Data Cleaning with Python

A Practical Guide

Handling Missing Values

- **isna()** – Detect missing (NaN) values
- **notna()** – Identify non-null entries
- **dropna()** – Remove rows or columns
- **fillna()** – Replace missing values with constants or statistics
- **interpolate()** – Fill gaps using trends
- **mean()** – Numeric imputation
- **median()** – Robust imputation for skewed data
- **mode()** – Categorical imputation

Removing Duplicates

- **duplicated()** – Detect duplicate rows
- **drop_duplicates()** – Remove duplicate rows
- **subset=** – Check duplicates in specific columns
- **keep='first'** – Keep the first occurrence
- **keep='last'** – Keep the last occurrence
- **value_counts()** – Identify repeated values
- **groupby()** – Aggregate data by duplicate keys
- **reset_index()** – Rebuild index after cleaning

Fixing Data Types

- **dtype** – Inspect column data types
- **astype()** – Convert data types
- **to_datetime()** – Fix and standardize date columns
- **to_numeric()** – Convert numeric text
- **errors='coerce'** – Handle invalid values
- **dt.year** – Extract year
- **dt.month** – Extract month
- **tz_localize()** – Assign timezone to date time data

Cleaning Text Data

- **str.lower()** – Normalize text to lowercase
- **str.upper()** – Enforce consistent uppercase text
- **str.strip()** – Remove leading and trailing spaces
- **str.replace()** – Fix inconsistent or incorrect text
- **str.contains()** – Identify patterns or keywords
- **str.split()** – Break compound strings into parts
- **str.len()** – Validate text length consistency
- **str.extract()** – Extract patterns using regex

Handling Outliers

- **describe()** – Review value distribution
- **quantile()** – Define IQR-based boundaries
- **between()** – Filter values within valid ranges
- **clip()** – Cap extreme values
- **mean()** – Reference central tendency
- **std()** – Measure data spread
- **abs()** – Remove sign-based noise
- **np.where()** – Apply conditional replacements

Standardizing Values

- **replace()** – Correct inconsistent labels
- **map()** – Apply controlled value mapping
- **round()** – Standardize decimal precision
- **apply()** – Apply custom transformation logic
- **astype()** – Normalize data types
- **mul()** – Convert values to new units
- **div()** – Scale values proportionally
- **clip()** – Normalize values to a range

Validating Data

- **unique()** – Check allowed values
- **nunique()** – Validate value cardinality
- **isin()** – Enforce domain constraints
- **between()** – Validate numeric ranges
- **duplicated()** – Verify key uniqueness
- **isna().sum()** – Review missing value distribution
- **is_monotonic_increasing** – Check sequence order
- **assert** – Enforce data rules

Fixing Inconsistent Data

- **loc[]** – Apply conditional updates
- **where()** – Replace values conditionally
- **mask()** – Inverse conditional replacement
- **merge()** – Correct data using references
- **update()** – Overwrite incorrect values
- **combine_first()** – Fill from trusted source
- **drop()** – Remove invalid records
- **query()** – Filter data using conditions

Final Pre-Analysis Checklist

- No unexpected nulls
- No duplicate primary keys
- Correct data types
- Clean text fields
- Valid ranges
- Business rules satisfied

Why Data Cleaning Matters

Clean data means:

- Accurate insights
- Reliable dashboards
- Better decisions
- Fewer stakeholder questions
- Trust in analytics

Data Cleaning = 70% of a Data Analyst’s job
Master it, and analysis becomes easy.