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

```
df = pd.read_csv ("/content/student_performance_prediction.csv")
```

```
# configuring display options for dataframe
pd.set_option('display.max_columns', None) # display max columns
pd.set_option('display.max_rows', None) # display max rows
```

```
df.sample(5)
```

	Student ID	Study Hours per Week	Attendance Rate	Previous Grades	Participation in Extracurricular Activities	Parent Education Level	Passed
29460	S29461	11.8	101.8	95.9	No	Bachelor	No
189	S00190	2.4	82.0	60.4	Yes	Associate	Yes
6084	S06085	9.5	50.4	28.2	No	Bachelor	Yes
25215	S25216	13.4	82.1	62.3	No	NaN	Yes

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 7 columns):
 #   Column                                       Non-Null Count  Dtype  
---  ------                                       --------------  -----  
 0   Student ID                                   40000 non-null  object 
 1   Study Hours per Week                         38005 non-null  float64
 2   Attendance Rate                              38008 non-null  float64
 3   Previous Grades                              38006 non-null  float64
 4   Participation in Extracurricular Activities  38000 non-null  object 
 5   Parent Education Level                       38000 non-null  object 
 6   Passed                                       38000 non-null  object 
dtypes: float64(3), object(4)
memory usage: 2.1+ MB
```

```
print ("DataSet Shape:", df.shape) # shape
print ("Total Size:", df.size) # size
```

```
DataSet Shape: (40000, 7)
Total Size: 280000
```

```
df.columns # columns list
```

```
Index(['Student ID', 'Study Hours per Week', 'Attendance Rate',
       'Previous Grades', 'Participation in Extracurricular Activities',
       'Parent Education Level', 'Passed'],
      dtype='object')
```

Handling Missing Values

```
# total no. of missing values
total_missin_vals = df.isna().sum().sum()
print (f"{total_missin_vals} values are missing in the dataset.")
```

```
11981 values are missing in the dataset.
```

```
# total % of missing values
missin_vals_percent = round((total_missin_vals/df.shape[0])*100)
print (f"{missin_vals_percent}% of values are missing in the dataset.")
```

```
30% of values are missing in the dataset.
```

```
cols_wid_missin_vals = df.columns[df.isna().any()].tolist()
print (f"List of columns with missing values: {cols_wid_missin_vals}")
```

```
List of columns with missing values: ['Study Hours per Week', 'Attendance Rate', 'Previous Grades', 'Participati
```

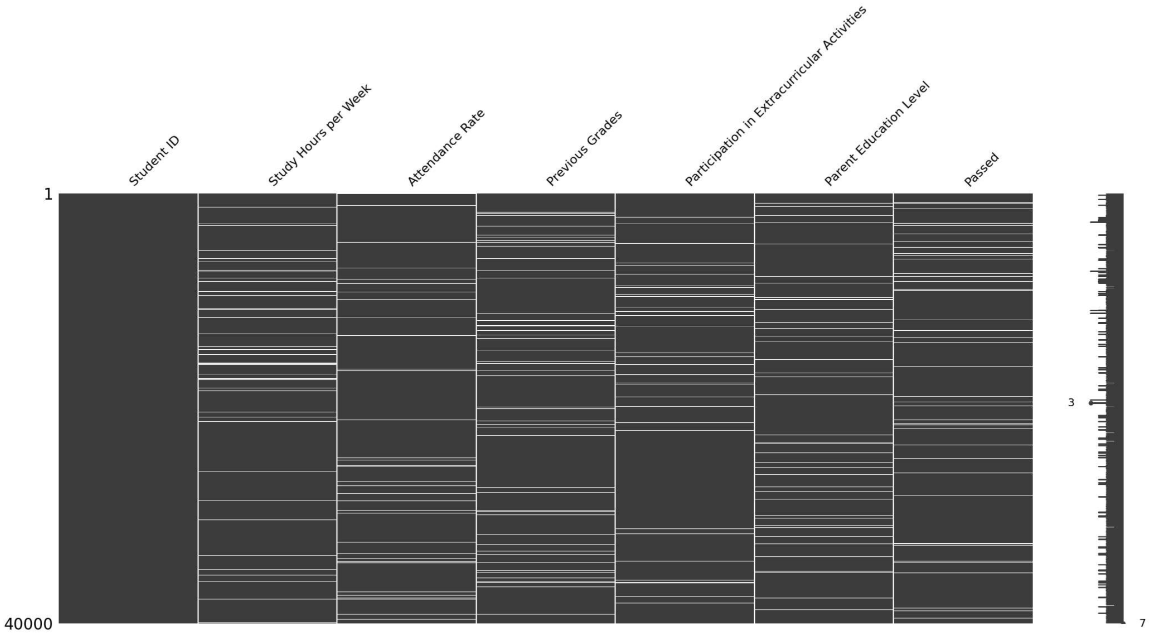
```
missin_vals_per_col = df [['Study Hours per Week', 'Attendance Rate', 'Previous Grades', 'Participation in Extraci
missin_vals_per_col
```

```
                                                 0
                 Study Hours per Week         1995
                      Attendance Rate         1992
                      Previous Grades         1994
Participation in Extracurricular Activities   2000
                Parent Education Level        2000
                               Passed         2000

dtype: int64
```

```
percent_missin_vals_per_col = (missin_vals_per_col/df.shape[0])*100
percent_missin_vals_per_col
```

```
                                                  0
                 Study Hours per Week         4.9875
                      Attendance Rate         4.9800
                      Previous Grades         4.9850
Participation in Extracurricular Activities   5.0000
                Parent Education Level        5.0000
                               Passed         5.0000

dtype: float64
```

```
# visual for missing values in the data
import missingno as msno
msno.matrix(df)
plt.show()
```



```
df[['Study Hours per Week', 'Attendance Rate', 'Previous Grades']].sample(25)
```

```
       Study Hours per Week  Attendance Rate  Previous Grades
39166                  10.6             65.3             76.5
23873                  15.0             45.5             49.6
1832                   14.8             95.7             83.0
38374                  14.0             72.4             74.6
3806                   10.2            103.4             43.0
26264                   7.8             75.9             77.7
31900                   3.2             45.5             54.5
15301                  15.4             82.5             36.7
29081                  12.3             99.4             72.1
12264                   2.5             64.4             46.1
13674                   8.9             94.2             54.6
15202                   3.5             74.4             53.4
13680                   NaN             74.7             60.5
13717                  14.1             78.4             84.5
24313                  17.8             81.5             52.8
27653                   6.6             93.0             73.6
13989                   3.4            120.4             57.7
36324                   8.3             62.4             38.8
3214                   13.7             80.8             69.2
877                    11.1            107.6             51.8
27401                  14.5             91.0             63.3
33272                  16.9             81.6              NaN
19766                  13.5             59.4             98.4
13918                  13.9             77.0             33.4
2620                   13.5             83.3             36.8
```

```python
# mean interpolation for numeric columns

# Calculate the mean for each column
mean_values = df[['Study Hours per Week', 'Attendance Rate', 'Previous Grades']].mean()

# Fill missing values with the calculated means
df.fillna(mean_values, inplace=True)
```

```python
df [['Study Hours per Week', 'Attendance Rate', 'Previous Grades']].isna().sum()
```

```
                      0
Study Hours per Week  0
Attendance Rate       0
Previous Grades       0

dtype: int64
```

```python
df [['Participation in Extracurricular Activities', 'Parent Education Level', 'Passed']].sample(25)
```

	Participation in Extracurricular Activities	Parent Education Level	Passed
23912	No	Associate	Yes
39156	Yes	Master	Yes
27364	NaN	Doctorate	No
18109	NaN	High School	No
1119	No	Bachelor	No
231	Yes	High School	No
10062	Yes	Doctorate	Yes
8388	Yes	Bachelor	Yes
4188	Yes	Associate	Yes
34485	No	Master	No
23753	No	Master	No
25530	NaN	High School	No
28458	No	High School	Yes
9388	Yes	High School	No
7863	Yes	Bachelor	No
8835	No	Bachelor	No
15635	No	Doctorate	Yes
37991	Yes	NaN	Yes
22483	No	Associate	Yes
35464	Yes	Doctorate	No
11579	Yes	Bachelor	NaN
2506	Yes	Bachelor	No
3186	No	NaN	Yes
39602	No	High School	No
30069	No	NaN	Yes

```python
# mode interpolation for categorical columns

# 1. Select categorical columns
categorical_columns = df.select_dtypes(include="object").columns

# 2. Iterate over categorical columns
for column in categorical_columns:
    # 3. Calculate the mode for the current column
    mode_value = df[column].mode()[0]

    # 4. Fill missing values with the mode
    df[column] = df[column].fillna(mode_value)
```

```python
df [['Participation in Extracurricular Activities', 'Parent Education Level', 'Passed']].isna().sum()
```

	0
Participation in Extracurricular Activities	0
Parent Education Level	0
Passed	0

dtype: int64

```python
df.columns
```

```
Index(['Student ID', 'Study Hours per Week', 'Attendance Rate',
       'Previous Grades', 'Participation in Extracurricular Activities',
       'Parent Education Level', 'Passed'],
      dtype='object')
```

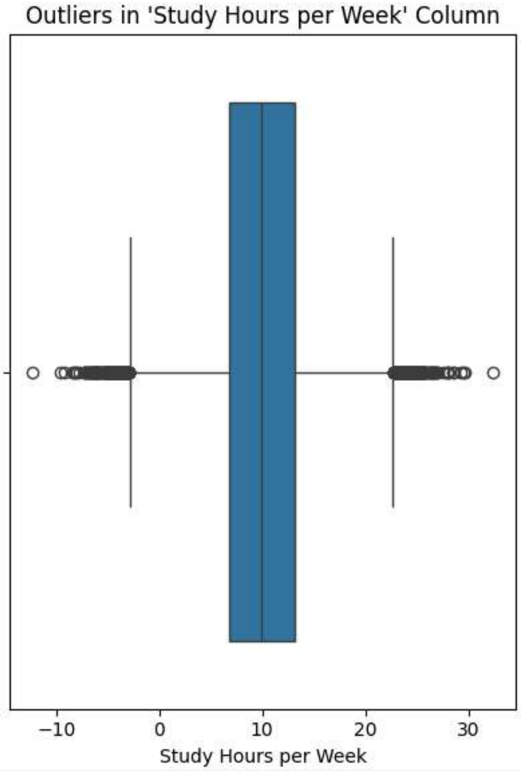
Handling Outliers & Skewness

col 1: 'Student ID'

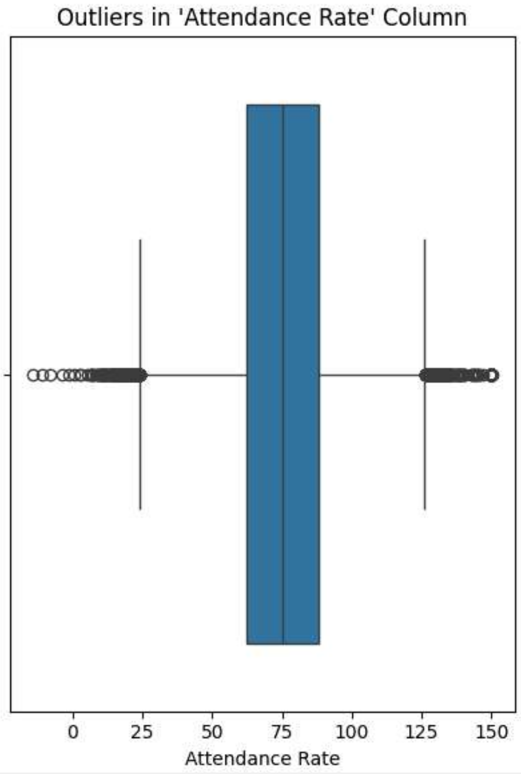
```python
for col in ['Study Hours per Week', 'Attendance Rate', 'Previous Grades']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"Outliers in {col}:")
    print(outliers)
```

```
39516      No
39557     Yes
39571      No
39576      No
39678      No
39732     Yes
39797      No
39851     Yes
39890     Yes
```

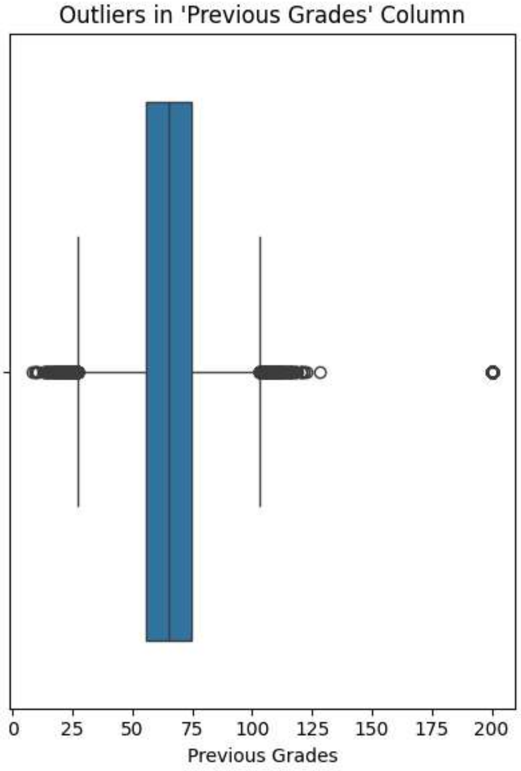
```python
# Create a box plot for each numeric column
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
sns.boxplot(x=df['Study Hours per Week'])
plt.title("Outliers in 'Study Hours per Week' Column")

plt.tight_layout()
plt.show()
```



```python
# Create a box plot for each numeric column
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
sns.boxplot(x=df['Attendance Rate'])
plt.title("Outliers in 'Attendance Rate' Column")

plt.tight_layout()
plt.show()
```



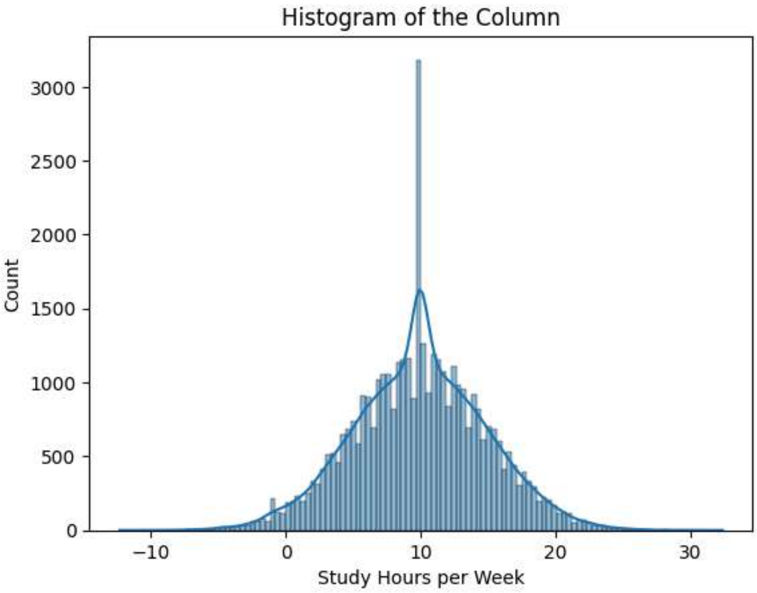
```
# Create a box plot for each numeric column
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
sns.boxplot(x=df['Previous Grades'])
plt.title("Outliers in 'Previous Grades' Column")

plt.tight_layout()
plt.show()
```

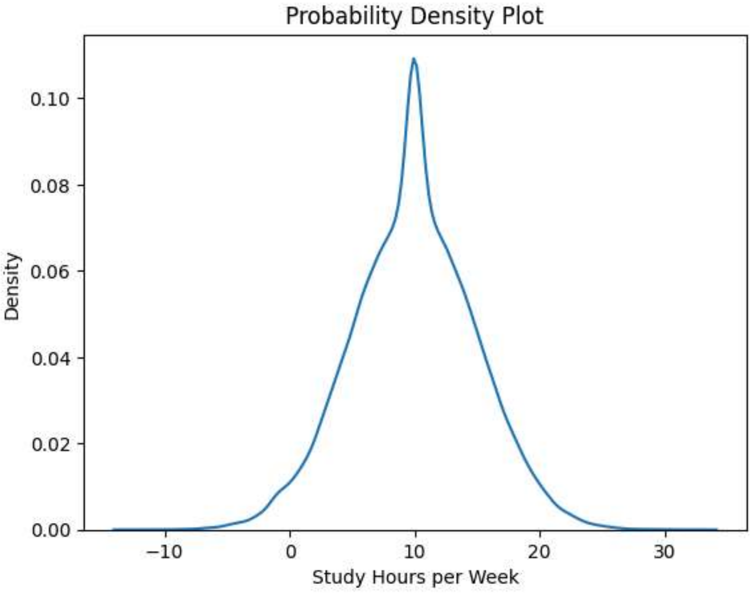


```
skewness = df[['Study Hours per Week', 'Attendance Rate', 'Previous Grades']].skew()
print("Skewness in Columns -\n", skewness)
```

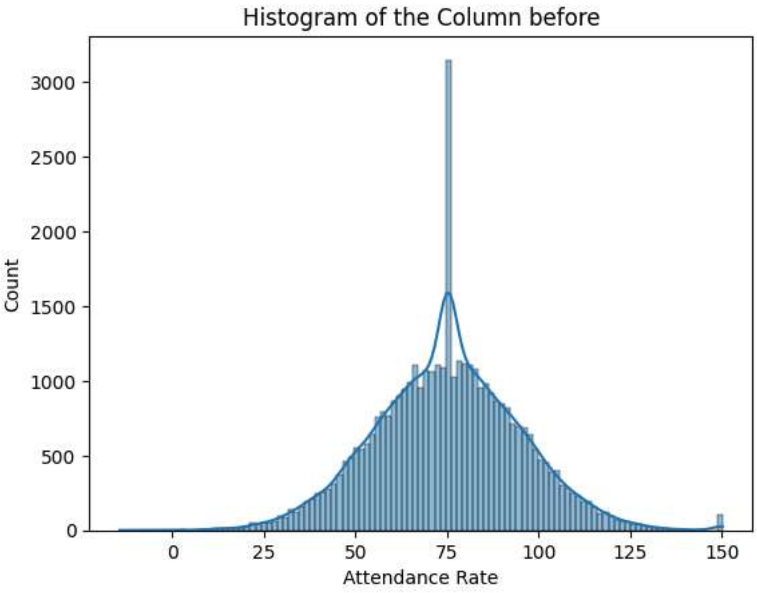
```
Skewness in Columns -
 Study Hours per Week   -0.011992
Attendance Rate         0.091591
Previous Grades         1.409301
dtype: float64
```

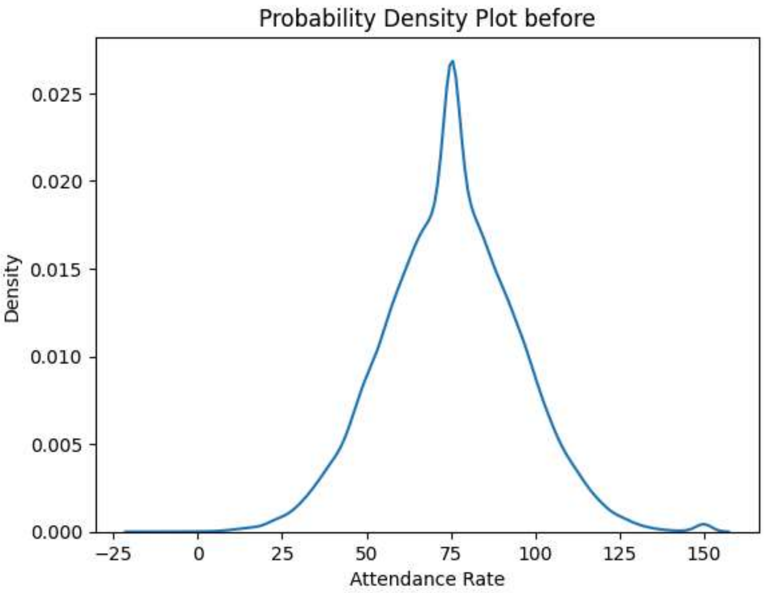
```
# Histogram
sns.histplot(df['Study Hours per Week'], kde=True)
plt.title('Histogram of the Column')
plt.show()

# Probability Density Plot
sns.kdeplot(df['Study Hours per Week'])
plt.title('Probability Density Plot')
plt.show()
```


Histogram of the Column

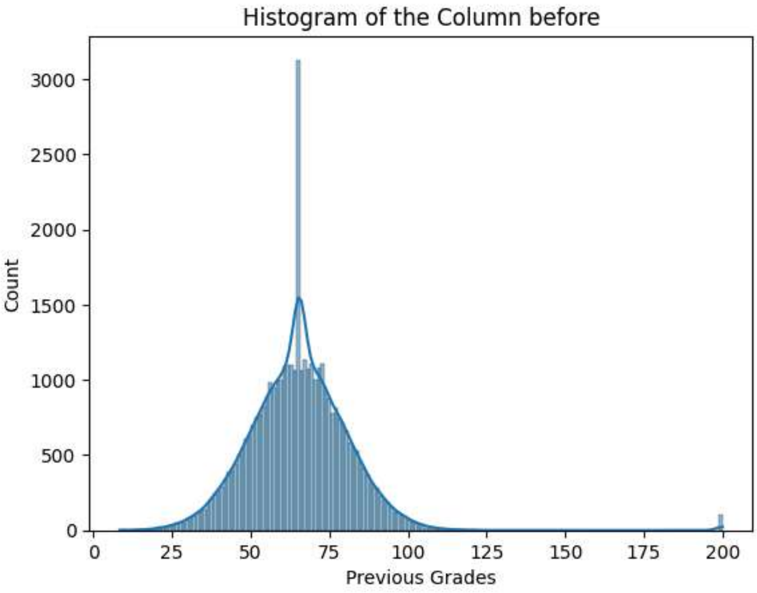

Probability Density Plot

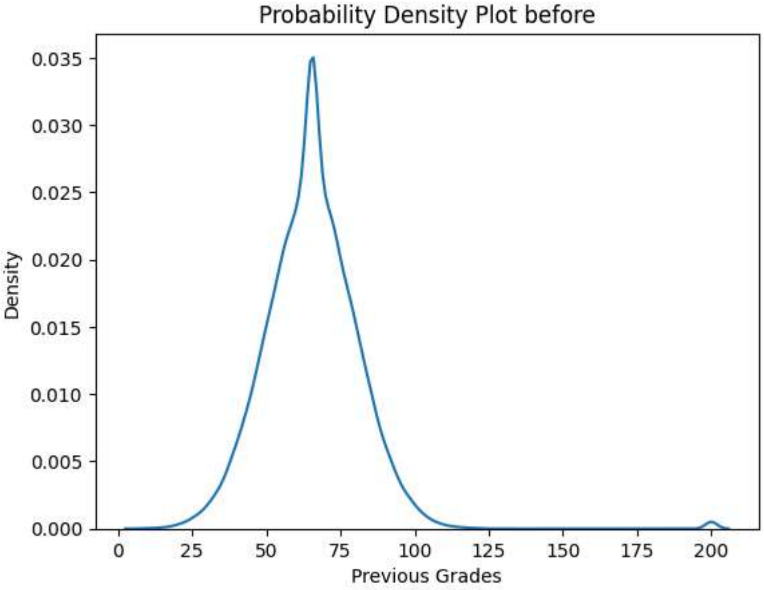
```
# Histogram
sns.histplot(df['Attendance Rate'], kde=True)
plt.title('Histogram of the Column before')
plt.show()

# Probability Density Plot
sns.kdeplot(df['Attendance Rate'])
plt.title('Probability Density Plot before')
plt.show()
```





```
# Histogram
sns.histplot(df['Previous Grades'], kde=True)
plt.title('Histogram of the Column before')
plt.show()

# Probability Density Plot
sns.kdeplot(df['Previous Grades'])
plt.title('Probability Density Plot before')
plt.show()
```



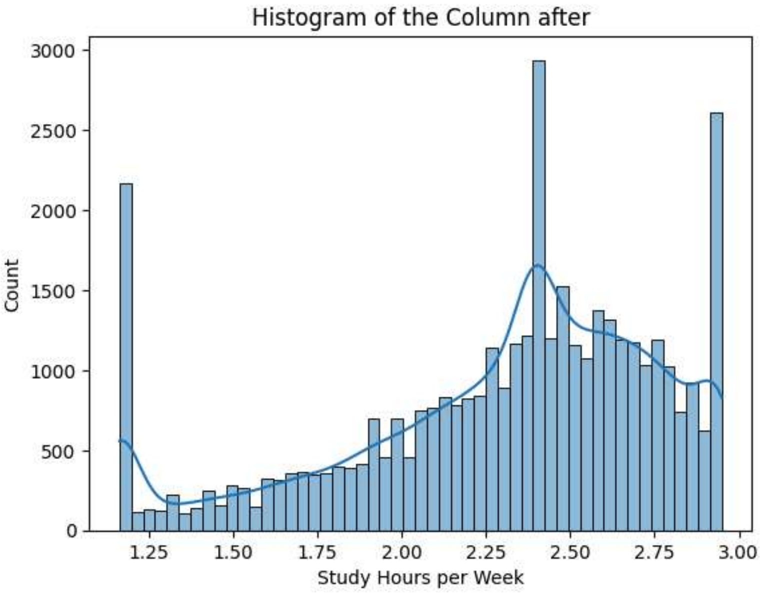


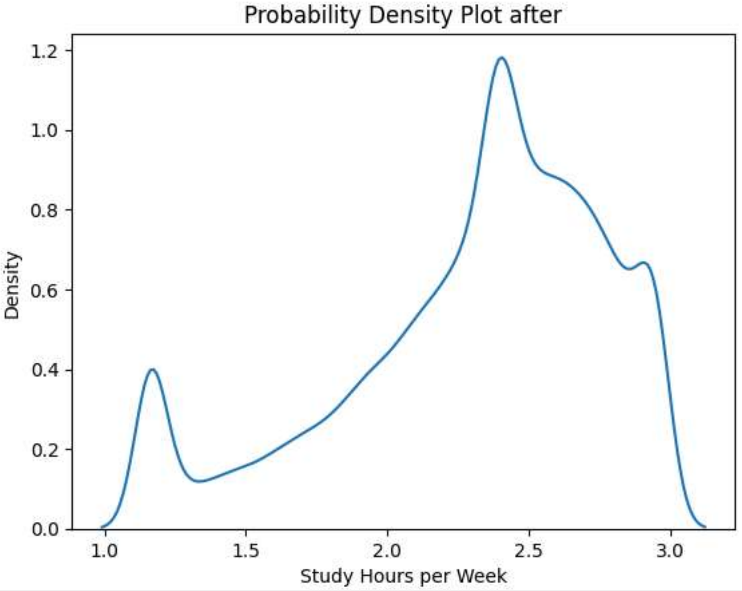
```
# Handling Negative Skewness and Outliers

# Handle negative values (if any):
df['Study Hours per Week'] = df['Study Hours per Week'].abs()

# Log transformation
df['Study Hours per Week'] = np.log1p(df['Study Hours per Week'])

# Define thresholds (adjust as needed)
lower_bound = df['Study Hours per Week'].quantile(0.05)
upper_bound = df['Study Hours per Week'].quantile(0.95)

# Cap outliers
df['Study Hours per Week'] = np.clip(df['Study Hours per Week'], lower_bound, upper_bound)
```

```
# Winsorization
lower_bound = df['Attendance Rate'].quantile(0.05)
upper_bound = df['Attendance Rate'].quantile(0.95)
df['Attendance Rate'] = np.clip(df['Attendance Rate'], lower_bound, upper_bound)

# Or, Box-Cox Transformation
from scipy.stats import boxcox
df['Attendance Rate'], _ = boxcox(df['Attendance Rate'] + 1)  # Add 1 to handle zero values
```

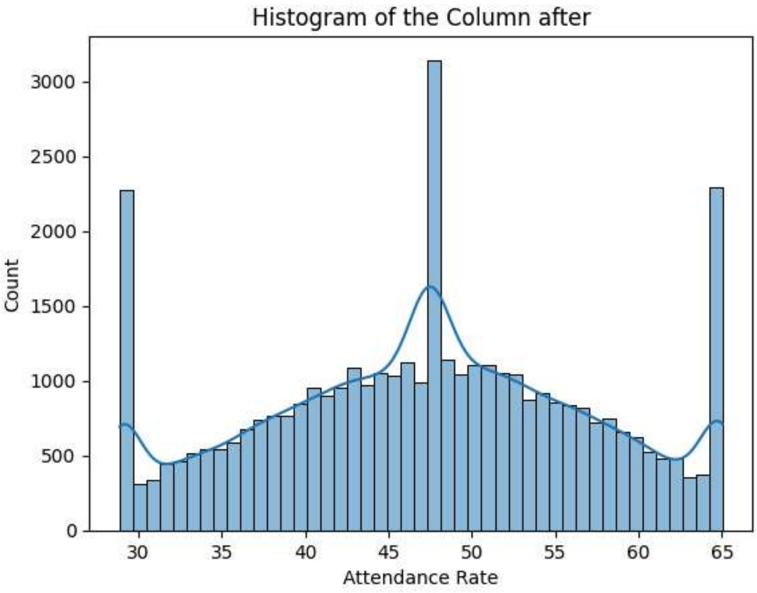
```
# Winsorization
lower_bound = df['Previous Grades'].quantile(0.05)
upper_bound = df['Previous Grades'].quantile(0.95)
df['Previous Grades'] = np.clip(df['Previous Grades'], lower_bound, upper_bound)

# Or, Box-Cox Transformation
from scipy.stats import boxcox
df['Previous Grades'], _ = boxcox(df['Previous Grades'] + 1)  # Add 1 to handle zero values
```

```
# Histogram
sns.histplot(df['Study Hours per Week'], kde=True)
plt.title('Histogram of the Column after')
plt.show()

# Probability Density Plot
sns.kdeplot(df['Study Hours per Week'])
plt.title('Probability Density Plot after')
plt.show()
```

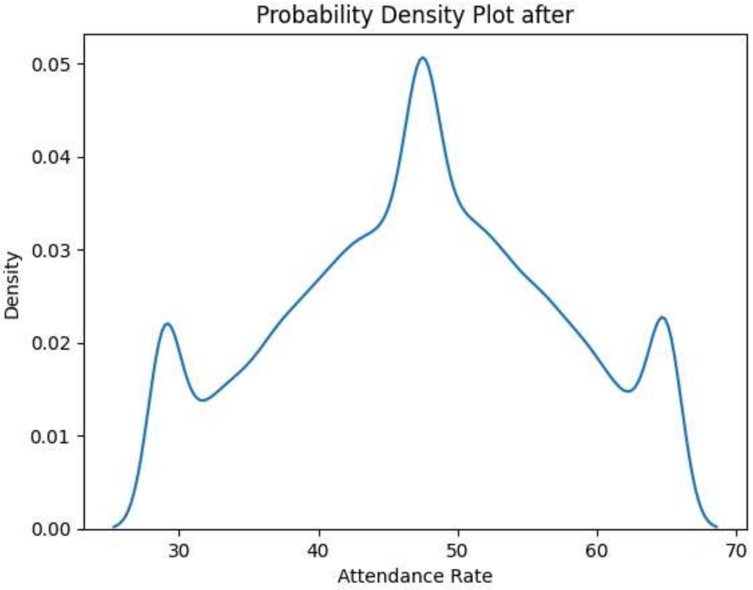




```
# Histogram
sns.histplot(df['Attendance Rate'], kde=True)
plt.title('Histogram of the Column after')
plt.show()

# Probability Density Plot
sns.kdeplot(df['Attendance Rate'])
plt.title('Probability Density Plot after')
plt.show()
```

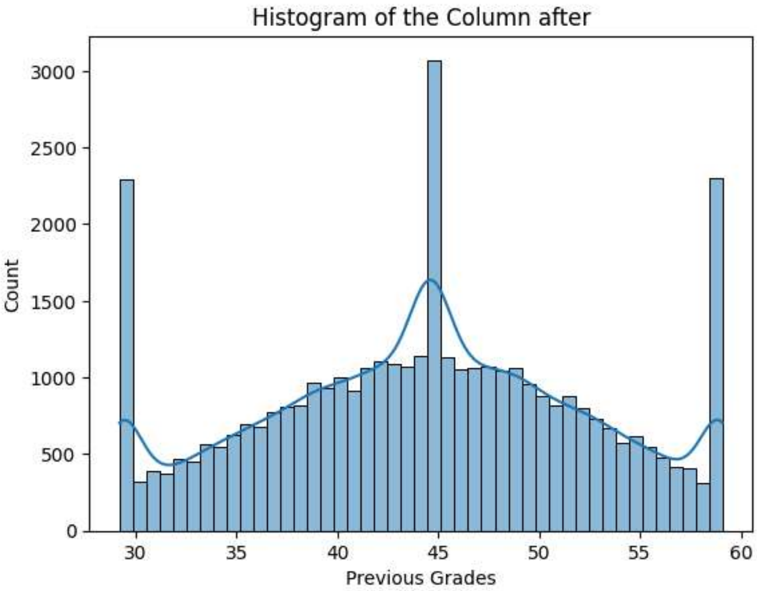


Histogram of the Column after

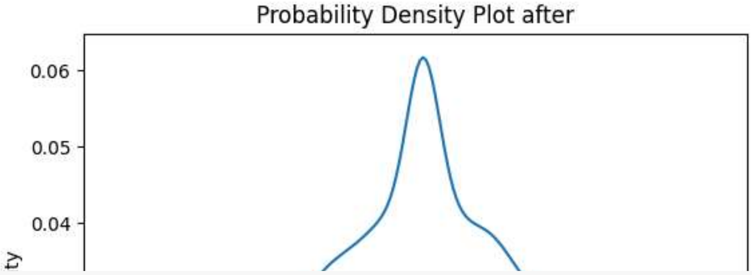


Probability Density Plot after

```
# Histogram
sns.histplot(df['Previous Grades'], kde=True)
plt.title('Histogram of the Column after')
plt.show()

# Probability Density Plot
sns.kdeplot(df['Previous Grades'])
plt.title('Probability Density Plot after')
plt.show()
```



Histogram of the Column after



Probability Density Plot after

```
df.sample(50)
```