Practical Experiment: Data Cleaning and Preprocessing

Objective:

To implement a Python script to read data and perform data cleaning and preprocessing steps.

Steps:

1. **Read Data**:
   - Use the Pandas library to read a CSV file.
2. **Handle Missing Values**:
   - Identify and handle missing values in the dataset.
3. **Remove Duplicates**:
   - Remove duplicate rows from the dataset.
4. **Transform Data**:
   - Apply transformations to the data using a function or mapping.
5. **Replace Values**:
   - Replace specific values in the dataset.
6. **Detect and Filter Outliers**:
   - Detect and filter outliers in the dataset.

Sample Data:

Let's use the 'Iris' dataset for this experiment. You can download it from here.

Python Script:

python

```
import pandas as pd
import numpy as np

# Step 1: Read Data
url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data'
columns = ['sepal_length', 'sepal_width', 'petal_length',
'petal_width', 'class']
data = pd.read_csv(url, header=None, names=columns)

# Display the first few rows of the dataset
print("Original Data:\n", data.head())

# Step 2: Handle Missing Values
# Introduce some missing values for demonstration
data.loc[5:10, 'sepal_length'] = np.nan

# Fill missing values with the mean of the column
data['sepal_length'].fillna(data['sepal_length'].mean(),
inplace=True)
```

```
# Display the dataset after handling missing values
print("\nData After Handling Missing Values:\n", data.head(12))

# Step 3: Remove Duplicates
# Introduce a duplicate row for demonstration
data = data.append(data.iloc[0], ignore_index=True)

# Remove duplicate rows
data.drop_duplicates(inplace=True)

# Display the dataset after removing duplicates
print("\nData After Removing Duplicates:\n", data.head())

# Step 4: Transform Data
# Apply a transformation to the 'sepal_length' column (e.g., scaling
by a factor of 10)
data['sepal_length'] = data['sepal_length'] * 10

# Display the dataset after transformation
print("\nData After Transformation:\n", data.head())

# Step 5: Replace Values
# Replace a specific class label 'Iris-setosa' with 'Setosa'
data['class'] = data['class'].replace('Iris-setosa', 'Setosa')

# Display the dataset after replacing values
print("\nData After Replacing Values:\n", data.head())

# Step 6: Detect and Filter Outliers
# Calculate the Z-scores to detect outliers
data['sepal_length_zscore'] = (data['sepal_length'] -
data['sepal_length'].mean()) / data['sepal_length'].std()

# Filter out rows where the Z-score is greater than 3 or less than -
3
data = data[(data['sepal_length_zscore'] < 3) &
(data['sepal_length_zscore'] > -3)]

# Display the dataset after filtering outliers
print("\nData After Filtering Outliers:\n", data.head())

# Drop the Z-score column for final cleaned data
data.drop(columns=['sepal_length_zscore'], inplace=True)

# Display the final cleaned dataset
print("\nFinal Cleaned Data:\n", data.head())
```

Explanation:

1. **Read Data**: The script reads the Iris dataset from a URL and loads it into a Pandas DataFrame.

2. **Handle Missing Values**: Missing values are introduced for demonstration and then handled by filling them with the mean of the column.
3. **Remove Duplicates**: A duplicate row is introduced for demonstration and then removed using the `drop_duplicates()` method.
4. **Transform Data**: The 'sepal_length' column is transformed by scaling its values by a factor of 10.
5. **Replace Values**: The class label 'Iris-setosa' is replaced with 'Setosa' for demonstration purposes.
6. **Detect and Filter Outliers**: Outliers are detected using Z-scores and filtered out based on a threshold of 3.