
1. Load &
Inspect
Your Data

```
import pandas as pd
df = pd.read_csv('your_data.csv)
print(df.head()
print(df.info()
print(df.describe()
print(df.isnull().sum))
```

What it does:

- Loads your dataset into a DataFrame.
- Shows the first few rows, data types, summary stats, and counts of missing values.

Why it's important:

- You can't clean what you don't understand.
- This step helps you spot issues early and plan your cleaning strategy.


2.
Handle
Missing Values

```
# Drop rows where critical fields are missing
df.dropna(subset=['important_column'],
inplace=True)

# Fill missing numeric values with mean
df['age'].fillna(df['age].mean), inplace=True)

# Fill missing categorical values with mode
df['city'].fillna(df['city].mode(), inplace=
True)
```

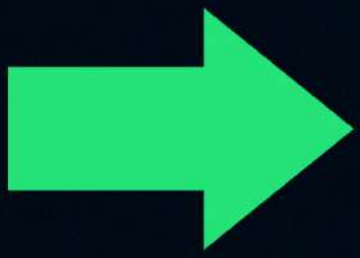
What it does:

- Removes rows with missing values in critical columns.
- Fills missing numeric values with the mean.
- Fills missing categorical values with mode.

Why it's important:

- Missing values can skew your analysis.
- Choosing the right method (drop, fill, or flag) depends on your data and business context.

3. Remove Duplicate Records



```
print(f“Duplicates:
{df.duplicated().sum()}")
df.drop_duplicates(inplace=True)
```

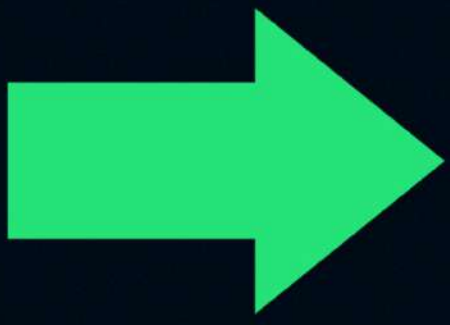
What it does:

- Counts and removes duplicate rows.

Why it’s important:

- Duplicates can distort your results and lead to incorrect conclusions.

4. Standardize Text & Formats



```
# Clean messy strings: trim spaces, fix
case, remove special.cnaracters
df['name'] = df['name].str.strip().str.title()
df['phone] = df['phone].str.replace(r\\D',''),
# Keep digits only
# Convert date string to datetime
df['date] = pd.to_datetime(df['date‟),
errors='coerce')
```

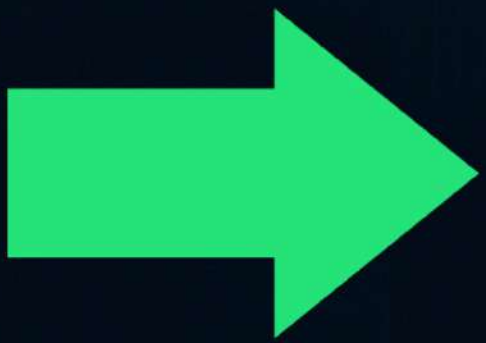
What it does:

- Cleans text by removing extra spaces, fixing case, and removing special characters.
- Converts date strings to datetime objects.

Why it's important:

- Inconsistent formats can cause errors and make analysis difficult.

5. Correct Data Types



```
df['price'] = pd.to_numeric(df'rice),
    errors='coerce)
df['category] = df['category].astype
    'category')
```

What it does:

- Converts columns to the correct data types (numeric, categorical, etc.).

Why it's important:

- Correct data types ensure accurate calculations and efficient storage.

6. Identify & Handle Outliers

```
import numpy as np
Q1 = df['salary')quantile(0.25)
Q3 = df['salary')quantile(0.75)
IQR = Q3 – Q1
outliers =  df(df['salary] < Q1–1,5
* IQR) | (df['salary] > Q3 +1,5 * IQR)]
print(f"Outliers found: {len(outliers))
```

What it does:

- Identifies outliers using the Interquartile Range (IQR) method.

Why it's important:

- Outliers can skew your analysis and lead to incorrect conclusions.

7. Combine Data Sources Carefully

```
df_combined = pd.merge(df1, df2,
on='key_column', how='inner')
print(f"Rows after merge:
      {len(df_combined)]})
```

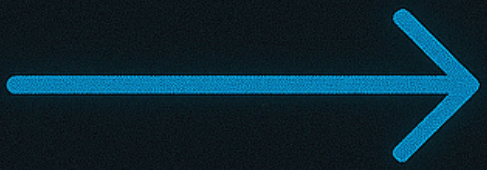
What it does:

- Merges two datasets on a common key.

Why it's important:

- Combining data sources correctly ensures you don't lose or duplicate information.

8. Final Validation & Save Cleaned Data



```
print(df_combined.info())
print(df_combined.isnull().sum())
df_combined.to_csv('cleaned_data
csv',index=False)
```

What it does:

- Re-checks for missing values, types, and duplicates.
- Saves the cleaned data to a CSV file.

Why it's important:

- Final validation ensures your data is ready for analysis.
- Saving the cleaned data allows you to reproduce your results.