Introduction to Data Wrangling in Python Using Pandas

Data wrangling is a critical step in the data analysis process, especially in psychological research. Before drawing insights from data, researchers must clean, structure, and transform raw data into a usable format. In this tutorial, we will use Python's Pandas library to perform common data wrangling tasks on a simulated psychology dataset.

We will cover:

1. Loading and inspecting data
2. Handling missing values
3. Transforming and categorizing data
4. Filtering and merging datasets

Let's get started!

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
```

In [2]:
```
# Simulating a psychology survey dataset
np.random.seed(42)
data = {
    "Participant_ID": range(1, 11),
    "Age": [23, 45, 34, np.nan, 29, 40, 36, 28, 33, np.nan],
    "Gender": ["F", "M", "M", "F", "F", "M", "F", "M", "M", "F"],
    "Stress_Level": [3, 7, 5, 6, 2, 8, 4, 5, np.nan, 6],
    "Anxiety_Score": [np.nan, 12, 8, 15, 9, 14, 6, 7, 10, 11]
}
df = pd.DataFrame(data)
```

In [3]:
```
# Display the first few rows
print("Original Dataset:")
print(df)
```

```
Original Dataset:
   Participant_ID   Age Gender  Stress_Level  Anxiety_Score
0               1  23.0      F           3.0            NaN
1               2  45.0      M           7.0           12.0
2               3  34.0      M           5.0            8.0
3               4   NaN      F           6.0           15.0
4               5  29.0      F           2.0            9.0
5               6  40.0      M           8.0           14.0
6               7  36.0      F           4.0            6.0
7               8  28.0      M           5.0            7.0
8               9  33.0      M           NaN           10.0
9              10   NaN      F           6.0           11.0
```

Example 1: Handling Missing Values

In [4]:
```
# Checking for missing values
def check_missing_values(df):
    print("\nMissing Values:")
    print(df.isnull().sum())

check_missing_values(df)
```

```
Missing Values:
Participant_ID    0
Age               2
Gender            0
Stress_Level      1
Anxiety_Score     1
dtype: int64
```

In [5]:
```
# Filling missing values in Age with the median
# Filling missing values in Stress_Level and Anxiety_Score with the mean
df["Age"].fillna(df["Age"].median(), inplace=True)
df["Stress_Level"].fillna(df["Stress_Level"].mean(), inplace=True)
df["Anxiety_Score"].fillna(df["Anxiety_Score"].mean(), inplace=True)

print("\nDataset after handling missing values:")
print(df)
```

```
Dataset after handling missing values:
   Participant_ID   Age Gender  Stress_Level  Anxiety_Score
0               1  23.0      F      3.000000      10.222222
1               2  45.0      M      7.000000      12.000000
2               3  34.0      M      5.000000       8.000000
3               4  33.5      F      6.000000      15.000000
4               5  29.0      F      2.000000       9.000000
5               6  40.0      M      8.000000      14.000000
6               7  36.0      F      4.000000       6.000000
7               8  28.0      M      5.000000       7.000000
8               9  33.0      M      5.111111      10.000000
9              10  33.5      F      6.000000      11.000000
```

Example 2: Categorizing Psychological Data

In [6]:
```
# Creating a categorical variable based on Anxiety Score
def categorize_anxiety(score):
    if score < 8:
        return "Low"
    elif 8 <= score < 12:
        return "Moderate"
    else:
        return "High"

df["Anxiety_Category"] = df["Anxiety_Score"].apply(categorize_anxiety)

print("\nDataset with Anxiety Categories:")
print(df)
```

```
Dataset with Anxiety Categories:
   Participant_ID   Age Gender  Stress_Level  Anxiety_Score Anxiety_Category
0               1  23.0      F      3.000000      10.222222         Moderate
1               2  45.0      M      7.000000      12.000000             High
2               3  34.0      M      5.000000       8.000000         Moderate
3               4  33.5      F      6.000000      15.000000             High
4               5  29.0      F      2.000000       9.000000         Moderate
5               6  40.0      M      8.000000      14.000000             High
6               7  36.0      F      4.000000       6.000000              Low
7               8  28.0      M      5.000000       7.000000              Low
8               9  33.0      M      5.111111      10.000000         Moderate
9              10  33.5      F      6.000000      11.000000         Moderate
```

Example 3: Filtering Data

In [7]:
```python
# Selecting participants with high stress levels (>=6)
high_stress_df = df[df["Stress_Level"] >= 6]

print("\nParticipants with High Stress Levels:")
print(high_stress_df)

### Example 4: Merging Data

# Creating a new dataset with additional participant information
demographics_data = {
    "Participant_ID": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    "Education_Level": ["Bachelor", "Master", "PhD", "Bachelor", "High Schoc
}
demographics_df = pd.DataFrame(demographics_data)

# Merging datasets on Participant_ID
df_merged = pd.merge(df, demographics_df, on="Participant_ID")

print("\nMerged Dataset:")
print(df_merged)
```

```
Participants with High Stress Levels:
   Participant_ID   Age Gender  Stress_Level  Anxiety_Score Anxiety_Category
1               2  45.0      M           7.0           12.0             High
3               4  33.5      F           6.0           15.0             High
5               6  40.0      M           8.0           14.0             High
9              10  33.5      F           6.0           11.0         Moderate

Merged Dataset:
   Participant_ID   Age Gender  Stress_Level  Anxiety_Score Anxiety_Category
\
0               1  23.0      F      3.000000      10.222222         Moderate
1               2  45.0      M      7.000000      12.000000             High
2               3  34.0      M      5.000000       8.000000         Moderate
3               4  33.5      F      6.000000      15.000000             High
4               5  29.0      F      2.000000       9.000000         Moderate
5               6  40.0      M      8.000000      14.000000             High
6               7  36.0      F      4.000000       6.000000              Low
7               8  28.0      M      5.000000       7.000000              Low
8               9  33.0      M      5.111111      10.000000         Moderate
9              10  33.5      F      6.000000      11.000000         Moderate

  Education_Level
0        Bachelor
1          Master
2             PhD
3        Bachelor
4     High School
5          Master
6             PhD
7        Bachelor
8          Master
9     High School
```

Exercise

Now it's your turn! Complete the following exercises to practice data wrangling on the psychology dataset.

1. **Filtering**: Create a new DataFrame that includes only participants with a "High" Anxiety Category. Display the resulting DataFrame.
2. **Data Transformation**: Add a new column called "Mental_Wellbeing_Score" by calculating the difference between Stress_Level and Anxiety_Score. Higher scores indicate better mental well-being. Display the updated dataset.

Write your code below each question and run the cells to test your solution!

#1 Filtering

In [ ]:
```
#Looking to see a suitable threshold for high anxiety
print(df.max(axis=0)['Anxiety_Score'])
```

```
print(df.min(axis=0)['Anxiety_Score'])
```

```
15.0
6.0
```

In [11]:
```
highStress = df[df['Anxiety_Score'] >= 10]
highStress
```

Out[11]:

	Participant_ID	Age	Gender	Stress_Level	Anxiety_Score	Anxiety_Category
0	1	23.0	F	3.000000	10.222222	Moderate
1	2	45.0	M	7.000000	12.000000	High
3	4	33.5	F	6.000000	15.000000	High
5	6	40.0	M	8.000000	14.000000	High
8	9	33.0	M	5.111111	10.000000	Moderate
9	10	33.5	F	6.000000	11.000000	Moderate

#2 Data Transformation

Add a new column called "Mental_Wellbeing_Score" by calculating the difference between Stress_Level and Anxiety_Score. Higher scores indicate better mental well-being. Display the updated dataset.

In [12]:
```
df['Mental_Wellbeing_Score'] = df['Stress_Level'] - df['Anxiety_Score']
df
```

Out[12]:

	Participant_ID	Age	Gender	Stress_Level	Anxiety_Score	Anxiety_Category	Ment
0	1	23.0	F	3.000000	10.222222	Moderate	
1	2	45.0	M	7.000000	12.000000	High	
2	3	34.0	M	5.000000	8.000000	Moderate	
3	4	33.5	F	6.000000	15.000000	High	
4	5	29.0	F	2.000000	9.000000	Moderate	
5	6	40.0	M	8.000000	14.000000	High	
6	7	36.0	F	4.000000	6.000000	Low	
7	8	28.0	M	5.000000	7.000000	Low	
8	9	33.0	M	5.111111	10.000000	Moderate	
9	10	33.5	F	6.000000	11.000000	Moderate	