example 1

```
import pandas as pd
data={
    'Name':['alice','BOB','charlie','DAVID'],
    'City':['pune','MUMBAI','delhi','chennai'],
    'Email':
['alice@gmail.com','bob@yahoo.com','charlie@outlook.com','david@compan
y.in'],
    'Feedback':['Good!!','bad','average','EXCELLENT'],
}

df= pd.DataFrame(data)
print('Original Data:\n', df)

df['Name'] = df['Name'].str.strip().str.title()
df['City'] = df['City'].str.strip().str.title()
df['Feedback'] = df['Feedback'].str.strip().str.capitalize()
df['Email_Domain'] = df['Email'].str.split('@').str.get(1)
df['City_Code'] = df['City'].str[:3].str.upper()
print("\n Cleaned Data:\n", df)
```

```
Original Data:
       Name     City                Email   Feedback
0    alice     pune      alice@gmail.com     Good!!
1      BOB   MUMBAI        bob@yahoo.com        bad
2  charlie    delhi  charlie@outlook.com    average
3    DAVID  chennai     david@company.in  EXCELLENT

 Cleaned Data:
       Name     City                Email   Feedback Email_Domain
City_Code
0    Alice     Pune      alice@gmail.com     Good!!    gmail.com
PUN
1      Bob   Mumbai        bob@yahoo.com        Bad    yahoo.com
MUM
2  Charlie    Delhi  charlie@outlook.com    Average  outlook.com
DEL
3    David  Chennai     david@company.in  Excellent   company.in
CHE
```

example 2

```
import pandas as pd
import numpy as np

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, -5, 120, np.nan],
    'Gender': ['F', 'M', 'M', 'F']
```

```
}

df = pd.DataFrame(data)
print("Original:\n", df)

#Replace invalid ages (<0 or >100) using `where`
df['Age'] = df['Age'].where((df['Age'] >= 0) & (df['Age'] <= 100),
np.nan)

#Fill missing ages with mean
df['Age'] = df['Age'].fillna(df['Age'].mean())

#Replace short gender codes
df['Gender'] = df['Gender'].replace({'M': 'Male', 'F': 'Female'})

#Clean names
df['Name'] = df['Name'].str.strip().str.title()

print("\nCleaned Data:\n", df)
```

```
Original:
      Name    Age Gender
0    Alice   25.0      F
1      Bob   -5.0      M
2  Charlie  120.0      M
3    David    NaN      F

Cleaned Data:
      Name   Age  Gender
0    Alice  25.0  Female
1      Bob  25.0    Male
2  Charlie  25.0    Male
3    David  25.0  Female
```

example 3

Data Preaparation and Cleaning Dataset

```
import pandas as pd

# Load from CSV
df = pd.read_csv(r"C:\Users\IT2\Desktop\prct4\sales_data.csv")
print(df)
#print(df.head()) to display few lines
# Load from Excel (alternative)
# df = pd.read_excel("sales_data.xlsx")

#pd.read_csv() and pd.read_excel() are used for reading data files.
#CSV is lightweight and commonly used in data analytics pipelines.
```

```
   CustomerID   Age  Gender  PurchaseAmount     City  Rating
0         101  25.0    Male           250.0   Mumbai       4
1         102  28.0  Female             NaN     Pune       5
2         103   NaN  Female           300.0    Delhi       3
3         104  45.0    Male          5000.0   Mumbai       5
4         105  23.0    Male           450.0  Chennai       2
5         106  30.0  Female         12000.0      NaN       4
6         107  29.0    Male           400.0    Delhi       3
7         108  40.0  Female           800.0  Chennai       5
```

data cleaning and transformation

```
# Check structure and data types
print(df.info())

# Rename columns for consistency
df = df.rename(columns={'PurchaseAmount': 'Purchase_Amount'})
#print(df.columns.tolist())

#df.rename(columns={'PurchaseAmount': 'Purchase_Amount'})

# Convert data types
df['CustomerID'] = df['CustomerID'].astype('str')

# Replace inconsistent categorical values
df['City'] = df['City'].str.title()

print(df.head())

#Cleaning = making data consistent, correct, and properly formatted.
# Transformation = changing types, formatting, or standardizing
strings.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CustomerID      8 non-null      int64
 1   Age             7 non-null      float64
 2   Gender          8 non-null      object
 3   PurchaseAmount  7 non-null      float64
 4   City            7 non-null      object
 5   Rating          8 non-null      int64
dtypes: float64(2), int64(2), object(2)
memory usage: 516.0+ bytes
None
  CustomerID   Age  Gender  Purchase_Amount     City  Rating
0        101  25.0    Male           250.0   Mumbai       4
1        102  28.0  Female             NaN     Pune       5
```

```
2          103   NaN  Female             300.0    Delhi       3
3          104  45.0    Male            5000.0   Mumbai       5
4          105  23.0    Male             450.0  Chennai       2
```

Check missing values

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

# Fill numeric missing values
df['Age'] = df['Age'].fillna(df['Age'].mean())

# Use the correct column name here (check with print(df.columns))
df['Purchase_Amount'] =
df['Purchase_Amount'].fillna(df['Purchase_Amount'].median())

# Fill categorical missing values
df['City'] = df['City'].fillna(df['City'].mode()[0])

print("After filling missing values:")
print(df)

#Missing data can distort analysis.
#Strategy:
#Numeric → fill with mean or median
#Categorical → fill with mode
```

```
CustomerID         0
Age                1
Gender             0
Purchase_Amount    1
City               1
Rating             0
dtype: int64
After filling missing values:
  CustomerID        Age  Gender  Purchase_Amount     City  Rating
0        101  25.000000    Male            250.0   Mumbai       4
1        102  28.000000  Female            450.0     Pune       5
2        103  31.428571  Female            300.0    Delhi       3
3        104  45.000000    Male           5000.0   Mumbai       5
4        105  23.000000    Male            450.0  Chennai       2
5        106  30.000000  Female          12000.0  Chennai       4
6        107  29.000000    Male            400.0    Delhi       3
7        108  40.000000  Female            800.0  Chennai       5
```

Detecting and Handling Outliers

```
import numpy as np
import matplotlib.pyplot as plt

# Calculate IQR
Q1 = df['Purchase_Amount'].quantile(0.25)
```

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

# Define outlier bounds
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

# Detect outliers
outliers = df[(df['Purchase_Amount'] < lower) | (df['Purchase_Amount']
> upper)]
print("Outliers:\n", outliers)

# Handle: Option 1 - Capping
df['Purchase_Amount'] = np.where(df['Purchase_Amount'] > upper, upper,
                          np.where(df['Purchase_Amount'] < lower,
lower, df['Purchase_Amount']))

#Outliers are extreme values that can skew analysis.
#IQR method is robust for detection.
#Handling = remove, cap, or transform.
```

```
Outliers:
   CustomerID   Age  Gender  Purchase_Amount     City  Rating
3         104  45.0    Male           5000.0   Mumbai       5
5         106  30.0  Female          12000.0  Chennai       4
```

Data Summarization (Statistics & Feature Transformation)

```
# Descriptive statistics
print(df.describe())

# Group by and summarize
city_summary = df.groupby('City')['Purchase_Amount'].agg(['mean',
'max', 'min', 'count'])
print(city_summary)

# Feature transformation: create new column
df['Log_Purchase'] = np.log1p(df['Purchase_Amount'])

#Summarization → helps understand spread, central tendency, and
variation.

#Transformation (like log) → reduces skewness and stabilizes variance.
```

```
             Age  Purchase_Amount    Rating
count   8.000000         8.000000  8.000000
mean   31.428571      1346.875000  3.875000
std     7.461466      1684.073903  1.125992
min    23.000000       250.000000  2.000000
25%    27.250000       375.000000  3.000000
50%    29.500000       450.000000  4.000000
```

```
75%    33.571429      1615.625000  5.000000
max    45.000000      4062.500000  5.000000
                mean     max    min  count
City
Chennai  1770.833333  4062.5  450.0      3
Delhi     350.000000   400.0  300.0      2
Mumbai   2156.250000  4062.5  250.0      2
Pune      450.000000   450.0  450.0      1
```

Example 4 Data Preaparation and Cleaning Dataset

```
import pandas as pd
import numpy as np

# Sample raw data with common issues
data = {
    'Customer_ID': [101, 102, 103, 104, 105, 106, np.nan],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', np.nan, 'Frank',
'Grace'],
    'Age': [25, np.nan, 35, 29, 42, 150, 28],
    'City': ['Pune', 'Mumbai', np.nan, 'Pune', 'Delhi', 'Delhi',
'Pune'],
    'Purchase_Amount': [2500, 3000, -500, 7000, 12000, 5000, np.nan]
}

df = pd.DataFrame(data)
print("Original Dataset:\n", df)
```

```
Original Dataset:
    Customer_ID     Name    Age    City  Purchase_Amount
0         101.0    Alice   25.0    Pune           2500.0
1         102.0      Bob    NaN  Mumbai           3000.0
2         103.0  Charlie   35.0     NaN           -500.0
3         104.0    David   29.0    Pune           7000.0
4         105.0      NaN   42.0   Delhi          12000.0
5         106.0    Frank  150.0   Delhi           5000.0
6           NaN    Grace   28.0    Pune              NaN
```

Check for Missing Values

```
print("\nMissing Values:\n", df.isnull().sum())
```

```
Missing Values:
 Customer_ID        1
Name                1
Age                 1
City                1
Purchase_Amount     1
dtype: int64
```

Handle Missing Values

```
#(a) Fill numeric columns (Age, Purchase_Amount) with mean or median
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Purchase_Amount'] =
df['Purchase_Amount'].fillna(df['Purchase_Amount'].mean())

#(b) Fill categorical columns (City, Name) with mode (most common
value)
df['City'] = df['City'].fillna(df['City'].mode()[0])
df['Name'] = df['Name'].fillna('Unknown')

#(c) Remove rows with missing Customer_ID (important for
identification)
df = df.dropna(subset=['Customer_ID'])
df
```

	Customer_ID	Name	Age	City	Purchase_Amount
0	101.0	Alice	25.0	Pune	2500.0
1	102.0	Bob	32.0	Mumbai	3000.0
2	103.0	Charlie	35.0	Pune	-500.0
3	104.0	David	29.0	Pune	7000.0
4	105.0	Unknown	42.0	Delhi	12000.0
5	106.0	Frank	150.0	Delhi	5000.0

Detect and Handle Outliers (IQR Method)

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

lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

outliers = df[(df['Purchase_Amount'] < lower) | (df['Purchase_Amount']
> upper)]
print("Outliers :", outliers)
# Cap the outliers (replace extreme values with bounds)
df['Purchase_Amount'] = np.where(df['Purchase_Amount'] < lower, lower,
                          np.where(df['Purchase_Amount'] > upper,
upper, df['Purchase_Amount']))

# You might have expected: -500 (negative value) and 12000 (very high
value) to be detected as a outlier.
#But statistically, both are still within the calculated IQR bounds.
Range is (-4250 to 13750).
#That's because IQR depends on your dataset's internal spread – and
with only a few data points (7 values), it's not very sensitive.

#Negative purchases don't make sense, so you can flag them directly:
#df[df['Purchase_Amount'] < 0]
```

```
# or Use a smaller IQR multiplier (more sensitive)
#lower = Q1 - 1.0 * IQR
#upper = Q3 + 1.0 * IQR

Outliers : Empty DataFrame
Columns: [Customer_ID, Name, Age, City, Purchase_Amount]
Index: []
```

Correct Invalid Data

```
# Example — Age 150 is unrealistic for a customer.
df.loc[df['Age'] > 90, 'Age'] = df['Age'].median()

#If you want to treat negative purchases as invalid (e.g., logically,
purchase amounts can't be negative),
#you can add a business rule after statistical cleaning:
df.loc[df['Purchase_Amount'] < 0, 'Purchase_Amount'] = 0

df
```

	Customer_ID	Name	Age	City	Purchase_Amount
0	101.0	Alice	25.0	Pune	2500.0
1	102.0	Bob	32.0	Mumbai	3000.0
2	103.0	Charlie	35.0	Pune	0.0
3	104.0	David	29.0	Pune	7000.0
4	105.0	Unknown	42.0	Delhi	12000.0
5	106.0	Frank	33.5	Delhi	5000.0

Standardize Text Data

```
#Make sure categorical values have consistent formatting.
df['City'] = df['City'].str.title().str.strip()   # "pune" → "Pune"
```

Final Cleaned Data

```
print("\nCleaned Dataset:\n", df)


Cleaned Dataset:
```

	Customer_ID	Name	Age	City	Purchase_Amount
0	101.0	Alice	25.0	Pune	2500.0
1	102.0	Bob	32.0	Mumbai	3000.0
2	103.0	Charlie	35.0	Pune	0.0
3	104.0	David	29.0	Pune	7000.0
4	105.0	Unknown	42.0	Delhi	12000.0
5	106.0	Frank	33.5	Delhi	5000.0

example 5 Data Preaparation and Cleaning Dataset

create a sample dataset

```
import pandas as pd
import numpy as np

data = {
    'Customer_ID': [1, 2, 3, 4, 5],
    'Name': [' Alice ', 'bob', 'CHARLIE', 'david@123', 'Eve!'],
    'Gender': ['M', 'Male', 'F', 'female', np.nan],
    'City': ['pune', 'Pune ', 'DELHI', 'mumbai', 'Mumbai '],
    'Feedback': ['Good!!', 'bad', 'AVERAGE', 'excellent ', ' good']
}

df = pd.DataFrame(data)
print("Original Dataset:\n", df)
```

```
Original Dataset:
    Customer_ID       Name  Gender     City    Feedback
0            1     Alice         M     pune      Good!!
1            2        bob    Male    Pune          bad
2            3    CHARLIE       F    DELHI     AVERAGE
3            4  david@123  female   mumbai  excellent
4            5       Eve!     NaN  Mumbai         good
```

Clean Text Columns

```
#We'll remove spaces, symbols, and make case consistent.
# Remove leading/trailing spaces and convert to title case
df['Name'] = df['Name'].str.strip().str.title()

# Remove special characters or numbers using regex
df['Name'] = df['Name'].str.replace('[^A-Za-z ]', '', regex=True)

# Clean City column: make all consistent
df['City'] = df['City'].str.strip().str.title()

print("\nAfter cleaning text columns:\n", df)
```

```
After cleaning text columns:
    Customer_ID     Name  Gender    City    Feedback
0            1    Alice       M    Pune      Good!!
1            2      Bob    Male    Pune         bad
2            3  Charlie       F   Delhi     AVERAGE
3            4    David  female  Mumbai  excellent
4            5      Eve     NaN  Mumbai        good
```

Standardize Categorical Values

```
#For example, gender has mixed forms – "M", "Male", "F", "female", and
missing.
df['Gender'] = df['Gender'].str.strip().str.lower()   # lowercase
df['Gender'] = df['Gender'].replace({
    'm': 'Male',
    'male': 'Male',
    'f': 'Female',
    'female': 'Female'
})

# Fill missing gender with the most common value
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])

print("\nAfter standardizing Gender:\n", df)
```

```
After standardizing Gender:
    Customer_ID     Name  Gender    City    Feedback
0            1    Alice    Male    Pune      Good!!
1            2      Bob    Male    Pune         bad
2            3  Charlie  Female   Delhi     AVERAGE
3            4    David  Female  Mumbai  excellent
4            5      Eve  Female  Mumbai        good
```

Encode Categorical Columns (for Machine Learning) (feature transformation)

```
#ML models (regression, decision trees) prefer to work with numbers,
not strings – so we'll convert categories to numeric codes.
# Label Encoding
df['Gender_Code'] = df['Gender'].map({'Male': 0, 'Female': 1})

# One-Hot Encoding for City (creates dummy columns)
#df = pd.get_dummies(df, columns=['City'], prefix='City')
(City_Delhi etc columns showing True and False )
df = pd.get_dummies(df, columns=['City'], prefix='City', dtype=int)
#(City_Delhi etc columns showing 1 and 0 )
print("\nAfter Encoding:\n", df)
```

```
After Encoding:
    Customer_ID     Name  Gender    Feedback  Gender_Code  City_Delhi
\
0            1    Alice    Male      Good!!            0           0

1            2      Bob    Male         bad            0           0

2            3  Charlie  Female     AVERAGE            1           1

3            4    David  Female  excellent            1           0
```

```
4            5      Eve  Female        good             1           0


   City_Mumbai  City_Pune
0            0          1
1            0          1
2            0          0
3            1          0
4            1          0

df.columns

Index(['Customer_ID', 'Name', 'Gender', 'Feedback', 'Gender_Code',
       'City_Delhi', 'City_Mumbai', 'City_Pune'],
      dtype='object')
```

Sentiment Mapping for “Feedback”

```
#If you want to use feedback in analytics (e.g., sentiment scoring):
sentiment_map = {'good': 1, 'excellent': 2, 'average': 0, 'bad': -1}
df['Feedback_Clean'] =
df['Feedback'].str.lower().str.strip().str.replace('[^a-z ]', '',
regex=True)
df['Sentiment_Score'] = df['Feedback_Clean'].map(sentiment_map)
print("\nAfter Feedback cleaning and mapping:\n", df[['Feedback',
'Feedback_Clean', 'Sentiment_Score']])


After Feedback cleaning and mapping:
      Feedback Feedback_Clean  Sentiment_Score
0      Good!!           good                1
1         bad            bad               -1
2     AVERAGE        average                0
3   excellent      excellent                2
4        good           good                1
```