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()
```

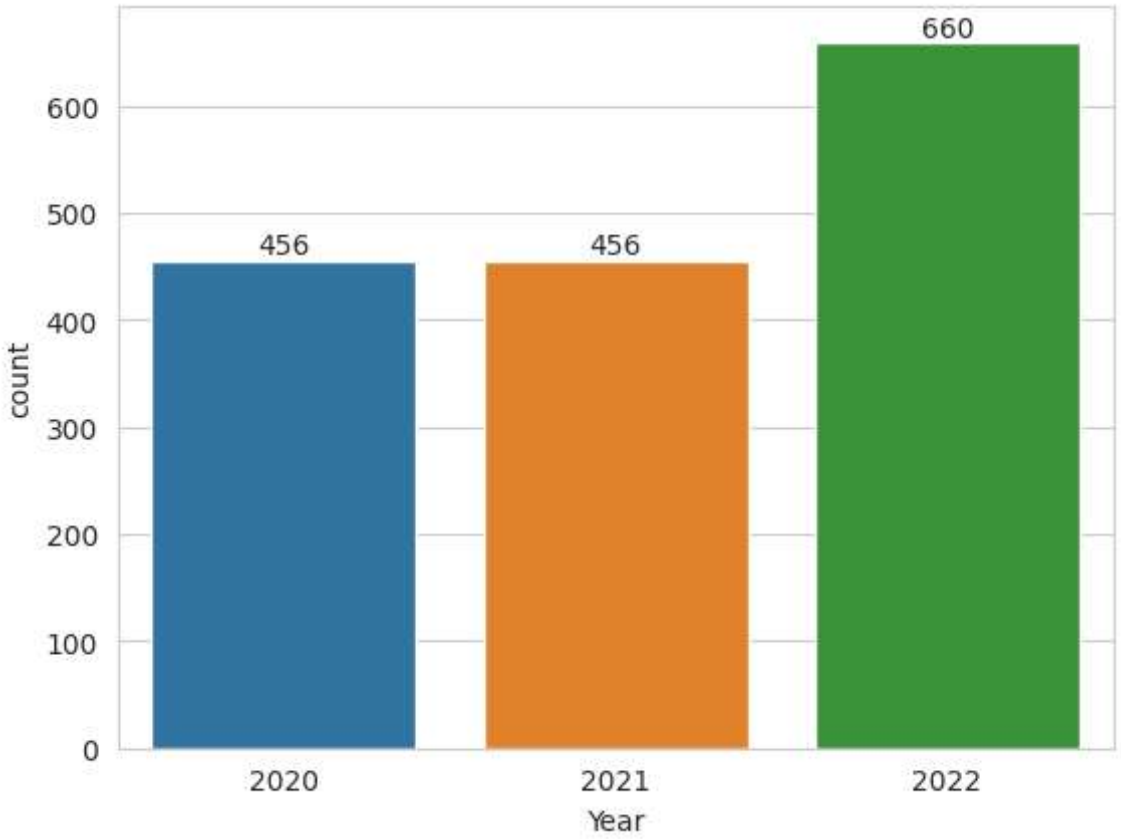
```
88949       1
330165      1
9334        1
469431      1
897         1
2250        1
3201        1
3616        1
4399        1
6103        1
6178        1
8489        1
527         1
Name: Value, dtype: int64
```

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

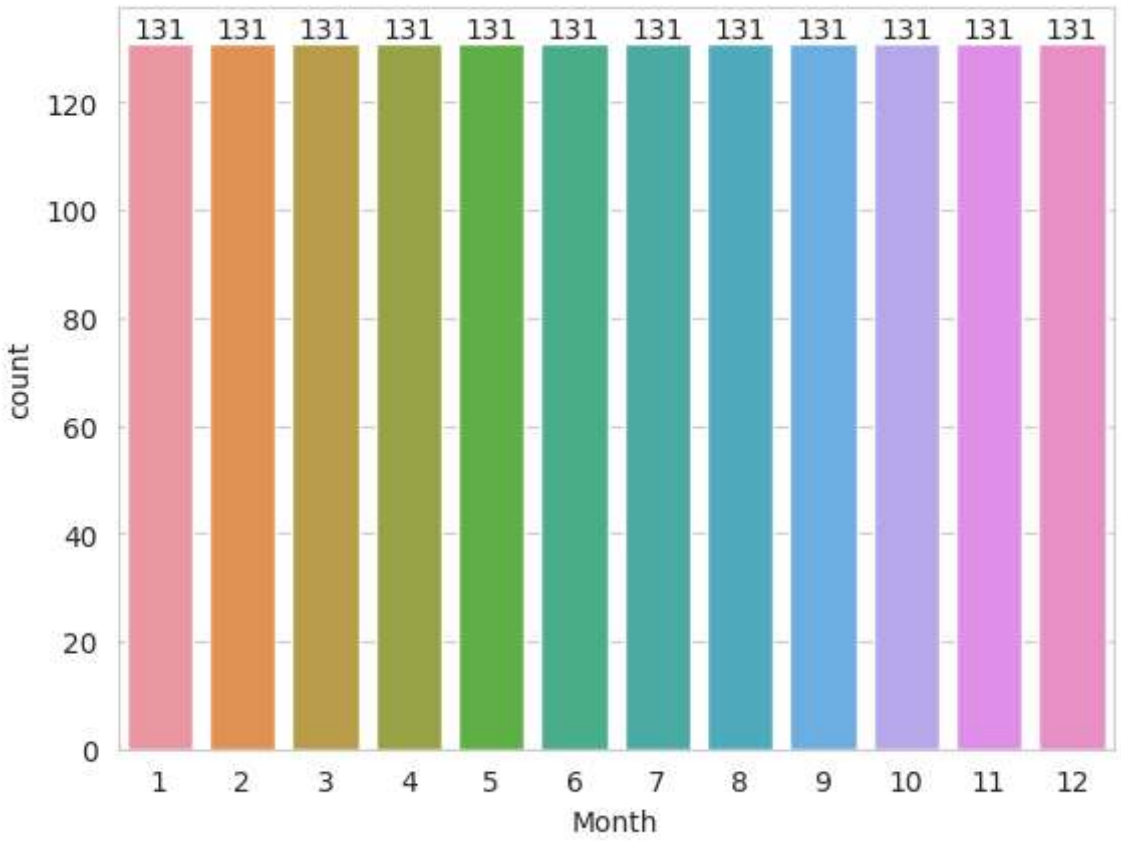
```
1558
```

Univerate analysis to find the value counts using plots

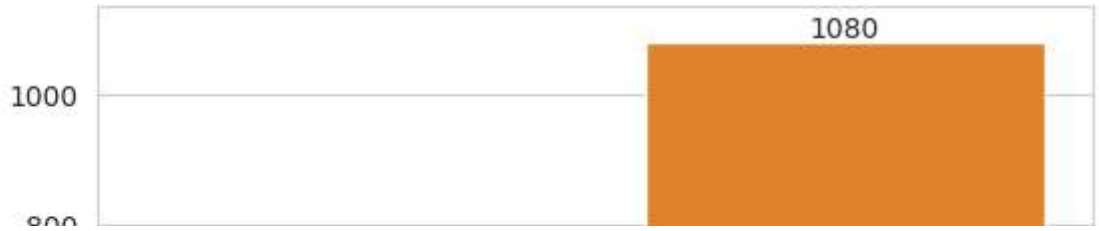
```
ax = sns.countplot(x=df["Year"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



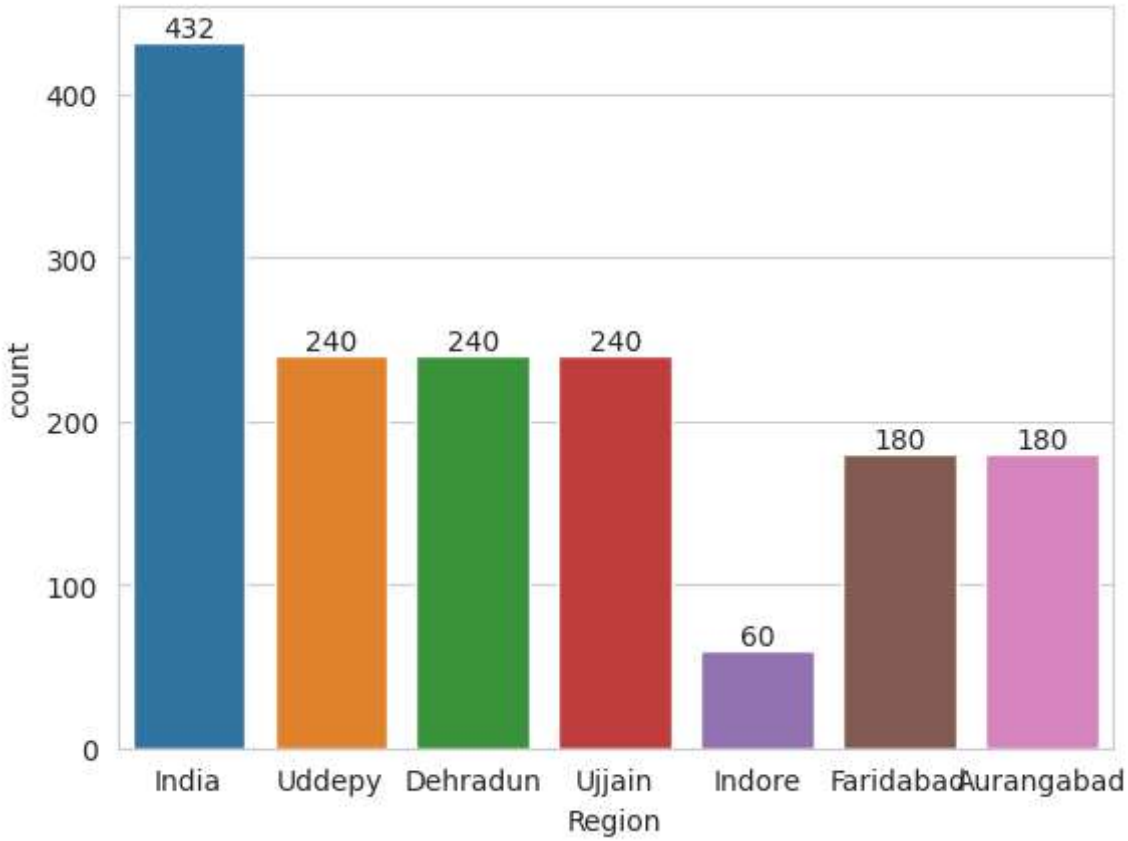
```
ax = sns.countplot(x=df["Month"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



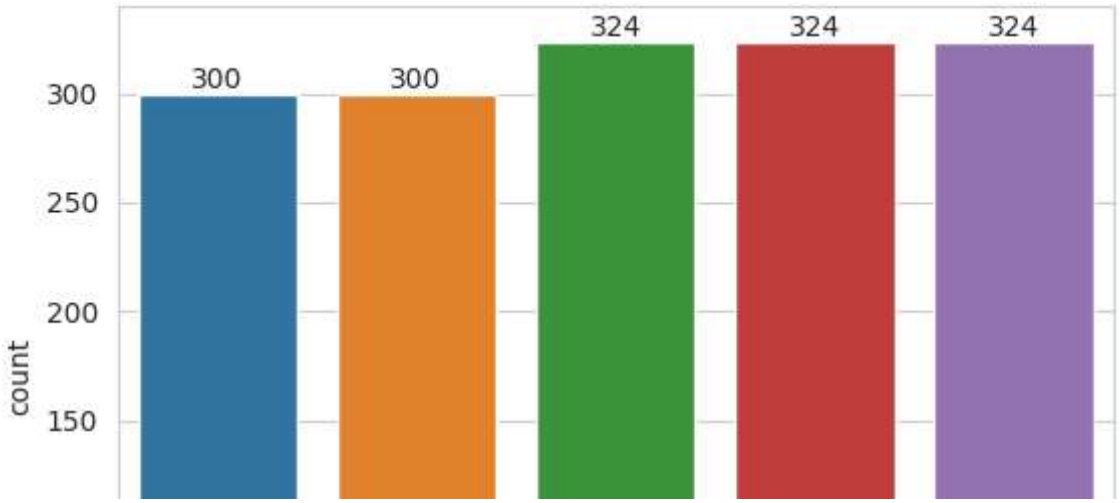

```
ax = sns.countplot(x=df["Segment"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



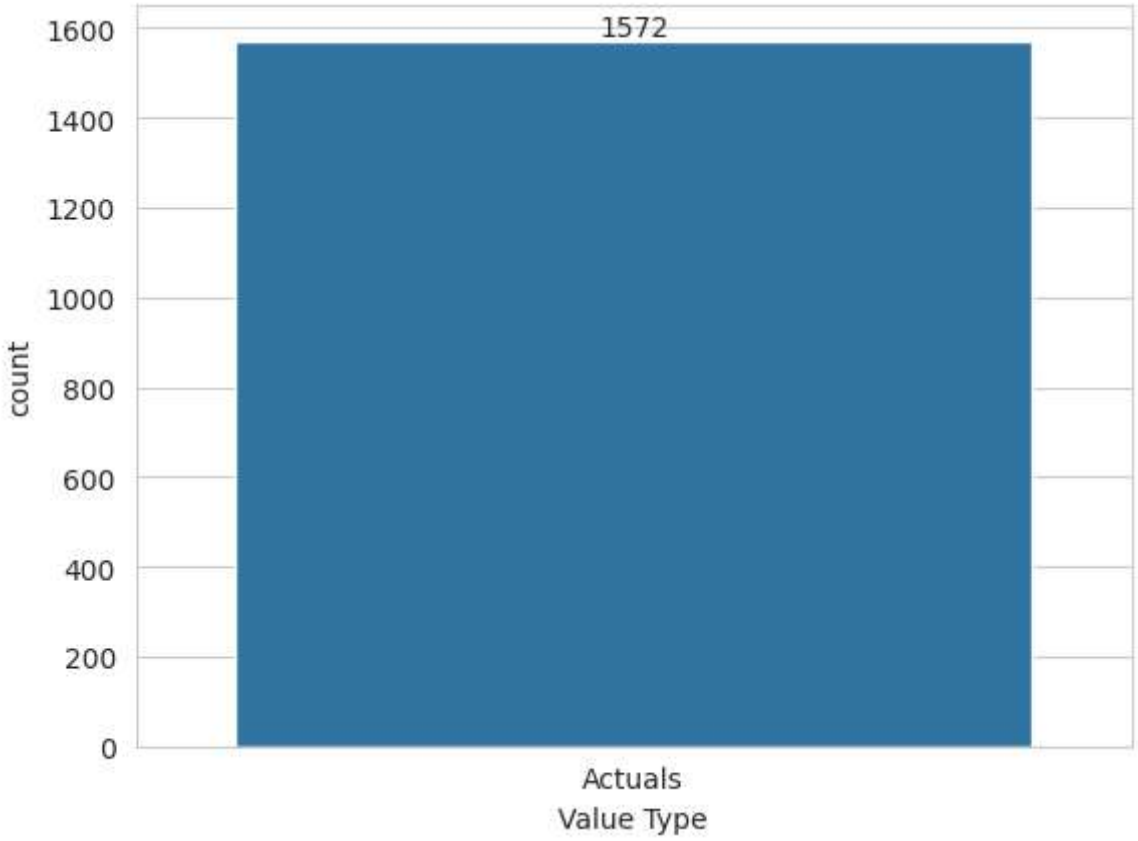

```
ax = sns.countplot(x=df["Region"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



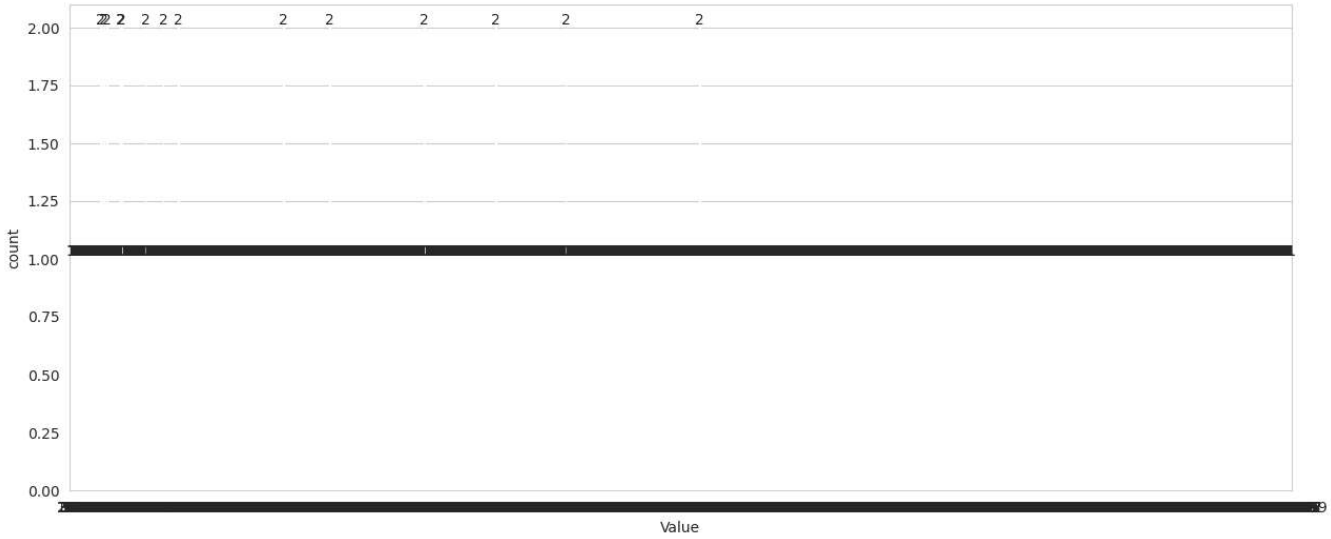

```
ax = sns.countplot(x=df["KPI"])
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```




```
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
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 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
```

Segment	Month	Year	Value
Clients	**1**	**2020**	9558865
		2021	9780901
		2022	12672657
	2	**2020**	8430064
		2021	7860680
		2022	11231462
	3	**2020**	9795800
		2021	7648506
		2022	11126527
	4	**2020**	13356362
		2021	7537218
		2022	10358367
	5	**2020**	50373958
		2021	8156773
		2022	10737298
	6	**2020**	11065448
		2021	6880124
		2022	10845050
	7	**2020**	9563651
		2021	7273951
		2022	11354138
	8	**2020**	8196088
		2021	8083934
		2022	10872735
	9	**2020**	8936728
		2021	7595927
		2022	10841918
	10	**2020**	11470848
		2021	8015893

```
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).astype
```

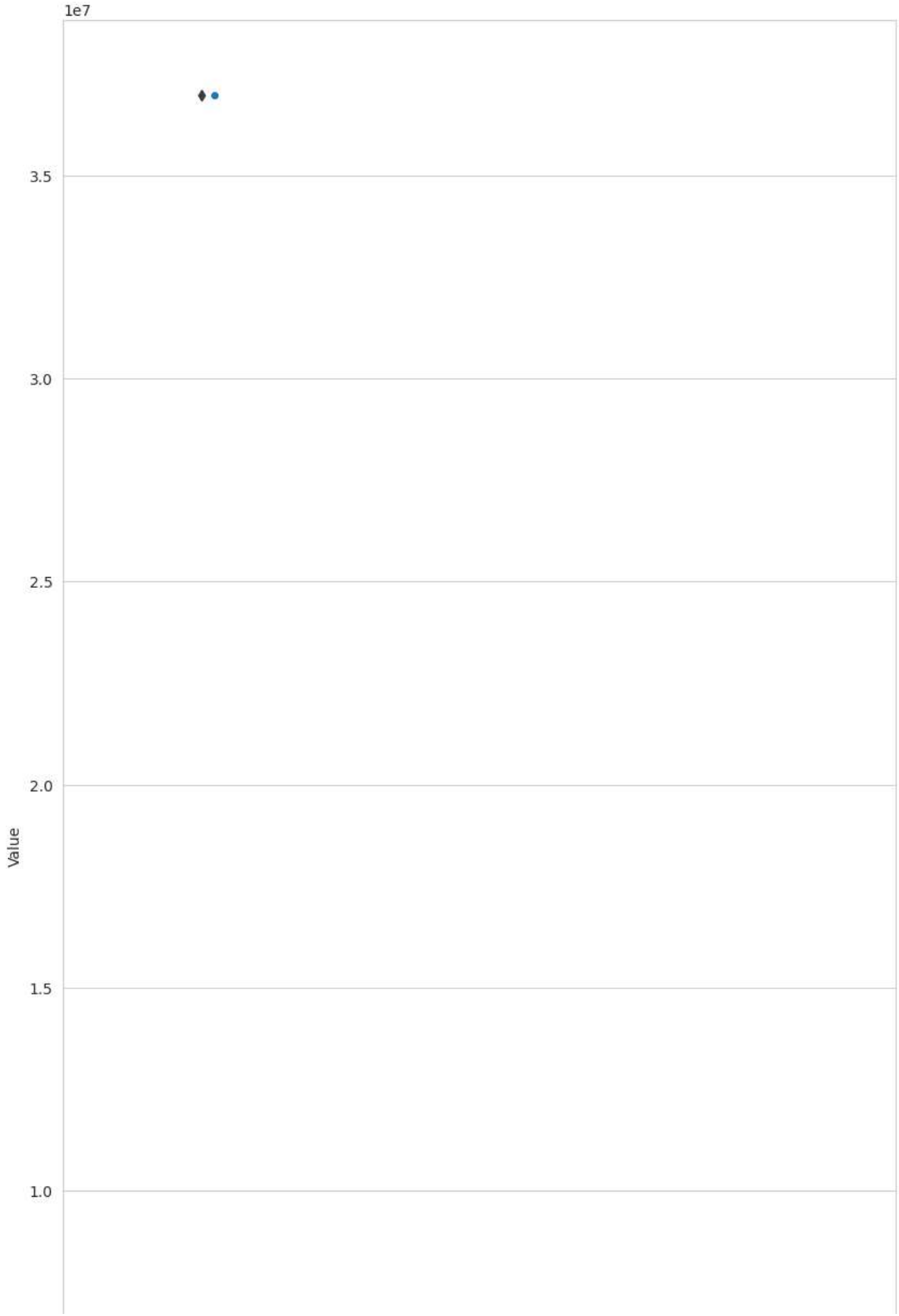
```
table
```

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%
Ujjain	**2020**	1176987	15.012056970428855%
	2021	920026	11.734609410533658%

finding the outliers using boxplot

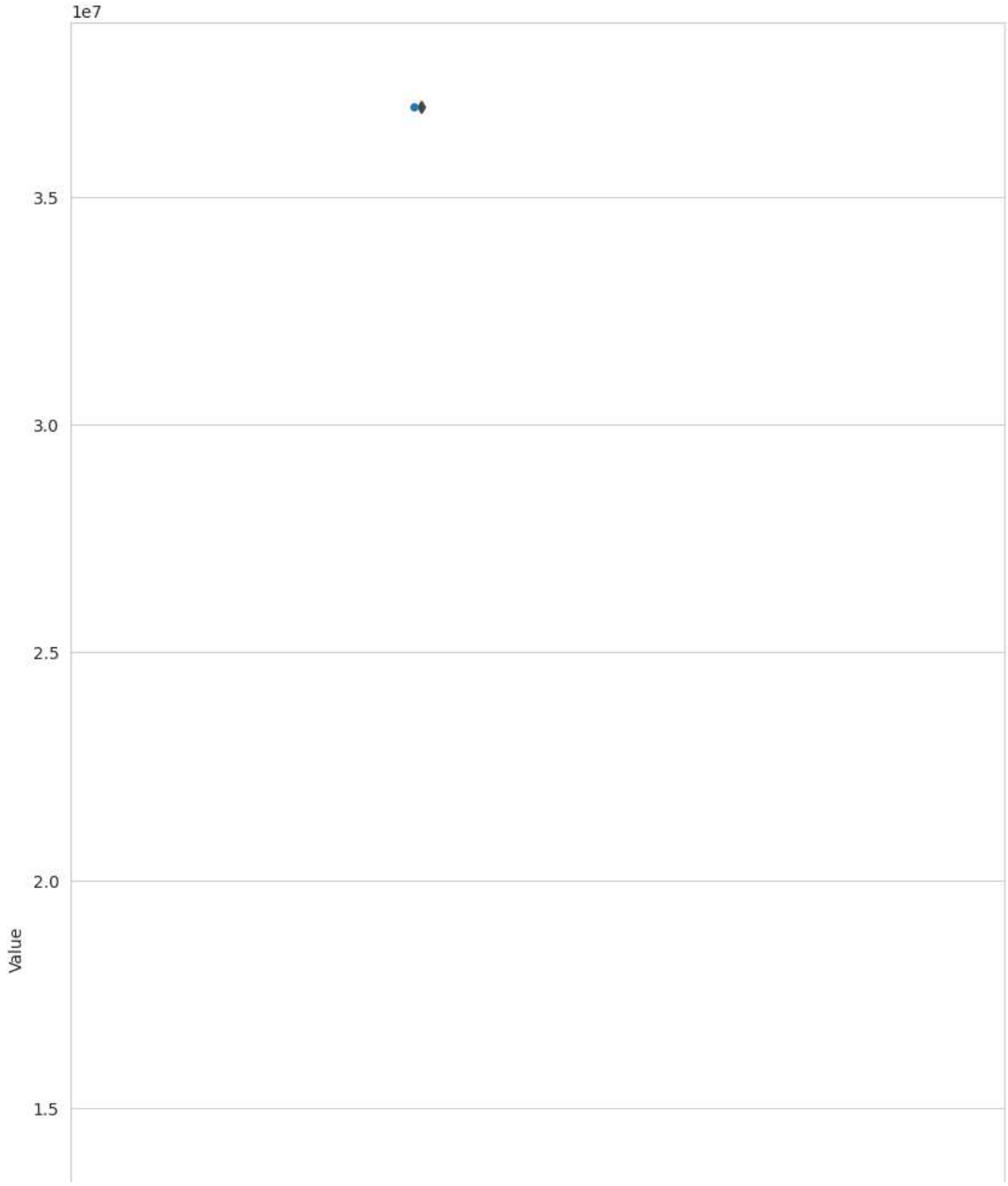
```
fig, ax = plt.subplots(figsize=(10, 20))

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


1e7
3.5
3.0
2.5
2.0
1.5
1.0
Value

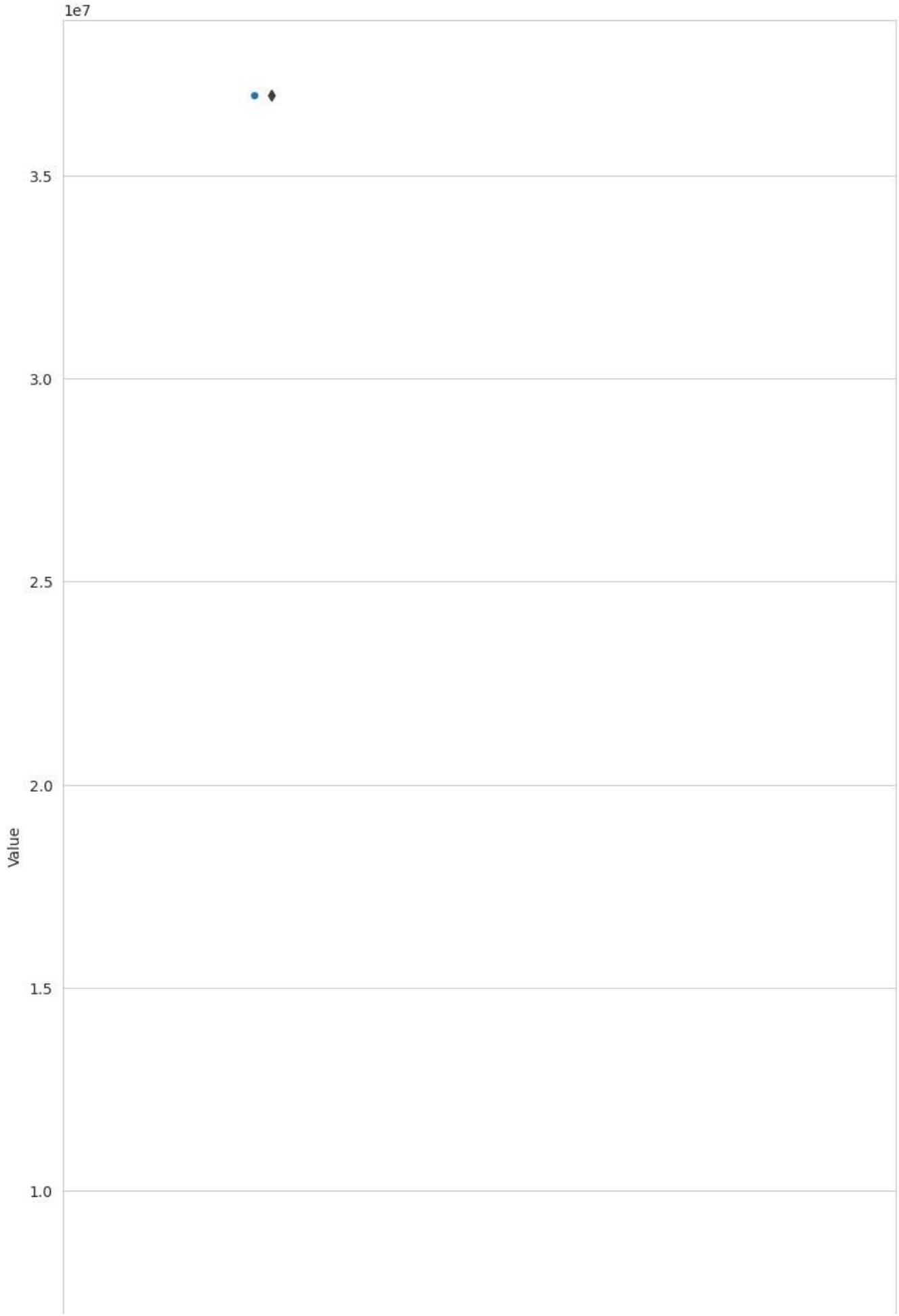
```
fig, ax = plt.subplots(figsize=(10, 20))

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



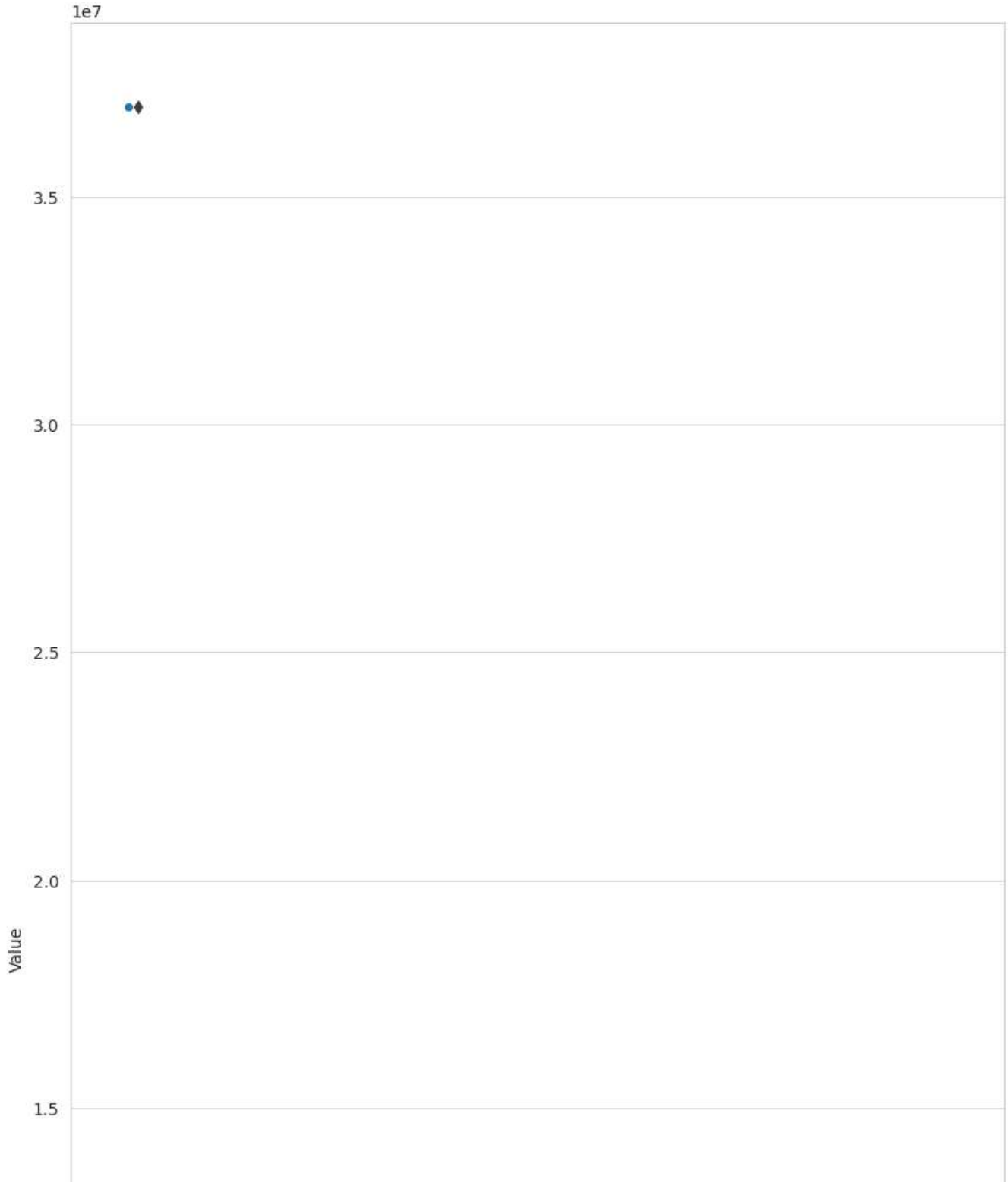

```
fig, ax = plt.subplots(figsize=(10, 20))

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


1e7
3.5
3.0
2.5
2.0
1.5
1.0
Value

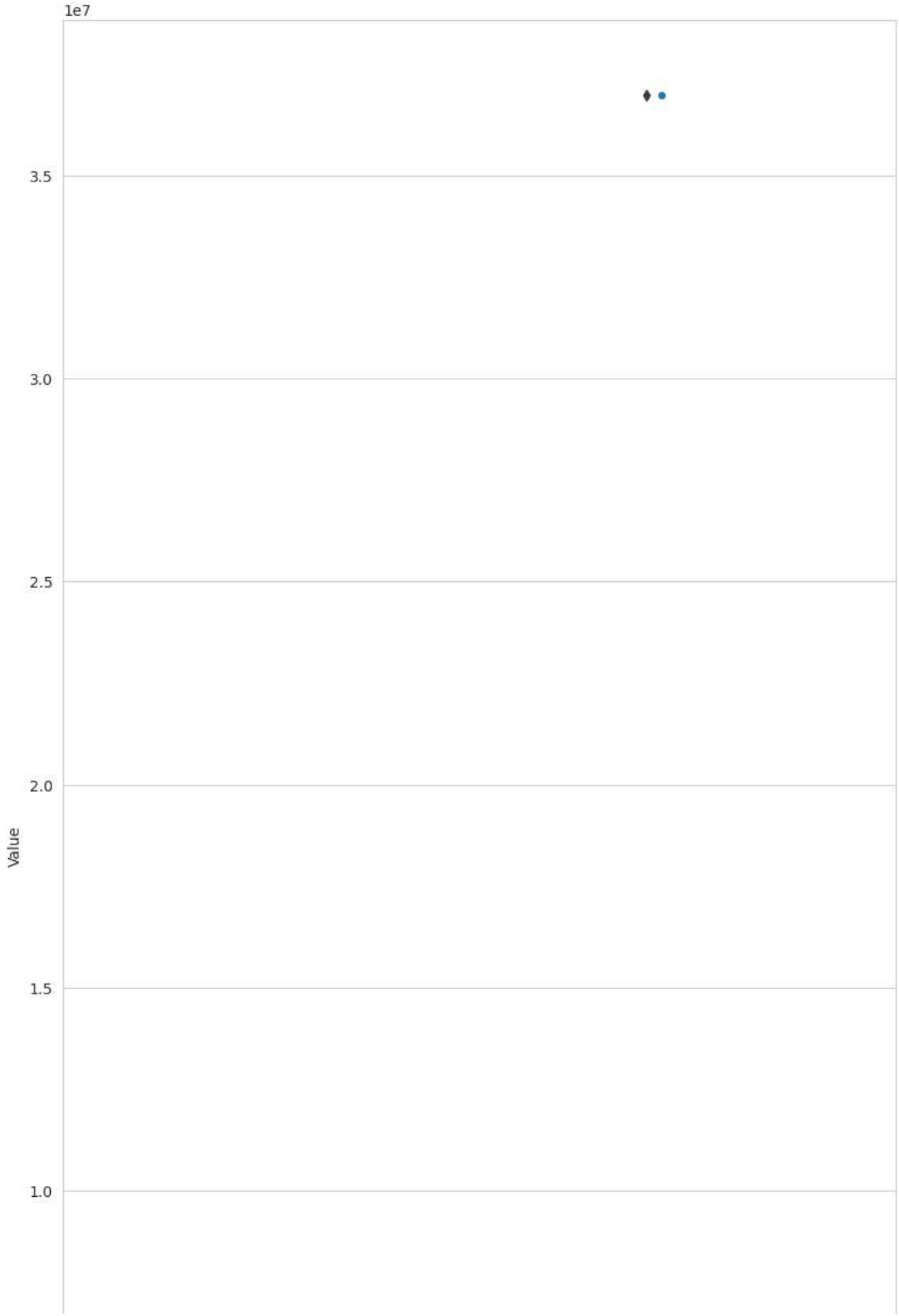
```
fig, ax = plt.subplots(figsize=(10, 20))

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



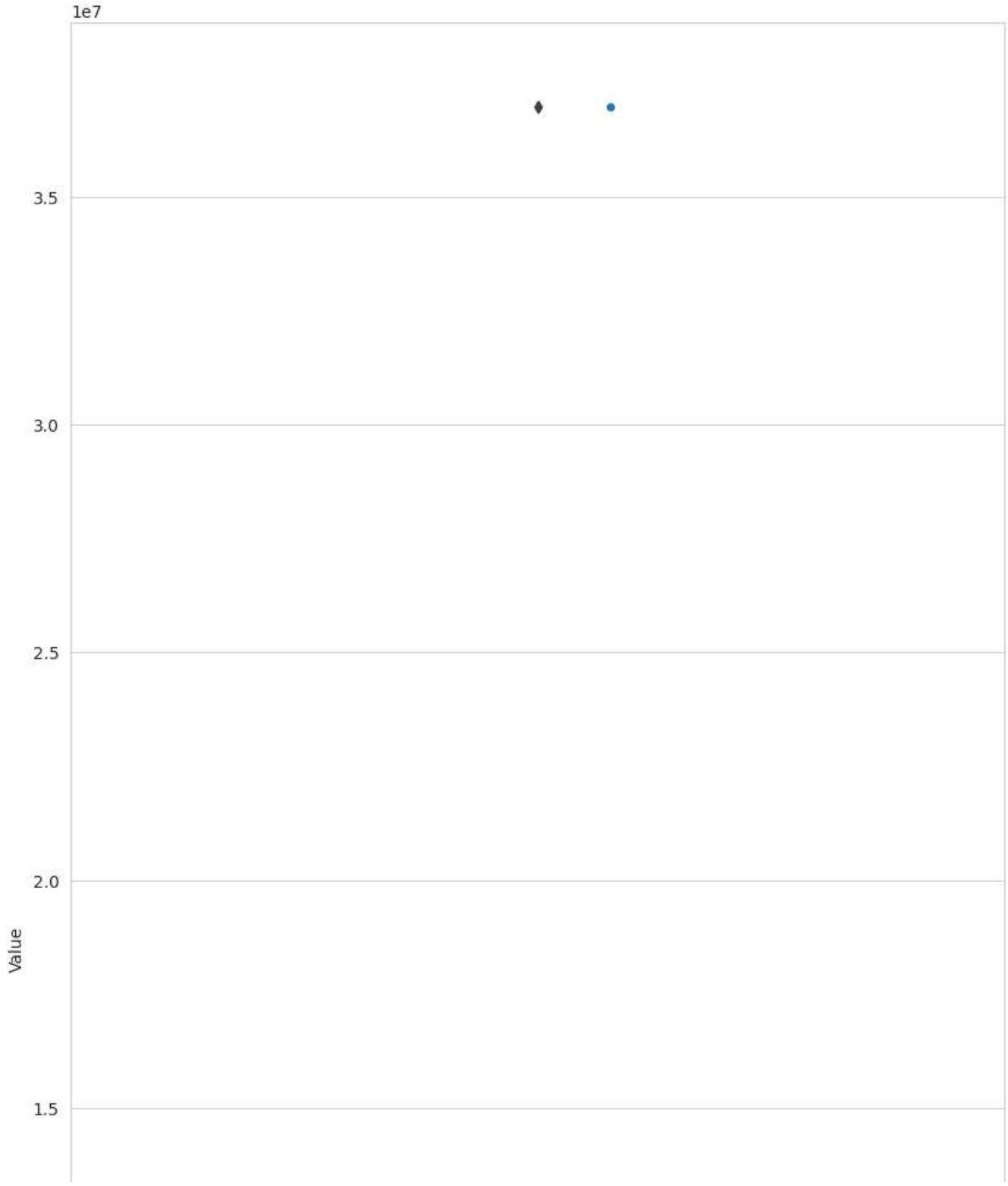

```
fig, ax = plt.subplots(figsize=(10, 20))

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


1e7
3.5
3.0
2.5
2.0
1.5
1.0
Value

```
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)]
```

	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
8	2021	12	Clients	India	Lv1_Visitors	Actuals	3543797
9	2021	12	Clients	India	Lv2_Visitors	Actuals	2199420
10	2021	12	Clients	India	Lv3_Visitors	Actuals	1105675
11	2021	12	Clients	India	Lv3_Visitors	Actuals	677096
12	2021	12	Clients	India	Lv4_Visitors	Actuals	632424
16	2022	12	Clients	India	Lv1_Visitors	Actuals	3223690
17	2022	12	Clients	India	Lv2_Visitors	Actuals	2106583
18	2022	12	Clients	India	Lv3_Visitors	Actuals	1725522
19	2022	12	Clients	India	Lv4_Visitors	Actuals	748273
20	2022	12	Clients	Uddepy	Lv1_Visitors	Actuals	629047
41	2020	11	Clients	India	Lv1_Visitors	Actuals	3634761
42	2020	11	Clients	India	Lv2_Visitors	Actuals	2510682
43	2020	11	Clients	India	Lv3_Visitors	Actuals	1346038
44	2020	11	Clients	India	Lv3_Visitors	Actuals	751349
45	2020	11	Clients	India	Lv4_Visitors	Actuals	750710
49	2021	11	Clients	India	Lv1_Visitors	Actuals	3665788
50	2021	11	Clients	India	Lv2_Visitors	Actuals	2261094
51	2021	11	Clients	India	Lv3_Visitors	Actuals	1154974
52	2021	11	Clients	India	Lv4_Visitors	Actuals	694316
53	2021	11	Clients	India	Lv3_Visitors	Actuals	688464
57	2022	11	Clients	India	Lv1_Visitors	Actuals	3558134
58	2022	11	Clients	India	Lv2_Visitors	Actuals	2299894
59	2022	11	Clients	India	Lv3_Visitors	Actuals	1753542
60	2022	11	Clients	India	Lv4_Visitors	Actuals	843778
61	2022	11	Clients	Uddepy	Lv1_Visitors	Actuals	668392

```
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)
```

92	2021	10	Clients	India	Lv3_Visitors	Actuals	1084128

```
df.shape
```

```
(1325, 7)
```

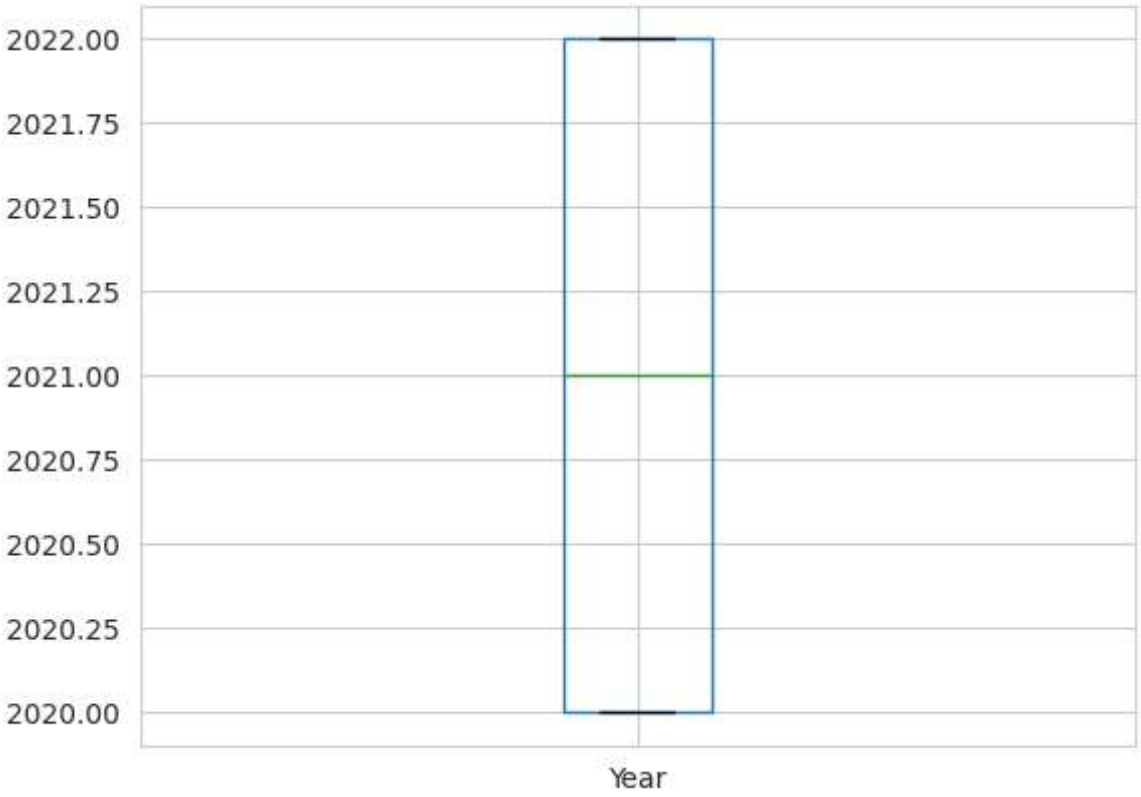
```
lowerbound,upperbound = outlier_treatment(df.Month)
df[(df.Month < lowerbound) | (df.Month > upperbound)]
df.drop(df[ (df.Month > upperbound) | (df.Month < lowerbound) ].index , inplace=True)
```

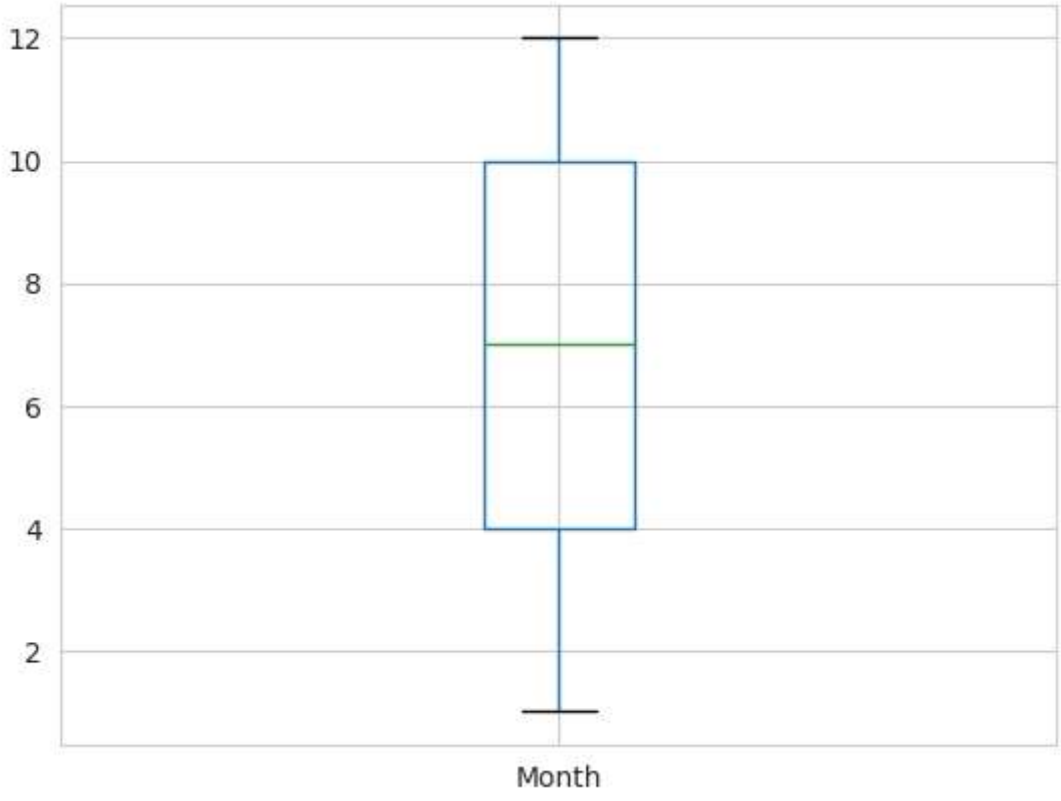
```
df.shape
```

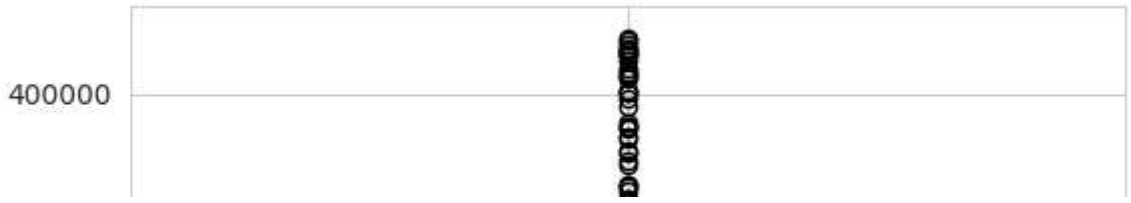
```
(1325, 7)
```

```
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])
```



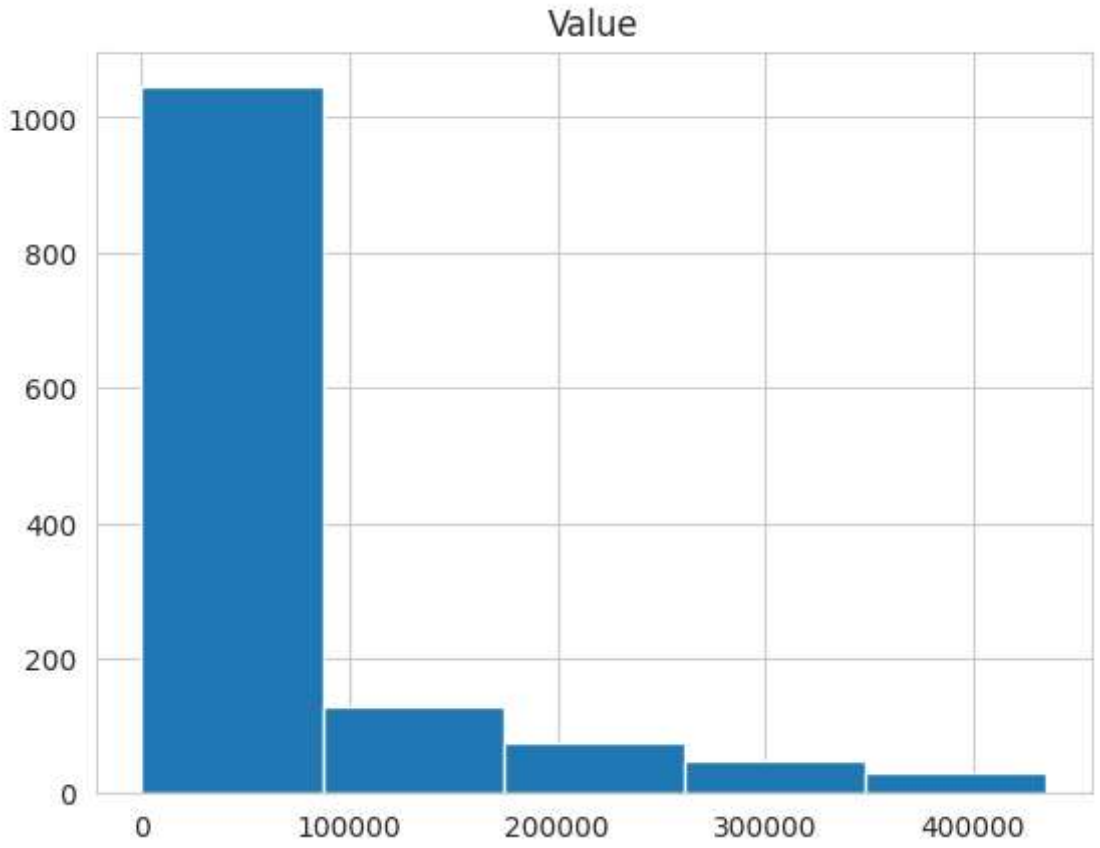




Histogram to find data disturbution

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

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



291	2020	5	Clients	India	Lv3_Visitors	Actuals	1166935

Part 2: Prescriptive Analysis

Transpose of data as given in problem statement

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

299	2021	5	Clients	India	Lv3_Visitors	Actuals	598734

```
data
```

	Value	KPI
5	255723	Lv5_Visitors
6	180125	Lv4_Visitors
7	74768	Lv5_Visitors
13	216178	Lv5_Visitors
14	180821	Lv4_Visitors
15	69345	Lv5_Visitors
21	378447	Lv2_Visitors
22	371666	Lv3_Visitors
23	241370	Lv1_Visitors
24	178976	Lv1_Visitors
25	149024	Lv2_Visitors
26	119475	Lv1_Visitors
27	118950	Lv4_Visitors
28	118570	Lv3_Visitors
29	112987	Lv2_Visitors
30	78884	Lv3_Visitors
31	75211	Lv5_Visitors
32	72804	Lv2_Visitors
33	64064	Lv3_Visitors
34	53146	Lv4_Visitors
35	37404	Lv4_Visitors
36	30773	Lv4_Visitors
37	14855	Lv5_Visitors
38	7880	Lv5_Visitors
39	5632	Lv5_Visitors
40	4335	Lv5_Visitors
46	272017	Lv5_Visitors
47	187646	Lv4_Visitors
48	81680	Lv5_Visitors
54	243224	Lv5_Visitors

55 193567 Lv4 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

70 132765 Lv4 Visitors

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

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

73 80884 Lv2_Visitors

```
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_Visitors
5	2020	12	Clients	India	Lv5_Visitors	Actuals	255723	255723.0	NaN
6	2020	12	Clients	India	Lv4_Visitors	Actuals	180125	NaN	180125.0
7	2020	12	Clients	India	Lv5_Visitors	Actuals	74768	74768.0	NaN
13	2021	12	Clients	India	Lv5_Visitors	Actuals	216178	216178.0	NaN
14	2021	12	Clients	India	Lv4_Visitors	Actuals	180821	NaN	180821.0

97 67336 Lv3_Visitors

```
df.reset_index(inplace = True)

df.head()
```

	index	Year	Month	Segment	Region	KPI	Value Type	Value	Lv5_Visitors	Lv4_Vi
0	5	2020	12	Clients	India	Lv5_Visitors	Actuals	255723	255723.0	
1	6	2020	12	Clients	India	Lv4_Visitors	Actuals	180125	NaN	18
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	18

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

```
df.head()
```

	Year	Month	Segment	Region	Value Type	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors	Lv2_V
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	
4	2021	12	Clients	India	Actuals	NaN	180821.0	NaN	

```
df
```

	Year	Month	Segment	Region	Value Type	Lv5_Visitors	Lv4_Visitors	Lv3_Vis
0	2020	12	Clients	India	Actuals	255723.0	NaN	
1	2020	12	Clients	India	Actuals	NaN	180125.0	
2	2020	12	Clients	India	Actuals	74768.0	NaN	
3	2021	12	Clients	India	Actuals	216178.0	NaN	
4	2021	12	Clients	India	Actuals	NaN	180821.0	
5	2021	12	Clients	India	Actuals	69345.0	NaN	
6	2022	12	Clients	Uddepy	Actuals	NaN	NaN	
7	2022	12	Clients	Uddepy	Actuals	NaN	NaN	37
8	2022	12	Clients	Dehradun	Actuals	NaN	NaN	
9	2022	12	Clients	Ujjain	Actuals	NaN	NaN	
10	2022	12	Clients	Dehradun	Actuals	NaN	NaN	
11	2022	12	Clients	Indore	Actuals	NaN	NaN	
12	2022	12	Clients	Uddepy	Actuals	NaN	118950.0	
13	2022	12	Clients	Dehradun	Actuals	NaN	NaN	11
14	2022	12	Clients	Ujjain	Actuals	NaN	NaN	
15	2022	12	Clients	Ujjain	Actuals	NaN	NaN	7
16	2022	12	Clients	India	Actuals	75211.0	NaN	
17	2022	12	Clients	Indore	Actuals	NaN	NaN	
18	2022	12	Clients	Indore	Actuals	NaN	NaN	6
19	2022	12	Clients	Dehradun	Actuals	NaN	53146.0	
20	2022	12	Clients	Ujjain	Actuals	NaN	37404.0	
21	2022	12	Clients	Indore	Actuals	NaN	30773.0	
22	2022	12	Clients	Uddepy	Actuals	14855.0	NaN	
23	2022	12	Clients	Dehradun	Actuals	7880.0	NaN	
24	2022	12	Clients	Ujjain	Actuals	5632.0	NaN	
25	2022	12	Clients	Indore	Actuals	4335.0	NaN	
26	2020	11	Clients	India	Actuals	272017.0	NaN	
27	2020	11	Clients	India	Actuals	NaN	187646.0	
28	2020	11	Clients	India	Actuals	81680.0	NaN	
29	2021	11	Clients	India	Actuals	243224.0	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
```

38	2022	11	Clients	Indore	Actuals	NaN	NaN

```
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
```

48	2022	11	Clients	Indore	Actuals	NaN	33489.0

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

```
df.head()
```

61 2022 10 Clients India Actuals 254773.0 NaN

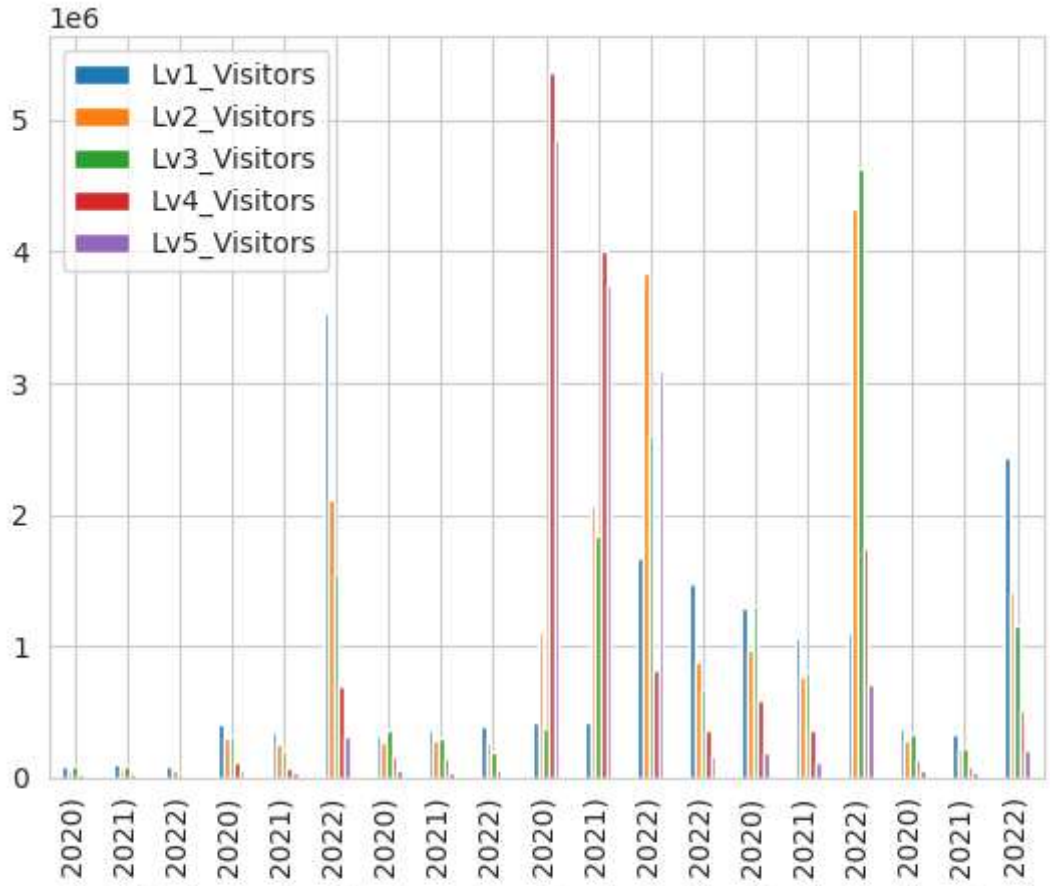
```
df1 = pd.pivot_table(df,index=['Region','Year'],aggfunc={'Lv5_Visitors':np.sum,'Lv4_Visitors'
```

```
df1
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Visitors
Aurangabad	**2020**	92083	63071	88109	29622	12612
	2021	102100	71711	81673	31773	12801
	2022	82786	54252	39666	14656	5088
Dehradun	**2020**	402599	295615	311845	111753	50258
	2021	352787	250456	214565	78363	34827
	2022	3536934	2110895	1550880	695236	318500
Faridabad	**2020**	312711	275273	357617	166721	57959
	2021	359870	286148	306888	140728	45724
	2022	397426	274584	186429	50478	12487
India	**2020**	424743	1102014	371396	5369144	4852246
	2021	420940	2075004	1838458	4010243	3744335
	2022	1675999	3841281	2602188	821483	3092548
Indore	**2022**	1482479	872927	661969	361346	165145
Uddepy	**2020**	1299320	979208	1304744	597155	195930
	2021	1068843	772889	809171	365173	114830
	2022	1104680	4325123	4627767	1754789	718495
Ujjain	**2020**	369639	285890	332701	136803	51954
	2021	335540	232055	220982	94301	37148
	2022	2432691	1420593	1158731	514700	216550

```
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

Region,Year

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

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

114 2022 8 Clients Ujjain Actuals NaN NaN

```
df2
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Visitors
Aurangabad	**2020**	92083	63071	88109	29622	12612
	2021	102100	71711	81673	31773	12801
	2022	82786	54252	39666	14656	5088
Dehradun	**2020**	402599	295615	311845	111753	50258
	2021	352787	250456	214565	78363	34827
	2022	3536934	2110895	1550880	695236	318500
Faridabad	**2020**	312711	275273	357617	166721	57959
	2021	359870	286148	306888	140728	45724
	2022	397426	274584	186429	50478	12487
India	**2020**	424743	1102014	371396	5369144	4852246
	2021	420940	2075004	1838458	4010243	3744335
	2022	1675999	3841281	2602188	821483	3092548

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

139	2022	7	Clients	India	Actuals	250201.0	NaN

```
df1
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Visitors
Aurangabad	**2020**	92083	63071	88109	29622	12612
	2021	102100	71711	81673	31773	12801
	2022	82786	54252	39666	14656	5088
Dehradun	**2020**	402599	295615	311845	111753	50258
	2021	352787	250456	214565	78363	34827
	2022	3536934	2110895	1550880	695236	318500

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

	2022	397426	274584	186429	50478	12487

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

```
df1
```

Region	Year	Lv1_Visitors	Lv2_Visitors	Lv3_Visitors	Lv4_Visitors	Lv5_Visitors
Aurangabad	**2020**	92083	63071	88109	29622	12612
	2021	102100	71711	81673	31773	12801
	2022	82786	54252	39666	14656	5088
Dehradun	**2020**	402599	295615	311845	111753	50258

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

Faridabad	**2020**	312711	275273	357617	166721	57959

Part 3: Prediction

	2021	420940	2075004	1838458	4010243	3744335

```
df
```

	Year	Month	Segment	Region	Value Type	Lv5_Visitors	Lv4_Visitors	Lv3_Vis
0	2020	12	Clients	India	Actuals	255723	0	
1	2020	12	Clients	India	Actuals	0	180125	
2	2020	12	Clients	India	Actuals	74768	0	
3	2021	12	Clients	India	Actuals	216178	0	
4	2021	12	Clients	India	Actuals	0	180821	
5	2021	12	Clients	India	Actuals	69345	0	
6	2022	12	Clients	Uddepy	Actuals	0	0	
7	2022	12	Clients	Uddepy	Actuals	0	0	
8	2022	12	Clients	Dehradun	Actuals	0	0	
9	2022	12	Clients	Ujjain	Actuals	0	0	
10	2022	12	Clients	Dehradun	Actuals	0	0	
11	2022	12	Clients	Indore	Actuals	0	0	
12	2022	12	Clients	Uddepy	Actuals	0	118950	
13	2022	12	Clients	Dehradun	Actuals	0	0	
14	2022	12	Clients	Ujjain	Actuals	0	0	
15	2022	12	Clients	Ujjain	Actuals	0	0	
16	2022	12	Clients	India	Actuals	75211	0	
17	2022	12	Clients	Indore	Actuals	0	0	
18	2022	12	Clients	Indore	Actuals	0	0	
19	2022	12	Clients	Dehradun	Actuals	0	53146	
20	2022	12	Clients	Ujjain	Actuals	0	37404	
21	2022	12	Clients	Indore	Actuals	0	30773	
22	2022	12	Clients	Uddepy	Actuals	14855	0	
23	2022	12	Clients	Dehradun	Actuals	7880	0	
24	2022	12	Clients	Ujjain	Actuals	5632	0	
25	2022	12	Clients	Indore	Actuals	4335	0	
26	2020	11	Clients	India	Actuals	272017	0	
27	2020	11	Clients	India	Actuals	0	187646	
28	2020	11	Clients	India	Actuals	81680	0	
29	2021	11	Clients	India	Actuals	243224	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))
```

35	2022	11	Clients	Dehradun	Actuals	0	0

```
df['DATE'] = DATE
```

```
df
```

	Year	Month	Segment	Region	Value Type	Lv5_Visitors	Lv4_Visitors	Lv3_Vis
0	2020	12	Clients	India	Actuals	255723	0	
1	2020	12	Clients	India	Actuals	0	180125	
2	2020	12	Clients	India	Actuals	74768	0	
3	2021	12	Clients	India	Actuals	216178	0	
4	2021	12	Clients	India	Actuals	0	180821	
5	2021	12	Clients	India	Actuals	69345	0	
6	2022	12	Clients	Uddepy	Actuals	0	0	
7	2022	12	Clients	Uddepy	Actuals	0	0	
8	2022	12	Clients	Dehradun	Actuals	0	0	
9	2022	12	Clients	Ujjain	Actuals	0	0	
10	2022	12	Clients	Dehradun	Actuals	0	0	
11	2022	12	Clients	Indore	Actuals	0	0	
12	2022	12	Clients	Uddepy	Actuals	0	118950	
13	2022	12	Clients	Dehradun	Actuals	0	0	
14	2022	12	Clients	Ujjain	Actuals	0	0	
15	2022	12	Clients	Ujjain	Actuals	0	0	
16	2022	12	Clients	India	Actuals	75211	0	
17	2022	12	Clients	Indore	Actuals	0	0	
18	2022	12	Clients	Indore	Actuals	0	0	

19	2022	12	Clients	Dehradun	Actuals	0	53146

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

20	2022	12	Clients	Ujjain	Actuals	0	37404

```
df.head()
```

	Segment	Region	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors	Lv2_Visitors	Lv1_Visitor
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
```

31	2021	11	Clients	India	Actuals	78573	0

```
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-170-74aa2f970831>:1: FutureWarning: Treating datetime data as categorica
  df.describe(include = 'all')
```

	Segment	Region	Lv5_Visitors	Lv4_Visitors	Lv3_Visitors	Lv2_Visitors
count	1325	1325	1325.000000	1325.000000	1325.000000	1325.000000

```
# 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
```

78	2020	9	Clients	India	Actuals	239246	0

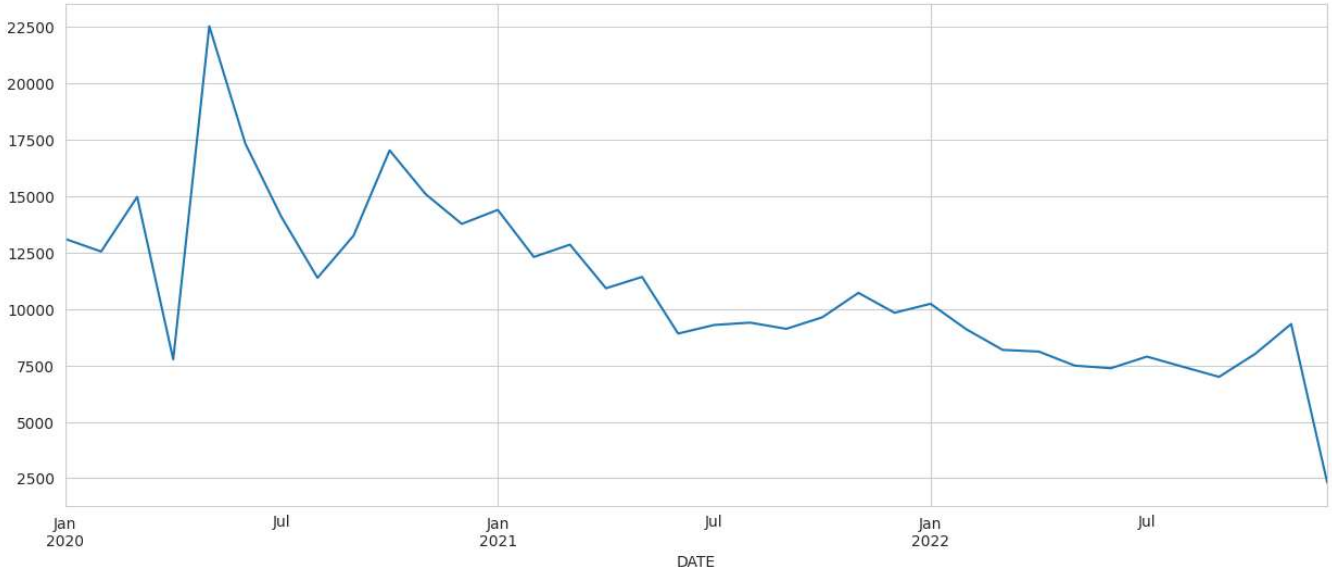
```
df = df.set_index('DATE')
df.index
```

```
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()
```




```
import statsmodels.api as sm
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8

decomposition = sm.tsa.seasonal_decompose(y, model = 'additive')
fig = decomposition.plot()
plt.show()
```


```
import itertools

# set the typical ranges for p, d, q
p = d = q = range(0, 2)

#take all possible combination for p, d and q
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

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

```
# Using Grid Search find the optimal set of parameters that yields the best performance
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y, order = param, seasonal_order = param_seasonal
            result = mod.fit()
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, result.aic))
        except:
            continue
```