```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set a random seed for reproducibility
np.random.seed(42)

# 1. Create the "Academic Performance" dataset
data = {
'Student_ID': range(1, 101),
'Math_Score': np.random.randint(50, 100, size=100),
'English_Score': np.random.randint(40, 95, size=100),
'Science_Score': np.random.randint(55, 98, size=100),
'Attendance_Percentage': np.random.uniform(70, 100, size=100),
'Study_Hours_Per_Day': np.random.uniform(1, 6, size=100),
}

academic_df = pd.DataFrame(data)

# Introduce missing values and inconsistencies for demonstration
academic_df.loc[10:20, 'Math_Score'] = np.nan
academic_df.loc[30:40, 'English_Score'] = np.nan
academic_df.loc[50:60, 'Science_Score'] = np.nan
academic_df.loc[70:80, 'Attendance_Percentage'] = np.nan

# Display first few rows of the dataset
print("First few rows of the Academic Performance dataset:")
print(academic_df.head())
```

```
# 1. Scan all variables for missing values and inconsistencies
# Use mean imputation for missing values and replace any negative values with NaN
academic_df.fillna(academic_df.mean(), inplace=True)
academic_df[academic_df < 0] = np.nan

# Display the updated dataset after handling missing values and inconsistencies
print("\nUpdated dataset after handling missing values and inconsistencies:")
print(academic_df.head())

# 2. Scan all numeric variables for outliers
# Use Z-score to identify and handle outliers
numeric_vars = ['Math_Score', 'English_Score', 'Science_Score', 'Attendance_Percentage',
'Study_Hours_Per_Day']

z_scores = (academic_df[numeric_vars] - academic_df[numeric_vars].mean()) /
academic_df[numeric_vars].std()
outliers = (z_scores > 3) | (z_scores < -3)

# Replace outliers with NaN
academic_df[outliers] = np.nan

# Display the dataset after handling outliers
print("\nDataset after handling outliers:")
print(academic_df.head())

# 3. Apply data transformations
# Log transformation on 'Study_Hours_Per_Day' to decrease skewness
academic_df['Log_Study_Hours'] = np.log1p(academic_df['Study_Hours_Per_Day'])

# Display the dataset after the log transformation
print("\nDataset after log transformation:")
```

```
print(academic_df.head())

# Visualize the distribution before and after the transformation
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(academic_df['Study_Hours_Per_Day'], kde=True)
plt.title('Study_Hours_Per_Day Distribution')

plt.subplot(1, 2, 2)
sns.histplot(academic_df['Log_Study_Hours'], kde=True)
plt.title('Log_Study_Hours Distribution')
```