Canvas: Data Wrangling II on Academic Performance Dataset

I. Dataset Creation: Simulating Real-World Data Issues

Code Head: `import pandas as pd`
Code Tail: `df = pd.DataFrame(data)`

Explanation:
We start by creating a simulated dataset named **`Academic performance`** for 100 students. It includes these columns:

- `Student_ID`: Unique ID for each student.
- `Math`, `Physics`, `English`: Subject scores.
- `Attendance`: Value between 0.7 to 1.0 (like a percentage).
- `GPA`: Grade Point Average between 2.5 and 4.

We also introduce **intentional problems** in the dataset to mimic real-world data:

- One missing value in `Math`.
- An unrealistic `Physics` score (200).
- A letter `'A'` instead of a number in the `English` score.

II. Finding Missing and Incorrect Values

Code Head: `missing_values = df.isnull().sum()`
Code Tail: `print("Inconsistencies in English_Score:", inconsistent_data)`

Explanation:
We check each column for:

- **Missing values** using `.isnull().sum()`.
- **Invalid data types**, especially in the `English` column, by trying to convert all values to numbers using `pd.to_numeric()` and counting how many fail.

This helps us identify where data cleaning is required.

III. Visualizing the Data: Spotting Outliers

Code Head: `plt.figure(figsize=(12, 6))`
Code Tail: `plt.title('Boxplot of Numeric Variables')`

Explanation:
We use a **boxplot** to visually find outliers in numeric columns. An outlier will show as a dot or line far away from the box. Here, the Physics score of **200** will clearly appear as an outlier because valid scores should be between 0–100.

IV. Cleaning the Data: Fixing Errors and Filling Gaps

Code Head:

df['Math'] = df['Math'].fillna(df['Math'].mean()).astype(float)

df['Physics'] = df['Physics'].apply(lambda x: np.nan if x > 100 else x)

df['English'] = pd.to_numeric(df['English'], errors='coerce')

Code Tail: `df.dropna(inplace=True)`

Explanation:
We fix the issues found earlier:

- **Math**: Fill missing value using the column's average score.

- **Physics**: Change any score > 100 (like 200) to NaN because it's invalid.
- **English**: Convert all values to numbers; `'A'` becomes NaN automatically.

Finally, we **drop all rows** that still have any missing values after cleaning.

V. Second Pass: Detecting More Subtle Outliers

Code Head: `z_scores = np.abs(zscore(df[numeric_cols]))`
Code Tail: `outlier_indices = np.where(z_scores > threshold)[0]`

Explanation:
We now apply a **statistical method** using the **Z-score**:

- Z-score tells how far a value is from the mean.
- We define an outlier as any value with a Z-score above 3 (i.e., it's very far from normal).
- We find all row indices where this happens.

This helps find **hidden outliers** that weren't obvious in the boxplot.

VI. Final Cleanup: Removing Outliers and Transforming Data

Code Head:

df_cleaned = df.drop(outlier_indices)

df_cleaned['GPA'] = np.log1p(df_cleaned['GPA'])

Code Tail:

sns.histplot(df_cleaned['GPA'], kde=True)

plt.title('Transformed GPA Distribution')

```
plt.show()
```

Explanation:

- We remove all rows that had outliers based on Z-scores.
- We apply **log1p transformation** (log(x + 1)) to the `GPA` column.
  - This makes the GPA values less skewed (i.e., more normally distributed).
  - It helps when you're using machine learning or statistics later.

Finally, we show a **histogram** of the transformed GPA to see its new shape.

VII. Summary and Justification

Code Head: *(Conceptual section)*
Code Tail: *(Conceptual section)*

Explanation:
Why we did what we did:

- **Missing values** were filled (mean for Math) or removed (for other columns).
- **Incorrect values** (like 'A', or 200) were fixed or removed.
- **Outliers** were handled using a statistical method (Z-score).
- **Data transformation** helped in normalizing GPA, useful for further modeling or analysis.

This makes the dataset **clean**, **consistent**, and **ready for analysis or machine learning**.