

```

1 # This Python 3 environment comes with many helpful analytics libraries installed
2 # It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
3 # For example, here's several helpful packages to load
4
5 import numpy as np # linear algebra
6 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
7
8 # Input data files are available in the read-only "../input/" directory
9 # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
10
11 import os
12 for dirname, _, filenames in os.walk('/kaggle/input'):
13     for filename in filenames:
14         print(os.path.join(dirname, filename))
15
16 # You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save"
17 # You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

```

```

1 import pandas as pd
2 import numpy as np
3 # Create a synthetic dataset
4 data = {
5     'Student_ID': range(1, 101),
6     'Gender': np.random.choice(['Male', 'Female'], 100),
7     'Age': np.random.randint(18, 25, 100),
8     'Math_Score': np.random.randint(0, 101, 100),
9     'English_Score': np.random.randint(0, 101, 100),
10    'Attendance': np.random.uniform(60, 100, 100),
11    'Final_Grade': np.random.randint(50, 100, 100),
12    'Family_Income': np.random.randint(20, 100, 100)
13 }
14 # Create a DataFrame
15 df = pd.DataFrame(data)
16 # Introduce missing values (for demonstration purposes)
17 df.loc[:10, 'Math_Score'] = np.nan # Missing values in Math_Score
18 df.loc[:5, 'Family_Income'] = np.nan # Missing values in
19
20 # Introduce inconsistencies (age > 30, attendance > 100)
21 df.loc[5, 'Age'] = 32
22 df.loc[15, 'Attendance'] = 105
23 # Introduce outliers
24 df.loc[20, 'Math_Score'] = 200 # Outlier in Math_Score
25 df.loc[30, 'English_Score'] = -5 # Outlier in English_Score
26
27 # Display first few rows of the dataset
28 # Check for missing values
29 missing_values = df.isnull().sum()
30 import pandas as pd
31 import numpy as np
32 # Create a synthetic dataset
33 data = {
34     'Student_ID': range(1, 101),
35     'Gender': np.random.choice(['Male', 'Female'], 100),
36     'Age': np.random.randint(18, 25, 100),
37     'Math_Score': np.random.randint(0, 101, 100),
38     'English_Score': np.random.randint(0, 101, 100),
39     'Attendance': np.random.uniform(60, 100, 100),
40     'Final_Grade': np.random.randint(50, 100, 100),
41     'Family_Income': np.random.randint(20, 100, 100)
42 }
43 # Create a DataFrame
44 df = pd.DataFrame(data)
45 # Introduce missing values (for demonstration purposes)
46 df.loc[:10, 'Math_Score'] = np.nan # Missing values in Math_Score
47 df.loc[:5, 'Family_Income'] = np.nan # Missing values in
48
49 # Introduce inconsistencies (age > 30, attendance > 100)
50 df.loc[5, 'Age'] = 32
51 df.loc[15, 'Attendance'] = 105
52 # Introduce outliers
53 df.loc[20, 'Math_Score'] = 200 # Outlier in Math_Score
54 df.loc[30, 'English_Score'] = -5 # Outlier in English_Score
55
56 # Display first few rows of the dataset
57 # Check for missing values

```

```

58 missing_values = df.isnull().sum()
59 # Impute missing values for numeric variables with mean
60 df['Math_Score'] = df['Math_Score'].fillna(df['Math_Score'].mean())
61 df['Family_Income'] = df['Family_Income'].fillna(df['Family_Income'].mean())
62 # For categorical variable, impute with mode (most frequent value)
63 df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
64 # Fix inconsistencies
65 df['Age'] = df['Age'].apply(lambda x: x if x <= 30 else 25) # Cap age
66
67 df['Attendance'] = df['Attendance'].apply(lambda x: x if x <= 100 else 100) # Cap attendance to 100
68 # Check the dataset after cleaning
69 df.head()
70 df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
71 # Fix inconsistencies
72 df['Age'] = df['Age'].apply(lambda x: x if x <= 30 else 25) # Cap age
73
74 df['Attendance'] = df['Attendance'].apply(lambda x: x if x <= 100 else
75 100) # Cap attendance to 100
76 # Check the dataset after cleaning
77 df.head()

1 # Function to detect outliers using IQR
2 def detect_outliers(df, column):
3     Q1 = df[column].quantile(0.25)
4     Q3 = df[column].quantile(0.75)
5     IQR = Q3 - Q1
6     lower_bound = Q1 - 1.5 * IQR
7     upper_bound = Q3 + 1.5 * IQR
8     return df[(df[column] < lower_bound) | (df[column] > upper_bound)]
9
10 # Check for outliers in numeric columns
11 outliers_math = detect_outliers(df, 'Math_Score')
12 outliers_english = detect_outliers(df, 'English_Score')
13 # Display outliers
14 outliers_math, outliers_english
15
16 # Handle outliers by capping them
17 def cap_outliers(df, column):
18     Q1 = df[column].quantile(0.25)
19     Q3 = df[column].quantile(0.75)
20     IQR = Q3 - Q1
21     lower_bound = Q1 - 1.5 * IQR
22     upper_bound = Q3 + 1.5 * IQR
23     df[column] = df[column].apply(lambda x: min(max(x, lower_bound), upper_bound))
24
25 # Cap outliers in Math_Score and English_Score
26 cap_outliers(df, 'Math_Score')
27 cap_outliers(df, 'English_Score')
28 # Check for outliers after capping
29 detect_outliers(df, 'Math_Score'), detect_outliers(df, 'English_Score')

1 # Before capping
2 print(df[['Math_Score', 'English_Score']].describe())
3
4 # Apply capping (your cap_outliers function)
5
6 # After capping
7 print(df[['Math_Score', 'English_Score']].describe())

1 import numpy as np
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 # Apply log transformation to Family_Income
6 df['Log_Family_Income'] = np.log(df['Family_Income'])
7
8 # Check the distribution before and after transformation
9 plt.figure(figsize=(12, 6))
10
11 # Before transformation
12 plt.subplot(1, 2, 1)
13 sns.histplot(df['Family_Income'], kde=True, color='blue')
14 plt.title('Family Income (Original)')
15
16 # After transformation

```

```
17 plt.subplot(1, 2, 2)
18 sns.histplot(df['Log_Family_Income'], kde=True, color='green')
19 plt.title('Log Transformed Family Income')
20
21 plt.tight_layout() # Adjusts subplot params for a tight layout
22 plt.show()
```