Step 1: Import the Required Libraries and Load Dataset into PandaFrame

[174]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from scipy.stats import skew
from scipy.stats import boxcox
data = pd.read_csv(r'C:\Users\UNKNOWN_CODER\DSDBA\Assign2\academic_performace.csv')
```

[194]:
```
data.head()
```

[194]:

	StudentID	Name	Age	Gender	MathScore	ScienceScore	EnglishScore	Attendance
0	1	Alice	20.0	Female	85.000000	88.0	82.0	95.0
1	2	Bob	21.5	Male	80.235294	75.0	85.0	80.0
2	3	Charlie	22.0	Female	90.000000	92.0	90.0	85.0
3	4	David	21.0	Male	95.000000	80.0	92.0	NaN
4	5	Eve	20.0	Unknown	80.000000	79.0	78.0	88.0

[176]:
```
df = pd.DataFrame(data)
```

[177]:
```
#missing values
missing_values = df.isnull().sum()
print(missing_values)
```

```
StudentID       0
Name            0
Age             2
Gender          0
MathScore       3
ScienceScore    2
EnglishScore    1
Attendance      2
dtype: int64
```

[178]:
```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['MathScore'].fillna(df['MathScore'].mean(), inplace=True)
df['EnglishScore'].fillna(df['EnglishScore'].mean(), inplace=True)

df['Attendance'] = pd.to_numeric(df['Attendance'], errors='coerce')
df['Attendance'].fillna(df['Attendance'].mean(), inplace=True)
df['Gender'] = df['Gender'].replace('Unknown', df['Gender'].mode()[0])
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['ScienceScore'].fillna(df['ScienceScore'].mean(), inplace=True)
```

[179]:
```
df
```

[179]:

	StudentID	Name	Age	Gender	MathScore	ScienceScore	EnglishScore	Attendance
0	1	Alice	20.0	Female	85.000000	88.000000	82.000000	95.000000
1	2	Bob	21.5	Male	80.235294	75.000000	85.000000	80.000000
2	3	Charlie	22.0	Female	90.000000	92.000000	90.000000	85.000000
3	4	David	21.0	Male	95.000000	80.000000	92.000000	87.333333
4	5	Eve	20.0	Male	80.000000	79.000000	78.000000	88.000000
5	6	Frank	21.5	Male	110.000000	81.722222	89.000000	92.000000
6	7	Grace	22.0	Female	85.000000	86.000000	94.000000	87.333333
7	8	Hannah	21.0	Male	-5.000000	91.000000	85.000000	95.000000
8	9	Ivy	19.0	Female	88.000000	84.000000	88.000000	80.000000

[180]:
```
#missing values
missing_values = df.isnull().sum()
print(missing_values)
```

```
StudentID       0
Name            0
Age             0
Gender          0
MathScore       0
ScienceScore    0
EnglishScore    0
Attendance      0
dtype: int64
```

Step 2: Scan for outliers in numeric variables

[181]:
```
def detect_outliers(df,column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]
```

[182]:
```
outliers_math = detect_outliers(df,'MathScore')
outliers_math
```

[182]:

	StudentID	Name	Age	Gender	MathScore	ScienceScore	EnglishScore	Attendance
5	6	Frank	21.5	Male	110.0	81.722222	89.0	92.0
7	8	Hannah	21.0	Male	-5.0	91.000000	85.0	95.0

[183]:
```
outliers_science = detect_outliers(df,'ScienceScore')
outliers_science
```

[183]:

	StudentID	Name	Age	Gender	MathScore	ScienceScore	EnglishScore	Attendance

[184]:
```
outliers_english = detect_outliers(df,'EnglishScore')
outliers_english
```

[184]:

	StudentID	Name	Age	Gender	MathScore	ScienceScore	EnglishScore	Attendance

[185]:
```
# Handle outliers using IQR method
for col in ['MathScore']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
    df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])

print("\nDataset After Handling Outliers:")
print(df)
```

```
Dataset After Handling Outliers:
    StudentID     Name   Age  Gender   MathScore  ScienceScore  EnglishScore  \
0           1    Alice  20.0  Female   85.000000     88.000000     82.000000
1           2      Bob  21.5    Male   80.235294     75.000000     85.000000
2           3  Charlie  22.0  Female   90.000000     92.000000     90.000000
3           4    David  21.0    Male   95.000000     80.000000     92.000000
4           5      Eve  20.0    Male   80.000000     79.000000     78.000000
5           6    Frank  21.5    Male  108.000000     81.722222     89.000000
6           7    Grace  22.0  Female   85.000000     86.000000     94.000000
7           8   Hannah  21.0    Male   62.000000     91.000000     85.000000
8           9      Ivy  19.0  Female   88.000000     84.000000     88.000000
9          10     Jack  25.0    Male   80.235294     78.000000     77.000000
10         11     Lily  23.0  Female   93.000000     85.000000     90.000000
11         12     Mike  18.0    Male   70.000000     79.000000     65.000000
12         13    Naomi  24.0  Female   65.000000     78.000000     80.000000
13         14    Oscar  20.0    Male   95.000000     87.000000     82.000000
14         15     Paul  21.0    Male   80.235294     80.000000     85.000000
15         16    Quinn  22.0  Female   98.000000     81.722222     93.000000
16         17   Rachel  23.0  Female   88.000000     82.000000     83.263158
17         18    Steve  26.0    Male   77.000000     70.000000     80.000000
18         19     Tina  21.0  Female   65.000000     77.000000     72.000000
19         20    Uriel  22.0    Male   85.000000     80.000000     75.000000

    Attendance
0    95.000000
1    80.000000
2    85.000000
3    87.333333
4    88.000000
5    92.000000
6    87.333333
7    95.000000
8    80.000000
9   100.000000
10   97.000000
11   75.000000
12   90.000000
13   98.000000
14   85.000000
15   92.000000
16   80.000000
17   60.000000
18   85.000000
19   95.000000
```

[186]:
```
outliers_math = detect_outliers(df,'MathScore')
outliers_math
```

[186]:

	StudentID	Name	Age	Gender	MathScore	ScienceScore	EnglishScore	Attendance

Step 3 : Data Transformation

[193]:
```
attendance_df = df
# Min-Max Scaling
scaler = MinMaxScaler()
attendance_df['MinMax_Scaled'] = scaler.fit_transform(attendance_df[['Attendance']])

# Standardization
standard_scaler = StandardScaler()
attendance_df['Standardized'] = standard_scaler.fit_transform(attendance_df[['Attendance']])

# Log Transformation
attendance_df['Log_Transformed'] = np.log(attendance_df['Attendance'])

# Square Root Transformation
attendance_df['Sqrt_Transformed'] = np.sqrt(attendance_df['Attendance'])

# Box-Cox Transformation
attendance_df['BoxCox_Transformed'], _ = boxcox(attendance_df['Attendance'])

# Check skewness
print("Skewness before:", skew(attendance_df['Attendance']))
print("Skewness after Log Transformation:", skew(attendance_df['Log_Transformed']))
print("Skewness after Sqrt Transformation:", skew(attendance_df['Sqrt_Transformed']))
print("Skewness after Box-Cox Transformation:", skew(attendance_df['BoxCox_Transformed']))

# Plot histograms
attendance_df[['Attendance', 'Log_Transformed', 'Sqrt_Transformed', 'BoxCox_Transformed']].hist(bins=10, figsize=(12, 8))
plt.show()
```

```
Skewness before: -1.1639932056541822
Skewness after Log Transformation: -1.5785021951155833
Skewness after Sqrt Transformation: -1.3664730732913304
Skewness after Box-Cox Transformation: -0.1114939769650313
```