Practical No-2

Date of Conduction: **Date of Checking:**

Data Wrangling II

Create an “Academic performance” dataset of students and perform the following operations using Python.

1. 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.

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

3. Apply data transformations on at least one of the variables.

The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly.

Python Code:

```
# 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
```

```python
# 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')

plt.show()
```

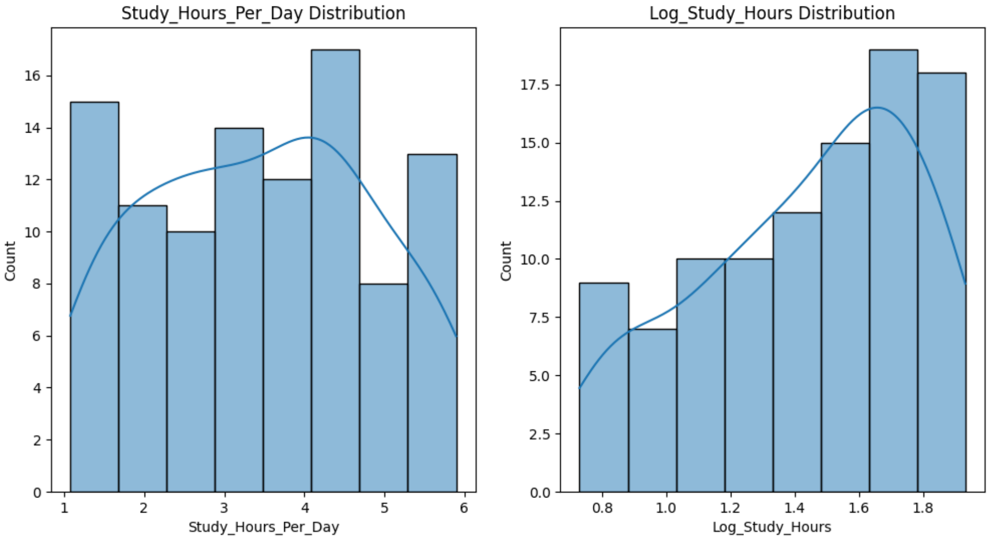
Explanation:

- The code starts by creating a sample "Academic Performance" dataset with variables such as Math_Score, English_Score, Science_Score, Attendance_Percentage, and Study_Hours_Per_Day.
- Some missing values and inconsistencies are introduced for demonstration purposes.
- Missing values and inconsistencies are handled using mean imputation for missing values and replacing negative values with NaN.
- Outliers are identified using Z-scores, and extreme values are replaced with NaN.
- A log transformation is applied to the 'Study_Hours_Per_Day' variable to decrease skewness and convert the distribution into a more normal shape.
- The code includes visualizations to compare the distribution before and after the log transformation.

Output:

```
"C:\Users\Ram Kumar Solanki\PycharmProjects\pythonProject\venv\Scripts\python.exe"
"C:\Users\Ram Kumar Solanki\PycharmProjects\MBA_BFS\main.py"

First few rows of the Academic Performance dataset:

   Student_ID  Math_Score  ...  Attendance_Percentage  Study_Hours_Per_Day
0           1        88.0  ...              81.168483             5.847684
1           2        78.0  ...              98.204003             4.572976
2           3        64.0  ...              99.209915             1.205338
3           4        92.0  ...              78.517629             2.994105
4           5        57.0  ...              79.160916             3.167604

[5 rows x 6 columns]


Updated dataset after handling missing values and inconsistencies:

   Student_ID  Math_Score  ...  Attendance_Percentage  Study_Hours_Per_Day
0           1        88.0  ...              81.168483             5.847684
1           2        78.0  ...              98.204003             4.572976
2           3        64.0  ...              99.209915             1.205338
3           4        92.0  ...              78.517629             2.994105
4           5        57.0  ...              79.160916             3.167604
```

```
[5 rows x 6 columns]
Dataset after handling outliers:
   Student_ID  Math_Score  ...  Attendance_Percentage  Study_Hours_Per_Day
0           1        88.0  ...              81.168483             5.847684
1           2        78.0  ...              98.204003             4.572976
2           3        64.0  ...              99.209915             1.205338
3           4        92.0  ...              78.517629             2.994105
4           5        57.0  ...              79.160916             3.167604
[5 rows x 6 columns]
Dataset after log transformation:
   Student_ID  Math_Score  ...  Study_Hours_Per_Day  Log_Study_Hours
0           1        88.0  ...             5.847684         1.923911
1           2        78.0  ...             4.572976         1.717929
2           3        64.0  ...             1.205338         0.790881
3           4        92.0  ...             2.994105         1.384819
4           5        57.0  ...             3.167604         1.427341

[5 rows x 7 columns]
```


Study_Hours_Per_Day Distribution
Count
Study_Hours_Per_Day
Log_Study_Hours Distribution
Count
Log_Study_Hours