Documentation of cleaning data set using pandas:

Step 1: Importing Required Library

import pandas as pd

Function:

Imports the **Pandas** library, which is used for handling, cleaning, and analyzing data in tabular (DataFrame) format.

Step 2: Loading the Dataset

file_path = "machine_downtime_copy (3).csv"

df = pd.read_csv(file_path)

Function:

- Loads the CSV file into a Pandas DataFrame named df.
- pd.read_csv() reads the dataset from the given path.

Step 3: Basic Data Exploration

df

df.head()

df.tail()

df.shape

df.columns

df.info()

df.dtypes

df.describe()

Function:

Used to **understand the structure and quality of the data** before cleaning.

- df.head() → shows the first 5 rows.
- df.tail() → shows the last 5 rows.
- df.shape → returns the number of rows and columns.
- df.columns → lists all column names.
- df.info() → shows data types and non-null counts.
- df.dtypes → lists data types for each column.

- df.describe() → provides statistical summary (mean, std, min, max, etc.) for numeric columns.

Step 4: Checking Missing Values

df.isnull().sum()

missing_before = df.isnull().sum()

Function:

- df.isnull().sum() counts how many missing (NaN) values exist in each column.
- The result is saved into missing_before for reference before cleaning.

Step 5: Handling Missing Values

```
for col in df.columns:
  if df[col].dtype in ['int64', 'float64']:
    df[col] = df[col].fillna(df[col].median())
  else:
    df[col] = df[col].fillna(df[col].mode()[0])
```

Function:

This loop **fills all missing values** based on the column type:

- **Numeric columns:** replaced with the **median** → more robust against outliers.
- **Text/categorical columns:** replaced with the **mode** (most common value).

Goal: eliminate all missing data without distorting distributions.

Step 6: Recheck Missing Values

df.isnull().sum()

Function:

Confirms that all missing values have been successfully filled.
Result should show all zeros.

Step 7: Checking for Duplicate Rows

df.duplicated().sum()

Function:

Checks whether the dataset contains any duplicate rows.
If duplicates exist, you can remove them using:

df.drop_duplicates(inplace=True)

Step 8: Cleaning Text Columns

df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)

Function:

Removes **leading and trailing spaces** in text columns.
Ensures that categorical values like " Machine_1 " and "Machine_1" are treated the same.

Step 10: Standardizing Text Formatting

for col in df.select_dtypes(include='object').columns:

  df[col] = df[col].str.title()

Function:

Converts all text values to **Title Case** (e.g., “machine_failure” → “Machine_Failure”).
This ensures consistent formatting across text columns.

Step 11: Detecting Outliers (First Pass)

numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

outlier_summary = {}

for col in numeric_cols:

  Q1 = df[col].quantile(0.25)

  Q3 = df[col].quantile(0.75)

  IQR = Q3 - Q1

  lower_bound = Q1 - 1.5 * IQR

  upper_bound = Q3 + 1.5 * IQR

  outlier_count = ((df[col] < lower_bound) | (df[col] > upper_bound)).sum()

  outlier_summary[col] = outlier_count

Function:

Detects **outliers** using the **IQR (Interquartile Range)** method:

- Calculates Q1, Q3, and IQR = Q3 - Q1.

- Values outside [Q1 - 1.5×IQR, Q3 + 1.5×IQR] are considered outliers.
- The result is stored in the dictionary outlier_summary.

Step 12: Printing Outlier Summary

```
print(" Outlier Detection ")
total_outliers = 0

for col, count in outlier_summary.items():
  if count > 0:
    print(f" Column '{col}' has {count} outliers.")
    total_outliers += count
  else:
    print(f" Column '{col}' has no outliers.")

print("\n Total outliers found across dataset:", total_outliers)
```

Function:

- Prints a summary showing how many outliers each column contains.
- Displays the total number of detected outliers across the dataset.

Step 13: Displaying Outlier Counts

```
outlier_df = pd.DataFrame(list(outlier_summary.items()), columns=["Column", "Outlier Count"])
display(outlier_df)
```

Function:

Creates and displays a **DataFrame summary** of the outlier counts per column for easy review.

Step 14: Outlier Treatment (Second Pass – Cleaning)

```
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
outlier_summary = {}

for col in numeric_cols:
  while True:
```

```
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outlier_mask = (df[col] < lower_bound) | (df[col] > upper_bound)
    outlier_count = outlier_mask.sum()

    if outlier_count > 0:
      median_value = df[col].median()
      df.loc[outlier_mask, col] = median_value
    else:
      break

  outlier_summary[col] = outlier_mask.sum()
```

Function:

This block **removes or corrects outliers**:

- Runs a loop until no outliers remain in the column.
- Outliers are replaced by the column's **median** to maintain statistical balance.
- Ensures that all numeric values are within acceptable range after replacement.

Step 15: Display Remaining Outliers

```
outlier_df = pd.DataFrame(list(outlier_summary.items()), columns=["Column", "Remaining Outliers"])
display(outlier_df)
```

Function:

Displays how many outliers remain after treatment.
Expected result → all "0".

Step 16: Saving the Cleaned Dataset

```
df.to_csv("machine_downtime_cleaned.csv", index=False)
```

Function:

Exports the fully cleaned dataset into a new CSV file.
index=False prevents adding an extra index column.
The final dataset is now ready for analysis, modeling, or visualization.

Final Results

Checkpoint	**Result**
Missing Values	All filled (0 remaining)
Duplicates	None found
Text Columns	Clean, formatted, and consistent
Numeric Columns	Free from outliers
Exported File	machine_downtime_cleaned.csv

Summary of Functions Uses:

Function	**Purpose**
pd.read_csv()	Load dataset
df.head(), df.info()	Inspect structure
df.isnull().sum()	Detect missing values
fillna()	Handle missing data
duplicated()	Check duplicates
apply(lambda x: x.str.strip())	Remove spaces
replace(regex=True)	Remove special chars
str.title()	Standardize text
quantile() + IQR	Detect outliers
median()	Replace outliers