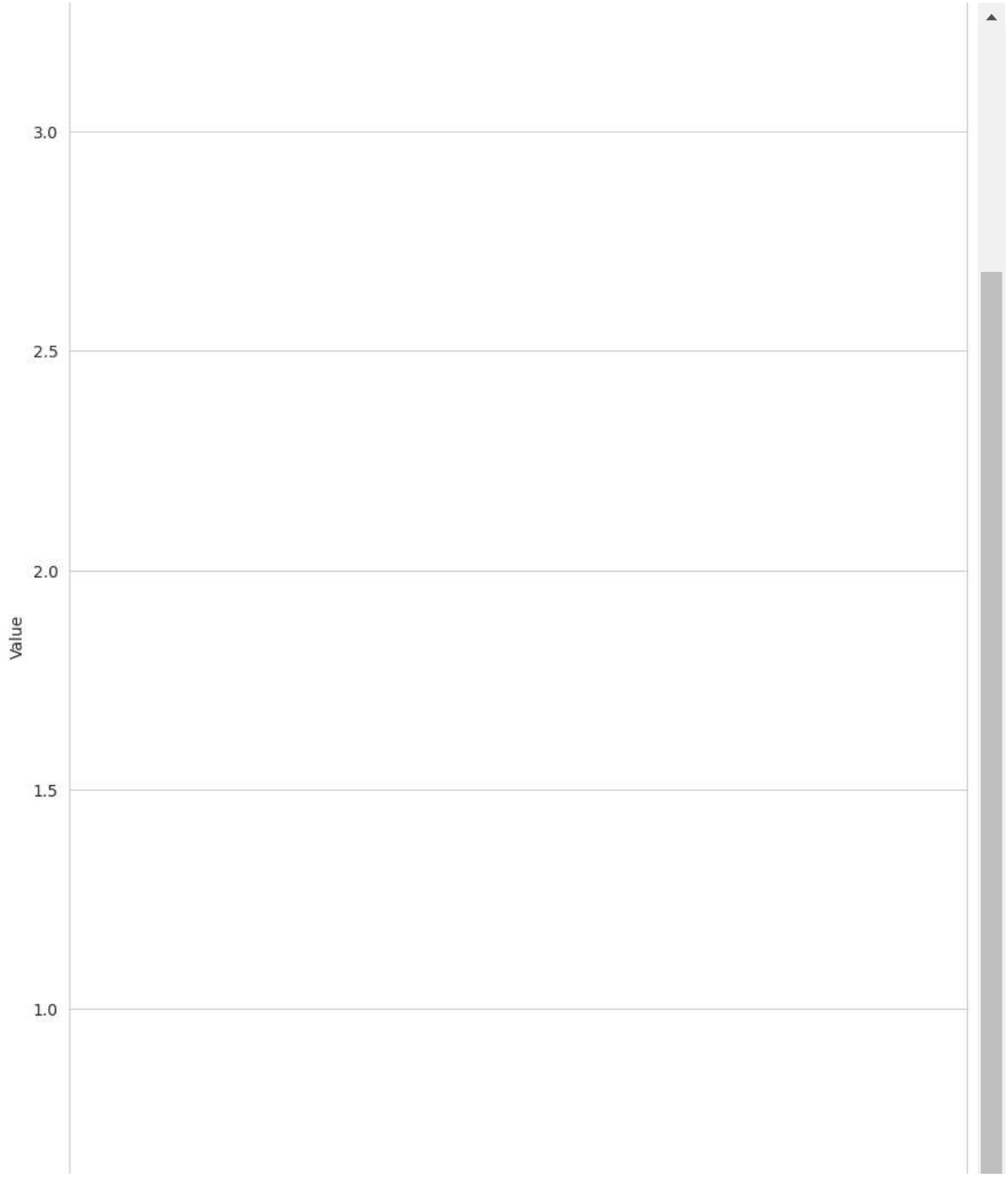

```
fig, ax = plt.subplots(figsize=(10, 20))

sns.set_style('whitegrid')
ax= sns.boxplot(x='Segment',y='Value',data=df)
ax = sns.stripplot(x='Segment',y='Value',data=df)
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Value
Clients
Customers
Segment

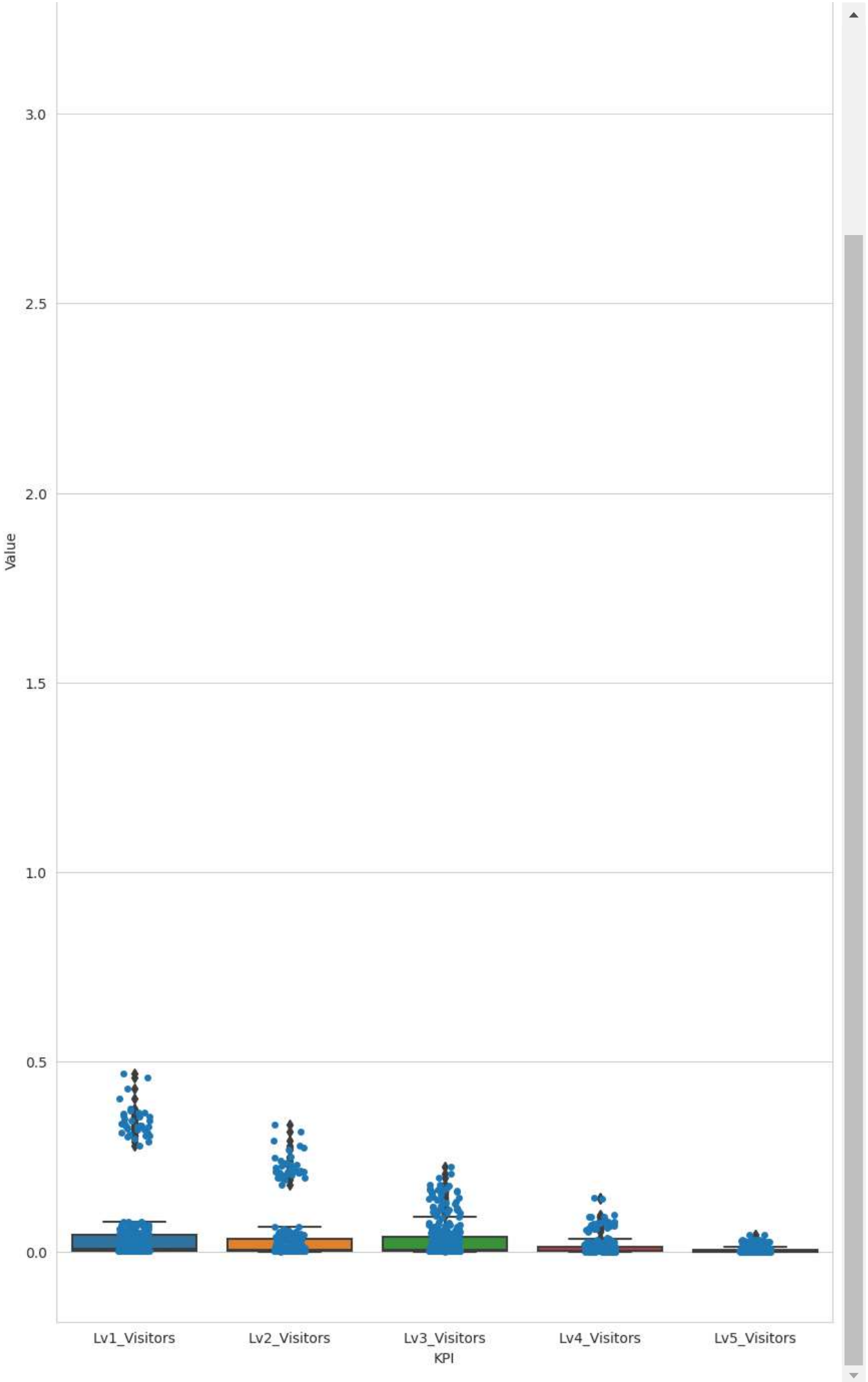
```
fig, ax = plt.subplots(figsize=(10, 20))

sns.set_style('whitegrid')
ax= sns.boxplot(x='Region',y='Value',data=df)
ax = sns.stripplot(x='Region',y='Value',data=df)
```



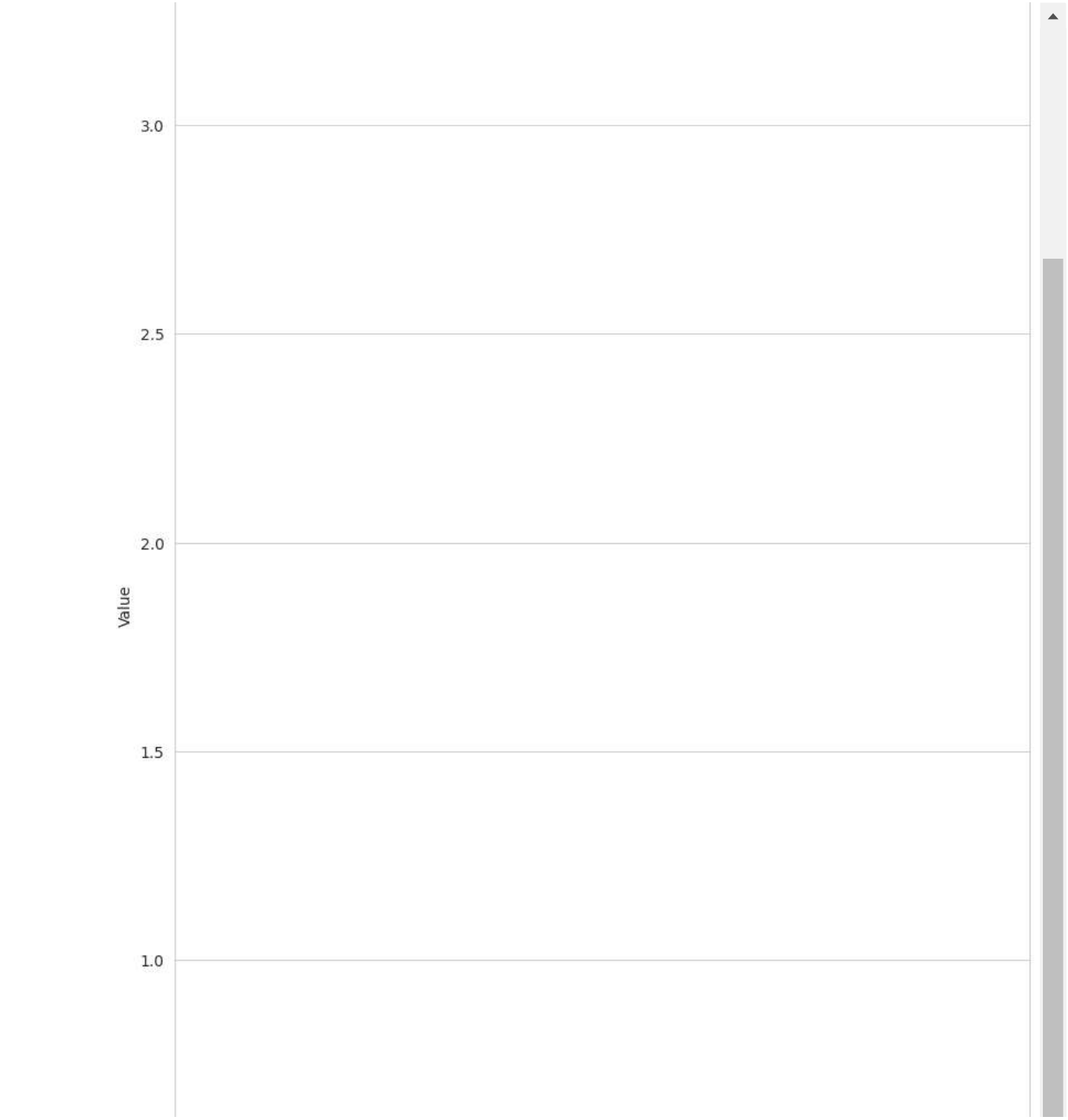

```
fig, ax = plt.subplots(figsize=(10, 20))

sns.set_style('whitegrid')
ax= sns.boxplot(x='KPI',y='Value',data=df)
ax = sns.stripplot(x='KPI',y='Value',data=df)
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Value
Lv1_Visitors
Lv2_Visitors
Lv3_Visitors
Lv4_Visitors
Lv5_Visitors
KPI

```
fig, ax = plt.subplots(figsize=(10, 20))

sns.set_style('whitegrid')
ax= sns.boxplot(x='Value Type',y='Value',data=df)
ax = sns.stripplot(x='Value Type',y='Value',data=df)
```



Removing the outliers form the data

```
def outlier_treatment(datacolumn):
    sorted(datacolumn)
    Q1,Q3 = np.percentile(datacolumn , [25,75])
    IQR = Q3 - Q1
    lower_range = Q1 - (1.5 * IQR)
    upper_range = Q3 + (1.5 * IQR)
    return lower_range,upper_range
```

```
lowerbound,upperbound = outlier_treatment(df.Value)
df[(df.Value < lowerbound) | (df.Value > upperbound)]
```

1063	2021	6	Customers	India	Lv3_Visitors	Actuals	469601
1064	2021	6	Customers	India	Lv2_Visitors	Actuals	449360
1092	2022	6	Customers	India	Lv1_Visitors	Actuals	506023
1122	2020	5	Customers	India	Lv3_Visitors	Actuals	714014
1123	2020	5	Customers	India	Lv1_Visitors	Actuals	710544
1124	2020	5	Customers	India	Lv2_Visitors	Actuals	590422
1152	2021	5	Customers	India	Lv1_Visitors	Actuals	608351
1153	2021	5	Customers	India	Lv3_Visitors	Actuals	513130
1154	2021	5	Customers	India	Lv2_Visitors	Actuals	492346
1182	2022	5	Customers	India	Lv1_Visitors	Actuals	504390
1212	2020	4	Customers	India	Lv1_Visitors	Actuals	689114
1213	2020	4	Customers	India	Lv3_Visitors	Actuals	643474
1214	2020	4	Customers	India	Lv2_Visitors	Actuals	554469
1242	2021	4	Customers	India	Lv1_Visitors	Actuals	642725
1243	2021	4	Customers	India	Lv3_Visitors	Actuals	530740
1244	2021	4	Customers	India	Lv2_Visitors	Actuals	520986
1272	2022	4	Customers	India	Lv1_Visitors	Actuals	472128
1302	2020	3	Customers	India	Lv1_Visitors	Actuals	518074
1303	2020	3	Customers	India	Lv3_Visitors	Actuals	506752
1332	2021	3	Customers	India	Lv1_Visitors	Actuals	785903
1333	2021	3	Customers	India	Lv2_Visitors	Actuals	648832
1334	2021	3	Customers	India	Lv3_Visitors	Actuals	637409
1362	2022	3	Customers	India	Lv1_Visitors	Actuals	578083
1392	2020	2	Customers	India	Lv1_Visitors	Actuals	461165
1393	2020	2	Customers	India	Lv3_Visitors	Actuals	437011
1422	2021	2	Customers	India	Lv1_Visitors	Actuals	675767
1423	2021	2	Customers	India	Lv3_Visitors	Actuals	552723
1424	2021	2	Customers	India	Lv2_Visitors	Actuals	543885
1512	2021	1	Customers	India	Lv1_Visitors	Actuals	664367
1513	2021	1	Customers	India	Lv3_Visitors	Actuals	558042
1514	2021	1	Customers	India	Lv2_Visitors	Actuals	521898
1542	2022	1	Customers	India	Lv1_Visitors	Actuals	447793

```
df.drop(df[ (df.Value > upperbound) | (df.Value < lowerbound) ].index , inplace=True)
```

```
df.shape
```

```
(1325, 7)
```

```
lowerbound,upperbound = outlier_treatment(df.Year)
df[(df.Year < lowerbound) | (df.Year > upperbound)]
df.drop(df[ (df.Year > upperbound) | (df.Year < lowerbound) ].index , inplace=True)
```

```
df.shape
```

```
(1325, 7)
```

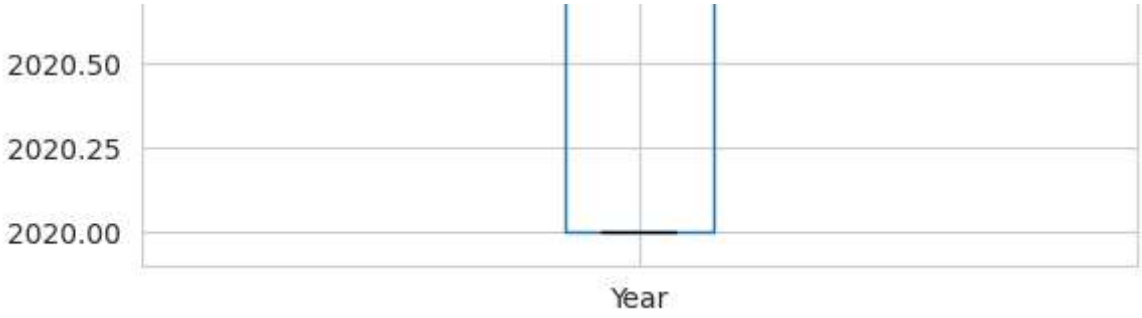
```
lowerbound,upperbound = outlier_treatment(df.Month)
df[(df.Month < lowerbound) | (df.Month > upperbound)]
df.drop(df[ (df.Month > upperbound) | (df.Month < lowerbound) ].index , inplace=True)
```

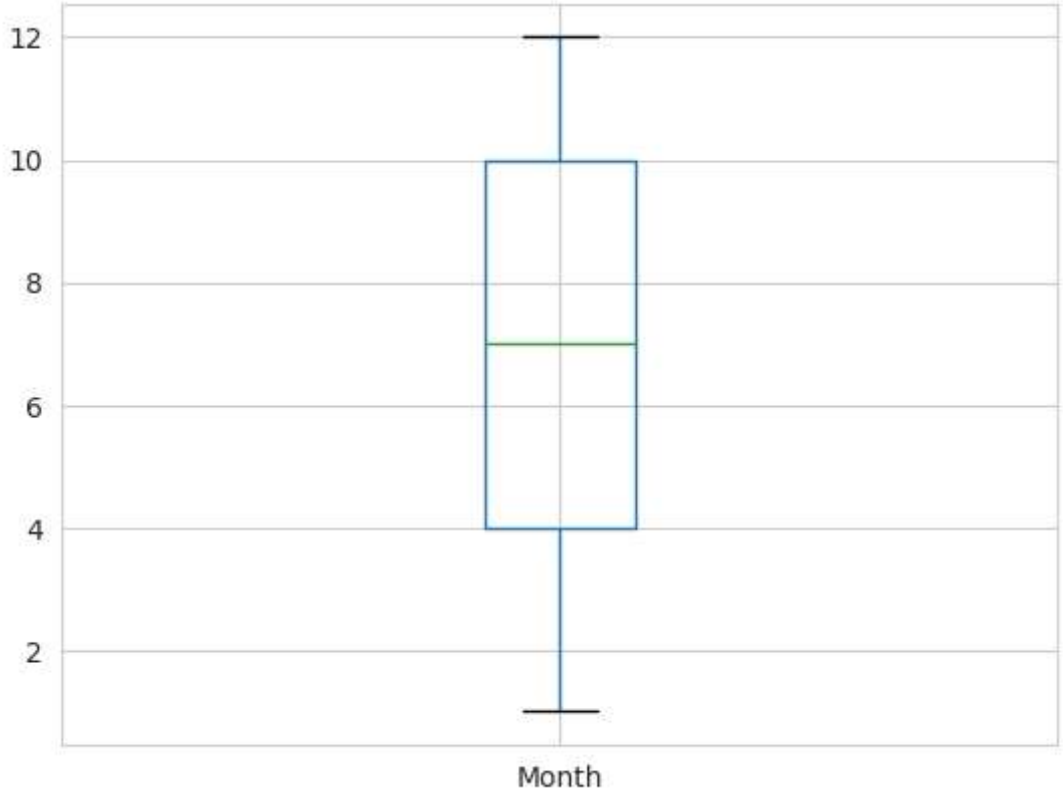
```
df.shape
```

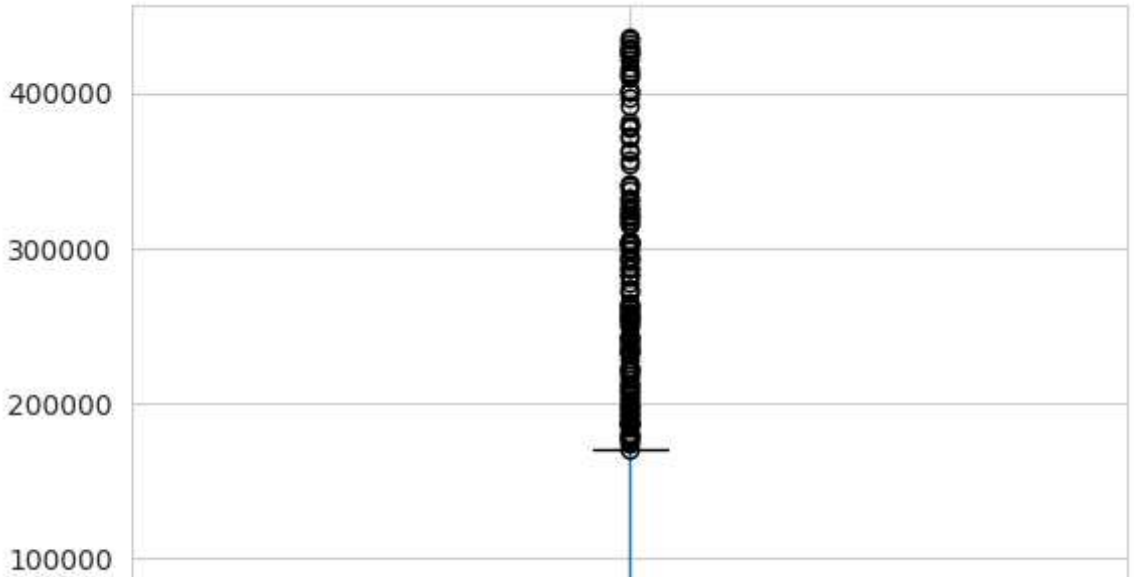
```
(1325, 7)
```

```
df_numerical_features = df.select_dtypes(exclude='object')
df_categorical_features = df.select_dtypes(include='object')
```

```
for column in df_numerical_features:
    plt.figure()
    df.boxplot([column])
```



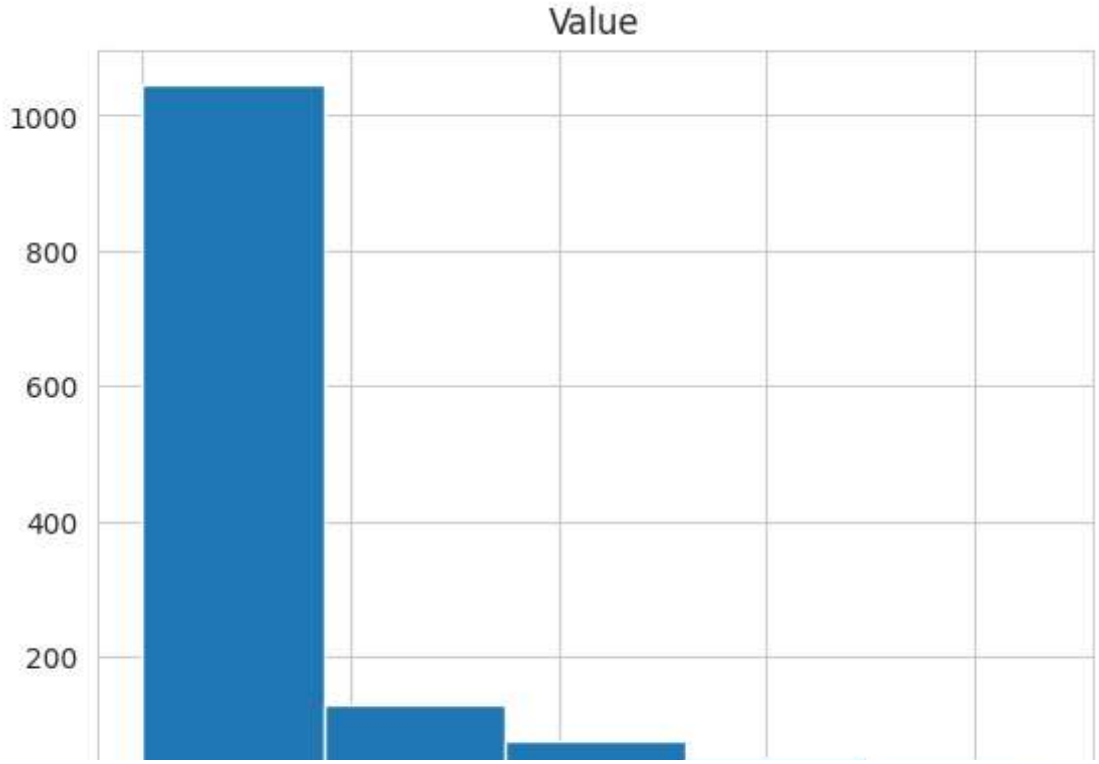




Histogram to find data disturbution

```
df.hist(column = 'Value', bins = 5)
```

```
array([[<Axes: title={'center': 'Value'}>]], dtype=object)
```



Part 2: Prescriptive Analysis

Transpose of data as given in problem statement

```
data = df[['Value', "KPI"]]
```

```
data
```

1539	5150	Lv5_Visitors
1540	3239	Lv4_Visitors
1541	1340	Lv5_Visitors
1543	299158	Lv2_Visitors
1544	235941	Lv3_Visitors
1545	105203	Lv1_Visitors
1546	81890	Lv4_Visitors
1547	57375	Lv2_Visitors
1548	47604	Lv3_Visitors
1549	30004	Lv1_Visitors
1550	27941	Lv1_Visitors
1551	22849	Lv4_Visitors
1552	19861	Lv5_Visitors
1553	19673	Lv2_Visitors
1554	19489	Lv1_Visitors
1555	16853	Lv2_Visitors
1556	13167	Lv2_Visitors
1557	11438	Lv3_Visitors
1558	9529	Lv3_Visitors
1559	9447	Lv3_Visitors
1560	7289	Lv1_Visitors
1561	7281	Lv5_Visitors
1562	4661	Lv2_Visitors
1563	4168	Lv4_Visitors
1564	3831	Lv4_Visitors
1565	3786	Lv4_Visitors
1566	3702	Lv3_Visitors
1567	1693	Lv5_Visitors
1568	1428	Lv4_Visitors
1569	1311	Lv5_Visitors
1570	1071	Lv5_Visitors
1571	527	Lv5_Visitors

```
df['Lv5_Visitors'] = data.query("KPI=='Lv5_Visitors'")["Value"]
```

```
df.head()
```

	Year	Month	Segment	Region	KPI	Value Type	Value	Lv5_Visitors
5	2020	12	Clients	India	Lv5_Visitors	Actuals	255723	255723.0
6	2020	12	Clients	India	Lv4_Visitors	Actuals	180125	NaN
7	2020	12	Clients	India	Lv5_Visitors	Actuals	74768	74768.0
13	2021	12	Clients	India	Lv5_Visitors	Actuals	216178	216178.0
14	2021	12	Clients	India	Lv4_Visitors	Actuals	180821	NaN

```
df['Lv4_Visitors'] = data.query("KPI=='Lv4_Visitors'")["Value"]
```

```
df['Lv3_Visitors'] = data.query("KPI=='Lv3_Visitors'")["Value"]
```

```
df['Lv2_Visitors'] = data.query("KPI=='Lv2_Visitors'")["Value"]
```

```
df['Lv1_Visitors'] = data.query("KPI=='Lv1_Visitors'")["Value"]
```

```
df.head()
```

	Year	Month	Segment	Region	KPI	Value Type	Value	Lv5_Visitors	Lv4_Vis
5	2020	12	Clients	India	Lv5_Visitors	Actuals	255723	255723.0	
6	2020	12	Clients	India	Lv4_Visitors	Actuals	180125	NaN	18
7	2020	12	Clients	India	Lv5_Visitors	Actuals	74768	74768.0	
13	2021	12	Clients	India	Lv5_Visitors	Actuals	216178	216178.0	
14	2021	12	Clients	India	Lv4_Visitors	Actuals	180821	NaN	18

```
df.reset_index(inplace = True)
```

```
df.head()
```

	index	Year	Month	Segment	Region	KPI	Value Type	Value	Lv5_Visitors	l
0	5	2020	12	Clients	India	Lv5_Visitors	Actuals	255723	255723.0	
1	6	2020	12	Clients	India	Lv4_Visitors	Actuals	180125	NaN	
2	7	2020	12	Clients	India	Lv5_Visitors	Actuals	74768	74768.0	
3	13	2021	12	Clients	India	Lv5_Visitors	Actuals	216178	216178.0	
4	14	2021	12	Clients	India	Lv4_Visitors	Actuals	180821	NaN	

```
df = df.drop(['index','KPI','Value'], axis=1)
```

```
df.head()
```

	Year	Month	Segment	Region	Value Type	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors
0	2020	12	Clients	India	Actuals	255723.0	NaN	NaN
1	2020	12	Clients	India	Actuals	NaN	180125.0	NaN
2	2020	12	Clients	India	Actuals	74768.0	NaN	NaN
3	2021	12	Clients	India	Actuals	216178.0	NaN	NaN

```
df
```

1295	2021	1	Customers	Aurangabad	Actuals	1340.0	NaN
1296	2022	1	Customers	India	Actuals	NaN	NaN
1297	2022	1	Customers	India	Actuals	NaN	NaN
1298	2022	1	Customers	Uddepy	Actuals	NaN	NaN
1299	2022	1	Customers	India	Actuals	NaN	81890.0
1300	2022	1	Customers	Uddepy	Actuals	NaN	NaN
1301	2022	1	Customers	Uddepy	Actuals	NaN	NaN
1302	2022	1	Customers	Dehradun	Actuals	NaN	NaN
1303	2022	1	Customers	Ujjain	Actuals	NaN	NaN
1304	2022	1	Customers	Uddepy	Actuals	NaN	22849.0
1305	2022	1	Customers	India	Actuals	19861.0	NaN
1306	2022	1	Customers	Dehradun	Actuals	NaN	NaN
1307	2022	1	Customers	Faridabad	Actuals	NaN	NaN
1308	2022	1	Customers	Ujjain	Actuals	NaN	NaN
1309	2022	1	Customers	Faridabad	Actuals	NaN	NaN
1310	2022	1	Customers	Dehradun	Actuals	NaN	NaN
1311	2022	1	Customers	Faridabad	Actuals	NaN	NaN
1312	2022	1	Customers	Ujjain	Actuals	NaN	NaN
1313	2022	1	Customers	Aurangabad	Actuals	NaN	NaN
1314	2022	1	Customers	Uddepy	Actuals	7281.0	NaN
1315	2022	1	Customers	Aurangabad	Actuals	NaN	NaN
1316	2022	1	Customers	Faridabad	Actuals	NaN	4168.0
1317	2022	1	Customers	Ujjain	Actuals	NaN	3831.0
1318	2022	1	Customers	Dehradun	Actuals	NaN	3786.0
1319	2022	1	Customers	Aurangabad	Actuals	NaN	NaN
1320	2022	1	Customers	Dehradun	Actuals	1693.0	NaN
1321	2022	1	Customers	Aurangabad	Actuals	NaN	1428.0
1322	2022	1	Customers	Ujjain	Actuals	1311.0	NaN
1323	2022	1	Customers	Faridabad	Actuals	1071.0	NaN
1324	2022	1	Customers	Aurangabad	Actuals	527.0	NaN

```
df.isnull().sum()
```

```
Year              0
Month             0
Segment           0
Region            0
Value Type        0
Lv5_Visitors    1002
Lv4_Visitors    1038
Lv3_Visitors    1078
Lv2_Visitors    1079
Lv1_Visitors    1103
dtype: int64
```

```
df = df.fillna(0)
```

```
df.isnull().sum()
```

```
Year            0
Month           0
Segment         0
Region          0
Value Type      0
Lv5_Visitors    0
Lv4_Visitors    0
Lv3_Visitors    0
Lv2_Visitors    0
Lv1_Visitors    0
dtype: int64
```

```
float_col = df.select_dtypes(include=['float64'])
for col in float_col.columns.values:
    df[col] = df[col].astype('int64')
```

```
df.head()
```

	Year	Month	Segment	Region	Value Type	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors
0	2020	12	Clients	India	Actuals	255723	0	0
1	2020	12	Clients	India	Actuals	0	180125	0
2	2020	12	Clients	India	Actuals	74768	0	0
3	2021	12	Clients	India	Actuals	216178	0	0

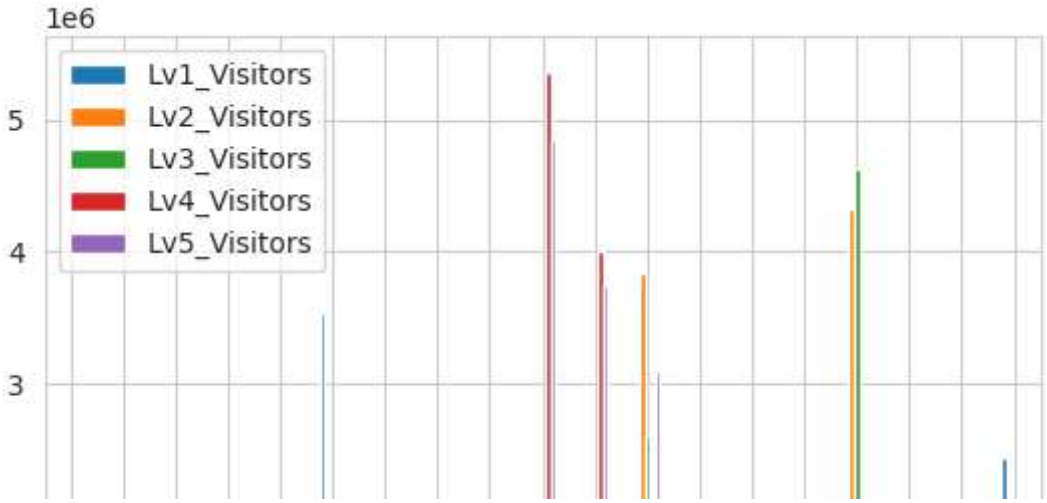
```
df1 = pd.pivot_table(df,index=['Region','Year'],aggfunc={'Lv5_Visitors':np.sum,'Lv4_Visitc
```

```
df1
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Vis
Aurangabad	**2020**	92083	63071	88109	29622	
	2021	102100	71711	81673	31773	
	2022	82786	54252	39666	14656	
Dehradun	**2020**	402599	295615	311845	111753	
	2021	352787	250456	214565	78363	
	2022	3536934	2110895	1550880	695236	3
Faridabad	**2020**	312711	275273	357617	166721	
	2021	359870	286148	306888	140728	
	2022	397426	274584	186429	50478	
India	**2020**	424743	1102014	371396	5369144	48
	2021	420940	2075004	1838458	4010243	37
	2022	1675999	3841281	2602188	821483	30
Indore	**2022**	1482479	872927	661969	361346	1
Uddepy	**2020**	1299320	979208	1304744	597155	1
	2021	1068843	772889	809171	365173	1
	2022	1104680	4325123	4627767	1754789	7
Ujjain	**2020**	369639	285890	332701	136803	
	2021	335540	232055	220982	94301	
	2022	2432691	1420593	1158731	514700	2

```
df1.plot(kind = 'bar')
```

```
<Axes: xlabel='Region,Year'>
```



As of my findings that Aurangabad is having the worst Performing, The reason behind there may be less publicity, due to less health issues in females. TO improve the visitors we can publish adds and campagins this helps us get clients and visitors for the poduct

- India is having the better growth 2020 , 2021 where uddepy is high in 2022

```
df2 = df1.drop_duplicates( keep='last')
```

```
df2
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Vis
Aurangabad	**2020**	92083	63071	88109	29622	
	2021	102100	71711	81673	31773	
	2022	82786	54252	39666	14656	
Dehradun	**2020**	402599	295615	311845	111753	
	2021	352787	250456	214565	78363	

```
df1['Visitors'] = df1['Lv5_Visitors']/df1['Lv1_Visitors']
```

Faridabad	**2020**	312711	275273	357617	166721	

```
df1
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Vis
Aurangabad	**2020**	92083	63071	88109	29622	
	2021	102100	71711	81673	31773	
	2022	82786	54252	39666	14656	
Dehradun	**2020**	402599	295615	311845	111753	
	2021	352787	250456	214565	78363	
	2022	3536934	2110895	1550880	695236	3
Faridabad	**2020**	312711	275273	357617	166721	
	2021	359870	286148	306888	140728	
	2022	397426	274584	186429	50478	
India	**2020**	424743	1102014	371396	5369144	48
	2021	420940	2075004	1838458	4010243	37
	2022	1675999	3841281	2602188	821483	30
Indore	**2022**	1482479	872927	661969	361346	1
Uddepy	**2020**	1299320	979208	1304744	597155	1
	2021	1068843	772889	809171	365173	1
	2022	1104680	4325123	4627767	1754789	7
Ujjain	**2020**	369639	285890	332701	136803	
	2021	335540	232055	220982	94301	
	2022	2432691	1420593	1158731	514700	2

top 3 states based on that created feature for all the available segments and each given year are India Uddepy Ujjain

```
df1['Visitors_1'] = df1['Lv4_Visitors']/df1['Lv3_Visitors']
```

```
df1
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Vis
Aurangabad	**2020**	92083	63071	88109	29622	
	2021	102100	71711	81673	31773	
	2022	82786	54252	39666	14656	
Dehradun	**2020**	402599	295615	311845	111753	
	2021	352787	250456	214565	78363	
	2022	3536934	2110895	1550880	695236	3
Faridabad	**2020**	312711	275273	357617	166721	
	2021	359870	286148	306888	140728	
	2022	397426	274584	186429	50478	
India	**2020**	424743	1102014	371396	5369144	48
	2021	420940	2075004	1838458	4010243	37
	2022	1675999	3841281	2602188	821483	30
Indore	**2022**	1482479	872927	661969	361346	1
Uddepy	**2020**	1299320	979208	1304744	597155	1
	2021	1068843	772889	809171	365173	1
	2022	1104680	4325123	4627767	1754789	7
Ujjain	**2020**	369639	285890	332701	136803	
	2021	335540	232055	220982	94301	
	2022	2432691	1420593	1158731	514700	2

created another metric but i have seen there is no significant change while comparing the both metric.

Part 3: Prediction

```
df
```

1295	2021	1	Customers	Aurangabad	Actuals	1340	0
1296	2022	1	Customers	India	Actuals	0	0
1297	2022	1	Customers	India	Actuals	0	0
1298	2022	1	Customers	Uddepy	Actuals	0	0
1299	2022	1	Customers	India	Actuals	0	81890
1300	2022	1	Customers	Uddepy	Actuals	0	0
1301	2022	1	Customers	Uddepy	Actuals	0	0
1302	2022	1	Customers	Dehradun	Actuals	0	0
1303	2022	1	Customers	Ujjain	Actuals	0	0
1304	2022	1	Customers	Uddepy	Actuals	0	22849
1305	2022	1	Customers	India	Actuals	19861	0
1306	2022	1	Customers	Dehradun	Actuals	0	0
1307	2022	1	Customers	Faridabad	Actuals	0	0
1308	2022	1	Customers	Ujjain	Actuals	0	0
1309	2022	1	Customers	Faridabad	Actuals	0	0
1310	2022	1	Customers	Dehradun	Actuals	0	0
1311	2022	1	Customers	Faridabad	Actuals	0	0
1312	2022	1	Customers	Ujjain	Actuals	0	0
1313	2022	1	Customers	Aurangabad	Actuals	0	0
1314	2022	1	Customers	Uddepy	Actuals	7281	0
1315	2022	1	Customers	Aurangabad	Actuals	0	0
1316	2022	1	Customers	Faridabad	Actuals	0	4168
1317	2022	1	Customers	Ujjain	Actuals	0	3831
1318	2022	1	Customers	Dehradun	Actuals	0	3786
1319	2022	1	Customers	Aurangabad	Actuals	0	0
1320	2022	1	Customers	Dehradun	Actuals	1693	0
1321	2022	1	Customers	Aurangabad	Actuals	0	1428
1322	2022	1	Customers	Ujjain	Actuals	1311	0
1323	2022	1	Customers	Faridabad	Actuals	1071	0
1324	2022	1	Customers	Aurangabad	Actuals	527	0

Convert to date Time

```
from datetime import date
DATE = []
for y, m in zip(df.Year, df.Month):
    DATE.append(date(y, m, 1))


df['DATE'] = DATE


df
```

1306	2022	1	Customers	Dehradun	Actuals	0	0
1307	2022	1	Customers	Faridabad	Actuals	0	0
1308	2022	1	Customers	Ujjain	Actuals	0	0
1309	2022	1	Customers	Faridabad	Actuals	0	0
1310	2022	1	Customers	Dehradun	Actuals	0	0
1311	2022	1	Customers	Faridabad	Actuals	0	0
1312	2022	1	Customers	Ujjain	Actuals	0	0
1313	2022	1	Customers	Aurangabad	Actuals	0	0
1314	2022	1	Customers	Uddepy	Actuals	7281	0
1315	2022	1	Customers	Aurangabad	Actuals	0	0
1316	2022	1	Customers	Faridabad	Actuals	0	4168
1317	2022	1	Customers	Ujjain	Actuals	0	3831
1318	2022	1	Customers	Dehradun	Actuals	0	3786
1319	2022	1	Customers	Aurangabad	Actuals	0	0
1320	2022	1	Customers	Dehradun	Actuals	1693	0
1321	2022	1	Customers	Aurangabad	Actuals	0	1428
1322	2022	1	Customers	Ujjain	Actuals	1311	0
1323	2022	1	Customers	Faridabad	Actuals	1071	0
1324	2022	1	Customers	Aurangabad	Actuals	527	0

```
df = df.drop(['Year','Month','Value Type'], axis=1)
```

```
df.head()
```

	Segment	Region	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors	Lv2_Visitors	Lv1_Vi
0	Clients	India	255723	0	0	0	
1	Clients	India	0	180125	0	0	
2	Clients	India	74768	0	0	0	

```
duplicate = df[df.duplicated()]
```

```
duplicate.sum()
```

```
Segment         0.0
Region          0.0
Lv5_Visitors    0.0
Lv4_Visitors    0.0
Lv3_Visitors    0.0
Lv2_Visitors    0.0
Lv1_Visitors    0.0
DATE            0.0
dtype: float64
```

```
df.describe()
```

	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors	Lv2_Visitors	Lv1_Visitors
count	1325.000000	1325.000000	1325.000000	1325.000000	1325.000000
mean	10369.386415	11580.729811	12879.833208	14784.142642	12267.298113
std	41711.399423	41305.758227	48404.177911	57147.755988	45385.746547
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	428245.000000	363340.000000	431464.000000	435056.000000	429564.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1325 entries, 0 to 1324
```

```
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Segment       1325 non-null   object
 1   Region        1325 non-null   object
 2   Lv5_Visitors  1325 non-null   int64 
 3   Lv4_Visitors  1325 non-null   int64 
 4   Lv3_Visitors  1325 non-null   int64 
 5   Lv2_Visitors  1325 non-null   int64 
 6   Lv1_Visitors  1325 non-null   int64 
 7   DATE          1325 non-null   object
dtypes: int64(5), object(3)
memory usage: 82.9+ KB
```

```
df['DATE'] =  pd.to_datetime(df['DATE'], format='%Y/%m/%d')
```

```
df.describe(include = 'all')
```

```
<ipython-input-173-74aa2f970831>:1: FutureWarning: Treating datetime data as categ
  df.describe(include = 'all')
```

	Segment	Region	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors	Lv2_Visit
count	1325	1325	1325.000000	1325.000000	1325.000000	1325.00(
unique	2	7	NaN	NaN	NaN	
top	Customers	Dehradun	NaN	NaN	NaN	
freq	1018	240	NaN	NaN	NaN	
first	NaN	NaN	NaN	NaN	NaN	
last	NaN	NaN	NaN	NaN	NaN	
mean	NaN	NaN	10369.386415	11580.729811	12879.833208	14784.142
std	NaN	NaN	41711.399423	41305.758227	48404.177911	57147.755
min	NaN	NaN	0.000000	0.000000	0.000000	0.00(
25%	NaN	NaN	0.000000	0.000000	0.000000	0.00(
50%	NaN	NaN	0.000000	0.000000	0.000000	0.00(
75%	NaN	NaN	0.000000	0.000000	0.000000	0.00(

```
# Remove columns which are not required in predictions

cols = ['Segment', 'Region', 'Lv4_Visitors','Lv3_Visitors','Lv2_Visitors','Lv1_Visitors']
df.drop(cols, axis = 1, inplace = True)
df.head()
```

	Lv5_Visitors	DATE
0	255723	2020-12-01
1	0	2020-12-01
2	74768	2020-12-01
3	216178	2021-12-01
4	0	2021-12-01

```
# Sort the Date
df = df.sort_values('DATE')

#print the sorted values
print(df.head(1))

#check any missing values
df.isnull().sum()
```

```
            Lv5_Visitors       DATE
     1245              0 2020-01-01
     Lv5_Visitors    0
     DATE            0
     dtype: int64
```

```
# grouping visitore according to Date
df.groupby('DATE')['Lv5_Visitors'].sum().reset_index()

# min and max values of Date
print(df['DATE'].min())
print(df['DATE'].max())
```

```
     2020-01-01 00:00:00
     2022-12-01 00:00:00
```

```
df = df.set_index('DATE')
df.index
```

```
     DatetimeIndex(['2020-01-01', '2020-01-01', '2020-01-01', '2020-01-01',
                    '2020-01-01', '2020-01-01', '2020-01-01', '2020-01-01',
                    '2020-01-01', '2020-01-01',
                    ...
                    '2022-12-01', '2022-12-01', '2022-12-01', '2022-12-01',
                    '2022-12-01', '2022-12-01', '2022-12-01', '2022-12-01',
                    '2022-12-01', '2022-12-01'],
                   dtype='datetime64[ns]', name='DATE', length=1325, freq=None)
```

```
y = df['Lv5_Visitors'].resample('MS').mean()
y['2020':]
```

```
     DATE
     2020-01-01    13106.090909
     2020-02-01    12543.000000
     2020-03-01    14963.387097
```

```
2020-04-01     7766.344828
2020-05-01    22528.241379
2020-06-01    17323.266667
2020-07-01    14076.833333
2020-08-01    11379.466667
2020-09-01    13246.633333
2020-10-01    17026.566667
2020-11-01    15085.066667
2020-12-01    13769.400000
2021-01-01    14388.466667
2021-02-01    12302.000000
2021-03-01    12850.866667
2021-04-01    10916.500000
2021-05-01    11418.866667
2021-06-01     8910.000000
2021-07-01     9290.187500
2021-08-01     9393.718750
2021-09-01     9117.937500
2021-10-01     9632.062500
2021-11-01    10714.424242
2021-12-01     9831.343750
2022-01-01    10229.208333
2022-02-01     9091.612245
2022-03-01     8187.104167
2022-04-01     8112.102041
2022-05-01     7489.122449
2022-06-01     7374.346939
2022-07-01     7889.204082
2022-08-01     7437.653061
2022-09-01     6992.000000
2022-10-01     8007.260000
2022-11-01     9333.530612
2022-12-01     2304.320000
Freq: MS, Name: Lv5_Visitors, dtype: float64
```

```
y.plot(figsize = (15, 6))
plt.show()
```