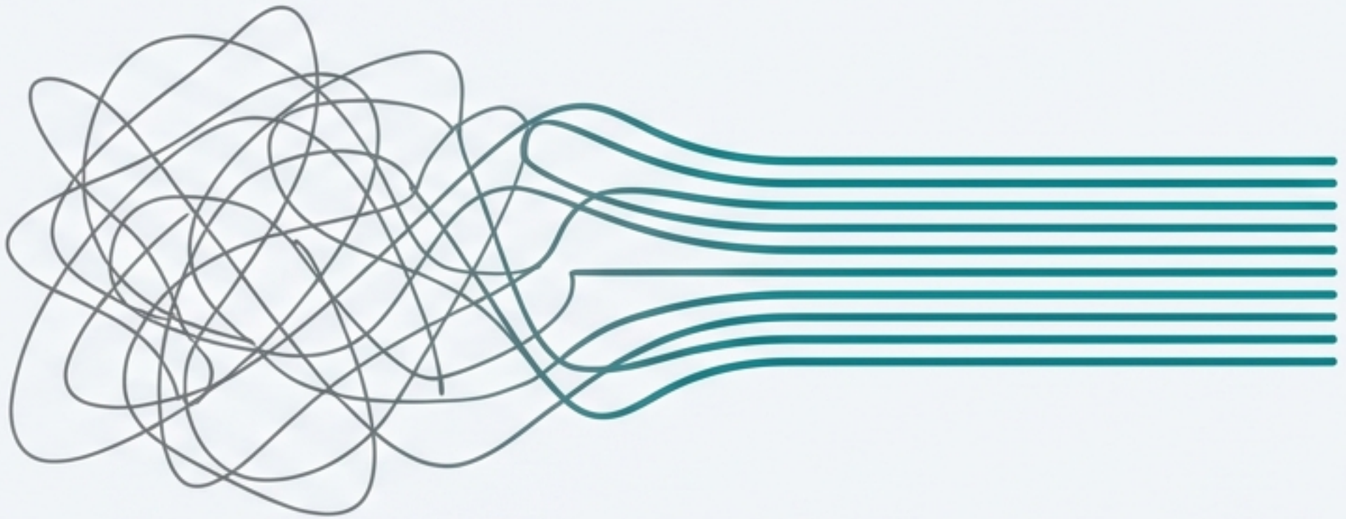
Data Preparation: The Foundation of Powerful Visualization

A Hands-On Workshop Using Python and pandas



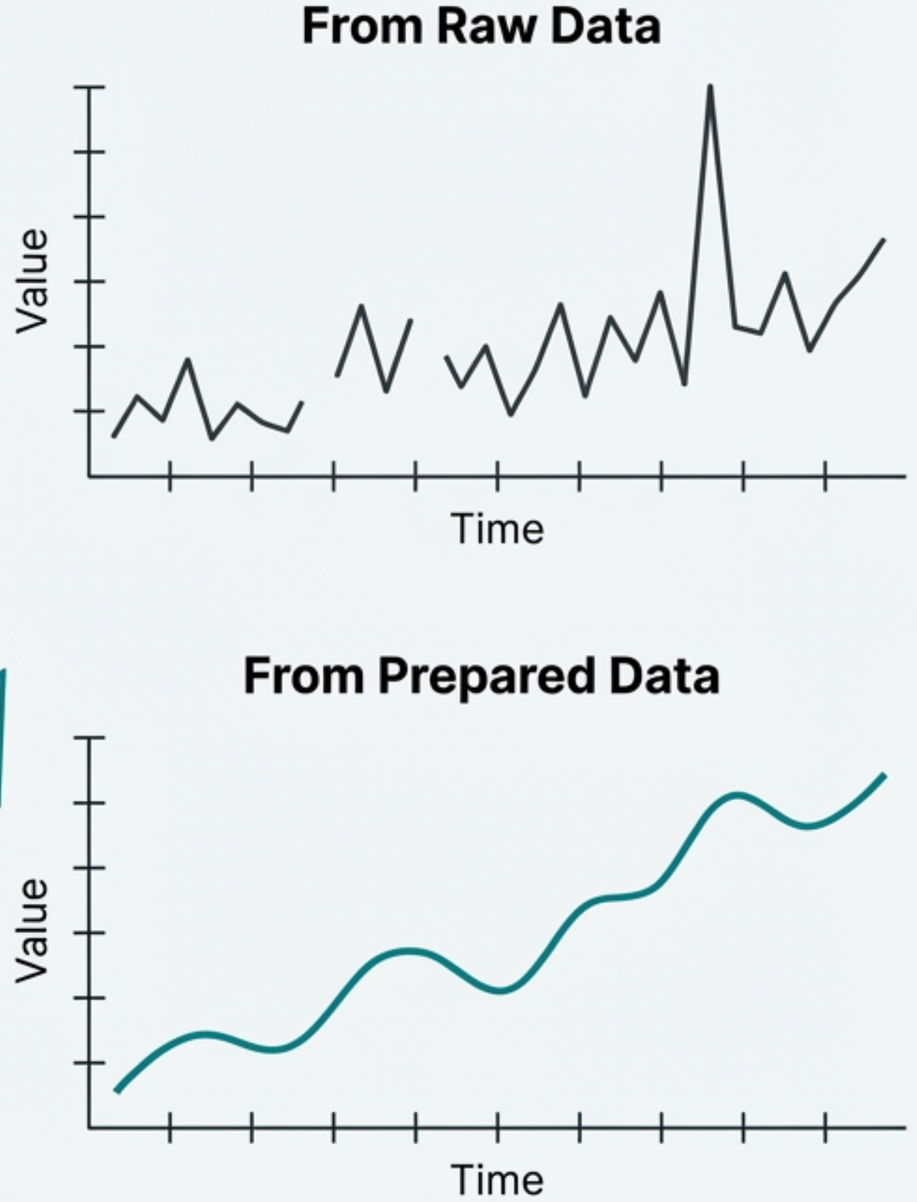
The Hidden Risk in Raw Data

Before we create insightful visualizations, we must confront a critical truth: raw data is often unreliable. Using it directly can lead to flawed conclusions.

- Missing Values: Gaps in your data that can skew results.

- Outliers: Atypical values that distort statistical summaries.

- Incorrect Data Types: Data stored in the wrong format, limiting analysis (e.g., dates stored as text).

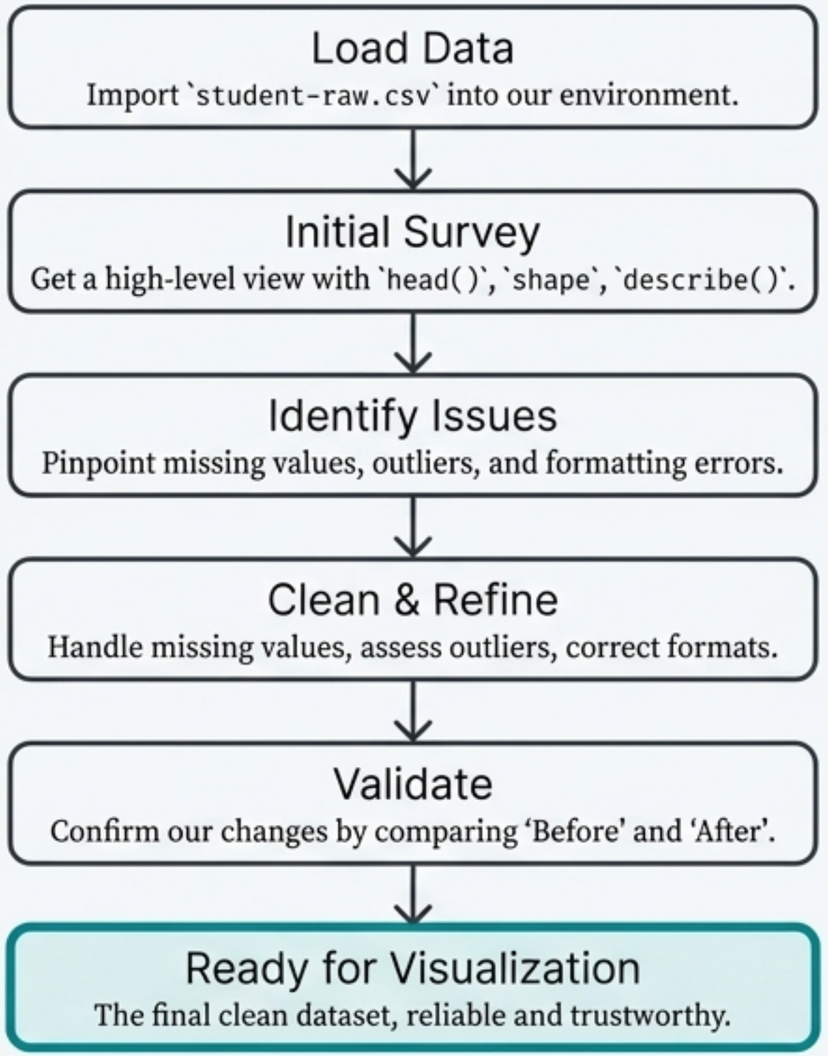


Your Mission: Mastering the Essentials of Data Preparation

By the end of this workshop, you will be able to:

- **Understand the Process Systematically:** Clearly explain the fundamental steps of preparing data for visualization.
- **Recognize Data Quality Issues:** Understand how data quality directly impacts the accuracy of your charts and interpretations.
- **Apply Essential Tools:** Use the pandas library in Python to explore, clean, and format a dataset effectively.

Our Workflow: From Raw Data to Visualization-Ready



Step 1: Importing the Toolkit and Loading Our Data

The first step is to bring our dataset into our Python environment using the pandas library, the industry standard for data manipulation. We will be working with the **`student-raw.csv`** dataset.

```
# Import the pandas library
import pandas as pd

# Load the raw dataset from a CSV file
df = pd.read_csv("student-raw.csv")
```

Imports the necessary library and assigns it the alias 'pd'.

Reads the CSV file and stores its contents in a DataFrame named ‘df’.

First Contact: A Quick Look at the Data's Structure

The `.head()` command is like a quick glance at the top of a document. It shows us the first 5 rows, helping us understand the column names and the type of data in each.

```
# Display the first 5 rows of the dataset
df.head()
```

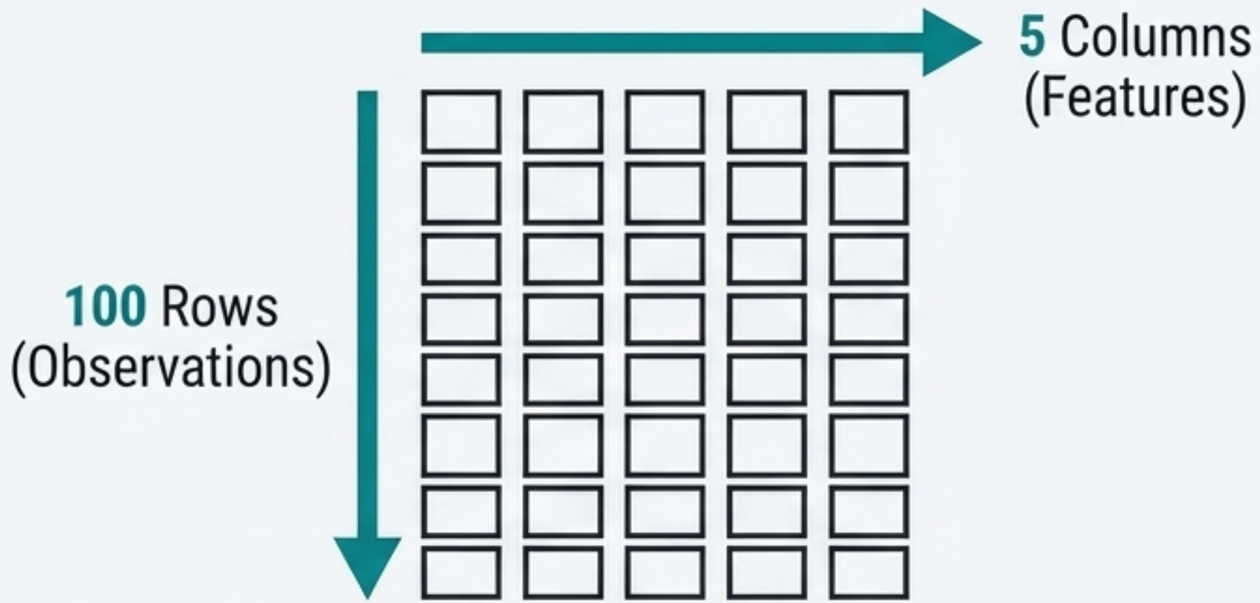
	student_id	full_name	gpa	major	enrollment_date	credits_earned
0	1001	Alice Johnson	3.8	Computer Science	2023-09-01	60
1	1002	Bob Smith	3.5	Mathematics	2022-09-01	45
2	1003	Charlie Brown	3.2	Engineering	2021-09-01	90
3	1004	David Lee	3.9	Physics	2023-01-15	30
4	1005	Eva Chen	3.6	Biology	2020-09-01	120

Sizing Up the Challenge: Understanding the Dataset's Dimensions

Before we dive deeper, we need to know the scale of our data. The `.shape` attribute tells us exactly how many rows (observations) and columns (features) we are working with.

```
# Get the dimensions of the dataset (rows, columns)
df.shape
```

```
(100, 5)
```



The Investigator's View: Uncovering Clues with a Statistical Summary

The `.describe()` command is one of the most powerful tools for initial exploration. It provides a statistical summary of each column, giving us immediate clues about potential data quality issues.

```
# Generate a descriptive statistical summary of the dataset
df.describe(include="all")
```

Producer Note: Pro Tip: Using `include='all'` ensures that non-numeric columns (like dates or categories) are included in the summary, giving us a more complete picture of the dataset.

Decoding the Clues: What `describe()` Reveals

```
df.describe(include="all")
```

	student_id	gender	major	gpa	start_date	graduation_date	notes
count	100	100	100	98	100	90	10
unique	100	2	15	NaN	95	85	9
top	S001	F	Computer Science	NaN	2020-09-01	2024-06-01	NaN
freq	1	52	18	NaN	3	4	2
mean	NaN	NaN	NaN	3.25	NaN	NaN	NaN
std	NaN	NaN	NaN	1.1	NaN	NaN	NaN
min	NaN	NaN	NaN	-1.5	NaN	NaN	NaN
25%	NaN	NaN	NaN	2.8	NaN	NaN	NaN
50%	NaN	NaN	NaN	3.3	NaN	NaN	NaN
75%	NaN	NaN	NaN	3.8	NaN	NaN	NaN
max	NaN	NaN	NaN	5.2	NaN	NaN	NaN

Notice the `count` for 'gpa' is lower than the total 100 rows from `.shape`. This indicates **Missing Values.**

Similarly, a `min` GPA less than 0 is impossible and signals another **Potential Outlier.**

A `max` GPA greater than 4.0 is a clear red flag. This points to a **Potential Outlier** or data entry error.

Issue #1: Strategically Handling Missing Values

Missing data doesn't always mean an error, but we must decide how to handle it. There are three primary strategies:

- **Drop**: Remove the rows. Best for small **amounts** of missing data where the loss is acceptable.
- **Impute**: Replace the missing value with a calculated one (e.g., mean, median). Preserves the row.
- **Keep**: Leave it as is for specific analysis on why the data is missing.

Diagnosis: Count Missing Values

```
# Count the number of missing (NaN) values in each column
df.isna().sum()
```

Taking Action: Imputing Missing GPA Values

For our dataset, we will choose to ***impute*** the missing ‘gpa’ values. We’ll replace the empty spots with the average (mean) GPA from the entire dataset. This allows us to keep the student records while filling the gap with a reasonable estimate.

```
# Calculate the mean of the 'gpa' column and fill missing values with it
df["gpa"] = df["gpa"].fillna(df["gpa"].mean())
```

Producer note: This action preserves our sample size but slightly alters the original data distribution, a trade-off we must acknowledge.

Issue #2: A Cautious Approach to Outliers

Outliers are data points that differ significantly from other observations. They can be caused by:

- Data entry errors (e.g., a typo).
- Genuine, but rare, occurrences.

> The Golden Rule: **Never delete an outlier automatically.** Always investigate the context. Is a GPA of 5.0 a typo, or does the scale go higher at this fictional university?

Investigation: Re-examine Summary

```
# Re-examine the summary statistics for the 'gpa' column
df["gpa"].describe()
```

Issue #3: Aligning Data Formats for Visualization

The *type* of data in a column is critical. If a date is stored as text (an "object" in pandas), you cannot create a time-series plot. If a number is stored as text, you cannot perform mathematical calculations.

Diagnosis: Check Data Types

```
# Check the data type of each column
df.dtypes
```

Taking Action: Converting Dates to a Usable Format

We can see that `enrollment_date` is currently an 'object'. To use this for any time-based analysis or visualization, we must convert it into a proper datetime format.

```
# Convert the 'enrollment_date' column from object to datetime
df["enrollment_date"] = pd.to_datetime(df["enrollment_date"])
```

```
df.dtypes
student_id                    int64
first_name                   object
last_name                    object
enrollment_date    datetime64[ns]
gpa                         float64
dtype: object
```

Verification: The data type is now correctly set.

The Transformation: A Side-by-Side Comparison

The true impact of our work is revealed when we compare the statistical summary of the original raw data with our newly cleaned data.

```
# Snapshot of the original data
df_raw = pd.read_csv("student-raw.csv")
```

```
# A copy of our cleaned data
df_clean = df.copy()
```

Before Cleaning (df_raw.describe())

	student_id	gpa	credits
count	9871500	98	285.0000
mean	50.383	5.3015	17.0011
std	4.3283	0.2213	1.00297
min	0	1.0	0.0
25%	183	3.0	1.0
50%	258	5.5	1.8
75%	678	5.2	1.2
max	1478	5.2	3.0

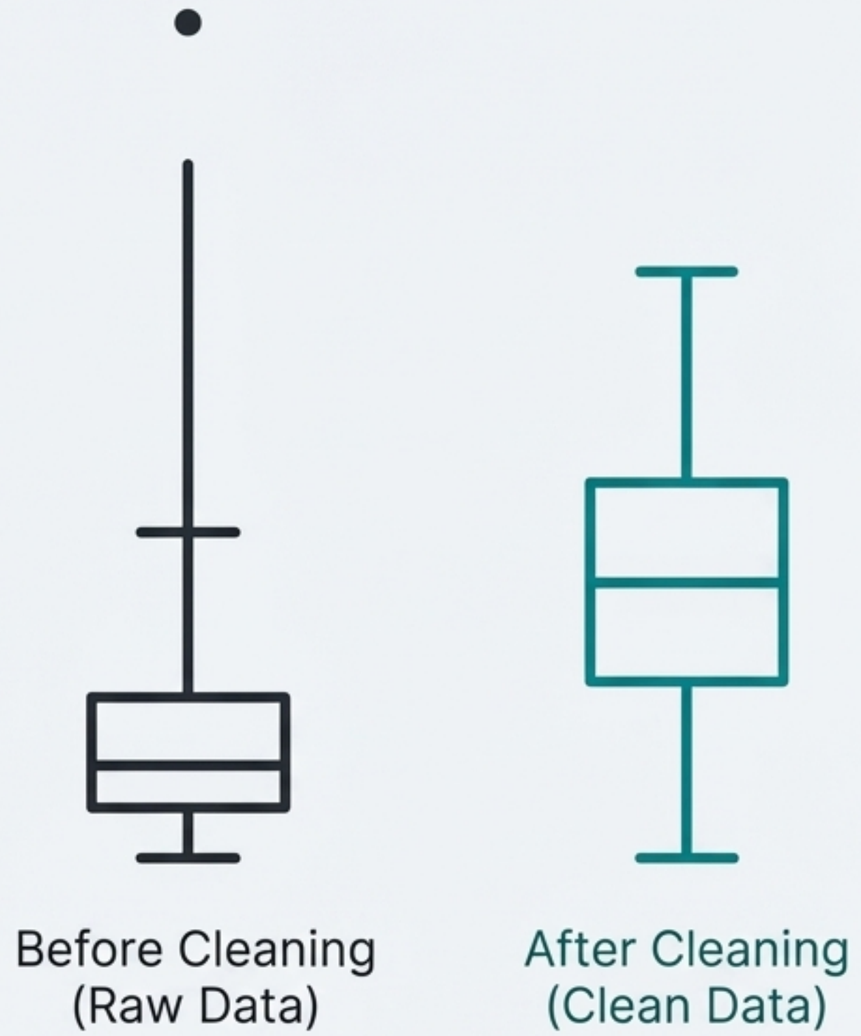
After Cleaning (df_clean.describe())

	student_id	gpa	credits
count	9871500	100	285.0000
mean	48.963	4.8817	16.0011
std	2.3251	0.3681	1.01429
min	0	1.0	0.0
25%	183	2.0	1.0
50%	258	3.5	1.0
75%	678	4.0	1.0
max	1478	4.0	5.0

The Impact: How Cleaning Changes the Narrative

The statistical changes are not just numbers; they fundamentally alter the story our data tells. Consider these questions:

- **Impact on Visualization:** How would a histogram or boxplot of GPA look different now? The range and distribution would be more accurate.
- **Impact on Interpretation:** What conclusions might we have drawn incorrectly from the raw data? (e.g., Overestimating the average GPA, or believing impossible values were real).



Core Principles for Trustworthy Data

1. **1. Preparation Dictates Quality**
   The quality of your visualization is directly determined by the quality of your data preparation.

2. **2. Survey Before You Act**
   A thorough initial overview is more important than rushing into cleaning without a clear plan.

3. **3. Context is King**
   Decisions about missing values and outliers require both technical skill and an understanding of the data's real-world context.

"Good visualization starts with good data."

Continue Your Journey: Practice and Exploration

To solidify these skills, we encourage you to explore further on your own. Try these next steps with the dataset:

- **Visualize the Difference:** Create a boxplot to compare the distribution of GPA before and after cleaning.

- **Experiment with Methods:** Try a different strategy for missing values. Instead of `mean()`, use `median()` or `dropna()`. How does the outcome change?

- **Analyze the Impact:** For each method you try, analyze how it would affect a final visualization and the story you would tell.