🐼 Data Processing Techniques in Pandas (A–Z)

1️⃣ Data Loading (Input)

✔ Read data

```
import pandas as pd

df = pd.read_csv("data.csv")
df = pd.read_excel("data.xlsx")
df = pd.read_json("data.json")
df = pd.read_sql(query, connection)
```

2️⃣ Data Inspection (Understanding Data)

✔ Basic info

```
df.head()
df.tail()
df.shape
df.columns
df.info()
df.describe()
```

📌 **Purpose**: Dataset size, column types, missing values

3️⃣ Handling Missing Data

✔ Detect missing values

```
df.isnull()
df.isnull().sum()
```

✔ Drop missing values

```
df.dropna()
```

```
df.dropna(axis=1)
```

✔ Fill missing values

```
df.fillna(0)
df.fillna(df.mean())
df['age'].fillna(df['age'].median())
```

4️⃣ Data Cleaning

✔ Remove duplicates

```
df.duplicated()
df.drop_duplicates()
```

✔ Rename columns

```
df.rename(columns={'old':'new'}, inplace=True)
```

✔ Change data type

```
df['age'] = df['age'].astype(int)
```

5️⃣ Data Selection & Indexing

✔ Column selection

```
df['name']
df[['name','age']]
```

✔ Row selection

```
df.loc[0]
df.iloc[0]
df.loc[0:5]
```

✔ Conditional selection

```
df[df['age'] > 18]
```

6️⃣ Sorting Data

```
df.sort_values(by='age')
df.sort_values(by='age', ascending=False)
```

7️⃣ Filtering Data

```
df[(df['age'] > 18) & (df['gender'] == 'Male')]
```

8️⃣ Data Transformation

✔ Apply function

```
df['salary'] = df['salary'].apply(lambda x: x*1.1)
```

✔ Map values

```
df['gender'] = df['gender'].map({'M':0, 'F':1})
```

9️⃣ Feature Engineering (ML Important ⭐)

✔ Create new column

```
df['BMI'] = df['weight'] / (df['height']**2)
```

✔ Binning

```
df['age_group'] = pd.cut(df['age'], bins=[0,18,40,60,100])
```

🔟 Encoding Categorical Data

✔ Label Encoding

```
df['gender'] = df['gender'].astype('category').cat.codes
```

✔ One-Hot Encoding

```
pd.get_dummies(df, columns=['city'])
```

1️⃣1️⃣ Grouping & Aggregation

```
df.groupby('gender')['salary'].mean()
df.groupby('city').agg({'salary':'mean','age':'max'})
```

📌 Used in **EDA & analytics**

1️⃣2️⃣ Merging & Joining Data

✔ Merge

```
pd.merge(df1, df2, on='id', how='inner')
```

✔ Join

```
df1.join(df2)
```

1️⃣3️⃣ Concatenation

```
pd.concat([df1, df2], axis=0)
```

1️⃣4️⃣ Handling Date & Time

```
df['date'] = pd.to_datetime(df['date'])
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
```

1️⃣5️⃣ Scaling / Normalization (ML Prep)

```
df['age_norm'] = (df['age'] - df['age'].min()) / (df['age'].max() -
df['age'].min())
```

1️⃣6️⃣ Outlier Detection

```
Q1 = df['salary'].quantile(0.25)
Q3 = df['salary'].quantile(0.75)
IQR = Q3 - Q1

df = df[(df['salary'] >= Q1 - 1.5*IQR) & (df['salary'] <= Q3 + 1.5*IQR)]
```

1️⃣7️⃣ Sampling Data

```
df.sample(n=100)
df.sample(frac=0.2)
```

1️⃣8️⃣ Export Data (Output)

```
df.to_csv("cleaned.csv", index=False)
df.to_excel("cleaned.xlsx")
```

🔚 Complete Data Processing Pipeline (Example)

```
df = pd.read_csv("data.csv")
df.drop_duplicates(inplace=True)
```

```
df.fillna(df.mean(), inplace=True)
df = pd.get_dummies(df, drop_first=True)
```

🧠 Exam Summary (5 Marks)

> Pandas provides various data processing techniques such as data loading, cleaning, handling missing values, transformation, encoding, grouping, merging, and exporting. These techniques help prepare raw data into clean and structured form suitable for analysis and machine learning.

🐼 Pandas Practice Problems + Solutions (50)

Assume:

```
import pandas as pd
import numpy as np

df = pd.read_csv("data.csv")
```

🟢 EASY (1–15)

1️⃣ Load CSV file

```
df = pd.read_csv("data.csv")
```

2️⃣ Show first 5 rows

```
df.head()
```

3️⃣ Show last 10 rows

```
df.tail(10)
```

4️⃣ Number of rows & columns

```
df.shape
```

5️⃣ Column names

```
df.columns
```

6️⃣ Dataset info

```
df.info()
```

7️⃣ Summary statistics

```
df.describe()
```

8️⃣ Select one column

```
df['age']
```

9️⃣ Select multiple columns

```
df[['name', 'salary']]
```

🔟 Check missing values

```
df.isnull().sum()
```

1️⃣1️⃣ Fill missing values with 0

```
df.fillna(0, inplace=True)
```

1️⃣2️⃣ Drop rows with missing values

```
df.dropna(inplace=True)
```

1️⃣3️⃣ Remove duplicates

```
df.drop_duplicates(inplace=True)
```

1️⃣4️⃣ Sort by age

```
df.sort_values(by='age')
```

1️⃣5️⃣ Sort salary descending

```
df.sort_values(by='salary', ascending=False)
```

🟡 MEDIUM (16–35)

1️⃣6️⃣ Filter age > 30

```
df[df['age'] > 30]
```

1️⃣7️⃣ Filter multiple conditions

```
df[(df['age'] > 25) & (df['gender'] == 'Male')]
```

1️⃣8️⃣ Rename column

```
df.rename(columns={'old_name':'new_name'}, inplace=True)
```

1️⃣9️⃣ Change datatype

```
df['age'] = df['age'].astype(int)
```

2️⃣0️⃣ Add new column

```
df['bonus'] = df['salary'] * 0.1
```

2️⃣1️⃣ Apply function

```
df['salary'] = df['salary'].apply(lambda x: x+500)
```

2️⃣2️⃣ Replace values

```
df['gender'].replace({'M':'Male','F':'Female'}, inplace=True)
```

2️⃣3️⃣ Unique values

```
df['city'].unique()
```

2️⃣4️⃣ Value counts

```
df['city'].value_counts()
```

2️⃣5️⃣ Group by city (mean salary)

```
df.groupby('city')['salary'].mean()
```

2️⃣6️⃣ Multiple aggregation

```
df.groupby('gender').agg({'salary':'mean','age':'max'})
```

2️⃣7️⃣ Count per group

```
df.groupby('city').size()
```

2️⃣8️⃣ One-hot encoding

```
pd.get_dummies(df, columns=['city'])
```

2️⃣9️⃣ Label encoding

```
df['gender'] = df['gender'].astype('category').cat.codes
```

3️⃣0️⃣ Random sample (10 rows)

```
df.sample(10)
```

3️⃣1️⃣ Fraction sample (20%)

```
df.sample(frac=0.2)
```

3️⃣2️⃣ Index reset

```
df.reset_index(drop=True, inplace=True)
```

3️⃣3️⃣ Set column as index

```
df.set_index('id', inplace=True)
```

3️⃣4️⃣ Export to CSV

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

3️⃣5️⃣ Check correlation

```
df.corr(numeric_only=True)
```

🔴 ADVANCED (36–50)

3️⃣6️⃣ Convert string to datetime

```
df['date'] = pd.to_datetime(df['date'])
```

3️⃣7️⃣ Extract year

```
df['year'] = df['date'].dt.year
```

3️⃣8️⃣ Filter by date

```
df[df['date'] > '2024-01-01']
```

3️⃣9️⃣ Merge two DataFrames

```
pd.merge(df1, df2, on='id', how='inner')
```

4️⃣0️⃣ Concatenate DataFrames

```
pd.concat([df1, df2])
```

4️⃣1️⃣ Detect outliers (IQR)

```
Q1 = df['salary'].quantile(0.25)
Q3 = df['salary'].quantile(0.75)
IQR = Q3 - Q1
df[(df['salary'] >= Q1 - 1.5*IQR) & (df['salary'] <= Q3 + 1.5*IQR)]
```

4️⃣2️⃣ Normalize column

```
df['salary_norm'] = (df['salary']-df['salary'].min()) /
(df['salary'].max()-df['salary'].min())
```

4️⃣3️⃣ Binning age

```
df['age_group'] = pd.cut(df['age'], bins=[0,18,40,60,100])
```

4️⃣4️⃣ Rank values

```
df['salary_rank'] = df['salary'].rank()
```

4️⃣5️⃣ Rolling mean

```
df['rolling_avg'] = df['salary'].rolling(3).mean()
```

4️⃣6️⃣ Apply function row-wise

```
df.apply(lambda x: x['salary']*2 if x['age']>30 else x['salary'], axis=1)
```

4️⃣7️⃣ Conditional column

```
df['status'] = np.where(df['salary'] > 50000, 'High', 'Low')
```

4️⃣8️⃣ Pivot table

```
pd.pivot_table(df, values='salary', index='city', columns='gender',
aggfunc='mean')
```

4️⃣9️⃣ Memory usage

```
df.memory_usage(deep=True)
```

5️⃣0️⃣ Complete cleaning pipeline

```
df = (
    df.drop_duplicates()
      .fillna(df.mean(numeric_only=True))
      .reset_index(drop=True)
)
```

🎯 Exam Tip (5 Marks)

> Pandas provides powerful tools for data loading, cleaning, filtering, transformation, aggregation, and feature engineering which are essential for data analysis and machine learning.

I'll show you a REAL-WORLD dataset cleaning workflow (step-by-step) exactly how it's done in industry ML / Data Science projects.

I'll use a **realistic messy dataset** (like Employee / Customer / Medical / ML dataset).

🧹 Real Project Dataset Cleaning (Step-by-Step)

🎯 Project Scenario

Problem:
We have a **raw dataset** collected from different sources.
It contains:

- Missing values
- Wrong data types
- Duplicates
- Outliers
- Categorical data
- Date columns
- Inconsistent values

Goal 👉 **Clean & prepare data for ML model**

📁 STEP 0: Load Dataset

```
import pandas as pd
import numpy as np

df = pd.read_csv("raw_data.csv")
```

🔍 STEP 1: Understand the Dataset (EDA)

```
df.head()
df.tail()
df.shape
df.columns
df.info()
df.describe()
```

✔ Identify:

- Missing values
- Datatypes
- Numerical vs categorical
- Wrong columns

❌ STEP 2: Remove Duplicate Records

```
df.duplicated().sum()
df.drop_duplicates(inplace=True)
```

📌 **Why?**
Duplicates bias ML models.

🚫 STEP 3: Handle Missing Values

🔹 Check missing values

```
df.isnull().sum()
```

🔹 Numerical columns → mean / median

```
df['age'].fillna(df['age'].median(), inplace=True)
df['salary'].fillna(df['salary'].mean(), inplace=True)
```

🔹 Categorical columns → mode

```
df['gender'].fillna(df['gender'].mode()[0], inplace=True)
```

🔹 Drop column if too many missing

```
df.drop(columns=['useless_column'], inplace=True)
```

🔄 STEP 4: Fix Data Types

```
df['age'] = df['age'].astype(int)
df['salary'] = df['salary'].astype(float)
```

Date column

```
df['join_date'] = pd.to_datetime(df['join_date'], errors='coerce')
```

🧽 STEP 5: Clean Text & Categorical Data

🔹 Standardize values

```
df['gender'] = df['gender'].str.lower().str.strip()
df['gender'].replace({'m':'male','f':'female'}, inplace=True)
```

🔹 Fix inconsistent city names

```
df['city'] = df['city'].str.title()
```

🚨 STEP 6: Handle Outliers (IQR Method)

```
Q1 = df['salary'].quantile(0.25)
Q3 = df['salary'].quantile(0.75)
IQR = Q3 - Q1

df = df[(df['salary'] >= Q1 - 1.5*IQR) &
        (df['salary'] <= Q3 + 1.5*IQR)]
```

📌 **Why?**
Outliers destroy ML accuracy.

🧠 STEP 7: Feature Engineering

🔹 New features

```
df['experience_level'] = pd.cut(
    df['experience'],
    bins=[0,2,5,10,40],
    labels=['Junior','Mid','Senior','Expert']
)
```

🔹 Date features

```
df['join_year'] = df['join_date'].dt.year
df['join_month'] = df['join_date'].dt.month
```

🔠 STEP 8: Encode Categorical Features (ML Ready)

🔹 Label Encoding

```
df['gender'] = df['gender'].astype('category').cat.codes
```

🔹 One-Hot Encoding

```
df = pd.get_dummies(df, columns=['city','experience_level'],
drop_first=True)
```

📏 STEP 9: Scaling / Normalization

```
df['salary_norm'] = (
    df['salary'] - df['salary'].min()
) / (
    df['salary'].max() - df['salary'].min()
)
```

🧪 STEP 10: Final Validation Check

```
df.isnull().sum()
df.info()
df.describe()
```

✔ No missing values
✔ Correct datatypes
✔ ML-ready numeric data

💾 STEP 11: Save Clean Dataset

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

🚀 COMPLETE PIPELINE (Industry Style)

```
df = (
    pd.read_csv("raw_data.csv")
      .drop_duplicates()
      .assign(
          age=lambda x: x['age'].fillna(x['age'].median()),
          salary=lambda x: x['salary'].fillna(x['salary'].mean())
      )
)

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

🧠 Interview / Viva Ready Summary (5 Marks)

> In a real project, dataset cleaning involves removing duplicates, handling missing values, fixing data types, cleaning categorical data, detecting outliers, performing feature engineering, encoding categorical features, scaling numerical values, and validating the dataset before saving it for model training.

🚨 Data Cleaning Mistakes Interviewers Ask (With Correct Answers)

1️⃣ Deleting Rows Without Analysis ❌

❌ Mistake

```
df.dropna(inplace=True)
```

❓ Why wrong?

You may delete **important data** and introduce **bias**.

✅ Correct Answer

> Analyze missing pattern first, then decide fill or drop.

```
df.isnull().sum()
df['age'].fillna(df['age'].median(), inplace=True)
```

2️⃣ Filling All Missing Values With Mean ❌

❌ Mistake

```
df.fillna(df.mean(), inplace=True)
```

❓ Why wrong?

Mean is bad for **skewed data & categorical features**.

✅ Correct

- Mean → normal distribution
- Median → skewed
- Mode → categorical

3️⃣ Ignoring Duplicate Records ❌

❌ Mistake

Not checking duplicates at all.

❓ Why wrong?

Duplicates distort **statistics & ML training**.

✅ Correct

```
df.duplicated().sum()
df.drop_duplicates(inplace=True)
```

4️⃣ Not Checking Data Types ❌

❌ Mistake

Treating numeric column as object.

❓ Why wrong?

Math operations fail or give wrong results.

✅ Correct

```
df.info()
df['salary'] = df['salary'].astype(float)
```

5️⃣ Removing Outliers Blindly ❌

❌ Mistake

```
df = df[df['salary'] < 100000]
```

❓ Why wrong?

Outliers may be **valid business cases**.

✅ Correct

> Understand domain before removal.

Use IQR or Z-score **with justification**.

6️⃣ Encoding Categorical Data Incorrectly ❌

❌ Mistake

Label encoding unordered categories.

❓ Why wrong?

Introduces **false order**.

✅ Correct

- Nominal → One-Hot
- Ordinal → Label

```
pd.get_dummies(df, columns=['city'])
```

7️⃣ Data Leakage ❌🔥 (Very Important)

❌ Mistake

Scaling before train-test split.

❓ Why wrong?

Model sees test data information.

✅ Correct

> Always split first, then scale.

```
from sklearn.preprocessing import StandardScaler
```

8️⃣ Dropping Columns Without Understanding ❌

❌ Mistake

```
df.drop(columns=['id'])
```

❓ Why wrong?

Column may be useful for joins or tracking.

✅ Correct

> Drop only non-informative features.

9️⃣ Not Handling Inconsistent Text ❌

❌ Mistake

```
Male, male, MALE
```

❓ Why wrong?

Creates multiple categories.

✅ Correct

```
df['gender'] = df['gender'].str.lower().str.strip()
```

🔟 Ignoring Date-Time Features ❌

❌ Mistake

Using date as string.

❓ Why wrong?

You miss time patterns.

✅ Correct

```
df['date'] = pd.to_datetime(df['date'])
df['year'] = df['date'].dt.year
```

1️⃣1️⃣ Over-Cleaning Data ❌

❌ Mistake

Removing too much data.

❓ Why wrong?

Loss of information → poor model.

✅ Correct

> Balance cleanliness & data richness.

1️⃣2️⃣ Not Validating After Cleaning ❌

❌ Mistake

Saving dataset without checking.

❓ Why wrong?

Hidden issues remain.

✅ Correct

```
df.isnull().sum()
df.describe()
df.info()
```

1️⃣3️⃣ Ignoring Class Imbalance ❌

❌ Mistake

Training without checking target distribution.

❓ Why wrong?

Model becomes biased.

✅ Correct

```
df['target'].value_counts(normalize=True)
```

1️⃣4️⃣ Cleaning Train and Test Separately ❌

❌ Mistake

Different preprocessing logic.

❓ Why wrong?

Inconsistent model behavior.

✅ Correct

> Use **same pipeline**.

1️⃣5️⃣ Not Saving Cleaning Steps ❌

❌ Mistake

Manual cleaning without reproducibility.

❓ Why wrong?

Impossible to debug or deploy.

✅ Correct

> Use scripts or pipelines.

⭐ TOP 5 MOST ASKED (Remember These)

1️⃣ Data leakage
2️⃣ Missing value strategy
3️⃣ Outlier handling
4️⃣ Encoding mistake
5️⃣ Over-cleaning data

🎯 Interview One-Line Answer (Golden)

> Data cleaning mistakes include improper handling of missing values, ignoring duplicates, incorrect encoding, data leakage, and removing outliers without domain understanding, all of which can negatively impact model performance.

🤖 Automated Data Cleaning Pipeline (Industry Standard)

🎯 What is an Automated Data Cleaning Pipeline?

> A reusable system that **automatically cleans raw data** using predefined rules so that the **same logic is applied every time** (no manual mistakes).

✅ Reproducible
✅ Scalable
✅ ML-ready
✅ Interview favorite topic

🧠 What This Pipeline Will Handle

- Remove duplicates
- Handle missing values (numeric & categorical)
- Fix data types
- Clean text columns
- Handle outliers
- Encode categorical features
- Scale numeric features
- Prevent data leakage

🔹 STEP 1: Import Libraries

```
import pandas as pd
import numpy as np
```

🔹 STEP 2: Create a Cleaning Configuration (Very Important)

This makes it **flexible & reusable**.

```
CONFIG = {
    "drop_columns": ["unnecessary_column"],
    "numeric_fill": "median",   # mean / median
    "categorical_fill": "mode",
    "outlier_method": "iqr",    # iqr / none
    "encode": "onehot",         # onehot / label
}
```

🔹 STEP 3: Define Cleaning Functions

1️⃣ Remove duplicates

```
def remove_duplicates(df):
    return df.drop_duplicates()
```

2️⃣ Handle missing values

```
def handle_missing_values(df):
    for col in df.columns:
        if df[col].dtype in ['int64','float64']:
            if CONFIG["numeric_fill"] == "median":
                df[col].fillna(df[col].median(), inplace=True)
            else:
                df[col].fillna(df[col].mean(), inplace=True)
        else:
            df[col].fillna(df[col].mode()[0], inplace=True)
    return df
```

3️⃣ Fix data types

```
def fix_data_types(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            try:
                df[col] = pd.to_numeric(df[col])
            except:
                pass
    return df
```

4️⃣ Clean text columns

```
def clean_text(df):
    for col in df.select_dtypes(include='object'):
        df[col] = df[col].str.lower().str.strip()
    return df
```

5️⃣ Handle outliers (IQR)

```python
def remove_outliers(df):
    if CONFIG["outlier_method"] != "iqr":
        return df

    for col in df.select_dtypes(include=['int64','float64']):
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        df = df[(df[col] >= Q1 - 1.5*IQR) &
                (df[col] <= Q3 + 1.5*IQR)]
    return df
```

6️⃣ Encode categorical features

```python
def encode_features(df):
    if CONFIG["encode"] == "onehot":
        df = pd.get_dummies(df, drop_first=True)
    else:
        for col in df.select_dtypes(include='object'):
            df[col] = df[col].astype('category').cat.codes
    return df
```

🔹 STEP 4: Build the Pipeline Function

```python
def clean_data_pipeline(df):
    df = remove_duplicates(df)
    df = df.drop(columns=CONFIG["drop_columns"], errors='ignore')
    df = handle_missing_values(df)
    df = fix_data_types(df)
    df = clean_text(df)
    df = remove_outliers(df)
    df = encode_features(df)
    df.reset_index(drop=True, inplace=True)
    return df
```

🔹 STEP 5: Run Pipeline on Raw Data

```python
raw_df = pd.read_csv("raw_data.csv")
```

```
clean_df = clean_data_pipeline(raw_df)

clean_df.to_csv("cleaned_data.csv", index=False)
```

🔥 BONUS: ML-SAFE PIPELINE (No Data Leakage)

> Interviewers LOVE this answer.

```
from sklearn.model_selection import train_test_split

X = df.drop('target', axis=1)
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

👉 **Cleaning & scaling must be fitted on TRAIN only**

🧠 Interview Explanation (Golden Answer)

> An automated data cleaning pipeline is a reusable preprocessing workflow that removes duplicates, handles missing values, fixes data types, cleans text, treats outliers, encodes categorical features, and prepares data consistently for machine learning while preventing data leakage.
