Assignment No. 2

Step-I

Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them

In [17]:

```
import numpy as np
import pandas as pd
```

In [4]:

```
df = pd.read_csv("Academic_performace.csv")
```

In [6]:

```
df
```

Out[6]:

	Sno	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Sem
0	1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
1	2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
2	3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
3	4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
4	5	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
...	...	...	...	...	...	...	...	...	
475	476	F	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	
476	477	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
477	478	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
478	479	F	Jordan	Jordan	MiddleSchool	G-08	A	History	
479	480	F	Jordan	Jordan	MiddleSchool	G-08	A	History	

480 rows × 18 columns

In [7]:

```
df.head()
```

Out[7]:

	Sno	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	R
0	1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
1	2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
2	3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
3	4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
4	5	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	

In [8]:

```
df.tail()
```

Out[8]:

	Sno	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Sem
475	476	F	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	
476	477	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
477	478	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
478	479	F	Jordan	Jordan	MiddleSchool	G-08	A	History	
479	480	F	Jordan	Jordan	MiddleSchool	G-08	A	History	

In [9]:

```
df.describe()
```

Out[9]:

	Sno	raisedhands	VisITedResources	AnnouncementsView	Discussion
count	480.000000	480.000000	480.000000	480.000000	478.000000
mean	240.500000	46.775000	54.797917	38.462500	43.278243
std	138.708327	30.779223	33.080007	30.095579	27.646238
min	1.000000	0.000000	0.000000	0.000000	1.000000
25%	120.750000	15.750000	20.000000	14.000000	20.000000
50%	240.500000	50.000000	65.000000	33.000000	39.000000
75%	360.250000	75.000000	84.000000	58.000000	70.000000
max	480.000000	100.000000	99.000000	350.000000	99.000000

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Sno                       480 non-null    int64  
 1   gender                    480 non-null    object 
 2   NationalITy               480 non-null    object 
 3   PlaceofBirth              480 non-null    object 
 4   StageID                   480 non-null    object 
 5   GradeID                   480 non-null    object 
 6   SectionID                 480 non-null    object 
 7   Topic                     480 non-null    object 
 8   Semester                  480 non-null    object 
 9   Relation                  480 non-null    object 
 10  raisedhands               480 non-null    int64  
 11  VisITedResources          480 non-null    int64  
 12  AnnouncementsView         480 non-null    int64  
 13  Discussion                478 non-null    float64
 14  ParentAnsweringSurvey     480 non-null    object 
 15  ParentschoolSatisfaction  480 non-null    object 
 16  StudentAbsenceDays        480 non-null    object 
 17  Class                     480 non-null    object 
dtypes: float64(1), int64(4), object(13)
memory usage: 67.6+ KB
```

In [11]:

```
df.shape
```

Out[11]:

```
(480, 18)
```

In [12]:

```
df.isnull().any().any()
```

Out[12]:

```
True
```

In [13]:

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

Out[13]:

```
Sno                       0
gender                    0
NationalITy               0
PlaceofBirth              0
StageID                   0
GradeID                   0
SectionID                 0
Topic                     0
Semester                  0
Relation                  0
raisedhands               0
VisITedResources          0
AnnouncementsView         0
Discussion                2
ParentAnsweringSurvey     0
ParentschoolSatisfaction  0
StudentAbsenceDays        0
Class                     0
dtype: int64
```

In [14]:

```
avg_val = df["Discussion"].astype("float").mean()
avg_val
```

Out[14]:

```
43.27824267782427
```

In [15]:

```
df["Discussion"].replace(np.NaN, avg_val, inplace=True)
```

In [16]:

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

Out[16]:

```
Sno                       0
gender                    0
NationalITy               0
PlaceofBirth              0
StageID                   0
GradeID                   0
SectionID                 0
Topic                     0
Semester                  0
Relation                  0
raisedhands               0
VisITedResources          0
AnnouncementsView         0
Discussion                0
ParentAnsweringSurvey     0
ParentschoolSatisfaction  0
StudentAbsenceDays        0
Class                     0
dtype: int64
```

Step-II

Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

In [20]:

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

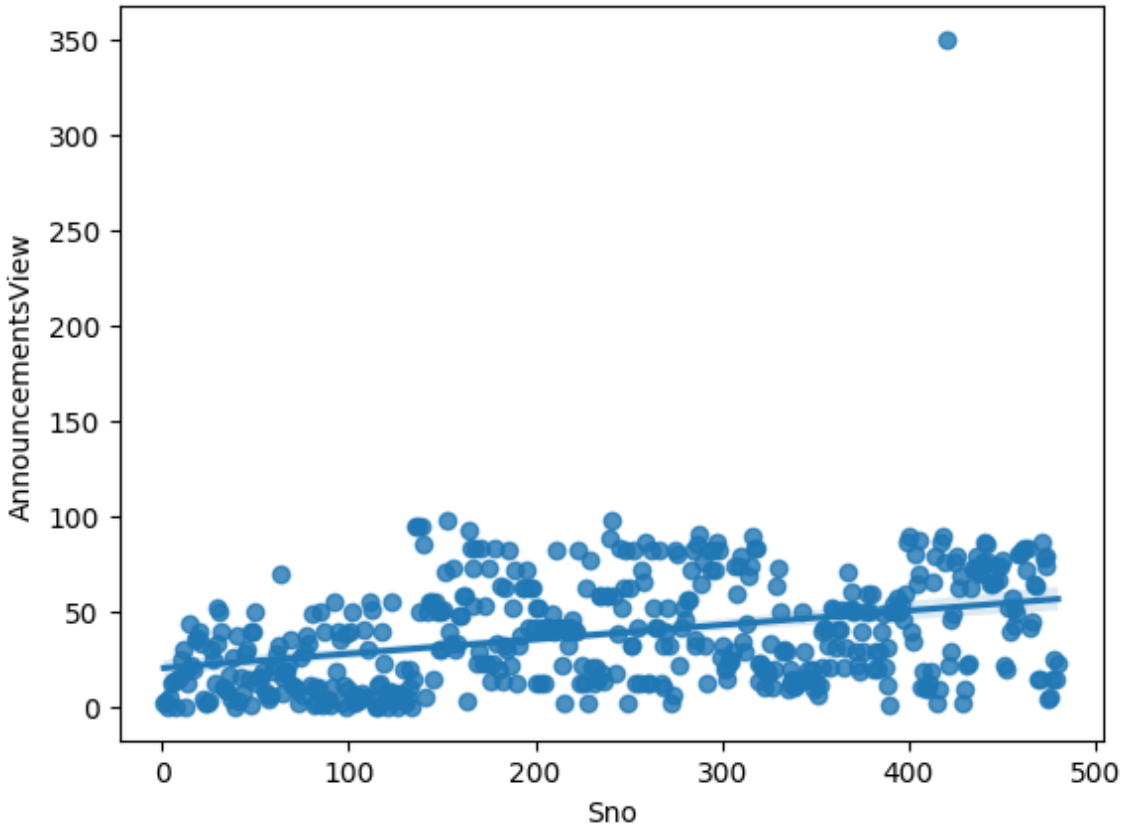
Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

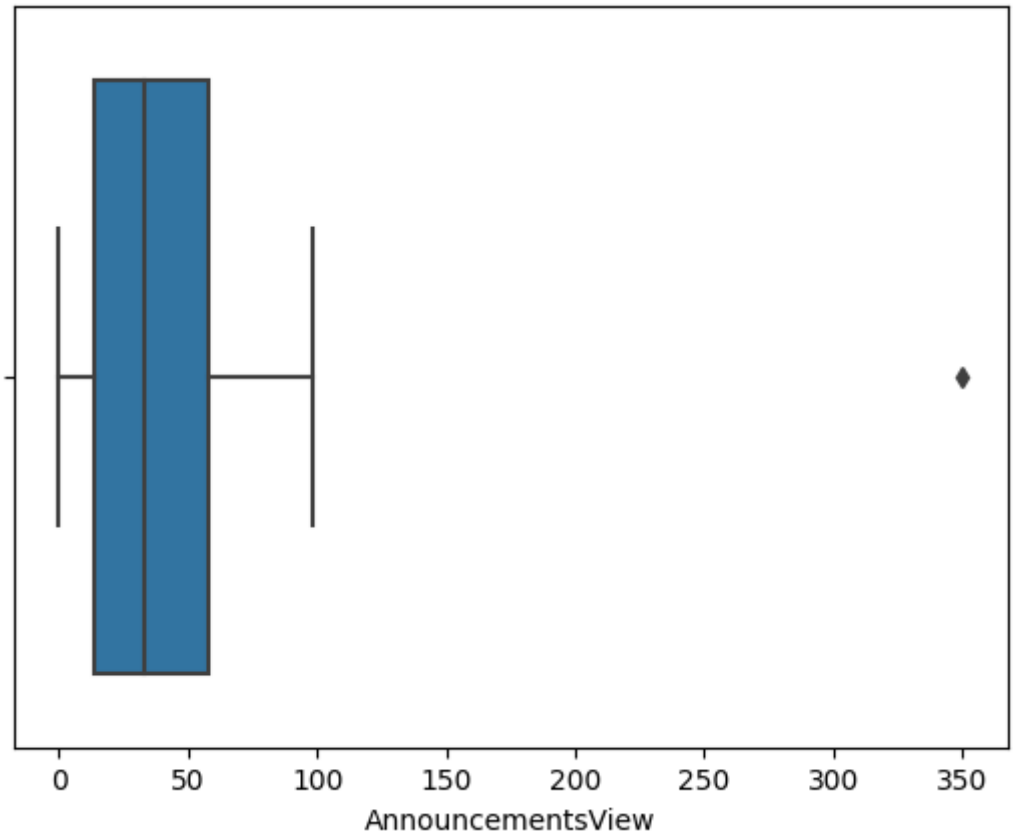
In [45]:

```
sns.regplot(x='Sno', y='AnnouncementsView', data=df)
plt.show()
```



In [47]:

```
sns.boxplot(x=df['AnnouncementsView'])
plt.show()
```



In [57]:

```
z = np.abs(stats.zscore(df['AnnouncementsView']))
print(z)
```

```
0      1.212821
1      1.179559
2      1.279345
3      1.113034
4      0.880199
         ...
475    1.113034
476    0.813675
477    0.447792
478    0.813675
479    0.514316
Name: AnnouncementsView, Length: 480, dtype: float64
```

In [51]:

```
threshold = 3
print(np.where(z > 3))
```

```
(array([419], dtype=int64),)
```

In [60]:

```
z[419]
```

Out[60]:

```
10.3624031636167
```

The standard z score is calculated by dividing the difference from the mean by the standard deviation. The modified z score is calculated from the mean absolute deviation (MeanAD) or median absolute deviation (MAD). These values must be multiplied by a constant to approximate the standard deviation.

Step-III

Apply data transformations on at least one of the variables

In [62]:

```
df1 = pd.DataFrame({ 'Income': [15000, 1800, 120000, 10000],
                     'Age': [25, 18, 42, 51],
                     'Department': ['HR','Legal','Marketing','Management']})
```

In [63]:

```
df1
```

Out[63]:

	Income	Age	Department
0	15000	25	HR
1	1800	18	Legal
2	120000	42	Marketing
3	10000	51	Management

In [65]:

```
df1_scaled = df1.copy()
col_names = ['Income', 'Age']
features = df1_scaled[col_names]
```

In [67]:

```
features
```

Out[67]:

	Income	Age
0	15000	25
1	1800	18
2	120000	42
3	10000	51

In [70]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df1_scaled[col_names] = scaler.fit_transform(features.values)
```

scikit-learn

- scikit-learn is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface.
- Important features of scikit-learn:
  - Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
  - Accessible to everybody and reusable in various contexts.
  - Built on the top of NumPy, SciPy, and matplotlib.
  - Open source, commercially usable – BSD license.

In [71]:

```
print(df1_scaled[col_names])
```

```
     Income       Age
0  0.111675  0.212121
1  0.000000  0.000000
2  1.000000  0.727273
3  0.069374  1.000000
```

```
There is another way of data scaling, where the minimum of feature is made equal to zero
and the maximum of feature equal to one. MinMax Scaler shrinks the data within the given
range, usually of 0 to 1. It transforms data by scaling features to a given range. It
scales the values to a specific value range without changing the shape of the original
distribution.

The MinMax scaling is done using:

x_std = (x - x.min(axis=0)) / (x.max(axis=0) - x.min(axis=0))

x_scaled = x_std * (max - min) + min

Where,

min, max = feature_range
x.min(axis=0) : Minimum feature value
x.max(axis=0):Maximum feature value
Sklearn preprocessing defines MinMaxScaler() method to achieve this.
```

In [ ]:

```

```